Brigham Young University
BYU ScholarsArchive

All Theses and Dissertations

2016-04-01

Stiffness Reduction Strategies for Additively Manufactured Compliant Mechanisms

Ezekiel G. Merriam
Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Mechanical Engineering Commons

BYU ScholarsArchive Citation

Merriam, Ezekiel G., "Stiffness Reduction Strategies for Additively Manufactured Compliant Mechanisms" (2016). *All Theses and Dissertations*. 5873.
https://scholarsarchive.byu.edu/etd/5873

This Dissertation is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

Stiffness Reduction Strategies for Additively Manufactured Compliant Mechanisms

Ezekiel G. Merriam

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Larry L. Howell, Chair
Spencer P. Magleby
Paul W. Richards
Eric R. Homer
Mark B. Colton

Department of Mechanical Engineering

Brigham Young University

April 2016

Copyright © 2016 Ezekiel G. Merriam

All Rights Reserved

ABSTRACT

Stiffness Reduction Strategies for Additively
Manufactured Compliant Mechanisms

Ezekiel G. Merriam
Department of Mechanical Engineering, BYU
Doctor of Philosophy

This work develops and examines design strategies for reducing the stiffness of 3D-printed compliant mechanisms. The three aspects of a flexure that determine its stiffness are well known: material, boundary conditions, and geometry. In a highly constrained design space however, flexure stiffness may remain unacceptably high even while arriving at the limits of design constraints. In this work, changes to geometry and boundary conditions are examined that lead to drastically reduced stiffness behavior without changing flexure thickness, width, or length. Changes to geometry can result in very complex mechanisms. However, 3D-printing enables almost arbitrarily complex geometries. This dissertation presents three design strategies for stiffness reduction: static balancing, lattice flexures, and compound joints.

Static balancing refers to changes in the boundary conditions that result in a near-zero net change in potential energy storage over the useful deflection of a flexure. In this work, I present a method for static balancing that utilizes non-dimensional parameters to quickly synthesize a joint design with stiffness reduced by nearly 90%. This method is not only simple and straightforward, it is applicable to a wide range of flexure topologies. The only requirements on the joint to be balanced are that it must be approximated as a pin joint and torsion spring, and it must have a well-understood stiffness when subjected to a compressive load.

Lattice flexures result from modifications to geometry that reduce cross-sectional area without changing width. However, the reduction in stiffness is *greater than* the reduction in cross sectional area. This can occur because the bending load is now carried by beams partially in torsion. Two lattice geometries are proposed and analyzed in detail using analytic and numeric techniques. It is shown that the off-axis stiffness behavior of lattice flexures can be better than that of conventional blade flexures while bending stiffness is reduced $>60\%$.

Compound joints are those that consist of arrays of flexures arranged co-axially. This arrangement provides increased range of motion, generally decreased stiffness, and improved stability. Additionally, a method is herein presented to reduce the parasitic center shift of a compound joint to nearly zero at a specified deflection.

The penultimate chapter demonstrates how all three strategies can be used together, and includes new results to facilitate their combination.

Keywords: compliant mechanisms, static balancing, lattice flexures, compound joints, load-dependent stiffness, 3D printing

ACKNOWLEDGMENTS

I could not have completed this dissertation without the help of many people. Larry Howell has been more than an advisor to me over the years; he has truly been a mentor and a source of wisdom in many of the large and small things that make a man worth admiring. Kevin Cole, Ken Forster, Nick Hawkins, Miriam Busch, Judy Stoudt, and the other members of the department staff have been wonderful support in fabricating prototypes, building DAQs, and handling the paperwork required to complete my studies.

My peers have also been an invaluable source of support. In particular, Jason Lund, Shannon Zirbel, Jared Bruton, and Todd Nelson have helped solve many issues, from de-bugging code to securing funding to technical discussion and simple comedy relief. Abraham Lee's particle swarm optimiztion algorithm proved invaluable in many of my design efforts. Patrick Walton's help constructing several of my prototypes is gratefully acknowledged.

This work would have been impossible without the financial support of NASA's Office of the Chief Technologist; its support is gratefully acknowledged, along with the mentorship of Jonathan Jones and Douglas Hofmann. Additionally, Jonathan Jones and Kenneth Cooper of NASA Marshall Space Flight Center are profusely thanked for the fabrication of the titanium prototypes. Additional funding was provided by NASA Grant No. NNX13AF52G.

Finally, my wife Kayla deserves the highest praise for her unflinching and unfailing support. She has been my wellspring of encouragement and motivation. It is to her that this dissertation is dedicated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Variables used in Chapter 1

E	Potential energy
l	Distance from center of rotation to the attachment point of the spring
λ	Included angle of the spring attachment points
x_0	Initial length of the springs
k	Dimensional stiffness—can be linear or rotational
T	Dimensional torque
θ	Angular deflection

Variables used in Chapter 2

S_y	Yield strength
S_{ut}	Ultimate strength
S_e	Endurance limit

Variables used in Chapter 3

k_θ	Torsional stiffness of a joint, either load-independent stiffness or corrected load-dependent stiffness
k_l	Stiffness of balancing spring
k	Stiffness of balanced system
d	Distance from pivot center to balancing spring attachment points
x_0	Free length of balancing spring
P	Preload applied to balancing spring
θ	Angle of deflection of the load-independent or load-dependent joint
$T = k\theta$	Torque required to deflect hinge through angle θ
$\Pi_1 = \frac{k_\theta}{Pd}$	Π group governing torsional stiffness
$\Pi_2 = \frac{k_l d}{P}$	Π group governing stiffness of balancing spring
E	Young's modulus of the flexure material
b	Width of CAFP flexure strip
t	Thickness of CAFP flexure strip
$I = \frac{bt^3}{12}$	Moment of inertia of CAFP flexure strip
L	Length of CAFP flexure strips
k'_θ	Uncorrected torsional stiffness of load-dependent joint (no applied loads)
V	Vertical load applied to hinge
H	Horizontal load applied to hinge
α	Half the intersection angle of the CAFP flexures
$v = \frac{VL^2 \sec(\alpha)}{EI}$	Non-dimensional applied vertical load
$h = \frac{HL^2 \csc(\alpha)}{EI}$	Non-dimensional applied horizontal load
β_i	Dimensionless parameter describing the forces in CAFP flexures
ϕ_i	Dimensionless parameter describing the stiffness of the individual CAFP flexures

Variables used in Chapter 4

T	Torque
$\Delta\theta$	Change in angular position
$\eta = \frac{HL^2}{EI}$	Non-dimensional applied horizontal load
$\nu = \frac{VL^2}{EI}$	Non-dimensional applied vertical load
H	Horizontal load applied to hinge
V	Vertical load applied to hinge
L	Characteristic flexure length
E	Young's modulus of the flexure material
I	Moment of inertia of CAFP flexure strip
κ	Non-dimensional rotational stiffness
K	Rotational stiffness
ϕ_i	Dimensionless parameter describing the stiffness of the individual CAFP flexures
β_i	Dimensionless parameter describing the forces in CAFP flexures
ρ	Dimensionless parameter that describes the location where the strips of a CAFP cross

Variables used in Chapter 5

k_b	Bending stiffness of a flexure subject to an end moment
E	Young's modulus of the flexure material
ν	Poisson's ratio of the flexure material
$I = \frac{bh^3}{12}$	Moment of inertia of CAFP flexure strip
h	Thickness of conventional flexure or lattice flexure
w	Overall width of blade flexure
b	Distance between rail centers of lattice flexure
α	Lattice angle
L_1	Lattice element half-length
η	Aspect ratio of lattice flexure
θ	Rotational deflection of lattice element
M_0	Total moment applied to lattice element
M_1	Moment applied to lattice rail or rails
M_2	Moment applied to diagonal lattice element
L_2	Length of diagonal lattice element
I_r	Second moment of area of the rail element in a lattice flexure
I_l	Second moment of area of the diagonal element in a lattice flexure
θ_t	Rotational deflection of lattice element due to torsion
θ_b	Rotational deflection of lattice element due to bending
K	Section property analogous to the second moment of area, but for torsional loading of the diagonal lattice element
G	Modulus of rigidity of the diagonal lattice element
k_X	Stiffness of a single lattice element of the X-type
k_V	Stiffness of a single lattice element of the V-type
k_t	Torsional stiffness of a flexure subject to an end torque

Variables used in Chapter 6

κ_x	Non-dimensional linear stiffness along x-axis
κ_y	Non-dimensional linear stiffness along y-axis
κ_z	Non-dimensional linear stiffness along z-axis
$\kappa_{\theta x}$	Non-dimensional rotational stiffness about x-axis
$\kappa_{\theta y}$	Non-dimensional rotational stiffness about y-axis
$\kappa_{\theta z}$	Non-dimensional rotational stiffness about z-axis
n	Number of joints in series, or on one side of a series-and-parallel joint
F_i	Applied force along the i (x, y, or z) axis
M_i	Applied moment about the i (x, y, or z) axis
L	Characteristic flexure length
δ_i	Deflection along the i (x, y, or z) axis
θ_i	Deflection about the i (x, y, or z) axis
EI	Flexural stiffness of flexure element
h	Flexure thickness
b	Flexure width

Variables used in Chapter 7

$\eta = \frac{HL^2}{EI}$	Non-dimensional horizontal load
$\nu = \frac{VL^2}{EI}$	Non-dimensional vertical load
$\kappa = \frac{KL}{EI}$	Non-dimensional rotational stiffness
L_1/b	Lattice length parameter
$h/h+b$	Lattice aspect ratio parameter
EI_{eff}	Effective flexural stiffness for a lattice flexure
E	Young's modulus
I_r	Second moment of area of the lattice rail
I_l	Second moment of area of the lattice diagonal element
$\frac{L_1}{b}$	Lattice length ratio
K	Torsional stiffness term for the diagonal element
β_i	Coefficients for the non-linear regression describing κ
$\Pi_1 = \frac{k_\theta}{Pd}$	Π group involving rotational stiffness
$\Pi_2 = \frac{k_l d}{P}$	Π group involving stiffness of linear spring
k_θ	Rotational load-dependent stiffness of joint to be balanced
k_l	Stiffness of linear spring used as balancer
d	Distance from center of rotation to anchor point of balancer
P	Preload on balancer
n	Number of flexures on one side of a series-and-parallel compound joint
L	Length of flexure
h	Thickness of flexure
L_i	Length of first and second legs of balancer spring
b_i	Width of first and second legs of balancer spring
h_i	Thickness of first and second legs of balancer spring
I_i	Second moment of area of first and second legs of balancer spring

Variables used in Chapter 7 (cont.)

θ_{10}	Angular offset of parallel guiding section of balancer spring
θ_{20}	Angular offset of driver dyad section of balancer spring
K_i	Stiffness of torsion springs representing first and second legs of balancer spring
$K_{i_{\text{eq}}}$	Equivalent stiffness of torsion springs representing first and second legs of balancer spring
P_{desired}	Target preload for the balancer spring
$k_{l\ \text{desired}}$	Target stiffness of the spring when preloaded with force P_{desired}
σ_{max}	Maximum bending stress in psuedo-rigid-body model

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

While compliant mechanisms have been shown to hold great promise in improving the performance of a variety of mechanisms [1], realizing these advantages can be difficult in some applications because of disadvantages inherent to compliant mechanisms. Some of the drawbacks of compliant mechanisms include the lack of sufficient flexibility (as in stress-limited design) and excessive actuation effort (due to energy storage in the deflected mechanism) [2]. In some cases the production of advanced or specialized compliant mechanisms is difficult because of limitations in production technology [3]. Although solving every design challenge is beyond the scope of this work, improving the desirable flexibility of compliant mechanisms will allow their application in a greater variety of products and mechanisms.

The rising technology of additive manufacturing holds great promise as a method for producing compliant mechanisms [4, 5]. However, materials produced by additive manufacturing are often somewhat inferior to traditional materials in terms of strength and fatigue properties [6, 7]. In general, this means that compliant mechanisms produced with additive manufacturing have a reduced range of motion. Most strategies for increasing flexibility rely on increased complexity [8], which usually precludes their adoption. However, additive manufacturing is relatively insensitive to design complexity, thus allowing complex geometries to be created [9, 10]. These complex geometries allow increased range of motion and decreased actuation effort, making compliant mechanisms a viable choice for many applications in space and elsewhere.

Therefore, this work seeks to improve flexibility of compliant mechanisms by finding novel strategies for reducing the actuation effort and improving range of motion. This will enable these mechanisms to take advantage of recent advances in additive manufacturing technologies. Thus, both compliant mechanisms and additive manufacturing will become more useful from taking advantage of each others' strengths.

Figure 1.1: A folded beam suspension. This device is often used where linear motion is needed. Distributing the deflection between two serial stages allows a greater displacement than would be possible otherwise.

1.2 Background

A compliant mechanism derives its motion from the deflection of its constituent members. Compliant mechanisms offer decreased part count, decreased complexity, lower weight, longer life, and lower cost. Since compliant mechanisms can be designed with no surface contact, wear and all its associated issues are eliminated. In many cases, bearings may be eliminated, along with their weight, complexity, and failure modes. [1] Preliminary work has shown the applicability of compliant mechanism technology to space applications [11]. Additionally, compliant mechanisms lend themselves to monolithic construction [12].

1.2.1 Improving Range of Motion

The range of motion of a compliant mechanism of given geometry is generally limited by stress. Depending on the application of the mechanism, it can be designed for finite or infinite fatigue life, but in both cases some threshold stress is determined, and the mechanism is designed such that this threshold stress is not exceeded [10]. Howell explains that the three factors that determine a mechanism's flexibility (and thus, its range of motion) are: material (its elastic properties and strength), boundary conditions (how it is attached to whatever needs movement), and geometry (thickness and length for rectangular cross sections) [1, 2].

(a) Unbalanced system where a non-zero value of F is required to maintain the position.

(b) A balanced system where the position can be maintained with $F = 0$.

Figure 1.2: An illustration of static balancing. Figure 1.2a shows an unbalanced system where a non-zero value of F is required to maintain the position. Figure 1.2b shows a balanced system where the position can be maintained with $F = 0$.

Although this seems to be a straightforward problem, non-obvious solutions can be found. For example, the range of motion of a linear displacement mechanism can be increased by using a folded-beam suspension [2]. An example of a folded-beam suspension is shown in Figure 1.1. Effectively, this configuration increases the flexure length without increasing the mechanism footprint. Other examples of similar strategies include the ortho-planar spring [13] and the Flex-16 [14]. Increasing effective length in this way is one of the most effective ways to increase a mechanism's range of motion, but can result in unwanted compromises in off-axis stiffness [15]. Other strategies for increasing range of motion could include manipulating boundary conditions, as well as exploring methods for increasing effective length without sacrificing off-axis stiffness. Finally, because the envelope of a mechanism can be an important design consideration, it is desirable to enable large motion in a small envelope.

While increasing the strength of the material or using a material of lower Young's Modulus can also increase the range of motion for a particular compliant mechanism, materials research to modify the material properties is not part of this work.

1.2.2 Reducing Actuation Effort

The strain energy associated with the actuated mechanism means that the force or torque required to maintain a mechanism's actuated position is often non-zero [16]. For many applications this results in increased actuator size compared to traditional mechanisms, which leads to higher mass and cost. Reducing actuation effort of compliant mechanisms could make them acceptable for applications that require compact or low-mass actuators, or where mechanism transparency is desired [8, 17]. Here again, the stiffness of a mechanism is governed by material properties, geometry, and boundary conditions [1, 2]. One simple way to reduce stiffness would be to reduce the width. However, this would result in reduced off-axis stiffness which could make the design unacceptable for its original application. Therefore, strategies must be found for reducing effective width without losing off-axis stiffness.

Another approach to reduced actuation effort requires the addition of auxiliary energy storage elements, and is known as static balancing [18, 19]. Static balancing offers the potential of exploiting the benefits of compliant mechanisms but mitigating the detrimental effects of strain energy.

Consider the gravity balanced mechanism in Figure 1.2. For the unbalanced system, a non-zero input force is needed to maintain position, except when $\theta = \pm 90^{\circ}$; the system's potential energy is a function of θ. In the balanced system, this need not be the case. It can be shown that with a zero-free-length spring and the proper selection of the spring constant, k, $F = 0$ for all θ; system energy is constant [20–22]. Thus, the introduction of a pre-loaded spring has decoupled mechanism position from mechanism potential energy, reducing the input force.

Herder described the concept of using springs to compensate for undesired changes in strain energy [18]. Much work has been done in reducing input force in surgical instruments and prosthetics [17, 23]. Leishman et al. applied static balancing to a haptic interface with good success [24]. These designs incorporate a pre-load and some finite potential energy. Stored energy is released from the pre-loading members as the device is actuated. This energy release aids in the actuation of the mechanism. Because the net change in energy is small, the input force is reduced, and in some cases can be nearly eliminated [25, 26]. However, statically balanced mechanisms are often complex and bulky, which constitutes an obstacle to their widespread adoption.

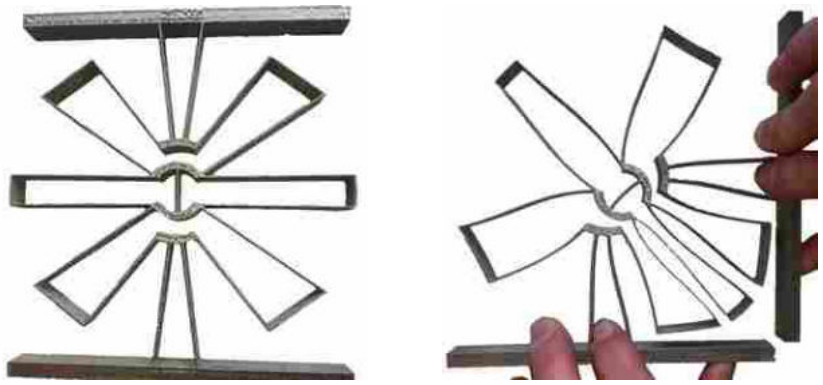


Figure 1.3: A compliant titanium hinge produced with EBM. This hinge is capable of $\pm 90^{\circ}$ of motion. Images provided courtesy of Robert Fowler.

1.2.3 Additive Manufacturing

Advances in Electron Beam Melting (EBM) enable additive manufacturing (also referred to as rapid manufacturing) in a variety of metals, including alloys of titanium. The EBM process is well documented [27–31]. Case studies have shown that rapid manufacturing offers reduced costs when production volumes are low, many design iterations are to be explored, high geometric complexity is needed, or when new materials are to be explored [32, 33]. Additionally, material scrap rate can be significantly reduced by printing a near-net-shape part rather than machining it from solid billet [34]. Combining compliant mechanisms with rapid manufacturing techniques opens up interesting possibilities for creating compliant space mechanisms that have unprecedented performance. Figure 1.3 shows a compliant hinge built by EBM in titanium. It features large deflection capabilities and linear torque-displacement behavior [14].

Rapid manufacturing processes have been used in multiple aerospace applications, including ductwork [32, 33] and a capacitor housing on the International Space Station [35]. These applications used selective laser sintered nylon parts, which established a basis for rapid manufacturing as a viable method for producing parts. Structural brackets [27], a shrouded cryogenic impeller [27,36], and brackets for the Juno spacecraft [34] have also been manufactured in titanium using rapid manufacturing processes. While most parts built thus far have been structural members (brackets, etc.) or non-structural assemblies (ductwork and housings), in our work we use additive manufacturing to create monolithic mechanisms for aerospace applications.

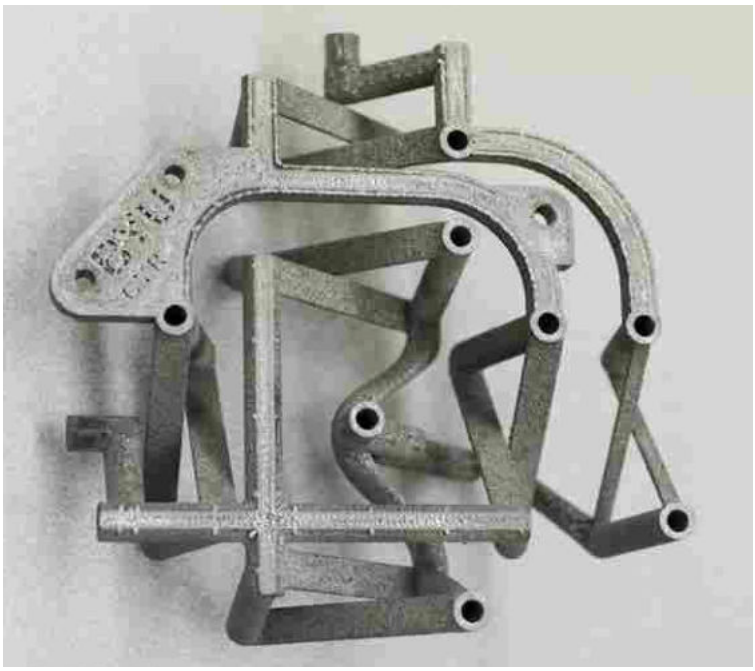


Figure 1.4: A two-degree-of-freedom pointing mechanism produced with EBM in titanium.

Figure 1.4 shows another mechanism developed as a proof-of-concept to demonstrate the potential of EBM-produced parts. This mechanism is a two-degree-of-freedom pointing mechanism with multiple axes of rotation, meant to replace complex, failure-prone assemblies for orienting instruments on spacecraft while enabling simplification of design and reduction of mass [10]. Other EBM-built parts are shown in Figure 1.5.

1.2.4 Combining Additive Manufacturing with Compliant Mechanisms

Thus far, it has been shown that the EBM process holds great promise as a method for producing highly complex geometries. Additionally, we have shown that compliant mechanism development is currently constrained by mechanism complexity. Because compliant mechanisms are often monolithic, they are well suited to adaption for additive manufacturing processes. Exploiting additive manufacturing together with compliant mechanisms could allow the design of highly complex compliant mechanisms. However, design of compliant mechanisms for additive manufacturing production is limited by fatigue life. Rafi et al. have published fatigue strengths for EBM produced titanium and shown that the fatigue strength is far below that of wrought titanium [6].

Figure 1.5: Some compliant mechanisms manufactured with electron beam melting. Clockwise from top left: linear suspension element, cryogenic valve concept, split-tube flexure, one-way torsion spring.

For this reason, it is difficult to design EBM-produced mechanisms for large displacements and low actuation efforts while maintaining reasonable stability and fatigue life.

1.3 Research Objectives

The purpose of this research is to determine general methods for improving flexibility of compliant mechanisms. This work builds on previous work to reduce actuation effort and increasing the range of motion of compliant mechanisms. This is accomplished through exploiting changes to boundary conditions and mechanism geometry. The use of additive manufacturing allows us to relax constraints on design complexity in favor of reduced actuation torque and increased range of motion. The following subsections outline the technical approach to reducing actuation effort and improving range of motion. While methods previously existed to accomplish my research objectives, they generally result in undesirable design trade-offs. In this work, we seek to circumvent these negative trade-offs using novel flexibility strategies.

1.3.1 Reducing Actuation Effort

Two strategies for stiffness reduction are investigated: static balancing and lattice flexures.

Static Balancing

Static balancing is one way to achieve reduced joint stiffness. It is usually accomplished by adding auxiliary springs to a compliant joint [3]. The results of static balancing are impressive, but a general method for the static balancing of rotational joints is still needed [25]. Mersch et al. showed one configuration of joint that added a negative stiffness mechanism to a hinge and optimized the assembly for constant potential energy [3].

Chapter 3 presents a general method for static balancing of rotational joints. Like previous methods, this uses an auxiliary spring to change how energy is stored in the joint. However, this method uses non-dimensional parameters that allow the rapid design of a balancer using only four physical parameters. It results in excellent stiffness reduction over a large range of motion. Stiffness reduction of 87% has been measured over $\pm 40^\circ$ of motion. One additional consideration to this method is the accounting for the change in joint stiffness due to the the loads introduced by the balancer spring. A method for determining this load-dependent stiffness behavior is provided in Chapter 4.

Lattice Flexures

Changes to flexure dimensions (eg. reducing thickness or width) will certainly reduce actuation effort. However, changing the joint in this way may be undesirable for other reasons (eg. the limits of the manufacturing process or reduced off-axis stiffness). One possibility for useful gains with minimal trade-offs is the reduction of effective width by using what we term here a "lattice flexure." A cross-axis flexural pivot using this principle is shown in Figure 1.6. The off-axis stiffness and other performance characteristics of this joint have not been fully evaluated, but its stiffness has been markedly reduced from the traditional flexure geometry. Finding an optimal geometry for such lattice geometry could greatly reduce actuation effort for many different joint topologies without seriously affecting off-axis stiffness.

Figure 1.6: A cross-axis flexural pivot designed and built with lattice flexures to reduce the effective width, and thus the stiffness of the joint. It should be noted that although the actuation effort will be reduced with such designs, changes to the effective width of a flexure do not increase its range of motion.

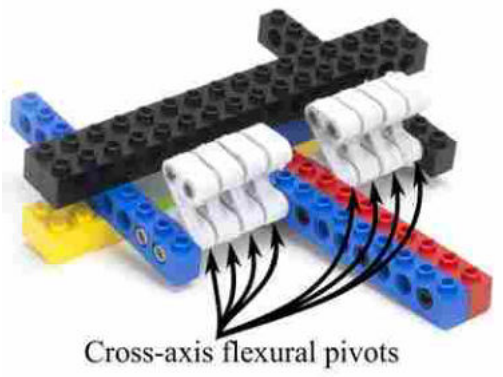


Figure 1.7: A LEGO® and FlexLinks [37] prototype that demonstrates cross-axis flexural pivots arranged in series and series-and-parallel. The blue and red bar rotates to a maximum displacement four times greater than it would be if only a single cross-axis flexural pivot were used.

In Chapter 5 we present analytic stiffness calculations for two lattice flexure geometries, and surface fits of finite element simulations that quantify off-axis stiffness behavior.

Finally, a statically balanced lattice flexure incorporates both of these design strategies. Chapter 7 details how to combine the advantages of lattice flexures with static balancing.

1.3.2 Improving Range of Motion

Joint range of motion and stiffness can be manipulated by employing compound joints, eg. arranging joints in series or parallel. For example, two hinges arranged in series that individually

have a $\pm 15^\circ$ range of motion will have a total range of motion of $\pm 30^\circ$. When the stiffness of such a compound joint is considered, recall that the effective stiffness of two springs in series is given by $k_{eq} = \frac{k_1 k_2}{k_1 + k_2}$. If $k_1 = k_2 = k$, $k_{eq} = k/2$. Thus, the range of motion is increased, but at the new maximum deflection the actuation torque will be the same (as $T = k\theta = \frac{k}{2}(2\theta)$). However, adding a second hinge in series has changed the off-axis stiffness of the joint. Adding symmetry restores off-axis stiffness, but it increases actuation torque unless other design changes are implemented. One example of a possible joint using these principles is shown in Figure 1.7.

Quantifying the design trade-offs of various arrangements of compound joints is the subject of Chapter 6. Other results include a method for dramatically reducing a joint's center shift.

1.4 Research Methods

The development of flexibility improvement strategies proceeded along the following general steps:

1. Survey of literature for general joint topologies
2. Development of analytic/pseudo-rigid-body model of proposed method
3. Development of finite element model incorporating method
4. Ensure agreement between analytic/PRB and FE models
5. Produce prototypes to validate methods

1.5 Relevance to Additive Manufacturing

Because the strategies for increasing range of motion and reducing actuation effort have the potential to greatly increase joint complexity [24], it is anticipated that non-traditional manufacturing techniques will be required for prototyping these new hinge configurations. For example, some of the obstacles to the adoption of the joint presented by Morsch et al. are its relative bulkiness and complexity [3]. Additive manufacturing can produce parts of great complexity, provided a few

guidelines are followed during the design process [10, 12]. As EBM is a relatively young technology, its potential has not been fully explored. Developing new design methods for compliant mechanisms will allow full exploitation of the capabilities of additive manufacturing.

1.6 Collaboration

The majority of this work was performed on BYU campus, but will include on-site experiences at NASA marshall Space Flight Center and the Jet Propulsion Laboratory. Because of our partnership with NASA on this project, the rapid manufacturing facilities at NASA Marshall Space Flight Center have been used for prototyping test mechanisms in titanium. Our lab at BYU is equipped with a MakerBot® Replicator™ 2 for in-house rapid prototyping work.

Much of this work has been done in collaboration with Jonathan Jones of NASA Marshall Space Flight Center. Collaboration with Dr. Jones has provided context and motivation for the developments of this dissertation. The on-site research provided exposure to the space and aerospace community, with many opportunities for peer review of work. All this has helped ensure that my research is relevant and aids in rapid dissemination of technical developments.

CHAPTER 2. DESIGN OF 3D PRINTED TITANIUM COMPLIANT MECHANISMS

2.1 Abstract

[1] This chapter describes 3D-printed titanium compliant mechanisms for aerospace applications. It is meant as a primer to help engineers design compliant, multi-axis, printed parts that exhibit high performance. Topics covered include brief introductions to both compliant mechanism design and 3D printing in titanium, material and geometry considerations for 3D printing, modeling techniques, and case studies of both successful and unsuccessful part geometries. Key findings include recommended flexure geometries, minimum thicknesses, and general design guidelines for compliant printed parts that may not be obvious to the first time designer.

2.2 Introduction

A compliant mechanism derives its motion from the deflection of its constituent members. Compliant mechanisms offer decreased part count, decreased complexity, lower weight, longer life, and lower cost. Since compliant mechanisms can be designed with no surface contact, wear and all its associated issues are eliminated. In many cases, bearings may be eliminated, along with their weight, complexity, and failure modes [1]. Preliminary work has shown the applicability of compliant mechanism technology to space applications [11]. Additionally, compliant mechanisms lend themselves to monolithic construction through additive manufacturing processes. Advances in Electron Beam Melting (EBM) enable additive manufacturing (also referred to as rapid manufacturing) in a variety of metals, including alloys of Titanium. The EBM process is well documented [27, 28]. Case studies have shown that rapid manufacturing offers reduced costs when production volumes are low, many design iterations are to be explored, high geometric complexity is needed, or when new materials are to be explored [32, 33]. Additionally, material scrap rate can be signif-

[1] This chapter has been published in *Proceedings of the 42nd Aerospace Mechanisms Symposium* with Jonathan Jones and Larry Howell contributing as co-authors.

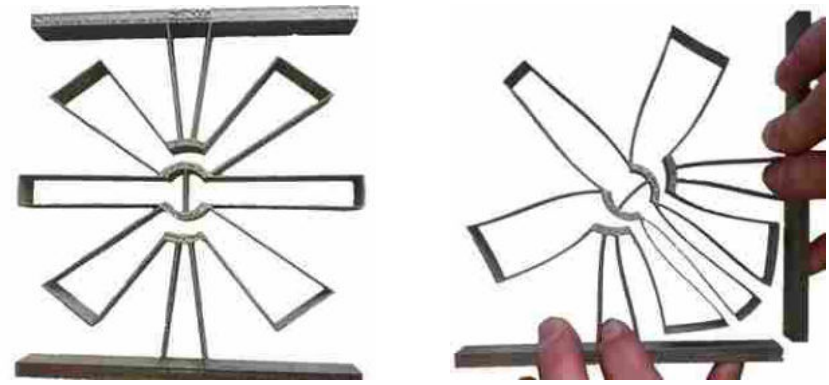


Figure 2.1: A compliant titanium hinge produced with EBM. This hinge is capable of $\pm 90^{\circ}$ of motion. Images provided courtesy of Robert Fowler [14]

icantly reduced by printing a near-net-shape part rather than machining it from solid billet [34]. Combining compliant mechanisms with rapid manufacturing techniques opens up interesting possibilities for creating compliant space mechanisms that have unprecedented performance.

Rapid manufacturing processes have been used in multiple aerospace applications, including ductwork [32, 33] and a capacitor housing on the International Space Station [35]. These applications used selective laser sintered nylon parts, which established a basis for rapid manufacturing as a viable method for producing parts. Structural brackets [27], a shrouded cryogenic impeller [27,36], and brackets for the Juno spacecraft [34] have also been manufactured in titanium using rapid manufacturing processes. A large-displacement hinge for space applications is shown in Figure 2.1 [14]. While most parts built thus far have been structural members (brackets, etc.) or non-structural assemblies (ductwork and housings), in our work we use additive manufacturing to create monolithic mechanisms for aerospace applications. As part of that effort, it is desirable to know what to expect when printing slender geometries, and maximum allowable stresses in EBM-produced titanium parts.

2.3 Material Considerations

2.3.1 Porosity of EBM-Produced Parts

EBM produced parts can achieve full density [38]. Wooten and Dennies claim that the fully dense region occurs in bulk parts about 1.25 mm (0.05 in) below the surface [27], but give

no explanation of how this figure was arrived at. This depth is more than the thickness of many printed flexures. While the region near the surface may not be fully dense, Murr et al. mention that such micropores have no effect on short-term tensile properties [7]. However, surface roughness and micro-cracks contribute to reduced fatigue life. Because of the slender geometry, machining of flexures is often impractical, so surface porosity is difficult to eliminate and must be accounted for in the design. This surface porosity constitutes a major obstacle to high cycle fatigue life. Hot isostatic pressing (HIP) improves the fatigue life of EBM produced parts [39]. If HIP treatment is impractical, property data obtained from raw (not treated with HIP or finish machined) tensile samples are available [6].

2.3.2 Thickness Correction Factor

Early design work for a two-degree-of-freedom (2 DOF) pointing mechanism [10] required testing the fabrication and performance of cross-axis flexural pivots (CAFP). These flexures have a number of good characteristics, including good stability and load carrying capacity [40]. The flexure was modeled in ANSYS to predict its torsional stiffness, which was compared to analytical solutions. Finally, the flexures were produced using EBM, and an example is shown in Figure 2.2, along with the pointing mechanism. The torque and deflection characteristics of three printed flexures were found. The FE model significantly over-predicted (30%) the stiffness of the printed. Because of high surface roughness, it was thought that perhaps not all of the thickness of the flexure contributes to its bending stiffness. Applying a correction factor of 0.83 to the thickness resulted in good agreement between the FEA and measured stiffness of the flexures. Later this correction factor was used to predict the overall stiffness of the pointing mechanism, again resulting in good agreement. Therefore, when using thin flexures, a thickness correction factor of 0.83 is recommended to accurately predict the torsional stiffness of printed flexures.

2.3.3 Allowable Stress

Two grades of titanium powder are currently produced for use in EBM machines: Ti6Al4V and Ti6Al4V ELI (ELI is extra low interstitials, which improves ductility and fracture toughness of the alloy). These two alloys have slightly different strength characteristics, but Ti6Al4V has

Figure 2.2: Cross-axis flexural pivot and 2 DOF pointing mechanism used to compare FEA and analytical models to measured stiffness.

slightly higher strength [39]. Table 2.1 presents strength data from several sources. These data were gathered from samples prepared in different ways; some used highly polished samples while other samples are tested in the as-built condition, with no post-processing or heat treatment. Rafi et al. found a strong correlation between build orientation and strength [6], while the manufacturer data make no distinction between build orientations [39].

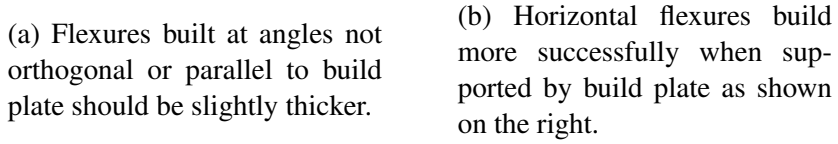
2.4 Geometry Constraints

2.4.1 Feature Geometry

Minimum wall or flexure thickness depends on feature orientation. A minimum thickness of 0.75 mm is recommended for flexures that have the thickness orthogonal or parallel to the build direction. If the flexure is built at other angles, 1.00 mm is recommended as the minimum thick-

Table 2.1: Summary of strength data gathered from other sources. (*) indicates that sample underwent HIP process. Strength units are MPa.

Material	S_y	S_{ut}	S_e	Notes
Ti6Al4V	950	1020	600*	Manufacturer data [39]
Ti6Al4V ELI	930	970	600*	Manufacturer data [39]
Ti6Al4V ELI	782	842	120	As-built vertical [6]
Ti6Al4V ELI	844	917	225	As-built horizontal [6]
Ti6Al4V ELI	869	928	325	Machined vertical [6]
Ti6Al4V ELI	899	978	300	Machined horizontal [6]



(a) Flexures built at angles not orthogonal or parallel to build plate should be slightly thicker.

(b) Horizontal flexures build more successfully when supported by build plate as shown on the right.

Figure 2.3: Two design modifications that increase likelihood of build success.

ness. If the flexure is built at some angle from the vertical, the larger dimension is recommended. Figure 2.3a illustrates this orientation dependence. The authors have had good success building flexures that rise at 45° from the horizontal when the flexures are 1.00 mm thick.

2.4.2 Thermal Stresses and Part Warping

Because the build chamber is maintained at between 500°C and 700°C, most stresses are relieved during the parts build cycle [11, 3], but some warping due to thermal stresses has been observed. Figure 2.4 shows a part where enough warping occurred that the part failed to build correctly. Although not fully understood, it is thought that this warping is due to stresses that occur when the molten metal solidifies but are subsequently relieved as the part is held at high temperature. Usually the part is bulky enough that these low stresses do not cause warping. For the geometry shown in Figure 2.4, the part was redesigned to have the flexures rest on the build plate (illustrated in Figure 2.3b). Supporting the flexures in this way eliminated the warping and allowed a successful build. It is postulated that other ways to avoid warping include better support of the cantilevered flexure from underneath (by having it connect to another portion of the part) or making it wider. In general, narrow, unsupported flexures are to be avoided.

Figure 2.4: Build failure due to warping of slender flexures.

2.4.3 Manufacturing Clearances

Clearances are important to ensure that the completed mechanism can move freely, without fusing sections that should move relative to one another. On a number of mechanisms with small (<2.0 mm) gaps, the final gap dimension was significantly less than was specified in the part file. Additionally, gaps must be wide enough that un-melted powder can be easily removed to allow motion in the mechanism. Experience with successful mechanisms suggests a minimum gap of 1.0 mm. The final gaps are less than the specified gap. In one instance, a gap as small as 0.66 mm was specified and the part successfully printed without fusing the two sections together; the measured clearance was 0.23 mm. These clearances were measured in the horizontal direction (parallel to the build plate). Vertical clearances should be specified larger, especially in areas where powder removal is difficult.

2.4.4 Powder and Support Removal

Another design consideration is that the geometry must allow for removal of un-melted powder and any support structure. Closed geometries should be avoided, or openings should be provided to allow access to loosen packed powder with hand tools or media blasting. In some cases this may not be possible, and post machining using special fixtures or tooling must be used. Figure 2.5 shows the tooling required to allow machining the inside of a particular feature. In another example, a linear spring was printed that consisted of Belleville washers stacked end-to-

Figure 2.5: The 2 DOF pointer mechanism in fixture for removal of powder and supports from inside a split-tube flexure.

end. The internal areas of the spring were inaccessible, but by using a press to compress the spring, enough powder was removed from between each washer segment to allow the spring to function as intended.

2.5 Summary

The following checklist can be used for designing compliant mechanisms for EBM manufacturing:

- Select minimum thickness for desired flexure orientation (0.75 mm for horizontal or vertical flexures and 1.0 mm for other angles)
- Find flexure length sufficient to bring stress into allowable range, subject to deflection and thickness
- Select flexure width to support applied loads without requiring excessive actuation torque
- Ensure minimum gap width is observed (1.0 mm)
- Ensure horizontal flexures are supported at both ends
- Ensure that geometry allows powder removal
- If post-machining is necessary, provide geometry for fixturing
- Orient part so that every feature is built up from build plate or supported in some way

CHAPTER 3. NON-DIMENSIONAL APPROACH FOR STATIC BALANCING OF ROTATIONAL FLEXURES

3.1 Introduction

[1] The objective of this research is to develop a general method for the approximate static balancing of compliant hinges. A compliant mechanism obtains its motion from the deflection of its constituent members. Because this eliminates sliding contact of surfaces, friction and subsequent wear can be avoided, leading to higher performance [1]. Because of the strain energy associated with bending the flexible members, compliant mechanisms generally have higher actuation effort compared to traditional mechanisms [16]. This can require larger actuators, which increases mass and cost. Static balancing is one strategy for reducing the actuation effort of compliant mechanisms [17, 18, 23, 26, 41, 42].

Static balancing is often accomplished by adding auxiliary springs that provide energy storage [18, 41]. As the mechanism is actuated, energy stored in the balancing elements is transfered to the deflected mechanism [42]. This means that less energy must be added during actuation, thus reducing actuation effort [26]. This strategy has been effectively incorporated into applications such as the design of surgical instruments and prosthetics [17, 23, 42].

Balancing elements commonly incorporate a negative stiffness mechanism, such as buckled beams in linear systems [43], or preloaded linear springs in rotational systems [3]. Other approaches use gravity balancing or systems of ideal springs [21, 42, 44].

Static balancing strategies do exist that do not require optimization; these rely on mathematically exact solutions [45, 46]. However, the design of statically balanced systems often requires the use of optimization routines [43, 47, 48]. Usually, the optimization problem minimizes the change in a mechanism's stored energy or searches for an appropriate negative-stiffness mechanism [3]. Depending on the system under consideration, this optimization may incorporate finite

[1] This chapter has been published in *Mechanism and Machine Theory* with Larry Howell contributing as a co-author.

element analysis (FEA) and topology optimization. This means that to design a statically balanced compliant mechanism, significant resources must be available to develop and validate the model being used. Additionally, optimization routines utilizing FEA can quickly become cumbersome due to the relatively long solution time of non-linear FEA and the many function calls of most optimization routines.

Finally, building practical statically balanced mechanisms is difficult because the balancing element is often bulky, making the system much larger than is convenient [21].

The method herein presented provides an approximate solution to the static balancing problem. Although perfect balancing is not achieved, an average reduction in stiffness of 87% is achieved for an eighty degree range of motion. This method does not require FEA, and can avoid some of the optimization required by other approaches for static balancing. For code examples, see Appendix A.

3.2 Nomenclature

In this work, "load-independent (LI) joint" is a joint with a rotational stiffness that is not a function of applied lateral loads. This is modeled as a pin joint with a torsional spring. A "load-dependent (LD) joint" is a joint whose stiffness changes when a lateral load is applied. An LD joint can be modeled as an LI joint if a relationship can be found between the applied lateral loads and joint stiffness.

In this work the statically balanced system consists of an LI joint of finite, constant stiffness that is balanced by the addition of a pre-loaded constant-stiffness linear spring. The spring connects at points equidistant from the pivot, as shown in Figure 3.1. This simplified system can represent load-dependent systems with proper application of the pseudo-rigid-body model [1].

Variables and their relationships are included in the following lists. The first list is for variables directly related to balancing of LI compliant hinges, illustrated in Figure 3.1

k_θ = Torsional stiffness of LI joint, or corrected stiffness of LD joint (with applied loads)

k_l = Stiffness of balancing spring

k = Stiffness of balanced system

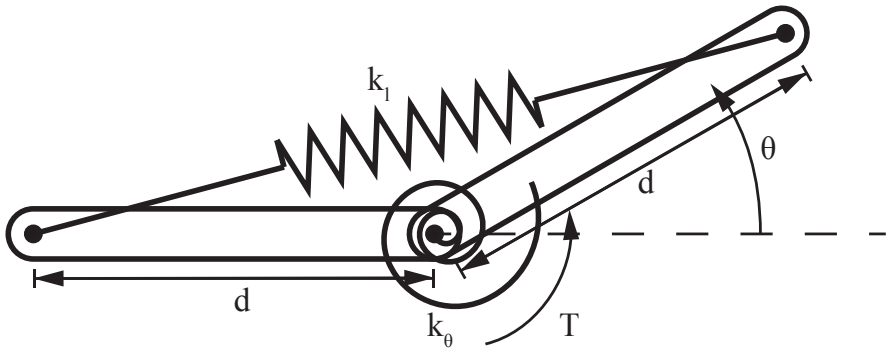


Figure 3.1: A LI system with balancing spring and associated variables.

d = Distance from pivot center to balancing spring attachment points

x_0 = Free length of balancing spring

$P = k_l(2d - x_0)$ = Preload applied to balancing spring

θ = Angle of deflection of the LI or LD joint

$T = k\theta$ = Torque required to deflect hinge through angle θ

$\Pi_1 = k_\theta/(Pd)$ = Pi group governing torsional stiffness

$\Pi_2 = k_l d/P$ = Pi group governing stiffness of balancing spring

The following list contains variables related to the design of a cross-axis flexural pivot (CAFP) that has a stiffness that is load-dependent. See Figure 3.2 for a depiction of geometric variables.

E = Young's modulus of the flexure material

b = Width of CAFP flexure strip

t = Thickness of CAFP flexure strip

$I = bt^3/12$ = Moment of inertia of CAFP flexure strip

L = Length of CAFP flexure strips

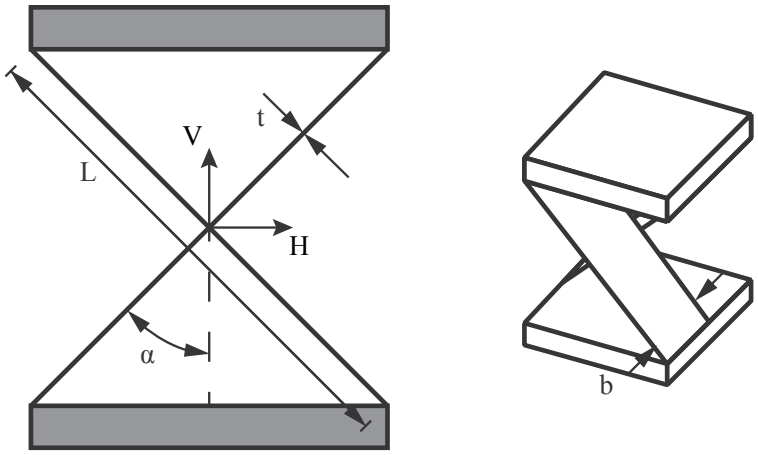


Figure 3.2: A cross-axis flexural pivot with associated variables.

k'_θ = Uncorrected torsional stiffness of LD joint (no applied loads)

The following list contains variables used to correct the stiffness of a CAFP to account for the effects of applied loads. Adapted from Wittrick [49]. See Figure 3.2 for a depiction of geometric variables. The loads V and H are applied to the moving block of the CAFP.

V = Vertical load applied to hinge

H = Horizontal load applied to hinge

α = Half the intersection angle of the CAFP flexures

$v = VL^2 \sec(\alpha)/(EI)$ = Non-dimensionalized applied vertical load

$h = HL^2 \csc(\alpha)/(EI)$ = Non-dimensionalized applied horizontal load

β_i = Dimensionless parameter describing the forces in CAFP flexures

ϕ_i = Dimensionless parameter describing the stiffness of the individual CAFP flexures

3.3 Balancing of Load-Independent Hinge-Spring System

Because a general solution to static balancing is sought, it is desirable to use dimensional analysis techniques to analyze the balanced systems.

Recall the Buckingham-Pi theorem:

> If an equation involving k variables is dimensionally homogeneous, it can be reduced to a relationship among $k-r$ independent dimensionless products, where r is the minimum number of reference dimensions required to describe the variables [50].

The energy of a load-independent system, E, can be written as the sum of the potential energies of the torsional and linear springs, as follows:

$$E = \frac{k_\theta}{2}\theta^2 + \frac{k_l}{2}(\sqrt{2d^2(1+\cos(\theta))} - x_0)^2 \tag{3.1}$$

Since the objective is to minimize torque (T), and $T = \frac{dE}{d\theta}$, we can take the derivative with respect to θ as

$$T = k_\theta\theta - \frac{k_l(\sqrt{2d^2(1+\cos(\theta))} - x_0)d^2\sin(\theta)}{\sqrt{2d^2(1+\cos(\theta))}}. \tag{3.2}$$

Approximating the torque is a linear function of theta allows us to divide by θ, giving the mechanism stiffness ($k = \frac{T}{\theta}$) as

$$k = \frac{k_\theta\theta - \frac{k_l(\sqrt{2d^2(1+\cos(\theta))} - x_0)d^2\sin(\theta)}{\sqrt{2d^2(1+\cos(\theta))}}}{\theta}. \tag{3.3}$$

This result will be used later. Setting the torque from Equation (3.2) equal to zero and using $x_0 = 2d - P/k_l$ gives:

$$k_\theta\theta = \frac{k_l(\sqrt{2d^2(1+\cos(\theta))} - (2d - P/k_l))d^2\sin(\theta)}{\sqrt{2d^2(1+\cos(\theta))}} \tag{3.4}$$

Equation (3.4) is a homogeneous equation with four dimensioned variables - k_θ, k_l, P, and d and two dimensions (force and length). Thus, $k = 4$ and $r = 2$, and the system can be described by two non-dimensional parameters, designated Π_1 and Π_2. Since it is desirable that these parameters have a physical, intuitive meaning it is convenient to select force P and distance d as repeating variables so that each pi-group deals with a stiffness term independently. Through application of dimensional analysis, we have:

$$\Pi_1 = \frac{k_\theta}{Pd} \tag{3.5}$$

and

$$\Pi_2 = \frac{k_l d}{P}. \tag{3.6}$$

We now assume that a relationship between our pi-groups exists, written as $\Pi_2 = \phi(\Pi_1)$. As a result of the Buckingham-Pi theorem, this relationship will govern the stiffness of the system. If a relationship between Π_2 and Π_1 can be found that results in a balanced system, any combination of system parameters k_θ, k_l, P, and d that follows this relationship will yield a statically balanced system. This relationship was found using an optimization algorithm, giving a relationship between the Π-groups that guarantees approximate statically balanced behavior for a system governed by k_θ, k_l, P, and d.

A program was written to find the relationship between Π_1 and Π_2 for $0.2 \leq \Pi_1 \leq 1$. This script used a particle swarm optimization [51] routine to minimize $|\frac{k}{k_\theta}|$ (see Equation (3.3)) calculated over a range of $0 < \theta \leq 20^\circ$ for a given value of Π_1. Optimization variables were k_l, P, and d. k_θ was found from Equation (3.5). After minimizing the normalized stiffness $|\frac{k}{k_\theta}|$, Π_2 was calculated from Equation (3.6) using the final variable values. A particle swarm algorithm was used because of its ability to find a global optimum. See Appendix A for more details on the optimization code.

This was repeated for other values of Π_1 in the range of $0.2 \leq \Pi_1 \leq 1$ to find Π_2 as a function of Π_1 (see Figure 3.3). Figure 3.4 plots $|\frac{k}{k_\theta}|$ against Π_1. The plot shows that in the region of $\Pi_1 = 0.5$ a close approximation of perfect balancing is achieved. By plotting Π_2 as a function of Π_1, as in Figure 3.3, we can see that using $\Pi_1 \geq 0.5$ gives $\Pi_2 < 0$, which is inconvenient for design purposes (it could require k_l to be negative). Table 3.1 lists convenient Π terms along with the expected reduction in stiffness. A curve fit for the data in Figure 3.3 is

$$\Pi_2 = -102.54\Pi_1 + 51.104 : 0.2 \leq \Pi_1 \leq 0.81 \tag{3.7}$$

with an R^2 value of 1.

Rearranging the data in Figure 3.4 and applying a curve fit gives the % reduction in stiffness as

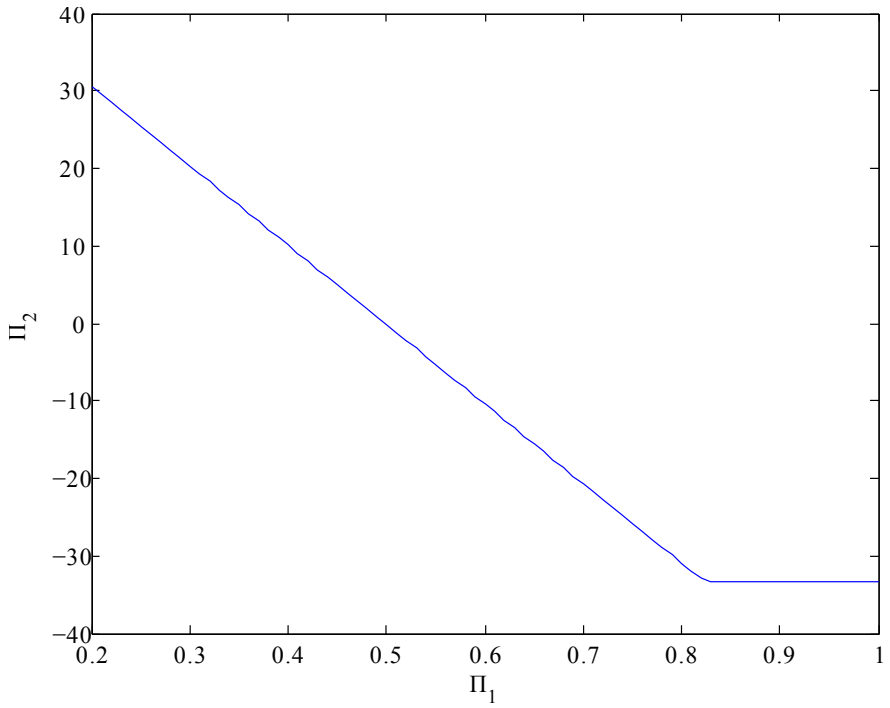


Figure 3.3: Π_2 plotted as a function of Π_1. The flat line occurring at $\Pi_1 = 0.82$ is an artifact of the optimization constraints.

$$\frac{k_\theta - k}{k_\theta} \times 100 = \begin{cases} 2414\Pi_1^3 - 3336\Pi_1^2 + 1679\Pi_1 - 206.7 & : 0.2 \leq \Pi_1 \leq 0.5 \\ 111.85\Pi_1^2 - 215.42\Pi_1 + 179.53 & : 0.5 \leq \Pi_1 \leq 0.8 \end{cases} \tag{3.8}$$

with an R^2 value of 0.9998.

Table 3.1: Tabulated values of Π_1 and Π_2, with the expected reduction in joint stiffness.

Π_1	Π_2	$\frac{k_\theta - k}{k_\theta} \times 100$
0.5	-0.1673	100.0
0.49	0.8581	98.8
0.48	1.884	97.6
0.47	2.909	96.3
0.46	3.934	95.0
0.45	4.96	93.6

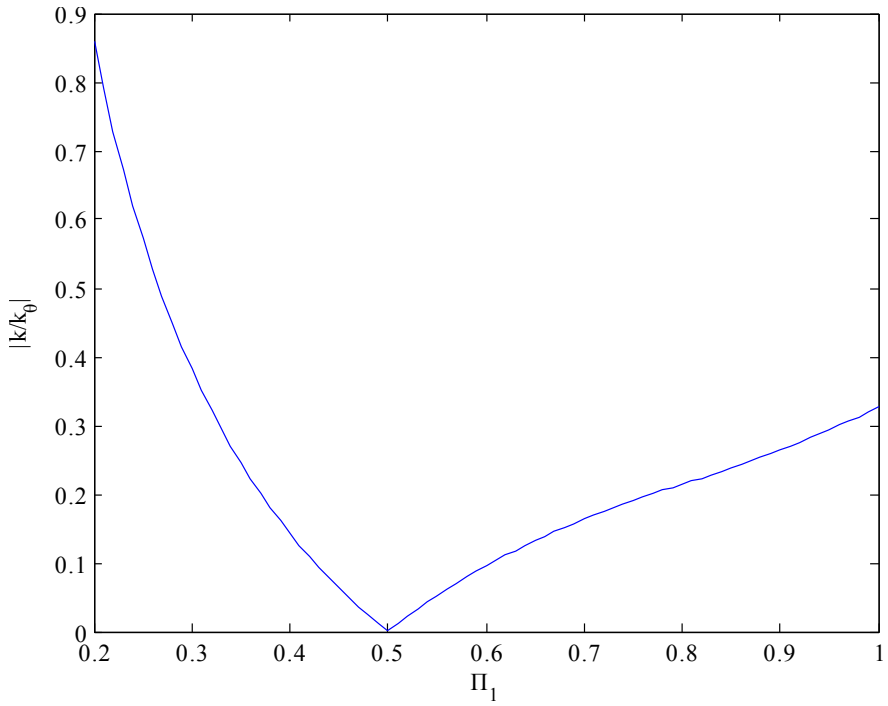


Figure 3.4: Normalized balanced stiffness as a function of Π_1.

This method only works when the flexure is load-independent; that is, when k_θ is not a function of applied lateral loads. The designer is free to choose two of any of the four parameters (k_θ, k_l, P, and d). The other two parameters are found from the Π group relation given in Equation 3.7 for the desired level of balancing (see Equation 3.8).

3.4 Extending to Load-Dependent Joints

In LD systems, torsional stiffness can vary with deflection and any applied loads. For simplicity here it is assumed that the stiffness dependence on deflection is small, and that load dependent effects dominate the joint behavior. Furthermore, it is assumed that only vertical and horizontal loads are applied to the flexure. Neither off-aixs moments nor axial loads are considered, although in some systems they certainly could have a strong influence on joint stiffness. Because the balancing spring exerts a pre-load on the flexure to be balanced, its stiffness (k_θ) is no longer the same as its stiffness without any applied loads (k'_θ). To use the Π-groups with a LD joint, a prediction of joint stiffness under load is required. Once the corrected stiffness k_θ is found for a specified pre-load P, the other joint parameters can be found from Π_1 and Π_2.

In this work, we will consider the static balancing of a cross-axis flexural pivot, sometimes called a cross-spring pivot. This is a type of flexure formed by crossing two flexible strips and has been used extensively to allow motion in many applications [40, 49, 52–54]. Additionally, it has been the subject of other investigations into static balancing strategies [3]. Morsch and Herder were able to balance a CAFP using a pair of zero-free-length springs with an average stiffness reduction of 70% in the physical prototype [3]. A final motivation for the use of a CAFP is the availability of published load-dependent behavior [49, 52]. In this work we employ the methods described here to take into account the change in CAFP stiffness when subjected to the compressive load.

Wittrick established that the stiffness of cross-axis flexural pivots is dependent on applied lateral loads [49,52]. He discussed how applied loads change the moments and loads applied to the constituent flexures, which effects their deflections. This same principle applies to many flexure systems commonly in use. A balancing method that accounts for the change in stiffness due to applied loads can provide a more balanced system. In this case, the applied load is due to the compressive pre-load of the balancing spring.

Wittrick's results are summarized here for convenience. He gives the stiffness of a CAFP as [49]:

$$k_\theta = \frac{EI}{L}(\phi_1 + \phi_2) \tag{3.9}$$

where

$$\begin{aligned} \phi_i &= \beta_i(\cot\beta_i - \beta_i) \\ \beta_1^2 &= \frac{1}{8}(v+h) \\ \beta_2^2 &= \frac{1}{8}(v-h) \end{aligned} \tag{3.10}$$

See Appendix A for a code implementation of these equations. Recall that v and h are non-dimensionalized horizontal and vertical loads. The balancing spring exerts a vertical load on the hinge because of its pre-load, P. Choosing an acceptable value of P and letting $V = -P$ and $H = 0$, allows the computation of k_θ for a given geometry. Choosing a Π_1 and its associated Π_2

for the desired stiffness reduction allows one to find the required d and k_l from Equations (3.5) and (3.6). Thus the Π-groups reduce the balancing problem to a system of two equations and four unknowns. Choosing two unknowns as design parameters allows the equations to be solved.

Alternatively, if it is desirable to select a value of P and k_l with flexures of a given moment of inertia, the associated k_θ can be found to satisfy Equation (3.5), and L can be found with an optimization loop. Because Equations (3.4) and (3.9) contain trigonometric terms, a non-gradient based optimization routine such as particle swarm optimization is effective.

3.5 Example Design

This approach was followed to design a CAFP. Convenient values of k_l and P were chosen to match those of a commercially available tension spring, and a flexure moment of inertia was selected so that the CAFP could be built from available spring steel. A flexure length was found along with torsional stiffness k_θ and d. The resulting design variables are listed in Table 3.2.

For comparison, let us attempt to also design a balanced joint using the same Π-groups and flexure geometry but without accounting for the LD behavior of the CAFP. Joint parameters for this joint are given in Table 3.3. Its behavior was simulated with FEA and is plotted in Figure 3.5 with the unbalanced CAFP and balanced CAFP that was designed with considerations made for LD behavior. From this plot it is clear that the load-dependent behavior cannot be neglected when

Table 3.2: Design parameters for balanced CAFP.

Parameter	Value	Units
Π_1	0.49	
Π_2	0.8581	
E	207	GPa
L	6.594	cm
t	0.381	mm
b	1.272	cm
I	5.865×10^{-6}	cm^4
k_θ	0.6446	N-m/rad
k_l	228	N/m
P	18.7	N
d	7.042	cm
x_0	5.877	cm

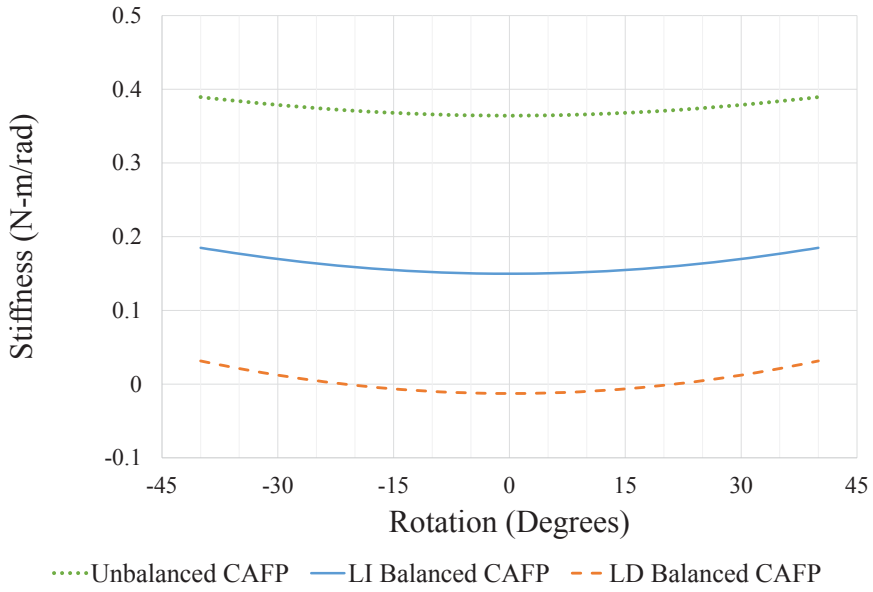


Figure 3.5: The stiffness of an unbalanced CAFP, together with its LI balanced and LD balanced versions.

designing a balancing mechanism. Even though the Π-groups predict balanced behavior, they do not capture load-dependent effects; this resulted in a joint that was only partially balanced.

The CAFP designed with considerations for load dependence was fabricated and tested. Simulations showed a stress-limited deflection of about 40°, so the prototype was designed with this physical limit in mind. The completed joint is shown in Figure 3.6. Experimental results are given in Section 3.6.

Table 3.3: Design parameters for hypothetical LI CAFP.

Parameter	Value	Units
k_θ	0.3672	N-m/rad
k_l	129.7	N/m
P	10.64	N
d	7.042	cm
x_0	3.347	cm

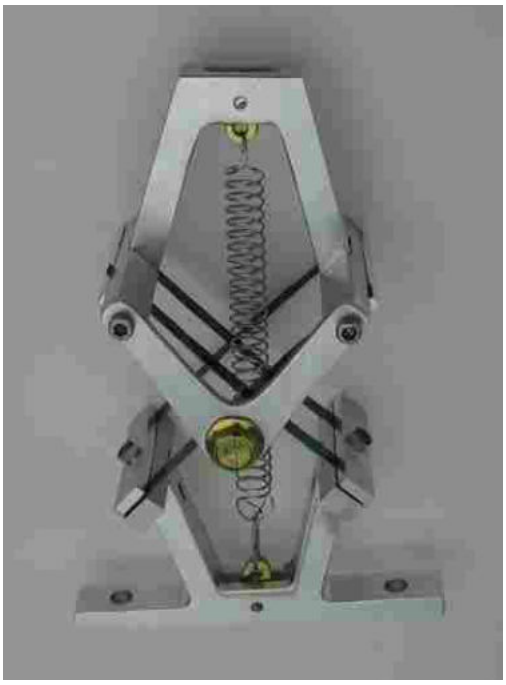


Figure 3.6: A prototype joint designed using the method detailed in this chapter.

3.6 Experimental Results and Discussion

The prototype balanced joint was designed and fabricated according to the design parameters of Table 3.2. Rigid sections were machined from 6061 aluminum bar stock while the flexures were cut from spring steel. Torque was measured using a torque transducer while the joint was displaced with a worm-wheel gear-set.

Figure 3.7 shows the predicted and actual stiffness in both the unbalanced and balanced configurations. The finite element results shown were obtained from an ANSYS simulation that used BEAM23 elements for the flexures and COMBIN14 elements for the linear spring. The balanced stiffness is not as low as predicted; this is due to deviation of the linear spring stiffness from nominal (193 N/m instead of 228 N/m), which causes P and k_θ to deviate from their optimal values. Changing any of these parameters changes the Π-groups that determine whether the system will be balanced. The curve labeled "Balanced, FEA Off-Nominal" was obtained by re-running the FEA using measured values for k_l, and P, with updated values of Π_1 and Π_2. This curve matches the measured stiffness values more closely than the FEA that used the nominal values for spring stiffness, etc; demonstrating that this was a significant source of error. The other source of error is

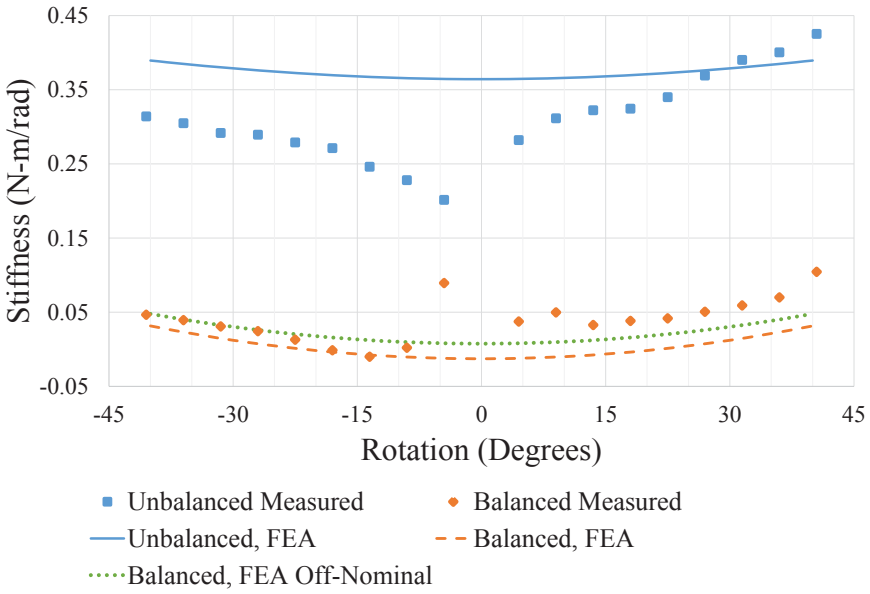


Figure 3.7: The stiffness of the balanced and unbalanced joints plotted for comparison.

in the measurement of the deflection angle, which was done by counting the revolutions applied to the worm of a 40:1 worm-wheel gearset. At small angles, small errors in angle measurement have a large effect on the observed stiffness. This explains the large errors that occurred near $\theta = 0$ in Figures 3.7 and 3.8, while at larger angles the error is significantly less.

Figure 3.8 shows the percent stiffness reduction calculated as $\frac{k'_\theta - k}{k'_\theta} \times 100$. Again, the stiffness reduction was not as great as was predicted due to manufacturing errors. An average stiffness reduction of 87% was achieved. The deviation from the reduction in Table 3.1 can be explained by how stiffness reduction is calculated. Stiffness reduction in Table 3.1 is calculated as $|(k_\theta - k)/k_\theta|$, while the stiffness reduction shown in Figure 3.8 is $|(k'_\theta - k)/k'_\theta|$, and the compressive load P makes $k'_\theta < k_\theta$.

This prototype verifies the balancing method presented here. Using non-dimensional parameters as a balancing criterion simplifies the design process for load-independent joints, which can be extended to load-dependent joints. This makes the rapid design of balanced joints practical in many applications. By taking into account the change in stiffness of a flexure due to joint pre-load, a better balancing solution can be achieved than if the flexure stiffness is assumed to be independent of load.

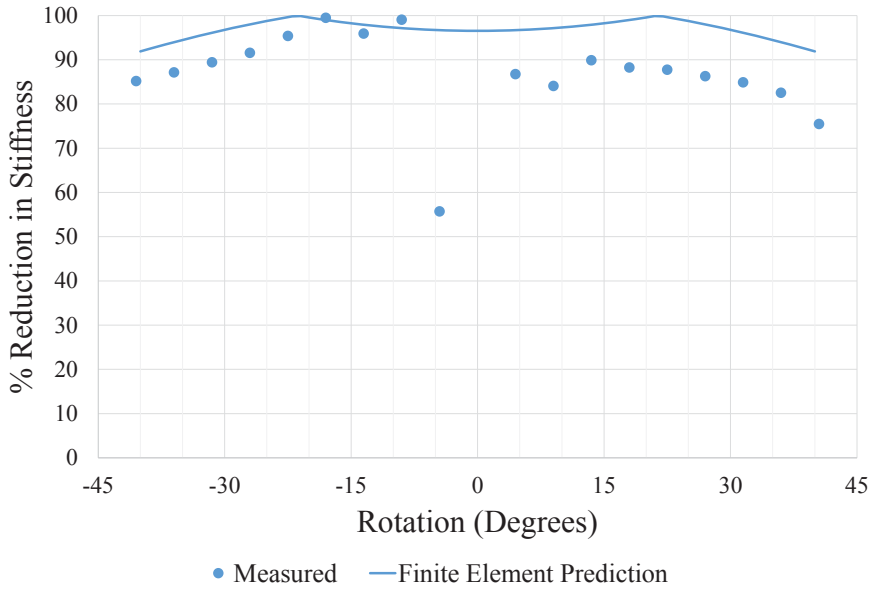


Figure 3.8: The percent reduction in stiffness of the balanced joint.

3.7 Conclusion

It has been shown that the use of the Π-groups can simplify the design of balancing mechanisms for compliant hinges that exhibit load-independent behavior. It has also been shown that the Π-groups are equally valid when used in conjunction with load-dependent joints whose stiffness under load can be predicted.

A prototype CAFP was built and tested. Results show that the stiffness-correction method results in highly balanced joints with large deflection capabilities. Although the balancing method and stiffness correction were demonstrated with a cross-axis flexural pivot, the result is general for any joint having load-dependent stiffness.

CHAPTER 4. A METHOD FOR DETERMINING LOAD-DEPENDENT STIFFNESS OF FLEXURES

4.1 Introduction

[1] Because compliant mechanisms derive their motion from the deflection of flexible members [1], they often posses inherent stiffness that may be undesirable [17, 18]. Static balancing is one method used to reduce mechanism stiffness [3,20–22,24,25,42,44,48,55,56]. Static balancing usually is done by adding auxiliary bodies or springs to an existing mechanism. By careful design of the auxiliary bodies and springs, the effective stiffness can be greatly reduced.

In Chapter 3 we outlined a method for static balancing that relies on a set of non-dimensional parameters to design a statically balanced mechanism [57]. The method uses a single auxiliary spring with a known preload P. By predicting the stiffness of a given flexure subjected to a compressive load P, a balanced system can be designed.

To make the non-dimensional balancing method more generally usable, in this chapter we present a method for determining the load-dependent stiffness behavior for a number of common flexure types. This behavior has interesting implications for static balancing. Interesting aspects of each joint's behavior are discussed while identifying new strategies for static balancing.

4.2 Background

Compliant mechanisms derive their motion from deflection [1]. The strain energy associated with deflection in the actuated mechanism means that the force required to maintain position is usually non-zero [16]. This often requires larger actuators, which have higher mass and cost. Static balancing mitigates the effects of strain energy, allowing the use of smaller actuators while still benefiting from the use of compliant mechanisms.

[1] This chapter has been published in *Proceedings of the ASME IDETC 2015* with Jared Bruton and Larry Howell contributing as co-authors.

Herder described the concept of using springs to compensate for undesired changes in strain energy [18]. Early work focused on reducing input force in surgical instruments and prosthetics [17, 23]. These designs incorporate a preload and some finite potential energy. Stored energy is released from the preloading members as the device is actuated. This energy release aids in the actuation of the mechanism. Because the net change in energy is small, the input force is reduced, and in some cases can be nearly eliminated [26].

By using a simplified set of joint parameters, it is possible to quickly design an approximately balanced rotational joint [57]. This balancer incorporates a linear spring exerting a compressive preload on the flexure to be balanced. This preload has the potential to alter the flexure stiffness. If the load-dependent stiffness behavior of a joint is known, it can be balanced using this method.

Wittrick [49, 52] quantified this load-dependent stiffness behavior for cross-axis flexural pivots (CAFPs) of various configurations. We show that this model can also be applicable to a triangle flexure (TF) [54,58] and, by extension, the cartwheel hinge (CH) [58,59]. This work seeks to establish a method for determining load-dependent stiffness using finite element analysis (FEA) and results are compared to analytic results to confirm that the method has the desired accuracy. Once the load-dependent stiffness behavior of a joint is understood, it is possible in some cases to find static balancing strategies that rely only on a pre-load and no balancing element.

4.3 Methods

Finite element modeling was done using commercial software (ANSYS). Models used beam elements (BEAM23) for flexible members. Vertical and horizontal loads were applied to the moving block at the pivot center (see Figure 4.1), following the convention established by Wittrick [49, 52]. This was done by extending a section of the moving block to the theoretical center of motion and applying force loads to that point. The moving block was rotated 1°, following Wittrick's convention of assuming small displacements.The reaction torque was recorded and stiffness found as $T/\Delta\theta$. During this study it was found that a high mesh density was required to obtain a mesh-independent solution when high loads were involved. Mesh densities used are listed in Table 4.1.

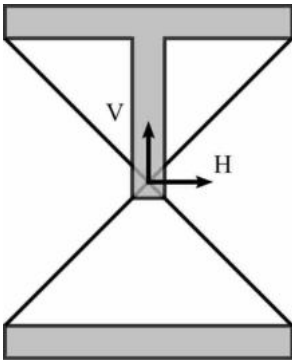


Figure 4.1: The joint arrangement and application of horizontal (H) and vertical (V) loads for a cross-axis flexural pivot (CAFP).

For convenience and to remain consistent with the literature [52, 54, 58] horizontal and vertical loads were non-dimensionalized using

$$\eta = \frac{HL^2}{EI} \tag{4.1}$$

and

$$\nu = \frac{VL^2}{EI} \tag{4.2}$$

where H and V are force loads in the horizontal and vertical directions, L is the characteristic flexure length, E is the material's Young's modulus, and I is the flexure's area moment of inertia. H is positive to the right, while a positive value of V is a tension load and a negative value a compression load. The rotational stiffness was non-dimensionalized using

Table 4.1: Mesh density used in different joint types.

Joint	Number of elements/flexure
CAFP	80
TF	160
CH	160
SLFP	40

$$\kappa = \frac{KL}{EI} \tag{4.3}$$

where K is the rotational stiffness of the flexure, and other variables are the same as defined previously. In these equations, Greek letters indicate non-dimensional parameters while Latin letters indicate dimensioned quantities.

A thorough analytic treatment of the CAFP is given in [49, 52]. Stiffness behavior was shown in Figure 4.2. The analytic solution is given as

$$\kappa = \phi_1 + \phi_2 \tag{4.4}$$

where

$$\phi_i = \beta_i(\coth\beta_i - \beta_i) + \frac{\rho^2\beta_i^3}{(\beta_i - \tanh\beta_i)} \tag{4.5}$$

and

$$\begin{aligned} \beta_1^2 &= (\nu + \eta)\sqrt{2}/8 \\ \beta_2^2 &= (\nu - \eta)\sqrt{2}/8 \end{aligned} \tag{4.6}$$

This formulation differs from Wittrick [49] because the non-dimensionalization used here for the vertical and horizontal loads does not account for variation in the angle at which the blades of the flexure cross. This model was developed to analyze CAFPs, but with additional modifications Equations (7.3), (4.5), (4.6) can represent other flexure types, including triangle flexures and cartwheel hinges. Namely, for a symmetric CAFP $\rho = 0$, and for a triangle flexure $\rho = 1$. A cartwheel hinge is simply two triangle flexures joined at the center. Thus, a single model can be used to analyze joints with distinctly different behavior.

Our method can be summarized in the following steps:

1. Identify for what range of non-dimensional loads the joint behavior need be evaluated
2. Develop an FEA model to represent the chosen joint
   (a) Find an analytical model for comparison

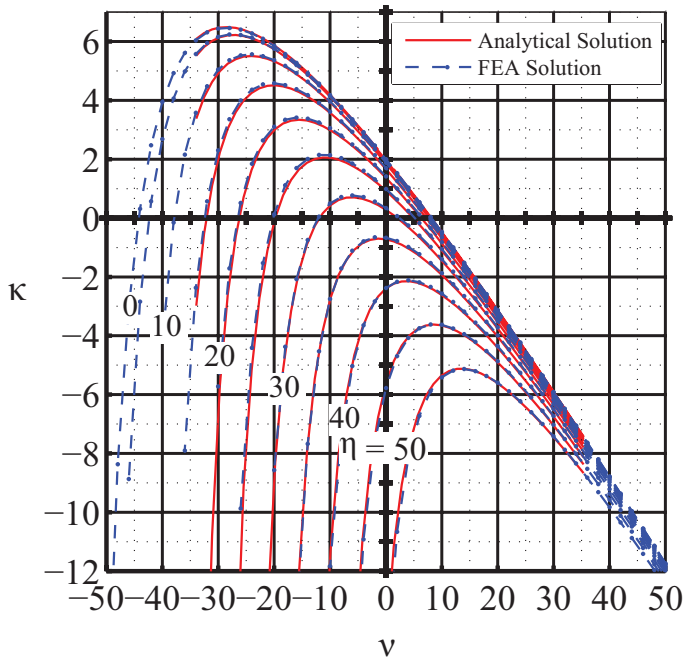


Figure 4.2: Comparison of FEA results with analytic solution for a symmetric CAFP [49].

(b) Beam elements are often the most appropriate for flexure analysis

(c) Displacement loads generally solve more reliably than force loads

(d) Using parametric geometry definitions can yield more versatile and useful models

(e) Carefully consider boundary conditions to avoid over- or under-constraining the problem

(f) Automating post-processing can save significant amounts of time

3. Ensure that the FEA model behaves as expected when unloaded

4. Calculate stiffness from rotation and reaction torque, and non-dimensionalize it

5. For a given η, iterate through the desired range of ν (or vice versa)

6. Repeat step 5 at increased mesh densities to ensure a mesh-independent solution

7. Repeat steps 5 and 6 for every desired value of η

4.4 Verification

The FEA results were compared to analytic results for the cross-axis flexural pivot. Figure 4.2 plots analytic results alongside results for the CAFP obtained through FEA, which agree with results reported in [49]. Additionally, stiffness values for each joint with zero applied loads in both the horizontal and vertical directions were compared to published values for stiffness of the unloaded flexures [40, 58, 60]. This data is tabulated in Table 4.2.

4.5 Load Dependent Stiffness

The above method was repeated for triangle flexures [58], cartwheel flexures [59], and small-length flexural pivots [60]. Schematics of these joints are shown in Figure 4.3. Their associated FEA models are shown in Figure 4.4.

4.5.1 Cross-Axis Flexural Pivot (CAFP)

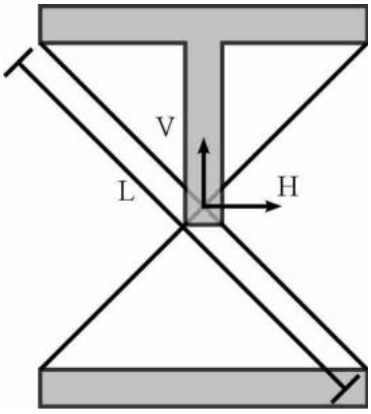
Equations (7.3), (4.5), (4.6) describe the behavior of the symmetric CAFP discussed here. Its behavior is shown in Figure 4.2.

4.5.2 Triangle Flexure (TF)

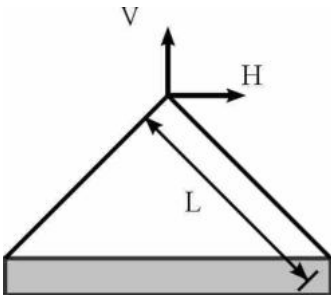
By recognizing that the triangle flexure is a special case of the CAFP where the strips cross at their extreme ends, we can use $\rho = 1$ in Equation (4.5) to calculate its stiffness.

Table 4.2: Comparison of unloaded stiffness determined by FEA to published stiffness values. Stiffnesses are non-dimensional.

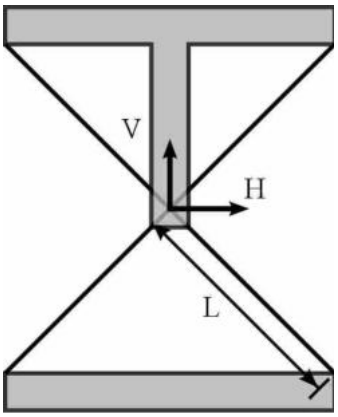
Joint	Unloaded Stiffness (κ)	Published Stiffness (κ)	% Difference
CAFP	2.00	2.154 [40]	7.70%
TF	7.79	8.0 [58]	2.70%
CH	4.01	4.0 [58]	0.25%
SLFP	1.00	1.0 [60]	0.00%



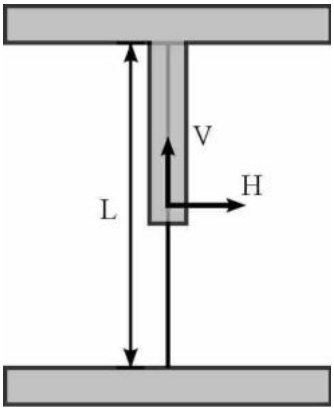
(a) The cross-axis flexural pivot. Note that the strips cross, but are not joined, at the center.



(b) The triangle flexure. This is a special case of CAFP where the strips cross at their extreme ends.



(c) The cartwheel hinge, composed of two triangle flexures joined in the center.



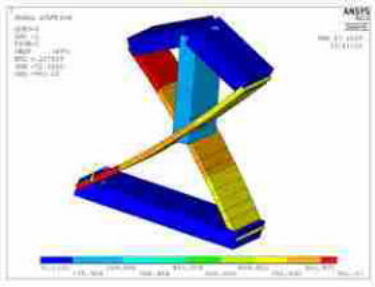
(d) The small-length flexural pivot.

Figure 4.3: Schematics for each flexure analyzed.

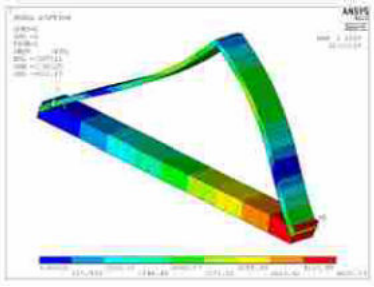
Stiffness behavior is shown in Figure 4.5. It was found that an applied rotation of 5° yielded a solution closer to the analytic model than the 1° rotation used in the other analyses.

4.5.3 Cartwheel Hinge (CH)

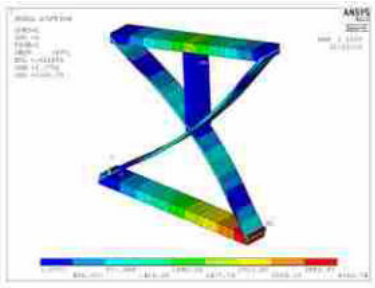
By recognizing that a cartwheel hinge consists of two triangle flexures in series [58] we can modify the analytic model to analyze its load-dependent stiffness, given in Equations (4.5), (4.6), and (4.7),



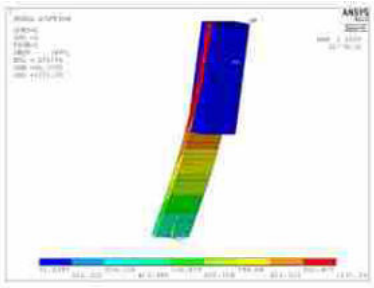
(a) The cross-axis flexural pivot.



(b) The triangle flexure. This is a special case of the CAFP where the strips cross at the extreme ends.



(c) The cartwheel hinge, composed of two triangle flexures connected in the center.



(d) The small-length flexural pivot.

Figure 4.4: Finite element models for each flexure analyzed. Deflection has been exaggerated.

$$\kappa = \frac{\phi_1 + \phi_2}{2} \tag{4.7}$$

where $\rho = 1$ in Equation (4.5). Note that the FEA solution is truncated for values of ν that cause the flexure to become unstable. Stiffness behavior is shown in Figure 4.6.

4.5.4 Small-Length Flexural Pivot (SLFP)

The small-length flexural pivot stiffness behavior is shown in Figure 4.7. As can be seen, the SLFP becomes unstable even under small values of η and ν. This characteristic makes the SLFP a poor choice for applications that requires load-carrying capability. Inverting the flexure may improve this behavior for some loading cases [61].

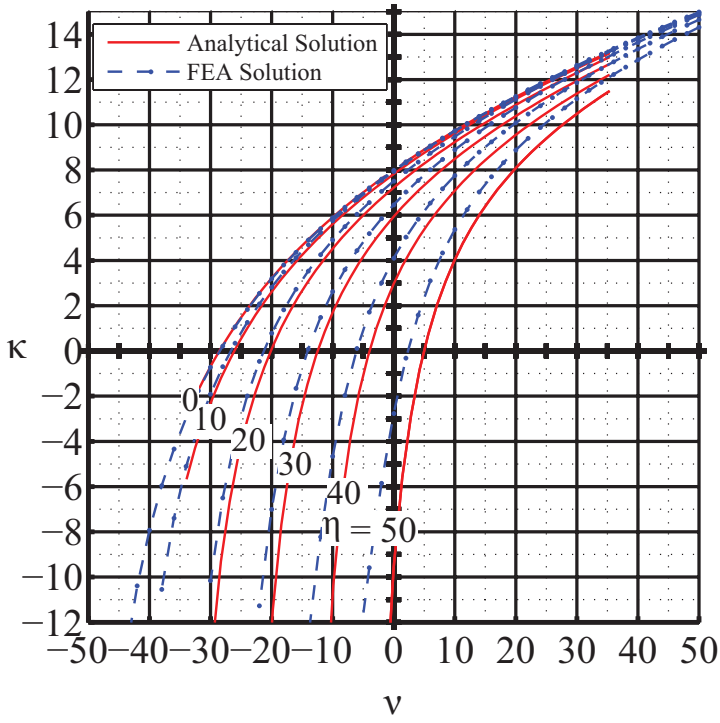


Figure 4.5: Stiffness behavior of the triangle flexure.

4.6 Discussion

The results of these models reveal interesting behavior of these joints, some of which is highlighted below.

4.6.1 Cross-Axis Flexural Pivot (CAFP)

The CAFP has the capability of carrying loads in all directions without becoming unstable. This makes it suitable for a large variety of load-bearing applications. However, its stiffness can vary significantly if the loading is not carefully controlled. Wittrick noted that a CAFP may be modified by changing the point at which the strips cross to have sensibly constant stiffness [49].

For the symmetric pivot considered here, a constant stiffness is achievable by operating in a loading regime where κ is at or near its maximum value for a given η. In fact, there exists a value of η at which $\kappa \approx 0$ for a range of ν. Solving Equations (7.3), (4.5), (4.6) gives $\eta \approx 32.5$, which results in $\kappa \approx 0$ for $-4.3 < \nu < -2.1$. When this loading condition is imposed, it

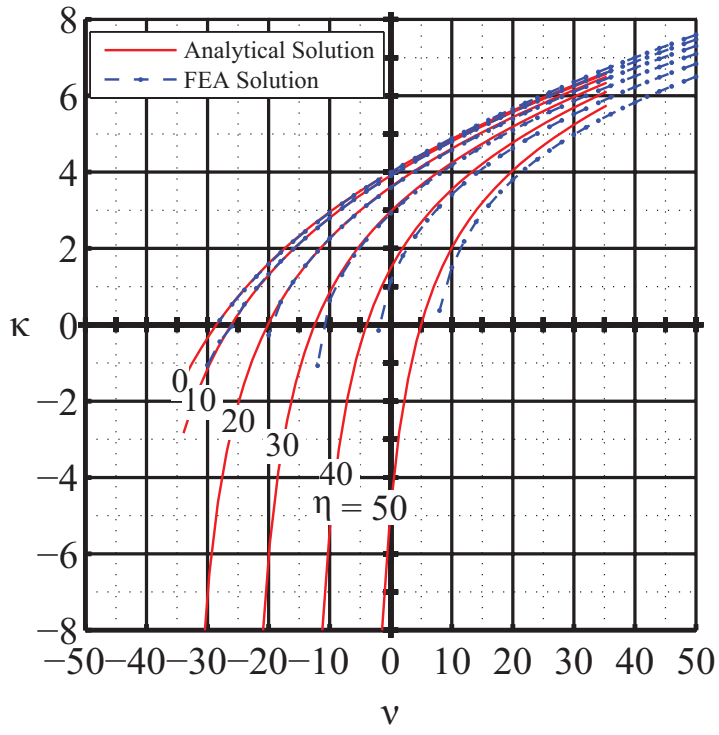


Figure 4.6: Stiffness behavior of the cartwheel hinge.

would result in an approximately statically balanced flexure with no additional springs or auxiliary bodies. Additionally, for given values of $\eta < 32.5$, two values of ν exist that yield a CAFP with zero stiffness. Finally, in regions where the CAFP exhibits negative stiffness behavior it could be coupled to a hinge with equal and opposite stiffness to yield a statically balanced mechanism [43].

4.6.2 Triangle Flexure and Cartwheel Hinge

Because of the similarity in the behavior of triangle flexures and cartwheel hinges they are considered together. Like the CAFP, there exist loading regimes where the stiffness of the joint becomes negative. However, in our simulations this negative stiffness tended to coincide with flexure buckling. Another principle difference between the behavior of these joints and the CAFP is that stiffness tends to increase with ν. Certain values of ν yield zero-stiffness joints, but these tend to be sensitive to variations in ν. Importantly though, both joints exhibit highly predictable load-dependent stiffness, making them candidates for statically balanced joints.

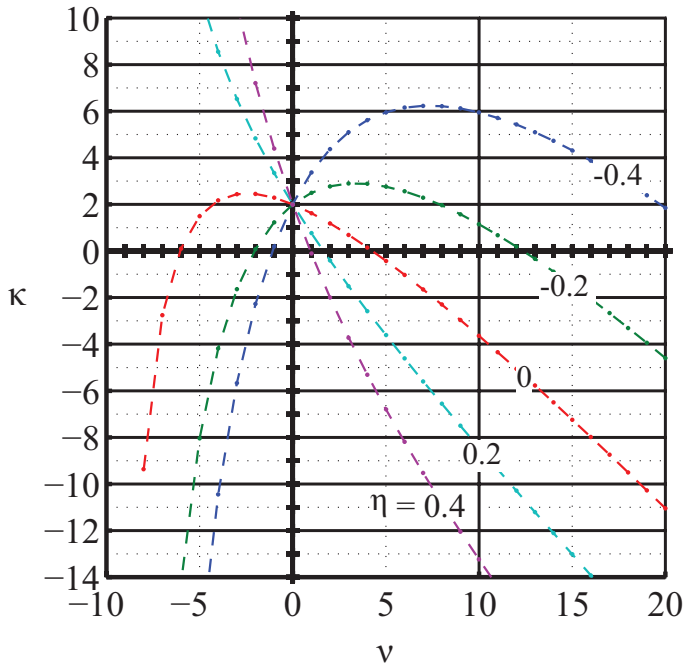


Figure 4.7: Stiffness behavior of the small-length flexural pivot.

4.6.3 Small-Length Flexural Pivot (SLFP)

The preloaded SLFP was analyzed using FEA and it exhibited the greatest range of behavior. Unlike the other flexures examined, at $\nu = 0$, κ remained invariant for all values of η. When subjected to moderate values of ν its stability decreased dramatically, and even very small changes in η drastically affect not only the magnitude of κ (when $\nu \neq 0$), but its slope in relation to ν. Because of this, the SLFP has been used in motion applications but has been avoided in load-carrying applications. Additionally, its sensitivity to varying loads makes the SLFP an unlikely candidate for static balancing.

4.7 Conclusion

The method presented here is a straightforward way to determine the load-dependent stiffness behavior of compliant flexures. We have demonstrated the method on four different flexure types, including cross-axis flexural pivots, triangle flexures, cartwheel flexures, and small-length flexural pivots. The approach prescribes a set of non-dimensional parameters consistent with the literature, a dense FEA mesh, and the application of horizontal and vertical force loads at a small

applied rotation. It was shown that this method accurately captured the behavior when compared with load-dependent stiffness predictions available in the literature.

We have also shown that by more fully understanding joint behavior while under load it is possible to identify strategies for static balancing that have heretofore gone unnoticed. It may be possible to statically balance flexures and mechanisms without the addition of springs or auxiliary bodies; perhaps the load applied to the mechanism could function as the balancer in some applications. Additionally, we have indicated strategies that would result in a more robust balancing solution; by operating a flexure in regimes where stiffness is nearly invariant, fluctuations in its loading conditions will not disturb its balanced behavior.

Because this method can be applied to models of arbitrary geometry, it can be used with any joint that can be modeled with FEA. Further, we have demonstrated its use on single-axis joints with forces applied in the plane perpendicular to the axis of rotation, but this method could be extended to multi-axis joints with three dimensional loadings.

CHAPTER 5. LATTICE FLEXURES: GEOMETRIES FOR STIFFNESS REDUCTION OF BLADE FLEXURES

5.1 Introduction

[1] This chapter introduces the lattice flexure as a means to reduce the motion-direction rotational stiffness of compliant mechanisms. A compliant mechanism obtains its motion from the deflection of its constituent members. This eliminates sliding contact of surfaces, avoiding friction and subsequent wear, and leading to higher performance [1]. Because of the strain energy associated with bending the flexible members, compliant mechanisms often have higher actuation effort compared to traditional mechanisms [16]. Static balancing is one way of reducing this actuation energy [17, 18, 23, 26, 41, 42]. Static balancing functions by introducing balancing elements that store and release energy as the mechanism is actuated. Because the net change in energy stored by the mechanism is small, the actuation effort is reduced. However, stiffer mechanisms require that more strain energy be stored by the balancing element [57]. By reducing the initial mechanism stiffness, simpler balancing elements can be used.

The stiffness of a flexure is governed by its material, boundary conditions, and geometry [1]. This work will consider the stiffness of an arbitrary material with elastic linear stiffness (i.e. constant Young's modulus in the elastic range). The boundary conditions are that of a cantilever beam subject to an end moment load. Therefore, the aspect of beam stiffness to be examined is beam geometry.

The conventional blade (or leaf-spring) flexure design is that of a prismatic rectangular-section beam [62] shown in Figure 5.1a. Much work has been done studying this kind of flexure to gain greater insight into its non-linear deflection and stiffness [63]. Changing the beam length, width, or thickness will result in a change in stiffness. The bending stiffness of a cantilever beam subject to a moment load is $k_b = EI/L$ [64]. E is the Young's Modulus, I is the second moment

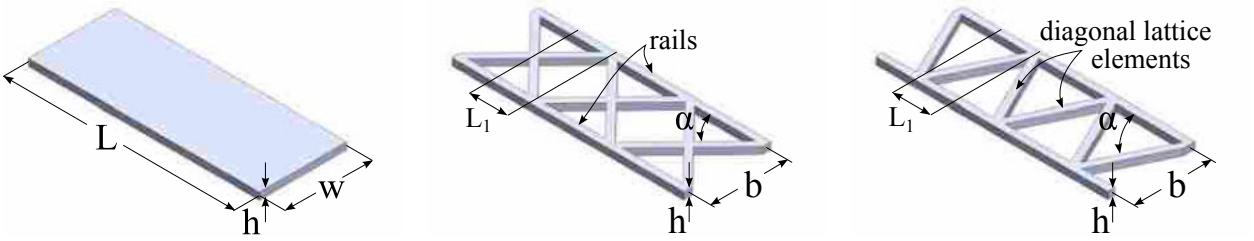
[1] This chapter has been published in *Precision Engineering* with Larry Howell contributing as a co-author.

of area, which for rectangular sections is given by $wh^3/12$. L, w, and h are the length, width, and thickness of the flexure, respectively.

Decreasing a flexures thickness can be a straightforward and efficient way of decreasing the bending stiffness. The lower bound of thickness is generally dictated by the available materials (stock sizes) and manufacturing processes or other design constraints. For example, in 3D printing, a process such as electron beam melting may be able to reliably print features no smaller than 1.0 mm thick [12]. Flexure width is limited in a similar way, with the addition that the flexure stability (its ability to withstand off-axis loads) decreases as the width decreases. Flexure length is often limited by mechanism envelope. Thus a flexure designer may arrive at the practical geometric limits of a flexure but still be unsatisfied with its performance [65]. The lattice flexure is introduced as one way of addressing this issue. This can be important in applications where it is necessary to reduce actuation effort while maintaining comparable stiffness in off-axis directions (such as in space applications where actuation effort can be proportional to actuator size, which can be proportional to actuator mass).

Flexures are important elements in many mechanical systems [47, 66, 67]. Different types of flexures have been the focus of recent studies, including cross-axis flexure pivots [68], cartwheel flexures [69], trapezoidal flexures [70], and others [71]. Methods for modeling and design of flexures include the pseudo-rigid-body model [60, 72], FACT [73], screw theory [74], matrix methods [75], and analytic methods paired with finite element analysis [62]. These methods differ in accuracy and complexity, but all are meant to aid the designer in arriving at a suitable configuration of flexures. Common concerns in flexure design include stiffness [47], stress and fatigue life [1], and off-axis (non motion-direction) stiffness [69].

In this work we introduce the lattice flexure, a new flexure type that has an envelope similar to a blade flexure but has dramatically reduced motion-direction bending stiffness and an increased ratio of support-direction to motion-direction bending stiffness. This reduced stiffness lowers the required actuation effort and simplifies the design of any static balancing system incorporating a lattice flexure. Lattice flexures have significantly lower mass while maintaining good off-axis stiffness. Figure 5.2 shows an early lattice flexure design in a 3D-printed titanium cross-axis flexural pivot. While this chapter cannot exhaustively investigate every aspect of lattice flexure behavior, some investigation of its bending and torsional stiffness properties is presented. The



(a) A conventional blade flexure. (b) An X-type lattice flexure. (c) A V-type lattice flexure.

Figure 5.1: Examples of different flexure types. Note that *b* is the distance between the rail centers, not the full width of the flexure.

Figure 5.2: A 3D printed titanium cross-axis flexural pivot with an early lattice flexure design.

improved performance is countered by increased manufacturing complexity, higher stresses, and lower load-carrying capacity. Advances in additive manufacturing and make monolithic fabrication of such flexures feasible. See Appendix E for examples of the ANSYS and Matlab code used in this work.

5.2 Approach

Figure 5.1 shows a conventional blade flexure and the proposed geometry for two lattice flexure types. Figure 5.1b shows the geometry of an X-type flexure, so named for the crossing of the diagonal lattice elements. Figure 5.1c shows the geometry of a V-type lattice flexure, so named because of the diagonal elements' resemblance to the letter "V." Both the X-type and V-type flex-

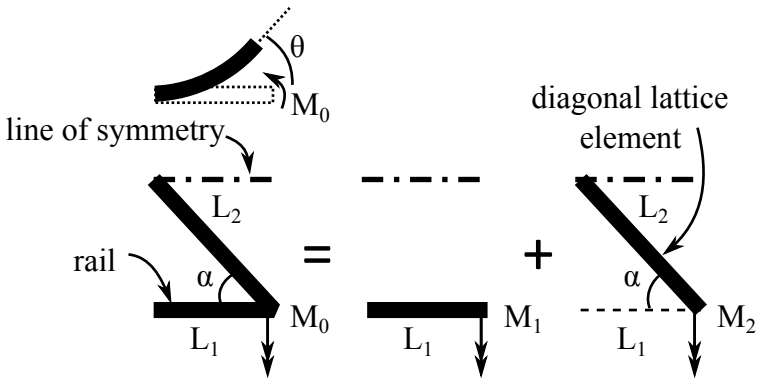


Figure 5.3: The X-type flexure is analyzed using the symmetry about the central plane of the flexure, using the variables L_1, L_2, and α.

ures are characterized by the ratio L_1/b and the aspect ratio η, where (for square lattice elements) $\eta = h/(h+b)$ (the thickness over the overall width). By removing material from the middle of the blade flexure the effective width is reduced. This reduces stiffness, and because the diagonal elements of the lattice are in combined bending and torsion (rather than pure bending), the percent reduction in bending motion stiffness is greater than the percent of material removed. While many geometries incorporating these concepts can be conceived, the geometries herein presented are meant to be a proof-of-concept and a starting point for further development.

5.2.1 Stiffness of Lattice Flexures

In this section we derive the stiffness of X-type and V-type lattice flexures. Figure 5.3 shows a single geometric unit of a lattice flexure and the nomenclature used in the derivation.

First we will derive the stiffness of an X-type lattice flexure. Variables used in this derivation are depicted in Figures 5.1 and 5.3. Note that the stiffness of only one quadrant of the X is analyzed and the overall stiffness is found from symmetry. The bending moment M_0 applied to the lattice element can be decomposed into the moments M_1 and M_2. M_1 is the moment carried by the "rails" of the lattice, while M_2 is the moment carried by the diagonal lattice element. The deflected angle θ induced by M_1 can be found from elementary beam theory as

$$\theta = \frac{M_1 L_1}{E I_r} \tag{5.1}$$

where I_r is the second moment of area of the rail, E is the Young's modulus, and L_1 is the length of one half-unit cell of the X-type flexure.

Because the ends of the two segments are rigidly joined, the diagonal lattice element must also be deflected to this angle (θ). The angular deflection of the diagonal lattice element due to torsion (θ_t) in the beam is given by

$$\theta_t = \frac{M_2 \sin(\alpha) L_2}{KG} \tag{5.2}$$

where L_2 is the length of the diagonal lattice element, M_2 is the component of the applied moment (M_0) reacted by the diagonal lattice element, K is a section property (a function of lattice element cross-section [64]), G is the modulus of rigidity, and α is the lattice angle (see Figures 5.1b and 5.3). The angular deflection due to bending (θ_b) is given by

$$\theta_b = \frac{M_2 \cos(\alpha) L_2}{EI_l} \tag{5.3}$$

where I_l is the second moment of area of the diagonal lattice element.

The vector sum of these two angular displacements perpendicular to the flexure axis must be equal to θ, and substituting $\sin\alpha = b/2L_2$ and $\cos\alpha = L_1/L_2$ can be written as

$$\theta = \frac{M_2 b^2}{4L_2 KG} + \frac{M_2 L_1^2}{L_2 EI_l} \tag{5.4}$$

This expression is similar to that arrived at by Delimont et al. for a surrogate fold with similar geometry [76].

By factoring out M_2 and dividing, we can find an expression for the load carried by the diagonal element

$$M_2 = \frac{4L_2 KGEI_l \theta}{b^2 EI_l + 4L_1^2 KG} \tag{5.5}$$

The total moment M_0 is the sum of M_1 and M_2, or

$$M_0 = \frac{EI_r \theta}{L_1} + \frac{4L_2 KGEI_l \theta}{b^2 EI_l + 4L_1^2 KG} \tag{5.6}$$

Recalling that $G = \frac{E}{2(1+\nu)}$, this can be simplified to:

$$M_0 = \frac{E\theta}{L_1}\left(I_r + \frac{2KL_1L_2I_l}{b^2I_l(1+\nu)+2L_1^2K}\right) \tag{5.7}$$

From Equation (5.7) we can find the stiffness of the lattice section, k, as $k = M_0/\theta$. To find the total X-type lattice stiffness k_X this should be doubled to account for symmetry:

$$k_X = \frac{2E}{L_1}\left(I_r + \frac{2KL_1L_2I_l}{b^2I_l(1+\nu)+2L_1^2K}\right) \tag{5.8}$$

With the substitution of $L_2 = \sqrt{L_1^2 + (b/2)^2}$ and some rearranging, Equation 5.8 then becomes

$$k_X = \frac{2E}{L_1}\left(I_r + \frac{2I_l\frac{L_1}{b}\sqrt{\left(\frac{L_1}{b}\right)^2 + 1/4}}{(1+\nu)\frac{I_l}{K} + 2\left(\frac{L_1}{b}\right)^2}\right) \tag{5.9}$$

The stiffness of a rectangular blade flexure is given by

$$k_{blade} = \frac{EI_b}{L_1} \tag{5.10}$$

The reduction in stiffness can be found from $(k_{blade} - k_X)/k_{blade}$.

$$\%\ \text{Reduction}_X = \left[1 - \frac{2}{I_b}\left(I_r + \frac{2I_l\frac{L_1}{b}\sqrt{\left(\frac{L_1}{b}\right)^2 + 1/4}}{(1+\nu)\frac{I_l}{K} + 2\left(\frac{L_1}{b}\right)^2}\right)\right] \times 100 \tag{5.11}$$

Thus, the stiffness reduction is a function of the cross section of the lattice elements, the cross section of the blade flexure it replaces, and the ratio of L_1/b.

Using a similar procedure to that outlined above, and the variables depicted in Figure 5.4, similar equations can be found for the V-type flexure. V-type flexures reduce stiffness more than the X-type. However, V-type flexures may be more difficult to fabricate because their slender sections are generally longer than those in similarly-sized X-type flexures. The stiffness of a V-type lattice

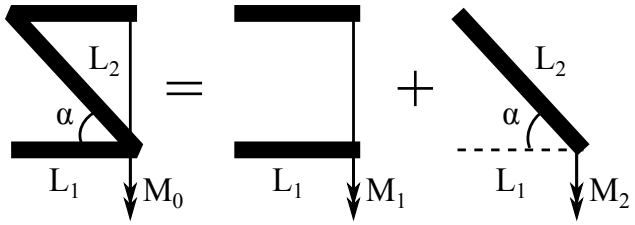


Figure 5.4: The V-type flexure is analyzed using the variables L_1, L_2, and α.

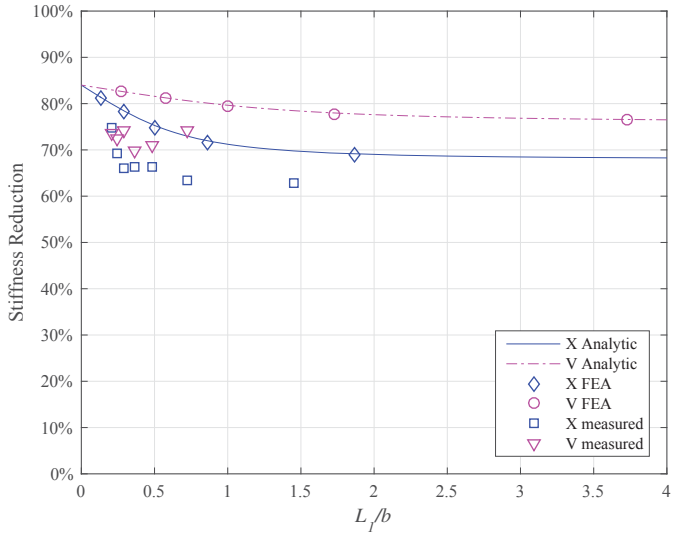


Figure 5.5: Stiffness reduction of the two lattice types as a function of L_1/b. The flexure geometry has $\eta = 0.08$ and the lattice elements have square cross sections with $I_r = I_l$.

flexure k_V is given by

$$k_V = \frac{E}{L_1}\left(2I_r + \frac{I_l \frac{L_1}{b}\sqrt{\left(\frac{L_1}{b}\right)^2 + 1}}{2(1+\nu)\frac{I_l}{K} + \left(\frac{L_1}{b}\right)^2}\right) \tag{5.12}$$

The stiffness reduction when compared to a rectangular blade flexure is given by

$$\%\ \text{Reduction}_V = \left[1 - \frac{1}{I_b}\left(2I_r + \frac{I_l \frac{L_1}{b}\sqrt{\left(\frac{L_1}{b}\right)^2 + 1}}{2(1+\nu)\frac{I_l}{K} + \left(\frac{L_1}{b}\right)^2}\right)\right] \times 100 \tag{5.13}$$

Equations (5.11) and (5.13) are plotted in Figure 5.5 and compared to calculated (using finite element analysis) and measured stiffness values of the two lattice types. The finite element

analysis (FEA) models were created in ANSYS using BEAM188 elements (see Appendix E for APDL code). This element type is suitable for 3D analysis, is based on Timoshenko beam theory, and includes shear deformation effects. The FEA results match the analytical solutions to within 0.58%. If the lattice elements are of a round cross section (holding η and h constant), stiffness reduction is even greater than for the square cross sections shown here. Equations (5.9) and (5.12) can be used to describe the stiffness of lattice flexures for any lattice element cross section. If a blade flexure is to be replaced with a lattice flexure of identical outer dimensions, $(EI)_{\text{lattice}} = (1 - \%\ \text{Reduction})(EI)_{\text{blade}}$. This allows us to calculate the stiffness of the new flexure (whether that flexure is a simple lattice flexure, or incorporated in a cross-axis flexural pivot or cartwheel hinge, etc.) as

$$k_{\text{lattice}} = (1 - \%\ \text{Reduction})k_{\text{blade}} \tag{5.14}$$

5.2.2 Off-Axis Stiffness Considerations

Considerations besides bending stiffness often come into play during mechanism design. One concern may be the off-axis stiffness of a flexure when subjected to transverse loads or moments [77].

One way to characterize the off-axis stiffness is to compare the support-direction bending stiffness to the motion-direction bending stiffness [15]. Here, the motion-direction bending stiffness is denoted as simply k, and the two support-direction stiffness values are denoted as k_b and k_t. k_b denotes bending stiffness perpendicular to the beam's long axis but orthogonal to its motion direction. k_t denotes the torsional stiffness of the beam (twisting along its long axis).

Because of the difficulty in obtaining an analytic solution, FEA was used to find the stiffness ratio for lattice flexures of both types. This was done by building a representative section of lattice flexure in the FEA model and meshing it with ANSYS BEAM188 elements (see Appendix E for APDL code). A z-axis rotation was applied to the free end of the flexure and the corresponding motion-direction bending moment calculated. This rotation was released and a warping-restrained x-axis (torsional) rotation was applied with the corresponding torque calculated. This torsional rotation was then released and a y-axis rotation was applied. The stiffness at each step was then

found. This process was repeated and the results calculated for multiple values of η and L_1/b for each flexure type. A blade flexure was also analyzed to provide a basis for comparison.

These FEA data were approximated by a polynomial surface fit. The behavior of the X-type lattice, shown in Figure 5.6 is approximated by

$$(k_t/k)_X = \left[0.03276 + 0.9915\frac{L_1}{b} - 0.6835\frac{L_1}{b}\eta + 0.332\left(\frac{L_1}{b}\right)^2 - 0.8216\eta^2\right] \times \left(\frac{L_1}{b}\right)^{-1.746+0.2498\eta-0.4555\frac{L_1}{b}} \tag{5.15}$$

and

$$(k_b/k)_X = \left[0.001667 + 11.17\eta - 3.911\eta\frac{L_1}{b} - 31.88\eta^2 + 0.1754\left(\frac{L_1}{b}\right)^2\right] \times \eta^{-2.869+0.2764\frac{L_1}{b}+1.675\eta} \tag{5.16}$$

which are valid for $0.0974 \leq \frac{L_1}{b} \leq 2.3659$ and $0.0196 \leq \eta \leq 0.2063$. These surface fits have R^2 values of 0.9995 and 0.9943, respectively.

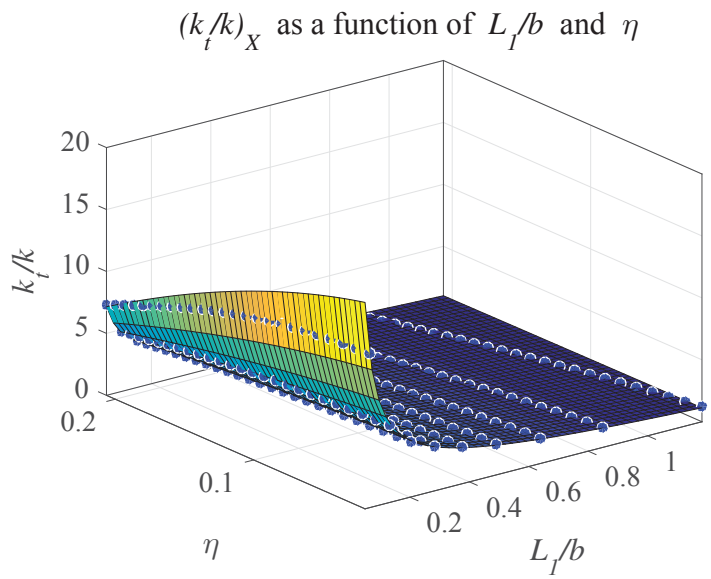
Similarly, the behavior of the V-type lattice, shown in Figure 5.7 is approximated by

$$(k_t/k)_X = \left[0.01173 + 0.299\frac{L_1}{b}1.813\frac{L_1}{b}\eta + 1.815\left(\frac{L_1}{b}\right)^2 - 0.05857\eta^2\right] \times \left(\frac{L_1}{b}\right)^{-2.223-0.9335\eta-0.1808\frac{L_1}{b}} \tag{5.17}$$

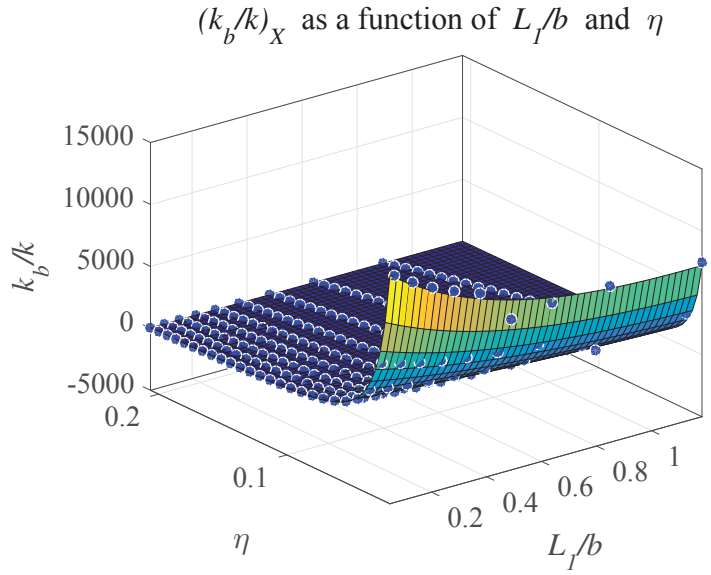
and

$$(k_b/k)_X = \left[0.5964 + 43.57\eta - 3.288\eta\frac{L_1}{b} - 131\eta^2 - 0.02594\left(\frac{L_1}{b}\right)^2\right] \times \eta^{-2.428-0.03889\frac{L_1}{b}+4.121\eta} \tag{5.18}$$

which are valid for $0.0974 \leq \frac{L_1}{b} \leq 2.3659$ and $0.0196 \leq \eta \leq 0.2063$. These surface fits have R^2 values of 0.9993 and 0.9992, respectively.



(a) Ratio of torsional stiffness to motion-direction bending stiffness.

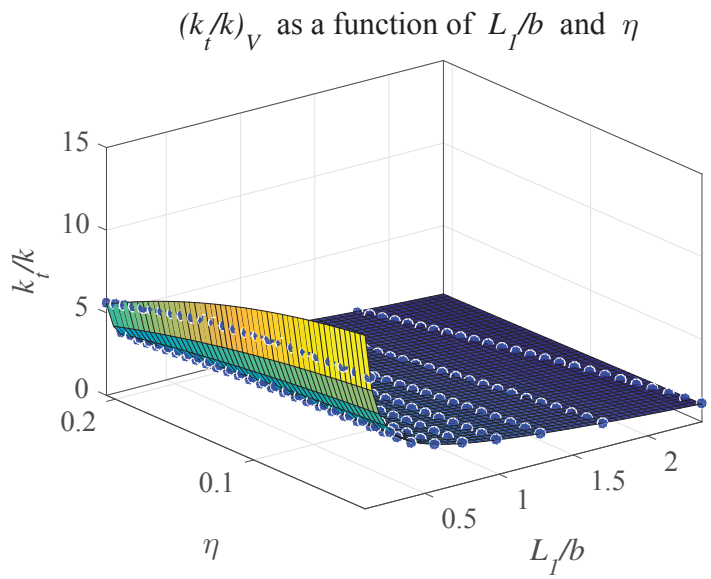


(b) Ratio of support-direction bending stiffness to motion-direction bending stiffness.

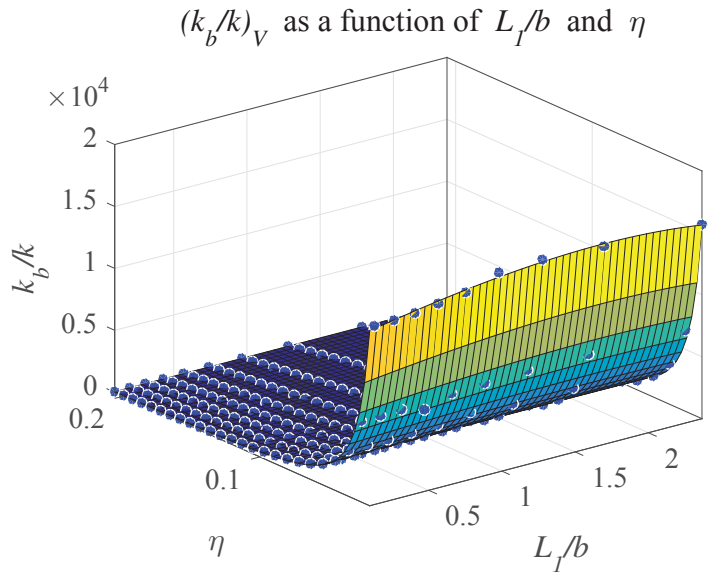
Figure 5.6: Off-axis stiffness behavior of the X-type lattice with square lattice elements.

The stiffness ratios for the two lattice types were plotted in Figure 5.8 side-by-side with the stiffness ratios for a blade flexure as a function of η. A constant value for L_1/b was chosen to be 0.75. Other values of L_1/b will yield similar curves, though the exact values will differ.

Other results are tabulated for comparison in Table 5.1. These predictions indicate that in some cases lattice flexures will have higher k_t/k and k_b/k ratios than conventional blade flexures.



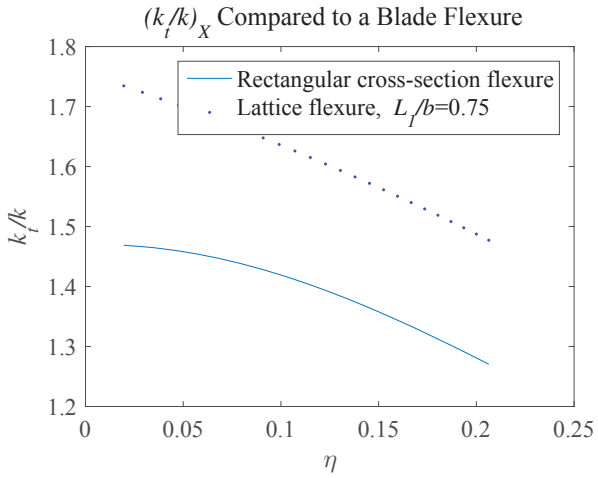
(a) Ratio of torsional stiffness to motion-direction bending stiffness.



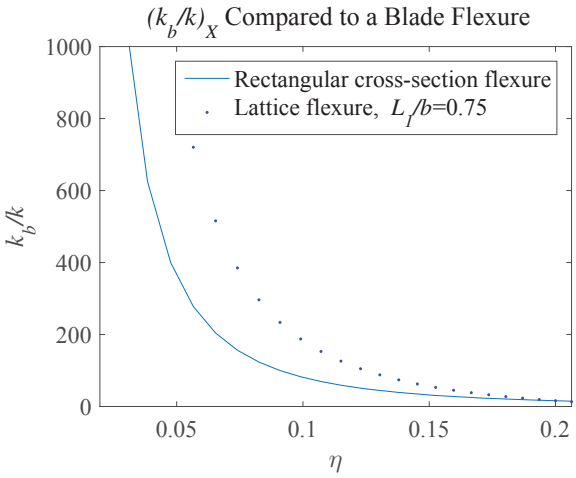
(b) Ratio of support-direction bending stiffness to motion-direction bending stiffness.

Figure 5.7: Off-axis stiffness behavior of the V-type lattice with square lattice elements.

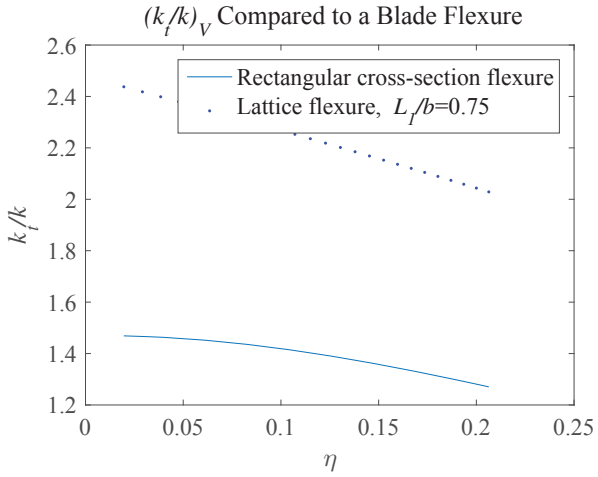
In the case of the V-type lattice flexure, the off-axis stiffness ratio can be increased by a factor of 6.5 over a blade flexure of equal aspect ratio. Other configurations (different L_1/b or non-square lattice elements) may yield even greater improvements. This could help designers achieve higher performance in situations where high support-direction stiffness and low motion-direction stiffness are important.



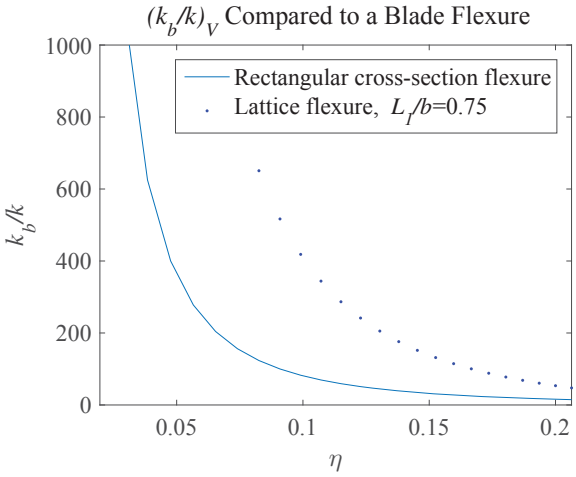
(a) Ratio of torsional stiffness to motion-direction bending stiffness.



(b) Ratio of support-direction bending stiffness to motion-direction bending stiffness.



(c) Ratio of torsional stiffness to motion-direction bending stiffness.



(d) Ratio of support-direction bending stiffness to motion-direction bending stiffness.

Figure 5.8: Off-axis stiffness behavior compared to a blade flexure for a range of η. L_1/b for the lattice flexures is fixed at 0.75.

5.3 Prototype Testing and Performance

This section describes the prototype flexures used to validate the analytic and numeric expressions developed in Section 5.2.

5.3.1 Bending Stiffness

An initial set of prototypes was built from polylactic acid (PLA) filament using a MakerBot® Replicator™ 2 desktop 3D printer. Table 5.2 lists the geometric parameters that were constant for all lattice flexures, while Table 5.3 includes values of L_1/b for each flexure. Cross-axis flexural pivots (CAFPs) were chosen as the test specimens because of their well-understood stiffness behavior and low center shift [40,49,52,53,72,78–81]. To improve flexure quality, the flexure strips were built flat on the build plate with 100% infill and press-fit into base and top sections to create the CAFPs. Representative prototypes are shown in Figure 5.9. Flexure geometries tested included conventional blade flexures, seven X-type lattice flexures and six V-type lattice flexures with different values of N for the same flexure length (listed in Table 5.3.)

Additionally, a pair of cross-axis-flexural pivots was built from 3D printed titanium using the electron beam melting process. These have $L = 2$, $b = 0.46$ in, and $h = 0.04$ in. One flexure was an X-type flexure with $n = 2$ and the other was a V-type flexure with $n = 5$. Figure 5.10 shows the titanium flexures.

To measure the stiffness of each CAFP a rotational displacement was applied using a worm-wheel gearset. Torque was measured using an Omega TQ103 socket torque sensor and displace-

Table 5.1: Comparison of lattice flexure off-axis stiffness to conventional blade flexure with the same value of η. Lattice elements are square with $I_r = I_l$.

Type	L_1/b	η	$\frac{k_t}{k} / \left(\frac{k_t}{k}\right)_{blade}$	$\frac{k_b}{k} / \left(\frac{k_t}{k}\right)_{blade}$
X	0.75	0.02	1.18	3.06
X	0.75	0.10	1.15	2.27
X	0.75	0.20	1.16	1.01
V	0.75	0.02	1.66	6.16
V	0.75	0.10	1.60	5.06
V	0.75	0.20	1.60	3.35
X	1.5	0.02	0.57	2.95
X	1.5	0.10	0.57	1.07
V	1.5	0.02	1.10	6.46
V	1.5	0.10	1.06	5.10
V	1.5	0.20	1.05	3.11

Figure 5.9: Cross-axis flexural pivot prototypes used in the measurement of lattice flexure stiffness.

ment was measured using a US Digital optical encoder with 5000 counts per revolution. By using quadrature counting, a resolution of 0.018° was achieved. Both the torque transducer and the encoder output were read using Labview. Figure 5.11 shows the experimental set-up. By applying rotations of approximately $\pm 2, 3, 4, 5, 6, 7,$ and 8° and recording torque and rotation, the static torque-displacement curves for each flexure were obtained. Matlab's fit function was used to approximate this torque-displacement curve with a linear curve fit (average R^2 value of 0.9912). The slope of this curve fit was recorded as the CAFP bending stiffness k_b'. Table 5.3 summarizes the stiffness results while the reduction in stiffness is shown in Figure 5.5.

The titanium flexures were tested with the same equipment and methods as the PLA flexures. Test results for the titanium prototypes are summarized in Table 5.4.

Table 5.2: Geometric parameters for test specimens. Note that a square cross section was used for both the rails and the diagonal lattice elements, giving $I_r = I_l$.

Parameter	Value
η	0.08
h	1.5 mm
b	17.5 mm
b_l	1.5 mm
L	5.08 cm

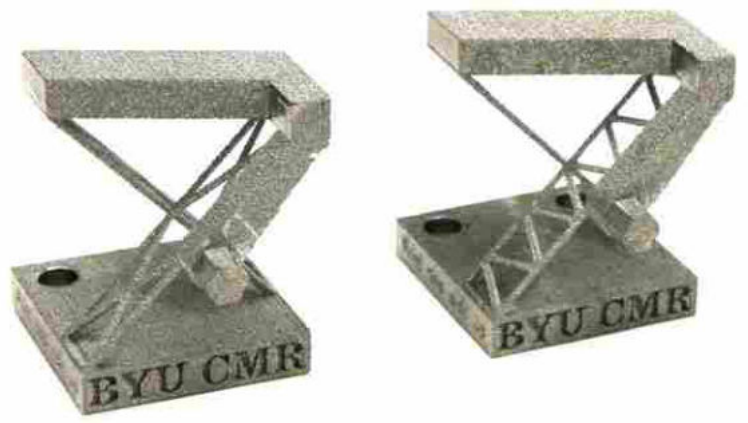


Figure 5.10: Titanium prototypes used in the measurement of lattice flexure stiffness.

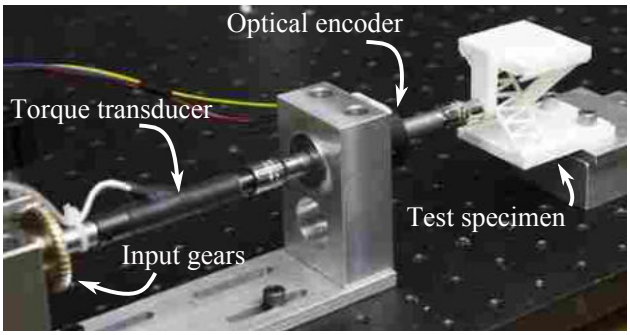


Figure 5.11: Experimental setup for measuring bending stiffness. A rotational displacement was applied using a known torque.

5.3.2 Torsional Stiffness

The same measurement equipment was used to measure the torsional stiffness of each flexure. However, instead of using the flexures in a CAFP, a single flexure was printed and assembled into a fixture that put the flexure in pure torsion. Figure 5.12 shows the test setup. Each flexure was deflected to $\pm 0.5, 1, 1.5, 2, 2.5$ and 3°, with stiffness calculated using a curve fit function as with the bending stiffness. To compare the torsional stiffness of a single flexure with the bending stiffness of a single flexure, the bending stiffness must be extracted from the measured bending stiffness of the CAFPs. Jensen [40] reports the stiffness of a CAFP as $k' = 2.154EI/L$. To extract the stiffness of a single flexure ($k = EI/L$), the measured k' values are divided by 2.154. The measured values of k_t are divided by this resulting stiffness. Thus, k_t/k values reported in Table 5.3 are for a single flexure. Results are presented graphically in Figure 5.13.

Table 5.3: Summary of measured CAFP stiffness and lattice flexure stiffness reduction. k' is the measured stiffness of the CAFP specimen. k_t is the torsional (off-axis) stiffness of a single flexure. k_t/k is computed as $k_t/\frac{k'}{2.154}$ so that only the stiffness of a single flexure is considered. % Reduction is based on a blade flexure with $k' = 0.6689$N-m/rad ($k = 0.3105$N-m/rad). Each design consists of whole unit-cells.

Type	L_1/b	k' (N-m/rad)	% Reduction	k_t (N-m/rad)	k_t/k
Blade	NA	0.6689	NA	0.6816	2.195
X	1.45	0.2478	63%	0.1633	1.4196
X	0.72	0.2443	63.5%	0.1965	1.7320
X	0.48	0.2260	66%	0.2031	1.9349
X	0.36	0.2252	66.5%	0.2412	2.3065
X	0.29	0.2279	66%	0.1708	1.6140
X	0.24	0.2067	69%	0.2869	2.9897
X	0.21	0.1679	75%	0.2477	3.1767
V	0.72	0.1720	74.5%	0.1585	1.9849
V	0.48	0.1943	71%	0.1418	1.5725
V	0.36	0.2017	70%	0.1933	2.1275
V	0.29	0.1733	74%	0.2260	2.8097
V	0.24	0.1848	72.5%	0.2337	2.7242
V	0.21	0.1771	73.5%	0.2177	2.6468

Table 5.4: Summary of measured titanium CAFP stiffness and lattice flexure stiffness reduction. % Reduction is in comparison to a CAFP with blade flexures of the same outer dimensions as the lattice flexures. % Error is in comparison to Equations (5.11) and (5.13); the reported stiffness reduction is greater than expected. This is due to the printed flexure thickness being slightly undersized.

Type	L_1/b	k' (N-m/rad)	% Reduction	% Error
X	1.3096	0.8195	84.7%	19.6%
V	0.5238	0.6126	88.6%	8.2%

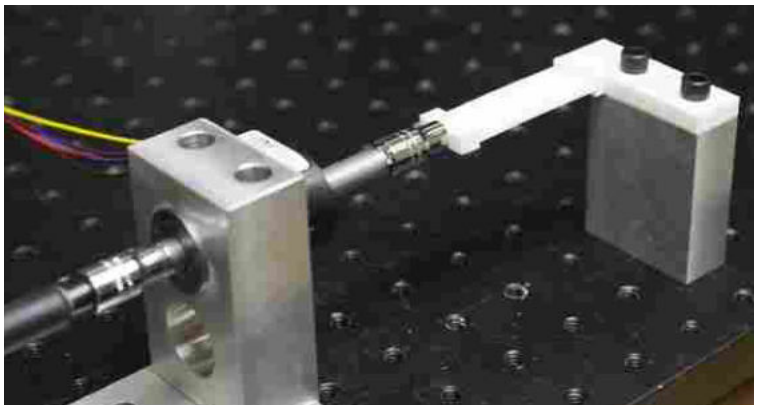


Figure 5.12: Experimental setup for determining torsional stiffness. A rotational displacement was applied co-linear with the axis of a single flexure using a known torque. Note that only the rotation is constrained on the left end of the specimen; displacement along the beam axis is not constrained.

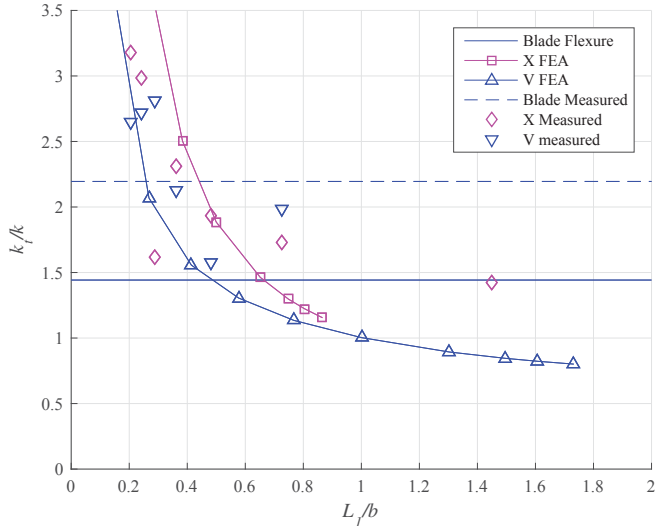


Figure 5.13: Off-axis (torsional) stiffness behavior of both lattice types for $\eta = 0.08$. FEA predictions compared to measured behavior.

5.4 Discussion

The lattice geometries presented here were selected as proof-of-concept geometries; they may not be optimal in terms of maximum stiffness reduction or off-axis stiffness behavior.

It has been shown that these lattice flexures have lower motion-direction stiffness than a conventional blade flexure of similar dimensions and material. The deviation of measured behavior from theoretical is approximately 11%, with measured stiffness being higher than predicted. This

deviation is attributed to lack of precision in the manufacturing process, as well as deviation from the beam model. Thickness of the lattice elements causes them to be joined not at points, but regions. This contributes to error in the analytic and FEA models, as well as stress concentrations. The increase in stress in areas where lattice elements join is small, but whould be accounted for when fatigue life is a concern. Additionally, the reduced motion-direction bending stiffness is accompanied by a reduction in the load-carrying capacity of any joint constructed from lattice flexures. Thus, alttice flexures may not be suited for applications expected to experience high loads.

It has been shown that for some values of L_1/b, both X-type and V-type lattice flexures with square diagonal lattice elements and $I_r = I_l$ exhibit higher transverse bending/motion-direction bending stiffness and torsion/bending stiffness ratios than a conventional blade flexure. This result is especially significant for designs where the off-axis stiffness drives design. For example, suppose a particular mechanism requires a flexure with $k_t = 10$ N-m. Suppose a candidate blade flexure has a k_t/k of 1.4. This results in a bending stiffness of $k = 7.14$ N-m. Now suppose this flexure is replaced with an X-type lattice flexure having an $k_t/k = 2.5$. This flexure can have $k_t = 10$ N-m with $k = 4.0$ N-m. This design change yields a 44% reduction in bending stiffness without any reduction in torsional stiffness.

This chapter has introduced lattice flexures and explored their fundamental characteristics. However, as a new device there remain other aspects that could be further studied and applied.

5.5 Conclusions

The lattice flexure has been introduced as a new flexure type not common before additive manufacturing techniques were widely available. X-type and V-type lattice flexures were presented and analytic models describing their bending stiffness were developed and the results were verified using finite element analysis and physical measurements. Finite element models were used to predict their off-axis stiffness. Lattice flexures can be readily used in engineering applications where their advantages of reduced bending stiffness and higher off-axis stiffness ratios are valuable. Additive manufacturing provides a production method that can provide monolithic lattice flexure systems.

The following conclusions can be made from the results presented in this chapter:

- Lattice flexures have a lower bending stiffness compared to blade flexures of the same size.
- The analytical models developed provide reliable estimates of the lattice flexure stiffness, as verified by FEA and experiment.
- The ratio of support-direction stiffness to motion-direction bending stiffness of some lattice flexures is higher than for blade flexures.
- Arrangements of lattice flexures can be built monolithically using additive manufacturing.

CHAPTER 6. COMPOUND JOINTS: BEHAVIOR AND BENEFITS OF FLEXURE ARRAYS

6.1 Introduction

[1] Compliant mechanisms achieve their motion through deflection of flexible members [1]. They can have significant advantages when compared to traditional mechanisms, including higher precision, reduced friction, reduced wear, lower mass, and lower cost [1, 82]. However, because actuating a compliant mechanism includes loading the flexible members with strain energy, actuation effort can be high [16]. Displacements are limited by the strength of the flexure material [1]. Many types of flexures do not have a constant center of rotation [15, 40, 52, 53, 58, 59, 78], limiting their applications to those where center shift can be neglected or accounted for with software. Finally, off-axis stiffness can be an important aspect of the design, and should be accounted for if off-axis loads are anticipated [77].

Flexures are important elements in many mechanical systems [47, 66, 67]. Different types of flexures have been the focus of recent studies, including but not limited to cross-axis flexural pivots [68], cartwheel flexures [69], and trapezoidal flexures [70]. Methods for modeling and design of flexures include the pseudo-rigid-body model [60, 72], FACT [73], screw theory [74], matrix methods [75], and analytic methods paired with finite element analysis [62]. These methods differ in accuracy and complexity, but all are meant to aid the designer in arriving at a suitable configuration of flexures.

Compound joints are proposed as one way to mitigate several challenges associated with flexures. This chapter introduces compound joints and evaluates their stiffness, displacements [53], off-axis stiffness characteristics, and center shift [53, 83] and compares them to conventional flexures of similar type and size.

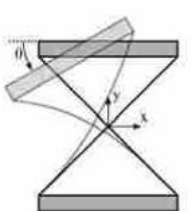
[1] This chapter has been published in *Precision Engineering* with Jason Lund and Larry Howell contributing as co-authors.

(a) An example of a single cross-axis flexural pivot to be arrayed into a compound joint.

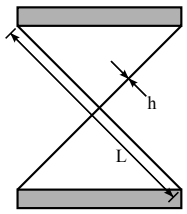
(b) An example of a compound joint with flexures arranged in series.

(c) An example of a compound joint with flexures arranged in series-and-parallel.

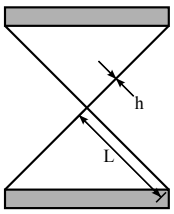
Figure 6.1: Flexure configurations.



(a) Axes used to define rotations and displacements. The positive z-axis is out of the page.



(b) Dimensions used for cross-axis flexural pivots. Note that b is the width out of the page for a single flexure, and the flexible members are not connected where they cross.



(c) Dimensions used for cartwheel hinges. Note that b is width out of the page and that the flexible members join in the center.

Figure 6.2: Variables used in this work.

6.2 Approach

In this work, a compound joint is an arrangement of identical flexures (see Figure 6.1a) along a common axis of rotation [53]. Two configurations are considered: series compound joints (Figure 6.1b) and series-and-parallel compound joints (Figure 6.1c). In such joints, n is the number of flexures in series on one side of the joint. Thus, even though the joint shown in Figure 6.1c consists of 8 flexures, it has $n = 4$. Figure 6.2 shows the coordinate system used in this work. Note that the z-axis is positive out of the page.

See Appendix C for examples of ANSYS and Matlab code used in this work.

6.2.1 Range of Motion

The force (F), stiffness (k), and displacement (x) of a single linear spring are related by $x = F/k$. If two identical springs are placed together in series and a force F is applied, its displacement

is given by $x_{total} = 2F/k = 2x$. The displacement for n springs in series is given by nx. Similarly, for rotational springs, $\theta_{total} = n\theta$. Maximum displacement increases linearly with the number of springs in series. Similarly, compliant flexures in series have the same relationship if they have a constant stiffness and their axes of rotation are co-linear.

6.2.2 Stiffness

The stiffness of two springs in series is given by $k_{eq} = k_1 k_2/(k_1 + k_2)$. If $k_1 = k_2$ this reduces to $k_{eq} = k/2$. For n identical springs in series $k_{eq} = k/n$. Thus, stiffness is reduced as more springs are added in series. This result applies to flexures in series, and is discussed more in Section 6.2.4. In the case of flexures arrayed in parallel, stiffness increases as $k_{eq} = nk$. Although rotational stiffness can change under transverse loads [57], load-dependent stiffness was not addressed in this work.

6.2.3 Center Shift

Center shift is evaluated as in [78]; the trajectory of a single point on the moving block is considered as a function of rotation angle. The point chosen coincides with the initial center of rotation. Minimizing center shift is important for precision mechanisms such as pointers and robots. For example, calculating the inverse kinematics of a robot arm constructed from non-stationary-center joints becomes very difficult. Finally, a load applied through the center of a joint exerts zero moment, but if the center shifts the moment caused by that force may be significant.

Consider the cross-axis flexural pivot (CAFP) [40, 49, 52], shown in Figure 6.3. Several authors have attempted to determine its center shift analytically [52, 53, 78–81]. Although there is not a consensus on the exact trajectory of the center, it is generally agreed that the center shift closely resembles a quadratic curve, and that the rotational center's trajectory is predictable in x and y.

A representative trajectory obtained by finite element analysis (FEA) using ANSYS with BEAM188 elements and an applied rotation is shown in Figure 6.4. The finite element model used to obtain these curves is substantially the same as that used in the off-axis stiffness analysis, and is detailed in Section 6.2.4, with the principal difference being in the loading conditions. Rather

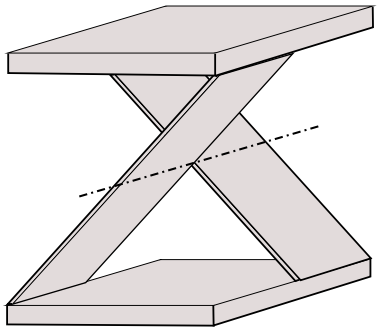
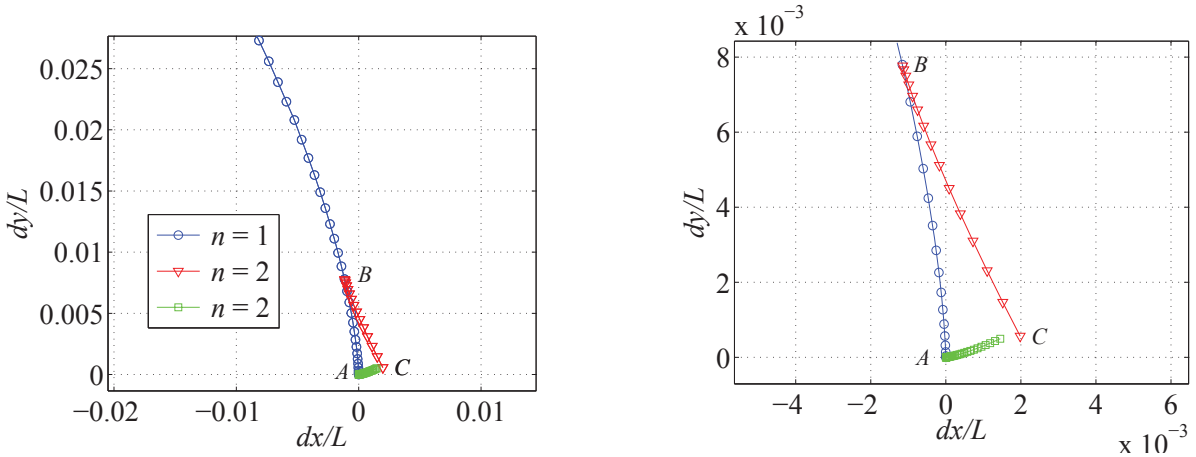


Figure 6.3: An example of a cross-axis flexural pivot.



(a) (b)

Figure 6.4: Normalized center shift behavior for cross-axis flexural pivots (CAFPs) as predicted by FEA. Figure 6.4b is a magnified view of Figure 6.4a. Blue circles show the trajectory of the center of rotation of a single CAFP rotated to 30° at 1° increments. The red triangles predict the trajectory of the center of rotation for a second flexure in series with the first (obtained using a rotated vector sum). Green squares show the trajectory of the center of rotation for two CAFPs arranged in series for the same total angular deflection.

than loading the model with forces in various directions, z-rotation displacements were applied in increments and the displacement of a node rigidly connected to the output and initially coincident with the center of rotation was recorded. This provided the displacement of the center of rotation as a function of angular displacement.

The path traced by the blue circles is the path of the center of rotation for a single CAFP. Point A marks the initial position of the center of rotation. Point B marks the center of rotation at 15° of deflection. If an undeflected CAFP were rotated 180° with its initial center of rotation at

B, point C would be the location of its center of rotation at 15° of deflection. Vector $\overrightarrow{AB}$ describes the center shift of the first CAFP at 15° of deflection. Vector $\overrightarrow{BC}$ describes the center shift of the second CAFP at 15° of deflection. The center shift for two CAFPs (each is deflected 15° for a total of 30°) in series can be found as the vector sum of $\overrightarrow{AB}+\overrightarrow{BC}=\overrightarrow{AC}$. Because the total angular deflection of the joint is distributed evenly between the two flexures, the vectors $\overrightarrow{AB}$ and $\overrightarrow{BC}$ are equal in magnitude. However, $\overrightarrow{BC}$ rotated by an angle $\pi+\theta/2$ (because the second flexure is oriented 180° from the first, which is then deflected an angle $\theta/2$). $\overrightarrow{AC}$ is then given by

$$\overrightarrow{AC}=\begin{bmatrix}\cos(\pi+\theta/2) & -\sin(\pi+\theta/2) & 0\\ \sin(\pi+\theta/2) & \cos(\pi+\theta/2) & 0\\ 0 & 0 & 1\end{bmatrix}\overrightarrow{AB}+\overrightarrow{AB} \tag{6.1}$$

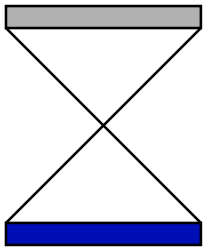
where θ is the total angular deflection of the compound joint (see Figure 6.2). This analysis assumes that the two flexures are co-planar. The green squares in Figure 6.4 show the trajectory of the center of rotation predicted by FEA for a compound joint whose flexures are not co-planar. The green squares are not coincident with point C (see Figure 6.4b) due to lateral flexibility in the joints not accounted for in the vector sum. Thus, the co-planar assumption introduces a small error.

For more than two flexures, Equation (6.1) can be extended to become

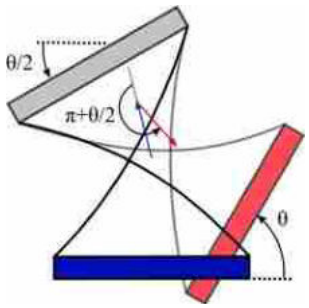
$$\overrightarrow{AC}=\sum_{i=1}^{n}\begin{bmatrix}\cos((i-1)(\pi+\theta/n)) & -\sin((i-1)(\pi+\theta/n)) & 0\\ \sin((i-1)(\pi+\theta/n)) & \cos((i-1)(\pi+\theta/n)) & 0\\ 0 & 0 & 1\end{bmatrix}\overrightarrow{AB} \tag{6.2}$$

which gives a first estimate of center shift. Equation (6.2) also gives a clue to another center-shift-reduction strategy. The term $\pi+\theta/n$ appears because we have so far assumed that flexures are added in series in a co-axial manner, where each flexure is oriented 180° from the previous flexure and θ is the total angular deflection of the joint.

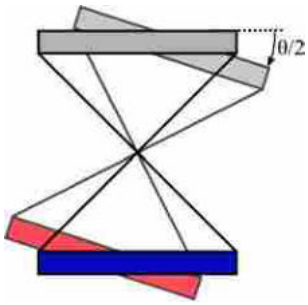
If instead of orienting each flexure 180° from the previous flexure, they were oriented $180°-(\theta/n)°$, the center shift of co-planar flexures would be reduced to zero at displacement θ for $n\in$ even. This is illustrated in Figure 6.5. Figure 6.5a shows a compound joint ($n=2$) where the CAFPs are oriented 180° from each other. When this joint is deflected to an angle θ (Figure 6.5b),



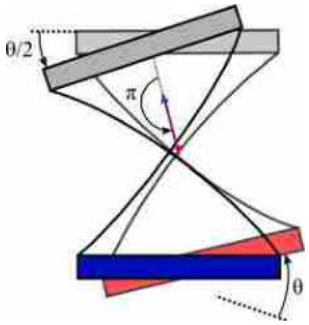
(a) An undeflected compound joint where each CAFP is oriented 180° from the previous CAFP



(b) The compound joint deflected to an angle θ. The center shift is non-zero.



(c) The compound joint redesigned to include an angular offset of $\theta/2$ between the flexures.

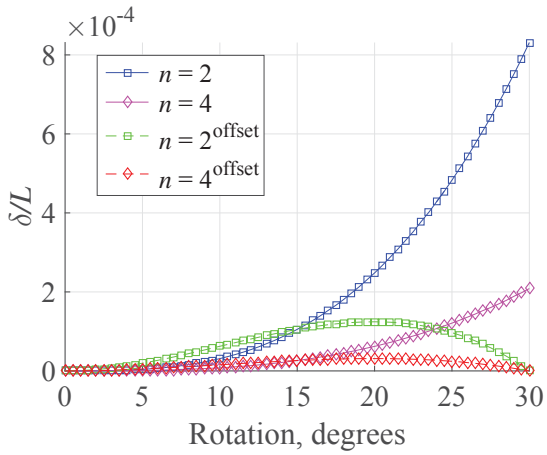


(d) The compound joint from Figure 6.5c, deflected to an angle θ. The center shifts sum to zero at full displacement.

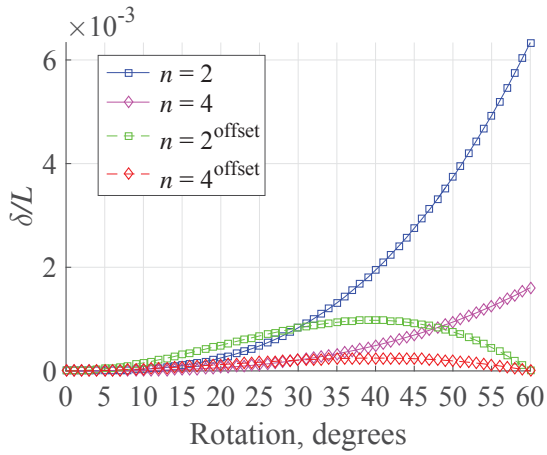
Figure 6.5: A compound joint consisting of cross-axis flexural pivots ($n = 2$). The blue block is grounded and the red block is the output. The gray blocks are the rigid connection between the first and second CAFPs of the joint. The blue and red arrows represent the center shifts of the individual CAFPs (not to scale).

the center shift is non-zero. The blue arrow tracks the change of the center of rotation of the first CAFP and the red arrow tracks the center of rotation of the second CAFP. In contrast, Figure 6.5c shows a compound joint ($n = 2$) where the CAFPs are oriented $\pi - \theta/2$ from each other. When this joint is displaced an angle θ (Figure 6.5d) the center shift is zero (the blue and red arrows are equal in magnitude and opposite in direction).

This behavior can be confirmed with finite element analysis. Figure 6.6 shows examples of cartwheel flexures arranged to have reduced center shift.



(a) Center shift of four different configurations of compound flexures deflected to 30°.



(b) Center shift of four different configurations of compound flexures deflected to 60°.

Figure 6.6: By choosing an appropriate angular offset for compound joints, center shift can be dramatically reduced at specific angles of deflection.

6.2.4 Off-Axis Stiffness

Off-axis stiffness is the stiffness of the joint in support directions when subjected to loads or moments in those directions. In this work, off-axis stiffness is evaluated using non-dimensional stiffness [49, 52]. This allows flexures of different sizes and topologies to be rapidly compared. We define the non-dimensional linear stiffness as

$$\kappa_i = \frac{F_i L^3}{\delta_i E I} \tag{6.3}$$

and the non-dimensional torsional stiffness as

$$\kappa_{\theta i} = \frac{M_i L}{\theta_i E I} \tag{6.4}$$

where F is applied force, L is a characteristic flexure length (which will be different for each flexure topology), E is the modulus of elasticity, I is the second moment of area of the flexure elements, δ is the deflection when subjected to force F, M is the applied moment, and θ is the rotational displacement when subjected to moment M. A θ in the subscript of κ indicates a rotational stiffness, while no subscript indicates a linear stiffness. The i in a subscript can be x, y, or z to denote action

along or about the indicated axis. For example, a cantilever beam with a force at the free end would have $\kappa_y = 3$ and a cantilever beam with a moment at the free end would have $\kappa_{\theta z} = 1$ [64].

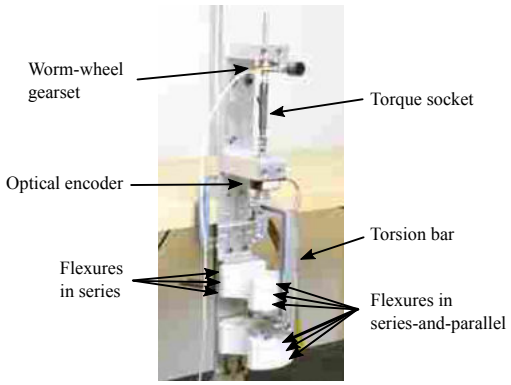
To evaluate off-axis stiffness, cross-axis flexural pivots and cartwheel flexures were analyzed using finite element software (ANSYS). Flexures were modeled with BEAM188 elements, which are capable of 3D and nonlinear analysis. For all flexures in this analysis, the slenderness ratio GAL^2/EI was calculated to be 2953, which is much greater than the recommended minimum value of 30. Each flexure was meshed with fifty elements along its length to ensure a mash-independent solution. A range of joint configurations was analyzed by varying the aspect ratio of the flexures (their width-to-thickness ratio b/h) and the number of flexures in each compound joint. A force or moment was applied along a single axis, the displacement along that axis was recorded, and then the force or moment was removed and applied to the next axis. In this way each geometry was rapidly analyzed for all six degrees of freedom. See Appendix C for APDL scripts of these analyses.

6.3 Experimental Setup

Two aspects of joint performance were subjected to physical testing: motion-direction stiffness and center shift. The procedures used in these measurements are outlined in the sections below.

6.3.1 Stiffness

To confirm the modeled results experimentally, a modular joint was built that could be configured as a series or series-and-parallel compound joint using cartwheel flexures (shown in Figure 6.7a). The flexures were made of 12.7 mm (0.5 in) thick polypropylene, with $L = 44.5$ mm (1.75 in), $h = 2.0$ mm (0.08 in), and $b = 12.7$ mm (0.5 in). This setup is shown in Figure 6.7b. To actuate the joint, a worm-wheel gearset applied a rotational displacement. The resulting reaction torque and displacement were measured by an Omega TQ103 torque socket in series with a US Digital optical encoder with 5000 counts per revolution. An angular resolution of 0.018 degrees was achieved by employing quadrature encoding. All instruments were read using Labview. The compound joint was configured for $n \in \{1,2,3,4\}$. The torque and displacement were recorded



(a) One of the cartwheel flexures that can be used to assemble the modular joint for testing.

(b) Experimental setup for determining κ_{θ_z} of cartwheel hinges in series-and-parallel with $n = 3$.

Figure 6.7: Components used in the experimental validation

for rotations about the z axis. To avoid the effects of gravity on the L-shaped torsion bar, this measurement was made with a vertical axis of rotation. The stiffness was obtained by taking a linear curve fit of the resulting torque-displacement data.

6.3.2 Center Shift

The center shift behavior was measured using a digital microscope and Matlab's Computer Vision System Toolbox. First the flexure to be measured was clamped on a workbench and gravity compensated to avoid any sagging. A flat panel with a random, high-contrast pattern and known grid (to supply an accurate scale) was attached to the flexure output. The digital microscope was positioned over the center of the flexure. The flexure was filmed as a deflection was applied and then slowly removed. A Matlab script extracted features common to sequential frames and calculated the rotation and translation between one frame and the next. From this data, and the initial center point of the flexure, center shift was obtained. A photo of the experimental setup is shown in Figure 6.8. Center shift was measured for a single cartwheel hinge deflecting to about 60°, and for compound joints with $n = 2$ with offsets of 0°, 15°, and 30°.

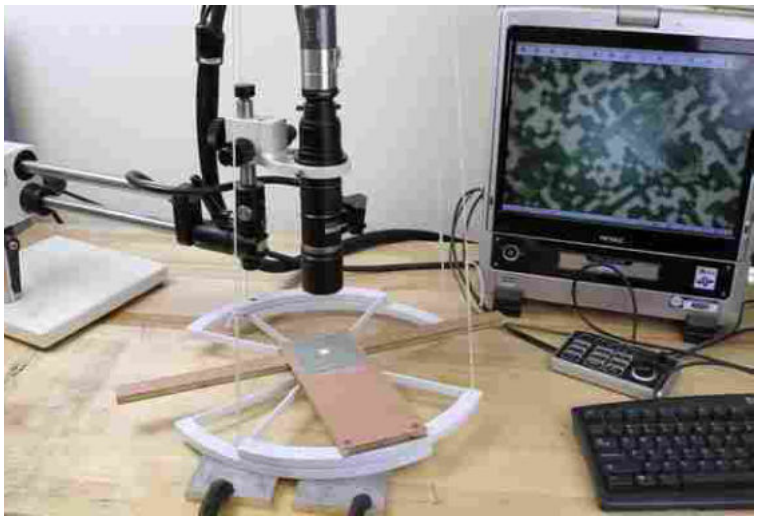


Figure 6.8: Test setup for measuring the center shift of various configurations of flexures.

6.4 Results

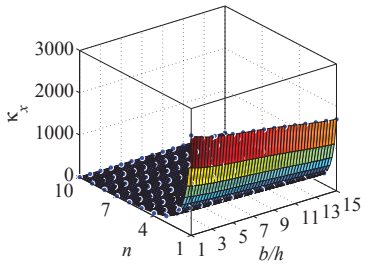
6.4.1 Stiffness and Off-Axis Stiffness

Figure 6.9 shows the stiffness data for cartwheel hinges in series and Figure 6.10 shows the stiffness data for cross-axis flexural pivots in series as a function of b/h (as defined in Figure 6.2) and n. The data were approximated with a surface-fit using Matlab's fit function with a custom-specified fit type. See Appendix C for the Matlab scripts used. Data for cartwheel hinges and cross-axis flexural pivots in series-and-parallel are shown in Figure 6.11 and 6.12, respectively. The equation for the surface fit is

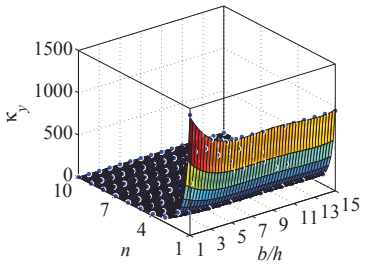
$$\kappa = (A(b/h)^4 + B(b/h)^3 + C(b/h)^2 + D(b/h) + E)n^{F(b/h)+G} \tag{6.5}$$

where the coefficients A, B, C, D, E, F, and G are listed in Table 6.1, along with the R^2 value for each surface fit.

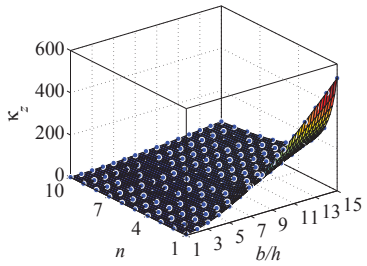
After measuring the motion-direction (z-axis rotation) stiffness of the physical prototype, its non-dimensional z-rotation stiffness was compared to values predicted by the surface fits given in Table 6.1. As can be seen in Figure 6.13, there is agreement between the finite element results, the surface fit predictions, and the measured stiffness.



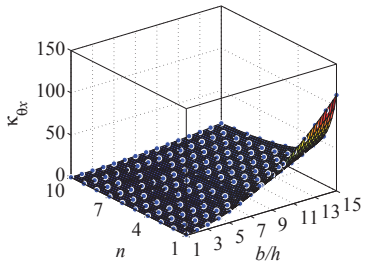
(a) κ_x as a function of n and b/h.



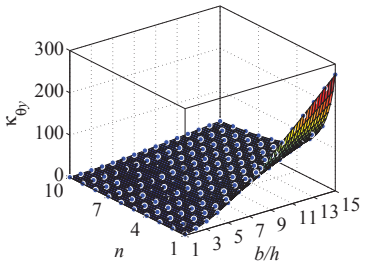
(b) κ_y as a function of n and b/h.



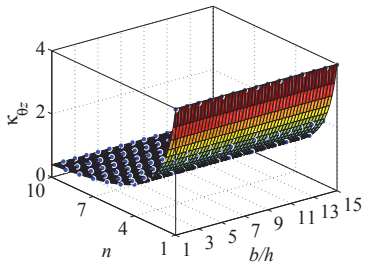
(c) κ_z as a function of n and b/h.



(d) $\kappa_{\theta x}$ as a function of n and b/h.



(e) $\kappa_{\theta y}$ as a function of n and b/h.



(f) $\kappa_{\theta z}$ as a function of n and b/h.

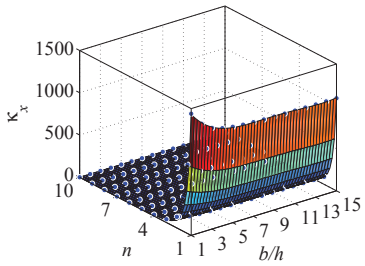
Figure 6.9: Non-dimensional stiffness behavior of cartwheel hinges in series.

Achieving accurate measurements of the off-axis stiffness in the remaining five degrees of freedom (κ_x, κ_y, κ_z, $\kappa_{\theta x}$, and $\kappa_{\theta y}$) is difficult [69] due to the small displacements generally involved with off-axis deflections. Attempts were made to measure rotational stiffness about the x- and y-axes, but these were unsuccessful because of relative motion between the assembled parts in the modular joint. Therefore, other methods of validating the FEA model were sought.

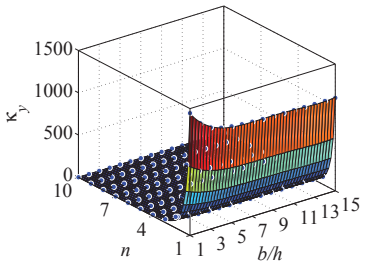
To validate the FEA model, the analytic results of Kang and Gweon [69] for a single cartwheel hinge were confirmed using the same finite element model used in the generation of the surface fits presented in Table 6.1. Close agreement between our FEA model and the analytic result of [69] for a single flexure was achieved, as seen in Table 6.2. This close agreement, combined with the agreement of $\kappa_{\theta z}$ with FEA indicate that the surface fits presented here are appropriate for $1 \leq n \leq 10$ and $1 \leq b/h \leq 15$.

6.4.2 Center Shift

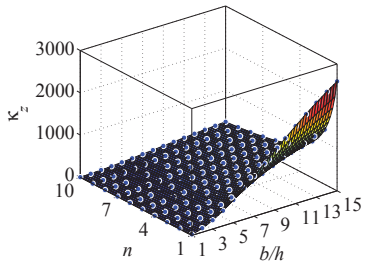
Unlike stiffness and off-axis stiffness, center shift is independent of flexure aspect ratio b/h (assuming slender beams); it is a function only of the flexure type. Figures 6.14 and 6.15 depict



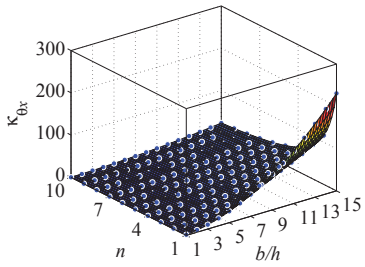
(a) κ_x as a function of n and b/h.



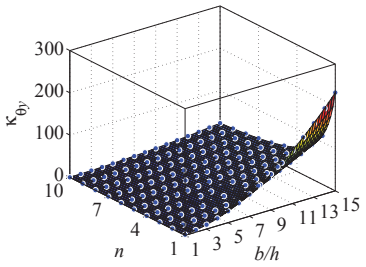
(b) κ_y as a function of n and b/h.



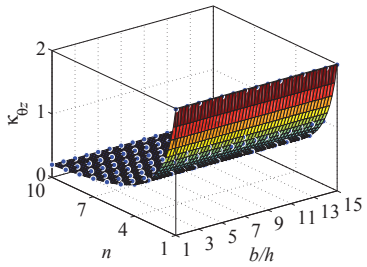
(c) κ_z as a function of n and b/h.



(d) $\kappa_{\theta x}$ as a function of n and b/h.



(e) $\kappa_{\theta y}$ as a function of n and b/h.



(f) $\kappa_{\theta z}$ as a function of n and b/h.

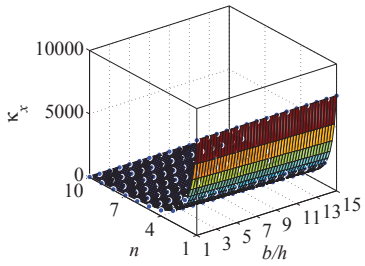
Figure 6.10: Non-dimensional stiffness behavior of cross-axis flexural pivots in series.

the trajectory and vector sum of center shift normalized by the characteristic length of the flexure, L, defined in Figure 6.2. For a cartwheel hinge, L is the length of a single spoke of the flexure, while for a CAFP L is the length of a single blade flexure. These data were obtained using the finite element model described in Sections 6.2.3 and 6.2.4.

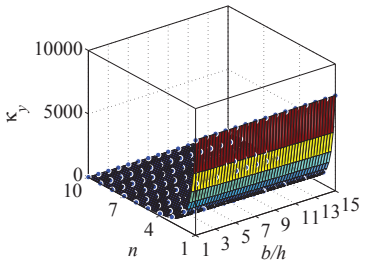
To confirm FEA predictions of center shift, data was gathered using the procedure outlined in Section6.3.2. This data is presented in Figure 6.16. Notice that for $n = 1$ there is very little disagreement between the FEA and measured center shift.

6.5 Example Mechanism

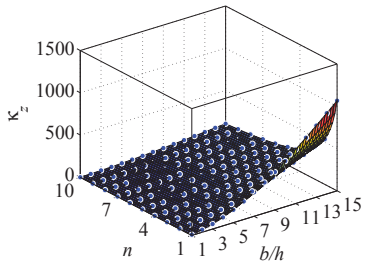
The mechanism shown in Figure 6.17a is a fully compliant 3D-printed titanium pointing mechanism [10]. It incorporates six cross-axis flexural pivots and a split-tube flexure to provide two degrees-of-freedom and $\pm 15^\circ$ of motion about the two axes. The output is mounted on a stage supported by two series-and-parallel compound joints with $n = 1$ in series. This arrangement allows the output stage to rotate around two orthogonal axes with high stability under load. This extra stability theoretically allows the mechanism to function under a 445 N (100 lb) load.



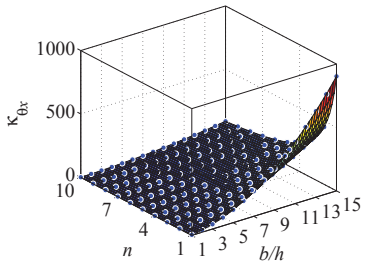
(a) κ_x as a function of n and b/h.



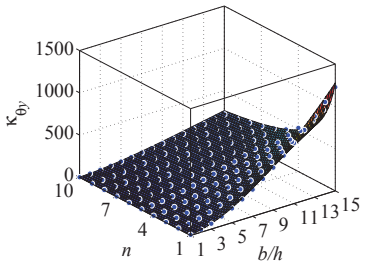
(b) κ_y as a function of n and b/h.



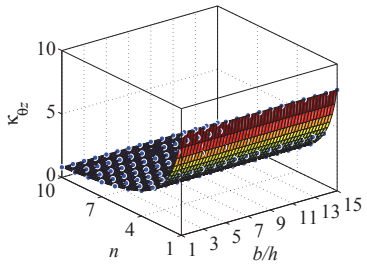
(c) κ_z as a function of n and b/h.



(d) $\kappa_{\theta x}$ as a function of n and b/h.



(e) $\kappa_{\theta y}$ as a function of n and b/h.



(f) $\kappa_{\theta z}$ as a function of n and b/h.

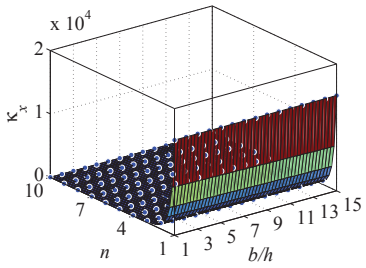
Figure 6.11: Non-dimensional stiffness behavior of cartwheel hinges in series-and-parallel.

To achieve a greater range of motion this pointer could be redesigned with series-and-parallel joints with $n = 2$, and the other CAFPs replaced with series compound joints, also of $n = 2$. If greater precision is sought, an angular offset could be incorporated. Increasing n further would continue to increase its range of motion, but complexity would greatly increase as well. From the equations presented in Table 6.1, it can be predicted that changing from $n = 1$ to $n = 2$ will reduce κ_x and κ_y by approximately a factor of 20. A version of what this might look like is shown in Figure 6.17b.

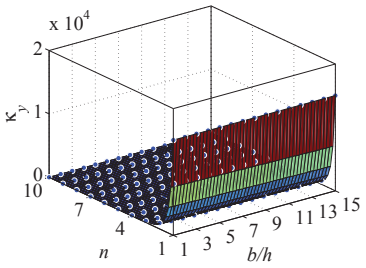
6.6 Discussion

6.6.1 Stiffness

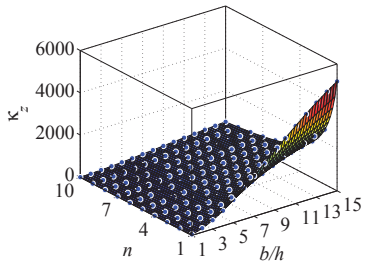
Comparison of Figure 6.9 with Figure 6.11 and comparison of Figure 6.10 with Figure 6.12 illustrate that adding flexures in parallel increases the stiffness and off-axis stiffness of compound joints. Stiffness increased by a factor of 2-8, depending on flexure geometry and which stiffness is



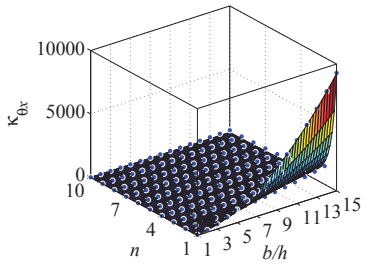
(a) κ_x as a function of n and b/h.



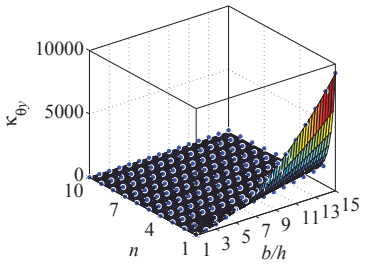
(b) κ_y as a function of n and b/h.



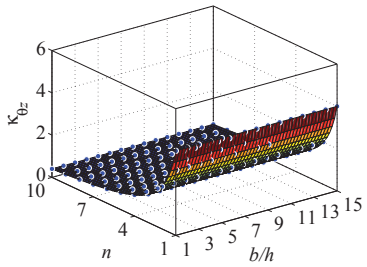
(c) κ_z as a function of n and b/h.



(d) $\kappa_{\theta x}$ as a function of n and b/h.



(e) $\kappa_{\theta y}$ as a function of n and b/h.



(f) $\kappa_{\theta z}$ as a function of n and b/h.

Figure 6.12: Non-dimensional stiffness behavior of cross-axis flexural pivots in series-and-parallel.

being considered. Although the figures show $1 \leq n \leq 10$, a proposed practical limit on n is $n \leq 4$. Past this point the joint grows ever larger without an appreciable increase in performance.

It can be seen from Figures 6.9, 6.10, 6.11, and 6.12 that κ_z, $\kappa_{\theta x}$ and $\kappa_{\theta y}$ increase as b/h increases. κ_x, κ_y and $\kappa_{\theta z}$ are generally independent of b/h.

As an example, consider a single cartwheel flexure with a maximum deflection Θ to a compound joint of identical cartwheel flexures in series with $n = 2$. The maximum deflection of the compound joint is 2Θ. Stiffness along all axes drops significantly — by 50% in most cases. Depending on the design constraints this may be unacceptable. However, if flexures are added in parallel, stiffness values return to their original (single flexure) value, with $\kappa_{\theta x}$ and $\kappa_{\theta y}$ improved by factors of 4 and 3, respectively. Thus, range of motion has been doubled with no appreciable loss in off-axis stiffness, and the desired stiffness $\kappa_{\theta z}$ remains unchanged.

Now compare a cartwheel flexure in series where $n = 2$ with a cartwheel flexure in series-and-parallel where $n = 4$. For these two compound joints, κ_x, κ_y, κ_z, and $\kappa_{\theta z}$ will be unchanged, but the off-axis stiffnesses $\kappa_{\theta x}$ and $\kappa_{\theta y}$ will be approximately four times higher with the parallel

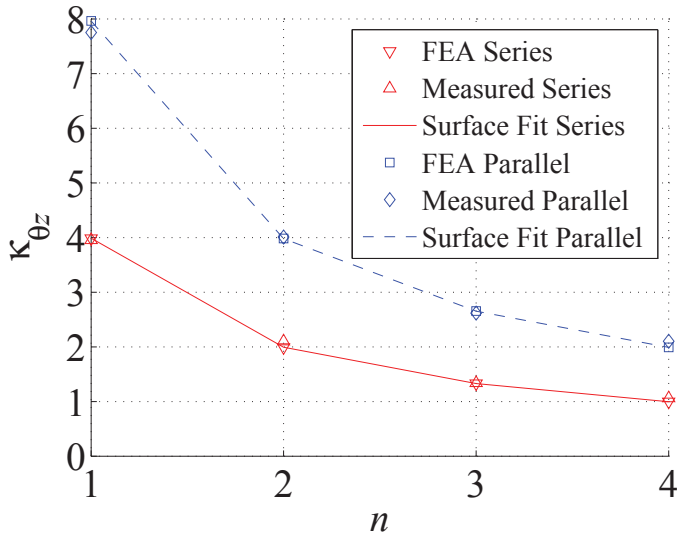


Figure 6.13: A comparison of FEA, surface fit predictions, and measured stiffness behavior for cartwheel flexures in series-and-parallel with $b/h = 5.05$.

configuration. Additionally, the range of motion of the series-and-parallel configuration will be double that of the series configuration.

6.6.2 Center Shift

As n increases, center shift generally decreases due to two complementary factors. First, center shift is roughly a quadratic function of rotation. If the same rotation is divided between more joints, the sum of the center shift will be less ($x^2 > n(x/n)^2 = x^2/n$).

Second, the total center shift of a compound flexure is the vector sum of the center shifts of the individual flexures. When n is even the center shift reduction is more dramatic. For $n = 2$ the relatively large center shift of the first flexure is counteracted by the equal and nearly opposite center shift of the second flexure. An approximately order-of-magnitude reduction in center shift can be achieved by using $n = 3$, while using $n = 2i$, where i is a non-zero positive integer, reduces center shift by two orders of magnitude or more. As shown in Section 6.2.3, if the flexures are properly arranged the theoretical center shift is zero for a given rotation.

From Figures 6.14 and 6.15 it can be seen that center shift of a series compound joint will be approximately equal to that of a series-and-parallel joint for equal n. This is because the

flexures added in parallel have the same center shift behavior as the original flexures in series. That is, flexures added in parallel do nothing to alter the vector sum that governs center shift.

Figure 6.16 shows measured center shift data for several configurations of a cartwheel hinge series joint. While not all data matches perfectly with what would be expected from FEA, the general trend of reduced center shift with $n = 2$ is reinforced.

Table 6.1: Coefficients for surface fits for non-dimensional stiffness of compound joints. The equation takes the form $\kappa = (A(b/h)^4 + B(b/h)^3 + C(b/h)^2 + D(b/h) + E)n^{F(b/h)+G}$. R^2 values are > 0.988 for all surface fits.

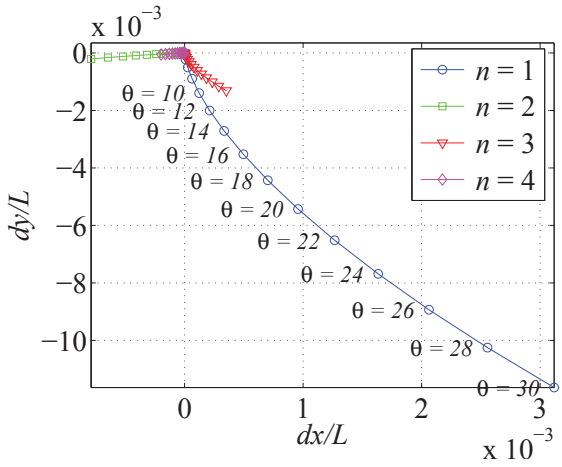
	A	B	C	D	E	F	G
Cartwheel hinge in series							
κ_x	0.05331	-1.916	23.74	-157.7	2489	-0.01846	-2.437
κ_y	0.1221	-4.645	62.86	-359.3	1699	-0.006181	-2.936
κ_z	-0.001132	-0.0342	3.098	0.5135	0.8849	0	-1
$\kappa_{\theta x}$	0	0.003429	0.4524	0.2865	0.1134	0	-1
$\kappa_{\theta y}$	0.002091	-0.1179	2.592	-1.134	1.095	0	-1
$\kappa_{\theta z}$	0	0.000192	-0.00242	0.01205	3.964	0.0001123	-1.001
Cartwheel hinge in series-and-parallel							
κ_x	0.009293	-0.3172	3.517	-15.16	7549	-0.02463	-1.872
κ_y	0.01332	-0.5096	6.893	-36.63	7565	-0.0233	-2.737
κ_z	-0.002264	-0.06839	6.196	1.027	1.77	0	-1
$\kappa_{\theta x}$	-0.0003471	0.01537	3.786	1.139	0.3603	-0.001232	0.9802
$\kappa_{\theta y}$	0.004218	-0.26	8.846	-3.472	3.272	-0.009498	-0.545
$\kappa_{\theta z}$	0	0	-0.0003177	0.001591	7.961	0	-1
Cross-axis flexural pivots in series							
κ_x	0.07014	-2.743	38.64	-234.3	1629	-0.01401	-3.444
κ_y	0.07092	-2.772	39.02	-236.4	1638	-0.01312	-3.459
κ_z	0.03476	-1.938	34.19	-22.56	15.55	0	-1
$\kappa_{\theta x}$	-0.000193	0.007113	0.922	0.5791	0.2221	0	-1.001
$\kappa_{\theta y}$	-0.0002213	0.008256	0.919	0.5862	0.2151	-0.0004537	-1.001
$\kappa_{\theta z}$	0	0	-0.000322	0.00167	1.987	0	-1
Cross-axis flexural pivots in series-and-parallel							
κ_x	0.004806	-0.1831	2.451	-12.91	15040	-0.03316	-4.259
κ_y	0.006235	-0.2228	2.81	-14.09	15040	-0.03313	-4.259
κ_z	0.06812	-3.841	68.1	-43.95	30.05	0	-1
$\kappa_{\theta x}$	-0.0007577	0.03358	40.66	2.371	0.8576	-0.0033	-2.331
$\kappa_{\theta y}$	-0.001892	0.06504	40.38	3.309	0.02026	-0.003264	-2.332
$\kappa_{\theta z}$	0	0	-0.0002778	0.001087	3.999	0	-1

6.7 Conclusions

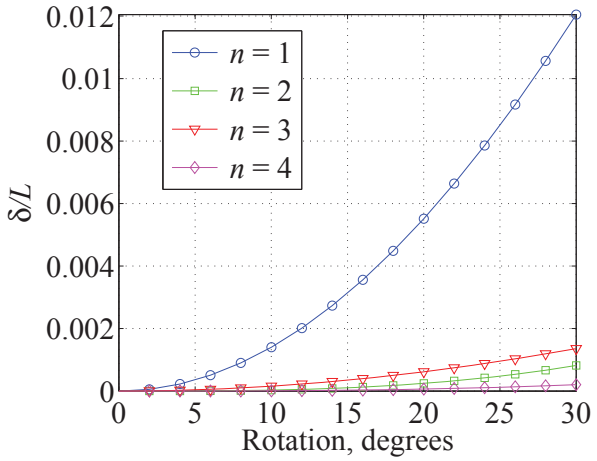
It has been shown that compound joints offer increased range of motion and reduced center shift compared to single flexures. Off-axis stiffness can be maintained while improving desired stiffness and joint range-of-motion. Compound joints offer performance advantages with increased off-axis stiffness, greater range of motion, and potentially zero center shift. However, compound joints have a larger envelope than a single flexure, are more complex, and the theoretically zero center shift only occurs at certain deflections.

Table 6.2: A comparison of the FEA model used for a single cartwheel flexure compared to analytical results obtained by Kang and Gweon [69]. Because the values of b/h used by Kang and Gweon are outside the range of the surface fits, the finite element model was re-run using the material properties and dimensions reported in [69]. Finally, the coordinate system used in [69] switched the x and y axes from what is defined in Figure 6.2 and the point at which the forces are applied is slightly different (which change was reflected in the FE model).

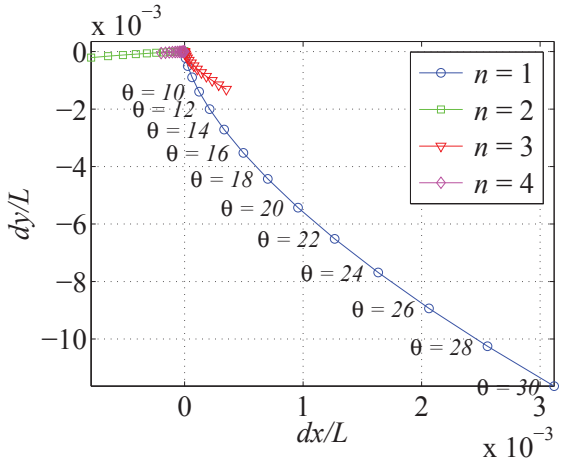
Quantity	Analytic [69]	Non-dimensional	FEA	% Difference
Set 1: $L = 15$ mm, $b = 10$ mm, $h = 0.4$ mm				
κ_x	9073 N/m	7.97	7.98	0.04%
κ_y	9.607×10^6 N/m	8443.7	8443.5	0.00%
κ_z	0.491×10^6 N/m	431.54	433.46	0.44%
$\kappa_{\theta x}$	80.19 N-m/rad	313.23	313.28	0.02%
$\kappa_{\theta y}$	179.27 N-m/rad	700.27	713.88	1.91%
$\kappa_{\theta z}$	1.02 N-m/rad	3.99	3.99	0.01%
Set 2: $L = 12$ mm, $b = 5$ mm, $h = 0.3$ mm				
κ_x	3740 N/m	7.98	7.98	0.02%
κ_y	4.503×10^6 N/m	9606.4	9605.98	0.00%
κ_z	95×10^3 N/m	202.67	202.76	0.05%
$\kappa_{\theta x}$	9.42 N-m/rad	139.61	139.67	0.04%
$\kappa_{\theta y}$	27.5 N-m/rad	407.47	413.19	1.39%
$\kappa_{\theta z}$	0.27 N-m/rad	3.99	3.99	0.21%
Set 3: $L = 8$ mm, $b = 10$ mm, $h = 0.2$ mm				
κ_x	7479 N/m	7.98	7.98	0.03%
κ_y	9.006×10^6 N/m	9606.4	9606.0	0.00%
κ_z	1.340×10^6 N/m	1429.3	1449.1	1.36%
$\kappa_{\theta x}$	75.04 N-m/rad	1250.7	1250.8	0.00%
$\kappa_{\theta y}$	112.54 N-m/rad	1875.7	1907.2	1.65%
$\kappa_{\theta z}$	0.24 N-m/rad	4.00	3.99	-0.16%



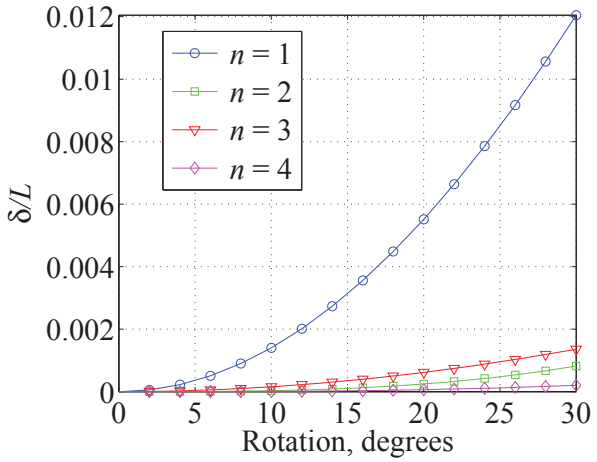
(a) Rotational center trajectory of a series joint.



(b) Vector sum of center shift of a series joint.



(c) Rotational center trajectory of a series-and-parallel joint.

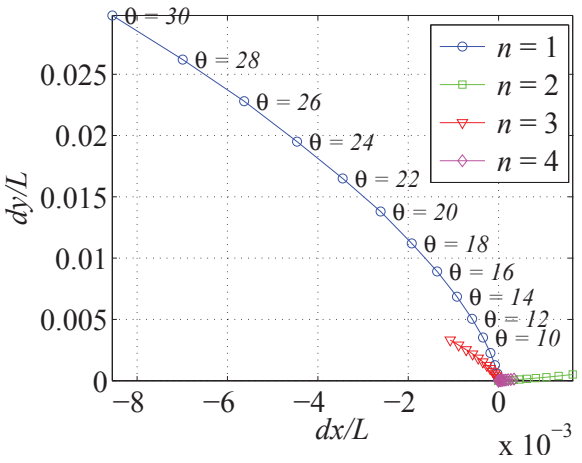


(d) Vector sum of center shift of a series-and-parallel joint.

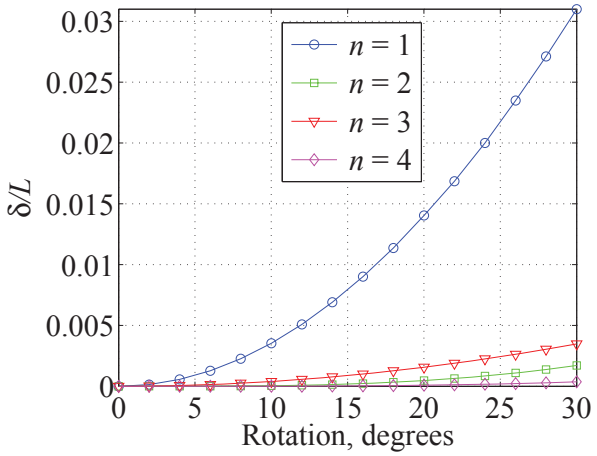
Figure 6.14: Center shift behavior for cartwheel hinges in series (Figures 6.14a and 6.14b) and series-and-parallel (Figures 6.14c and 6.14d). Note that the two cases are nearly identical.

The contributions of this chapter include:

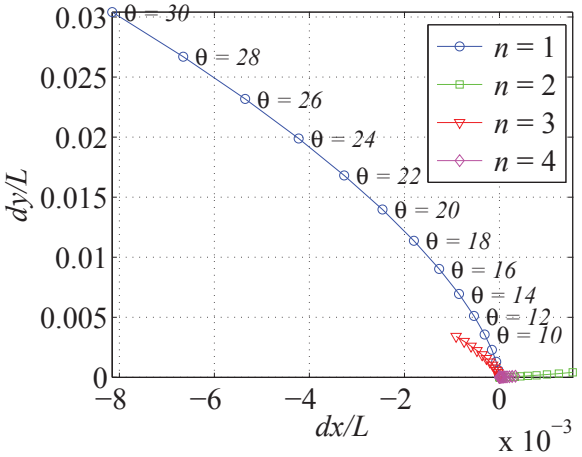
- Demonstrating the intuitive advantage of increased range of motion and increased off-axis stiffness
- Identifying the drawbacks of larger flexure envelope and increased complexity
- Creating models to describe compound joints' behavior
- Verifying models numerically and experimentally (where possible)
- Demonstrating the practical limit of $n \leq 4$



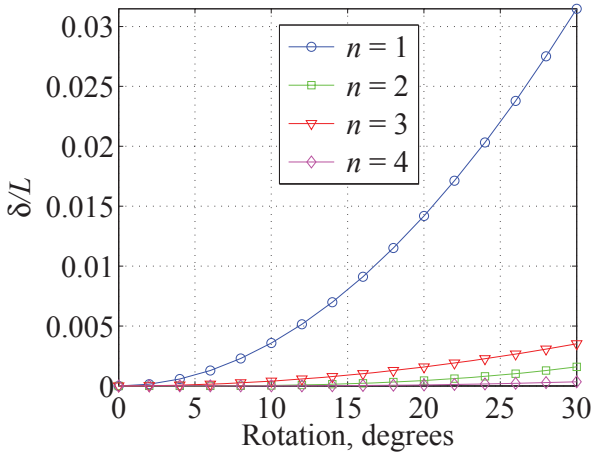
(a) Rotational center trajectory of a series joint.



(b) Vector sum of center shift of a series joint.



(c) Rotational center trajectory of a series-and-parallel joint.



(d) Vector sum of center shift of a series-and-parallel joint.

Figure 6.15: Center shift behavior for cross-axis flexural pivots in series (Figures 6.15a and 6.15d) and series-and-parallel (Figures 6.15c and 6.15d). Note that the two cases are nearly identical.

- Developing methods to predict and measure center shift
- Describing a method to design joints with zero center shift at a specified deflection

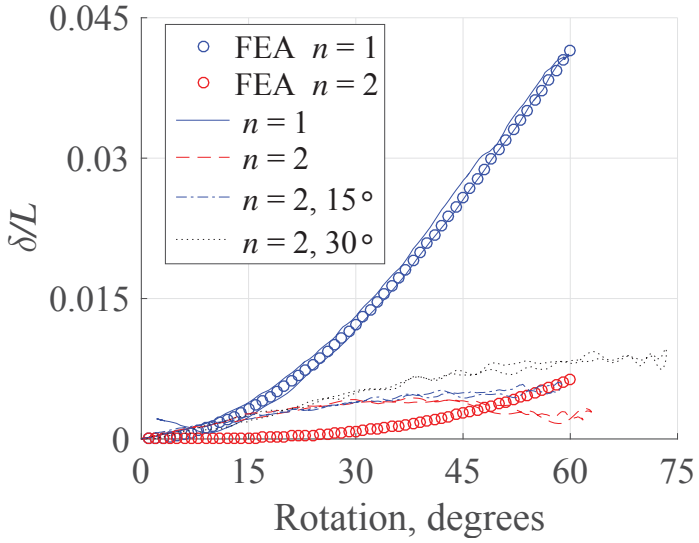
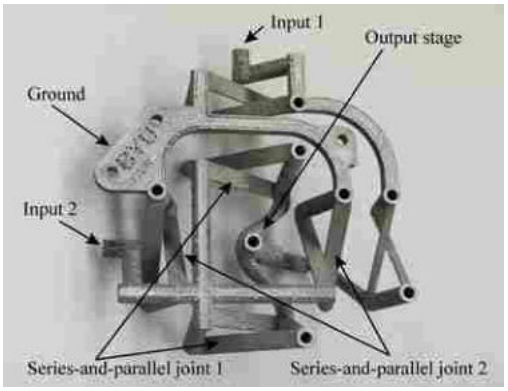


Figure 6.16: Measured center shift data for a cartwheel hinge, compared with FEA predictions. FEA predictions are for $n = 1$ and $n = 2$ with no angular offset, and experimental data is shown for $n = 1$, $n = 2$ (no offset), $n = 2$ (15° offset) and $n = 2$ (30° offset).



(a) A two-degree-of-freedom pointer mechanism that incorporates two compound joints of $n = 1$.

(b) The same mechanism, redesigned with $n = 2$ for joints along one axis. Also incorporated is an angular offset to reduce center shift to near zero at 15° rotation.

Figure 6.17: A pointer mechanism incorporating compound joints.

CHAPTER 7. INTEGRATION OF ADVANCED STIFFNESS-REDUCTION TECHNIQUES DEMONSTRATED IN A 3D-PRINTABLE JOINT

7.1 Background

[1] Static balancing is a method whereby the required actuation effort of a joint is decreased [17, 18, 23, 26, 41, 42, 46]. This is usually done through the addition of springs or auxiliary bodies that function to store and release energy in a manner opposite that of the target joint [18, 41, 44, 84–86]. This results in a small net input of energy to the joint during actuation [21,23,26,42,43,47,86]. One challenge to balancer design is the pre-stressing required [86,87]. Creep and stress relaxation resulting from this pre-stressing negatively effect mechanism performance [1]. Joints of lower initial stiffness generally require less pre-stressing of the balancer, making the system easier to design.

The recently introduced lattice flexure [88] (see Figure 7.1) has drastically reduced stiffness (60-80% lower) compared to a blade flexure of the same material and outer dimensions. It will be shown here that using compound lattice flexures in concert with static balancing drastically reduces actuation effort.

The development of 3D printing technology has made many advances in recent years, but obtaining useful motion from 3D-printed mechanisms is difficult without incorporating flexures. The minimum feature size of most 3D printers puts a lower-bound constraint on flexure thickness that results in high mechanism stiffness. By applying stiffness reduction strategies to 3D printing, it is demonstrated that a 3D-printed titanium part can have low stiffness.

To statically balance a flexure using the non-dimensional approach of Merriam et al., a balancer spring is required [57]. If this joint is to be 3D-printed, its balancer spring must be 3D-printable. Previous balancer designs have used coil springs or leaf springs [3, 57]. However, coil springs do not lend themselves to 3D printing, and leaf springs lack the pre-load and stiffness

[1] This chapter has been submitted for publication to *Mechanism and Machine Theory* with Kyler Tolman and Larry Howell contributing as co-authors.

(a) Front view of cross-axis flexural pivot (CAFP) made with lattice flexures.

(b) Side view of lattice-flexured CAFP.

Figure 7.1: An example lattice flexure printed in titanium.

behavior required by the balancing method used here. In this work a fully printable balancer will be presented.

The presence of the balancer spring introduces another challenge; it exerts a compressive load on the flexure to be balanced. Moreover, if this compressive load is not applied through the center of the joint it will result in an off-axis torque on the joint. To remedy this, a compound joint will be employed [89]. By arranging two flexures in parallel with the balancer spring between them, the stability of the joint is improved and the load from the balancer spring does not exert an unbalanced torque on the flexures.

Figure 7.2 shows the statically balanced 3D-printed lattice-flexured cross-axis flexural pivot developed here. Each element of the mechanism is discussed herein. An overview of lattice flexures is presented first, along with an evaluation of their load-dependent stiffness that allow them to be statically balanced. Next static balancing is discussed, followed by a detailed discussion on the design of the nonlinear tension spring used as a balancer. Finally, integration of these elements is discussed and experimental results are presented for a titanium prototype.

The objectives of this work are:

1. Evaluate the load-dependent stiffness behavior of lattice-flexured CAFPS.

2. Design a printable balancing spring.

Figure 7.2: Printed titanium flexure with its parts labeled.

3. Design a statically balanced CAFP incorporating compound lattice flexures.

4. Validate the resulting design with measurements of a physical prototype.

7.2 Lattice Flexures

7.2.1 Overview

Lattice flexures were recently introduced as a new flexure geometry [88]. They are characterized by reduced motion-direction stiffness and improved off-axis stiffness behavior compared to blade flexures. This is achieved by the removal of material in a perforated lattice pattern. The reduction in stiffness is predictable using an analytic model, and off-axis stiffness behavior can be predicted using finite element analysis (FEA) [88].

7.2.2 Load-Dependent Stiffness Behavior

The stiffness of a joint when subjected to transverse loads is an important factor for consideration when designing balancers [57, 90]. To successfully design a statically balanced joint incor-

porating lattice flexures, this load-dependent behavior must be evaluated. Following the methods outlined in [90], dimensionless loads η and ν are defined as

$$\eta = \frac{HL^2}{EI} \tag{7.1}$$

and

$$\nu = \frac{VL^2}{EI} \tag{7.2}$$

with dimensionless stiffness defined as

$$\kappa = \frac{KL}{EI} \tag{7.3}$$

In addition to varying η and ν, the dimensionless parameters L_1/b and $h/(h+b)$ were varied to fully investigate the variation of load-dependent stiffness behavior with lattice geometry. In all, 11,900 data points were analyzed for the X- and V-type flexures with square lattice elements.

To non-dimensionalize the loads and stiffness of a lattice flexure, a flexural stiffness term is needed. For a blade flexure, this term is EI, the Young's modulus multiplied by the second moment of area. The stiffness is then calculated as $k = EI/L$. Similarly, a flexural stiffness term can be defined for a lattice flexure by multiplying the stiffness from [88] (Equations (9) and (12) in that publication) by a length term. The result is given by

$$EI_{\text{eff}} = \begin{cases} 2E\left[I_r + \dfrac{2I_l\frac{L_1}{b}\sqrt{\left(\frac{L_1}{b}\right)^2+\frac{1}{4}}}{(1+\nu)\frac{I_l}{K}+2\left(\frac{L_1}{b}\right)^2}\right] & \text{X-type} \\ E\left[2I_r + \dfrac{I_l\frac{L_1}{b}\sqrt{\left(\frac{L_1}{b}\right)^2+1}}{2(1+\nu)\frac{I_l}{K}+\left(\frac{L_1}{b}\right)^2}\right] & \text{V-type} \end{cases}, \tag{7.4}$$

where E is the Young's modulus, I_r is the second moment of area of the lattice rail, I_l is the second moment of area of the lattice diagonal element, $\frac{L_1}{b}$ is a lattice length ratio, and K is a torsional stiffness term for the diagonal element.

A parametric FE model was developed that could analyze the two lattice types with varying applied loads and geometries. Applying a nonlinear fit function to the data, it was found that the load-dependent dimensionless stiffness of X- and V-type lattice flexures is given by

$$\kappa = \nu\beta_1 + \frac{L_1}{b}\beta_2 + \beta_3 + \eta^2\beta_4 + \nu^2\beta_5 \tag{7.5}$$

where β_i are coefficients given in Table 7.1, κ is dimensionless stiffness, and η and ν are dimensionless horizontal and vertical loads, respectively. This expression was found by assuming that the stiffness expression would follow a quadratic expression of all four variables, then eliminating all terms that could be eliminated without significantly lowering the value of R^2. Equation (7.5) is valid for $-6 \leq \eta \leq 6$, $-8 \leq \nu \leq 8$, $0.02 \leq \frac{h}{h+b} \leq 0.20$, and $0.2041 \leq \frac{L_1}{b} \leq 2.5000$.

7.3 Static Balancing

Once the load-dependent stiffness behavior is known and a lattice geometry has been selected, the balancer can be designed using the method outlined in [57]. In summary, a pair of non-dimensional parameters $\Pi_1 = k_\theta/Pd$ and $\Pi_2 = k_l d/P$ is selected such that the resulting mechanism is approximately balanced for $\pm 20^\circ$ of rotation. In this way a balanced joint can be designed knowing only its stiffness k_θ under compressive load P due to a tension spring of stiffness k_l mounted a distance d from the center of rotation.

Table 7.1: Values of the β_i coefficients used in Equation (7.5), and R^2 values for each set of coefficients. These coefficients are used to describe the stiffness of X-type and V-type lattice flexures subjected to lateral loads.

	X-type	V-type
β_1	-0.2263	-0.2280
β_2	-0.0911	-0.0541
β_3	2.0862	2.0502
β_4	-0.0031	-0.0027
β_5	-0.0029	-0.0026
R^2	0.9949	0.9964

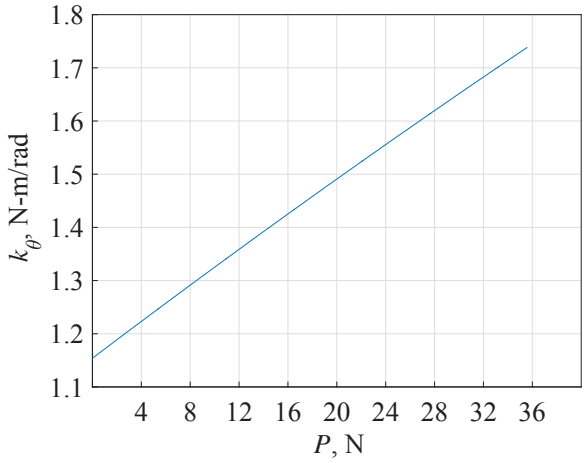
In previous work on static balancing, the figure-of-merit is often stiffness reduction. This is useful if the torque-displacement curve of the statically balanced mechanism can be approximated as linear. In this work the figure-of-merit is actuation energy reduction. The actuation energy is evaluated as the energy required to displace a joint to its full range of motion. The actuation energy reduction (AER) is then calculated as the difference in actuation energies of the original and modified joints divided by the original joint actuation energy.

A V-type lattice was selected for the joint because it generally is less stiff than a similar X-type flexure. To improve stability, a compound joint with two CAFPs arranged in parallel (co-axially and both attached to either side of ground and output) was used [89]. To account for this symmetry, the EI_{eff} in Equation (7.4) was doubled. Parameters for the flexure were determined and are listed in Table 7.2. To choose balancer parameters, the load-dependent stiffness k_θ of the lattice flexure was found from Equation (7.5) for a range of preload P. The corresponding k_l and d were then found for the resulting k_θ and P. Figure 7.3 shows how the selection of the value of preload, P, affects the other parameters' values. P was chosen so that the value of d would be acceptably small.

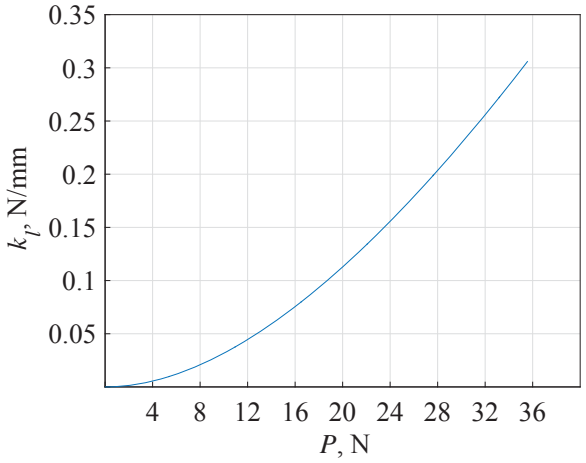
To confirm that the resulting cross-axis flexural pivot is statically balanced, a finite element model was built using the commercial code ANSYS. The model uses BEAM188 elements for

Table 7.2: Parameters for the balanced lattice flexure system.

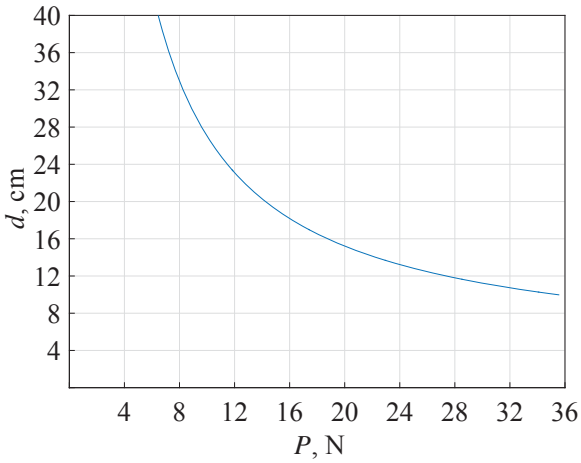
Parameter	Value	Units
L	76.2	mm
Type	V	—
E	111×10^9	GPa
n	5	—
h	1.0	mm
$\frac{h}{h+b}$	0.04	—
$\frac{L_1}{b}$	0.3125	—
Π_1	0.49	—
Π_2	0.8581	—
k_θ	1.7385	N-m
k_l	0.3063	N/mm
P	35.6	N
d	9.97	cm



(a) Load-dependent stiffness k_θ as a function of preload P.



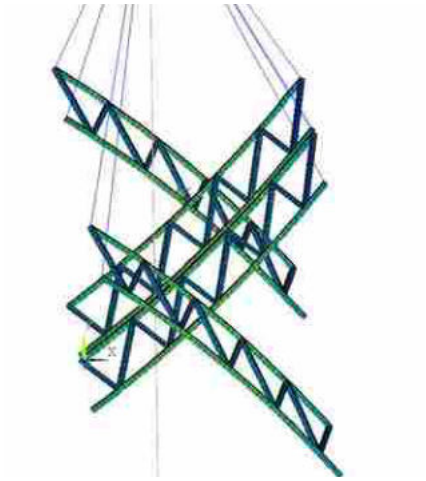
(b) Balancer linear stiffness k_l as a function of preload P.



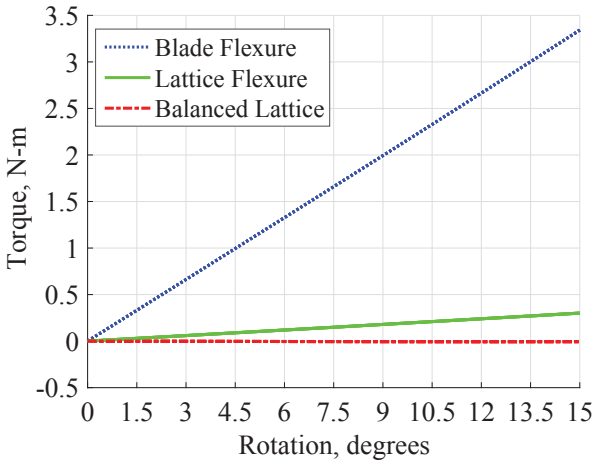
(c) Distance d as a function of preload P.

Figure 7.3: Plots used in the selection of the balancer parameters. These plots allow a designer to quickly visualize the effects of P on joint stiffness, spring stiffness, and mounting distance.

the lattice flexures, COMBIN14 elements for an ideal balancing spring with a specified initial preload, and MPC184 elements for the rigid sections that attach the spring to the flexible elements. The model and results (compared to an unbalanced version and a CAFP with conventional blade flexures) are shown in Figure 7.4. The predicted actuation energy reduction (AER) compared to the blade flexure is 99.7%, i.e. the actuation energy of the balanced joint is 0.3% that of the unbalanced joint. Compared to the unbalanced lattice flexure, the predicted AER is 97.2%.



(a) FEA model of a CAFP with lattice flexures and ideal spring used as a balancer.



(b) Torque-rotation curve obtained from FEA, confirming that the mechanism in 7.4a is statically balanced.

Figure 7.4: Preliminary statically balanced design confirmed by FEA.

7.4 Printable Balancer

While the statically balanced system presented in Section 7.3 appears to perform well, it relies on a mathematically ideal spring for its balancer. To build a physical prototype, a printable spring is desired. As listed in Table 7.2, this spring needs a stiffness of 0.3063 N/mm at a preload of 35.6 N. This could be achieved by designing a linear spring with the desired stiffness and deflecting it until the preload is reached. Alternatively, a nonlinear spring could be designed that quickly reaches the desired preload and then matches the desired stiffness. In this work, the latter strategy was adopted.

Because the desired stiffness is relatively small, it was decided to adapt a constant-force mechanism into a nonlinear balancing spring [91]. A diagram of the proposed balancer topology is shown in Figure 7.5. The balancer consists of an angularly offset parallel-guiding mechanism and a driver dyad. This is mirrored about a centerline. Its geometry is completely defined by lengths L_1, L_2, angle θ_{10}, the cross section widths and thicknesses (b_1 & b_2, h_1 & h_2), and the offset distance. θ_{20} is found as $\theta_{20} = \arccos(L_1 \cos(\theta_{10})/L_2)$.

A pseudo-rigid-body model of this mechanism was developed. Symmetry was employed to reduce the model to a slider-crank mechanism with two torsion springs. A diagram of this model

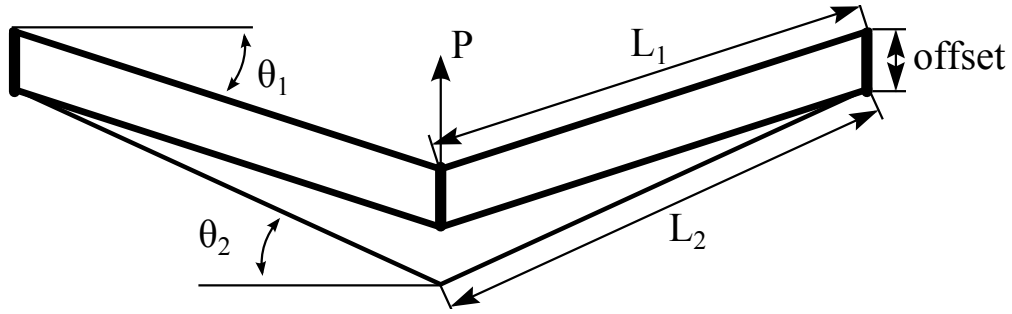
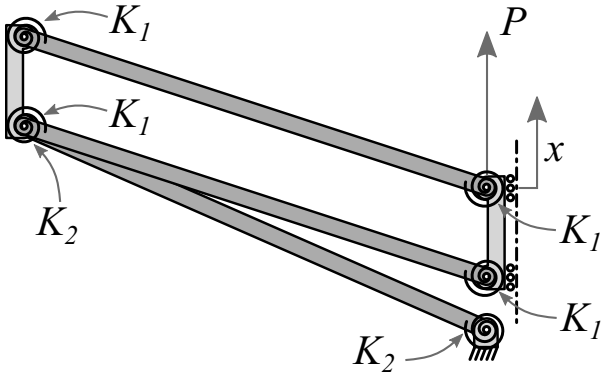
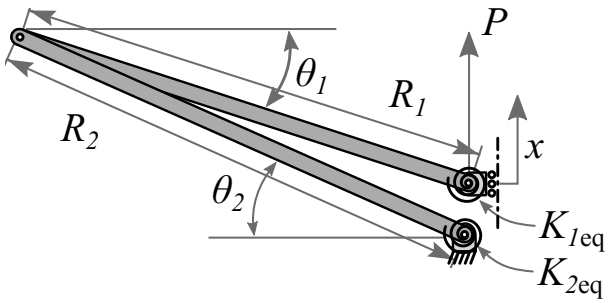


Figure 7.5: Topology of the proposed printable balancer.



(a) Full model of the parallel-guiding mechanism and driver dyad.



(b) The model in Figure 7.6a reduced to a slider-crank. Letting $K_{1_{eq}} = 4K_1$ and $K_{2_{eq}} = 2K_2$ results in equal force-displacement behavior.

Figure 7.6: Pseudo-rigid-body approximation of the balancer topology shown in Figure 7.5.

is shown in Figure 7.6, and the force output, which is also the compressive preload for the system ($-P$), is

$$P = K_{1_{eq}}(\theta_1 - \theta_{10})\left(\frac{-\sin(\theta_2)}{R_1\sin(\theta_2-\theta_1)}\right) + K_{2_{eq}}(\theta_2 - \theta_{20})\left(\frac{-\sin(\theta_1)}{R_2\sin(\theta_2-\theta_1)}\right) \quad (7.6)$$

To adapt this mechanism for use as a balancing spring, its geometry was optimized using Matlab's built-in interior-point optimization algorithm. The objective function was designed to give the desired preload P and the desired (locally) constant stiffness k_l. Therefore, the objective function was

$$\text{fit} = \left(\frac{P}{P_{\text{desired}}} - 1\right)^2 + \left(\frac{\frac{dP}{dx}}{k_{l\ \text{desired}}} - 1\right)^2 + \frac{d^2P}{dx^2} \tag{7.7}$$

subject to the constraint that bending stress not exceed 413.6 MPa. The bending stress was calculated as

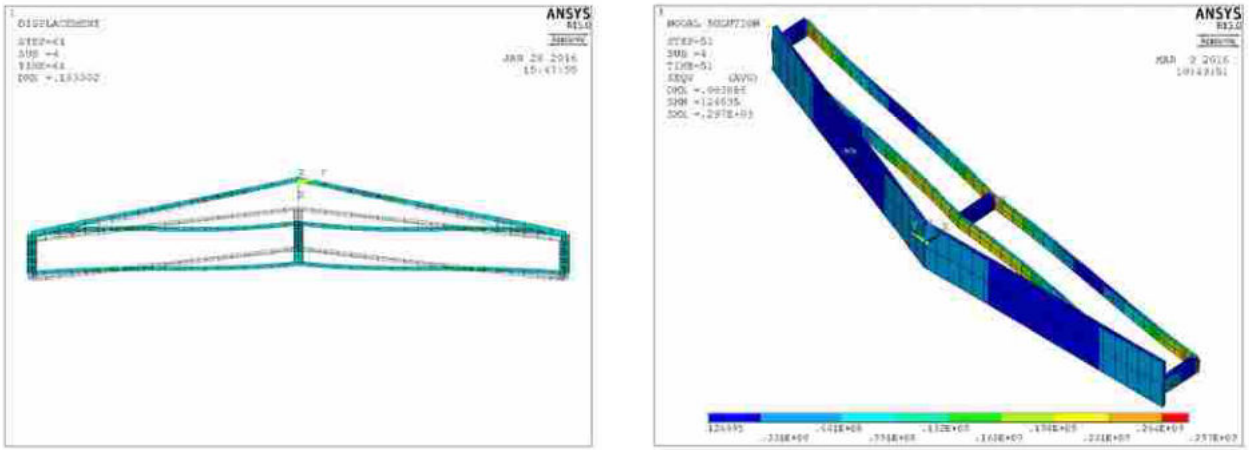
$$\sigma_{\max} = \max \begin{cases} \frac{K_1(\theta_1 - \theta_{10})\frac{t_1}{2}}{I_1} \\ \frac{K_2(\theta_2 - \theta_{20})\frac{t_2}{2}}{I_2} \end{cases} \tag{7.8}$$

Equation (7.7) was minimized with respect to the variables listed in Table 7.3. First, the pseudo-rigid-body model (PRBM) was optimized. The resulting optimal values were used as the starting point for a second optimization routine. This second routine used a finite element model to evaluate the fitness function. The constraint on stress was included in the PRBM as a nonlinear constraint, while in the finite element code it was included as a penalty on the fitness function. This tiered optimization strategy saved time compared with trying to optimize a solution using only a finite element model, and results in a better solution than relying exclusively on the PRBM. The finite element model used is shown in Figure 7.7. It used ANSYS BEAM188 elements of the appropriate cross-sections for all flexible members.

The resulting force-displacement curves of the optimized balancer spring are shown in Figure 7.8 Note that the performance predictions for the PRBM and FEA Initial seem close to the target values, but the optimal value obtained from FEA is substantially different. It was found that the PRBM under-predicted stress enough that its optimal values were not feasible. The proposed

Table 7.3: Variables to be optimized in the design of the balancer spring. PRBM value is the optimized value according to the PRBM equations, while final value indicates the optimized value obtained from the finite element model.

Parameter	Units	Lower Bound	Upper Bound	PRBM Value	Final Value
L_1	mm	25.4	152.4	89.26	86.54
L_2	mm	25.4	152.4	90.65	88.05
b_1	mm	0.254	17.8	4.653	4.98
b_2	mm	2.54	25.4	17.31	19.51
h_1	mm	1.0	1.52	1.183	1.16
h_2	mm	1.0	1.52	1.35	1.35
θ_{10}	rad	0.0	0.7854	0.0974	0.10319



(a) Initial (gray) and deflected (blue) position of the balancer.

(b) Plot of Von Mises stress of the balancer while deflected. Units are Pa.

Figure 7.7: FEA model of the balancer used in the balancer optimization routine.

topology does not have enough freedom in the feasible design space to match both the target preload and the target stiffness. Thus, the balancer can be operated at the optimal preload, the optimal stiffness, or somewhere in between. It was found that operating at the desired preload yielded better performance than operating at the desired stiffness. This is because the preload has a great effect on rotational stiffness k_θ, which has more effect on the balancing effectiveness than variation in the stiffness k_l.

The performance metrics for the optimized balancer are summarized in Table 7.4. Although the desired stiffness was not met, these values were tested in the FEA model shown in Figure 7.4a (where the balancer is represented as an ideal spring with the given preload and stiffness). It was found that the actuation energy reduction (AER) of the off-nominal balancer was nearly the same as the ideal balancer and no further optimization was pursued.

Table 7.4: Balancer parameters achieved with printable balancer.

Parameter	Units	Target	Final
P	N	35.6	35.6
k_l	N/mm	0.3063	0.9422
Δx	mm	—	4.47

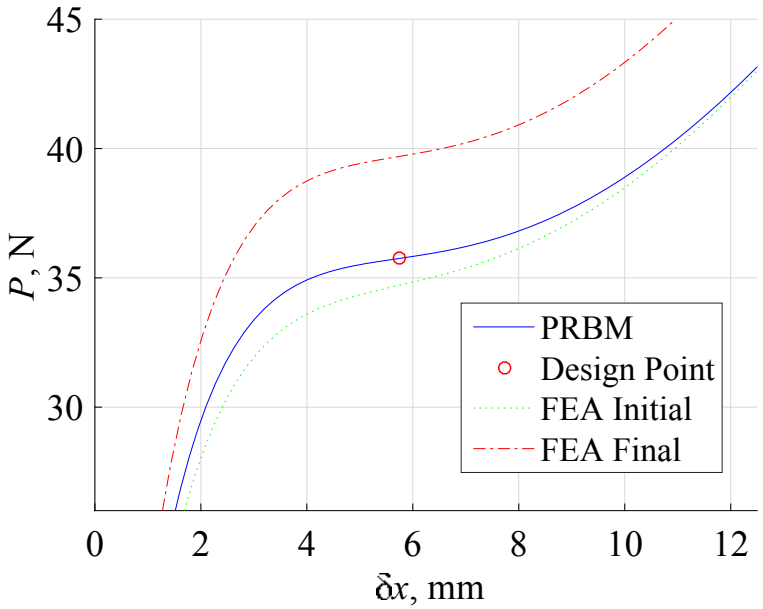
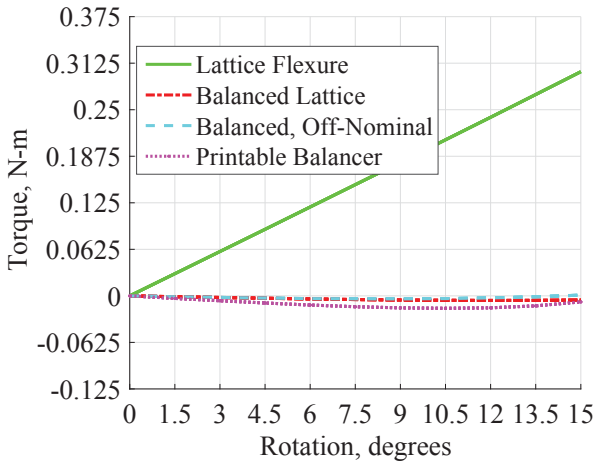


Figure 7.8: Force-displacement curve of the optimized balancer spring. The solid line labeled "PRBM" is the curve predicted by the pseudo-rigid-body model at the analytically predicted optimum, with the ideal preload indicated by the red circle for the design point. The dotted line labeled "FEA Initial" shows the FEA prediction of performance for the parameters obtained from the PRBM. Finally, the dashed line labeled "FEA Final" indicated the optimum performance obtained from the FE model.

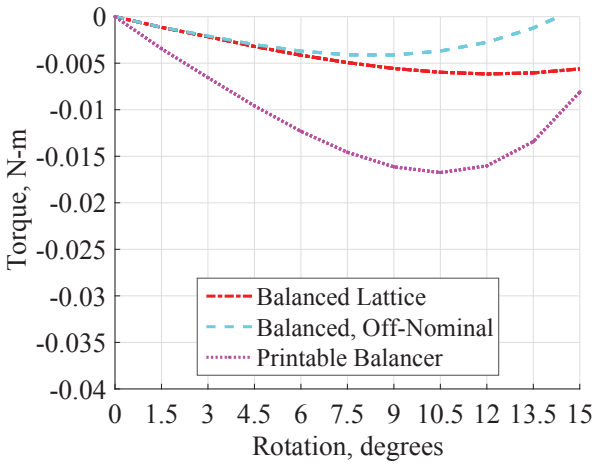
7.5 Flexure-Balancer Integration

To ensure that the balancer performed adequately, the finite element model of the lattice flexure and ideal spring was re-run using the balancer values listed in Table 7.4. The results are plotted in Figure 7.9. The predicted actuation energy reduction (AER) (compared with the unbalanced lattice) was 99.8%.

Finally, the finite element models of the lattice flexure and the printed balancer were completely integrated into a single finite element model, shown in Figure 7.10. The balancer was anchored to connection points on the lattice model using a slender rod consisting of solid circular ANSYS BEAM188 elements 1.0 mm in diameter (the minimum feature size in an ARCAM 3D printer is 1.0 mm). To apply the preload, the lower end of the balancer was displaced downward by the preload distance Δx, then fixed in all other degrees of freedom. This model exhibits an AER (compared with the unbalanced lattice) of 92.5%.



(a) Comparison of the blade flexure to the lattice flexure design with and without balancing mechanisms.



(b) Comparison of the printable balancer performance with performance of an ideal spring (with nominal and off-nominal stiffness and preload values).

Figure 7.9: Torque-displacement plots. "Balanced Lattice" shows results for the lattice flexure with an ideal spring used as the balancer with spring parameters at their nominal values. "Balanced, Off-Nominal" shows results for the lattice flexure balanced by an ideal spring, but with spring parameters at values predicted for the printable balancer. "Printable Balancer" shows results for the lattice flexure with the printable balancer.

The slender rod (see Figure 7.2) must undergo the same total deflection as the lattice flexure. It was found that deflection in the rod was concentrated near the anchor points where it attaches to the flexure. The distance d from the center of rotation to the spring attachment point assumes a frictionless pivot. To preserve the location of this pivot, the characteristic pivot of the rod must be located a distance d from the center of rotation. Therefore, the distance to the anchor point of the connecting rod was found by d/γ, where $\gamma \approx 0.85$ is a parameter for describing the location of the characteristic pivot in the pseudo-rigid-body-model [1]. In this way, the distance from the center of rotation to the pivot point of the spring is preserved at near its optimal value.

The resulting torque-displacement data are shown in Figure 7.9.

7.6 Experimental Validation

The mechanism was designed and built using EBM 3D-printed titanium at NASA Marshall Space Flight Center. The printed flexure (after removal of support material) is shown in Figure 7.11. This was tested to determine its torque-displacement behavior. An Omega TQ-103

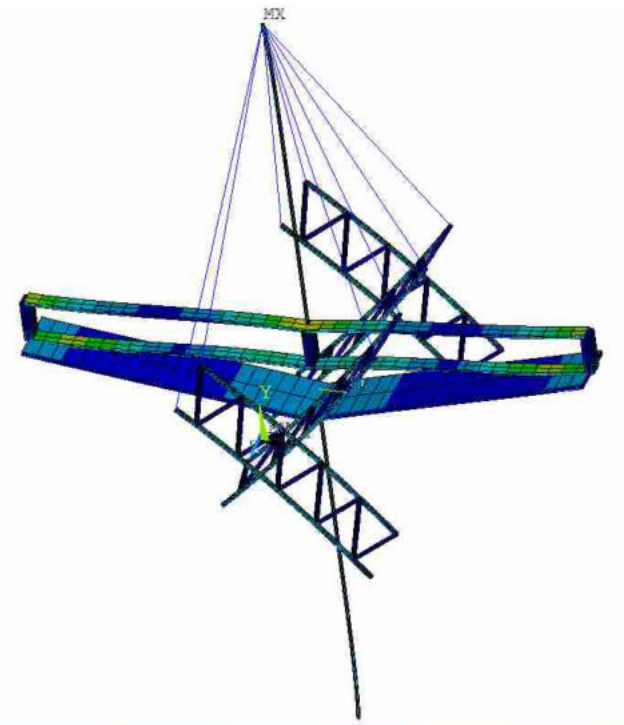


Figure 7.10: Finite element model of the lattice flexure complete with printable balancer.

torque transducer was used to measure applied torque, while position was measured with a US Digital optical encoder. The instruments were read using Labview. The test setup is shown in Figure 7.12.

The flexure was deflected to various positions between $\pm 15^{\circ}$. Although the analysis predicts a displacement for optimal preload, the roughness of the titanium surface made an accurate measurement of the preload difficult. Therefore, the preload was adjusted by turning the jack-screw to apply more or less tension to the balancing spring.

Figure 7.13 shows the torque-displacement data. Actuation energy reduction (AER) was calculated from the torque-displacement data using the trapezoidal rule for numerical integration. Table 7.5 summarizes these results. The maximum observed AER was 99%. This is in comparison to a CAFP made from blade flexures of the same width, thickness, and length as the lattice flexures used here. In other words, the actuation energy of the balanced flexure was 1% of the actuation energy of an equally sized blade flexure.

Figure 7.11: Printed titanium flexure.

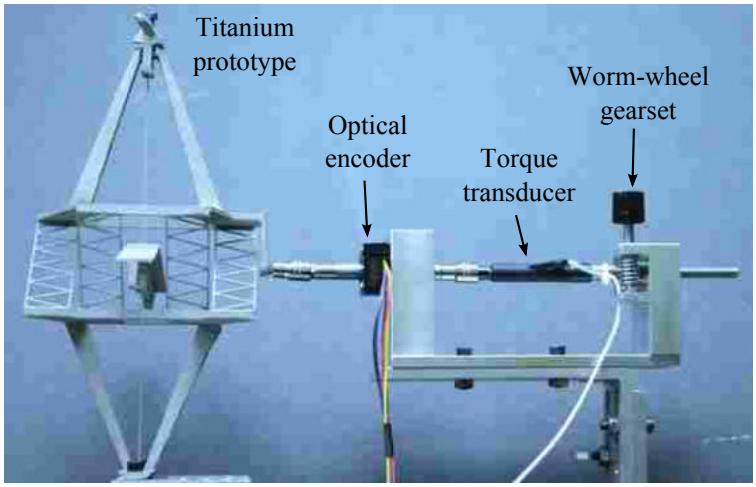


Figure 7.12: Printed titanium flexure in the test setup.

7.7 Discussion and Conclusion

We have presented the design of a mechanism exhibiting very high AER accomplished using three methods of stiffness modification: compound joints, lattice flexures, and static balancing. While each method has been presented in the literature, this work discusses an example of integrat-

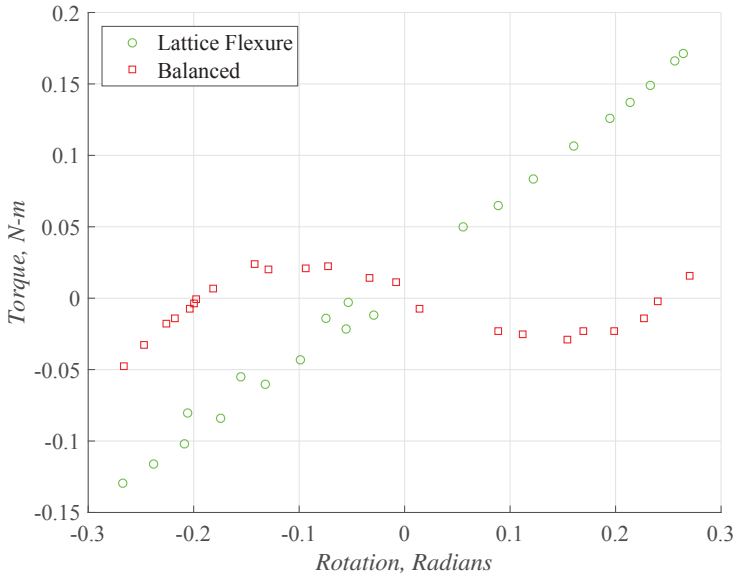


Figure 7.13: Torque-displacement behavior of the prototype without a preload (lattice flexure) and with a preload (balanced).

ing all three methods. Additionally we have presented a solution describing the load-dependent stiffness behavior of lattice-flexured CAFPs, and the design of a 3D-printable nonlinear tension spring has been presented.

Key results include:

- Defined a flexural stiffness term for use in lattice flexure analysis
- Quantified load-dependent stiffness behavior for two lattice flexure types
- Proposed a figure-of-merit for comparing nonlinear stiffness behavior
- Presented a method and topology for 3D-printable nonlinear tension spring design

Table 7.5: Actuation energy reduction measured in a titanium prototype, compared to an analytic model of a blade flexure (AER (blade)) and prototype lattice flexure (AER (lattice)) with no preload applied to the balancer spring.

	AER (blade)	AER (lattice)
Lattice Flexure	96%	—
Balanced Joint	99%	75%

- Demonstrated a statically-balanced 3D-printed mechanism
- Measured two orders of magnitude actuation energy reduction (AER) compared to conventional blade flexures

The presented joint was designed to demonstrate the validity of integrating compound joints, lattice flexures, and static balancing. Using lattice flexures or static balancing individually can yield impressive reductions in required actuation energy, but employing both strategies is feasible and beneficial when reducing actuation energy is required. Not only does 3D printing facilitate the fabrication of such mechanisms, but continuing to explore new design methods tailored for 3D printing will increase the utility and appeal of additive manufacturing technologies.

CHAPTER 8. CONCLUSIONS AND RECOMMENDATIONS

8.1 Summary

This work has explored three strategies for improving the flexibility of flexures. Compliant mechanisms designers often seek flexures that are less stiff, have a greater range of motion, are more precise, or have an improved response to off-axis loads. This work provides strategies to accomplish each of those design goals. Not only does 3D printing facilitate the fabrication of such mechanisms, but design methods discussed herein allow for greater exploitation of additive manufacturing. Achieving low-stiffness motion from 3D printed parts allows additive manufacturing to be employed in a wider range of applications where motion is an important consideration in the design.

Chapter 2 is a brief discussion of some considerations when designing for 3D printing, especially using metal alloys. The principles therein have been used throughout this work.

Chapter 3 details a non-dimensional approach for the static balancing of rotational flexures. By using a set of simple non-dimensional parameters, a static balancer can be designed with a very small set of calculations. This chapter is a significant contribution to the field of static balancing because it greatly simplifies the design of static balancing systems. The mothed presented in this chapter is general, and can be applied to any flexure system that can be approximated as a pin joint with a torsion spring.

Chapter 4 is a supplement to Chapter 3, and presents a method for determining the load-dependent stiffness of flexures. Any flexure topology may be analyzed in this way. Understanding of the load-dependent stiffness behavior is necessary to use the static balancing method presented in Chapter 3.

Chapter 5 introduces the concept of the lattice flexure. By removing material from the flexure, its stiffness is reduced. However, the stiffness is reduced more than the simple reduction in cross section would predict. This is due to the diagonal elements of the lattice being in combined

bending-torsion. Additionally, lattice flexures can exhibit improved off-axis stiffness behavior compared with conventional blade flexures. These benefits come with the trade-offs of increased design complexity, increased stresses, and lower load-carrying capacity.

Chapter 6 presents the concept of compound joints. By arranging arrays of flexures in series and series-and-parallel, the range of motion increases, the stiffness can be decreased, the off-axis stiffness can be increased, and the rotational precision of the joint can be increased.

Chapter 7 expands and combines the methods and strategies detailed in the other chapters. A definition of the flexural stiffness (EI_{eff}) of a lattice flexure is derived, and used to quantify lattice flexure load-dependent stiffness behavior. A fully printable balancing spring design is also presented. It is demonstrated that static balancing, lattice flexures, and compound joints can be used in concert to achieve unprecedented performance.

8.2 Contributions

A number of publications have been produced as part of this work, listed below:

- Merriam, E.G., Jones, J.E., and Howell, L.L., "Design of 3D Printed Titanium Compliant Mechanisms," in: *Proceedings of the Aerospace Mechanisms Symposium*, 2014 (included as Chapter 2)
- Merriam, E.G., and Howell, L.L., 2015."Non-dimensional approach for static balancing of rotational flexures." *Mechanism and Machine Theory,* **84**, pp. 90-98 (included as Chapter 3)
- Merriam, E.G., Bruton, J.T., Magleby, S.P., and Howell, L.L. "A Method for Determining Load-Dependent Stiffness of Flexures," in: *Proceedings of ASME 2015 IDETC*, 2015 (included as Chapter 4)
- Merriam, E.G., and Howell, L.L. "Lattice Flexures: Geometries for Stiffness Reductions of Blade Flexures." *Precision Engineering,* Available online 24 February 2016 (included as Chapter 5)
- Merriam, E.G., Lund, J.M., Howell, L.L. "Compound Joints: Behavior and Benefits of Flexure Arrays." *Precision Engineering,* Available online 2 February 2016 (included as Chapter 6)

- Merriam, E.G., Berg, A.B., Willig, A., Parness, A., Frey, T., and Howell, L.L., "Microspine Gripping Mechanism for Asteroid Capture," in: *Proceedings of the Aerospace Mechanisms Symposium*, 2016

Additionally, two papers have been submitted for publication and are currently under review:

- Merriam, E.G., Tolman, K.A., and Howell, L.L., 2016."Integration of Advanced Stiffness-Reduction Techniques Demonstrated in a 3D-Printable Joint." submitted to *Mechanism and Machine Theory,* submitted April 7 (included as Chapter 7)

- Tolman, K.A., Merriam, E.G., and Howell, L.L., 2016."Compliant Constant-Force Linear-Motion Mechanism." in review for publication in *Mechanism and Machine Theory,* submitted March 5

I have been able to do some outreach work with artists and capstone teams:

- Consulted on creation of animations describing satellite antenna and thruster applications for the 3D-printable space pointing mechanism

- Consulted with capstone team designing 3D-printable BSM covers

- Consulted with capstone team designing 3D-printable Isotruss Interstage for NanoLaunch1200 launch vehicle

I have submitted two invention disclosures to NASA, covering work done during my masters degree and during my time as a visiting technologist at NASA Marshall Space Flight Center:

- "Monolithic Two-Degree-of-Freedom Compliant Pointing Mechanisms, Merriam, E.G., Howell, L.L., and Jones, J.J., with Brigham Young University and NASA Marshall Space Flight Center (NASA Case number MFS-33068-1).

- "Aft-End Plug-Style 3D-Printable Solid-Rocket-Motor Igniter, Merriam, E.G., Matthias, S. and Jones, J.J., with NASA Marshall Space Flight Center.

During my time as a visiting technologist at NASAs Marshall Space Flight Center I helped design an aft-end plug-style second-stage solid-rocket-motor 3D-printable igniter for the Nanolaunch rocket program. Additionally, during my time at the Jet Propulsion Laboratory I helped develop a compliant suspension for a microspine gripping mechanism intended for use on an asteroid capture mission.

As of the conclusion of this work, I have produced:

- Three distinct and complementary methods for improving the flexibility of compliant mechanisms
- Analytic and numeric models that describe these methods
- A method for determining load-dependent stiffness of flexures
- Prototypes demonstrating the validity of these models

8.3 Conclusions

The following conclusions may be drawn from the work herein presented:

- Complex mechanisms can be built using additive metal manufacturing
- Static balancing can be accomplished using simple non-dimensional parameters
- Load-dependent stiffness must be adequately understood before a joint can be statically balanced
- Load-dependent stiffness can be determined from a straightforward finite element analysis
- Lattice flexures reduce bending stiffness and increase off-axis stiffness ratios
- Compound joints can improve flexibility and off-axis stiffness
- Compound joints can have significantly reduced center shift for specified target deflections
- Static balancing, lattice flexures, and compound joints can be used in concert to dramatically reduce flexure stiffness

- The potential utility of additively manufactured parts has been increased by low-actuation-effort design strategies

Additionally, several general conclusions can be drawn in light of this work. At the outset of this work it seemed that reducing stiffness would not be anything more than making a flexure thinner or longer. In fact, very subtle alterations to a flexure's design and use can result in dramatic changes in behavior. The reduced center shift discussed in Chapter 6 was not an anticipated contribution of this work. It was discovered as data was examined and as we attempted to explain observed behavior, then asking 'what if' questions to see if that behavior could be manipulated. Finally, in many cases I would have been satisfied with only basic results, but was spurred onward by my adviser, anonymous reviewers, my wife, and others, to achieve more meaningful, profound results than I could have hoped for. There is always more to see, more to explore, and more to learn than meets the eye.

8.4 Recommendations

This work is by no means the definitive answer when it comes to reducing the stiffness of 3D printed compliant mechanisms. The mechanisms demonstrated herein can still be improved, made smaller, and simplified.

The Π groups presented in Chapter 3 were determined for a joint undergoing $\pm 20^\circ$ of deflection; for larger or smaller deflections, a slightly different relationship may give better results.

A catalog of the load-dependent stiffness behavior of common flexures other than those presented in Chapter 4 should be compiled to aid mechanism designers, not only in static balancing, but in other applications where flexures are subject to transverse loading.

The X- and V-type lattice flexures presented in Chapter 5 were never optimized in any way. Significant benefit may be derived from optimizing lattice flexure geometry for greater stiffness reduction or higher off-axis stiffness ratios.

Other configurations of compound joints should be explored, perhaps allowing for hybrid joints where not all flexures are identical, or where the axes of rotation are not co-linear.

A more compact and parameterized printable balancer spring is one step that would greatly simplify the static balancing of printed flexures.

In short, this work could be taken forward in many different directions. If I had to choose one I would investigate the application of static balancing to multi-axis mechanisms. That could provide a compliant approximation of a ball-and socket joint, which has many potential application in space as a pointer mechanism or on earth in prosthetics or other fields.

REFERENCES

[1] Howell, L. L., 2001. *Compliant Mechanisms*. John Wiley Sons, Inc. 1, 2, 4, 13, 21, 22, 35, 47, 48, 67, 87, 99

[2] Howell, L. L., Magleby, S. P., and Olsen, B. M., 2013. *Handbook of Compliant Mechanisms*. Wiley Online Library. 1, 2, 3, 4

[3] Morsch, F. M., and Herder, J. L., 2010. "Design of a generic zero stiffness compliant joint." In *ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 427–435. 1, 8, 10, 21, 29, 35, 87

[4] Ma, R. R., Belter, J. T., and Dollar, A. M., 2015. "Hybrid deposition manufacturing: Design strategies for multimaterial mechanisms via three-dimensional printing and material deposition." *Journal of Mechanisms and Robotics,* **7**(2), p. 021002. 1

[5] Moon, S. K., Tan, Y. E., Hwang, J., and Yoon, Y.-J., 2014. "Application of 3d printing technology for designing light-weight unmanned aerial vehicle wing structures." *International Journal of Precision Engineering and Manufacturing-Green Technology,* **1**(3), pp. 223–228. 1

[6] Rafi, H. K., Karthik, N. V., Starr, T. L., and Stucker, B. E. "Mechanical property evaluation of ti-6al-4v parts made using electron beam melting.". 1, 6, 15, 16

[7] Murr, L. E., Gaytan, S., Ceylan, A., Martinez, E., Martinez, J., Hernandez, D., Machado, B., Ramirez, D., Medina, F., Collins, S., et al., 2010. "Characterization of titanium aluminide alloy components fabricated by additive manufacturing using electron beam melting." *Acta materialia,* **58**(5), pp. 1887–1894. 1, 15

[8] Lamers, A., Snchez, J. A. G., and Herder, J. L., 2015. "Design of a statically balanced fully compliant grasper." *Mechanism and Machine Theory,* **92**, pp. 230 – 239. 1, 4

[9] Queral, V., 2015. "3d-printed fusion components concepts and validation for the ust 2 stellarator." *Fusion Engineering and Design,* **9697**, pp. 343 – 347 Proceedings of the 28th Symposium On Fusion Technology (SOFT-28). 1

[10] Merriam, E. G., Jones, J. E., Magleby, S. P., and Howell, L. L., 2013. "Monolithic 2 DOF fully compliant space pointing mechanism." *Mechanical Sciences,* **4**, pp. 381–390. 1, 2, 6, 11, 15, 78

[11] Fowler, R. M., Howell, L. L., and Magleby, S. P., 2011. "Compliant space mechanisms: a new frontier for compliant mechanisms." *Mechanical Sciences,* **2**, pp. 205–215. 2, 13

[12] Merriam, E. G., Jones, J. E., and Howell, L. L., 2014. "Design of 3d printed titanium compliant mechanisms." In *Proceedings of the Aerospace Mechanisms Symposium*. 2, 11, 48

[13] Parise, J. J., Howell, L. L., and Magleby, S. P., 2001. "Ortho-planar linear-motion springs." *Mechanism and machine theory,* **36**(11), pp. 1281–1299. 3

[14] Fowler, R. M., 2012. "Investigation of compliant space mechanisms with application to the design of a large-displacement monolithic compliant rotational hinge." Master's thesis, Brigham Young University, Fulton College of Engineering and Technology. 3, 5, 14

[15] Trease, B. P., Moon, Y.-M., and Kota, S., 2005. "Design of large-displacement compliant joints." *Journal of mechanical design,* **127**, p. 788. 3, 54, 67

[16] Kim, C., and Ebenstein, D., 2012. "Curve decomposition for large deflection analysis of fixed-guided beams with application to statically balanced compliant mechanisms." *Journal of Mechanisms and Robotics,* **4**(4). 4, 21, 35, 47, 67

[17] Stapel, A., and Herder, J. L., 2004. "Feasibility study of a fully compliant statically balanced laparoscopic grasper." Vol. 2004, ASME, pp. 635–643. 4, 21, 35, 36, 47, 87

[18] Herder, J. L., 2001. *Energy-free Systems. Theory, conception, and design of statically balanced spring mechanisms*. Ponsen en Looijen BV. 4, 21, 35, 36, 47, 87

[19] Martini, A., Troncossi, M., Carricato, M., and Rivola, A., 2015. "Static balancing of a parallel kinematics machine with linear-delta architecture: theory, design and numerical investigation." *Mechanism and Machine Theory,* **90**, pp. 128 – 141. 4

[20] Schenk, M., Guest, S., and Herder, J., 2007. "Zero stiffness tensegrity structures." *International Journal of Solids and Structures,* **44**(20), pp. 6569 – 6583. 4, 35

[21] Deepak, S. R., and Ananthasuresh, G., 2012. "Static balancing of a four-bar linkage and its cognates." *Mechanism and Machine Theory,* **48**(0), pp. 62 – 80. 4, 21, 22, 35, 87

[22] Deepak, S. R., and Ananthasuresh, G. K., 2012. "Perfect static balance of linkages by addition of springs but not auxiliary bodies." *Journal of Mechanisms and Robotics,* **4**(2), p. 021014. 4, 35

[23] Tuijthof, G. J., and Herder, J. L., 2000. "Design, actuation and control of an anthropomorphic robot arm." *Mechanism and Machine Theory,* **35**(7), pp. 945 – 962. 4, 21, 36, 47, 87

[24] Leishman, L. C., Ricks, D., and Colton, M. B., 2010. "Design and evaluation of statically balanced compliant mechanisms for haptic interfaces." *Proc. ASME Dynamic Systems and Control Conference*, September. 4, 10, 35

[25] Radaelli, G., Gallego, J. A., and Herder, J. L., 2011. "An energy approach to static balancing of systems with torsion stiffness." *Journal of Mechanical Design,* **133**, p. 091006. 4, 8, 35

[26] Pluimers, P., Tolou, N., Jensen, B. D., Howell, L. L., and Herder, J., 2012. "A compliant on/off connection mechanism for preloading statically balanced compliant mechanisms, DETC2012-71509." In *Proceedings of the ASME International Design Engineering Technical Conferences*, ASME. 4, 21, 36, 47, 87

[27] Wooten, J., and Dennies, D. P., 2008. "Electron beam melting manufacturing for production hardware, paper number 08amt-0061." *SAE International*. 5, 13, 14

[28] Murr, L. E., Gaytan, S. M., Ramirez, D. A., Martinez, E., Hernandez, J., Amato, K. N., Shindo, P. W., Medina, F. R., and Wicker, R. B., 2012. "Metal fabrication by additive manufacturing using laser and electron beam melting technologies." *Journal of Materials Science & Technology,* **28**(1), pp. 1–14. 5, 13

[29] Smith, C., Derguti, F., Nava, E. H., Thomas, M., Tammas-Williams, S., Gulizia, S., Fraser, D., and Todd, I., 2016. "Dimensional accuracy of electron beam melting (ebm) additive manufacture with regard to weight optimized truss structures." *Journal of Materials Processing Technology,* **229**, pp. 128 – 138. 5

[30] Vutova, K., and Donchev, V., 2016. "Non-stationary heat model for electron beam melting and refining–an economic and conservative numerical method." *Applied Mathematical Modelling,* **40**(2), pp. 1565–1575. 5

[31] Baumers, M., Dickens, P., Tuck, C., and Hague, R., 2016. "The cost of additive manufacturing: machine productivity, economies of scale and technology-push." *Technological Forecasting and Social Change,* **102**, pp. 193 – 201. 5

[32] Fox, B., 2006. *Rapid Manufacturing: An Industrial Revolution for the Digital Age*. John Wiley & Sons, Ltd, ch. Rapid Manufacture in the Aeronautical Industry, pp. 221–231. 5, 13, 14

[33] Wooten, J., 2006. *Rapid Manufacturing: An Industrial Revolution for the Digital Age*. John Wiley & Sons, Ltd, ch. Aeronautical Case Studies Using Rapid Manufacturing, pp. 233–239. 5, 13, 14

[34] Rawal, S., Brantley, J., and Karabudak, N., 2013. "Additive manufacturing of Ti-6Al-4V alloy components for spacecraft applications." In *Recent Advances in Space Technologies (RAST), 2013 6th International Conference on*, IEEE, pp. 5–11. 5, 14

[35] Spielman, R., 2006. *Rapid Manufacturing: An Industrial Revolution for the Digital Age*. John Wiley & Sons, Ltd, ch. Space Applications, pp. 241–248. 5, 14

[36] Halchak, J., Wooten, J., and McEnerney, B., 2005. Layer build of titanium alloy components for complex-geometry rocket engine. 5, 14

[37] Brigham Young University Compliant Mechanisms Research Group Flexlinks http://compliantmechanisms.byu.edu/downloads/flexlinks Accessed 2012-11-1. 9

[38] Arcam AB Ebm electron beam melting - in the forefront of additive manufacturing http://www.arcam.com/technology/electron-beam-melting/ Accessed 2013-10-17. 14

[39] Arcam AB Ebm-built materials - way beyond average http://www.arcam.com/technology/electron-beam-melting/materials/ Accessed 2013-9-20. 15, 16

[40] Jensen, B., and Howell, L., 2002. "The modeling of cross-axis flexural pivots." *Mechanism and Machine theory,* **37**(5), May, pp. 461–476. 15, 29, 40, 59, 61, 67, 69

[41] Sangamesh, D. R., 2012. *Static balancing of rigid-body linkages and compliant mechanisms*. India Institute of Science. 21, 47, 87

[42] Herder, J. L., 1998. "Design of spring force compensation systems." *Mechanism and Machine Theory,* **33**(12), pp. 151 – 161. 21, 35, 47, 87

[43] Tolou, N., 2012. Statically balanced compliant mechanisms for mems and precision engineering Dissertation. 21, 44, 87

[44] Wang, J., and Gosselin, C., 2000. "Static balancing of spatial four-degree-of-freedom parallel mechanisms." *Mechanism and Machine Theory,* **35**(4), APR, pp. 563–592. 21, 35, 87

[45] French, M., and Widden, M., 2000. "The spring-and-lever balancing mechanism, george carwardine and the anglepoise lamp." *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science,* **214**(3), pp. 501–508. 21

[46] LaCoste Jr, L. J., 1934. "A new type long period vertical seismograph." *Journal of Applied Physics,* **5**(7), pp. 178–180. 21, 87

[47] Dunning, A., Tolou, N., and Herder, J., 2013. "A compact low-stiffness six degrees of freedom compliant precision stage." *Precision Engineering,* **37**(2), pp. 380 – 388. 21, 48, 67, 87

[48] de Lange, D. J., Langelaar, M., and Herder, J. L., 2008. "Towards the design of a statically balanced compliant laparoscopic grasper using topology optimization." In *ASME 2008 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 293–305. 21, 35

[49] Wittrick, W., 1951. "The properties of crossed flexure pivots, and the influence of the point at which the strips cross." *Aeronautical Quarterly,* **2**(4), pp. 272–292. 24, 29, 36, 38, 39, 40, 43, 59, 69, 73

[50] Munson, B. R., Young, D. F., and Okiishi, T. H., 2006. *Fundamentals of Fluid Mechanics*. John Wiley & Sons. 25

[51] Kennedy, J., Eberhart, R., et al., 1995. "Particle swarm optimization." In *Proceedings of IEEE international conference on neural networks*, Vol. 4, Perth, Australia, pp. 1942–1948. 26

[52] Wittrick, W., 1948. "The theory of symmetrical crossed flexure pivots." *Australian Journal of Scientific Research A Physical Sciences,* **1**, p. 121. 29, 36, 37, 38, 59, 67, 69, 73

[53] Hongzhe, Z., and Shusheng, B., 2010. "Accuracy characteristics of the generalized cross-spring pivot." *Mechanism and Machine Theory,* **45**(10), pp. 1434–1448. 29, 59, 67, 68, 69

[54] Zhao, H., and Bi, S., 2010. "Stiffness and stress characteristics of the generalized cross-spring pivot." *Mechanism and Machine Theory,* **45**(3), pp. 378–391. 29, 36, 37

[55] Dede, E., and Trease, B., 2004. "Statically-balanced compliant four-bar mechanism for gravity compensation." *2004 ASME Student Mechanism Design Competition*. 35

[56] Merriam, E. G., Colton, M., Magleby, S., and Howell, L. L., 2013. "The design of a fully compliant statically balanced mechanism." In *ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers. 35

[57] Merriam, E. G., and Howell, L. L., 2015. "Non-dimensional approach for static balancing of rotational flexures." *Mechanism and Machine Theory,* **84**, pp. 90–98. 35, 36, 47, 69, 87, 89, 91

[58] Shusheng, B., Hongzhe, Z., and Jingjun, Y., 2009. "Modeling of a cartwheel flexural pivot." *Journal of Mechanical Design,* **131**(6), p. 061010. 36, 37, 40, 41, 67

[59] Pei, X., Yu, J., Zong, G., Bi, S., and Su, H., 2009. "The modeling of cartwheel flexural hinges." *Mechanism and Machine Theory,* **44**(10), pp. 1900–1909. 36, 40, 67

[60] Howell, L., and Midha, A., 1994. "A method for the design of compliant mechanisms with small-length flexural pivots." *Journal of Mechanical Design,* **116**(1), March, pp. 280–290. 40, 48, 67

[61] Guerinot, A. E., Magleby, S. P., Howell, L. L., and Todd, R. H., 2005. "Compliant joint design principles for high compressive load situations." *Journal of Mechanical Design,* **127**(4), pp. 774–781. 42

[62] Brouwer, D., Meijaard, J., and Jonker, J., 2013. "Large deflection stiffness analysis of parallel prismatic leaf-spring flexures." *Precision Engineering,* **37**(3), pp. 505 – 521. 47, 48, 67

[63] Teo, T. J., Chen, I.-M., Yang, G., and Lin, W., 2010. "A generic approximation model for analyzing large nonlinear deflection of beam-based flexure joints." *Precision Engineering,* **34**(3), pp. 607 – 618. 47

[64] Young, W. C., Budynas, R. G., and Sadegh, A. M., 2012. *Roark's Formulas for Stress and Strain.*, 8 ed. McGraw-Hill New York. 47, 51, 74

[65] Rahmatalla, S., and Swan, C. C., 2005. "Sparse monolithic compliant mechanisms using continuum structural topology optimization." *International Journal for Numerical Methods in Engineering,* **62**(12), pp. 1579–1605. 48

[66] Awtar, S., and Quint, J. M., 2012. "In-plane flexure-based clamp." *Precision Engineering,* **36**(4), pp. 658 – 667. 48, 67

[67] Chu, C.-L., and Fan, S.-H., 2006. "A novel long-travel piezoelectric-driven linear nanopositioning stage." *Precision Engineering,* **30**(1), pp. 85 – 95. 48, 67

[68] Zhao, H., Bi, S., and Yu, J., 2011. "Nonlinear deformation behavior of a beam-based flexural pivot with monolithic arrangement." *Precision Engineering,* **35**(2), pp. 369 – 382. 48, 67

[69] Kang, D., and Gweon, D., 2013. "Analysis and design of a cartwheel-type flexure hinge." *Precision Engineering,* **37**(1), pp. 33 – 43. 48, 67, 77, 83

[70] Noll, T., Holldack, K., Reichardt, G., Schwarzkopf, O., and Zeschke, T., 2009. "Parallel kinematics for nanoscale cartesian motions." *Precision Engineering,* **33**(3), pp. 291 – 304. 48, 67

[71] Wiersma, D. H., Boer, S. E., Aarts, R. G. K. M., and Brouwer, D. M., 2014. "Design and Performance Optimization of Large Stroke Spatial Flexures." *Journal of Computational and Nonlinear Dynamics,* **9**(1), JAN. 48

[72] Pei, X., Yu, J., Zong, G., and Bi, S., 2010. "An effective pseudo-rigid-body method for beam-based compliant mechanisms." *Precision Engineering,* **34**(3), pp. 634 – 639. 48, 59, 67

[73] Hopkins, J. B., and Culpepper, M. L., 2011. "Synthesis of precision serial flexure systems using freedom and constraint topologies (FACT)." *Precision Engineering,* **35**(4), pp. 638 – 649. 48, 67

[74] Hopkins, J. B., and Panas, R. M., 2013. "Design of flexure-based precision transmission mechanisms using screw theory." *Precision Engineering,* **37**(2), pp. 299 – 307. 48, 67

[75] Lobontiu, N., 2014. "Compliance-based matrix method for modeling the quasi-static response of planar serial flexure-hinge mechanisms." *Precision Engineering,* **38**(3), pp. 639 – 650. 48, 67

[76] Delimont, I. L., Magleby, S. P., and Howell, L. L., 2015. "A family of dual-segment compliant joints suitable for use as surrogate folds." *Journal of Mechanical Design,* **137**(9), p. 092302. 51

[77] Goldfarb, M., and Speich, J. E., 1999. "A well-behaved revolute flexure joint for compliant mechanism design." *Journal of Mechanical Design,* **121**(3), pp. 424–429. 54, 67

[78] Zelenika, S., and De Bona, F., 2002. "Analytical and experimental characterisation of high-precision flexural pivots subjected to lateral loads." *Precision Engineering,* **26**(4), pp. 381–388. 59, 67, 69

[79] Young, W. E., 1944. "An investigation of the cross-spring pivot." *Journal of Applied Mechanics,* **11**(2), pp. 113–120. 59, 69

[80] Haringx, J. A., 1949. "The cross-spring pivot as a constructional element." *Applied Scientific Research,* **1**(1), pp. 313–332. 59, 69

[81] Troeger, H., 1962. "Considerations in the application of flexural pivots." *Automatic Control,* **17**(4), pp. 41–46. 59, 69

[82] Henein, S., and Spanoudakis, P., 2003. "Flexural pivot for aerospace mechanisms.". 67

[83] Pei, X., Yu, J., Zong, G., Bi, S., and Hu, Y., 2009. "A novel family of leaf-type compliant joints: Combination of two isosceles-trapezoidal flexural pivots." *Journal of Mechanisms and Robotics,* **1**(2), p. 021005. 67

[84] Wang, J., and Gosselin, C. M., 1999. "Static balancing of spatial three-degree-of-freedom parallel mechanisms." *Mechanism and Machine Theory,* **34**(3), pp. 437 – 452. 87

[85] Walsh, G., Streit, D., and Gilmore, B., 1991. "Spatial spring equilibrator theory." *Mechanism and Machine Theory,* **26**(2), pp. 155 – 170. 87

[86] Chen, G., and Zhang, S., 2011. "Fully-compliant statically-balanced mechanisms without prestressing assembly: concepts and case studies." *Mech. Sci,* **2**(2), pp. 169–174. 87

[87] Hoetmer, K., Herder, J. L., and Kim, C. J., 2009. "A building block approach for the design of statically balanced compliant mechanisms." In *ASME 2009 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. 313–323. 87

[88] Merriam, E. G., and Howell, L. L., 2016. "Lattice flexures: Geometries for stiffness reduction of blade flexures." *Precision Engineering*. 87, 89, 90

[89] Merriam, E. G., Lund, J. M., and Howell, L. L., 2016. "Compound joints: Behavior and benefits of flexure arrays." *Precision Engineering*, pp. –. 88, 92

[90] Merriam, E. G., Bruton, J. T., Magleby, S. P., and Howell, L. L., 2015. "A Method for Determining Load-Dependent Stiffness of Flexures, DETC2015-46628." In Proceedings of the ASME International Design Engineering Technical Conferences, ASME. 89, 90

[91] Tolman, K. A., Merriam, E. G., and Howell, L. L., 2016. "Compliant constant-force linear-motion mechanism." *Submitted to Mechanism and Machine Theory*. 94

APPENDIX A. CODES AND SCRIPTS USED IN STATIC BALANCING

The code and scripts used in this appendix are referenced in Chapter 3. In some of the ANSYS scripts, the ellipses (...) are used to indicate that the command continues on the next line. However, ... is not a valid ANSYS command. All ellipses should be removed from the file and the commands consolidated in a text editor so they are all on one line. This will enable the script to run correctly. Matlab scripts, however, should need no such cleanup.

A.1 Finding the Π-Group Relationship

This Matlab script finds the relationship between two non-dimensional numbers for a wide range of values. It requires the functions pi_stiffness.m and pso.m (included below).

```
%Pi1 is Kt/(P*l) - a relation of torsional stiffness
%Pi2 is Kl*P/l - a relation of linear stiffness
clc
clear
start = .2;
step = .01;
final = 1.;
global Pi1;
PI1 = start:step:final;
PI2 = zeros(1,length(PI1));
en = PI2;
options = optimoptions('fmincon','Algorithm','sqp');
LB = [-1 .3 .3];  %Kl P l
UB = [60 10 10];  %Kl P l
X = [1 3 1]; %Kl P l
```

```
  x0=X;
for R = 1:length(PI1)
    Pi1 = PI1(R);
    opt = pso(@pi_stiffness, LB, UB);
    X = opt{1};
    E = opt{2};
    PI2(R) = X(1)*X(3)/X(2); %Pi2 = Kl*l/P
    en(R) = E;
end
A = [PI1' PI2'];
A = sortrows(A,1);
figure(1)
plot(A(:,1),A(:,2))
xlabel('\Pi_1')
ylabel('\Pi_2')
figure(2)
plot(PI1',en(:))
ylabel('|k/k_\theta|')
xlabel('\Pi_1')
title('Average Normalized Stiffness')
```

This is the function pi_stiffness.m, used with the above script to solve for the relationship between Π_1 and Π_2.

```
%this function is called as part of optimization routine
%Pi1 = Kt/(P*l) - a relation of torsional stiffness
%Pi2 = Kl*l/P - a relation of linear stiffness
function E = pi_stiffness(x)
global Pi1;
Kl = x(1);
P = x(2);
```

```
l = x(3);
%Kl = Pi2*P/l;
Kt = Pi1*P*l;
step = .02;
theta_final = 20*pi/180;
theta = 0.02:step:theta_final;
energy = zeros(size(theta));
stiffness = energy;
torque = stiffness;
for R = 1:length(theta)
torque(R) = (Kt*theta(R)-Kl*((sqrt(2*l^2*(1+cos(theta(R))))-(2*l-P/Kl))*...
l^2*sin(theta(R)))/sqrt(2*l^2*(1+cos(theta(R)))));
stiffness(R) = torque(R)/theta(R);
end
E = mean(abs(stiffness))/Kt;
```

This particle swarm optimization algorithm (for Matlab) was used with the above script to solve for the relationship between Π_1 and Π_2. Much thanks to Abraham Lee for the original code.

```
function out = pso(func, lb, ub)
%    Perform a particle swarm optimization (PSO)
%
%    Parameters
%    ==========
%    func : function
%        The function to be minimized
%    lb : array
%        The lower bounds of the design variable(s)
%    ub : array
%        The upper bounds of the design variable(s)
%
```

```
%    Optional
%    ========
%    ieqcons : list
%      A list of functions of length n such that ieqcons[j](x,*args)
%       >= 0.0 in a successfully optimized problem (Default: [])
%    f_ieqcons : function
%      Returns a 1-D array in which each element must be greater or equal
%      specified, to 0.0 in a successfully optimized problem. If f_ieqcons
%       is ieqcons is ignored (Default: None)
%    swarmsize : int
%      The number of particles in the swarm (Default: 10)
%    omega : scalar
%      Particle velocity scaling factor (Default: 0.5)
%    phip : scalar
%      Scaling factor to search away from the particle's best known
%     position (Default: 0.5)
%    phig : scalar
%      Scaling factor to search away from the swarm's best known
%     position (Default: 0.5)
%    maxiter : int
%      The maximum number of iterations for the swarm to search
% (Default: 100)
%    minstep : scalar
%      The minimum stepsize of swarm's best position before the search
%      terminates (Default: 1e-8)
%    minfunc : scalar
%      The minimum change of swarm's best objective value before the
%     search terminates (Default: 1e-8)
%    debug : boolean
%      If True, progress statements will be displayed every iteration
```

```
%      (Default: False)
%    Returns
%    =======
%    g : array
%        The swarm's best known position (optimal design)
%    f : scalar
%        The objective value at ``g``
%
    assert(length(lb)==length(ub),...
 'Lower- and upper-bounds must be the same length');
    assert(all(ub>lb),...
 'All upper-bound values must be greater than lower-bound values');
    % Check the default options
        ieqcons = [];
        f_ieqcons = @(x) [0];
    swarmsize = 300;
    omega = 0.5;
    phip = 0.5;
    phig = 0.5;
    maxiter = 100;
    minstep = 1e-8;
    minfunc = 1e-8;
    debug = false;
    vhigh = abs(ub - lb);
    vlow = -vhigh;
    % Check for constraint function(s) #####################
    obj = @(x) feval(func, x);
    cons = @(x) f_ieqcons(x);
    function check = is_feasible(x)
        check = all(cons(x)>=0);
```

```
end
function out = zeros_like(x)
    out = zeros(size(x));
end
% Initialize the particle swarm ######################
S = swarmsize;
D = length(lb);  % the number of dimensions each particle has
x = rand(S, D);  % particle positions
v = zeros_like(x);  % particle velocities
p = zeros_like(x);  % best particle positions
fp = zeros(1, S);  % best particle function values
g = [];  % best swarm position
fg = 1e100;  % artificial best swarm position starting value
for i=1:S
    % Initialize the particle's position
    x(i, :) = lb + x(i, :).*(ub - lb);
    % Initialize the particle's best known position
    p(i, :) = x(i, :);
    % Calculate the objective's value at the current particle's
    fp(i) = obj(p(i, :));
    % If the current particle's position is better than the swarm's,
    % update the best swarm position
    if fp(i)<fg && is_feasible(p(i, :))
        fg = fp(i);
        g = p(i, :);
    end
    % Initialize the particle's velocity
    v(i, :) = vlow + rand(1, D).*(vhigh - vlow);
end
% Iterate until termination criterion met ##############
```

```
    it = 1;
    while it<=maxiter
        rp = rand(S, D);
        rg = rand(S, D);
        for i=1:S
% Update the particle's velocity
            v(i, :)=omega*v(i, :)+phip*rp(i, :).*(p(i, :)-x(i, :))+...
                      phig*rg(i, :).*(g - x(i, :));
% Update the particle's position, correcting lower and upper bound
% violations, then update the objective function value
            x(i, :) = x(i, :) + v(i, :);
            mark1 = find(x(i, :)<lb);
            mark2 = find(x(i, :)>ub);
            x(i, mark1) = lb(mark1);
            x(i, mark2) = ub(mark2);
            fx = obj(x(i, :));
% Compare particle's best position (if constraints are satisfied)
            if fx<fp(i) && is_feasible(x(i, :))
                p(i, :) = x(i, :);
                fp(i) = fx;
% Compare swarm's best position to current particle's position
% (Can only get here if constraints are satisfied)
 if fx<fg
tmp = x(i, :);
stepsize = sqrt(sum((g-tmp).^2));
if abs(fg - fx)<=minfunc
fprintf(...
'Stopping search: Swarm best objective change less than: %12.8f\n',...
 minfunc)
out = {tmp, fx};
```

```
return
elseif stepsize<=minstep
fprintf(...
'Stopping search: Swarm best position change less than: %12.8f\n',...
 minstep);
out = {tmp, fx};
return
else
g = tmp;
fg = fx;
end
end
end
end
it = it + 1;
end
    fprintf('Stopping search: maximum iterations reached --> %d\n',...
 maxiter);
    out = {g, fg};
    return
end
```

A.2 Balanced Spring Design

This Matlab script (balanced_spring_design.m) accepts design parameters for a cross axis flexural pivot and outputs design parameters to statically balance that CAFP. It is assumed that this CAFP has strips that cross in the middle and are at 90° angles to each other. Requires files get_Kt.m, get_P.m, get_I.m, and , get_L.m (included below).

```
% units are in ips
clc
```

```
clear
global E I L Pi1 Pi2 d a Klin Kt P;
Pi1 = 0.49;      %kt/(P*d)
Pi2 = 0.8581;  %k*d/P
L = 2.596;    %length of flexures
t = .015;    %thickness of flexures
b = .25*2;     %width of flexures
E = 30.e6;%Elastic modulus (steel)
I = b*t^3/12.;  %moment area of inertia
alpha = pi/4;   %half angle of joint intersection angle
ro = 0;
%method 1 specifies joint geometry and d. outputs P and Kl
%method 2 specifies joint geometry, and Kl. outputs Kt and d
%method 3 takes joint geometry and preload, outputting Klin and d
%method 4 takes Klin, P, and flexure L and finds flexure I and d
%method 5 takes Klin, P, and flexure I and finds flexure L and d
%method 6 accepts Klin and d and finds P and joint geometry
%method 7 accepts Klin, P, d, Ktheta, and flexure length and
%finds flexure I
%other methods pending
method = 1; % choose solution method based on what variables knowns and
%unknowns
switch method
    case 1 %method 1 specifies joint geometry and d. outputs P and Kl
        a = 0.65;  %a = d/L %this overconstrains the problem
        d = a*L;
%an initial guess of the stiffness of the CAFP
        Kguess = 4.30792*E*I/(2*L);
        x0 = Kguess;
        options = optimoptions('fmincon','Algorithm','sqp');
```

```
    LB = .1;  %Kt
    UB = 20;  %Kt
    [x fval] = fmincon(@get_Kt,x0,[],[],[],[],LB,UB);
    Kt = x(1);
    P = Kt/(Pi1*d);  %pounds of preload from spring
    Klin = Pi2*P/d; %find the stiffness of the balancing spring
    x0 = 2*d-P/Klin; %find the unstretched spring length
case 2 %method 2 specifies joint geometry, and Kl. outputs Kt and d
    Klin = 8.35; %pounds/inch
    Pguess = 3;
    x0 = Pguess;
    options = optimoptions('fmincon','Algorithm','sqp');
    LB = .01;  %P
    UB = 20;  %P
    x = fmincon(@get_P,x0,[],[],[],[],LB,UB);
    P = x(1);
    V = -P;
    H = 0;
    v = V*L^2*sec(alpha)/(E*I);
    h = H*L^2*csc(alpha)/(E*I);
    beta1 = sqrt((v+h)/8);
    beta2 = sqrt((v-h)/8);
    phi1 = beta1*(coth(beta1)-beta1)+ro^2*beta1^3/(beta1-tanh(beta1));
    phi2 = beta2*(coth(beta2)-beta2)+ro^2*beta2^3/(beta2-tanh(beta2));
    %find an adjusted stiffness that accounts for applied preload
    Kprime = phi1 + phi2;
    %the sping stiffness of the joint with applied preload
    Kt = Kprime*E*I/L;
    d = Pi2*P/Klin;
    x0 = 2*d-P/Klin; %find the unstretched spring length
```

```
case 3 %method 3 takes joint geometry and preload, outputting Klin and d
    P = 4; %lbs
    V = -P;
    H = 0;
    v = V*L^2*sec(alpha)/(E*I);
    h = H*L^2*csc(alpha)/(E*I);
    beta1 = sqrt((v+h)/8);
    beta2 = sqrt((v-h)/8);
    phi1 = beta1*(coth(beta1)-beta1)+ro^2*beta1^3/(beta1-tanh(beta1));
    phi2 = beta2*(coth(beta2)-beta2)+ro^2*beta2^3/(beta2-tanh(beta2));
    %find an adjusted stiffness that accounts for applied preload
    Kprime = phi1 + phi2;
    %the sping stiffness of the joint with applied preload
    Kt = Kprime*E*I/L;
    d = Kt/(P*Pi1);
    Klin = Pi2*P/d;
    x0 = 2*d-P/Klin; %find the unstretched spring length
case 4 %method 4 takes Klin, P, and flexure L and finds flexure I and d
    Klin = 1.3;
    P = 4.2;
    L = 2.596; %specify length
    d = Pi2*P/Klin;
    Kt = Pi1*P*d;
    P = P;
    LB = 0;
    UB = Kt*2*L/(4.30792*E)*10;
    opt = pso(@get_I, LB, UB);
    x = opt{1};
    fval = opt{2};
    I = x(1)
```

```
        x0 = 2*d-P/Klin; %find the unstretched spring length
    case 5 %method 5 takes Klin, P, and flexure I and finds flexure L and d
        Klin = 1.3;
        P = 4.2;
        d = Pi2*P/Klin;
        Kt = Pi1*P*d;
        LB = 0;
        UB = 4.30792*E*I/(2*Kt)*10;
        opt = pso(@get_L, LB, UB);
        x = opt{1};
        fval = opt{2};
        L = x(1)
        x0 = 2*d-P/Klin; %find the unstretched spring length
    case 6 %method 6 accepts Klin and d and finds P and joint geometry
        %one for L
        %one for I
    case 7 %accepts Klin, P, d, Ktheta, and flexure length and finds
%flexure I
        Klin = 1.3;
        P = 4.2;
        L = 3.; %specify length
        d = Pi2*P/Klin;
        Kt = Pi1*P*d;
        LB = 0;
        UB = Kt*2*L/(4.30792*E)*10;
        opt = pso(@get_I, LB, UB);
        x = opt{1};
        fval = opt{2};
        I = x(1)
        x0 = 2*d-P/Klin; %find the unstretched spring length
```

```
end
Kt     %torsional stiffness adjusted for applied compressive load
Klin   %linear stiffness of balancing spring. must be positive
x0     %unstretched length of balancing spring. must be positive
d      %distance from pivot center to spring attachement point
P      %preload resulting from stretched spring
```

This is the file get_Kt.m, used by balanced_spring_design.m.

```
%This funtion is called to solve for stiffness of a
% CAFP with an applied load
function K_diff = get_Kt(x)
global E I L Pi1 d a;
Kguess = x(1);
P = Kguess/(Pi1*d);  %pounds of preload from spring
V = -P;
H = 0;
alpha = pi/4;   %half angle of joint intersection angle
ro = 0;
%we need to non-dimensionalize our vertical and horizontal loads
v = V*L^2*sec(alpha)/(E*I);
h = H*L^2*csc(alpha)/(E*I);
beta1 = sqrt((v+h)/8);
beta2 = sqrt((v-h)/8);
phi1 = beta1*(coth(beta1)-beta1)+ro^2*beta1^3/(beta1-tanh(beta1));
phi2 = beta2*(coth(beta2)-beta2)+ro^2*beta2^3/(beta2-tanh(beta2));
%find an adjusted stiffness that accounts for applied preload
Kprime = phi1 + phi2;% + a*(v*cos(alpha)*sin(10*pi/180))/(10*pi/180);
Kt = Kprime*E*I/L;   %the sping stiffness of the joint with applied preload
K_diff = abs(Kt-Kguess);
end
```

This is the file get_P.m, used by balanced_spring_design.m.

```
%This funtion is called to solve for load on a CAFP with an applied load
function P_diff = get_P(x)
global E I L Pi1 Pi2 d Klin;
        Pguess = x(1); %pounds
        V = -Pguess;
        H = 0;
        alpha = pi/4;   %half angle of joint intersection angle
        ro = 0;
        v = V*L^2*sec(alpha)/(E*I);
        h = H*L^2*csc(alpha)/(E*I);
        beta1 = sqrt((v+h)/8);
        beta2 = sqrt((v-h)/8);
        phi1 = beta1*(coth(beta1)-beta1)+ro^2*beta1^3/(beta1-tanh(beta1));
        phi2 = beta2*(coth(beta2)-beta2)+ro^2*beta2^3/(beta2-tanh(beta2));
        %find an adjusted stiffness that accounts for applied preload
        Kprime = phi1 + phi2;
        %the sping stiffness of the joint with applied preload
        Kt = Kprime*E*I/L;
        d = Kt/(Pguess*Pi1);
        P = Klin*d/Pi2;
        P_diff = abs(Pguess-P);
end
```

This is the file get_I.m, used by balanced_spring_design.m.

```
%This funtion is called to solve for load on a CAFP with an applied load
function I_diff = get_I(x)
global E L Kt P;
      Iguess = x(1); %in^4
      V = -P;
```

```
      H = 0;
      alpha = pi/4;   %half angle of joint intersection angle
      ro = 0;
      v = V*L^2*sec(alpha)/(E*Iguess);
      h = H*L^2*csc(alpha)/(E*Iguess);
      beta1 = sqrt((v+h)/8);
      beta2 = sqrt((v-h)/8);
      phi1 = beta1*(coth(beta1)-beta1)+ro^2*beta1^3/(beta1-tanh(beta1));
      phi2 = beta2*(coth(beta2)-beta2)+ro^2*beta2^3/(beta2-tanh(beta2));
      %find an adjusted stiffness that accounts for applied preload
      Kprime = phi1 + phi2;% + a*(v*cos(alpha)*sin(10*pi/180))/(10*pi/180);
      %the sping stiffness of the joint with applied preload
      Ktheta = Kprime*E*Iguess/L;
      I_diff = abs(Ktheta-Kt);
end
```

This is the file get_L.m, used by balanced_spring_design.m.

```
%This funtion is called to solve for load on a CAFP with an applied load
function L_diff = get_L(x)
global E I Kt P;
      Lguess = x(1); %in
      V = -P;
      H = 0;
      alpha = pi/4;   %half angle of joint intersection angle
      ro = 0;
      v = V*Lguess^2*sec(alpha)/(E*I);
      h = H*Lguess^2*csc(alpha)/(E*I);
      beta1 = sqrt((v+h)/8);
      beta2 = sqrt((v-h)/8);
      phi1 = beta1*(coth(beta1)-beta1)+ro^2*beta1^3/(beta1-tanh(beta1));
```

```
      phi2 = beta2*(coth(beta2)-beta2)+ro^2*beta2^3/(beta2-tanh(beta2));
      %find an adjusted stiffness that accounts for applied preload
      Kprime = phi1 + phi2;
     %the sping stiffness of the joint with applied preload
      Ktheta = Kprime*E*I/Lguess;
      L_diff = abs(Ktheta-Kt);
end
```

A.3 FEA Confirmation

This script builds a cross-axis-flexural pivot with an ideal spring balancer. The spring is designed and preloaded according to the Π group relations to result in a statically balanced flexure.

```
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
pi = acos(-1) !pi
len = 2.5961 !length of flexible segments
thk = .015 !thickness of compliant members
wid = .5  !width of material
alpha = pi/4
cor = len*sin(alpha) !x and y coordinates
!mod = 1.61e7 !modulus of titanium
!mod = 246500. !modulus of SLS nylon
mod = 30.e6 !modulus of steel
mod2 = 10.e6 !modulus of aluminium for rigid sections
nu = 0.3 !PR of steel
thk2 = .125 !thickness of rigid members
I1 = wid*thk**3/12 !Iz for compliant members
I2 = wid*thk2**3/12  !Iz for rigid members
```

```
esz1 = len/20
esz2 = len/8
ang = 40 !20 degrees of rotation
ang = ang*pi/180
Kt = 5.7054
d = 2.7723
Pi1 = 0.49
Pi2 = 0.8581
P = Kt/(Pi1*d)
Klin = P*Pi2/d
x0 = 2*d-P/Klin
/prep7
!element type for flexures
et,1,beam23
keyopt,1,6,0
r,1,thk*wid,I1,thk
!element type for rigid sections
et,2,beam23
keyopt,2,6,0
r,2,thk2*wid,I2,thk2
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
mptemp
mptemp,1,0
mpdata,ex,2,,mod2
mpdata,prxy,2,,nu
!element type for balancing spring
ET,3,COMBIN14
```

```
KEYOPT,3,1,0
KEYOPT,3,2,0
KEYOPT,3,3,0
!enter real constants for balancing spring
R,3,Klin,0,0, , ,x0,
rmore,,
nlgeom,1
k,1,0,0,0 !define keypoints
k,2,cor,cor,0
k,3,0,cor,0
k,4,cor,0,0
k,5,cor/2,cor/2+d,0 !spring attachement point 2
k,6,cor/2, cor/2-d, 0 !spring attachement point 1
lstr,1,2 !line 1 define lines
lstr,3,4 !line 2
lstr,2,5 !line 3
lstr,1,6 !line 4
lstr,5,3 !line 5
lstr,4,6 !line 6
esize,esz1,0
lsel,s,line,,1,2 !selects lines for flexible members
latt,1,1,1
lmesh,all
lsel,all
esize,esz2,0
lsel,s,line,,3,6 !selects lines for rigid members
latt,2,2,1
lmesh,all
lsel,all
!displays mechanism and depicts relative size and shape of beam elements
```

```
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
!we need to find nodes to attach our spring to
ksel, all
ksel,s,kp,,6
nslk,s
*get,spr1,node,,num,max
ksel, all
ksel,s,kp,,5
nslk,s
*get,spr2,node,0,num,max
nsel, all
ksel, all
type,3
real,3
e,spr1,spr2
*get,spring,elem,0,num,maxd
!set boundary conditions
dk,6,all,0,
!d,1,uy,0,,,,uz,
!fk,5,fy,-P !applies force to moving block
nsteps = 40
lswrite,1
*do, count,1,nsteps,1
dk,5,rotz,ang*count/nsteps
nsubst,4,7,3
lswrite,count+1
```

```
*enddo
finish
/solve
LSSOLVE,1,nsteps+1,1
FINISH
/POST1
AVPRIN,0, ,
ETABLE,smxi,NMISC, 1
AVPRIN,0, ,
ETABLE,smxm,NMISC, 3
AVPRIN,0, ,
ETABLE,smxj,NMISC, 5
AVPRIN,0, ,
ETABLE,smni,NMISC, 2
AVPRIN,0, ,
ETABLE,smnm,NMISC, 4
AVPRIN,0, ,
ETABLE,smnj,NMISC, 6
finish
/post26
! Define variables
NSOL,2,spr2,rot,z,rot
rforce,3,spr2,M,z,torque
esol,5,spring,,smisc,1,spr_force
!divide torque by rotation and call resulting variable stiffness
quot,4,3,2,,stiff
PRVAR,rot,torque,stiff,spr_force
```

APPENDIX B. CODES AND SCRIPTS USED IN THE ANALYSIS OF LATTICE FLEXURES

The codes and scripts in this appendix are referenced in Chapter E. In some of the ANSYS scripts, the ellipses (...) are used to indicate that the command continues on the next line. However, ... is not a valid ANSYS command. All ellipses should be removed from the file and the commands consolidated in a text editor so they are all on one line. This will enable the script to run correctly. MatLab scripts, however, should need no such cleanup.

B.1 ANSYS Scripts

Building lattice flexures in finite element code may seem daunting at first due to the geometric complexity, but by building macros to build each lattice type parametrically, the task was greatly simplified. The first script included is on that builds a cross-axis-flexural pivot in ANSYS. To build each flexure it calls a macro that must be located in teh working directory.

```
!This script buils a CAFP with lattice flexures. It requires
!the files x_lat.mac and v_lat.mac in the same directory
!also, change the /cwd command to the directory where this file,
! x_lat.mac, and v_lat.mac are stored.
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
len = 2. !length of flexible segments
cor = len*sin(pi/4) !x and y coordinates
!mod = 320000. !modulus of ABS-M30
```

```
mod = 1.61e7  !modulus of titanium
nu = .34
thk = .04 !*.83 !thickness of compliant members
r = 0.02
n = 10 !number of lattice cells along length use 6 for b and 8 for c
wid = .5   !total width of flexure
b = wid-thk !center-to-center distance
I_l = thk**4/12
I_r = I_l
L1b = (len/(2*n))/b
bigK = 2.25*thk**4
EI_eff = mod*(2*I_r+(I_l*L1b*(L1b**2+1.0)**.5)/(2*(1+nu)*I_l/bigK+L1b**2))
stiff_pred = EI_eff*2.154/len
alphac = atan(b/(len/(n/2))) !the lattice angle for X-type
alphab = atan(b/(len/n)) !the lattice angle for V-type
type = 'v'!may be type x or v
ang = 45 !20 degrees of rotation
ang = ang*pi/180
/prep7
et,1,beam188
!section 1 is for the lattice elements
sectype,1,beam,rect
secdata,thk,thk
sectype,2,beam,rect
secdata,thk*8,thk*8
sectype,3,beam,csolid
secdata,r,,,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,1
```

```
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
!define keypoints
k,1,0,0,b+.05
k,2,0,0,.05
!build a lattice flexure anchored to KPs 1
!and 2 with n divisions and alpha angle
%type%_lat,1,2,n,len,45
k,,0,cor,-.05
k,,0,cor,-.05-b
*get,maxkp,kp,,num,maxd
%type%_lat,maxkp-1,maxkp,n,len,-45
*dim,anchor,array,4 !store KP numbers in array called anchor
*get,maxkp,kp,,num,maxd
anchor(1) = 1
anchor(2) = 2
anchor(3) = maxkp-1
anchor(4) = maxkp
ksel,s,loc,y,cor
*get,topend,kp,,num,max
!create a top section with a point for applying loads
lstr,topend,topend-1
lstr,topend-1,topend-2
lstr,topend-2,topend-3
lstr,topend,topend-3
k,,cor/2,cor/2
lstr,topend,maxkp+1
```

```
allsel
*get,maxl,line,,num,maxd !get the max line defined
esize,,20
lsel,s,line,,1,maxl-5 !selects lines for flexible members
!latt,MAT,REAL,TYPE,ESYS,KB,KE,SECNUM
!use secnum = 3 for round and 1 for square
latt,1,,1,,,,1  !
lsel,inve
!latt,,,2,,,,  !MAT,REAL,TYPE,ESYS,KB,KE,SECNUM
latt,1,,1,,,,2  !
lsel,all
lmesh,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
ksel, all
ksel,s,kp,,topend
nslk,s
*get,end,node,0,num,max
ksel, all
*do,i,1,4,1
dk,anchor(i),all,0,
*enddo
!d,end,rotx,0
!d,end,roty,0
allsel, all
*do, count,1,10,1
```

```
d,end,rotz,ang*count/10
lswrite,count
*enddo
finish
/SOL
!/STATUS,SOLU
LSSOLVE,1,10,1
finish
/post26
! Define variables
NSOL,2,end,rot,z,rot
rforce,3,end,M,z,torque
!divide torque by rotation and call resulting variable stiffness
quot,4,3,2,,stiff
PRVAR,rot,torque,stiff
*get,stiffness,vari,4,rtime,10 !get stiffness value at final load step
kappa = stiffness*len/EI_eff
FINISH
/POST1
/SHOW,WIN32C
SET,FIRST
/PLOPTS,INFO,3
/CONTOUR,ALL,18
/PNUM,MAT,1
/NUMBER,1
/REPLOT,RESIZE
PLNSOL,S,EQV
```

B.1.1 Lattice Macros

This file is x_lat.mac. It accepts several arguments and builds an X-type lattice flexure from the specified keypoints in the given direction

```
!this file builds an X-type lattice flexure connected to the two
!given keypoints angled up at angle theta (degrees), and
!with the specified number of lattice cells at the given lattice
! angle
/pmacro
kp1 = arg1 !number of first keypoint
kp2 = arg2 !number of second keypoint
num_cells = arg3 !number of lattice cells
leng = arg4 !length of flexure desired
theta = arg5 !incline angle
!get location of these keypoints
!x location and y location should be the same for both kp
*get,xloc,kp,kp1,loc,x
*get,yloc,kp,kp1,loc,y
*get,zloc1,kp,kp1,loc,z
*get,zloc2,kp,kp2,loc,z
zoffst = (zloc1+zloc2)/2 !find location for new origin
!define a new coordinate system relative to default global CS
clocal,11,CART,xloc,yloc,zoffst,theta
xpos = 0.0
delx = leng/(2*num_cells)
*do,i,1,n,1
xpos = xpos+delx
k,,xpos,0,0
xpos = xpos+delx
k,,xpos,0,b/2
```

```
k,,xpos,0,-b/2
!xpos = xpos+delx
!k,,xpos,0,0
!xpos = xpos+delx
!k,,xpos,0,b/2
!k,,xpos,0,-b/2
*enddo
*get,maxkp,kp,,num,maxd !get the value of the
!highest defined keypoint and store it as maxkp
!generate lines to represent geometry
*do,i,kp1,maxkp-4,3
lstr,i,i+2
lstr,i+1,i+2
lstr,i,i+3
lstr,i+1,i+4
lstr,i+2,i+3
lstr,i+2,i+4
*enddo
csys,0
```

This file is v_lat.mac. Its operation is similar to x_lat.mac, given above.

```
!this file builds a V-type lattice flexure connected to the two given
!keypoints angled up at angle theta, and with the specified number
!of lattice cells at the given lattice angle
/pmacro
kp1 = arg1 !first keypoint
kp2 = arg2 !second keypoint
num_cells = arg3 !number of cells
leng = len !desired flexure overall length
theta = arg5 !incline angle
```

```
!get location of these keypoints
!x location and y location should be the same for both kp
*get,xloc,kp,kp1,loc,x
*get,yloc,kp,kp1,loc,y
*get,zloc1,kp,kp1,loc,z
*get,zloc2,kp,kp2,loc,z
zoffst = (zloc1+zloc2)/2 !find location for new origin
!define a new coordinate system relative to default global CS
clocal,11,CART,xloc,yloc,zoffst,theta
xpos = 0.0
delx = leng/(num_cells*2)
*do,i,1,n,1
xpos = xpos+delx
k,,xpos,0,b/2
xpos = xpos+delx
k,,xpos,0,-b/2
*enddo
k,,xpos,0,b/2
*get,maxkp,kp,,num,maxd !get the value of the
!highest defined keypoint and store it as maxkp
!generate lines to represent geometry
lstr,kp1,kp2+1
lstr,kp2,kp2+1
lstr,kp2,kp2+2
*do,i,kp1,maxkp-4,2
lstr,i,i+2
lstr,i+1,i+2
lstr,i+1,i+3
lstr,i+2,i+3
*enddo
```

```
!lstr,maxkp-2,maxkp-1
lstr,maxkp-2,maxkp
csys,0
```

B.1.2 Off-Axis Stiffness Analysis

This script finds the off-axis stiffness characteristics of the X-type lattice flexure. It generates results files for a range of lattice angles.

```
!this file analyzes the effect of perforating flexure blades
!as a method of reducing stiffness
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
/PNUM,KP,1   !turn line and keypoint numbering on
/PNUM,LINE,1
pi = acos(-1)  !pi
*dim,angs,array,9,1,1
angs(1) = 22.5
angs(2) = 30.
angs(3) = 37.5
angs(4) = 45.
angs(5) = 52.5
angs(6) = 60.
angs(7) = 67.5
angs(8) = 75.
angs(9) = 82.5
b = .5 !total width of flexure
n = 8. !number of truss units
ndiv = 20.
```

```
/prep7
mod = 1.61e7 !modulus of titanium
nu = .36 !actual value of nu is 0.34, but use 0.36 for consistency
!with other flexure data
nlgeom,on !use non-linear solving
!define material properties
mp,ex,1,mod
mp,prxy,1,nu
*do,index,1,9,1
alpha = angs(index) !angle that defines incline of truss elements
alpha = alpha*pi/180.
nstep = 25
*dim,k_ratio,array,nstep,5,1
*do,j,1,nstep,1
rot = .05 !scale factor for curvature
hoverb = 0.01*j+0.01 !dimensionless parameter
h = b*hoverb
eta = h/(b+h)
r = h/2
!define beam element
et,1,beam188
sectype,1,beam,csolid
secdata,r,,,
sectype,2,beam,rect
secdata,h,h,4,4
et,2,mpc184
keyopt,2,1,1
wid = b/2-r
xpos = 0.0
delx = .5*(b-2*r)/tan(alpha)
```

```
k,,0,0,wid
k,,0,0,-wid
*do,i,1,n,1
xpos = xpos+delx
k,,xpos,0,0
xpos = xpos+delx
k,,xpos,0,wid
k,,xpos,0,-wid
xpos = xpos+delx
k,,xpos,0,0
xpos = xpos+delx
k,,xpos,0,wid
k,,xpos,0,-wid
*enddo
l1overb = delx/b
*get,maxkp,kp,,num,maxd !get the value of the
!highest defined keypoint and store it as maxkp
!generate lines to represent geometry
*do,i,1,maxkp-4,3
lstr,i,i+2
lstr,i+1,i+2
lstr,i,i+3
lstr,i+1,i+4
lstr,i+2,i+3
lstr,i+2,i+4
*enddo
!set mesh size and line attributes
esize,,ndiv
!use secnum 1 for round, 2 for rectangular
latt,1,,1,,,,2  !MAT,REAL,TYPE,ESYS,KB,KE,SECNUM
```

```
lmesh,all
!get node numbers at maxkp and maxkp-1
ksel,s,kp,,maxkp-1,maxkp,1,
nslk,s
*get,n1,node,,num,max
*get,n2,node,,num,min
allsel
!get node numbers to take reactions
ksel,s,kp,,maxkp
nslk,s
*get,end1,node,0,num,max
allsel
ksel,s,kp,,maxkp-1
nslk,s
*get,end2,node,0,num,max
allsel
ksel,s,kp,,1
nslk,s
*get,base1,node,0,num,max
allsel
ksel,s,kp,,2
nslk,s
*get,base2,node,0,num,max
allsel
type,2
e,end1,end2
!define boundary conditions
d,base1,all,0
d,base2,all,0
d,end1,rotz,rot
```

```
lswrite,1
d,end1,rotz,0
!ddele,end1,rotz !remove motion direction constraint
!ddele,end2,rotz
lswrite,2
ddele,end1,rotz !remove motion direction constraint
d,end1,rotx,rot !apply torsional ('support direction' constraint)
!d,end2,rotx,rot/10.
lswrite,3
d,end1,rotx,0
!ddele,end1,rotx !remove torsional constraint
lswrite,4
ddele,end1,rotx !remove torsional constraint
d,end1,roty,rot/1000. !apply transverse bending constraint
lswrite,5
finish
/sol
lssolve,1,5,1,
finish
/post26
numvar,20
rforce,2,end1,m,z,m1
rforce,3,end1,m,x,tor1
rforce,4,end1,m,y,tb1
nsol,5,end1,rot,z,zrot
nsol,6,end1,rot,x,xrot
nsol,7,end1,rot,y,yrot
prvar,m1,tor1,tb1
prvar,xrot,yrot,zrot
*get,momz,vari,2,rtime,1
```

```
*get,thetaz,vari,5,rtime,1
k_tz = momz/thetaz
*get,momx,vari,3,rtime,3
*get,thetax,vari,6,rtime,3
k_tx = momx/thetax
*get,momy,vari,4,rtime,5
*get,thetay,vari,7,rtime,5
k_ty = momy/thetay
k_ratio(j,1) = l1overb
k_ratio(j,2) = eta
k_ratio(j,3) = k_tx
k_ratio(j,4) = k_ty
k_ratio(j,5) = k_tz
fini
/prep7
lclear,all
ldele,all,,,1
*enddo
*CFOPEN,k_rat%index%,txt,,
*VWRITE,k_ratio(1,1),k_ratio(1,2),k_ratio(1,3),k_ratio(1,4),k_ratio(1,5)
(F16.8, F16.8, F16.8, F16.8, F16.8)
*CFCLOS
*enddo
```

This script finds the off-axis stiffness characteristics of the V-type lattice flexure. It generates results files for a range of lattice angles.

```
!this file analyzes the effect of perforating flexure blades
!as a method of reducing stiffness
/cwd, 'C:\ANSYS'
finish
```

```
/clear
!Units in, lbf, psi
/PNUM,KP,1   !turn line and keypoint numbering on
/PNUM,LINE,1
pi = acos(-1)  !pi
*dim,angs,array,9,1,1
angs(1) = 22.5
angs(2) = 30.
angs(3) = 37.5
angs(4) = 45.
angs(5) = 52.5
angs(6) = 60.
angs(7) = 67.5
angs(8) = 75.
angs(9) = 82.5
b = .5 !total width of flexure
n = 8. !number of truss units
ndiv = 20.
/prep7
mod = 1.61e7 !modulus of titanium
nu = .36 !actually 0.34, but use 0.36 for consistency
!with other flexure data
nlgeom,on !use non-linear solving
!define material properties
mp,ex,1,mod
mp,prxy,1,nu
*do,index,1,9,1
alpha = angs(index) !angle that defines incline of truss elements
alpha = alpha*pi/180.
nstep = 25
```

```
*dim,k_ratio,array,nstep,5,1
*do,j,1,nstep,1
rot = .05 !scale factor for curvature
hoverb = 0.01*j+0.01 !dimensionless parameter
h = b*hoverb
eta = h/(b+h)
r = h/2
!define beam element
et,1,beam188
sectype,1,beam,csolid
secdata,r,,,
sectype,2,beam,rect
secdata,h,h,4,4
et,2,mpc184
keyopt,2,1,1
wid = b/2-r
xpos = 0.0
delx = (b-2*r)/tan(alpha)
k,,0,0,wid
k,,0,0,-wid
*do,ii,1,n,1
xpos = xpos+delx
k,,xpos,0,wid
xpos = xpos+delx
k,,xpos,0,-wid
*enddo
k,,xpos,0,wid
l1overb = delx/b
*get,maxkp,kp,,num,maxd !get the value of the
!highest defined keypoint and store it as maxkp
```

```
!generate lines to represent geometry
lstr,1,3
lstr,2,3
lstr,2,4
*do,i,3,maxkp-3,2
lstr,i,i+1
lstr,i,i+2
lstr,i+1,i+2
lstr,i+1,i+3
*enddo
lstr,maxkp-2,maxkp-1
lstr,maxkp-2,maxkp
!set mesh size and line attritbutes
esize,,ndiv
latt,1,,1,,,,2  !MAT, REAL, TYPE, ESYS, KB, KE, and SECNUM
lmesh,all
!get node numbers to take reactions
ksel,s,kp,,maxkp
nslk,s
*get,end1,node,0,num,max
allsel
ksel,s,kp,,maxkp-1
nslk,s
*get,end2,node,0,num,max
allsel
ksel,s,kp,,1
nslk,s
*get,base1,node,0,num,max
allsel
ksel,s,kp,,2
```

```
nslk,s
*get,base2,node,0,num,max
allsel
type,2
e,end1,end2
!define boundary conditions
d,base1,all,0
d,base2,all,0
d,end1,rotz,rot !get 'motion direction' stiffness
lswrite,1
d,end1,rotz,0
lswrite,2
ddele,end1,rotz !remove motion direction constraint
d,end1,rotx,rot !apply torsional ('support direction' constraint)
!d,end2,rotx,rot/10.
lswrite,3
d,end1,rotx,0
!ddele,end1,rotx !remove torsional constraint
lswrite,4
ddele,end1,rotx !remove torsional constraint
d,end1,roty,rot/1000. !apply transverse bending constraint
lswrite,5
finish
/sol
lssolve,1,5,1,
finish
/post26
numvar,20
rforce,2,end1,m,z,m1
rforce,3,end1,m,x,tor1
```

```
rforce,4,end1,m,y,tb1
nsol,5,end1,rot,z,zrot
nsol,6,end1,rot,x,xrot
nsol,7,end1,rot,y,yrot
prvar,m1,tor1,tb1
prvar,xrot,yrot,zrot
*get,momz,vari,2,rtime,1
*get,thetaz,vari,5,rtime,1
k_tz = momz/thetaz
*get,momx,vari,3,rtime,3
*get,thetax,vari,6,rtime,3
k_tx = momx/thetax
*get,momy,vari,4,rtime,5
*get,thetay,vari,7,rtime,5
k_ty = momy/thetay
k_ratio(j,1) = l1overb
k_ratio(j,2) = eta
k_ratio(j,3) = k_tx
k_ratio(j,4) = k_ty
k_ratio(j,5) = k_tz
fini
/prep7
lclear,all
ldele,all,,,1
*enddo
*CFOPEN,k_rat%index%,txt,,
*VWRITE,k_ratio(1,1),k_ratio(1,2),k_ratio(1,3),k_ratio(1,4),k_ratio(1,5)
(F16.8, F16.8, F16.8, F16.8, F16.8)
*CFCLOS
*enddo
```

B.1.3 Analysis of Conventional Blade Flexures

This script generates results similar to the lattice flexure scripts given above, but for a rectangular-cross-section beam, as a comparison.

```
!this file analyzes the effect of perforating flexure blades
!as a method of reducing stiffness
!/cwd, 'C:\Users\emerriam\ANSYS'
finish
/clear
!Units in, lbf, psi
/PNUM,KP,1   !turn line and keypoint numbering on
/PNUM,LINE,1
pi = acos(-1)  !pi
b = .5 !total width of flexure
rot = .05 !scale factor for curvature
len = 3. !length of beam
rot = rot*len
mod = 1.61e7 !modulus of titanium
nu = .36
/prep7
nlgeom,on !use non-linear solving
!define material properties
mp,ex,1,mod
mp,prxy,1,nu
nstep = 25
*dim,k_ratio,array,nstep,5,1
*do,i,1,nstep,1
!i = 20
hoverb = 0.01*i+0.01 !dimensionless parameter
h = b*hoverb
```

```
eta = h/(b+h)
l1overb = len/b
!define beam element
et,1,beam188
sectype,1,beam,rect
secdata,h,b,,,
k,,0,0,0
k,,len,0,0
*get,maxkp,kp,,num,maxd !get the value of the
!highest defined keypoint and store it as maxkp
lstr,1,2
!set mesh size and line attributes
esize,,10
latt,1,,1,,,,1  !MAT, REAL, TYPE, ESYS, KB, KE, and SECNUM
lmesh,all
!get node numbers to take reactions
ksel,s,kp,,maxkp
nslk,s
*get,end1,node,0,num,max
allsel
ksel,s,kp,,1
nslk,s
*get,base1,node,0,num,max
allsel
!define boundary conditions
d,base1,all,0
d,end1,rotz,rot
lswrite,1
d,end1,rotz,0
lswrite,2
```

```
ddele,end1,rotz !remove motion direction constraint
d,end1,rotx,rot !apply torsional ('support direction' constraint)
lswrite,3
d,end1,rotx,0
lswrite,4
ddele,end1,rotx !remove torsional constraint
d,end1,roty,rot/1000. !apply transverse bending constraint)
lswrite,5
finish
/sol
!nsubst,5,10,1
lssolve,1,5,1,
finish
/post26
numvar,20
rforce,2,end1,m,z,m1
rforce,3,end1,m,x,tor1
rforce,4,end1,m,y,tb1
nsol,5,end1,rot,z,zrot
nsol,6,end1,rot,x,xrot
nsol,7,end1,rot,y,yrot
prvar,m1,tor1,tb1
prvar,xrot,yrot,zrot
*get,momz,vari,2,rtime,1
*get,thetaz,vari,5,rtime,1
k_tz = momz/thetaz
*get,momx,vari,3,rtime,3
*get,thetax,vari,6,rtime,3
k_tx = momx/thetax
*get,momy,vari,4,rtime,5
```

```
*get,thetay,vari,7,rtime,5
k_ty = momy/thetay
k_ratio(i,1) = l1overb
k_ratio(i,2) = eta
k_ratio(i,3) = k_tx
k_ratio(i,4) = k_ty
k_ratio(i,5) = k_tz
fini
/prep7
lclear,all
ldele,all,,,1
*enddo
*CFOPEN,k_rat,txt,,
*VWRITE,k_ratio(1,1),k_ratio(1,2),k_ratio(1,3),k_ratio(1,4),k_ratio(1,5)
(F16.8, F16.8, F16.8, F16.8, F16.8)
*CFCLOS
```

B.2 MatLab Scripts

This script runs in MatLab. It reads the results files written by the ANSYS scripts above, finds a surface fit equation to describe the data, and plots the results.

```
clc
clear
set = 2; %use 1 for X type and 2 for V type
filename = cell(1,9);
fullname = cell(1,9);
fsize = 18; %size of font in figures
textfsize = 16;
dataset = [];
for R = 1:9
```

```
filename{R} = sprintf('k_rat%d.txt',R);
switch set
    case 1
        fullname{R} = fullfile('J:','Lattice_flexures','X_OAS',...
            filename{R});
    case 2
        fullname{R} = fullfile('J:','Lattice_flexures','V_OAS',...
            filename{R});
end
dataset = [dataset; load(fullname{R})];
end
baseline = load(fullfile('J:','Lattice_flexures','baseline_beam',...
    'k_rat.txt'));
% data columns are l1overb, eta, ktx, kty, ktz
% z is motion direction
% x is torsion
% y is transverse bending
torfittype = fittype('(a+b*l1b+d*l1b*eta+e*l1b^2+f*eta^2)*l1b^'...
    '(g+h*eta+k*l1b)','independent',{'l1b', 'eta'});
benfittype = fittype('(a+b*eta+d*eta*l1b+e*eta^2+f*l1b^2)*eta^'...
    .(g+h*l1b+k*eta)','independent',{'l1b', 'eta'});
fot = fitoptions(torfittype);
fob = fitoptions(benfittype);
switch set
    case 1
        fot.StartPoint = [0.03 1.0 -0.6 0.3 -0.8 -1.7 0.25 -0.45];
        fob.StartPoint = [0.0 11.0 -4.0 -32.0 0.2 -2.9 0.3 1.7];
    case 2
        fot.StartPoint = [0.01 0.3 -1.8 1.8 -0.05 -2.2 -1.0 -0.2];
        fob.StartPoint = [0.6 43.0 -3.2 -131. -0.026 -2.4 -0.04 4.1];
```

```
end
[fit_xtor gf_xtor] = fit([dataset(:,1), dataset(:,2)], dataset(:,3)./...
    dataset(:,5), torfittype, fot)
[fit_xben gf_xben] = fit([dataset(:,1), dataset(:,2)], dataset(:,4)./...
    dataset(:,5), benfittype, fob)
figure(1);
plot(fit_xtor,[dataset(:,1), dataset(:,2)],dataset(:,3)./dataset(:,5));
switch set
    case 1
        title('\rm \it (k_t/k)_X \rm as a function of \it L_1/b \rm '...
            'and \it \eta \rm','FontName','Times New Roman', 'FontSize',...
            fsize+4)
    case 2
        title('\rm \it (k_t/k)_V \rm as a function of \it L_1/b \rm '...
            .and \it \eta \rm','FontName','Times New Roman', 'FontSize',...
            fsize+4)
end
xlabel('L_1/b','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
ylabel('\eta','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
zlabel('k_t/k','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
ax = gca;
ax.FontName = 'Times New Roman';
ax.FontSize = textfsize;

figure(2)
plot(fit_xben,[dataset(:,1), dataset(:,2)],dataset(:,4)./dataset(:,5));
switch set
```

```
    case 1
        title('\rm \it (k_{b}/k)_X \rm as a function of \it L_1/b '...
            '\rm and \it \eta \rm','FontName','Times New Roman', ...
            'FontSize', fsize+4)
    case 2
        title('\rm \it (k_{b}/k)_V \rm as a function of \it L_1/b '...
            '\rm and \it \eta \rm','FontName','Times New Roman', ...
            'FontSize', fsize+4)
end
% title('(k_{by}/k_{bz})_X as a function of L_1/b and \eta')
xlabel('L_1/b','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
ylabel('\eta','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
zlabel('k_{b}/k','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
% yTicks = get(gca,'ytick');
% minY = min(yTicks);
% verticalOffset = 0.015;
ax = gca;
ax.FontName = 'Times New Roman';
ax.FontSize = textfsize;
% data columns are l1overb, eta, ktx, kty, ktz
% z is motion direction
% x is torsion
% y is transverse bending
% choose a value of l1overb to compare to the baseline flexure
% 0.0974 < l1overb < 2.3659
l1overb = 0.75;
tor_comp = (fit_xtor.a+fit_xtor.b*l1overb+fit_xtor.d*l1overb.*...
```

```
    baseline(:,2)+fit_xtor.e*l1overb^2+fit_xtor.f.*baseline(:,2).^2)...
    .*l1overb.^(fit_xtor.g+fit_xtor.h.*baseline(:,2)+fit_xtor.k*l1overb);
ben_comp = (fit_xben.a+fit_xben.b.*baseline(:,2)+fit_xben.d*l1overb.*...
    baseline(:,2)+fit_xben.e.*baseline(:,2).^2+fit_xben.f.*l1overb.^2)...
    .*baseline(:,2).^(fit_xben.g+fit_xben.h*l1overb+fit_xben.k.*...
    baseline(:,2));
figure(3)
plot(baseline(:,2),baseline(:,3)./baseline(:,5),baseline(:,2),...
    tor_comp,'b.')    %plot k_t/k for a baseline beam
switch set
    case 1
        title('\rm \it(k_t/k)_X\rm Compared to a Blade Flexure',...
            'FontName','Times New Roman', 'FontSize', fsize+4)
    case 2
        title('\rm \it(k_t/k)_V\rm Compared to a Blade Flexure',...
            'FontName','Times New Roman', 'FontSize', fsize+4)
end
xlabel('\eta','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
ylabel('k_t/k','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
leg1 = legend('Rectangular cross-section flexure','Lattice flexure,'...
    '\it L_1/b\rm=0.75');
leg1.FontName='Times New Roman';
leg1.FontSize=16;
ax = gca;
ax.FontName = 'Times New Roman';
ax.FontSize = textfsize;
figure(4)
plot(baseline(:,2),baseline(:,4)./baseline(:,5),baseline(:,2),...
```

```
    ben_comp,'b.')    %plot k_t/k for a baseline beam
xlabel('\eta','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
ylabel('k_b/k','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
axis([min(baseline(:,2)) max(baseline(:,2)) 0 1000]);
switch set
    case 1
        title('\rm \it(k_b/k)_X\rm Compared to a Blade Flexure',...
            'FontName','Times New Roman', 'FontSize', fsize+4)
    case 2
        title('\rm \it(k_b/k)_V\rm Compared to a Blade Flexure',...
            'FontName','Times New Roman', 'FontSize', fsize+4)
end
leg2 = legend('Rectangular cross-section flexure',...
    'Lattice flexure, \it L_1/b\rm=0.75');
leg2.FontName='Times New Roman';
leg2.FontSize=16;
ax = gca;
ax.FontName = 'Times New Roman';
ax.FontSize = textfsize;
ben_ratio = ben_comp./(baseline(:,4)./baseline(:,5));
tor_ratio = tor_comp./(baseline(:,3)./baseline(:,5));
[baseline(:,2), ben_ratio, tor_ratio]
```

This script plots stiffness results and calculates stiffness reduction.

```
clc
clear
%Load all my data
b1 = importdata('baseline.mat');
```

```
x1 = importdata('x1.mat');
x2 = importdata('x2.mat');
x3 = importdata('x3.mat');
x4 = importdata('x4.mat');
x5 = importdata('x5.mat');
x6 = importdata('x6.mat');
x7 = importdata('x7.mat');
v2 = importdata('v2.mat');
v3 = importdata('v3.mat');
v4 = importdata('v4.mat');
v5 = importdata('v5.mat');
v6 = importdata('v6.mat');
v7 = importdata('v7.mat');
names = {'b1', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'v2', 'v3',...
 'v4', 'v5', 'v6', 'v7'};
data = {b1,x1,x2,x3,x4,x5,x6,x7,v2,v3,v4,v5,v6,v7};
%the FEA stiffness reduction results
%coulmns are alpha, x red, v red
FEA_red = [15, 0.6901, 0.765;
    30, 0.7157, 0.7779;
    45, 0.7486, 0.7943;
    60, 0.7822, 0.8111;
    75, 0.8126, 0.8263];
fo = cell(1,14);
go = fo;
rsq = zeros(1,14);
%find curve fits to the torque and use that to calculate stiffness
for R = 1:14
    [fo{R} go{R}] = fit(data{R}(:,1)*pi/180, data{R}(:,2), 'poly1');
    rsq(R) = go{R}.rsquare;
```

```
end
ave_rsq = mean(rsq)
%Find average stiffness values for each flexure type
k = zeros(1,14);
for R = 1:14
    k(R) = fo{R}.p1;
end
kx = zeros(1,7);
kv = zeros(1,6);
for R = 1:7
    kx(R) = fo{R+1}.p1;
end
for R = 1:6
    kv(R) = fo{R+8}.p1;
end
r_x = (k(1)-kx)./k(1);
r_v = (k(1)-kv)./k(1);
%Define the expected reduction in stiffness
alpha = 0:pi/128:pi/2;
b = 0.69;
h = 0.06;
Is = h^4/12;
nu = 0.35;
x_perc_red_sq = zeros(1,length(alpha));
v_perc_red_sq = zeros(1,length(alpha));
    for R = 1:1:length(alpha)
    x_perc_red_sq(R) = 1-24*Is/((b+h)*h^3)*(1+1/(1.185*(1+nu)...
*tan(alpha(R))*sin(alpha(R))+cos(alpha(R))));
    v_perc_red_sq(R) = 1-12*Is/((b+h)*h^3)*(2+1/(1.185*(1+nu)...
*tan(alpha(R))*sin(alpha(R))+cos(alpha(R))));
```

```
    end
    %calculate error between analytic and FEA solution
    for R = 1:1:5
    x_err(R) = (1-24*Is/((b+h)*h^3)*(1+1/(1.185*(1+nu)*tan(FEA_red(R,1)...
*pi/180)*sin(FEA_red(R,1)*pi/180)+cos(FEA_red(R,1)*pi/180)))-...
FEA_red(R,2))/FEA_red(R,2);
    v_err(R) = (1-12*Is/((b+h)*h^3)*(2+1/(1.185*(1+nu)*tan(FEA_red(R,1)...
*pi/180)*sin(FEA_red(R,1)*pi/180)+cos(FEA_red(R,1)*pi/180)))-...
FEA_red(R,3))/FEA_red(R,3);
    end
    x_mean_err = mean(x_err)
    v_mean_err = mean(v_err)
Nsx = [1; 2; 3; 4; 5; 6; 7];
Nsv = [2; 3; 4; 5; 6; 7];
x_alphas = zeros(1,length(Nsx));
v_alphas = zeros(1,length(Nsv));
x_lens = x_alphas;
v_lens = v_alphas;
x_error = x_alphas;
v_error = v_alphas;
for R = 1:1:length(Nsx)
   x_alphas(R) = atan(b/(2/Nsx(R)));
   x_lens(R) = b/(2*tan(x_alphas(R)));
   x_error(R) = ((1-24*Is/((b+h)*h^3)*(1+1/(1.185*(1+nu)*tan(x_alphas(R))...
*sin(x_alphas(R))+cos(x_alphas(R)))))-r_x(R))/(1-24*Is/((b+h)*h^3)*...
(1+1/(1.185*(1+nu)*tan(x_alphas(R))*sin(x_alphas(R))+cos(x_alphas(R)))));
end
for R = 1:1:length(Nsv)
  v_alphas(R) = atan(b/(2/(2*Nsv(R))));
  v_lens(R) = b/tan(v_alphas(R));
```

```
  v_error(R) = ((1-24*Is/((b+h)*h^3)*(1+1/(1.185*(1+nu)*tan(v_alphas(R))...
*sin(v_alphas(R))+cos(v_alphas(R)))))-r_v(R))/(1-24*Is/((b+h)*h^3)*...
(1+1/(1.185*(1+nu)*tan(v_alphas(R))*sin(v_alphas(R))+cos(v_alphas(R)))));
end
figure(6)
plot(0.5./tan(alpha),x_perc_red_sq,'b-')
hold on
plot(1./tan(alpha),v_perc_red_sq,'m-.')
plot(0.5./tan(FEA_red(:,1)*pi/180),FEA_red(:,2),'bd')%unchanged
plot(1./tan(FEA_red(:,1)*pi/180),FEA_red(:,3),'mo')%unchanged
plot(x_lens./b,r_x,'bs')
plot(v_lens./b,r_v,'mv')
xlabel('L_1/b','FontSize',12,'FontName','Times New Roman','FontAngle',...
'italic')
ylabel('Stiffness Reduction','FontSize',12,'FontName','Times New Roman')
axis([0 4 0 1]);
legend1 = legend('X Analytic','V Analytic','X FEA','V FEA','X measured',...
'V measured','Location','SouthEast');
set(legend1,'Location','SouthEast','FontName','Times New Roman');
a=cellstr(num2str(get(gca,'ytick')'*100));
pct = char(ones(size(a,1),1)*'%');
new_yticks = [char(a),pct];
set(gca,'yticklabel',new_yticks,'FontName','Times New Roman')
yTicks = get(gca,'ytick');
minY = min(yTicks);
verticalOffset = 0.03;
grid on
hold off
err = [v_error, x_error];
mean(err)
```

APPENDIX C. CODES AND SCRIPTS USED IN THE ANALYSIS OF COMPOUND JOINTS

The codes and scripts in this appendix are referenced by Chapter 6. In some of the ANSYS scripts, the ellipses (...) are used to indicate that the command continues on the next line. However, ... is not a valid ANSYS command. All ellipses should be removed from the file and the commands consolidated in a text editor so they are all on one line. This will enable the script to run correctly. Matlab scripts, however, should need no such cleanup.

C.1 Center Shift Scripts

C.1.1 ANSYS Scripts

This ANSYS script anayzes the center shift of an array of cartwheel hinges with an angular offset between successive joints, specified by the parameter 'ang_deg'.

```
!cannot be executed with copy/paste,
!must be read in using file/read input from/
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
/PNUM,KP,1
/PNUM,LINE,1
alpha = pi/4
len = 7 !length of flexible segments
cor=len*sin(alpha) !x and y coordinates
boverh = 15
```

```
thk=.08 !thickness of compliant members
ang_deg = 0 !angular offset between flexures
ang = 60*pi/180 !30 degrees of rotation
maxstep = 60
wid = 0.5! boverh*thk  !width of material
!mod = 1.61e7 !modulus of titanium
!mod = 30e6 !modulus of steel
mod = 254600. !modulus of polypropylene
nu = .27
thk2 = .125 !thickness of rigid members
maxn = 4 !maximum number of joints
*dim,shift,array,maxn,4,
esz1 = len/50
/prep7
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
*do,n,1,maxn,1
!n = 2
!maxn = 1
et,1,beam188
sectype,1,beam,rect
secdata,thk,wid,
sectype,2,beam,rect
secdata,thk2,wid,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,0
```

```
z = 0
!Create keypoints and flexible segments
*do,i,1,n,1
k,,-cor,-cor,z !define keypoints
k,,cor,-cor,z
k,,0,0,z
k,,-cor,cor,z
k,,cor,cor,z
z = z-(wid+.0001)
!get the highest defined kp (a multiple of 5)
*get,kpmax,kp,0,num,maxd
lstr,kpmax-4,kpmax-2
lstr,kpmax-3,kpmax-2
lstr,kpmax-1,kpmax-2
lstr,kpmax,kpmax-2
clocal,11,CART,,,,-ang_deg/n
*enddo
!create rigid segments
*do,i,1,kpmax-3,5
lstr,i,i+1
*enddo
*do,i,4,kpmax-1,5
lstr,i,i+1
*enddo
*do,i,4,kpmax-6,10
lstr,i,i+5
*enddo
*if,n,gt,2,then
*do,i,7,kpmax-8,10
lstr,i,i+5
```

```
*enddo
*endif
!create rigid segment to measure deflection of center
!and apply loads. Must be attached to proper node.
*if,mod(n,2),eq,1,then
k,,0,0,z+wid/2+.001
lstr,kpmax,kpmax+1
*else
k,,0,0,z+wid/2+.001
lstr,kpmax-3,kpmax+1
*endif
*get,kpmax,kp,0,num,maxd !get the highest defined kp (a multiple of 5)
esize,esz1,0
lsel,s,line,,1,4*n !selects lines for flexible members
secnum,1
type,1
lmesh,all
lsel,inve
!secnum,2
type,2
lmesh,all
lsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
allsel,all
!select a node where we can apply displacements
```

```
ksel,s,kp,,kpmax
nslk,s
*get,input,node,,num,max
allsel,all
ksel,s,kp,,1
nslk,s
*get,grnd,node,0,num,max
allsel,all
!fix the ground node
ddele,all,all
d,grnd,all,0
*do,i,1,maxstep,1
d,input,rotz,ang*i/maxstep
lswrite,i
*enddo
finish
/SOL
!/STATUS,SOLU
allsel,all
LSSOLVE,1,maxstep,1
FINISH
/post1
!select elements to output max stress
*DIM,mystress,array,maxstep
*DO,inc,1,maxstep,1
SET,inc
PLNSOL,s,eqv
*GET,temp,PLNSOL,0,MAX
mystress(inc)=temp
*ENDDO
```

```
/post26
! Define variables
numvar,20
NSOL,2,input,u,x,dx%n%
NSOL,3,input,u,y,dy%n%
NSOL,4,input,u,z,dz%n%
NSOL,5,input,rot,x,rotx%n%
NSOL,6,input,rot,y,roty%n%
NSOL,7,input,rot,z,rotz%n%
/output,CH_offset_cs%n%,txt,,
PRVAR,dx%n%,dy%n%,dz%n%,rotz%n%
/output
!pull out variables to perform scalar operations
*get,kx%n%,vari,2,rtime,1
*get,ky%n%,vari,3,rtime,1
*get,kz%n%,vari,4,rtime,1
*get,rotz%n%,vari,7,rtime,1
fini
/prep7
lclear,all
ldele,all,,,1
!uncomment the following lines to output data
!*CFOPEN,series_CH_maxs%n%,txt,,
!/output,non_dim_ks%n%,txt,,append,
!*VWRITE,shift(1,1),shift(1,2),shift(1,3),shift(1,4)
!(F12.4,F12.4,F12.4,F12.4)
!*VWRITE,mystress(1)
!%18.6G
!*CFCLOS
!/output
```

```
*del,shift
*del,mystress
*enddo
```

This script is similar to the one above, but analyzes the center shift of arrays of cross-axis-flexural pivots.

```
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
/PNUM,KP,1
/PNUM,LINE,1
alpha = pi/4
len = 10 !length of flexible segments
cor=len*sin(alpha) !x and y coordinates
boverh = 15
thk=.025 !thickness of compliant members
ang_deg = 15.
maxstep = 30.
ang = ang_deg*pi/180 !30 degrees of rotation
wid = 0.5 !8*thk  !width of material
!mod = 1.61e7 !modulus of titanium
mod = 30e6 !modulus of steel
nu = .27
thk2 = .5 !thickness of rigid members
maxn = 4 !maximum number of joints
*dim,shift,array,maxn,4,
esz1 = len/100
/prep7
```

```
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
*do,n,1,maxn,1
!n = 2
!maxn = 1
et,1,beam188
sectype,1,beam,rect
secdata,thk,wid,
sectype,2,beam,rect
secdata,thk2,wid,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,0
z = 0
!Create keypoints and flexible segments
*do,i,1,n,1
k,,-cor/2,-cor/2,z !define keypoints
k,,cor/2,cor/2,z
z = z-wid-.001
k,,-cor/2,cor/2,z
k,,cor/2,-cor/2,z
z = z-wid-.001
*get,kpmax,kp,0,num,maxd !get the highest defined kp
lstr,kpmax-3,kpmax-2
lstr,kpmax-1,kpmax
clocal,11,CART,,,,-ang_deg/n
*enddo
```

```
!create rigid segments
*do,i,2,kpmax-2,4
lstr,i,i+1
*enddo
*do,i,1,kpmax-2,4
lstr,i,i+3
*enddo
*do,i,3,kpmax-4,8
lstr,i,i+4
*enddo
*do,i,8,kpmax-4,8
lstr,i,i+4
*enddo
!create rigid segment to measure deflection of center
!and apply loads. Must be attached to proper node.
*if,mod(n,2),eq,1,then
k,,0,0,z+wid
lstr,kpmax-1,kpmax+1
*else
k,,0,0,z+wid
lstr,kpmax,kpmax+1
*endif
esize,esz1,0
lsel,s,line,,1,2*n !selects lines for flexible members
secnum,1
type,1
lmesh,all
lsel,inve
!secnum,2
type,2
```

```
lmesh,all
lsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
allsel,all
!select a node where we can apply displacements
ksel,s,kp,,kpmax+1
nslk,s
*get,input,node,,num,max
allsel,all
ksel,s,kp,,1
nslk,s
*get,grnd,node,0,num,max
allsel,all
!fix the ground node
ddele,all,all
d,grnd,all,0
*do,i,1,maxstep,1
d,input,rotz,ang*i/maxstep
lswrite,i
*enddo
finish
/SOL
allsel,all
LSSOLVE,1,maxstep,1
FINISH
```

```
/post26
! Define variables
numvar,20
NSOL,2,input,u,x,dx%n%
NSOL,3,input,u,y,dy%n%
NSOL,4,input,u,z,dz%n%
NSOL,5,input,rot,x,rotx%n%
NSOL,6,input,rot,y,roty%n%
NSOL,7,input,rot,z,rotz%n%
/output,CAFP_offset_cs%n%,txt,,
PRVAR,dx%n%,dy%n%,dz%n%,rotz%n%
/output
!pull out variables to perform scalar operations
*get,kx%n%,vari,2,rtime,1
*get,ky%n%,vari,3,rtime,1
*get,kz%n%,vari,4,rtime,1
*get,rotz%n%,vari,7,rtime,1
fini
/prep7
lclear,all
ldele,all,,,1
!*CFOPEN,CAFP_offset_cs%n%,txt,,
!/output,CAFP_offset_cs%n%,txt,,append,
!*VWRITE,shift(1,1),shift(1,2),shift(1,3),shift(1,4)
!(F12.4,F12.4,F12.4,F12.4)
!*CFCLOS
!/output
*del,shift
*enddo
```

C.1.2 MatLab Scripts

This script was used to compare the center shift of compound joints with and without an angular offset.

```
clc
clear
%use 1 for CAFP series, 2 for CAFP parallel, 3 for CH series, 4 for CH
%parallel
fsize = 20; %size of font in figures
textfsize = 16;
dataset = 3;
% for dataset = 1:1:4
switch dataset
%use cat(3,A,B,C...) to concatenate arrays together along the third
%dimension.
    case 1
        length = 10;
        A = importdata('CAFP_slender_15.mat');
        B = importdata('CAFP_offset_slender_15.mat');
        titletext = 'Center Shift of CAFPs in Series';
        filename = 'cafp_s_cs';
        filename2 = 'cafp_s_dl';
    case 3
        length = 7;
        A = importdata('CH_slender_30.mat');
        B = importdata('CH_offset_slender_30.mat');
        titletext = 'Center Shift of CHs in Series';
        filename = 'ch_s_cs';
        filename2 = 'ch_s_dl';
end
```

```
vsum = ((A(:,1,:).^2+A(:,2,:).^2).^.5)./length;
vsumB = ((B(:,1,:).^2+B(:,2,:).^2).^.5)./length;
theta = [0;A(:,4,1)*180/pi];
theta2 = [0;B(:,4,1)*180/pi];
figure(2)
hold on
plot(theta,[0;vsum(:,1,2)],'bs-',theta,[0;vsum(:,1,4)],'md-')
plot(theta2,[0;vsumB(:,1,2)],'gs--',theta2,[0;vsumB(:,1,4)],'rd--')
legend('\it n\rm = 2','\it n\rm = 4','\it n\rm = 2^{offset}',...
'\it n\rm = 4^{offset}','Location','Best')
xlabel('Rotation, degrees','FontName','Times New Roman',...
 'FontSize', fsize)
ylabel('\delta/L','FontName','Times New Roman', 'FontSize',...
 fsize,'FontAngle','italic')
xmin = min(theta);
xmax = max(theta);
ymin = min([0;vsum(:,1,2);vsum(:,1,4);vsumB(:,1,2);vsumB(:,1,4)]);
ymax = max([0;vsum(:,1,2);vsum(:,1,4)]);
axis([xmin xmax ymin ymax])
set(gca,'xtick',0:5:xmax,'FontName','Times New Roman', 'FontSize', fsize)
grid on
set(gcf, 'PaperPositionMode', 'auto')
saveas(gcf,filename2,'eps2c')
```

C.2 Off-Axis Stiffness

The off-axis stiffness behavior was also analyzed.

C.2.1 ANSYS Scripts

This script analyzes series joints composed of cartwheel hinges.

```
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
/PNUM,KP,1
/PNUM,LINE,1
alpha = pi/4
len = 0.75 !length of single spoke 1.0
cor=len*sin(alpha) !x and y coordinates
boverh = 15
thk=.03 !thickness of compliant members 0.04
mod = 1.61e7 !modulus of titanium
!mod = 30e6 !modulus of steel
nu = .27
thk2 = .125 !thickness of rigid members
maxn = 10 !maximum number of joints
esz1 = len/50
/prep7
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
*do,bh,1,boverh,1
!bh = 3
wid=bh*thk  !width of material
*dim,stiff%bh%,array,maxn,7,
*do,n,1,maxn,1
!n = 2
```

```
et,1,beam188
sectype,1,beam,rect
secdata,thk,wid,
sectype,2,beam,rect
secdata,thk2,wid,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,0
z = 0
!Create keypoints and flexible segments
*do,i,1,n,1
k,,0,0,z !define keypoints
k,,2*cor,0,z
k,,cor,cor,z
k,,0,2*cor,z
k,,2*cor,2*cor,z
z = z-(wid+.0001)
*get,kpmax,kp,0,num,maxd !get the highest defined kp (a multiple of 5)
lstr,kpmax-4,kpmax-2
lstr,kpmax-3,kpmax-2
lstr,kpmax-1,kpmax-2
lstr,kpmax,kpmax-2
*enddo
!create rigid segments
*do,i,1,kpmax-3,5
lstr,i,i+1
*enddo
*do,i,4,kpmax-1,5
lstr,i,i+1
*enddo
```

```
*do,i,4,kpmax-6,10
lstr,i,i+5
*enddo
*if,n,gt,2,then
*do,i,7,kpmax-8,10
lstr,i,i+5
*enddo
*endif
!create rigid segment to measure deflection of center
!and apply loads. Must be attached to proper node.
*if,mod(n,2),eq,1,then
k,,cor,cor,z+wid/2+.001
lstr,kpmax,kpmax+1
*else
k,,cor,cor,z+wid/2+.001
lstr,kpmax-3,kpmax+1
*endif
*get,kpmax,kp,0,num,maxd !get the highest defined kp (a multiple of 5)
esize,esz1,0
lsel,s,line,,1,4*n !selects lines for flexible members
secnum,1
type,1
lmesh,all
lsel,inve
!secnum,2
type,2
lmesh,all
lsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
```

```
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
!select a node where we can apply displacements
ksel,s,kp,,kpmax
nslk,s
*get,input,node,,num,max
allsel,all
ksel,s,kp,,1
nslk,s
*get,grnd,node,0,num,max
allsel,all
!fix the ground node
ddele,all,all
d,grnd,all,0
!fix the input node
!d,input,all,0
M = .001
P = .001
!now perturb the input node in one DOF at a time
!d,input,rotz,.5
f,input,fx,P
lswrite,1
fdele,input,fx
f,input,fy,P
lswrite,2
fdele,input,fy
f,input,fz,P
lswrite,3
```

```
fdele,input,fz
f,input,mx,M
lswrite,4
fdele,input,mx
f,input,my,M
lswrite,5
fdele,input,my
f,input,mz,M
lswrite,6
finish
/SOL
allsel,all
LSSOLVE,1,6,1
FINISH
/post26
! Define variables
numvar,20
NSOL,2,input,u,x,dx%n%
NSOL,3,input,u,y,dy%n%
NSOL,4,input,u,z,dz%n%
NSOL,5,input,rot,x,rotx%n%
NSOL,6,input,rot,y,roty%n%
NSOL,7,input,rot,z,rotz%n%
rforce,8,grnd,f,x,fx%n%
rforce,9,grnd,f,y,fy%n%
rforce,10,grnd,f,z,fz%n%
rforce,11,grnd,m,x,mx%n%
rforce,12,grnd,m,y,my%n%
rforce,13,grnd,m,z,mz%n%
!find stiffness in each direction
```

```
quot,14,8,2,,kx%n%
quot,15,9,3,,ky%n%
quot,16,10,4,,kz%n%
quot,17,11,5,,ktx%n%
quot,18,12,6,,kty%n%
quot,19,13,7,,ktz%n%
PRVAR,dx%n%,dy%n%,dz%n%,rotx%n%,roty%n%,rotz%n%
prvar,fx%n%,fy%n%,fz%n%,mx%n%,my%n%,mz%n%
prvar,kx%n%,ky%n%,kz%n%,ktx%n%,kty%n%,ktz%n%
!pull out variables to perform scalar operations
*get,kx%n%,vari,14,rtime,1
*get,ky%n%,vari,15,rtime,2
*get,kz%n%,vari,16,rtime,3
*get,ktx%n%,vari,17,rtime,4
*get,kty%n%,vari,18,rtime,5
*get,ktz%n%,vari,19,rtime,6
stiff%bh%(n,1) = n
stiff%bh%(n,2) = -kx%n%*len**3/(mod*wid*thk**3/12)
stiff%bh%(n,3) = -ky%n%*len**3/(mod*wid*thk**3/12)
stiff%bh%(n,4) = -kz%n%*len**3/(mod*wid*thk**3/12)
stiff%bh%(n,5) = -ktx%n%*len/(mod*wid*thk**3/12)
stiff%bh%(n,6) = -kty%n%*len/(mod*wid*thk**3/12)
stiff%bh%(n,7) = -ktz%n%*len/(mod*wid*thk**3/12)
fini
/prep7
lclear,all
ldele,all,,,1
*enddo
*CFOPEN,CH_non_dim_ks_%bh%,txt,,
*VWRITE,stiff%bh%(1,1),stiff%bh%(1,2),stiff%bh%(1,3),...
```

```
stiff%bh%(1,4),stiff%bh%(1,5),stiff%bh%(1,6),stiff%bh%(1,7)
(F12.4,F12.4,F12.4,F12.4,F12.4,F12.4,F12.4)
*CFCLOS
*enddo
```

This script analyzes the off-axis stiffness for a series-and-parallel joint composed of cross-axis flexural pivots. While series-and-parallel cartwheel hinges and series cross-axis-flexural pivots were also analyzed, those scripts are omitted because they are so similar to the scripts included here.

```
/cwd, 'C:\ANSYS'
finish
/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
/PNUM,KP,1
/PNUM,LINE,1
alpha = pi/4
len = 1 !length of flexible segments
cor=len*sin(alpha) !x and y coordinates
thk=.04 !thickness of compliant members
!angdeg = 0.
ang_deg = angdeg
maxstep = 30.
ang = ang_deg*pi/180 !30 degrees of rotation
!mod = 1.61e7 !modulus of titanium
mod = 30e6 !modulus of steel
nu = .27
thk2 = .125 !thickness of rigid members
maxn = 10 !maximum number of joints
boverhmax = 15
```

```
*do,boverh,1,boverhmax,1
wid=boverh*thk  !width of material
*dim,stiff,array,maxn,7,
esz1 = len/50
/prep7
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
*do,n,1,maxn,1
!n = 3
!maxn = 1
et,1,beam188
sectype,1,beam,rect
secdata,thk,wid,
sectype,2,beam,rect
secdata,thk2,wid,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,0
z = 0
!Create keypoints and flexible segments
*do,i,1,n,1
k,,-cor/2,-cor/2,z !define keypoints
k,,cor/2,cor/2,z
z = z-wid-.001
k,,-cor/2,cor/2,z
k,,cor/2,-cor/2,z
z = z-wid-.001
```

```
*get,kpmax,kp,0,num,maxd !get the highest defined kp
lstr,kpmax-3,kpmax-2
lstr,kpmax-1,kpmax
clocal,11,CART,,,,-ang_deg/n
*enddo
!create rigid segments
*do,i,2,kpmax-2,4
lstr,i,i+1
*enddo
*do,i,1,kpmax-2,4
lstr,i,i+3
*enddo
*do,i,3,kpmax-4,8
lstr,i,i+4
*enddo
*do,i,8,kpmax-4,8
lstr,i,i+4
*enddo
!create rigid segment to measure deflection of center
!and apply loads. Must be attached to proper node.
*if,mod(n,2),eq,1,then
k,,0,0,z+wid
lstr,kpmax-1,kpmax+1
*else
k,,0,0,z+wid
lstr,kpmax,kpmax+1
*endif
esize,esz1,0
lsel,s,line,,1,2*n !selects lines for flexible members
secnum,1
```

```
type,1
lmesh,all
lsel,inve
!secnum,2
type,2
lmesh,all
lsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
allsel,all
!select a node where we can apply displacements
ksel,s,kp,,kpmax+1
nslk,s
*get,input,node,,num,max
allsel,all
ksel,s,kp,,1
nslk,s
*get,grnd,node,0,num,max
allsel,all
!fix the ground node
ddele,all,all
d,grnd,all,0
!fix the input node
!d,input,all,0
M = .001
P = .0001
```

```
!now perturb the input node in one DOF at a time
d,input,ux,P
lswrite,1
ddele,input,ux
d,input,uy,P
lswrite,2
ddele,input,uy
d,input,uz,P
lswrite,3
ddele,input,uz
d,input,rotx,M
lswrite,4
ddele,input,rotx
d,input,roty,M
lswrite,5
ddele,input,roty
d,input,rotz,M
lswrite,6
finish
/SOL
allsel,all
LSSOLVE,1,6,1
FINISH
/post26
! Define variables
numvar,20
NSOL,2,input,u,x,dx%n%
NSOL,3,input,u,y,dy%n%
NSOL,4,input,u,z,dz%n%
NSOL,5,input,rot,x,rotx%n%
```

```
NSOL,6,input,rot,y,roty%n%
NSOL,7,input,rot,z,rotz%n%
rforce,8,grnd,f,x,fx%n%
rforce,9,grnd,f,y,fy%n%
rforce,10,grnd,f,z,fz%n%
rforce,11,grnd,m,x,mx%n%
rforce,12,grnd,m,y,my%n%
rforce,13,grnd,m,z,mz%n%
!find stiffness in each direction
quot,14,8,2,,kx%n%
quot,15,9,3,,ky%n%
quot,16,10,4,,kz%n%
quot,17,11,5,,ktx%n%
quot,18,12,6,,kty%n%
quot,19,13,7,,ktz%n%
PRVAR,dx%n%,dy%n%,dz%n%,rotx%n%,roty%n%,rotz%n%
prvar,fx%n%,fy%n%,fz%n%,mx%n%,my%n%,mz%n%
prvar,kx%n%,ky%n%,kz%n%,ktx%n%,kty%n%,ktz%n%
!pull out variables to perform scalar operations
*get,kx%n%,vari,14,rtime,1
*get,ky%n%,vari,15,rtime,2
*get,kz%n%,vari,16,rtime,3
*get,ktx%n%,vari,17,rtime,4
*get,kty%n%,vari,18,rtime,5
*get,ktz%n%,vari,19,rtime,6
stiff(n,1) = n
stiff(n,2) = -kx%n%*len**3/(mod*wid*thk**3/12)
stiff(n,3) = -ky%n%*len**3/(mod*wid*thk**3/12)
stiff(n,4) = -kz%n%*len**3/(mod*wid*thk**3/12)
stiff(n,5) = -ktx%n%*len/(mod*wid*thk**3/12)
```

```
stiff(n,6) = -kty%n%*len/(mod*wid*thk**3/12)
stiff(n,7) = -ktz%n%*len/(mod*wid*thk**3/12)
fini
/prep7
lclear,all
ldele,all,,,1
*enddo
*CFOPEN,cafp_non_dim_%boverh%,txt,,
!/output,non_dim_ks,txt,,append,
*VWRITE,stiff(1,1),stiff(1,2),stiff(1,3),stiff(1,4),stiff(1,5),...
stiff(1,6),stiff(1,7)
(F12.4,F12.4,F12.4,F12.4,F12.4,F12.4,F12.4)
*CFCLOS
!/output
*del,stiff
*enddo
```

C.2.2 MatLab Scripts

This script analyzes the off-axis stiffness of compound joints with various angular offsets. It first generates results files by calling an ANSYS script, then loading and analyzing the results.

```
clc
clear
angs = 0:5:60;
for R = 1:length(angs)
    ang = angs(R);
command  = ['"C:\Program Files\ANSYS Inc\v150\ANSYS\bin\winx64\ansys150'...
    '.exe"  -p aa_r -dir "C:\ANSYS" -j "shredder" -s read -l en-us '...
    '-angdeg ' num2str(ang) '  -b -i "J:\Compound_Flexures\CAFP\CAFP_'...
    'series_off_axis.txt" -o "C:\ANSYS\file3.out"'];
```

```
% dos(command)
system(['SET KMP_STACKSIZE=2048k & ' command])
%Error code 100 means the jobname is locked
    data = zeros(10,7,15);
    for bh = 1:15
    infile = sprintf('cafp_non_dim_%d.txt',bh);
    f = fullfile('C:','ANSYS',infile);
        data(:,:,bh) = load(f, 'ascii');
    end

    outfile = sprintf('CAFP_OAS_offset_%d.mat', ang);
    save(outfile, 'data', '-mat');
end
fsize = 22; %size of font in figures
% use 1 for CAFP series, 2 for CAFP parallel, 3 for CH series,
% 4 for CH parallel
dataset = 1;
% angs = [0,15,30,45,60];
A = cell(7,length(angs));
switch dataset
    case 1
        for R = 1:length(angs)
            infile = sprintf('CAFP_OAS_offset_%d.mat', angs(R));
            A{1,R} = importdata(infile);
        end
    case 2
        %Load all my data on CAFPs in parallel
        A = importdata('cafp_parallel_nds_ML.mat');
    case 3
        %Load all my data on CHs in series
```

```
        A = importdata('ch_series_nds_ML.mat');
        %Load theta z data on CHs in series
        B = importdata('CW_series.mat');
    case 4
        %Load all my data on CHs in parallel
        A = importdata('ch_parallel_nds_ML.mat');
        %Load theta z data on CHs in series
        B = importdata('CW_parallel.mat');
end
%This is a 3D array. The first index is the number of flexures
%The second index selects which data: [n kx ky kz ktx kty ktz]
%The third index is b/h
%I want to find curve fits that relate the variable kx, ky, etc, to n and
%b/h
n = 1.:1.:10.;
for R = 1:length(angs)
    %put kx in a column vector
    A{2,R} = reshape(permute(A{1,R}(:,2,:),[1,3,2]),[150,1]);
    %put ky in a column vector
    A{3,R} = reshape(permute(A{1,R}(:,3,:),[1,3,2]),[150,1]);
    %put kz in a column vector
    A{4,R} = reshape(permute(A{1,R}(:,4,:),[1,3,2]),[150,1]);
    %put kx in a column vector
    A{5,R} = reshape(permute(A{1,R}(:,5,:),[1,3,2]),[150,1]);
    %put ky in a column vector
    A{6,R} = reshape(permute(A{1,R}(:,6,:),[1,3,2]),[150,1]);
    %put kz in a column vector
    A{7,R} = reshape(permute(A{1,R}(:,7,:),[1,3,2]),[150,1]);
end
% % so I can use it in the fit function
```

```
n = [n,n,n,n,n,n,n,n,n,n,n,n,n,n,n]';
bh = ones(10,1);
for R = 2:1:15
bh = [bh;R*ones(10,1)];
end
%define a custom fittype to use with my surface fitting
% he definition includes a function and a cell array denoting which
% are the independent variables (bh and n). Matlab assumes that the rest
% are the curve fit parameters.
myfittype = fittype('(a*bh^4+b*bh^3+c*bh^2+d*bh+e)*n^(f*bh+g)',...
    'independent',{'bh', 'n'});
fo = fitoptions(myfittype);
fo.StartPoint = [0 -1 30 -10 20 0 -1];
%% This section finds curve fits for all the data.
%rows are kx, ky, kz, ktx, kty, ktz. columns are angles
fits = cell(6,length(angs));
%rows are kx, ky, kz, ktx, kty, ktz. columns are angles
gfs = cell(6,length(angs));
for R = 1:length(angs)
% fo = fitoptions('myfittype','StartPoint',[0 -1 30 -10 20 0 -1]);
%The x dimension is bh, the y dimension is n
[fits{1,R}, gfs{1,R}] = fit([bh, n], A{2,R},myfittype,fo);
%The x dimension is bh, the y dimension is n
[fits{2,R}, gfs{2,R}] = fit([bh, n], A{3,R},myfittype,fo);
 %The x dimension is bh, the y dimension is n
[fits{3,R}, gfs{3,R}] = fit([bh, n], A{4,R},myfittype,fo);
%The x dimension is bh, the y dimension is n
[fits{4,R}, gfs{4,R}] = fit([bh, n], A{5,R},myfittype,fo);
%The x dimension is bh, the y dimension is n
[fits{5,R}, gfs{5,R}] = fit([bh, n], A{6,R},myfittype,fo);
```

```
%The x dimension is bh, the y dimension is n
[fits{6,R}, gfs{6,R}] = fit([bh, n], A{7,R},myfittype,fo);
end
%*****************************************************
%% find an average deviation in the stiffness from the 0-offset stiffness
err = cell(length(angs)-1,6);
for R = 2:length(angs)
    for S = 1:6
        %%find error as a percentage of 0-offset joint
        err{R-1,S} = abs(A{S+1,R}-A{S+1,1})./A{S+1,1}.*100;
    end
end
mean_err = zeros(length(angs)-1,6);
for R = 1:length(angs)-1
    for S = 1:6
    mean_err(R,S) = mean(err{R,S});
    end
end
results = sprintf('CAFP_data_summary.mat');
save(results, 'fits', 'gfs', 'err', 'mean_err', '-mat');
```

This script analyzes the data files assembled by the above script.

```
clc
clear
%use cat(3,A,B,C...) to concatenate arrays together along the third
%dimension.
fsize = 22; %size of font in figures
dataset = 1; %use 1 for CAFP series, 2 for CAFP parallel,
% 3 for CH series, 4 for CH parallel
ang_deg = 15;
```

```
infile = sprintf('CAFP_OAS_offset_%d.mat', ang_deg);
switch dataset
    case 1
        %Load all my data on CAFPs in series
%         A = importdata('cafp_series_nds_ML.mat');
        A = importdata(infile);
    case 2
        %Load all my data on CAFPs in parallel
        A = importdata('cafp_parallel_nds_ML.mat');
    case 3
        %Load all my data on CHs in series
        A = importdata('ch_series_nds_ML.mat');
        %Load theta z data on CHs in series
        B = importdata('CW_series.mat');
    case 4
        %Load all my data on CHs in parallel
        A = importdata('ch_parallel_nds_ML.mat');
        %Load theta z data on CHs in series
        B = importdata('CW_parallel.mat');
end
%This is a 3D array. The first index is the number of flexures
%The second index selects which data: [n kx ky kz ktx kty ktz]
%The third index is b/h
%I want to find curve fits that relate the variable kx, ky, etc, to n and
%b/h
n = 1.:1.:10.;
%put kx in a column vector
kx = reshape(permute(A(:,2,:),[1,3,2]),[150,1]);
%put ky in a column vector
ky = reshape(permute(A(:,3,:),[1,3,2]),[150,1]);
```

```
%put kz in a column vector
kz = reshape(permute(A(:,4,:),[1,3,2]),[150,1]);
%put kx in a column vector
ktx = reshape(permute(A(:,5,:),[1,3,2]),[150,1]);
%put ky in a column vector
kty = reshape(permute(A(:,6,:),[1,3,2]),[150,1]);
%put kz in a column vector
ktz = reshape(permute(A(:,7,:),[1,3,2]),[150,1]);
%so I can use it in the fit function
n = [n,n,n,n,n,n,n,n,n,n,n,n,n,n,n]';
bh = ones(10,1);
for R = 2:1:15
bh = [bh;R*ones(10,1)];
end
%define a custom fittype to use with my surface fitting
%the definition includes a function and a cell array denoting which
% are the independent variables (bh and n). Matlab assumes that the rest
% are the curve fit parameters.
myfittype = fittype('(a*bh^4+b*bh^3+c*bh^2+d*bh+e)*n^(f*bh+g)',...
    'independent',{'bh', 'n'});
fo = fitoptions(myfittype);
fo.StartPoint = [0 -1 30 -10 20 0 -1];
 %The x dimension is bh, the y dimension is n
[fitkx, gfkx] = fit([bh, n], kx,myfittype,fo)
figure(1)
plot(fitkx,[bh,n],kx);
xlabel('b/h','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
ylabel('n','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
```

```
zlabel('\kappa_x','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
set(gca,'xtick',1:2:15,'ytick',1:3:10,'FontName','Times New Roman',...
    'FontSize', fsize)
 %The x dimension is bh, the y dimension is n
[fitky, gfky] = fit([bh, n], ky,myfittype,fo)
figure(2)
plot(fitky,[bh,n],ky);
% title('\kappa_y as a function of n and b/h','FontName',...
% 'Times New Roman', 'FontSize', fsize)
xlabel('b/h','FontName','Times New Roman', 'FontSize', fsize,...
'FontAngle','italic')
ylabel('n','FontName','Times New Roman', 'FontSize', fsize,...
'FontAngle','italic')
zlabel('\kappa_y','FontName','Times New Roman', 'FontSize', fsize,...
'FontAngle','italic')
set(gca,'xtick',1:2:15,'ytick',1:3:10,'FontName','Times New Roman',...
'FontSize', fsize)
 %The x dimension is bh, the y dimension is n
[fitkz, gfkz] = fit([bh, n], kz,myfittype,fo)
figure(3)
plot(fitkz,[bh,n],kz);
% title('\kappa_z as a function of n and b/h','FontName',...
% 'Times New Roman', 'FontSize', fsize)
xlabel('b/h','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
ylabel('n','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
zlabel('\kappa_z','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
```

```
set(gca,'xtick',1:2:15,'ytick',1:3:10,'FontName','Times New Roman',...
    'FontSize', fsize)
 %The x dimension is bh, the y dimension is n
[fitktx, gfktx] = fit([bh, n], ktx,myfittype,fo)
figure(4)
plot(fitktx,[bh,n],ktx);
% title('\kappa_\theta_x as a function of n and b/h','FontName',...
% 'Times New Roman', 'FontSize', fsize)
xlabel('b/h','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
ylabel('n','FontName','Times New Roman', 'FontSize', fsize,'FontAngle',...
    'italic')
zlabel('\kappa_\theta_x','FontName','Times New Roman', 'FontSize', fsize...
    ,'FontAngle','italic')
set(gca,'xtick',1:2:15,'ytick',1:3:10,'FontName','Times New Roman',...
    'FontSize', fsize)
%The x dimension is bh, the y dimension is n
[fitkty, gfkty] = fit([bh, n], kty,myfittype,fo)
figure(5)
plot(fitkty,[bh,n],kty);
% title('\kappa_\theta_y as a function of n and b/h','FontName'...
% ,'Times New Roman', 'FontSize', fsize)
xlabel('b/h','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
ylabel('n','FontName','Times New Roman', 'FontSize', fsize,'FontAngle',...
    'italic')
zlabel('\kappa_\theta_y','FontName','Times New Roman', 'FontSize', fsize...
    ,'FontAngle','italic')
set(gca,'xtick',1:2:15,'ytick',1:3:10,'FontName','Times New Roman',...
    'FontSize', fsize)
```

```
%The x dimension is bh, the y dimension is n
[fitktz, gfktz] = fit([bh, n], ktz,myfittype,fo)
figure(6)
plot(fitktz,[bh,n],ktz);
% title('\kappa_\theta_z as a function of n and b/h','FontName',...
% 'Times New Roman', 'FontSize', fsize)
xlabel('b/h','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
    ,'italic')
ylabel('n','FontName','Times New Roman', 'FontSize', fsize,'FontAngle',...
    'italic')
zlabel('\kappa_\theta_z','FontName','Times New Roman', 'FontSize', fsize...
    ,'FontAngle','italic')
set(gca,'xtick',1:2:15,'ytick',1:3:10,'FontName','Times New Roman',...
    'FontSize', fsize)
%*****************************************************
%   Now I need to find the stiffness values for the experimental data and
%   plot them together with FEA results
b = 0.495; %inches
h = 0.098;
I = b*h^3/12;
L = 1.75;
E = 254000; %psi
fo = cell(1,4);
for N = 1:4
    fo{N} = fit(B(:,1,N)*pi/180, B(:,2,N)*8.850745792, 'poly1');
end
ktz_meas = [fo{1}.p1, fo{2}.p1, fo{3}.p1, fo{4}.p1]*L/(E*I)
ktz_fea = A(:,7,5)
figure(7)
plot([1,2,3,4],ktz_meas,'bo',[1,2,3,4],[ktz_fea(1),ktz_fea(2),...
```

```
ktz_fea(3),ktz_fea(4)],'rs')
xlabel('n','FontName','Times New Roman', 'FontSize', fsize,'FontAngle'...
,'italic')
ylabel('\kappa_\theta_z','FontName','Times New Roman', 'FontSize', fsize...
,'FontAngle','italic')
set(gca,'xtick',1:1:4,'FontName','Times New Roman', 'FontSize', fsize)
```

C.3 Image Processing

This section includes the MatLab code used for measuring the center shift of the physical prototypes. These scripts require the installation of the Computer Vision System Toolbox. The first script reads in a video file, separates it into its frames, and then analyzes the rotation and translation from one frame to the next, and then saves this data.

```
% This script reads in a video file, extracts its frames, then compares
% rotation in each frame
clc
clear
scale = 1.0;
%read in a video file
savename = 'n2CH60_v5.mat';
obj = VideoReader('n2CH60_v2.avi');
folder = sprintf('n2CH60_v2_frames');
% Save each frame as a grayscale image
ii = 1;
while hasFrame(obj)
   img = rgb2gray(readFrame(obj));
   filename = [sprintf('%04d',ii) '.jpg'];
   fullname = fullfile(folder,filename);
   % Write out to a JPEG file (img1.jpg, img2.jpg, etc.)
   imwrite(img,fullname)
```

```
   ii = ii+1;
end
tot_frames = ii-1;
fnamorg = [sprintf('%04d',1) '.jpg'];
full_org = fullfile(folder,fnamorg);
original = imread(full_org);
x = zeros(tot_frames-1,3);
%calibrate distances by grabbing two points, and the center
imshow(original)
[x1,y1] = ginput(3);
dist = sqrt((x1(1)-x1(2))^2+(y1(1)-y1(2))^2);
%multiply by this number to convert pixel distance to inches
inch_convert = .2/dist;
trans = zeros(tot_frames-1,4);
options = optimoptions('fminunc', 'algorithm', 'quasi-newton', 'TolX', ...
    1, 'TolFun', 1);
for R = 2:tot_frames
fnamdist = [sprintf('%04d',R) '.jpg'];
full_dist = fullfile(folder,fnamdist);
distorted = imread(full_dist);
%detect features in both images
ptsOriginal  = detectSURFFeatures(original);
ptsDistorted = detectSURFFeatures(distorted);
%extract feature descriptors
[featuresOriginal,   validPtsOriginal]  = extractFeatures(original, ...
    ptsOriginal);
[featuresDistorted, validPtsDistorted]  = extractFeatures(distorted,...
    ptsDistorted);
%Match the features using their descriptors
indexPairs = matchFeatures(featuresOriginal, featuresDistorted);
```

```
%retrieve locations of corresponding points for each image
matchedOriginal  = validPtsOriginal(indexPairs(:,1));
matchedDistorted = validPtsDistorted(indexPairs(:,2));
% %estimate the transform from one image to the other
[tform, inlierDistorted, inlierOriginal] = estimateGeometricTransform(...
    matchedDistorted, matchedOriginal, 'similarity');
%solve for scale and angle. tx and ty are x and y translations (center
%shift!)
% Let sc = scale*cos(theta)
% Let ss = scale*sin(theta)
% Then, Tinv = [sc -ss  0;
%               ss  sc  0;
%               tx  ty  1]
% where tx and ty are x and y translations, respectively.
Tinv  = tform.invert.T;
ss = Tinv(2,1);
sc = Tinv(1,1);
tx = Tinv(3,1);
ty = Tinv(3,2);
trans(R-1,3) = sqrt(ss*ss + sc*sc); %scale
trans(R-1,4) = atan2(ss,sc)*180/pi; %theta
if R == 2
    trans(R-1,1) = (tx+x1(3)*cos(trans(R-1,4)*pi/180)+y1(3)*...
        sin(trans(R-1,4)*pi/180));
    trans(R-1,2) = (ty+y1(3)*cos(trans(R-1,4)*pi/180)-x1(3)*...
        sin(trans(R-1,4)*pi/180));
else
    trans(R-1,1) = (tx+trans(R-2,1)*cos(trans(R-1,4)*pi/180)+...
        trans(R-2,2)*sin(trans(R-1,4)*pi/180));
    trans(R-1,2) = (ty+trans(R-2,2)*cos(trans(R-1,4)*pi/180)-...
```

```
        trans(R-2,1)*sin(trans(R-1,4)*pi/180));
end
original = distorted;
end
trans(:,1:2) = trans(:,1:2).*inch_convert;
save(savename,'trans', '-mat')
```

Once the above script has been run, the following script analyzes the translation and rotation data and plots it with other data.

```
clc
clear
num = 4;
trans = cell(1,num);
fsize = 20; %size of font in figures
textfsize = 16;
infile1 = sprintf('n1CH_v2.mat');
fullname1 = fullfile('J:','Image_Processing',infile1);
infile2 = sprintf('n2CH_v2.mat');
fullname2 = fullfile('J:','Image_Processing',infile2);
len = 7;
fullname3 = fullfile('J:','Image_Processing','n2CH30_v2.mat');
fullname4 = fullfile('J:','Image_Processing','n2CH60_v2.mat');
names = {fullname1, fullname2, fullname3, fullname4};
%files for comparison are in columns dx,dy,dz,theta
compare_in = sprintf('CH_slender_60.mat');
full_compare = fullfile('J:','Compound_Flexures','Matlab_Analysis',...
    compare_in);
compare = importdata(full_compare);
cs_compare = (compare(:,1,:).^2+compare(:,2,:).^2).^.5;
cs_angle = compare(:,4,:)*180/pi;
```

```
for R = 1:num
    trans{R} = importdata(names{R});
end
x_abs = cell(1,num);
y_abs = cell(1,num);
theta_abs = cell(1,num);
cs = cell(1,num);
remove = cell(1,num);
for R = 1:num
    x_abs{R} = trans{R}(:,1)-trans{R}(1,1);
    y_abs{R} = trans{R}(:,2)-trans{R}(1,2);
    theta_abs{R} = trans{R}(:,4);
    cs{R} = zeros(size(trans{R}(:,1)));
end
for S = 1:num
    for R = 2:length(trans{S}(:,1))
        theta_abs{S}(R) = theta_abs{S}(R-1)+trans{S}(R,4);
        cs{S}(R) = sqrt(x_abs{S}(R).^2+y_abs{S}(R).^2);
    end
end
figure(1)
hold on
plot(compare(:,1,1)/len,compare(:,2,1)/len)
for R = 1:num
    plot(x_abs{R}./len,-y_abs{R}./len)
end
legend('FEA','n=1','n=2','n=2 30', 'n=2 60')

figure(2)
hold on
```

```
plot(cs_angle(:,1),cs_compare(:,1)/len, 'bo')
plot(cs_angle(:,2),cs_compare(:,2)/len, 'ro')
sym = {'b-', 'r--', 'b-.', 'k:'};
for R = 1:num
    plot([0; -theta_abs{R}],[0; cs{R}./len], sym{R})
end
legend('FEA \it n\rm = 1','FEA \it n\rm = 2','\it n\rm = 1',...
    '\it n\rm = 2','\it n\rm = 2, 15{\circ}', '\it n\rm = 2, 30{\circ}'...
    , 'location', 'Best')
xlabel('Rotation, degrees','FontName','Times New Roman', 'FontSize', fsize)
ylabel('\delta/L','FontName','Times New Roman', 'FontSize', fsize,...
    'FontAngle','italic')
set(gca,'xtick',0:15:75,'ytick',0:0.015:0.045,'FontName',...
    'Times New Roman', 'FontSize', fsize)
xmin = 0;
xmax = 75;
ymin = 0;
ymax = 0.045;
axis([xmin xmax ymin ymax])
grid on
```

APPENDIX D. CODES AND SCRIPTS USED IN STATIC BALANCING OF A COMPOUND LATTICE-FLEXURED CAFP

This appendix references codes and scripts used in Chapter 7. In some of the ANSYS scripts, the ellipses (...) are used to indicate that the command continues on the next line. However, ...is not a valid ANSYS command. All ellipses should be removed from the file and the commands consolidated in a text editor so they are all on one line. This will enable the script to run correctly. Matlab scripts, however, should need no such cleanup.

D.1 Determining Load-Dependent Stiffness

The first step in determining the load-dependent stiffness was gathering lots of data, which was best accomplished by automatically analyzing lots and lots of flexures. This script iterates through one hundred geomtries for each lattice type, which are in turn analyzed at different loadings.

```
% This script sends lattice parameters to an FEA script
clc
clear
n = 1:1:10; %number of lattice cells
thk = 1:1:10;   %thickness in hundredths of an inch
nlen = length(n);
tlen = length(thk);
type = 'v'; %may be x or v
filename = 'CAFP_lattice_auto.txt';
fullname = fullfile('J:','SB_Lattice',filename);
for R = 1:nlen
    for S = 2:tlen
```

```
        command  = ['"C:\Program Files\ANSYS Inc\v150\ANSYS\bin\winx64\'...
            'ansys150.exe"  -p aa_r -dir "C:\ANSYS" -j shredder  -s '...
            'read -l en-us -nn ' num2str(n(R)) ' -thk ' num2str(thk(S))...
            ' -type ' type ' -b -i ' fullname ' -o "C:\ANSYS\file3.out"'];
        system(['SET KMP_STACKSIZE=2048k & ' command])
    end
end
```

The above Matlab script instructs ANSYS to execute the following file, using the arguments passed in by Matlab. This script outputs a bunch of results files that contain the load-dependent stiffness bahavior of a CAFP composed of lattice flexures of the prescribed geometry.

```
!This script buils a CAFP with lattice flexures. It requires
!the files x_lat.mac and v_lat.mac in the same directory
!also, change the /cwd command to the directory where this
! file, x_lat.mac, and v_lat.mac are stored.

/cwd, 'C:\ANSYS'
finish
!/clear
!Units in, lbf, psi
!**********************Set parameters *****************
pi = acos(-1) !pi
len = 2. !length of flexible segments
cor = len*sin(pi/4) !x and y coordinates
!thk = 4
!type = 'x'!may be type x or v
!nn = 3.
thk = thk*.01 !*.83 !thickness of compliant members
n = nn !number of lattice cells along length
r = 0.02
```

```
wid = .5   !total width of flexure
b = wid-thk !center-to-center distance
I_l = thk**4/12
I_r = I_l
L1b = (len/(2*n))/b
bigK = 2.25*thk**4/16
eta = thk/(thk+b)
mod = 1.61e7  !modulus of titanium
!mod = 320000. !modulus of ABS-M30
nu = .34
ang = 45 !20 degrees of rotation
ang = ang*pi/180
/prep7
!****************Select element types ******************
et,1,beam188
!section 1 is for the lattice elements
sectype,1,beam,rect
secdata,thk,thk
sectype,2,beam,rect
secdata,thk*3,thk*3
sectype,3,beam,csolid
secdata,r,,,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,1
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
```

```
!***************Define geometry*********************
!define keypoints
k,1,0,0,b+.05
k,2,0,0,.05
!build a lattice flexure anchored to KPs 1 and 2 with n divisions and
! alpha angle
%type%_lat,1,2,n,len,45
k,,0,cor,-.05
k,,0,cor,-.05-b
*get,maxkp,kp,,num,maxd
%type%_lat,maxkp-1,maxkp,n,len,-45
*get,maxkp,kp,,num,maxd
ksel,s,loc,y,cor
*get,maxl,line,,num,maxd !get the max line defined
!******************Mesh geometry ******************************
!lsel,s,line,,1,maxl-5 !selects lines for flexible members
!latt,MAT,REAL,TYPE,ESYS,KB,KE,SECNUM
!use secnum = 3 for round and 1 for square
esize,,20
latt,1,,1,,,,1  !
lmesh,all
type,2
*get,topend,kp,,num,max
ksel,s,kp,,topend
nslk,s
*get,top1,node,0,num,max
allsel, all
ksel,s,kp,,topend-1
nslk,s
*get,top2,node,0,num,max
```

```
allsel, all
ksel,s,kp,,topend-2
nslk,s
*get,top3,node,0,num,max
allsel, all
ksel,s,kp,,topend-3
nslk,s
*get,top4,node,0,num,max
allsel, all
n,,cor/2,cor/2
*get,end,node,,num,maxd
allsel, all
!create a top section with a point for applying loads
e,end,top1
e,end,top2
e,end,top3
e,end,top4
ksel,s,kp,,2
nslk,s
*get,base1,node,0,num,max
allsel, all
ksel,s,kp,,maxkp
nslk,s
*get,base2,node,0,num,max
allsel, all
ksel,s,kp,,maxkp-1
nslk,s
*get,base3,node,0,num,max
allsel, all
e,1,base1
```

```
e,1,base2
e,1,base3
allsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
!***************Calculate EI_eff**********************
!Used to non-dimensionalize loads
*if,type,eq,'x',then
!use evaluation of EI_eff for x-type lattice flexures
EI_eff = 2*mod*(I_r+(2*I_l*L1b*(L1b**2+.25)**.5)/...
((1+nu)*I_l/bigK+2*L1b**2))
*else
!use evaluation of EI_eff for v-type lattice flexures
EI_eff = mod*(2*I_r+(I_l*L1b*(L1b**2+1.0)**.5)/...
(2*(1+nu)*I_l/bigK+L1b**2))
*endif
!******************Set loop parameters*********************
Vstart = -8.0
Vend = 8.0
Vstep = 1. !Increments for V load
Hstart = -6.0
Hend = 6.0
Hstep = 2.0 !Step value for H curves
steps = 3 !number of load steps being used in dk *do loop
lstep = 3 !number of steps when applying force load
array_len = (Vend-Vstart)/Vstep+1
```

```
*dim,kappa,array,array_len,1,1
*dim,nu_load,array,array_len,1,1
*dim,eta_load,array,array_len,1,1
angEnd = 1*pi/180 !degrees of rotation
!******************Start load loop*********************
/SOLU
!changeH = 0.0
Hindex = 1
!Iterate through values of non-dimensional horizontal load
*DO,changeH,Hstart,Hend,Hstep
!Convert dimensionless horizontal force to force in pounds
H = (changeH*EI_eff)/(len**2)
index = 1
! changeV=0.0
!Iterate through values of non-dimensional vertical load
*DO,changeV,Vstart,Vend,Vstep
!Convert dimensionless horizontal force to force in pounds
V = (changeV*EI_eff)/(len**2)
!fix base of flexure in all DOFs
d,1,all,0
d,end,rotz,0
allsel,all
*do,i,1,lstep,1
!apply horizontal load on center of cross-axis flexural pivot
f,end,fx,H*i/lstep
!apply vertical load on center of cross-axis flexural pivot
f,end,fy,V*i/lstep
lswrite,i
*enddo
*DO,i,1,steps,1
```

```
d,end,rotz,angEnd*i/steps
lswrite,i+lstep
*ENDDO
finish
/SOLU
LSSOLVE,1,steps+lstep,1 !Solve all load steps
FINISH
/post26
!Define variables
numvar,20
rforce,3,end,m,z,wrench
NSOL,2,end,rot,z,rot
prvar,rot,wrench
!divide torque by rotation and call resulting variable stiffness
quot,4,3,2,,stiff ,,,1,1
PRVAR,stiff
!get stiffness value at final load step
*get,stiffness,vari,4,rtime,steps+lstep
!convert to dimensionless stiffness
kappa(index) = stiffness*len/EI_eff
nu_load(index) = changeV
eta_load(index) = changeH
index = index+1
FINISH
/solu
fdele,all,all
ddele,all,all
dkdele,all,all
*ENDDO
*CFOPEN,results_H%Hindex%_n%n%_t%thk/0.01%,txt,,
```

```
*VWRITE,eta_load(1,1),nu_load(1,1),kappa(1,1)
(F16.8, F16.8, F16.8)
*CFCLOS
Hindex = Hindex+1
*ENDDO
FINISH
```

The above ANSYS script references x_lat.mac and v_lat.mac, which are included in Appendix E.

Once the data on load-dependent stiffness has been collected, it must be checked visually and saved into an easily accessed format. This is accomplished with the following Matlab script.

```
clc
clear
format short;
close all;
numfiles = 7;
numns = 10;
numts = 10;
len = 2.;
wid = 0.5;
set = 2; %use 1 for X type and 2 for V type
filename = cell(1,numfiles);
fullname = cell(1,numfiles);
fsize = 18; %size of font in figures
textfsize = 16;
importdataset = cell(1,numfiles);
dataset = cell(numns,numts);
%This code block loads files and creates the cell array dataset
%To save time, save dataset in a separate file and comment this block on
%subsequent runs
```

```
for N = 1:numns
    for T = 1:numts
        for R = 1:numfiles
            filename{R} = sprintf('results_H%d_n%d_t%d.txt',R,N,T);
            switch set
                case 1
                    fullname{R} = fullfile('J:','SB_Lattice','X_type',...
                        filename{R});
                case 2
                    fullname{R} = fullfile('J:','SB_Lattice','V_type',...
                        filename{R});
            end
            importdataset{R} = load(fullname{R});
        end
        dataset{N,T} = cat(3,importdataset{1},importdataset{2});
        for R = 3:numfiles
            dataset{N,T} = cat(3,dataset{N,T},importdataset{R});
        end
    end
end
switch set
    case 1
        save(fullfile('J:','SB_Lattice','X_type','X_data.mat'),'dataset');
    case 2
        save(fullfile('J:','SB_Lattice','V_type','V_data.mat'),'dataset');
end
%If the dataset cell array from the block above has been saved, uncomment
%this code block to run faster
switch set
    case 1
```

```
        load(fullfile('J:','SB_Lattice','X_type','X_data.mat'));
    case 2
        load(fullfile('J:','SB_Lattice','V_type','V_data.mat'));
end
%dataset has columns of eta, nu, and kappa (horizontal load, vertical load,
%and stiffness, all dimensionless
        linewidth = 1; %linewidth of graphed lines
%% Creating graph and setting graph properties
for N = 6:7; %numns
    for T = 8:9; %numts
        eta = dataset{N,T}(:,1,:);
        nu = dataset{N,T}(:,2,:);
        kappa = dataset{N,T}(:,3,:);
        figure()
        b = wid-T*0.01;
        l1b = len/(2*N)/b;
        little_eta = T*0.01/(T*0.01+b);
        hold on
        %Change font size of axis labels and title
        fontsize = 16; %Font size for labels
        %Max and min X and Y values to graph
        xmin = min(nu(:,1,1));
        xmax = max(nu(:,1,1));
        ymin = -6;
        ymax = 6;
        %Set bounds of graph that will be displayed
        axis([xmin xmax ymin ymax]);
        %Set major and minor gridline widths
        majorgridlinewidth = 2;
        minorgridlinewidth = 1;
```

```
%Create tick marks on 0 axes in center of graph
ax = gca;
ax.XColor = 'k';
ax.YColor = 'k';
ax.FontSize = fontsize;
ax.FontName = 'Times New Roman';
ax.XTick = xmin:2:xmax;
ax.YTick = ymin:1:ymax;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
ax.XGrid = 'on';
ax.YGrid = 'on';
xlabh = get(gca,'XLabel');
xlabh.Position = [0 ymin 0];
ylabh = get(gca,'YLabel');
ylabh.Position = [xmin 0 0];
xlabel('\nu','FontSize',fontsize,'FontName','Times New Roman',...
    'FontAngle','italic')
ylabel('\kappa','FontSize',fontsize,'FontName','Times New Roman'...
    ,'FontAngle','italic')
switch set
    case 1
        str = ['\rm X-type lattice with \it L_1/b \rm =',...
            num2str(l1b),' and \it \eta \rm = ',...
            num2str(little_eta)];
        title(str,'FontSize',fontsize,'FontName','Times New Roman')
    case 2
        str = ['\rm LD V-type lattice with \it L_1/b \rm =',...
            num2str(l1b),' and \it \eta \rm = ',...
            num2str(little_eta)];
```

```
                title(str,'FontSize',fontsize,'FontName','Times New Roman')
        end
        %This loop plots all the data from the FEA results.
        place = 0;
        for i = 1:numfiles
            plot(dataset{N,T}(:,2,i),dataset{N,T}(:,3,i),'--.',...
                'LineWidth',linewidth)
            str = ['\eta = ',num2str(dataset{N,T}(1,1,i))];
            text(dataset{N,T}(3+place,2,i),dataset{N,T}(3+place,3,i),...
                str,'FontSize',fontsize,'FontName','Times New Roman');
            place = place+1;
        end
    end
end
```

Finally, a surface fit for the data is found using a non-linear regression.

```
clc
clear
set = 1;    %use 1 for X-type and 2 for V-type
len = 2.;
wid = 0.5;
switch set
    case 1
        load(fullfile('J:','SB_Lattice','X_type','X_data.mat'));
    case 2
        load(fullfile('J:','SB_Lattice','V_type','V_data.mat'));
end
[N, T] = size(dataset);
[etas, nums, nus] = size(dataset{1,1});
vec_len = N*T*nus*etas;
```

```
Y = zeros(vec_len, 1);
X = zeros(vec_len,5);
index = 1;
for n = 1:N
    for t = 1:T
        for v = 1:nus
            for h = 1:etas
                b = wid-t*0.01;
                l1b = len/(2*n)/b;
                little_eta = t*0.01/(t*0.01+b);
                Y(index) = dataset{n,t}(h,3,v);
                X(index,1) = dataset{n,t}(h,1,v);   %horizontal load
                X(index,2) = dataset{n,t}(h,2,v);   %vertical load
                X(index,3) = l1b;                   %L1 over b
                X(index,4) = little_eta;            %aspect ratio
                X(index,5) = 1;                     %constant
                index = index+1;
            end
        end
    end
end
beta = ones(1, 5);
modelfun = @(beta,X) X(:,2).*beta(1)+X(:,3).*beta(2)+X(:,5).*beta(3)+...
    X(:,1).^2.*beta(4)+X(:,2).^2.*beta(5);
[BETA, R, J, COVB, MSE] = nlinfit(X,Y,modelfun,beta);
BETA
MSE
%   Find an R^2 value for this non-linear regression
SS_reg = sum((modelfun(BETA,X)-Y).^2);
SS_tot = sum(((Y-mean(Y)).^2));
```

```
R_sqr = 1-SS_reg/SS_tot
switch set
    case 1
        save(fullfile('J:','SB_Lattice','X_type','beta_X.mat'),'BETA');
    case 2
        save(fullfile('J:','SB_Lattice','V_type','beta_V.mat'),'BETA');
end
```

D.2 System Design

Now that the load-dependent behavior of the lattice-flexured CAFP is known, we can design an ideal static balancing system for it. The following Matlab script explores the design space for balancing lattice-flexured CAFPs for given geometries and limits on *P*.

```
clc
clear
type = 2;   %use 1 for X-type and 2 for V-type
% Pi group values
Pi1 = 0.49; %k_t/Pd
Pi2 = 0.8581;   %k_l*d/P
%material parameters
E = 1.61e7;
nu = 0.342;
%lattice parameters
len = 3.0;
n = 5;
t = 0.04;
wid = 1.0;
b = wid-t;
L1b = len/(2*n)/b;
little_eta = t/(t+b);
```

```
I_l = t^4/12;
I_r = I_l;
L1 = len/n;
bigK = 2.25*t^4/16;
%stiffness without applied loads
switch type
    case 1
%use evaluation of EI_eff for x-type lattice flexures
K_free = 2*E*(I_r+(2*I_l*L1b*(L1b^2+.25)^.5)/((1+nu)*I_l/bigK+2*...
        L1b^2))/len*2.154;
    case 2
%use evaluation of EI_eff for v-type lattice flexures
K_free = E*(2*I_r+(I_l*L1b*(L1b^2+1.0)^.5)/(2*(1+nu)*I_l/bigK+...
        L1b^2))/len*2.154;
end
switch type
    case 1
%use evaluation of EI_eff for x-type lattice flexures
EI_eff = 2*2*E*(I_r+(2*I_l*L1b*(L1b^2+.25)^.5)/((1+nu)*I_l/bigK+2*...
        L1b^2));
    load(fullfile('J:','SB_Lattice','X_type','beta_X.mat'));
    case 2
%use evaluation of EI_eff for v-type lattice flexures
EI_eff = 2*E*(2*I_r+(I_l*L1b*(L1b^2+1.0)^.5)/(2*(1+nu)*I_l/bigK+...
        L1b^2));
    load(fullfile('J:','SB_Lattice','V_type','beta_V.mat'));
end
% kappa_free = K_free*len/EI_eff
P = 0:0.0025:8;
P_nd = -P.*len^2/EI_eff; %calculate non-dimensional compressive load P
```

```
%dimensionless stiffness due to loads and lattice geometry
stiff_fun = @(beta,X) X(2).*beta(1)+X(3).*beta(2)+X(4).*beta(3)+...
    X(1).^2.*beta(4)+X(2).^2.*beta(5);
kappa = zeros(1,length(P));
for R = 1:length(P)
    X = [0, P_nd(R), L1b, 1];
    kappa(R) = stiff_fun(BETA,X);
end
K = kappa*EI_eff/len;
d = K./(P.*Pi1);
K_l = Pi2.*P./d;
fontsize = 18;
figure(1)
plot(P,K)
grid on
ax = gca;
ax.XColor = 'k';
ax.YColor = 'k';
ax.FontSize = fontsize;
ax.FontName = 'Times New Roman';
ax.XTick = min(P):1:max(P);
ax.YTick = floor(min(K)):1:ceil(max(K));
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
ax.XGrid = 'on';
ax.YGrid = 'on';
xlabel('\it P\rm, lbs','FontSize',fontsize,'FontName','Times New Roman',...
    'FontAngle','italic')
ylabel('\it k_{\theta}\rm, in-lbs','FontSize',fontsize,'FontName',...
    'Times New Roman','FontAngle','italic')
```

```
title('\rmLD stiffness \itk_{\theta}\rm as function of \itP')
filename = fullfile('J:','SB_lattice_paper','k_theta');
set(gcf, 'PaperPositionMode', 'auto')
saveas(gcf,filename,'eps2c')
figure(2)
plot(P,K_l)
grid on
ax = gca;
ax.XColor = 'k';
ax.YColor = 'k';
ax.FontSize = fontsize;
ax.FontName = 'Times New Roman';
ax.XTick = min(P):1:max(P);
ax.YTick = floor(min(K_l)):0.25:ceil(max(K_l));
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
ax.XGrid = 'on';
ax.YGrid = 'on';
xlabel('\it P\rm, lbs','FontSize',fontsize,'FontName','Times New Roman',...
    'FontAngle','italic')
ylabel('\it k_{l}\rm, lbs/in','FontSize',fontsize,'FontName',...
    'Times New Roman','FontAngle','italic')
title('\rm Balancer linear stiffness \itk_l\rm as function of \itP')
filename = fullfile('J:','SB_lattice_paper','k_l');
set(gcf, 'PaperPositionMode', 'auto')
saveas(gcf,filename,'eps2c')
figure(3)
plot(P,d)
grid on
ax = gca;
```

```
ax.XColor = 'k';
ax.YColor = 'k';
ax.FontSize = fontsize;
ax.FontName = 'Times New Roman';
ax.XTick = min(P):1:max(P);
ax.YTick = 0:2:20;
axis([min(P), max(P), 0, 20]);
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
ax.XGrid = 'on';
ax.YGrid = 'on';
xlabel('\it P\rm, lbs','FontSize',fontsize,'FontName','Times New Roman',...
    'FontAngle','italic')
ylabel('\it d\rm, in','FontSize',fontsize,'FontName','Times New Roman',...
    'FontAngle','italic')
title('\rmDistance \itd\rm as function of \itP')
filename = fullfile('J:','SB_lattice_paper','d');
set(gcf, 'PaperPositionMode', 'auto')
saveas(gcf,filename,'eps2c')
% The most compact joint results when P is as large as practical
load = P(length(P))
dist = d(length(P))
K_lin = K_l(length(P))
K_t_prime = stiff_fun(BETA,[0,0,L1b,1])
K_t = K(length(P))
```

D.3 Balancer Design

Now that we know what the stiffness of the balancer should be at a chosen preload, we can move on to the design of the balancer itself. the following Matlab script performs a tiered optimization strategy. First an optimal balancer design is found using gradient based methods and

a pseudo-rigid-body model of the proposed balancer. Then a finite element model is used to do a second pass at the optimization routine. The second routine uses the result of the first as its starting point. The files nonlcon.m and fit_.m are included following this script, and are necessary for the optimization.

```
% This script calculates geometric parameters for a printable balancer
% based on the constant force mechanism by Kyler Tolman.
% We use the PRBM to find parameters that will then be refined by an FEA
% model.
clc
clear
P_targ = 8.0;
k_l_targ = 1.7228;
smax = 60.e3;   %max stress (psi) allowed
E = 1.61e7; %modulus of titanium
K_theta = 2.56;
gamma = 0.85;
% E = 480000.;    % psi
bs = [0.11811, 0.11811];
hs = [0.023622, 0.03937];
lens = [1.89, 1.83]*2;
getrs = @(lens)gamma.*lens;
getr2 = @(ls)gamma.*ls(1);
getr3 = @(ls)gamma.*ls(2);
rs = getrs(lens);
t_nauts = [asin(-getr3(ls)*sin(255.5*pi/180)/getr2(ls))*180/pi, 255.5];
thet_nauts = t_nauts.*pi/180;
stiff = @(b,h,l)2*gamma*K_theta*E*b*h^3/12/l;
get_K1 = @(bs, hs, lens)2*stiff(bs(1),hs(1),lens(1));
get_K2 = @(bs, hs, lens)4*stiff(bs(2),hs(2),lens(2));
% Ks = [get_K1(bs, hs, lens), get_K2(bs, hs, lens)];
```

```
gett2 = @(theta, ls) asin(-getr3(ls)*sin(theta)/getr2(ls));
% F_out = @(thetas, Ks, rs)...
F_out = @(thetas, bs, hs, ls, t30)...
    2*(get_K1(bs, hs, ls).*(thetas(1)-gett2(t30,ls))*(-cos(thetas(2))/...
    (getr2(ls)*sin(thetas(1)-thetas(2))))+...
    get_K2(bs, hs, ls).*(thetas(2)-t30)*(cos(thetas(1))/(getr3(ls)*...
    sin(thetas(1)-thetas(2)))));
dist = @(t3, ls, t30) getr2(ls)*(cos(gett2(t3, ls))-cos(gett2(t30,ls)))+...
    getr3(ls)*(cos(t3)-cos(t30));
%**********define functions for taking the derivative of force
dt2dt3 = @(t3, ls) -getr3(ls)*cos(t3)/(getr2(ls)*sqrt(1-getr3(ls)^2*...
    sin(t3)^2/getr2(ls)^2));
dxdt3 = @(t3, ls) -getr2(ls)*sin(gett2(t3, ls))*dt2dt3(t3, ls)-...
    getr3(ls)*sin(t3);
dFdt3 = @(t3, bs, hs, ls, t30) 2*(get_K1(bs, hs, ls)*(-dt2dt3(t3, ls)*...
    cos(t3)/(getr2(ls)*sin(gett2(t3, ls)-t3))+...
    (gett2(t3, ls)-gett2(t30,ls))*(sin(t3)/(getr2(ls)*...
    sin(gett2(t3, ls)-t3))+cos(t3)*cos(gett2(t3, ls)-t3)*...
    (dt2dt3(t3, ls)-1)/(getr2(ls)*sin(gett2(t3, ls)-t3)^2)))+...
    get_K2(bs, hs, ls)*(cos(gett2(t3, ls))/(getr3(ls)*...
    sin(gett2(t3, ls)-t3))+(t3-t30)*(-sin(gett2(t3, ls))*dt2dt3(t3, ls)/...
    (getr3(ls)*sin(gett2(t3, ls)-t3))-...
    cos(gett2(t3, ls))*cos(gett2(t3, ls)-t3)/(getr3(ls)*...
    sin(gett2(t3, ls)-t3)^2)*(dt2dt3(t3, ls)-1))));
dFdx = @(t3, bs, hs, ls, t30) dFdt3(t3, bs, hs, ls, t30)/dxdt3(t3, ls);
%*****************************************************
%check with numeric derivative
theta_3 = thet_nauts(2):pi/1000:thet_nauts(2)+pi/6;
delx = zeros(1,length(theta_3));
Force = zeros(1,length(theta_3));
```

```
theta_2 = zeros(1,length(theta_3));
for R = 1:length(delx)
    theta_2(R) = gett2(theta_3(R), lens);
    delx(R) = dist(theta_3(R), lens, thet_nauts(2));
    Force(R) = F_out([theta_2(R), theta_3(R)], bs, hs, lens, ...
        thet_nauts(2));
end
dtheta2 = zeros(length(theta_3),1);
dx = dtheta2;
dF = dtheta2;
ddFdx = dtheta2;
num_dtheta2 = dtheta2;
num_dx = dtheta2;
num_dF = dtheta2;
num_dFdx = dtheta2;
for R = 1:length(theta_3)-1
    dtheta2(R) = dt2dt3(theta_3(R), lens);
    dx(R) = dxdt3(theta_3(R), lens);
    dF(R) = dFdt3(theta_3(R), bs, hs, lens, thet_nauts(2));
    ddFdx(R) = dFdx(theta_3(R), bs, hs, lens, thet_nauts(2));
    num_dtheta2(R) = (theta_2(R+1)-theta_2(R))/(theta_3(R+1)-theta_3(R));
    num_dx(R) = (delx(R+1)-delx(R))/(theta_3(R+1)-theta_3(R));
    num_dF(R) = (Force(R+1)-Force(R))/(theta_3(R+1)-theta_3(R));
    num_dFdx(R) = (Force(R+1)-Force(R))/(delx(R+1)-delx(R));
end
errt2 = mean((dtheta2-num_dtheta2).^2)^.5;
errx = mean((dx-num_dx).^2)^.5;
errF = mean((dF-num_dF).^2)^.5;
errdFdx = mean((ddFdx-num_dFdx).^2)^.5;
%*******************************************************
```

```
% fitness = @([t3, b1, b2, h1, h2, l1, l2, t03])...
fitness1 = @(X)...
    (F_out([gett2(X(1), [X(6), X(7)]), X(1)], [X(2), X(3)], [X(4), ...
    X(5)], [X(6), X(7)], X(8))/P_targ-1)^2+...
    (dFdx(X(1), [X(2), X(3)], [X(4), X(5)], [X(6), X(7)], X(8))/...
    k_l_targ-1)^2;
%this fitness1 function seems to work pretty well, but I would like a
%nearly constant stiffness for some range. So I will add a term to the
%fitness function that computes the numeric derivative of dfdx as part of
%the fitness function
dh = 0.01;
fitness2 = @(X) 0.25*((dFdx(X(1)+dh, [X(2), X(3)], [X(4), X(5)], [X(6),...
    X(7)], X(8))-...
    dFdx(X(1)-dh, [X(2), X(3)], [X(4), X(5)], [X(6), X(7)], X(8)))/...
    (2*dh))^2;
% It seems that allowing the PRBM to also consider stress allowable is
% necessary. The FEA model is too slow and cumbersome; it doesn't handle
% the stress condition very well.
NONLCON = @(X)nonlcon(X,E,gamma,K_theta,smax);
fitness = @(X)fitness1(X)+fitness2(X);
X0 = [4.7169    0.7    0.2    0.04    0.04    4.01    3.9    4.59];
LB = [thet_nauts(2), 0.10, 0.01, 0.04, 0.04, 1.0, 1.0, 225*pi/180];
UB = [thet_nauts(2)+pi/6, 1.0, 0.70, 0.06, 0.06, 6.0, 6.0, 3*pi/2];
options = optimoptions('fmincon');
options.TolX = 1.0e-12;
[X, FVAL] = fmincon(fitness, X0,[],[],[],[],LB,UB,NONLCON,options);
X;
FVAL;
t3_opt = X(1);
bs_opt = [X(2) X(3)];
```

```
hs_opt = [X(4) X(5)];
ls_opt = [X(6) X(7)];
t30_opt = X(8);
% theta_3 = thet_nauts(2):pi/1000:thet_nauts(2)+pi/6;
% delx = zeros(1,length(theta_3));
% Force = zeros(1,length(theta_3));
% theta_2 = zeros(1,length(theta_3));
for R = 1:length(delx)
    theta_2(R) = gett2(theta_3(R), ls_opt);
    delx(R) = dist(theta_3(R), ls_opt, t30_opt);
    Force(R) = F_out([theta_2(R), theta_3(R)], bs_opt, hs_opt, ls_opt,...
        t30_opt);
end
P_out = F_out([gett2(X(1), ls_opt), X(1)], bs_opt, hs_opt, ls_opt,...
    t30_opt);
dFdX_out = dFdx(X(1), bs_opt, hs_opt, ls_opt, t30_opt);
fontsize = 18;
figure(1)
hold on
plot(delx,Force, 'b-', dist(X(1), ls_opt, t30_opt),...
    F_out([gett2(X(1), ls_opt), X(1)], bs_opt, hs_opt, ls_opt, ...
    t30_opt), 'ro')
axis([-0.6, 0.6, 2, P_targ+6])
grid on
ax = gca;
ax.XColor = 'k';
ax.YColor = 'k';
ax.FontSize = fontsize;
ax.FontName = 'Times New Roman';
% ax.XTick = min(P):1:max(P);
```

```
% ax.YTick = floor(min(delx)):0.25:ceil(max(K_l));
ax.XAxisLocation = 'origin';
% ax.YAxisLocation = 'origin';
ax.XGrid = 'on';
ax.YGrid = 'on';
axis([0,0.5,6,10])
xlabel('\it \deltax\rm, in','FontSize',fontsize,'FontName',...
    'Times New Roman')
ylabel('\itP\rm, lbs','FontSize',fontsize,'FontName','Times New Roman')
title('\rm Balancer Force-Displacement Curve')
filename = fullfile('J:','SB_lattice_paper','bal_opt_ML');
set(gcf, 'PaperPositionMode', 'auto')
saveas(gcf,filename,'eps2c')
%*****************************************************
%write physical parameters to file params.txt
fileID = fopen(fullfile('params_base.txt'),'w');
fprintf(fileID,'mod = %0.5g\n',E);
fprintf(fileID,'len1 = %0.5g\n',ls_opt(1));
fprintf(fileID,'len2 = %0.5g\n',ls_opt(2));
fprintf(fileID,'b1 = %0.5g\n',bs_opt(1));
fprintf(fileID,'b2 = %0.5g\n',bs_opt(2));
fprintf(fileID,'h1 = %0.5g\n',hs_opt(1));
fprintf(fileID,'h2 = %0.5g\n',hs_opt(2));
fprintf(fileID,'theta_02 = %0.5g\n',gett2(t30_opt,ls_opt));
fprintf(fileID,'theta_03 = %0.5g\n',t30_opt);
fprintf(fileID,'P_targ = %0.5g\n',P_targ);
fprintf(fileID,'k_l_targ = %0.5g\n',k_l_targ);
fclose(fileID);
% type params.txt %show file contents
fileID = fopen(fullfile('params.txt'),'w');
```

```
fprintf(fileID,'mod = %0.5g\n',E);
fprintf(fileID,'len1 = %0.5g\n',ls_opt(1));
fprintf(fileID,'len2 = %0.5g\n',ls_opt(2));
fprintf(fileID,'b1 = %0.5g\n',bs_opt(1));
fprintf(fileID,'b2 = %0.5g\n',bs_opt(2));
fprintf(fileID,'h1 = %0.5g\n',hs_opt(1));
fprintf(fileID,'h2 = %0.5g\n',hs_opt(2));
fprintf(fileID,'theta_02 = %0.5g\n',gett2(t30_opt,ls_opt));
fprintf(fileID,'theta_03 = %0.5g\n',t30_opt);
fprintf(fileID,'P_targ = %0.5g\n',P_targ);
fprintf(fileID,'k_l_targ = %0.5g\n',k_l_targ);
fclose(fileID);
%*******************************************
%run the ansys model with the parameters calculate above
% "C:\Program Files\ANSYS Inc\v150\ANSYS\bin\winx64\ansys150.exe"  -p '...
%'aa_r -np 2 -dir "C:\ANSYS" -j "print_bal" -s read -l en-us -b -i '...
%'"J:\SB_Lattice\Printable_Balancer\print_bal.txt" -o "C:\ANSYS\file.out"
% command  = ['"C:\Program Files\ANSYS Inc\v150\ANSYS\bin\winx64\'...
%'ansys150.exe"  -p aa_r -dir "C:\ANSYS" -j shredder  -s read -l en-us'...
%'-nn ' num2str(n(R)) ' -thk ' num2str(thk(S)) ' -type ' type ' -b -i ' ...
%fullname ' -o "C:\ANSYS\file3.out"'];
command = ['"C:\Program Files\ANSYS Inc\v150\ANSYS\bin\winx64\ansys150'...
    '.exe" -p aa_r -np 2 -dir "C:\ANSYS" -j "print_bal_graph" -s read '...
    '-l en-us -b -i "J:\SB_Lattice\Printable_Balancer\print_bal_graph.'...
    'txt" -o "C:\ANSYS\file_graph.out"'];
sys = @()system(['SET KMP_STACKSIZE=2048k & ' command]);
if exist(fullfile('C:','ANSYS','print_bal_graph.lock'), 'file')
    delete(fullfile('C:','ANSYS','print_bal_graph.lock'))
end
sys();
```

```
%read the results file and plot the results for comparison
import_data = @()load(fullfile('J:','SB_Lattice','Printable_Balancer',...
    'ANS_out.txt'));
% columns are displacement, force, dF/dX, and d^2F/dx^2
data = import_data();
plot(data(:,1),data(:,2),'g:')
fit_fun2 = @(X) fit_fun(X, P_targ, k_l_targ, E, smax);
X0 = [ls_opt, bs_opt, hs_opt, t30_opt];
% l1, l2, b1, b2, h1, h2, t03
% t3, b1, b2, h1, h2, l1, l2, t03
LB = [LB(6), LB(7), LB(2), LB(3), LB(4), LB(5), LB(8)];
UB = [UB(6), UB(7), UB(2), UB(3), UB(4), UB(5), UB(8)];
options.TypicalX = X0;
options.DiffMinChange = 1.0e-5;
tic
[X, FVAL] = fmincon(fit_fun2, X0,[],[],[],[],LB,UB,[],options);
toc
fileID = fopen(fullfile('params.txt'),'w');
fprintf(fileID,'mod = %0.5g\n',E);
fprintf(fileID,'len1 = %0.5g\n',X(1));
fprintf(fileID,'len2 = %0.5g\n',X(2));
fprintf(fileID,'b1 = %0.5g\n',X(3));
fprintf(fileID,'b2 = %0.5g\n',X(4));
fprintf(fileID,'h1 = %0.5g\n',X(5));
fprintf(fileID,'h2 = %0.5g\n',X(6));
fprintf(fileID,'theta_02 = %0.5g\n',gett2(X(7),[X(1), X(2)]));
fprintf(fileID,'theta_03 = %0.5g\n',X(7));
fprintf(fileID,'P_targ = %0.5g\n',P_targ);
fprintf(fileID,'k_l_targ = %0.5g\n',k_l_targ);
fclose(fileID);
```

```
sys();
data = import_data();
plot(data(:,1),data(:,2),'r-.')
legend('PRBM','Design Point','FEA Initial','FEA Final','FontSize',...
    fontsize,'FontName','Times New Roman','Location', 'Best');
filename = fullfile('J:','SB_lattice_paper','printed_bal_opt');
set(gcf, 'PaperPositionMode', 'auto')
saveas(gcf,filename,'eps2c')
```

This is the file nonlcon.m, which is used by the optimization routine as a constraint function.

```
function [C, Ceq] = nonlcon(X,E,gamma,K_theta,smax)
stiff = @(b,h,l)2*gamma*K_theta*E*b*h^3/12/l;
C = (X(1)-X(8))*stiff(X(3), X(5), X(7))*...
    (X(5)/2)/(X(3)*X(5)^3/12)/smax-1;
Ceq = 0;
end
```

This is the file fit_fun.m, which is used by the optimization routine to call an ANSYS script and return a fitness value for a given design of the balancer.

```
% This function calls ANSYS to evaluate the fitness of a potential design
% for a balancer spring
function [fitness] = fit_fun(X, P_targ, k_l_targ, E, smax)
gamma = 0.85;
getr2 = @(ls)gamma.*ls(1);
getr3 = @(ls)gamma.*ls(2);
gett2 = @(theta, ls) asin(-getr3(ls)*sin(theta)/getr2(ls));
if exist(fullfile('C:','ANSYS','print_bal.lock'), 'file')
    delete(fullfile('C:','ANSYS','print_bal.lock'))
end
```

```
com = @(X) ['"C:\Program Files\ANSYS Inc\v150\ANSYS\bin\winx64\'...
    'ansys150.exe" -p aa_r -np 2 -dir "C:\ANSYS" -j "print_bal" -s '...
    'read -l en-us -b -mod ' num2str(E,'%2.10g') ' -len1 ' num2str(X(1),...
    '%2.10g') ' -len2 ' num2str(X(2),'%2.10g') ' -b1 ' num2str(X(3),...
    '%2.10g') ' -b2 ' num2str(X(4),'%2.10g') ' -h1 ' num2str(X(5),...
    '%2.10g') ' -h2 ' num2str(X(6),'%2.10g') ' -theta_02 ' ...
    num2str(gett2(X(7), [X(1),X(2)]),'%2.10g') ' -theta_03 ' ...
    num2str(X(7),'%2.10g')  ' -i "J:\SB_Lattice\Printable_Balancer\'...
    'print_bal.txt" -o "C:\ANSYS\file_opt.out"'];
command = com(X);
flag = system(['SET KMP_STACKSIZE=2048k & ' command]);
data = load(fullfile('J:','SB_Lattice','Printable_Balancer',...
    'ANS_out.txt'));
[steps, ~] = size(data);
fit = zeros(steps,1);
max_s = 0;
if flag == 0 || flag == 8
    for R = 1:steps
        if data(R,5) > max_s
            max_s = data(R,5);
        end
        if max_s < smax
            fit(R) = ((data(R,2)/P_targ-1)^2+(data(R,3)/k_l_targ-1)^2+...
                data(R,4)^2);
        else
            fit(R) = ((data(R,2)/P_targ-1)^2+(data(R,3)/k_l_targ-1)^2+...
                data(R,4)^2)+max_s/smax;
        end
    end
    fitness = min(fit)
```

```
else
    fitness = 100000
end
```

The fitness function above invokes an ANSYS script to analyze the design. This script is included below.

```
/cwd, 'C:\ANSYS'
!units in meters, Newtons
finish
!/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
/PNUM,KP,1
/PNUM,LINE,1
step = 100 !number of load steps in y
delx = 0.8
!/INPUT,'params','txt','J:\SB_Lattice\Printable_Balancer\',, 0
b2 = 2*b2
nu = .34
ndiv = 20
/prep7
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
et,1,beam188
!use section 1 first beam and section 2 for second beam
sectype,1,beam,rect
secdata,h1,b1,
```

```
sectype,2,beam,rect
secdata,h2,b2,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,0
!build geometry
k,1,0,0,0
k,2,len1*cos(theta_02),len1*sin(theta_02)
k,3,len1*cos(theta_02)+len2*cos(theta_03),len1*sin(theta_02)+...
len2*sin(theta_03)
!create compliant segments
lstr,1,2
lstr,2,3
esize,,ndiv
secnum,1
type,1
lsel,s,line,,1,1,
lmesh,all
lsel,s,line,,2,2
secnum,2
lmesh,all
allsel,all
!select a node where we can apply displacements
ksel,s,kp,,1
nslk,s
*get,grnd,node,0,num,max
allsel,all
ksel,s,kp,,2
nslk,s
*get,cntr,node,,num,max
```

```
allsel,all
ksel,s,kp,,3
nslk,s
*get,output,node,,num,max
allsel,all
!fix the ground node
d,grnd,all,0
d,cntr,rotz,0
d,output,rotz,0,,,uz,ux
d,output,uy,0
*do,i,1,step,1
d,output,ux,delx*i/step
nsubst,4,10,2,
lswrite,i
*enddo
finish
/SOL
allsel,all
LSSOLVE,1,step
FINISH
! Define variables
/post26
numvar,200
NSOL,2,output,u,x,dx
rforce,3,output,f,x,fx
prod,4,3,,,for,,,2.0 !double force to account for symmetry
prvar,dx,fx
deriv,5,4,2,,dfdx
deriv,6,5,2,,d2fdx2
prvar,dx,for,dfdx,d2fdx2
```

```
*dim,disp,array,step,1,1
*dim,force,array,step,1,1
*dim,dforce,array,step,1,1
*dim,d2force,array,step,1,1
*do,i,1,step,1
*get,dis,vari,2,rtime,i !get force at step i
*get,for,vari,4,rtime,i !get force at step i
*get,dfor,vari,5,rtime,i !get force at step i
*get,d2for,vari,6,rtime,i !get force at step i
disp(i) = dis
force(i) = for
dforce(i) = dfor
d2force(i) = d2for
*enddo
/post1
*dim,stress,array,step,1,1
/REPLOT
SET,FIRST
/PLOPTS,INFO,3
/CONTOUR,ALL,18
/PNUM,MAT,1
/NUMBER,1
*do,i,1,step,1
SET,,,,,,,i !select set number to view
etable,ben_str,smisc,32,37
esort,etab,ben_str,0,0
*GET,max_str,sort,,max
stress(i) = max_str
*enddo
*CFOPEN,J:\SB_Lattice\Printable_Balancer\ANS_out,txt,,
```

```
*VWRITE,disp(1,1),force(1,1),dforce(1,1),d2force(1,1),stress(1,1)
%18.6G, %18.6G, %18.6G, %18.6G, %18.6G
*CFCLOS
```

The optimization routine also invokes an ANSYS script called print_bal_graph.txt. This script is included below. It is nearly identical to the above script.

```
/cwd, 'C:\ANSYS'
!units in meters, Newtons
finish
/clear
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
/PNUM,KP,1
/PNUM,LINE,1
step = 100 !number of load steps in y
delx = 0.8
/INPUT,'params','txt','J:\SB_Lattice\Printable_Balancer\',, 0
b2 = 2*b2
nu = .34
ndiv = 20
/prep7
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
et,1,beam188
!use section 1 first beam and section 2 for second beam
sectype,1,beam,rect
secdata,h1,b1,
```

```
sectype,2,beam,rect
secdata,h2,b2,
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,0
!build geometry
k,1,0,0,0
k,2,len1*cos(theta_02),len1*sin(theta_02)
k,3,len1*cos(theta_02)+len2*cos(theta_03),len1*sin(theta_02)+...
len2*sin(theta_03)
!create compliant segments
lstr,1,2
lstr,2,3
esize,,ndiv
secnum,1
type,1
lsel,s,line,,1,1,
lmesh,all
lsel,s,line,,2,2
secnum,2
lmesh,all
allsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
!select a node where we can apply displacements
ksel,s,kp,,1
```

```
nslk,s
*get,grnd,node,0,num,max
allsel,all
ksel,s,kp,,2
nslk,s
*get,cntr,node,,num,max
allsel,all
ksel,s,kp,,3
nslk,s
*get,output,node,,num,max
allsel,all
!fix the ground node
d,grnd,all,0
d,cntr,rotz,0
d,output,rotz,0,,,uz,ux
d,output,uy,0

*do,i,1,step,1
d,output,ux,delx*i/step
nsubst,4,10,2,
lswrite,i
*enddo
finish
/SOL
allsel,all
LSSOLVE,1,step
FINISH
/post26
! Define variables
numvar,200
```

```
NSOL,2,output,u,x,dx
rforce,3,output,f,x,fx
prod,4,3,,,for,,,2.0 !double force to account for symmetry
prvar,dx,fx
deriv,5,4,2,,dfdx
deriv,6,5,2,,d2fdx2
prvar,dx,for,dfdx,d2fdx2
!plot force-displacement curve
*dim,disp,array,step,1,1
*dim,force,array,step,1,1
*dim,dforce,array,step,1,1
*dim,d2force,array,step,1,1
*do,i,1,step,1
*get,dis,vari,2,rtime,i !get force at step i
*get,for,vari,4,rtime,i !get force at step i
*get,dfor,vari,5,rtime,i !get force at step i
*get,d2for,vari,6,rtime,i !get force at step i
disp(i) = dis
force(i) = for
dforce(i) = dfor
d2force(i) = d2for
*enddo
/post1
*dim,stress,array,step,1,1
/REPLOT
SET,FIRST
/PLOPTS,INFO,3
/CONTOUR,ALL,18
/PNUM,MAT,1
/NUMBER,1
```

```
!plot all stress states
*do,i,1,step,1
SET,,,,,,,i !select set number to view
etable,ben_str,smisc,32,37
esort,etab,ben_str,0,0
*GET,max_str,sort,,max
stress(i) = max_str
*enddo
/post26
!XVAR,2
!PLVAR,4,
*CFOPEN,J:\SB_Lattice\Printable_Balancer\ANS_out,txt,,
*VWRITE,disp(1,1),force(1,1),dforce(1,1),d2force(1,1),stress(1,1)
%18.6G, %18.6G, %18.6G, %18.6G, %18.6G
!(F16.8, F16.8, F16.8, F16.8, F16.8)
*CFCLOS
```

D.4 Confirmation of Final Design

This script uses the balancer design developed so far to statically balance a compound lattice-flexured CAFP. The referenced files x_lat.mac and v_lat.mac are included above.

```
!This script buils a CAFP with lattice flexures. It requires
!the files x_lat.mac and v_lat.mac in the same directory
!also, change the /cwd command to the directory where this file,
! x_lat.mac, and v_lat.mac are stored.
/cwd, 'C:\ANSYS'
finish
/clear
/PNUM,KP,1
/INPUT,'params_opt','txt','J:\SB_Lattice\Printable_Balancer\',, 0
```

```
!Units in, lbf, psi
*SET,pi,acos(-1) !pi
len = 3. !length of flexible segments
cor = len*sin(pi/4) !x and y coordinates
!mod = 320000. !modulus of ABS-M30
!mod = 1.61e7  !modulus of titanium
nu = .34
x_PrL = 0.176
pr_step = 5
step = 10
ndiv = 20
gap_wid = 1.0
offset = 0.5
thk = .04 !*.83 !thickness of compliant members
n = 5 !number of lattice cells along length use 6 for b and 8 for c
wid = 1.0   !total width of flexure
b = wid-thk !center-to-center distance
I_l = thk**4/12
I_r = I_l
L1b = (len/(2*n))/b
bigK = 2.25*thk**4/16
EI_eff = mod*(2*I_r+(I_l*L1b*(L1b**2+1.0)**.5)/(2*(1+nu)*I_l/bigK+L1b**2))
P = 8.6071 !8
d = 3.9846/0.85
k_l = 1.7763 !1.7228
type = 'v'!may be type x or v
ang = 45 !20 degrees of rotation
ang = ang*pi/180
/prep7
et,1,beam188
```

```
!section 1 is for the lattice elements
sectype,1,beam,rect
secdata,thk,thk
sectype,2,beam,rect
secdata,thk*8,thk*8
et,2,mpc184
keyopt,2,1,1
keyopt,2,2,1
!element type for balancing spring
ET,3,COMBIN14
KEYOPT,3,1,0
KEYOPT,3,2,0
KEYOPT,3,3,0
!enter real constants for balancing spring
R,3,k_l,0,0, , ,,
rmore,P,
mptemp
mptemp,1,0
mpdata,ex,1,,mod
mpdata,prxy,1,,nu
nlgeom,1
!build the lattice geometry
!define keypoints
k,1,0,0,b+gap_wid/2
k,2,0,0,gap_wid/2
!build a lattice flexure anchored to KPs 1 and 2 with n divisions and
! alpha angle
%type%_lat,1,2,n,len,45
*get,top1,kp,,num,maxd
k,,0,0,-gap_wid/2
```

```
k,,0,0,-gap_wid/2-b
*get,maxkp2,kp,,num,maxd

%type%_lat,maxkp2-1,maxkp2,n,len,45
*get,top2,kp,,num,maxd

k,,cor,,0.1+gap_wid/2+2*b
k,,cor,0,0.1+gap_wid/2+b
*get,maxkp3,kp,,num,maxd
ksel,s,kp,,1,2
ksel,a,kp,,maxkp2-1,maxkp2,1
ksel,a,kp,,maxkp3-1,maxkp3
cm,anc_kps,kp
allsel,all
!build a lattice flexure anchored to KPs 1 and 2 with n divisions and
! alpha angle
%type%_lat,maxkp3-1,maxkp3,n,len,135
*get,top3,kp,,num,maxd
k,,cor,0,-0.1-gap_wid/2-b
k,,cor,0,-0.1-gap_wid/2-2*b
*get,kpmax,kp,,num,maxd
%type%_lat,kpmax-1,kpmax,n,len,135
*get,top4,kp,,num,maxd
cmsel,s,anc_kps
ksel,a,kp,,kpmax-1,kpmax,1
cm,anc_kps,kp
allsel,all
ksel,s,kp,,top1-1,top1
*do,i,2,4,1
ksel,a,kp,,top%i%-1,top%i%
```

```
*enddo
cm,top_kps,kp
allsel,all
esize,,10
!mesh flexible members
latt,1,,1,,,,1  !MAT,REAL,TYPE,ESYS,KB,KE,SECNUM
lmesh,all
!*************************************************************
!put a new origin in the model where we want the balancer
clocal,11,CART,cor/2,cor/2-(len1*cos(theta_02)+len2*...
cos(theta_03)+offset)/2+x_PrL,0,90
!build the spring geometry
!use section 11 first beam and
sectype,11,beam,rect
secdata,h1,b1,
!section 12 for second beam
sectype,12,beam,rect
secdata,h2,b2,
!section 13 for rigid sections
sectype,13,beam,rect
secdata,2*h1,b2,
!sectype 14 for connecting wires
sectype,14,beam,csolid
secdata,0.02
!build geometry
k,,0,0,0
*get,bal_kp,kp,,num,maxd
k,,len1*cos(theta_02),len1*sin(theta_02) !+1
k,,len1*cos(theta_02),-len1*sin(theta_02) !+2
k,,len1*cos(theta_02)+len2*cos(theta_03),0 !+3
```

```
k,,len1*cos(theta_02)+offset,len1*sin(theta_02) !+4
k,,len1*cos(theta_02)+offset,-len1*sin(theta_02) !+5
k,,len1*cos(theta_02)+len2*cos(theta_03)+offset,0 !+6
!create compliant segments
lstr,bal_kp,bal_kp+1 !line 1
*get,bal_line,line,,num,maxd
lstr,bal_kp,bal_kp+2 !line 2
lstr,bal_kp+3,bal_kp+1 !line 3
lstr,bal_kp+3,bal_kp+2 !line 4
lstr,bal_kp+6,bal_kp+4 !line 5
lstr,bal_kp+6,bal_kp+5 !line 6
lstr,bal_kp+1,bal_kp+4 !line 7
lstr,bal_kp+3,bal_kp+6 !line 8
lstr,bal_kp+2,bal_kp+5 !line 9
esize,,ndiv
type,1
lsel,s,line,,bal_line,bal_line+1,
secnum,11
lmesh,all
lsel,s,line,,bal_line+2,bal_line+5
secnum,12
lmesh,all
lsel,s,line,,bal_line+6,bal_line+8
secnum,13
lmesh,all
allsel,all
csys,0
!*************************************************************
k,,cor/2,cor/2+d,0
*get,top_kp,kp,,num,maxd
```

```
k,,cor/2,cor/2-d+x_PrL,0
*get,bot_kp,kp,,num,maxd
lstr,top_kp,bal_kp+6
*get,wire1,line,,num,maxd
lstr,bot_kp,bal_kp
lsel,s,line,,wire1,wire1+1
secnum,14
lmesh,all
allsel,all
!displays mechanism and depicts relative size and shape of beam elements
/ESHAPE,1
/EFACET,1
/RATIO,1,1,1
/CFORMAT,32,0
/REPLOT
ksel,s,kp,,top_kp
nslk,s
*get,topn1,node,0,num,max
allsel,all
ksel,s,kp,,bot_kp
nslk,s
*get,topn2,node,0,num,max
allsel,all
ksel,s,kp,,bal_kp
nslk,s
*get,bal_low,node,0,num,max
allsel,all
ksel,s,kp,,bal_kp+6
nslk,s
*get,bal_hi,node,0,num,max
```

```
allsel,all
!create constraints
type,2
cmsel,s,top_kps
nslk,s
*get,num_top,node,0,count
*dim,top_ns,array,num_top
*do,i,1,num_top,1
*get,nod,node,,num,min
top_ns(i)=nod
nsel,u,node,,nod
*enddo
allsel,all
*do,i,1,num_top,1
e,topn1,top_ns(i)
*enddo
d,topn2,ux,0,,,,uz,rotx,roty,rotz
d,topn1,roty,0,,,,
cmsel,s,anc_kps
nslk,s
d,all,all,0
allsel,all
*do,i,1,pr_step,1
d,topn2,uy,-x_PrL*i/pr_step
lswrite,i
*enddo
*do,i,1,step,1
d,topn1,rotz,15*pi/180*i/step
lswrite,i+pr_step
*enddo
```

```
finish
/SOL
LSSOLVE,1,pr_step+step,1
finish
finish
/post26
! Define variables
NSOL,2,topn1,rot,z,rot
rforce,3,topn1,M,z,torque
XVAR,2
PLVAR,3
!divide torque by rotation and call resulting variable stiffness
quot,4,3,2,,stiff
PRVAR,rot,torque,stiff
```

APPENDIX E. SETUP AND USE OF THE TORQUE-ROTATION MEASURING APPARATUS

During much of the research presented herein, it was necessary to measure torques and rotations simultaneously. A test setup was developed to do this, which has been shown in several chapters throughout this work. This appendix details the hardware, software, and procedures that make this test setup run.

E.1 Overview

In my compliant mechanisms work I have often found it necessary to measure the torque-deflection behavior of various mechanisms. This is useful to validate numeric or analytic models, especially if you are designing a new kind of joint or flexure. Once you have torque and deflection data, you can calculate stiffness or look for special behavior such as multi-stability. To fill this need I built the torque-rotation setup, shown with a variety of mechanisms in Figure E.1.

The torque-rotation test setup consists of hardware and software. The hardware can be divided into sensors, DAQ equipment, and the mechanisms. These will be considered in turn. The software consists of the Labview DAQ and an Excel spreadsheet designed to aid in calibration of the device. These systems function together to allow the collection of data.

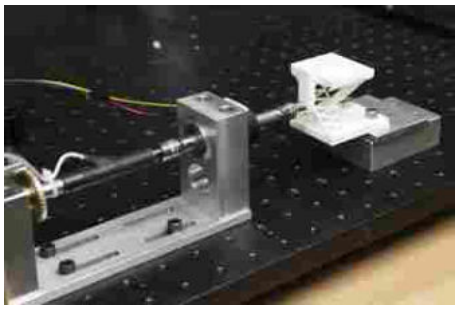
Finally, the calibration and data gathering procedures will be discussed.

E.2 Hardware

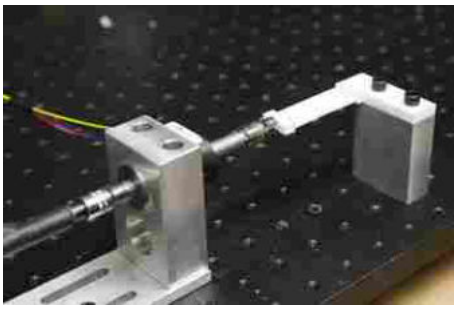
As mentioned, the hardware consists of the DAQ, the sensors, and the mechanisms. All part numbers and vendors are listed in Table E.1.

E.2.1 DAQ

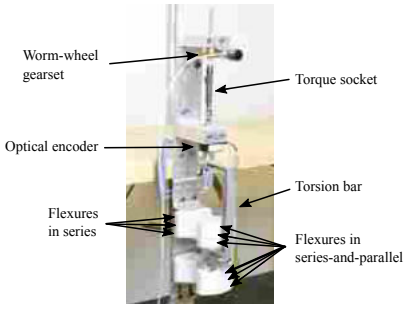
The DAQ itself is a product of National Instruments. Its elements are shown in Figure E.2.



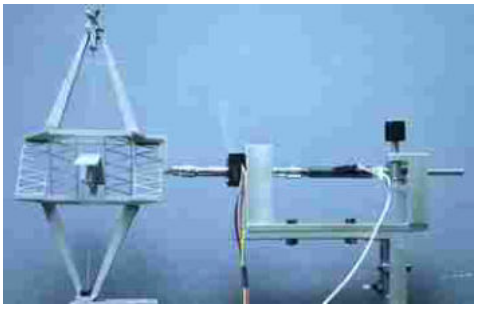
(a) Measuring the bending stiffness of a plastic lattice-flexured cross-axis flexural pivot.



(b) Measuring the torsinal stiffness of a blade flexure.



(c) Measuring the bending stiffness of a compound joint. The setup is mounted vertically so the weight of the apparatus does not affect the measured torque.



(d) Measuring the stiffness of a statically balanced lattice-flexured cross-axis flexural pivot.

Figure E.1: A sampling of measurement setups.

E.2.2 Sensors

The two sensors used are an optical encoder and an Omega torque transducer. The sensors are shown in Figure E.3.

Table E.1: Part numbers of purchased hardware.

Equipment	Part Number	Supplier	Purpose
DAQ Chassis	cDAQ 9174	National Instruments	Provides an interface between the DAQ modules and the computer.
DAQ Module	NI 9411	National Instruments	Read digital input
DAQ Module	NI 9219	National Instruments	read analog input
Torque transducer	TQ103-50	Omega	Torque measurement
Optical encoder	E2-500-375-IE-H-D-B	US Digital	Rotation measurement
Steel worm gear	A 1C 5-N32	sdp-si.com	Torque input
Brass worm wheel	A 1B 6-N32040	sdp-si.com	Torque input

(a) DAQ chassis. This is what the modules plug into so the computer can read your data.

(b) NI 9411 module. This is used for digital input. In this case we use it to read the optical encoder.

(c) NI 9219 module. This is a very versatile 4 channel analog input. We use it here for reading torque transducers, but it can read any sensor with an analogue voltage output up to 5 volts.

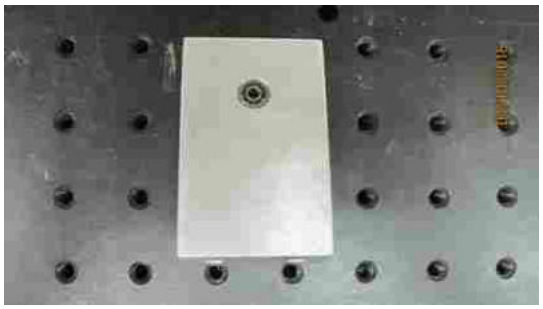
Figure E.2: Elements of the DAQ hardware.

(a) Digital encoder from US Digital. Note that the one we have has a hole in the cover to allow it to mount to a through-shaft. Measures rotation at resolutions up to 0.018°.

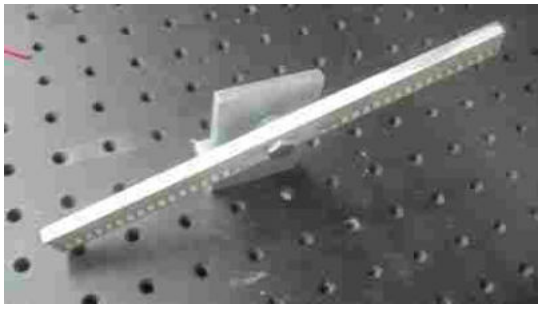
(b) Omega TQ103-50 torque transducer. This measures torque, up to ±50 in-lbs.

Figure E.3: Sensors used to collect torque and displacement data.

(a) Balance bar with holes spaced 0.5 cm apart.



(b) Balance pivot - essentially this is a piece of angled aluminum inset with a small bearing.



(c) The balance bar and pivot assembled.

Figure E.4: Mechanism used during calibration of the torque transducer.

E.2.3 Mechanisms

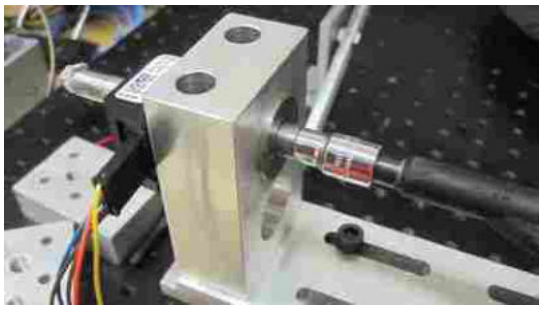
Several mechanisms were built to aid in calibration and measurement. The balance bar and balance pivot are used during calibration to apply a known torque. The holes are spaced 0.5 cm apart, and by hanging weights from either side of the balance bar at different distances, very small torques can be applied to the torque transducer. More information on this procedure will be included below. These elements are shown in Figure E.4. CAD files for the hardware described can be found in `\cmrvault\Measurements\Torque_and_Position\CAD files`.

To apply and hold a torque, a worm-wheel gearset is used. This applies a rotation to a shaft, which in turn rotates the torque transducer. The resulting rotation is passed to another shaft to whatever mechanism is being actuated. Along the way the output shaft's rotation is measured by the optical encoder, mounted to a pillow block. These elements are shown in Figure E.5.

E.3 Software

Two main pieces of software are used: an Excel spreadsheet and a Labview DAQ.

(a) Worm-wheel assembly. Turning the knob at the top rotates the shaft.



(b) This pillow block is essentially a piece of aluminum with a bearing pressed in to support the output shaft and mount the optical encoder.

(c) The worm-wheel and pillow block are mounted to a slotted plate so the whole assembly can slide forward or back.

Figure E.5: Elements of the torque application mechanism.

E.3.1 Spreadsheet

The Excel spreadsheet (located in the CMR Vault under `\cmrvault\Measurements\Torque_and_Position\torque_calibration.xlsx`) evaluates the equation $T = g(m_1 l_1 - m_2 l_2)$ to calculate the torque applied by weights hanging from the balance bar. By relating the output from the torque transducer to the known applied load, a calibration can be achieved.

E.3.2 Labview DAQ

A copy of the Labview DAQ is located in the same location as the Excel spreadsheet (`\cmrvault\Measurements\Torque_and_Position\torque_and_position.vi`). There are several nuances to using this software that will be explained in Section E.4.1.

The DAQ software reads the output from the torque sensor and optical encoder and allows you to save the data to a text or Excel file.

E.4 Procedures

This section lists the procedures of setting up and using the data collection system. I tried to make this as complete as possible by tearing down the system and rebuilding it to capture each step.

E.4.1 Setup

Before the system may be used it must be set up. First the hardware is assembled and then the software is configured.

Hardware Assembly

Hardware assembly is accomplished with the following steps:

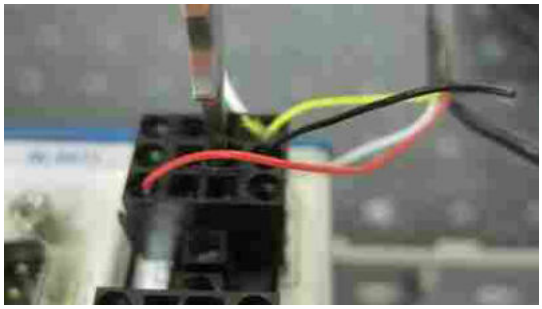
1. Connect DAQ chassis to computer and power (see Figure E.6) using the appropriate cords.

Figure E.6: Connect the chassis to the computer via USB and connect to power using the power cord.

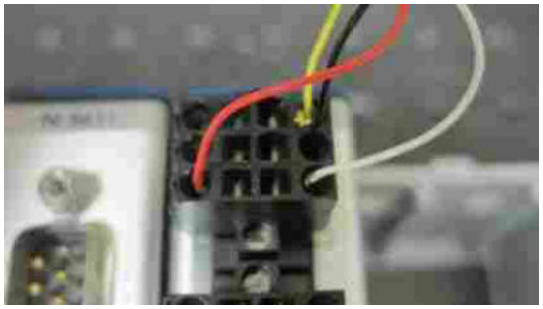
2. Plug in the DAQ modules 9411 and 9219. It doesn't matter which slots you use except that you will need to know which ones they're in. See Figure E.7.

3. Connect the torque transducer wires to a channel on the 9219 module. Refer to Table E.2 for which wires go to which terminal. See Figures E.8a and E.8b.

Figure E.7: Plug the modules into the DAQ chassis. Where they go doesn't matter, but keep track of which is where.



(a) Attaching wires to the DAQ module is done by inserting a small flathead screwdriver as shown (into the square hole) and then inserting the wire into the round hole. Don't force the screwdriver; wiggle it gently back and forth and you'll feel it slide down. Removing the screwdriver secures the wire (tug a little to be sure its connected).



(b) All transducer wires assembled to the module. The terminal number is found written on the module itself. Double check these connections.

Figure E.8: Assembly of the hardware - connection of the transducer.

4. Connect the optical encoder to the 9411 module, and supply 10 volts DC power to the module using the wiring terminals marked "VSUP" and "COM." VSUP goes to the red terminal on the power supply, COM to the black terminal. Refer to Figures E.9a and E.9b.

Your hardware is now connected, but the software must be configured to whatever ports you plugged things into.

Software Setup

Setup of the software is accomplished in the following steps:

(a) Plug the five-pin connector into the encoder. Be careful to not damage the encoder pins. Ensure the brown wire is connected to the pin marked "GND". The pins are labeled on the side of the encoder, but you have to look closely to see them.

(b) Plug the serial port into the 9411 module. You will also need to supply the module with approximately ten volts (DC). This is most easily done from a bench power supply.

Figure E.9: Assembly of the hardware - wiring the DAQ modules.

1. Open the DAQ VI file (torque_and_position.vi). VI is the file type. It is located either on the desktop of the computer in the back room or on the vault at `\cmrvault\Measurements\(torque_and_position.vi`. The front panel should open and look something like what is shown in Figure E.10

2. Open the block diagram by pressing Ctrl+E. A screen that looks something like Figure E.11 should appear.

3. Find the DAQ assistant for the torque transducer input (highlighted in blue in Figure E.11 - you may have to scroll around) and double click on it to bring up the window shown in Figure E.12.

4. From this window you can add or modify channels. For example, you could add another transducer to take multiple measurements. We need to simply tell Labview which transducer

Table E.2: Wiring connection of the torque transducer. The black wire indicated is the small black wire, not the large shield wire that surrounds the others as they are gathered into their sheath.

Wire color	Terminal number
Red	3
Black	5
Yellow	4
White	6

Figure E.10: The main interface of the DAQ software.

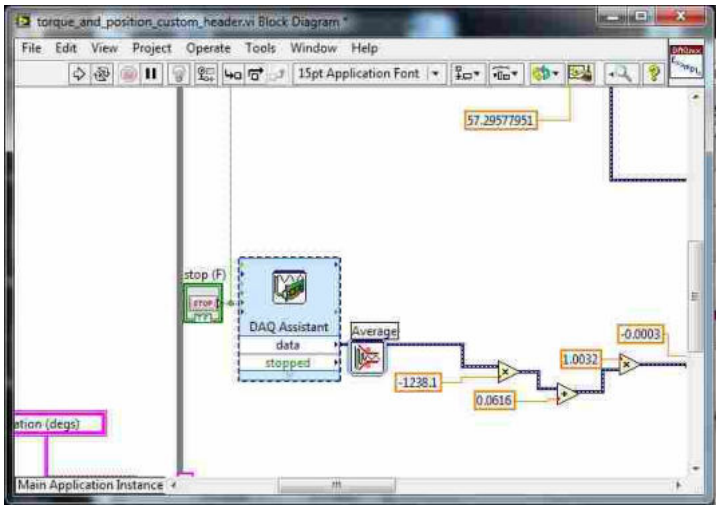


Figure E.11: The block diagram that determines how Labview collects your data and what happens to it. This is a fairly large thing, so there is more to it; this is only a portion.

channel we wish to read. Right-click on Torque1 and select "Change Physical Channel." See Figure E.13.

5. Select the channel to which you wired your torque transducer. For example, in Figure E.14 we select cDAQ4Mod2 ai0.

6. Click "okay" and "okay" to return to the block diagram. A window should briefly appear saying that "Building VI."

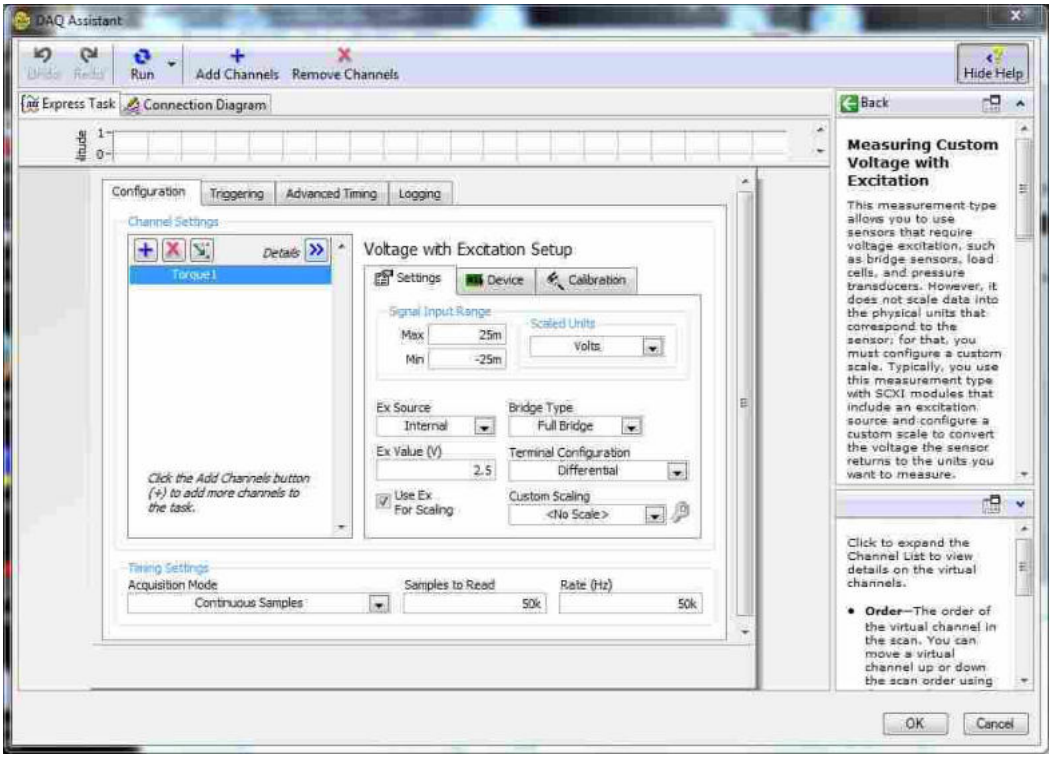


Figure E.12: The DAQ assistant helps reconfigure your measurement channels.

7. To set which channel the encoder is connected to, minimize the block diagram and return to the front panel window (from Figure E.10).

8. Use the dropdown menu on the front panel to select which channel reads the optical encoder (see Figure E.15).

   Your system is now ready to calibrate.

E.4.2 Calibration

After setup, the system must be calibrated. This is accomplished with the following steps:

1. Reset the calibration constants in the block diagram. See Figure E.16.

   (a) Press Ctrl+E to get to the block diagram from the front panel.

   (b) From the DAQ assistant, trace the orange wires to a set of constants. These constants multiply or offset the torque output.

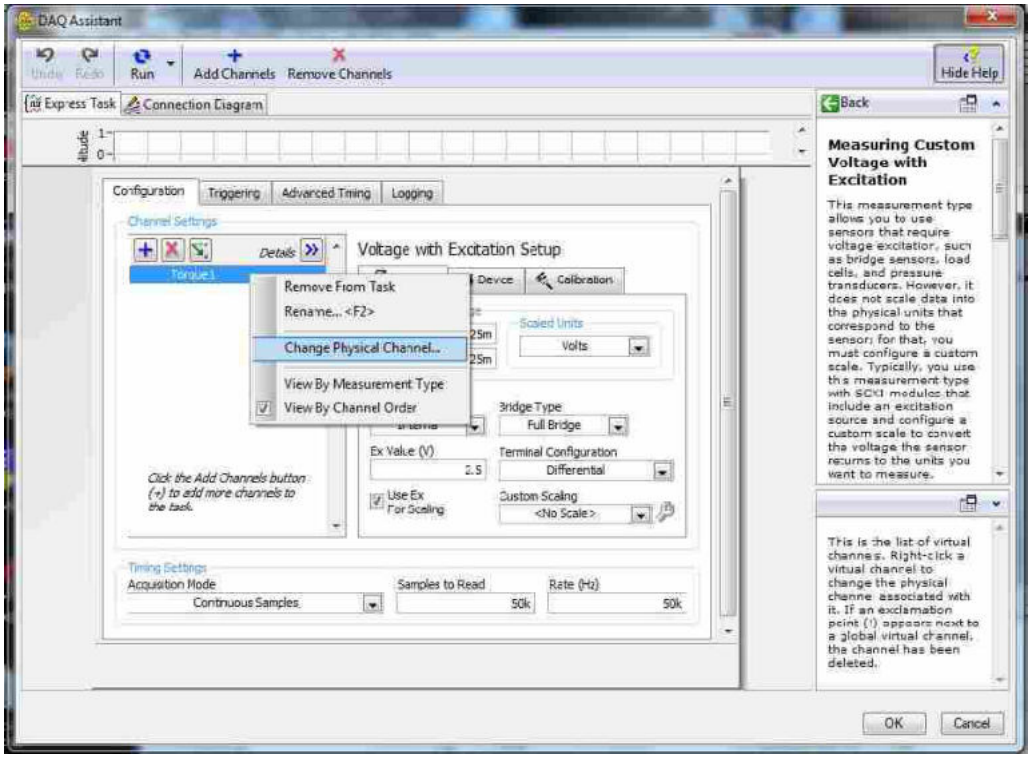


Figure E.13: Right-click on the channel you wish to use and select "Change Physical Channel."

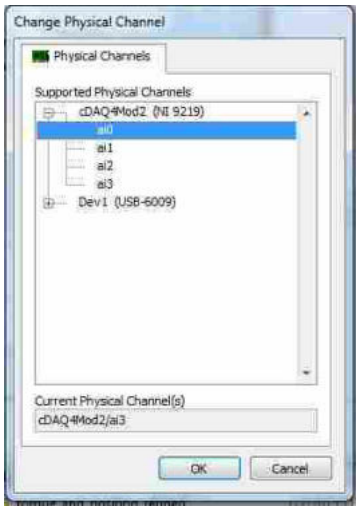


Figure E.14: Select the channel to which you wired the torque transducer. For example, cDAQ4Mod2, ai0 means that we are selecting the DAQ chassis plugged into the 4th USB port (DAQ4), using the 2nd module in that chassis (Mod2), and we want the output from the 0th channel (ai0) in that module (the channel number is printed on the side of the module as CH0, CH1, etc.).

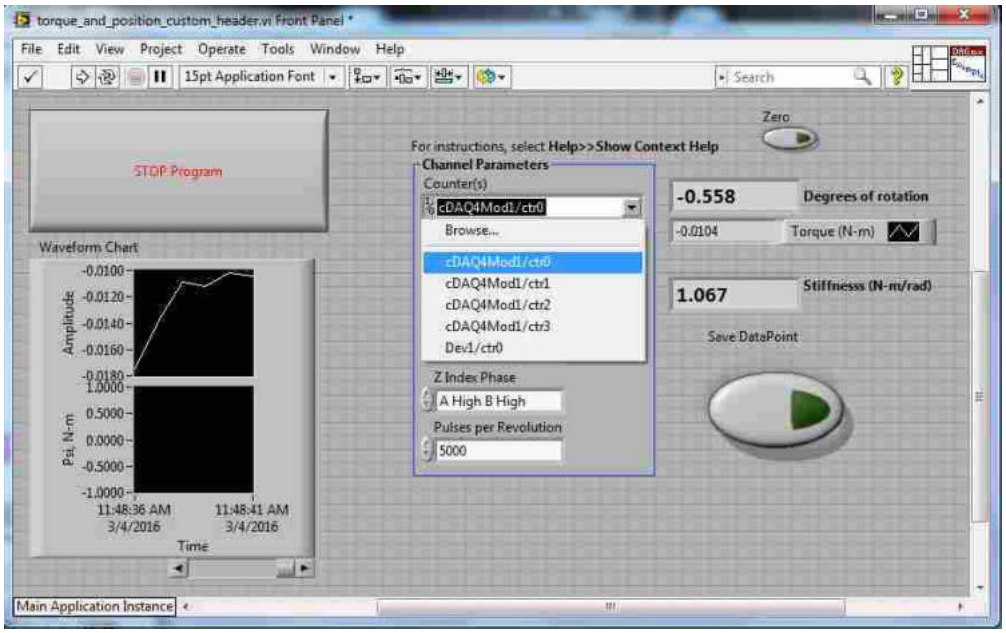


Figure E.15: Select into which channel the encoder and module are connected.

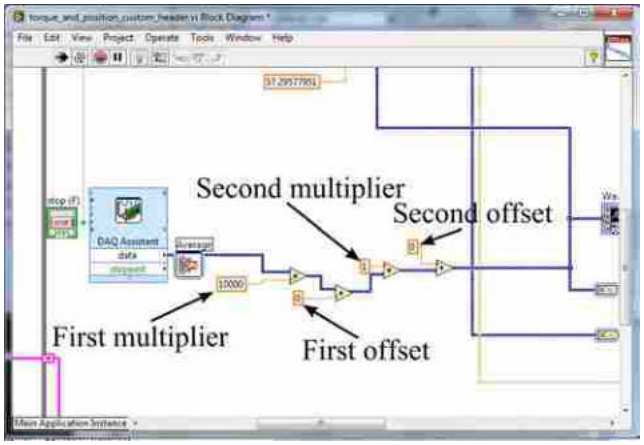


Figure E.16: Reset the calibration constants before performing the calibration procedure. The first multiplier should be 10,000, the first and second offsets should be 0, and the second multiplier should be 1.

(c) Double click to edit the numbers

(d) Set the first multiplier to 10,000, the second to 1, and both offsets to zero.

2. Set up the balance bar and measurement assembly. The shaft of the torque-displacement mechanism should be co-axial with the rotation of the balance bar. You may need to place spacers under the balance pivot. Then attach a 7/16” socket to the output shaft to interface

with the bolt head on the balance bar. If there is excessive backlash in the connection, wrap a narrow strip of paper around the bolt head, or use a small dab of hot glue to connect everything temporarily. See Figure E.17.

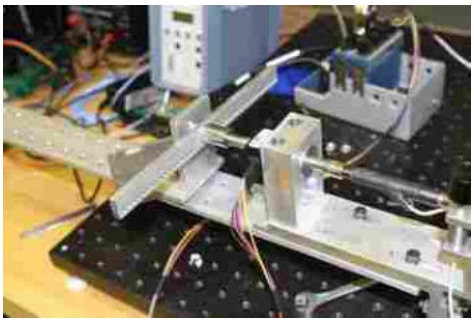


Figure E.17: Raising the balance bar and torque assembly above the table will give you more room to hang masses.

3. Run the DAQ by pushing the "run" button on the DAQ front panel (see Figure E.10).

4. Apply torques to the balance bar and record the torque output. See Figure E.18

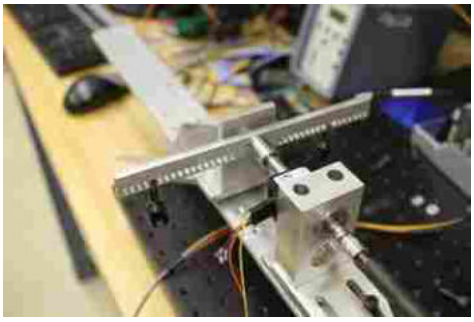
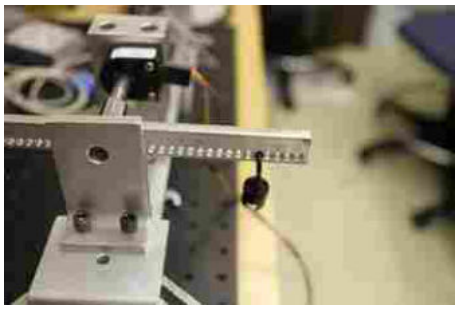


Figure E.18: Apply torque by hanging weights from the balance bar.

(a) You can get a set of masses from the ME checkout room.

(b) Hang the masses from the holes in the balance bar, ensure that the bar is level. You can hang them from one side or both, at different distances from the center or equidistant, equal masses or different. The purpose is to apply a net torque on the bar that is similar in magnitude to what you expect to measure.



(a) Distances on the balance bar are marked in centimeters, with holes spaced every 0.5 cm. Here, the mass is hanging at 8 cm.

(b) Enter the masses and their distance from the center pivot into the Excel spreadsheet. Mass on the right is entered in the far left column (labeled "mass R (g)"), its distance from center in centimeters is entered in the column labeled "LR (cm)". Mass on the left is entered in the column labeled "mass L (g)", its distance from center in centimeters is entered in the column labeled "LL (cm)." the spreadsheet will calculate the applied torque in column E (labeled "torque (N-m)"). Enter the current torque from the DAQ under "output."

Figure E.19: Calibration of the system.

(c) Record the hanging mass and distance in the Excel spreadsheet to calculate applied torque. For example, Figure E.18 shows an 10 gram mass 8 cm from center on the bar's right (left in the picture) and another 10 gram mass as 6.5 cm from center on the bar's left (right in the picture). See Figures E.19a and E.19b.

(d) Record the torque output in the Excel spreadsheet.

5. Repeat step 4 for a range of torques across your expected load range.

6. When you've finished, stop the DAQ by clicking the large "STOP Program" button.

7. Apply a linear curve fit to the calibration data you've collected.

   (a) Make a scatter plot of the data in Excel with torque output on the x-axis and actual torque on the y-axis

   (b) Click on the data in the plot, then right-click and select "Add Trendline" (see Figure E.20).

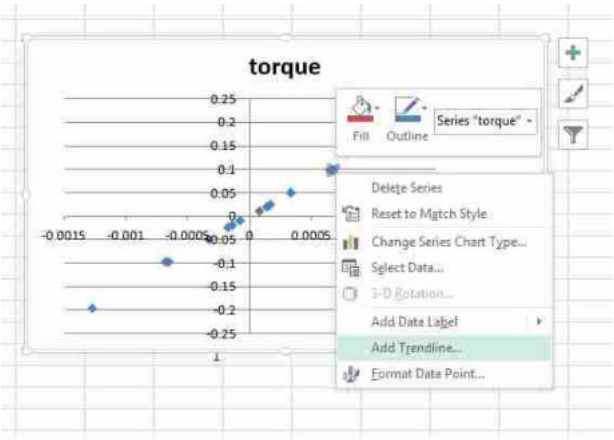


Figure E.20: Add a trendline to your plot.

(c) Under the trendline options, select "linear," and check the box for "Display Equation on chart." See Figure E.21.

Figure E.21: Select "linear," and check the box for "Display Equation on chart."

8. The coefficients from the trendline become your calibration terms. If this is your first pass at calibration, multiply the slope of the trendline by the current first multiplier (10,000) and enter that as the new first multiplier. Replace the first offset coefficient with the *b* term of the trendline. If this is your second pass at calibration, enter the slope of the trendline as the new second multiplier. Replace the second offset coefficient with the *b* term of the trendline. See Figure E.22.

Figure E.22: The slope and offset from the trendline are your calibration coefficients. For a line written as $y = mx + b$, m is multiplier, b is your offset.

9. If you desire higher measurement accuracy, repeat steps 3-8, but now putting the trendline coefficients in the block diagram as the second multiplier and offset coefficients. This may get you a slightly better calibration, but may be skipped if you don't feel like doing it.

E.4.3 Measurement

Now that the system is calibrated you can use it to measure rotation and displacement. Attach the test specimen to the torque assembly. Try to minimize backlash in the connection. Run the DAQ. Zero the displacement when you are at your desired start position. Displace the mechanism, saving data points when desired. When you finish collecting data stop the DAQ and save the data file. It saves as plain text or as .xls, with columns of torque, rotation, and stiffness. Sample measurement setups are shown in Figure E.1.