Empirical Characterization of Unconstrained Tape Spring Deployment Dynamics

THESIS

Daniel M. Williams, Captain, USAF

AFIT/GSS/ENY/12-M07

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GSS/ENY/12-M07

Empirical Characterization of Unconstrained Tape Spring Deployment Dynamics

THESIS

Presented to the Faculty

Department of Aeronautics and Astronautics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Astronautical Engineering

Daniel M. Williams, B.S. Aerospace Engineering

Captain, USAF

March 2012

AFIT/GSS/ENY/12-M07

Empirical Characterization of Unconstrained Tape Spring Deployment Dynamics

Daniel M. Williams, B.S. Aerospace Engineering
Captain, USAF

Approved:

Dr. Jonathan T. Black, (Advisor)	Date
Dr. Richard G. Cobb, (Member)	Date
Dr. Donald L. Kunz, (Member)	Date

Abstract

Satellite cost and mission capability are very sensitive to mass requirements. Mass or volume savings in one component will either lower total spacecraft mass, or provide greater margin for design error in other components. Solar arrays and antennas in particular are often driven by launch vehicle volume constraints instead of mission needs. Deployable structures provide more on-orbit area for both solar arrays and antennas, while occupying less space during launch. These structures therefore, help address these constraints and have been proven as efficient solutions for many years. Tape springs are one example in this category of structures which offer greater reliability for less mass and for a variety of configurations.

This thesis compares a large number of trials on bending tape springs to characterize the behavior with respect to the tape spring geometry (radius, subtended angle, thickness and width) and load conditions (inertia, bend angle and skew angle). These tape spring trials measure representative unconstrained deployment trajectories to provide a representative dataset for comparison against previous and future models. Error trends are identified with respect to bend angle, skew angle, tape spring width, and tape spring thickness. Then an optimized model which characterizes these trends is fitted to the experimental data. This optimized model supports the hinge dependence on bend angle and skew angle, but is inconclusive with respect to the remaining design parameters.

Acknowledgements

I would like to thank to my faculty advisor, Dr. Black, for his guidance throughout the course of this thesis. The breadth of insight and experience that he offered was invaluable. Also, thanks to Mr. Alan Jennings for his extensive expertise on the VACE lab, previous testing, and every other random subject. Sincere thanks to Mr. Chris Zickefoose for help constructing test equipment, without whose help the experiment could not have happened.

Daniel M. Williams

Table of Contents

Page

Page

List of Figures

List of Tables

List of Symbols

Empirical Characterization of Unconstrained Tape Spring Deployment Dynamics

I. Introduction

Because of limited volume and cost, many solutions have been explored for satellite structures. Deployable structures currently offer increased flexibility over static structures by enabling on-orbit structure dimensions larger than would otherwise fit into a launch vehicle. Deployable structures may be either active or passive. Active deployment requires continued input of energy typically via motors or heaters. Passive deployment merely involves the release of the structure which assumes a stable configuration after a period of time without further action, typically due to gas inflation or internal stiffness. Many passive, self deploying structures deploy from a folded (stowed) state to a deployed state entirely through the release of elastic spring strain energy inherent to the structure. Tape springs are a mature example of this category of structures. Tape springs are very long, thin strips of metal or composite material that are curved across their width. A typical tape spring will be meters long, centimeters wide, millimeters thick, and curve across 30°–360°. When stowed, tape springs may be rolled into a much smaller volume without exceeding the material's yield stress, and therefore without loss of stiffness. When properly designed, these structures will exhibit high stiffness when deployed and resist any further deformation.

Therefore tape spring structures are suitable for antennas, solar arrays and any other mission sensors that must deploy into a larger dimension on orbit than will fit into a launch vehicle. However, because of their light weight and stored strain energy, the deployment paths are not easily predicted. More general and accurate models would reduce the potential for collisions against other spacecraft components. Once designers can show that tape springs will not damage nearby components, tape

springs would allow lighter structures for a wider range of applications. This in turn would decrease cost or increase capabilities on any satellite mission that requires the use of deployable structures.

1.1 Deployable Structures

The simple metal measuring tape first appeared in 1922, following a patent by Hiram Farrand [1]. Tape springs technology was also adopted quickly in the space race. As early as 1962 an actively-deployed tape spring boom was flown. Called STEM (Storable Tubular Extendible Mast) and BI-STEM (two STEM units attached back-to-back), these components are tightly controlled rolls of tape springs deployed by a motor. The motor is attached directly to the tapespring roll and as it turns, one end of the tapespring is pushed out and deployed. This variant is very reliable and hundreds of booms have been flown with 100% mission success, according to the manufacturer. [2]

Design requirements eventually drove development of deployables for larger structures. Extendable truss booms such as Northrop Grumman's Astromast [3] and ATK's Coilable Boom Systems [4] fulfilled this role, providing $7.4x10^6\, lb - in^2$ of bending stiffness, ten times the stiffness of $222,000\, in^2 - lb$ BI-STEM booms. Furthermore the truss booms extended farther, up to 14m as flown on the MILSTAR satellites. A competing technology is telescoping booms which provide a similar stiffness and higher load limit compared to the extendable truss booms at a higher mass per unit length. STEM masts, extendable truss booms and telescoping booms only extend in one direction, however. For larger areas, different solutions have been developed.

Large deployable reflectors have been built as large as 12m wide. One such example is the AstroMesh deployable reflector, expends from a stowed diameter of 1.14m to a deployed diameter 12.25m (both at a length of 3.81m). [5] Composed of stiff composite rods and a webwork of tension cables, this reflector is an example of a deployable tensegrity structures which enable even larger applications. However, one

limitation of these structures is the requirement for active deployment. Compared to a typical launch vehicle fairing 5m in diameter, a 12m wide component would not be possible without deployable structures. Further research into deployable structures in general continues. In 1981 discussion of systematic design was published by H. W. Stoll [6], and discussion of the dynamics was investigated by Weeks [7]. Further research has also explored inflatable booms [8] and heat-enabled deployments with RIGEX [9].

1.2 Self Deployable Tape springs

Despite over ten years of flight heritage, tape springs were not studyed in depth until 1973, when E.H. Mansfield derived the full equations describing "Large-deflexion [sic] torsion and flexure of initially curved strips". Starting from general shell theory, Mansfield produced solutions for the moment-curvature relationships for lenticular (parabolic) cross sections, and more importantly, constant thickness cross-sections that describe the tape springs under consideration here [10]. This work laid the foundation for further investigations of tape spring hinges, which unfold passively due to internal strain energy. Because internally driven deployment require less parts than active deployments, tape spring hinges have the capability to be lighter and more reliable than existing tape spring solutions (such as BI-STEM).

As new spacecraft designs of all types push demand for more antenna and solar array area, new structural solutions offer solutions. Anything from small parts to entire structures have been proposed to be replaced by tape springs: a panel antenna: [11], spars for a parabolic high gain antenna [12], and even very cheap simple dipole antennas for CubeSats [13]). All of these designs are stowed compactly during launch, then deployed into larger shapes for use on orbit. Beyond these similarities, the designs may employ a variety of folding methods– most commonly accordion folds and rolling (the latter case being the most common). These designs are typically tape springs: very thin strips of material with transverse curvature, and rolled

up like a roll of tape while stowed. The benefit of tape springs are that when properly designed, they can be rolled up into a very small packed configuration without plastically deforming; when deployed, the curvature induces stiffness, allowing use as rigid structures. However, the lack of full behavior predictions has hindered these proposals.

1.3 Hinge Equivalence

The principle that most simplifies predictions of tapespring behavior is that under bending, a tape spring forms a hinge and that hinge acts independently of conditions else where on the tape spring. Seffen and Pellegrino explicitly state that "Therefore, *a symmetric fold can be modeled as a point hinge with zero shear force and constant moment.*" (Author's Emphasis). Furthermore, they modeled (both analytically and by FEA (Finite-Element Analysis) a long length of tape spring as one of these hinges in the middle attached to longer end regions which behaved linearly as simple beams. This pattern shall be revisited later.

The above result was very quickly followed up by combination with a slightly less restricted case of pure moments **and** forces [12] – in other words, the expected case for actual use on satellites. Unfortunately, for a given pair of moments and reactions, a given piece of tape spring was indeterminate under force-balancing. This was also discussed and expanded in a paper by Seffen [14] where an energy formulation is used to settle the ambiguity.

Subsequent research expanded the formulations to allow three-dimensional movement, allowing measurment of bending at an angle ('twist angle' γ), imperfect folding, and wider sheets or discs constraining the tape springs in certain ways. [15]. Attempts have been made to ascertain the kinematics of unconstrained twist-skew-bending motion under only inertial loads, however, the difficulties of mitigating aerodynamic and gravitational effects has prevented complete results.

1.3.1 Materials. Once the basic mechanics were discovered, research expanded from simple homogeneous materials into the lighter, stiffer composites such as S-glass and carbon fiber reinforced plastic (CFRP). Allowable thickness and number of layups as well orientation of these layups continue to be investigated. [16] Notably, by appropriate choice of layups, the tape springs will exhibit bistability and required no restraint while stowed during launch.

1.3.2 Structures. Current research is focusing on the application of this knowledge to larger structures. The first example is an isolated single composite hinged with a specified geometry [16], and of entire structures such as hexapod telescope mirror mounts [17], monolithic synthetic aperture radar antenna [11], simpler folded trusses [18], and square lattice supports of membranes [19].

To date, several satellites have flown with these systems, and the technology has been proven enough for simple cases for commercial off the shelf (COTS) solutions to supply them for CubeSats [13] The best known case may be MARSIS, which is a radar sounding experiment on the ESA Mars Express spacecraft. Marsis is notable for a boom deployment issue caused by thermal conditions which was subsequently solved with no visible ill effects. Full technical details available in [20].

1.4 Research Focus

The research looks at the deployment dynamics of a fully three-dimensional tape spring. Because testing tape spring structures is difficult, a variety of approaches have been attempted, to varying degrees of success. Even the MARSIS instrument was eventually launched without full testing: ". . . unless there is a zero-g environment— the tube hinges do not have enough torque to deploy and any significant friction will prevent deployment." [21] Some contemporary researchers have focused on mitigating this problem through offload mechanisms [21, 22], often in conjuction with FEA simulation. However, the difficulties encountered by these

experiments suggests that gravity offload is not an effective testing strategy. Although options exist to test in microgravity either through airplane-flown parabolic arcs or obviously orbital launch, the cost of these options is prohibitive for all but the most mature designs.

In response, this paper investigates tape springs by a much less expensive method of tracking by reflective markers. To adapt to the limitations of ground based testing, the affects of gravity and atmosphere have been mitigated in the experiment design.

Two tactics were used to control both gravity and air drag effects. The first tactic was to test the variation of parameters against the background of gravity and atmosphere– i.e. focusing on the variation with respect to the design parameters, while acknowledging that the absolute deployment times and shocks will be less accurate. The second tactic was to organize the tests such that the effects of gravity and atmosphere were both minimized and predictable. In practice this means that gravity is easily modeled, and air resistance was slow enough to be negligible.

Another difference from contemporary work is the method of investigation. Past research efforts are predominantly focused on FEA methods [14, 16, 21], plane-restricted deployment [23] [21], and restrained motion [16] [15] [22]. To date these have been overwhemingly 2-dimensional or otherwise restrained. An excellent example of this is the MARSIS payload on the ESA Mars Express mission, shown below in figure 1.1. The MARSIS payload underwent limited ground-based deployment testing, and full deployment was only simulated in FEA software.

The aim of this thesis is to improve the model of unrestricted tape spring hinge dynamics. By comparing current models' predicted path to the measured path under varying experimental parameters, the accuracy of current and new models will be shown. Success will be achieved if a model that predicts the tape spring motion to within a standard deviation of the measured data. The experiments here are designed to determine the effect of tape spring geometry on unconstrained

deployment dynamics. The parameters of interest are thickness, width, radius, and subtended angle, and applied force.

Because current models do not take into account tape spring width and are expected to have significant error.

Because this work varies design variables over a wider range than previous experiments, the work is not directly comparable. Nevertheless, agreement may be verified where certain combinations of design variables match will match those used in previous work. In these test cases, the calculated hinge reaction moments should match the published work, and the models arrived at in those works should reproduce similiar results here.

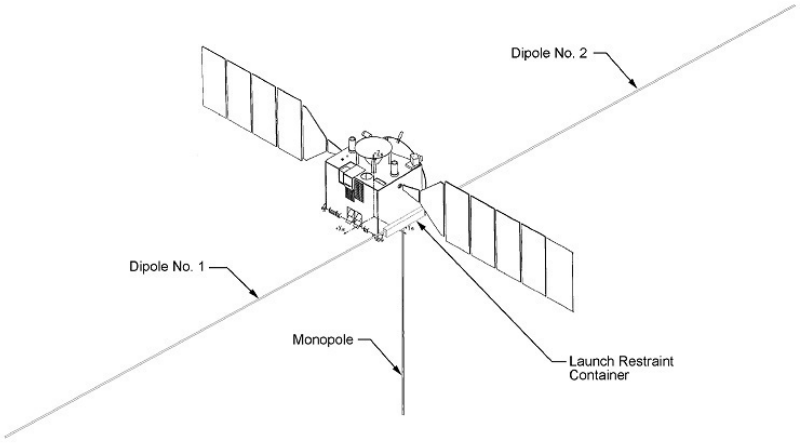

Figure 1.1: The MARSIS payload on Mars Express is a millimeter-wave sounding radar composed primarily of two tape spring dipole antennas. Credit: ESA/Marks et al. [21]. Originial caption reads: "Mars Express Spacecraft with Deployed MARSIS Experiment"

1.5 Thesis Organization

To aid navigation, this thesis is organized according to the AFIT Guidelines. Chapter I describes preceeding and current research, followed by a summary of work in this thesis. Chapter II lays out the current state of theory, as well as the organization and layout of these theories. Chapter III details the equipment and procedures for all experiments, and any other details required to produce or reproduce the results. Chapter IV contains first the processed data results (omitting the voluminous raw data), then analysis results, progressing from the simplest and most direct analysis results to the more complex and indirect results. A model is proposed to correct any defects found, then fitted to the experimental data. The implications of this model are then discussed. Chapter V concludes the paper, and then lays out several options for future work. Appendix A contains full properties and dimensions of all tape spring hinges. Appendix B contains all code used for analysis. Finally, due to its large volume, the raw data is provided in an attached CDROM.

II. Background

2.1 Bend, Twist, and Skew

To facilitate discussion of the tape springs, the notation used here follows the existing conventions started by [24]. Figure 2.1 below shows how the deflection of the tape spring is labeled, while Figure 2.2 shows the orientation of axes and curvatures on the tape spring. These transformations are obtained by a 3-2-1 rotation from undeformed (inertial) coordinates I to body coordinates B, which are each associated with each rigid body measured.

The axes are orthonormal: the x axis points down the length of the tape spring, y points across the tape spring, and z completes the right-handed set, pointing up. The bending angles around the x, y, and z axes are γ for twist angle, θ for bending angle, and μ for skew angle, respectively. Also from these three axes are defined the curvatures. Curvature along x is called alternatively 'longitudinal curvature', κ_ℓ, κ_{xx} or simply κ_x. Similarly, κ_y represents 'transverse curvature' along y, and twist curvature is κ_{xy}. In turn, the initial (undeformed) curvatures are also shown in Figure 2.2 as $\kappa_{x,0}$ and $\kappa_{y,0}$ although the initial longitudinal (x) curvature is zero for all tape springs considered.

2.2 Equal Sense and Opposite Sense

Because the behavior of the hinge depends on whether θ is positive or negative, these two cases have been termed equal and opposite sense bending in the literature. Equal sense bending corresponds to positive θ, and will generate hinge curvature of the same sign as $\kappa_{y,0}$ (initial transverse curvature). Opposite sense bending corresponds to negative θ, and will generate hinge curvature of opposite sign to the $\kappa_{y,0}$. These configurations are shown in figure 2.3.a.i and 2.3.b.i below, as defined by [24]. Their nomenclature is preserved here (except θ is of opposite sign to the figure below).

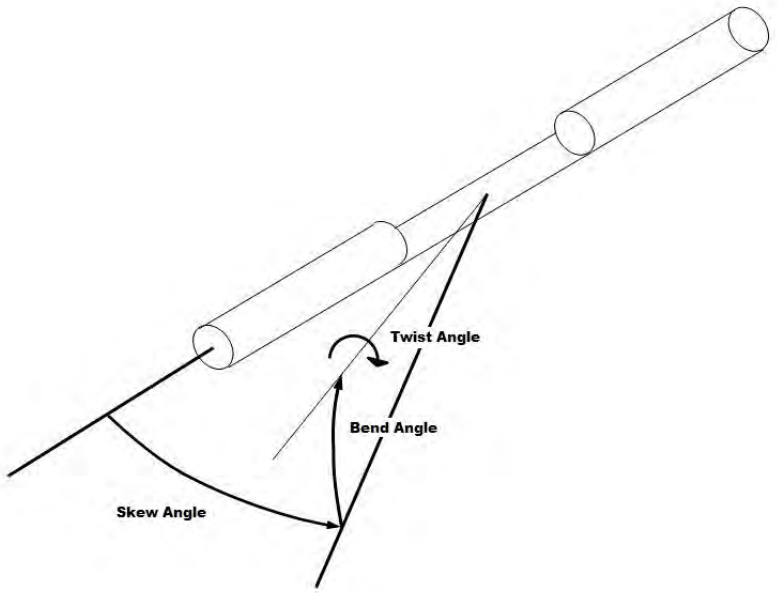

Figure 2.1: Elementary Deformation Angles. Three angles define the orientation of the tape spring free end: skew angle μ, bending angle θ, and twist angle γ.

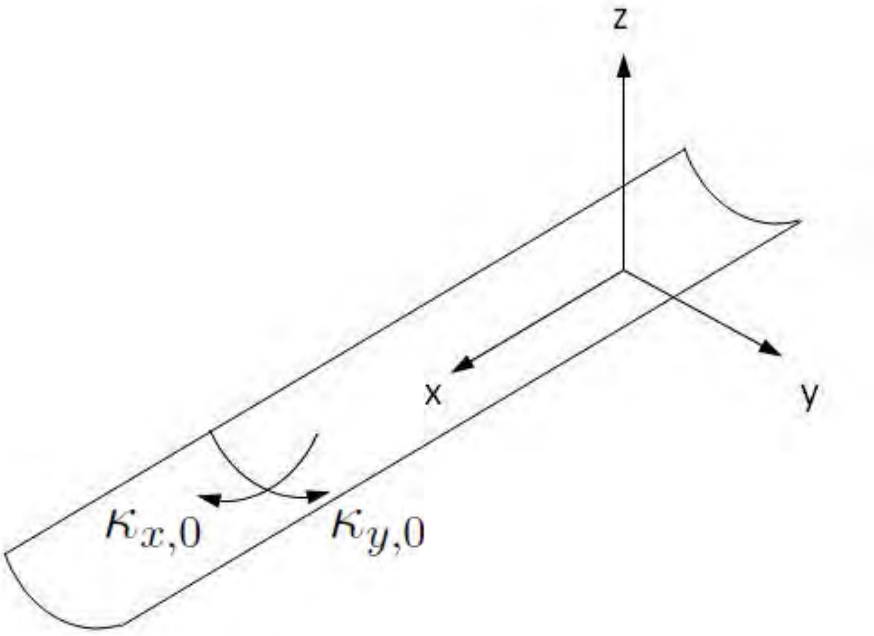

Figure 2.2: The directions of curvatures κ_y, κ_x, and x,y,z unit axes. κ_y is the initial curvature, while κ_x is only nonzero during bending.

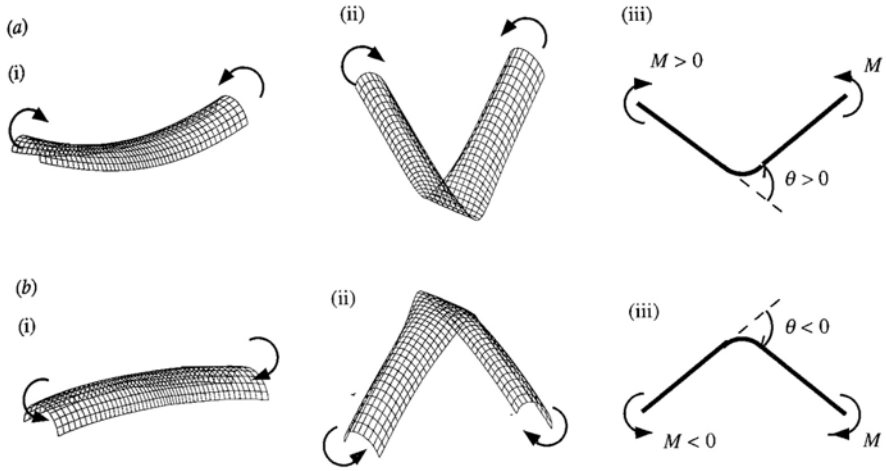

Figure 2.3: Reproduction of Fig 10 from [24]: "Perspective views of tape springs subject to end moments. (a) Opposite-sense bending under a positive bending moment. (i) initial curved deformation. (ii) post-buckled shape; (iii) schematic diagram defining positive fold angle θ. (b) Equal-sense bending under a negative moment."

2.3 One Degree of Freedom

Initially research characterized only one-dimensional bending and the moment required to bend a tape spring to a given bending angle, θ. This response is characterized by "an elastically deformed region with zero transverse curvature and uniform longitudinal curvature. The process of formation and growth of a fold belongs to a wide class of propagating instabilities. It is characterized by a high peak moment and a lower propagation moment." [24] Or in graphical form, Figure 2.4 reproduces Figure 12 from [24]. This figure clearly shows the expected moment from a tape spring hinge as a function of bend angle (or θ) in a simple, one dimensional case. This figure illustrates the deformation as linear and bend angle dependent at low angles (θ) but non-linear and independent of bend angle at higher angles.

More explicitly, one end of the tape spring is constrained in all six degrees of freedom (6DOF), while the other end is allowed to move freely, but with a pure bending moment applied. (In the non-ideal case, the movement may be constrained by equipment to ensure a negligible skew angle.) For small θ, the tape spring will behave linearly, and may be accurately predicted as a Timoshenko Beam [16]. At a specific moment and deflection, depending on bend sense, the tape spring will buckle, developing a hinge shape. This hinge is characterized by zero transverse curvature (κ_y). The limits of θ where the tape spring buckles into a hinge are determined by the maximum moments M_+^{max} and M_-^{max}. This is well illustrated by Figure 2.4 below. In between A and B, the low angle behavior is mostly linear, but for greater bending angles, to the right of A and left of B, the tape spring buckles into a hinge and provides less reaction force.

The moments bounding the low angle region are expressed in Heald et al. [19] by equation (2.1).

$$M_+^{max} = \frac{bT^2\sigma_y}{6} \qquad M_-^{max} = \frac{bT^2\sigma_y}{6}\frac{\nu-1}{\nu+1} \tag{2.1}$$

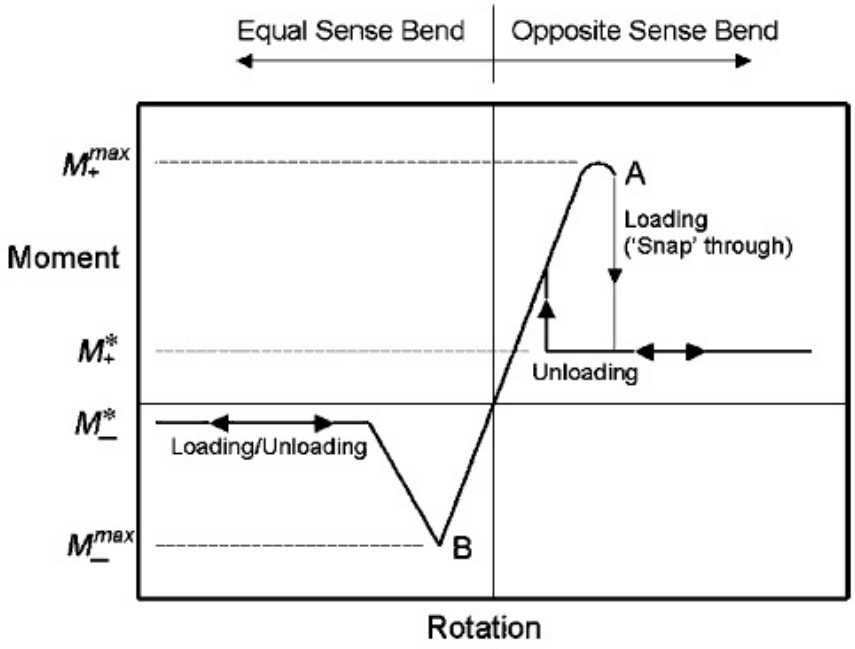

Figure 2.4: Reproduction of Fig. 12 from [15] "Moment-rotation relationship for a two-dimensional tape-spring fold."

As θ increases past point A, the tape spring will buckle and form a hinge. This behavior is illustrated by the flat moment-rotation relationship to the right of point A (and the left of point B) in Figure 2.4 above. This hinge is characterized by "an elastically deformed region with zero transverse curvature and uniform longitudinal curvature. The process of formation and growth of a fold belongs to a wide class of propagating instabilities. It is characterized by a high peak moment and a lower propagation moment." [24] In this high θ region, the curvature will be equal to the unstressed initial curvature ($\kappa_{y,0} = \kappa_x$).

Furthermore the steady state moment this hinge produces is shown in Equation (2.2) and will be independent of angle and instead driven by flexural rigidity (D). [24] Flexural rigidity originates in plate theory, and is represented by equation (2.3) where

E is Young's Modulus, t is thickness, and ν is Poisson's ratio. The signs for these high-angle bending moments are shown in Figure 2.3 above.

$$M_+^* = (1+\nu)D\alpha \qquad\qquad M_-^* = (1-\nu)D\alpha \tag{2.2}$$

$$D = \frac{Et^3}{12(1-\nu^2)} \tag{2.3}$$

2.3.1 Validity Restrictions. Finally, in addition to the θ angle restrictions, the hinge position is restricted. The above equations only hold far away from the rigid supports at each end. Otherwise, the curvature of the tape spring is reinforced and produces a higher resistance to bending. The distance required for this effect to subside is $x \geq 1.5R\alpha^2$ according to [24].

2.4 Two Degree of Freedom Case

The next case allows the location of the hinge to move, but still allows only rotation bending. Obtained by adding an axial force to the free end of the hinge, this is only a slight expansion of the simple one degree of freedom case. Still, the addition is enough to make the case indeterminate under only equilibrium equations. The indeterminancy was recognized and solved by Seffen, You, and Pellegrino [12] by means of an energy formulation. Their solution was to combine energy with the equilibrium formulations, which calculates a minimum energy configuration the tape spring will collapse into.

2.5 Three Dimensional Case

Subsequent research expanded the formulations to allow three-dimensional movement. Tape springs were considered in equilibrium under moment (for bending) and torsion (for twist). Although the following moments and torques are identically equal to the external torques, they are expressions of the internal mechanical re-

actions of the hinges, and should be valid under minimally constrained conditions. The three dimensional solution was originially developed by Mansfield in 1973 [10] as a function $M = M(\kappa_x) \Leftrightarrow M(\theta)$. It was then developed further by [15] into the following form in equations (2.4) through (2.5).

$$\begin{aligned}
\overline{M} = \quad & \overline{\kappa}_x - \overline{\kappa}_{x,0} + \left\{ \tfrac{\lambda}{(1-\nu^2)} \right\} \{\mu \Psi_i(\overline{\kappa}_x) - \lambda \overline{\kappa}_x \Psi_2(\overline{\kappa}_x)\} \qquad (2.4) \\
\mu = \quad & 2\nu\overline{\kappa}_x - \overline{\kappa}_{y,0} - \nu\overline{\kappa}_{x,0} \\
\lambda = \quad & \overline{\kappa}_{xy}^2 - \overline{\kappa}_{xy,0}^2 + (\overline{\kappa}_x - \overline{\kappa}_{x,0})(\nu\overline{\kappa}_x - \overline{\kappa}_{y,0}) \\
\Psi_1(\overline{\kappa}_x) = \quad & \tfrac{1}{\overline{\kappa}_x^2} \left[1 - \tfrac{1}{\overline{\kappa}_x^{1/2}} \tfrac{\cosh(2\overline{\kappa}_x^{1/2}) - \cos(2\overline{\kappa}_x^{1/2})}{\sinh(2\overline{\kappa}_x^{1/2}) + \sin(2\overline{\kappa}_x^{1/2})} \right] \\
\Psi_2(\overline{\kappa}_x) = \quad & \tfrac{1}{\overline{\kappa}_x^4} \left[1 + \tfrac{\sinh(2\overline{\kappa}_x^{1/2}) \sin(2\overline{\kappa}_x^{1/2})}{(\sinh(2\overline{\kappa}_x^{1/2}) + \sin(2\overline{\kappa}_x^{1/2}))^2} - \tfrac{5}{4\overline{\kappa}_x^{1/2}} \tfrac{\cosh(2\overline{\kappa}_x^{1/2}) - \cos(2\overline{\kappa}_x^{1/2})}{\sinh(2\overline{\kappa}_x^{1/2}) + \sin(2\overline{\kappa}_x^{1/2})} \right] \\
\overline{T} = \quad & \overline{\kappa}_{xy} - \overline{\kappa}_{xy,0} + \tfrac{\lambda \overline{\kappa}_{xy} \Psi_1(\overline{\kappa}_x)}{1-\nu} \qquad (2.5)
\end{aligned}$$

Where the curvatures, moment and torsion have been non-dimensionalized per equations (2.6) through (2.8).

$$\overline{M} = \left(\tfrac{3a\{3(1-\nu^2)\}^{1/2}}{Et^4} \right) M \qquad (2.6)$$

$$\overline{\kappa_i} = \left(\tfrac{a^2\{3(1-\nu^2)\}^{1/2}}{4t} \right) \kappa_i \qquad (2.7)$$

$$\overline{T} = \left(\tfrac{3a\{3(1-\nu^2)\}^{1/2}}{4Gt^4} \right) T \qquad (2.8)$$

Rotation and bending at an angle ('twist angle') were added, or equivalently, imperfect folding and other movement constraints. [15]. Several researchers have taken different approaches to the third degree of freedom. Soykasap [25] and Walker and Aglietti [15] formulated the bending of several tape springs embedded at different angles in flat sheet. The sheet was then singly folded, producing bending and skew in the tape springs.

This development led to a simplified model most useful for their specific case, displayed below in Equation (2.9). This equation already takes into account bend (θ), twist (γ), and skew (μ) but also adds the angle of the sheet's fold (β), in a single trigonometric factor.

$$M_{hinge} = M(\theta)\sin(\beta)\cos(\mu) + T(\gamma)\sin(\mu) \tag{2.9}$$

In this equation, θ is the total fold angle, β is the angle of the plane containing the tape spring, γ is the total twist angle, and μ is the skew angle.

2.6 Materials

Once the basic mechanics were discovered, research expanded from simple homogeneous materials into the lighter, stiffer composites such as S-Glass and carbon fiber reinforced plastic (CFRP). Allowable thickness, layup count and layup orientation continue to be investigated [16], [26], [17], [18]. While composites do not change the governing equations discussed above, the combination of fibers and composites will affect the effective structural rigidity of the material. Because the composites are anisotropic, the skew contributions of reaction forces will also differ, but this particular effect has not been investigated yet. Also of note, certain choices of tape spring layups may exhibit bistability and require no restraint while stowed.

2.7 Summary

The analysis of deployable tape springs has already seen significant research. Already, mature solutions for simple structures and antennas are available. However, current models are not yet capable of accurate predictions for less constrained hinges. Unconstrained three dimensional unfolding is what this paper will proceed to investigate.

III. Experimental and Analysis Procedure

The procedures outlined below have been designed to investigate the effect of tape spring geometry on deployment dynamics. Specifically, the motion of a hinge in three degree of freedom unconstrained deployment. The parameters of interest are thickness, width, radius, additional load, subtended angle and skew angle. When varying these design parameters, the current model should take as many of these parameters as necessary into account to accurately predict the motion of the tape spring hinge. If successful, the model will predict hinge motion accurately over the entire range of these design parameters. For this thesis, sufficient accuracy is interpreted as within the error bars of measured data, which are one standard deviation.

3.1 Equipment Properties

3.1.1 Tape Spring Properties. The physical properties of the tape are obtained from simple micrometer measurements. The dimensions of the tape, as measured, are chord length (c), arc length (a), arc depth (d), and thickness (t). What quantities each of these represent is shown below in Figure 3.1.

The initial transverse curvature, $\kappa_{y,0}$ and radius R, are calculated from the arc depth and chord length by equation (3.1). Then the (initial) subtended angle α is calculated via Equation (3.2) derived through simple geometry. Because α and t are representative of initial geometry of the tape spring, they are kept and used in later models. The remainder of the measurements are recorded, but have no further use.

$$\kappa_{y,0} = \frac{1}{R} = \frac{2d}{d^2 + c^2} \tag{3.1}$$

$$\alpha = \frac{a}{R} = a\kappa_{y,0} \tag{3.2}$$

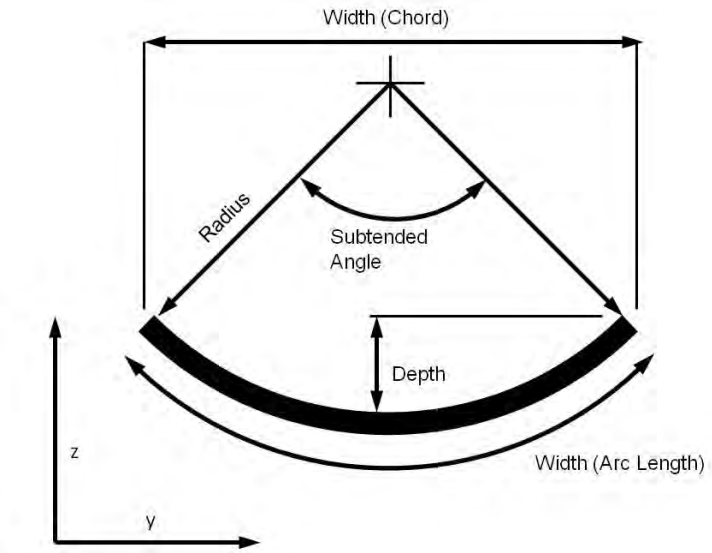

Figure 3.1: The dimensions used to characterize each tape spring are shown here. Curvature and subtended angle are calculated from the above measurements.

3.1.1.1 Steel Tape Springs. The steel tape springs used as hinges are lengths of common, commercial off-the-shelf (COTS) measuring tape. The tape measures are initially cold-formed ribbons (of thickness of 't' and width equal to 'a', its final arc length). The initial ribbons are cold-rolled into transversely curved strips, then spooled onto the holder. Purchased in this form, the tape is then cut to length, measured, and installed in the test equipment. The material properties used for these steel tape springs are representative of industrial stainless steel, and are 205 GPa for Young's Modulus, and 0.29 for Poisson's Ratio.

E ν

3.1.1.2 CFRP Tape Springs. The carbon fiber reinforced plastic (CFRP) tape springs used as hinges were produced at AFRL. These were fabricated with a variety of properties, listed below in properties listed in Table 3.1 and shown in Figure 3.3. The layups among the tape springs vary widely:

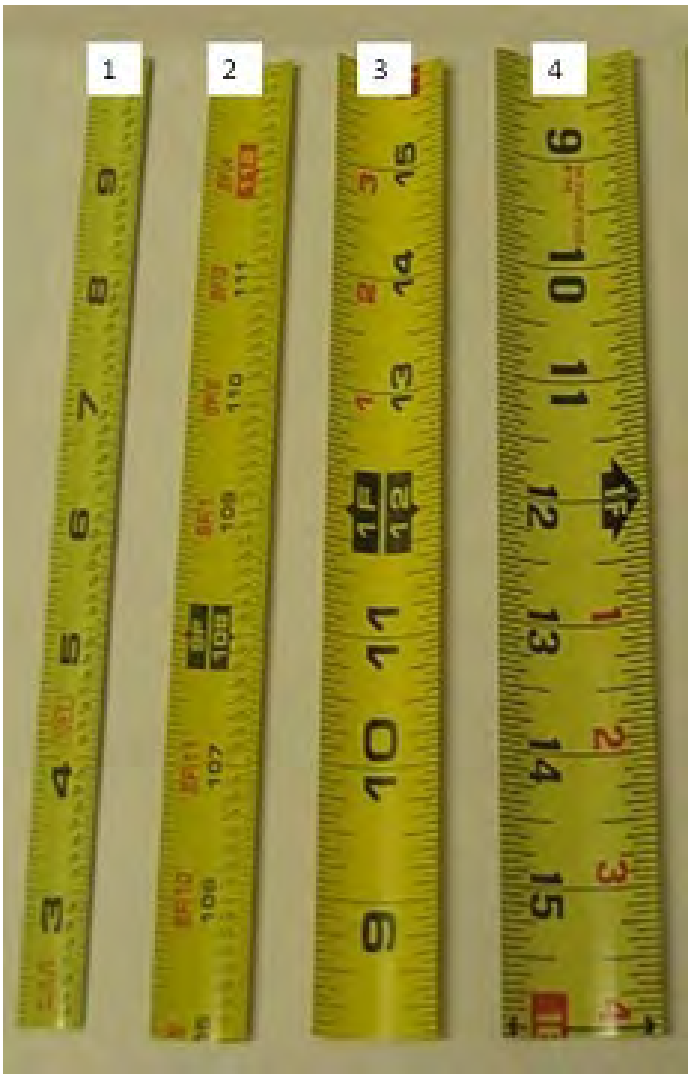

Figure 3.2: Steel Tape Springs procured for the experiments.

3.1.2 Pipe Properties. Each end of the tape spring was attached to Polyvinyl Chloride (PVC) (Schedule 40) plumbing pipe to provide a measurable rigid body and provide inertial resistance. The primary purpose of the pipe is a constraint imposed by the VICON measurement system. VICON triangulates points corresponding to reflective markers. Each of these points is then combined to locate a rigid body. However, the accuracy with which VICON locates the rigid body depends upon the separation of composing points and how much the rotation and position is dependant upon the separation. VICON therefore requires at least three

Table 3.1: Tape Spring Properties. All numbers here were measured directly. Young's Modulus and Poisson's Ratio were approximated by reference values for steel.

#	Thickness t (mm)	Angle α (deg)	Curvature κ (mm$^-$1)	Width a (mm)	Chord c (mm)	Radius R (mm)	Depth d (mm)
Steel							
1	0.09	16.0	0.022	12.44	12.02	44.61	1.65
2	0.10	21.5	0.020	18.78	17.39	50.02	3.12
3	0.10	29.6	0.021	25.03	22.15	48.45	5.36
4	0.11	23.6	0.013	31.63	28.88	76.64	5.65
CFRP							
8	0.210	66.1	0.057	20.31	13.59	17.59	6.42
9	0.130	55.1	0.049	19.65	14.08	20.45	5.62
10	0.130	50.1	0.044	20.10	15.07	22.98	5.63
11	0.280	71.0	0.054	23.13	14.88	18.66	7.40

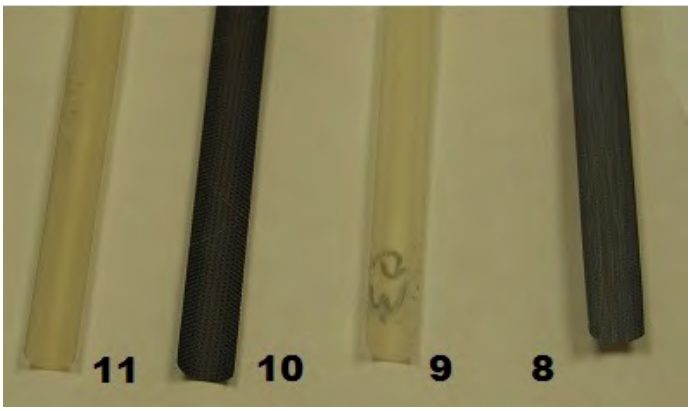

Figure 3.3: CFRP Tape Springs procured for the experiments.

reflective markers to locate the rigid body, and from the rigid body, define the angles of the free end of the tape spring. The PVC pipe primarily provides something rigid to mount these reflectors on.

The inertial resistance of the pipe also counteracts the reaction torque of the hinge, slowing down the acceleration of the free end, and providing more time for the system to track movement. After analysis, this affect will increase the accuracy of the measurements by providing more measurements to process.

The attached to the hinge's free end COTS schedule 40 PVC pipe. In Table 3.2 below, "Size" is the labeled diameter at the store while "Diameter" are actual measured diameters. The specific diameters measured are shown in figure 3.4. While the diameters, thickness and length are not directly used, they provide a useful validation. They are used to calculate the volumetric density, which is known to be about 1.4 g/cc. The same measurements for mass and length can then be used to calculate the second moment of inertia (MOI) of the pipe, shown in the last column. The MOI is a constant used in the model, as discussed below in Section 3.5.

Table 3.2: PVC Pipe Properties. Diameters and thickness were directly measured, while Density and MOI were calculated from measurements and mass.

	Inner Diameter (cm)	Outer Diameter (cm)	Thickness (cm)	Volumetric Density (g/cc)	MOI (g-cm^2)
1	1.57	2.14	0.29	1.41	12699

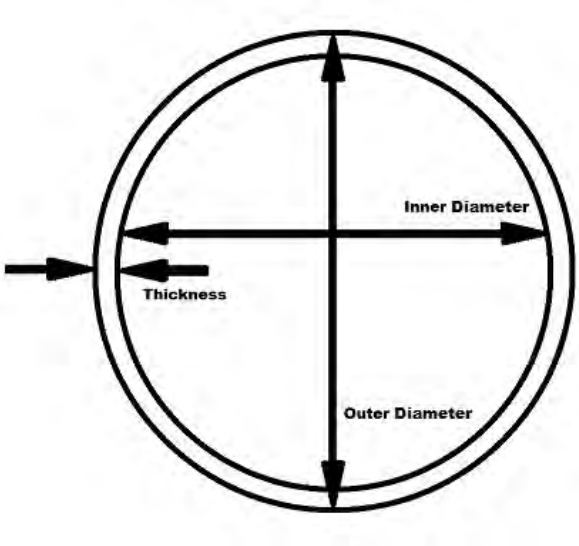

Figure 3.4: Pipe cross section

3.2 Experimental Procedure

The investigation takes the form of fixed-free testing. All experiments are carried out using a VICON motion capture system [27] (performed at AFIT in the

VACE lab). This system consists of 10 cameras which track objects by spherical reflective markers attached to any object of interest. Each marker is detected by each camera, and all of these reflections are correlated with each other to generate a set of points in three dimensional space. These points are then fitted to a number of defined rigid bodies which represent objects the user wishes to track. In this way, the vision system can reconstruct the three-dimensional position and orientation of real-world test equipment, as long as at least three of the markers are mutually visible from at least two cameras. A full description is available in [28].

The VICON system produces the position and orientation of each object as six numbers– three for position in cartesian coordinates, and three for orientation in Euler angles. Through experiment, Jennings et al. [28] have validated a direction cosine matrix (DCM) that reproduces object orientations given the mesaured angles produced by VICON. By inspection, DCM corresponds to a 3-2-1 Inertial-to-Body rotation. This rotation is shown explicitly in (3.3), where γ, θ, and μ are the angles provided from the VICON system for each body.

$$C^{bi} = R_1(\gamma)R_2(\theta)R_3(\mu) = \begin{bmatrix} c_\theta c_\mu & c_\theta s_\mu & -s_\theta \\ -c_\gamma s_\mu + s_\gamma s_\theta c_\mu & c_\gamma c_\mu + s_\gamma s_\theta s_\mu & s_\gamma c_\theta \\ s_\gamma s_\mu + c_\gamma s_\theta c_\mu & -s_\gamma c_\mu + c_\gamma s_\theta s_\mu & c_\gamma c_\theta \end{bmatrix} \tag{3.3}$$

$$\vec{B} = C^{bi}\vec{I} \tag{3.4}$$

3.2.1 Fixed-Free Hinge. For the fixed-free trials, two pieces of pipe are connected by 6" of separation with tape spring connecting them. This setup is displayed as a whole in Figure 3.5. The 6" length was chosen to be long enough for negligible end effects, described extensively as fold-support interaction in [24]. In that paper, the limit for negligible effects is noted as $1.5R\alpha^2$. Using the measurements for tape

springs three and four, the required distances are both 0.76" on each end. Applying these separations leaves about 4.5" for hinge formation.

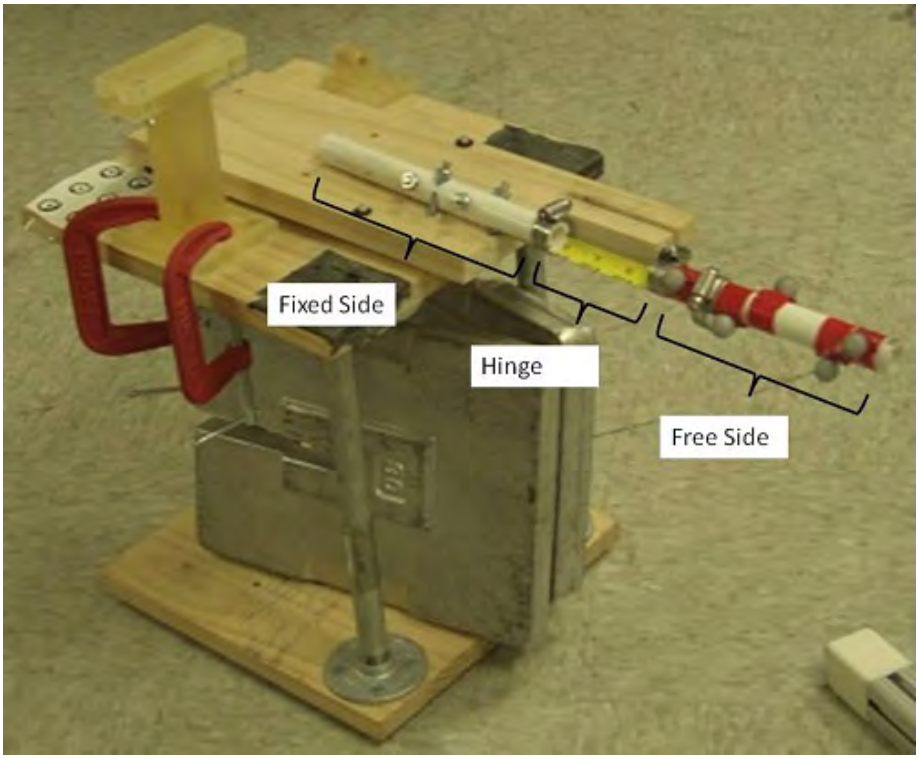

Figure 3.5: Test stand for hinges

One of the PVC lengths is secured rigidly to a weighted platform set on the floor, and the other allowed to freely move. This provides the advantage of unlimited time windows, at the cost of being affected by gravity during deployment. The applied force used to deform the hinge is applied to the free end, and the movements of both ends are recorded.

3.2.2 Gravity Considerations. However, several considerations are necessary due to gravity. The first problem is that gravity must be parallel to the x-z plane (See Figure 2.2). If not, then the tape spring will not finish in a unbended (fully deployed) state. The fully deployed state is not just the minimum strain energy state, but the final state of any on-orbit tape spring deployment. To be of any value in predicting on-orbit structures, the tape spring must finish fully deployed.

The second problem is that once the x-z plane is constrained to be vertical, the direction of the final x axis must be chosen within this plane. Because opposite sense bending is stronger than equal sense bending, any direction where the fully deployed state resists gravity with opposite sense bending will enable the use of weaker tape springs. Although a tape spring strong enough to withstand gravity only by equal sense bending resistance is obviously possible, the necessary size is not available as a COTS component and would require custom production. This leaves roughly half of a circle where gravity will not prevent full deployment (corresponding to the z axis pointing up). Movement throughout this entire range is valid, but the direction used in this thesis was chosen for ease and cost of apparatus construction.

3.3 Repetition and Parameter Variation

Once the test equipment was complete, a large number of trials were run on the fixed-free Hinge. These trials are first broken up into blocks, and each block varies only a single parameter. The First block is geometry, the second block is thickness, the third weight, and the fourth skew angle. Within each block, are a number of sets. For each set, the design variables are kept as consistent as possible. Within each set is a number of trials, aimed at around 30, but sometimes more. Each trial is a single deflection of the hinge under conditions specified by its parent set and block. These blocks and sets are detailed in the following sections.

For each trial, the data is generated from every valid trial in the set, and is analyzed along the deployment path from release until just after the first bounce. Measured from the time of release, position at every time sample is used to generate the mean and standard deviation. In general these show consistent movement until impact. At impact, the error bars grow dramatically, even if errors shrink again during rebound. The large errors at impact indicate the spread of impact times, due to timing inaccuracies, and the variation in actual fall time. Trials for fixed-free hinges last until the steady-state position is achieved by the deployed hinge.

3.3.1 Block 1: Tape Spring Width. To investigate the effects of tape geometry, a variety of tape springs were obtained, and compared against each other. Each of these samples has a similar structural thickness (about 0.1 mm) but a range of width, curvatures and layups. The tape number in Table 3.3 refers entries in Table 3.1 above. The pipe used on the free end is pipe number one from Table 3.2. The model being tested only includes the subtended angle α of the tape spring in predictions, so agreement would indicate that a tape spring's movement would depend only on its initial curvature and not on its width. Because this variation is not included in the Seffen-Pellegrino Model, significant errors are expected.

Table 3.3: Tape Spring Material Testing Block

Set (#)	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8
Tape (#)	1	2	3	4	8	9	10	11
Pipe (#)	1	1	1	1	1	1	1	1

3.3.2 Block 2: Tape Spring Thickness. Next, to simulate thicker tapes, multiple thicknesses of each tape are combined together as the same joint. Block two will compare the response with single, double, triple, and quadruple thickness tape springs. Table 3.4 below lists the sets to perform. Again, the Tape # refers to the tape springs listed in Table 3.1. The notation $3x\,4$ indicates that three thickness of tape spring number four were combined into a single hinge, where $t_{total} = 4t_{single}$.

The Seffen-Pellegrino model already predicts a cubic dependence on thickness, so the model is expect to track with negligible error according to Equation (2.2).

Table 3.4: Thickness Testing Block

Set (#)	2.1	2.2	2.3	2.4
Tape (#)	1x 4	2x 4	3x 4	4x 4
Pipe (#)	1	1	1	1

In this block, several tape springs are combined to emulate a thicker monolithic tape spring. In this case, slip effects are assumed negligible. Because both ends of

the tape spring are clamped to the end pipes and constrained in place, each end is assumed to be constrained in six degree of freedom. Consider the tape spring under a pure moment such that it is bent into a curve past its region of purely linear response. For a material element within the bent hinge, and on the interior of the curve, stress will be primarily compression. The two compressive forces acting on this element will mostly cancel out, but will also create a resultant force away from the center of the curve. Conversely, a material element in a layer on the exterior of the hinge will experience less curvature, less relative extension, and therefore will be under tension. The combination of tensions pulling on this element will create a resultant pulling towards the center of the curve. The exterior layer will therefore be pulled inward towards the interior layers. In short, the assumption is that shear between adjacent layers is negligible because of end constraints. If this assumption is wrong, the discrepancy should be visible in the resulting data.

3.3.3 Block 3: Tape Spring Load. The tape spring is then run with additional weights attached to the free side to test the hinge response with respect to inertial force. Current models predicted that the force produced by a hinge will not depend on load, only angle. If this is true, the response will be linear with respect to inertia load. This load applied is simple weight attached to the free end pipe. Set #1 has no additional weight, Set #2 is the pipe + 19g, etc. Note that the weight increments below were only chosen to be increments on the order of 10% of the free end pipe. In other words enough weight to be visible, but not enough to swamp the data. After a quick survey, a large bolt, washers and nuts were found to provide the appropriate range of weight, as well as ease of attachment. Each combination is labeled as a set according to Table 3.5. The 4x 4 tape spring was used for all sets and is described in Table 3.1.

3.3.4 Block 4: Tape Spring Skew Angle. Next, to investigate the effects of skew on tape spring behavior, a hinge is run at a range of skew rotation angles.

Table 3.5: Additional Weight Testing Block.

Set (#)	3.1	3.2	3.3	3.4
Additional Weight (g)	0	+19	+51	+83
Tape (#)	4x 4	4x 4	4x 4	4x 4

Each combination of tape spring and angle is labeled as a set according to Table 3.6. The 4x 4 tape spring was used for all sets and is described in Table 3.1. The expected behavior here is not yet established. However the default model does not correcting for skew angle at all, and is not expected to be sufficient. This data block is expected to provide enough data to develop a suitable model.

Table 3.6: Skew Angle Testing Block.

Set (#)	4.1	4.2	4.3	4.4
Tape (#)	4x 4	4x 4	4x 4	4x 4
Angle (°)	0	15	30	45

3.4 Data Processing

Through the course of experimentation, a large amount of data was collected across multiple dates, equipment configurations, and sets of procedures. Because of this, trial processing was automated as much as possible.

3.4.1 Hinge Deformation. The actual tests start with the deformation of a hinge repeatedly under each chosen combination of design parameters. To conduct each trial, a length of a simple length of PVC pipe is attached to a geared electric motor, which runs at one revolution per minute. This length of pipe therefore causes one deformation per minute to the tape spring, in a sequence illustrated by Figure 3.7 and in a relative position shown in Figure 3.6. This sequence is driven by the motor rotating the actuator arm in counter clockwise direction. The arm contacts the pipe connected to the free end of the tape spring hinge, pushing it upwards. The arm continues to push the free end upwards until it rotates out of the way (middle bottom picture in Figure 3.7). At this point, the hinge is no longer held

by the arm, and releases into the deployment dynamics of interest. The hinge will then settle down through numerous oscillations and will be effectively at rest by the time the actuator arm rotates back around. The speed of the actuator arm was chosen primarily to match the performance of an affordably sized COTS motors while supplying enough torque to overcome the hinge's initial reaction force through gearing. A faster deformation is easily obtainable with a more powerful motor and an appropriate gearing ratio.

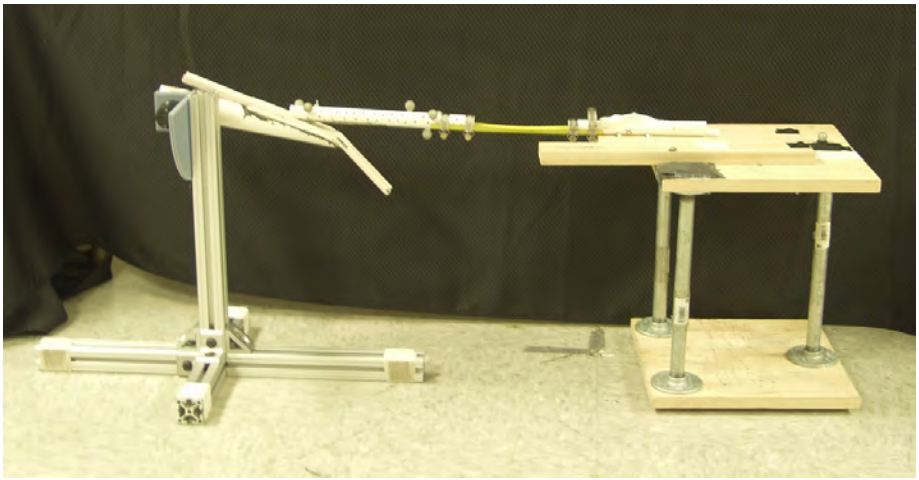

Figure 3.6: The actuator arm and a fixed-free hinge positioned as they would be in a trial. No other connection is necessary besides electric power.

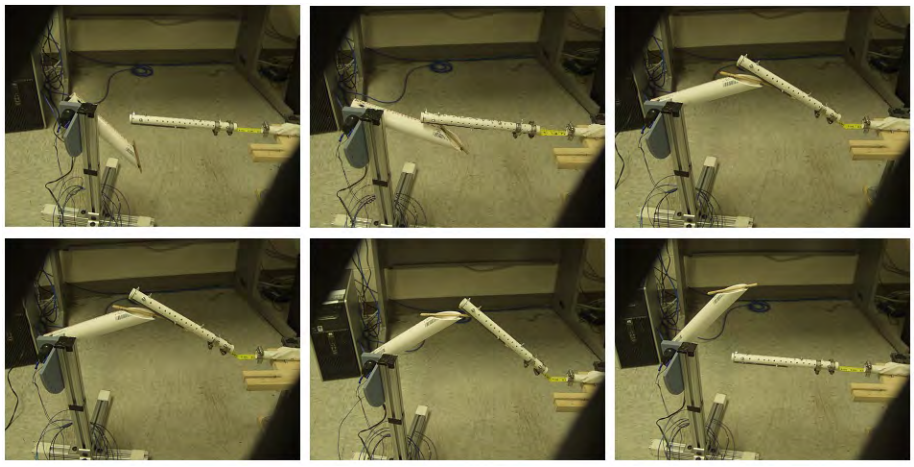

Figure 3.7: The actuator arm deforming the tape spring hinge. Images proceed from left to right, row by row.

Because inducing buckling in a tape spring is not a deterministic process, the hinge may buckle to either side of the intended axis. Effectively an instability, this tendency requires care in fashioning the actuator arm. Several revisions of the arm itself were performed before consistent deformation were produced with a large enough range of motion. As an example, Figure 3.8 shows one of the unsuccesful iterations where the tape spring free end would fall off either side of the arm before 10° of deformation. The intended behavior was for a tongue-in-groove arrangement where the wooden dowl attached to the actuator arm on the bottom would fit in between the dowls glued to the free end pipe on the middle or top. The specific issue was that when misaligned, the actuator would have no restoring force to realign the tape spring free end. When a wider catch area was added to the actuator arm is subsequent revisions, this issue was solved. This feature is shown in Figure 3.9, the final design, which generates about 45° of deformation.

3.4.2 Human Induced Deformation. Due to equipment failure in the actuator arm, not all of the trials were motor-driven. Sets three and four in Block 3 were deflected by a human operator pulling on a string looped around the free end of the hinge. This string was run through a series of hinges to reproduce the same release conditions as the motor driven trials.

3.4.3 Data Flow from the VICON System. Data is first gathered by the VICON system. Ten cameras spread around the walls of the experiment room (The VACE Lab at AFIT). Each camera illuminates its field of view with an array of LEDs at a specific wavelength. Each return for a camera is recorded, and passed to a collecting computer which triangulates the returns, and produces a tracking point for each return which is consistent among three cameras (this number is a user setting). Each of these tracking points may be assigned to a body, which the collecting computer can then track as a body. Because the orientation of this body is uniquely identified by at least three points (assuming these points are arranged

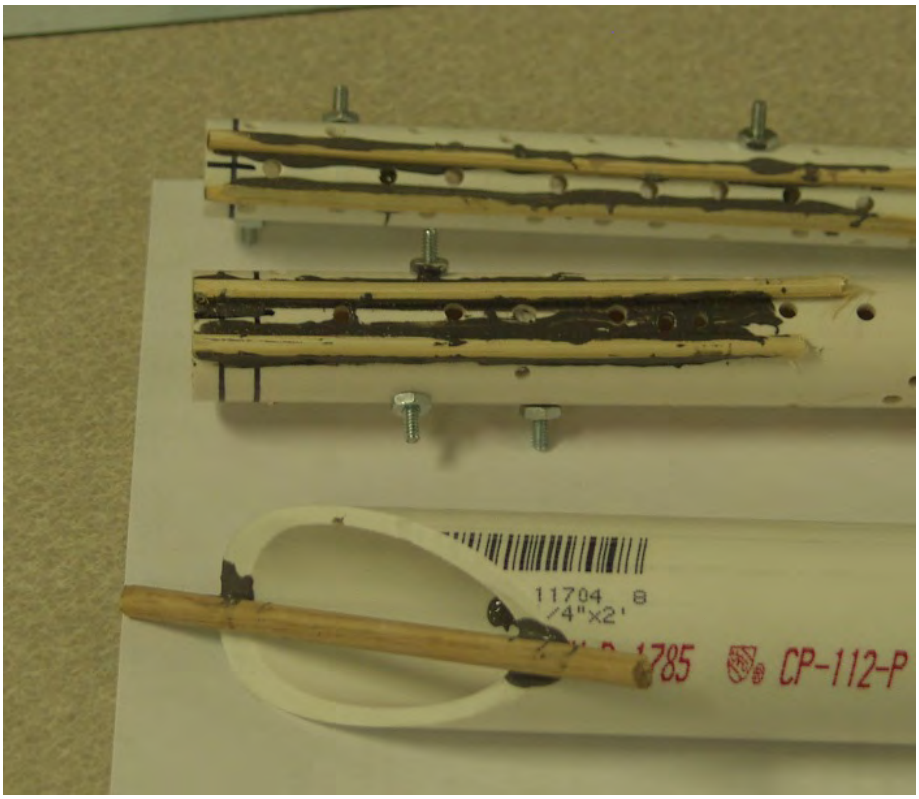

Figure 3.8: Unsuccessful actuator arm head. This particular revision was unstable in the lateral direction.

assymetrically.) Thus, the VICON system provides the orientation and position of each body it is asked to track.

3.4.4 Data Recording in the Lab. A separate LabVIEW project is used to poll the VICON system, which returns the requested data frames. At any given time, these data frames will be include a timestamp, along with the orientation and position for each rigid body. This LabVIEW software operates instructions written by AFIT, and is the first AFIT software in the data processing chain. It was predominantly written by Alan Jennings and Greg Briggs during the course of [28]. Once received, the data is recorded to a data file, and kept for later analysis in Matlab.

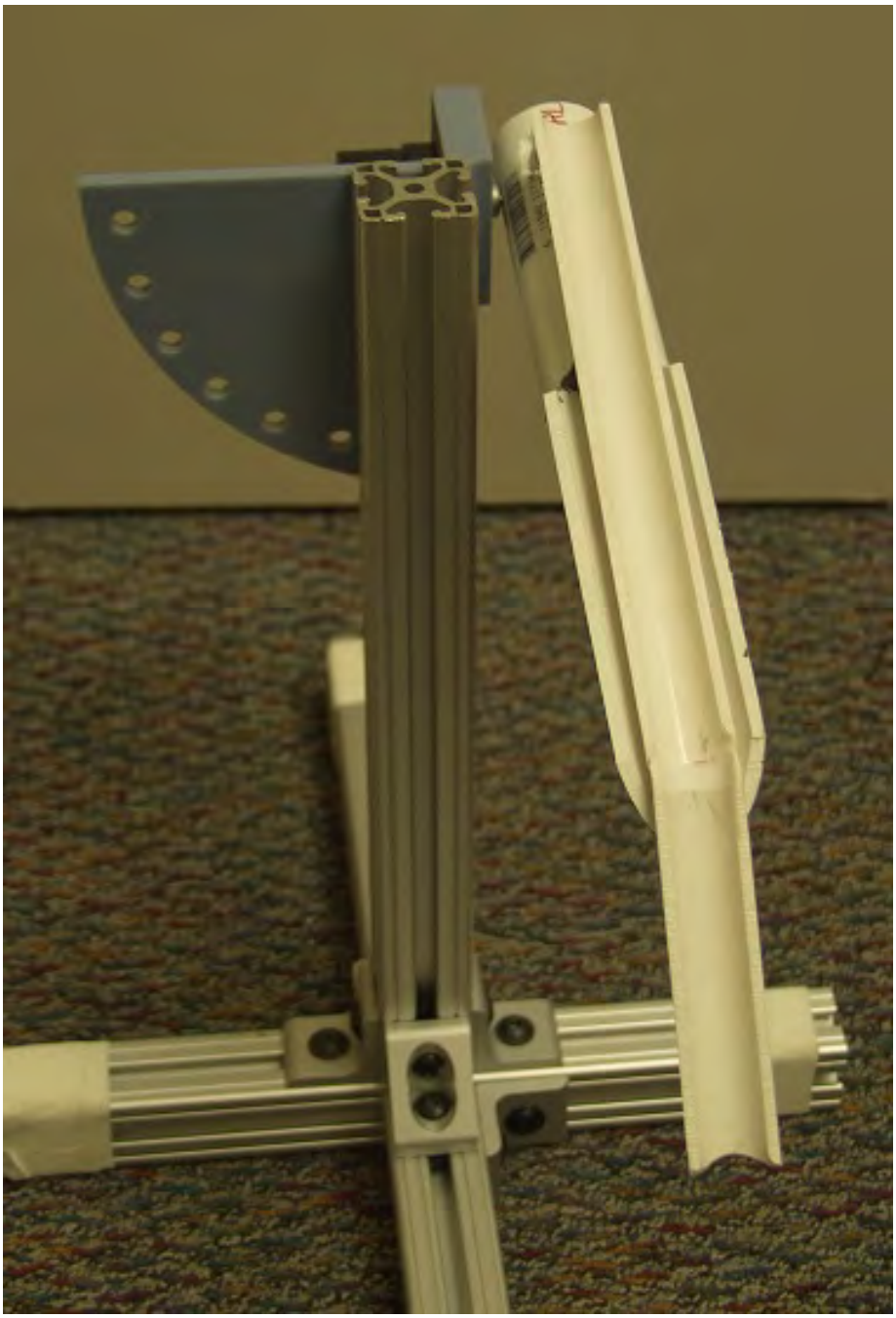

Figure 3.9: Final actuator arm head design. This configuration allowed moderate stability with an acceptable deformed angle in the hinge.

3.4.5 Raw Data Processing. The stored data files are the input for analysis which was performed in Matlab® Version 7.14.0.739 (R2012a) [29]. Processing encompasses several steps— first, simple checks are run to reject improper formatting,

and duplicate or missing frames. Second, the data is split into individual trials, one for each deformation. Third, each of these trials are scanned for snap-through or similiar failures. Snap-through is detected when the free end falls through the home position (which may happen either directly or rolling off to either side). In data this is flagged by a u_3 (vertical) displacement values below a specific negative threshold. Fourth, each deformation is scanned to locate the release point. The release point is found by searching from the maximum vertical displacement forward in time until the displacement starts to fall beyond a certain threshold. The interval from this release until the first bounce is the interval where the model is applied. The model is given the initial conditions of the release point, then the model is integrated forward and compared to the measured values.

3.5 Deflection Model

But before the hinge torque model can be compared to the measured torque, the effect of gravity must be corrected for. The hinge's deformation is measured about the y axis by θ, and about the z axis by μ. The second time derivative of this angle is the downward angular acceleration and is related to the applied torques by Equation (3.5). Rearranging to separate the hinge torque and gravitational torque produces Equation (3.6). In these equations, I is a general moment of inertia as observed by the hinge.

$$\sum \tau_i = \ddot{\theta} I_{yy} \tag{3.5}$$

$$\ddot{\theta} = \frac{\tau_{hinge}}{I_{yy}} + \frac{\tau_{grav}}{I_{yy}} \tag{3.6}$$

These τ terms may be integrated twice (with respect to time) to generate an θ profile governed by Equation (3.10,3.9) below. In this research specifically, these integrals are numerically calculated from initial conditions. When supplying the

torque predicted by a given model, the θ profile generated will correspond to that model. This θ profile then, is a prediction by a given hinge torque model. When the θ profile is subtracted from the measured θ values, the difference is the error on which any hinge torque model is judged. The quality of fit is dependent upon both the overall error magnitude, as well as the rate of growth.

3.5.1 Second Moment of Inertia. The moment of inertia used in Equations (3.5, 3.6) is estimated by Equation (3.7) below, where the m is the mass of the PVC pipe including all fittings (screws, nuts, clamps and reflectors), L_h is the distance from the hinge center to the pipe end, L_p is the length of the pipe, L_w is the distance from the hinge center to the mounting point of additional mass, and w is the additional mass. As implied by Equation (3.7), the mass of the pipe is assumed to be uniformly distributed along its length; the pipe's center-of-mass moment of inertia is represented by the first term. The second term of the same Equation (3.7) is a parallel axis correction from the pipe center of mass to the hinge center. The third term is only nonzero when additional mass is added in block 3. Although expressed as I_{yy}, this MOI is also valid for any rotation axis through the hinge and perpendicular to the x axis– in other words, I_{yy} is valid for all values of bend angle (θ) and skew angle (μ).

$$I_{yy} = \frac{1}{3}mL_p^2 + mL_h^2 + wL_w^2 \tag{3.7}$$

3.5.2 Gravity Correction. Before this any hinge torque model can be compared to the measured angles, the effect of gravity must be taken into account. The effect of gravity is modeled as a simple point mass at a given distance from the hinge center, at the bend angle (θ) upwards, and the skew angle (μ) to the side. The torque due to gravity (τ_{grav}), expressed by Equation (3.8), is divided by the second moment of inertia of the pipe free end (I_{yy}) then integrated twice to produce the expected deflection due to gravity. These operations are shown by Equation

(3.9). This gravity-only deflection is useful as a baseline for what better models may be compared to, and further highlights how much effect the hinge torque has on deflection.

$$\tau_{grav} = r(m+w)g\cos\theta\cos\mu \tag{3.8}$$

$$\theta_{model,grav} = \iint \frac{\tau_{grav}}{I_{yy}} dt\, dt \tag{3.9}$$

3.5.3 Complete Model. For the experiments discussed in this paper, the torque and movement models are started at hinge release– the moment the tape spring starts behaving in unconstrained deployment dynamics. This implies that all initial conditions for the integrals in Equations (3.9 are the initial conditions for release. Conditions at the time of release, t_r, are zero initial velocity ($\dot{\theta}_0 = 0$), and intial position that matches the measured position at release ($\theta_0 = \theta(t_r)$). These models are numerically integrated forward until they are no longer interesting. In this thesis, that point is when the tape spring hits $\theta = 0$, and rebounds. Behavior past this point is drastically different and outside the scope of this research. In order to fully predict the result of a given hinge torque model, the integrated torque due to a model is added with the torque due to gravity, and combined into a full model prediction, $\theta_{model,full}$ shown below in Equation (3.10). The data resulting from this equation will then be compared to the measured data, and judged by how accurately it matches.

$$\theta_{model,full} = \iint \frac{\tau_{hinge}}{I_{yy}} dt\, dt + \iint \frac{\tau_{grav}}{I_{yy}} dt\, dt \tag{3.10}$$

3.6 Hinge Moment, Seffen and Pellegrino Model

The hinge torque evaluated here is was proposed in a paper by Seffen and Pellegrino [24]. The model is a simple prediction for tape spring hinge's reaction

moment, duplicated here from Chapter II as Equations (3.11), (3.12), and (3.13). These equations describe the torque applied by the hinge to attached material under equilibrium, but used here to predict the force under dynamic conditions. According to this model, the reaction torque is independent of load, skew angle, bending angle, and tape spring width, but depend primarily on tape spring thickness and subtended angle. All of these parameters are varied in this thesis, and the predicted θ angle calculated by Equations (3.12, 3.13, and 3.10 should match the measured θ.

$$M_+^* = (1+\nu)D\alpha \tag{3.11}$$

$$M_-^* = (1-\nu)D\alpha \tag{3.12}$$

$$D = \frac{Et^3}{12(1-\nu^2)} \tag{3.13}$$

Because the hinges are deflected (intentionally) in only equal sense bending, the full model reaction torque is

$$\tau_{hinge} = M_-^* \tag{3.14}$$

The hinge torque is then evaluated at each point in the trajectory, divided by the inertia, and integrated forward through time twice to produce a θ profile. A profile that corresponds to the prediction of the hinge torque model and is suitable for comparison to measured data. Once the measured data was obtained, several problems and noise sources were identified.

3.7 Snap-Through Occurrence

The most dramatic problem exhibited was snap-through. Snap-through occurs when the tape spring is not strong enough to prevent rotation through the fully deployed position, and the free end continues rotating downwards. Although accept-

able in microgravity, in these experiments the tape spring was not strong enough to subsequently return the free end to its initial, undeformed position.

Unfortunately, most of the tape springs chosen were too weak to resist this motion. As discussed above, the free end pipe that adds so much momentum is necessary to track the movemont of the tape spring. Without the pipe, no experiment data is available whether snap through occurs or not. Snap through only occurred in testing Block 1, and the sets where it did occured are shown specifically in Table 3.7. The sets with sufficient stiffness to rebound are numbered, while the sets with snap-through have a '-' in place of the set number. As above, the tape spring numbers reference Table 3.1.

Table 3.7: Material Testing Block, Showing Successful Tests

Set	-	-	1.3	1.4	-	-	-	-
Tape Spring #	1	2	3	4	8	9	10	11
Pipe #	1	1	1	1	1	1	1	1

3.8 Noise Sources

3.8.1 Inconsistent Deflection. One source of error was noisy data sets from fuzzy, noisy, or inconsistent releases. An example is shown below in Figure 3.10. Even if the data path from sensing to analysis is clean, the error source displayed here is of the hinge movement itself. If the deformation force was unsteady, noisy, or inconsistent not only is the release point ambigious, but the release conditions will vary from trial to trial. This type of data is caused by a careless human operator, although it may be lessened through sufficient attention and re-running of bad trials. On the other hand, automatic motor-driven actuation reliably produces more consistent and smoother data sets. Of the 14 data sets used for analysis, only three were actuated by hand, and only then because of equipment failure. In either case, human attention is still required to ensure that invalid trials are discarded.

Figure 3.10 below shows the displacements (u_1, u_2, u_3 in the x,y,z directions, respectively) over an entire time trial. This Figure is not intended for detailed discussion, but primarily to demonstrate the amount of noise in comparison to Figure 3.11.

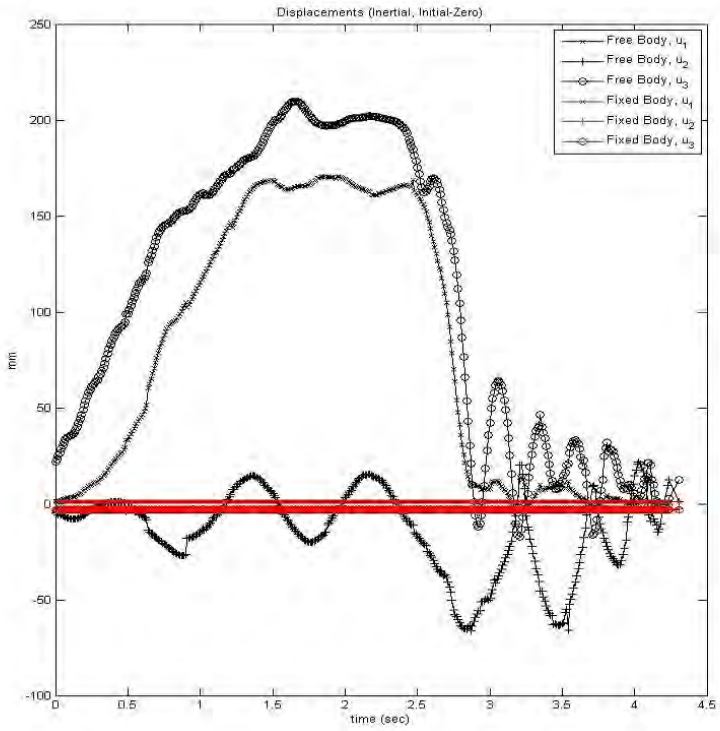

Figure 3.10: Invalid time trial. Relative Displacement (mm).

The black samples track the free end (aka 'free side' or 'free body'), and the red samples track the fixed side of the tape spring anchored to the floor. (See Figure 3.5 for reference). The trial consists of several stages. At first, the tape spring is laying in its undeformed position, shown by the upper right frame of Figure 3.7 (this stage is not shown in the trial below). The next stage is deformation, during which the actuator arm pushes the free end up, cause a hinge to form in the tape spring. When the arm rotates out of the way, the tape spring is released and starts to deploy. Release occurs at about 12 seconds in Figure 3.11. From release, the tape spring falls (or deploys) freely until it hits the original, undeformed position, and rebounds. Rebound occurs at about 12.1 seconds below. The tape spring continues

to bounce and settle down for some time afterwards (not completely shown). The most prevalent cause of processing errors is the ambiguity of release. For comparison, Figure 3.10 suggests release at about 2.5 seconds, and again at about 2.7 seconds but Figure 3.11 shows a clear, sharp corner at 12 seconds. Even if the second fall in Figure 3.10 is unconstrained, it has no appropriate point to assume release and start modeling (i.e. no point that has zero velocity initial condition). A secondary concern is that the deployment trajectory seems too slow, and may be externally dragged by some part of the test apparatus. Because of these issues, the setup used for Figure 3.10 above was not used for any further trials.

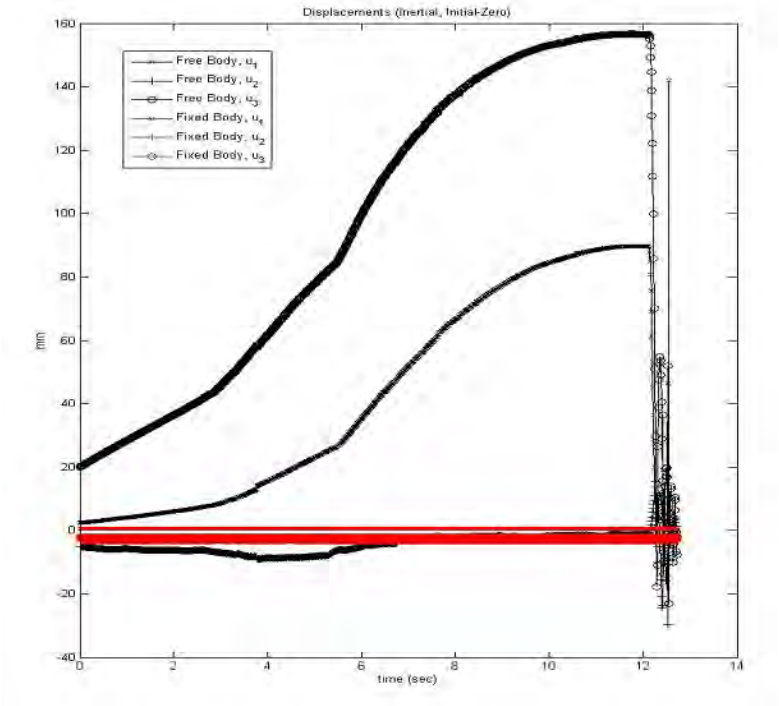

Figure 3.11: Valid time trial. Relative Displacement (mm).

3.8.2 Noise in Sensing. A second source of noise is due to a specific processing step within the VICON system. When triangulating the reflective markers and then fitting the marker points to a rigid body, the system generates path discontinuities. An example of this is shown below in Figure 3.12. In other words, the measurements jump up or down for a short period of time, then return to the

original path. Unfortunately, this sort of error resists automated correction, because the magnitude of this discontinuity is smaller than the movement during the deployments of interest. Also, the example shown in Figure 3.12 is during the relatively smooth and slow deployment phase, and is thus obvious to an observer. The discontinuities that may cause problems would occur during deployment, and will be hidden by the large magnitude movements. They will only be visible as increases in the confidence intervals, and are probably not correctable.

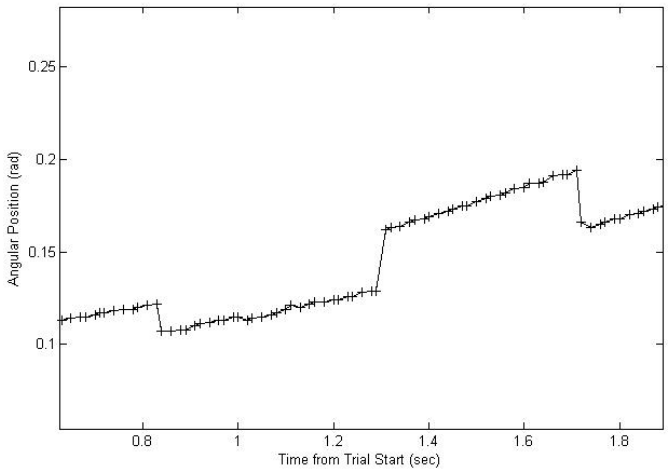

Figure 3.12: Data Discontinuity in VICON data. Relative Displacement (rad).

3.8.3 Summary. This chapter starts off by giving the physical measurements and parameters most relevant to the lab equipment. describes a method for repeatedly deflecting a tape spring hinge, recording its movement, and passing the recorded data on for analysis. This chapter then describes the model being evaluated, and how well it compares to the measured data. The next chapter will continue with the results obtained, how well it matches the predicted model, what the implications are for an improved model.

IV. Analysis

The organization for all of the blocks is similar. For all subsequent trial plots in this chapter, the top plot compares measured and modeled angles θ and μ. The bottom plots are the error between the measured and modeled bend angle θ compared to the variance in the measured data. The number of tape springs combined to simulate a thicker tape spring is indicated by the notation $3x\,4$ where the first number is how many tape springs were combined and the second number is which number tape spring was used. The number tape spring used refers to Table 3.1 above.

For each plot, the data is generated as a mean of every trial in the set. The data follows the path of movement from release t_r through rebound until the first bounce t_s. This path is literally $\theta_{measured}(t)$ on the interval (t_r, t_s). For comparison, the times of each trial are aligned to $t_r = 0$ so that the time measurements are time from release. The position at every point in time from every trial is then used to generate the mean and standard deviation of the measured angles at that time. When theta reaches zero, impact occurs, and the tape spring rebounds. Impact may be recognized in each of the deflection angle plots by the dramatically larger error bars at that point.

Each of the plots below show the deployment of the hinge compared to the models. Typically, from top to bottom, the lines are a gravity-only model ($\triangle$, calculated by (3.9)), a full model (∇, calculated by (3.10)), measured tape spring bend θ angle, and measured tape spring skew (μ) angle (where visible). The second plot for each trial is a simple error: the difference between the full-model-predicted angle and the measured angle. This indicates how well (or badly) the model predicts actual movement.

4.1 Geometry Variations

Block one compared two tape springs of similar thicknesses, but different geometries. The tape spring width varied from 12.4 to 32 mm and the subtended angle

from 16 to 29.6°— although only the subtended angle is actually accounted for in the Seffen-Pellegrino model. Unfortunately some geometries were too weak, and failed by snapping through the fully deployed position. Those data sets were not included in the analysis. The range of geometries for measured data was therefore smaller. The tape spring width varied from 25 mm in set four to 32 mm in Set four. The subtended angle varied from 16° in Set three to 29.6° in Set four.

In Set three, the resulting error terms are greater than 0.2 rad (11°) and seem to correlate with θ. Furthermore, the error terms do not show this same correlation with θ in Set three as they do in Set four. In other words, in Set four the path matches very well (within 0.1 rad (5°)), while in Set three the model overestimates the path, even after initially matching.

These patterns suggest two corrections. First, because the path in Figure 4.1 underpredicts then overpredicts, the hinge torque probably depends on bend angle θ. Second, because the path mismatch in Figure 4.1 is opposite from the path mismatch in Figure 4.2, the hinge torque is probably dependent upon tape spring width.

The dependence on bend angle is not focused on in the literature but is certainly corroborated by other authors' data (for example, Figure 16 in [24] clearly shows a dependence on bend angle θ). Although the error on hinge number four is less (0.1 vs 0.2 rad) than that of three, the error still grows until impact. Because the model error contributes less under these conditions, the random noise has a correspondingly greater contribution. This dependence on bend angle (θ) is the first model correction term identified in this research, and the dependence on tape spring width a is the second term.

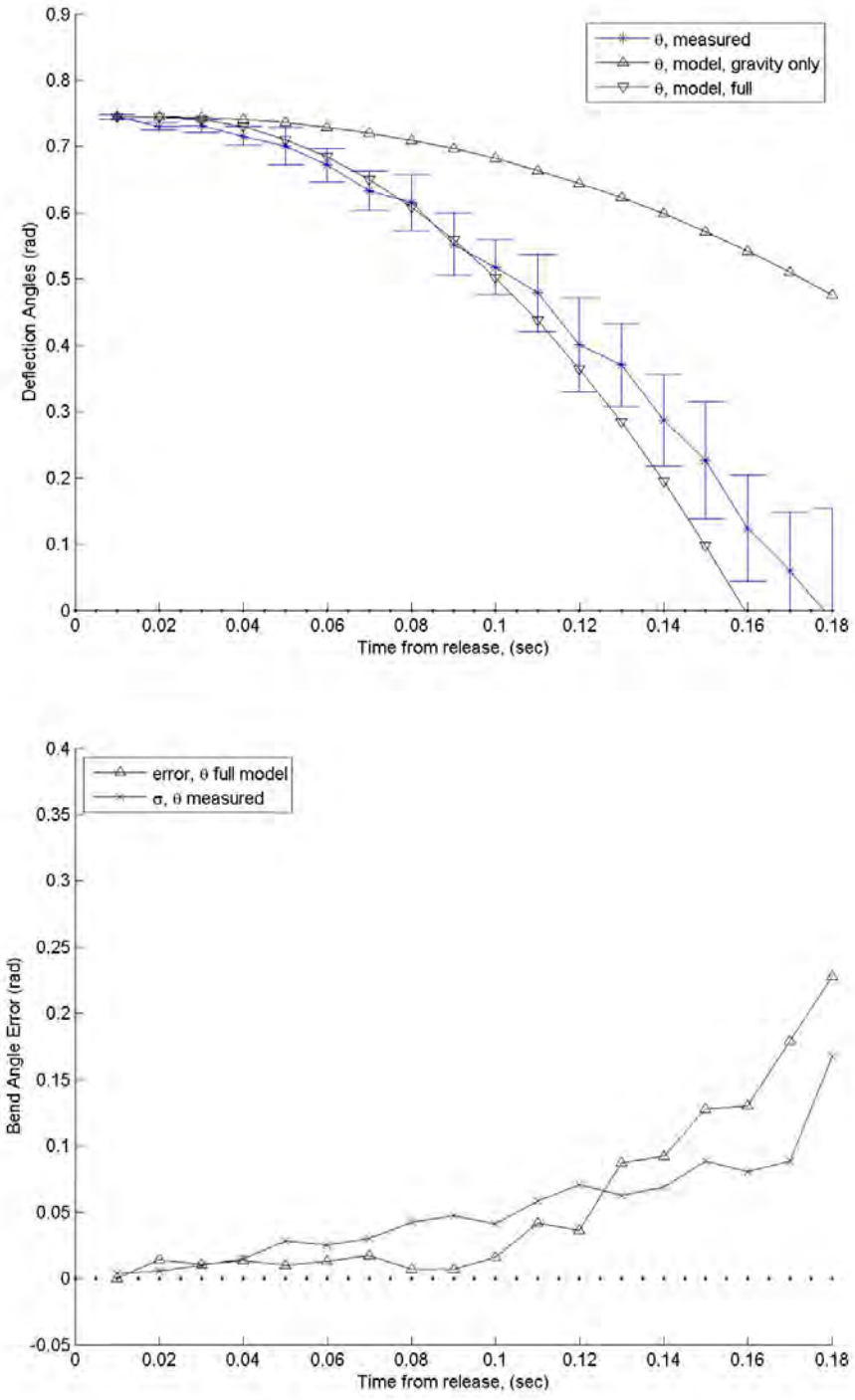

Figure 4.1: Block 1, Set 3, 31 Trials, Tape spring 3x4. Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

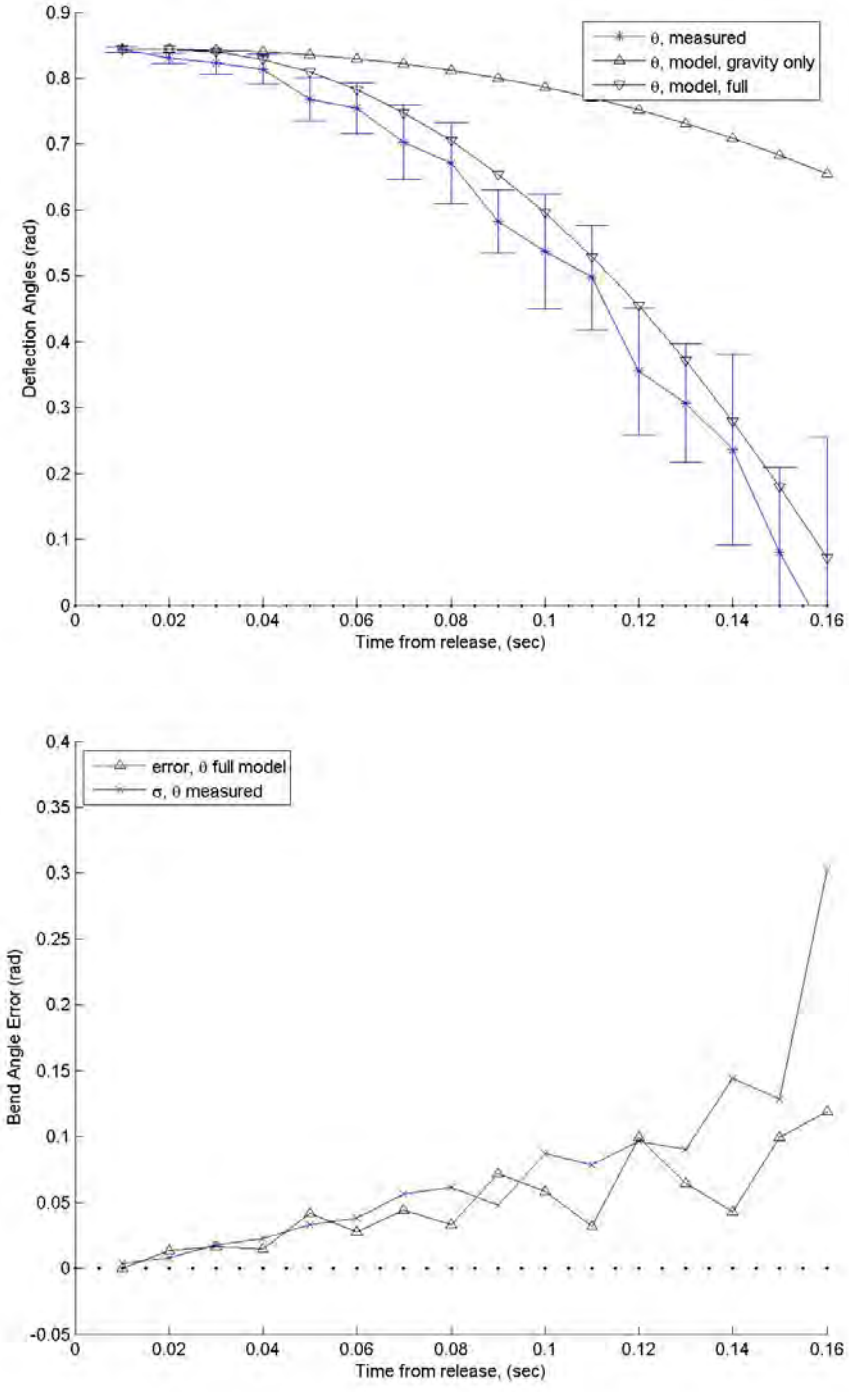

Figure 4.2: Block 1, Set 4, 27 Trials, Tape spring 4x4. Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

4.2 Thickness Variations

The second block of trials varied only the thickness of the tape springs. The average angles and errors for these sets are shown in Figures 4.3 through 4.6. As expected, increasing the tape spring thickness increases the hinge stiffness, which in turn causes greater acceleration and faster deployments. Impact time decreases from about 0.2 seconds with a single thickness (Figure 4.3) to 0.16 seconds with a quadruple thickness (Figure 4.6). However, as the thickness increases, the accuracy of the model increases, this indicates that the real behavior varies less with thickness than current theory suggests, but that the moment is stronger, especially for thinner materials.

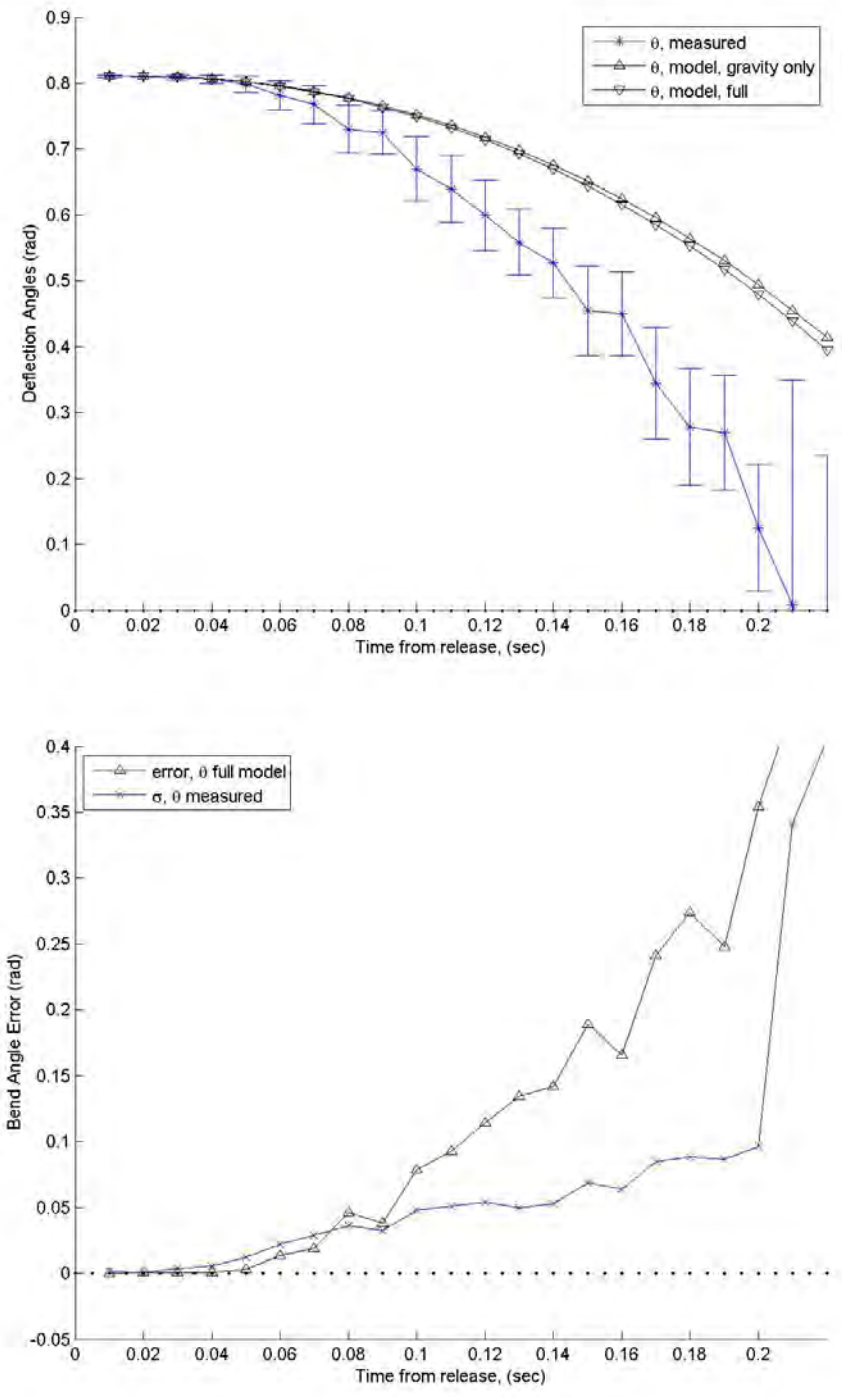

Figure 4.3: Block 2, Set 1, 32 Trials, Tape spring 4x1, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

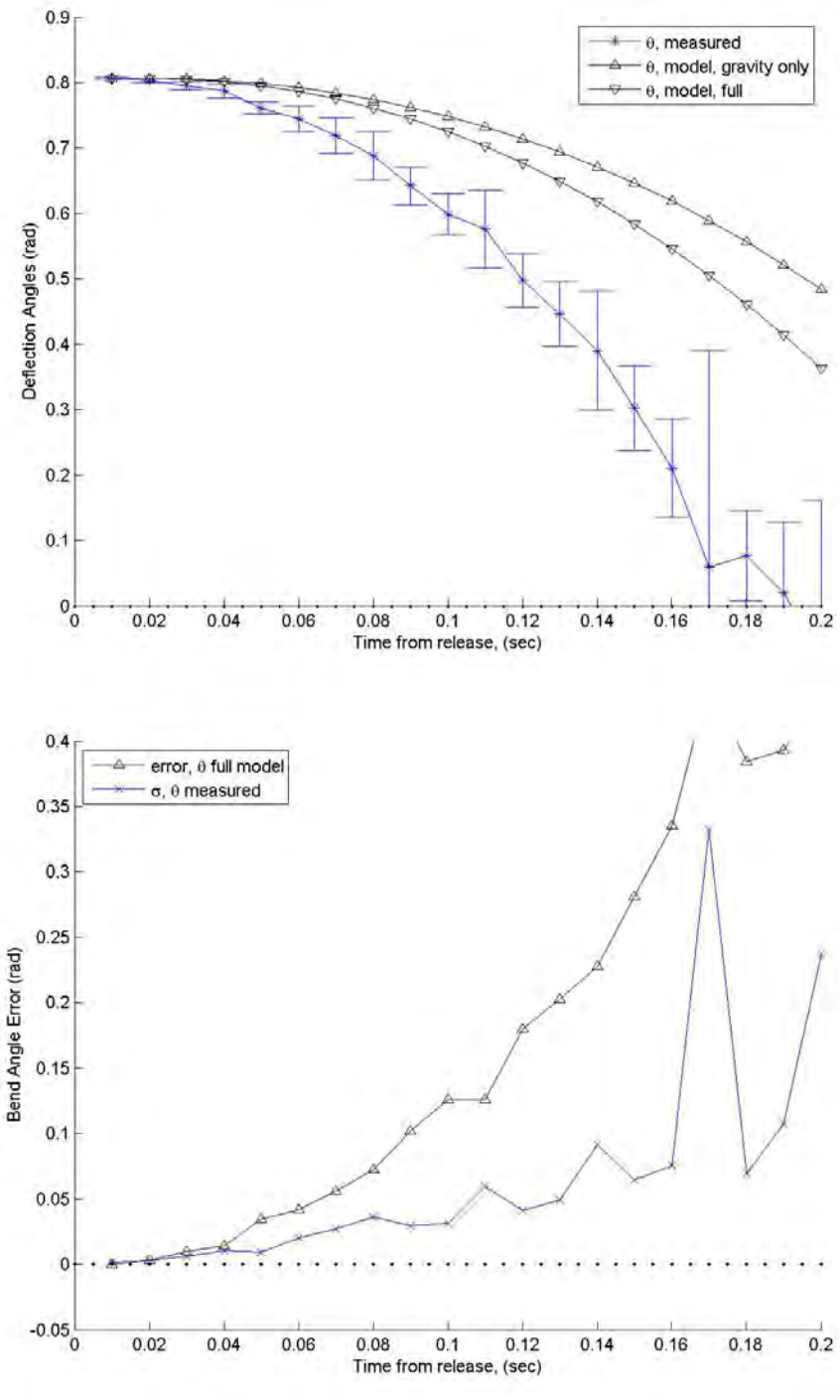

Figure 4.4: Block 2, Set 2, 27 Trials, Tape spring 4x2, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

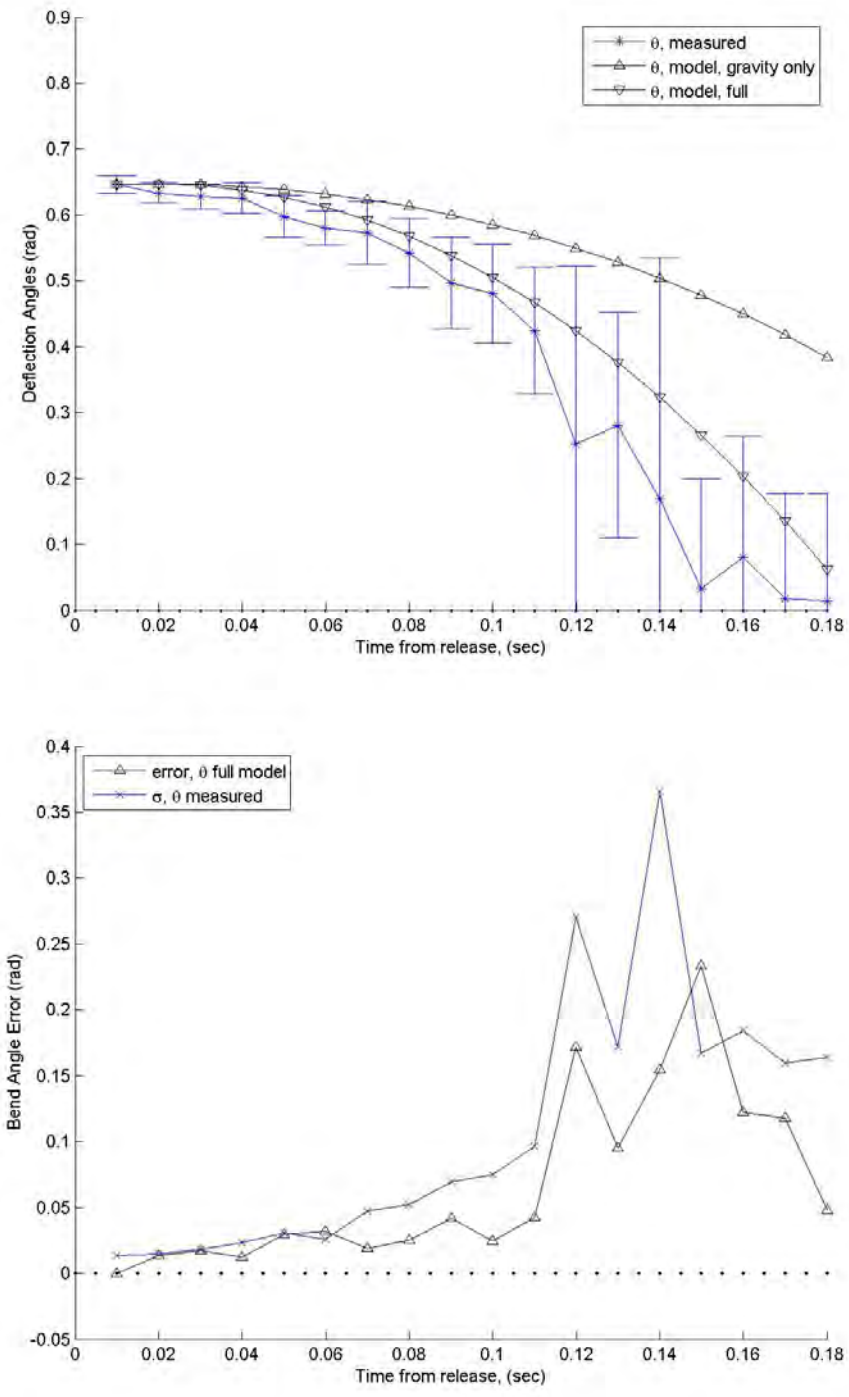

Figure 4.5: Block 2, Set 3, 29 Trials, Tape spring 4x3, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

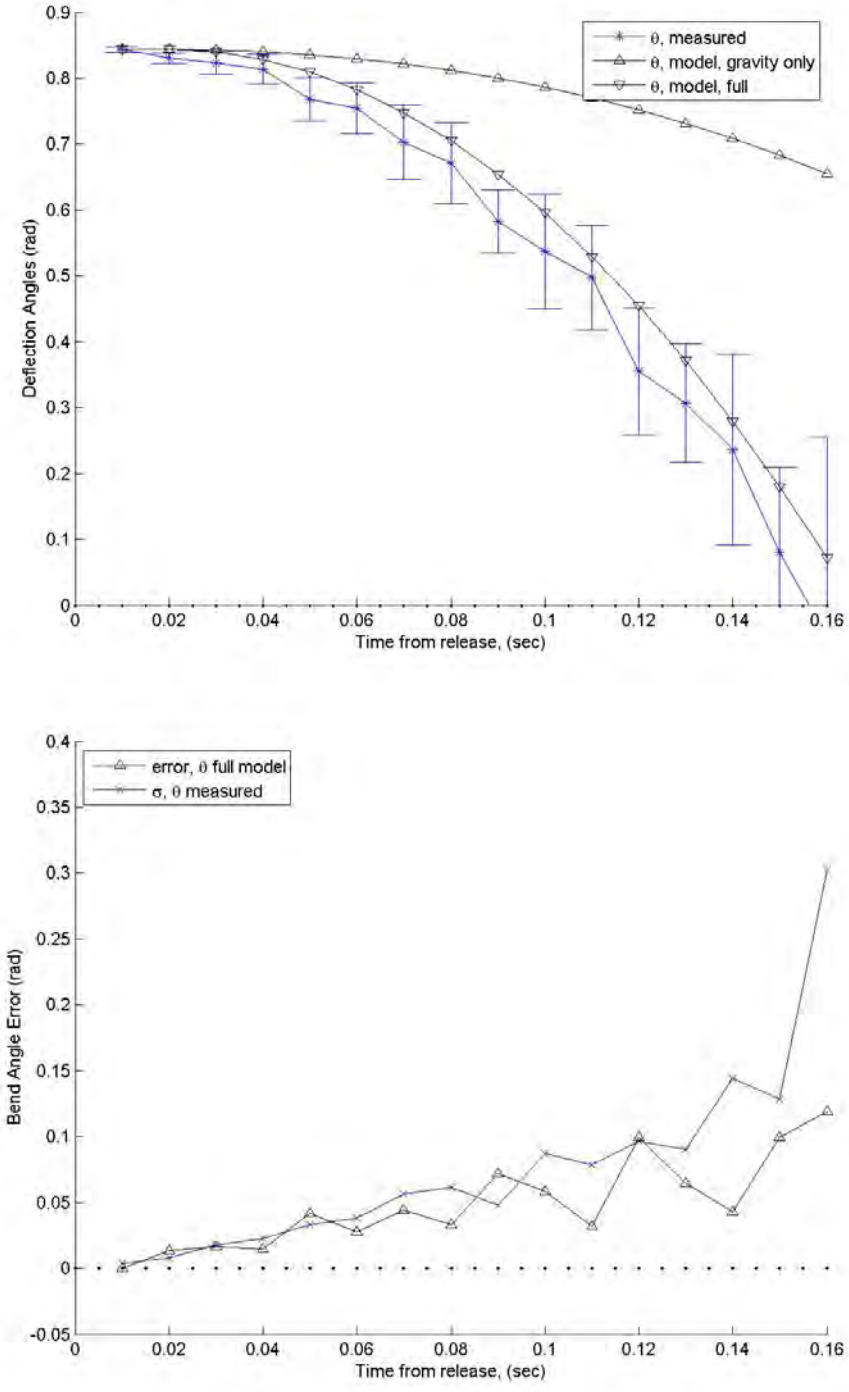

Figure 4.6: Block 2, Set 4, 27 Trials, Tape spring 4x4, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

4.3 Loading Variations

The third block of trials investigated the effect of loading on the tape spring hinge. Figures 4.7 through 4.10 display these data sets in order of increasing skew angle. As expected, the hinge moment does not depend on the load, and the faster response is still the simple product of torque divided by inertia (τ/I_{yy}). Furthermore, this block of trials shows better agreement than any other set. The low errors on these sets indicate that the model does not need to be modified for loading, and that the error that do exist may be attributed to variance in release time.

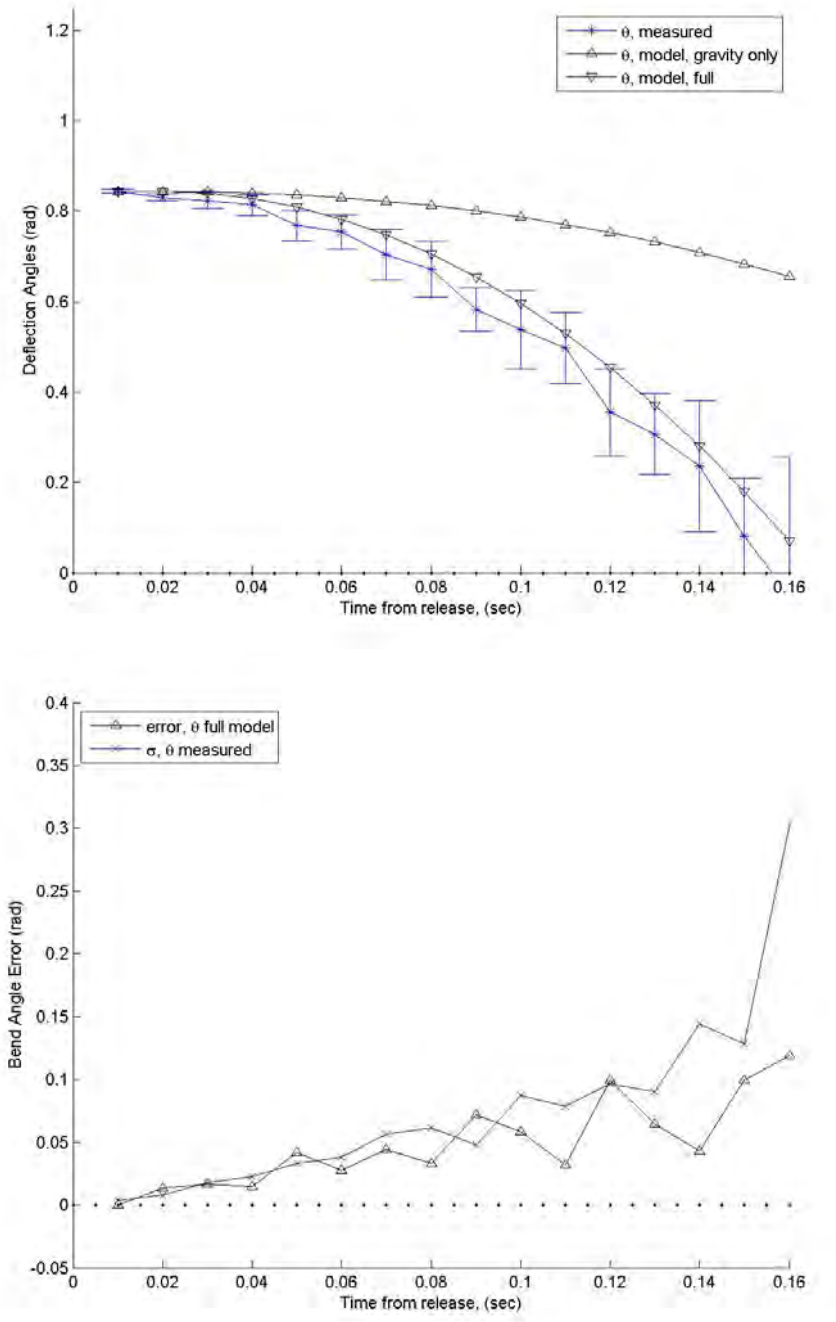

Figure 4.7: Block 3, Set 1, 27 Trials, Pipe Only, Top: Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

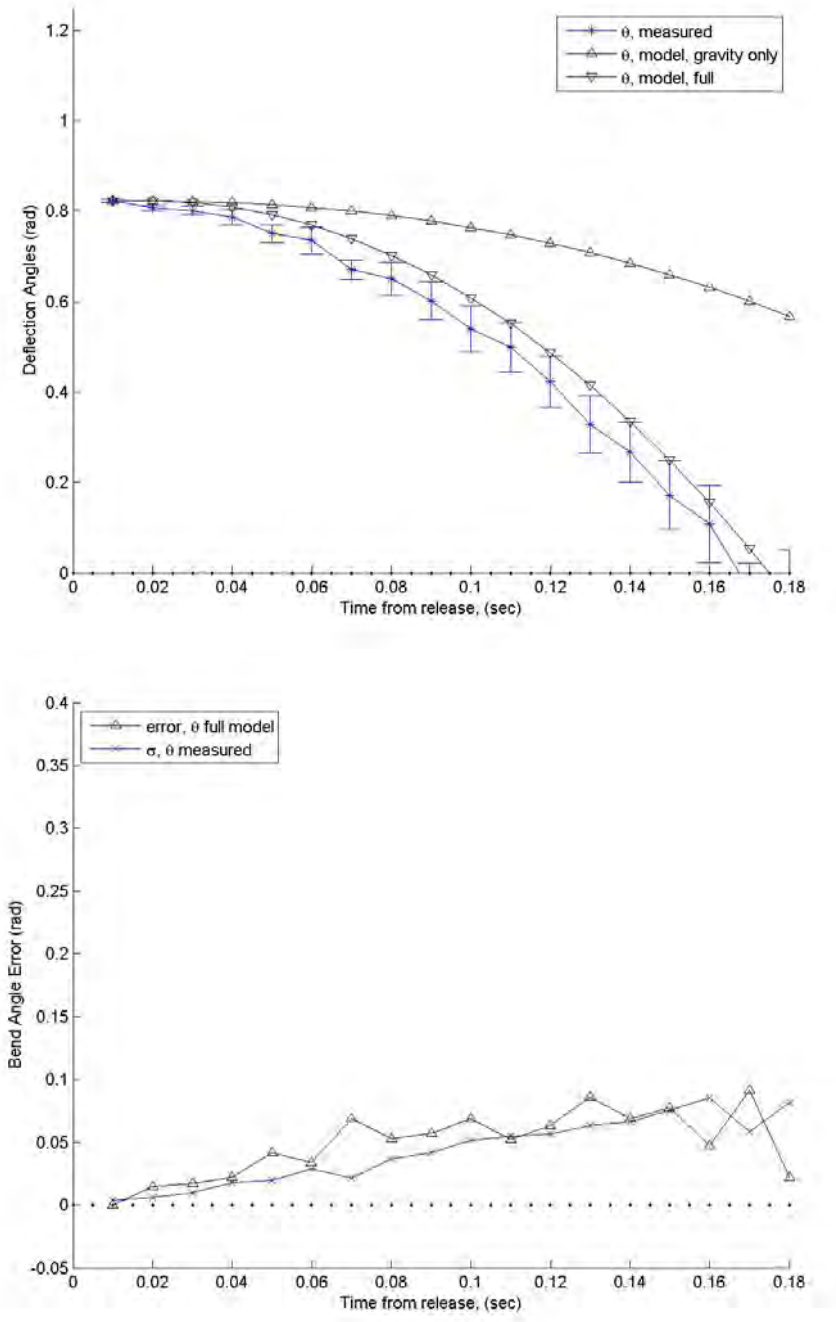

Figure 4.8: Block 3, Set 2, 15 Trials, Pipe +19g @12", Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

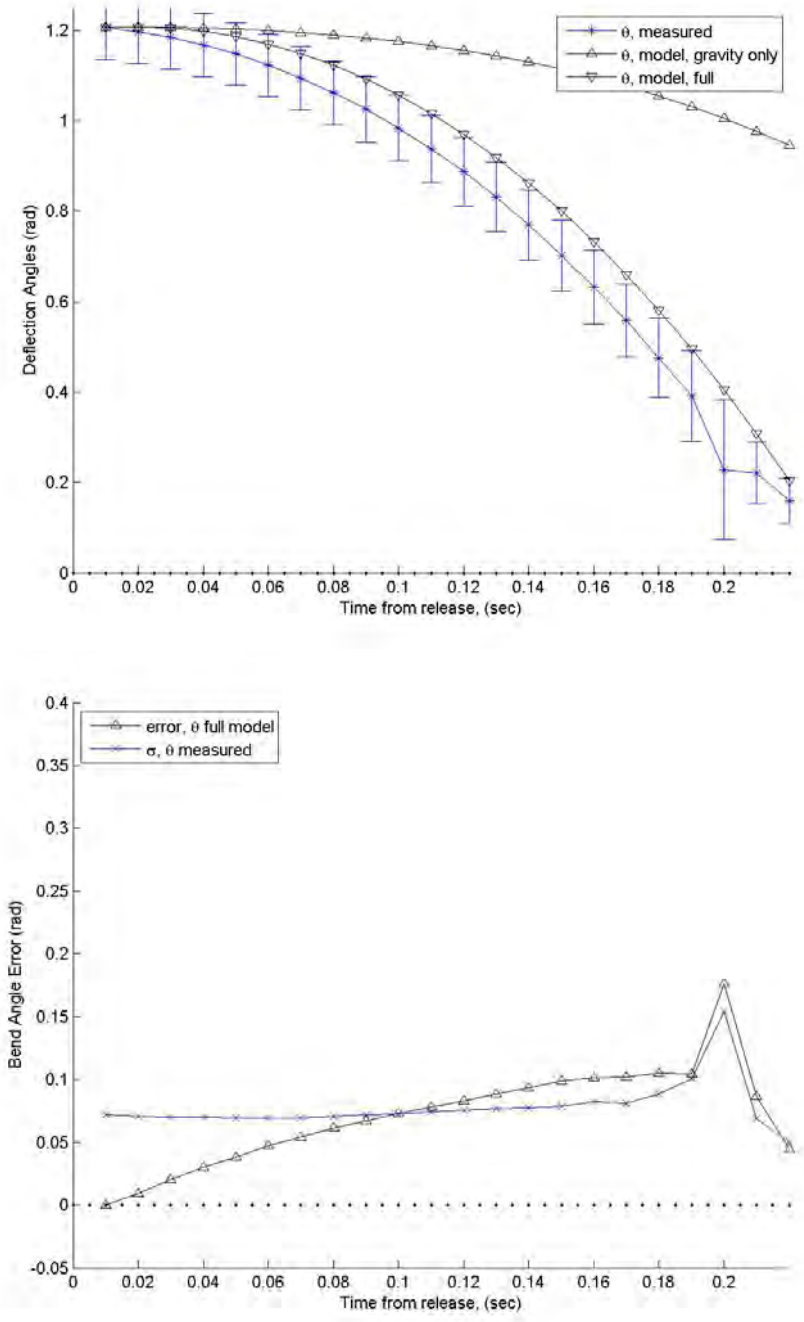

Figure 4.9: Block 3, Set 3, 12 Trials, Pipe +51g @12", Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

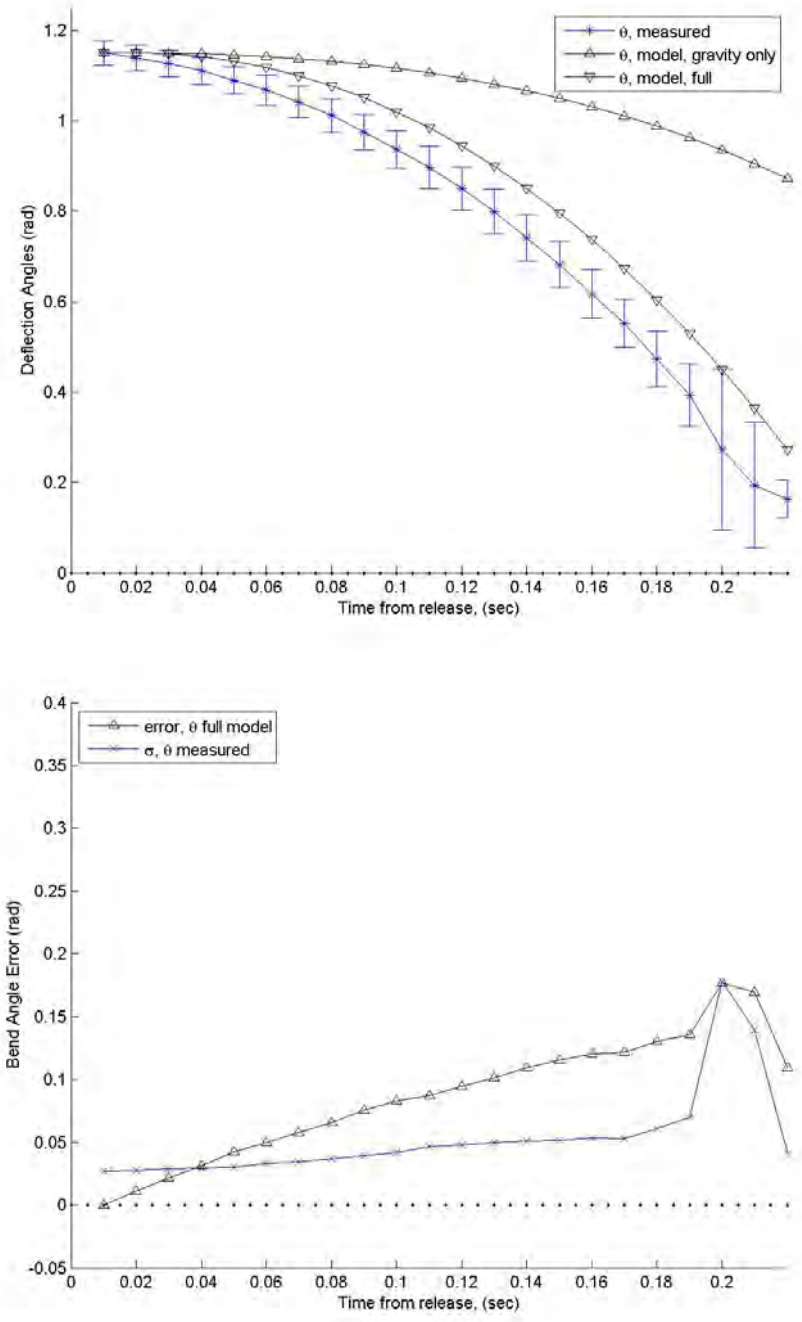

Figure 4.10: Block 3, Set 4, 10 Trials, Pipe +83g @12", Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

4.4 Skew Angle Variations

The last block of trials varied the skew angle of deformation. Each set is shown in Figures 4.11 through 4.14, in order of increasing skew. Compared to the previous blocks, these trials exhibit good fit except for the 45° case. Although the errors ($|\theta_{model,full} - \theta_{measured}$—) for Set four are unusually high at 0.5 rad, the large variance of the measured data cautions against any strong conclusions based on this set. In contrast, the first three sets display very small errors all less than 0.3 rad, and display no new (statistically significant) error patterns. Although there is a slight correlation of skew angle and increasing absolute error from 0.1 to 0.15 to 0.2 to 0.3 rad, the trials simply have too much noise to conclude that this pattern is due to any underlying mechanics rather than an artifact data noise. Despite heavy noise, the visible trajectories do match the set's mean path well, a fact that suggests the existing model is a sufficiently accurate prediction (at least for $\mu \leq 45°$). In addition, The lack of skew angle μ dependence also agrees with the one-dimensional Seffen-Pellegrino model. However, the small range of μ present in these experiments, combined with the lack of skew angle response suggests that more data is necessary to conclusively demonstrate skew angle dependence— either affirmatively or negatively. Further data of most use would be at higher skew angles and lower variance. However, to test for any presence of skew angle in this thesis, no matter how slight, a correction may be added to the hinge torque model of the form $c\mu$. As an aside, because skew angles are so low, $\sin(\mu) \approx \mu$, and the two terms will be indistinguishable.

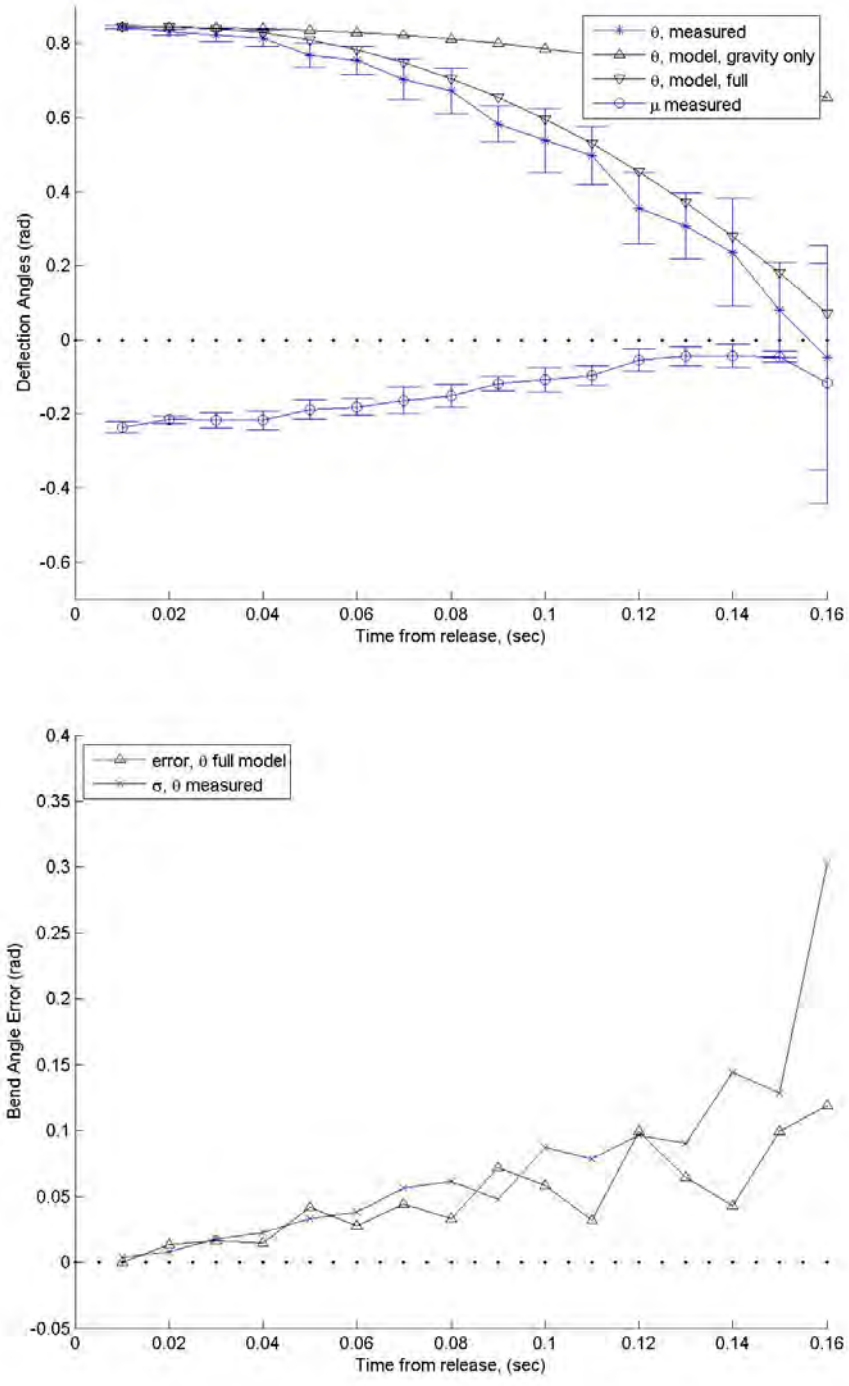

Figure 4.11: Block 4, Set 1, 27 Count , 0° Skew, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

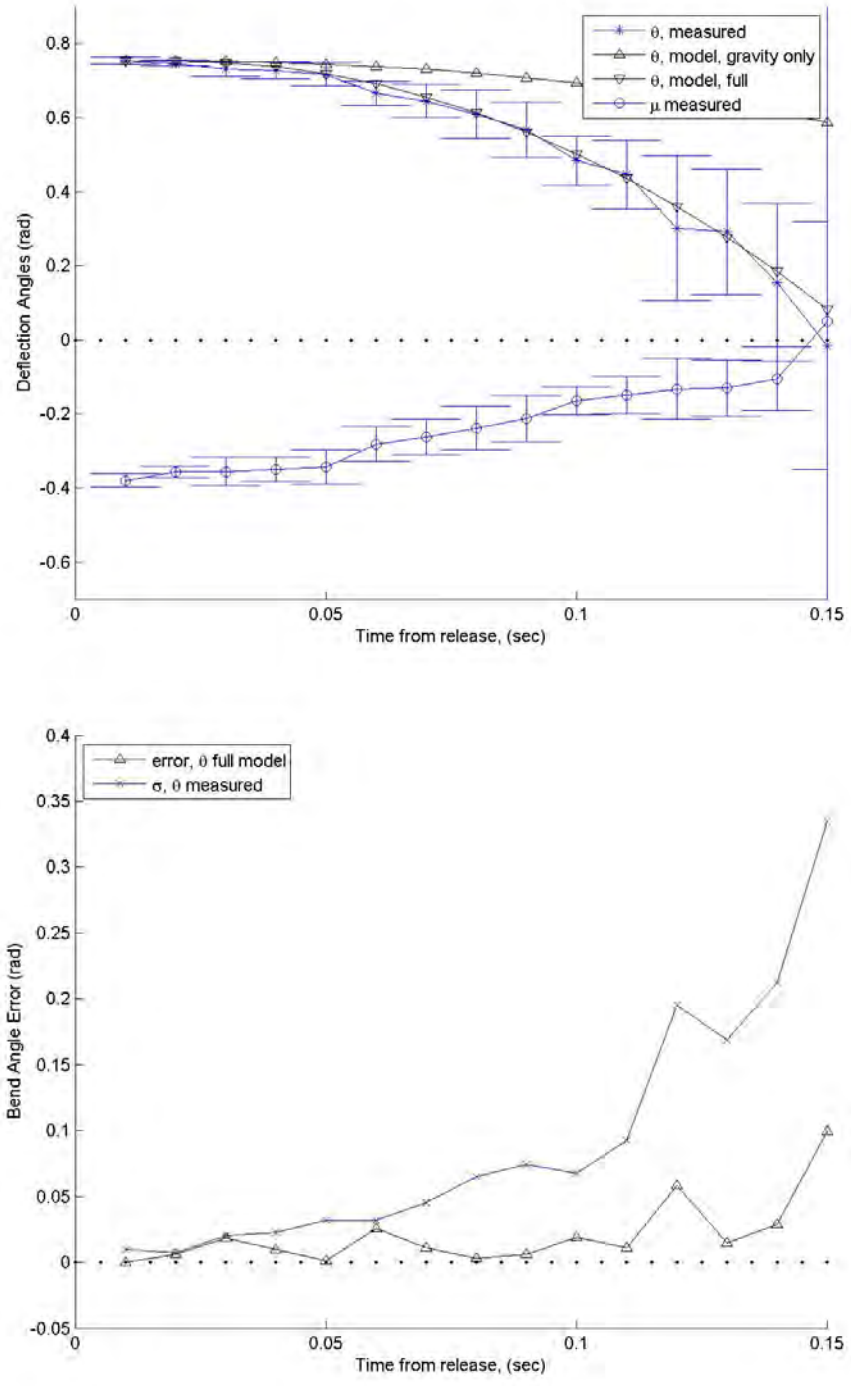

Figure 4.12: Block 4, Set 2, 47 Count, 15° Skew, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

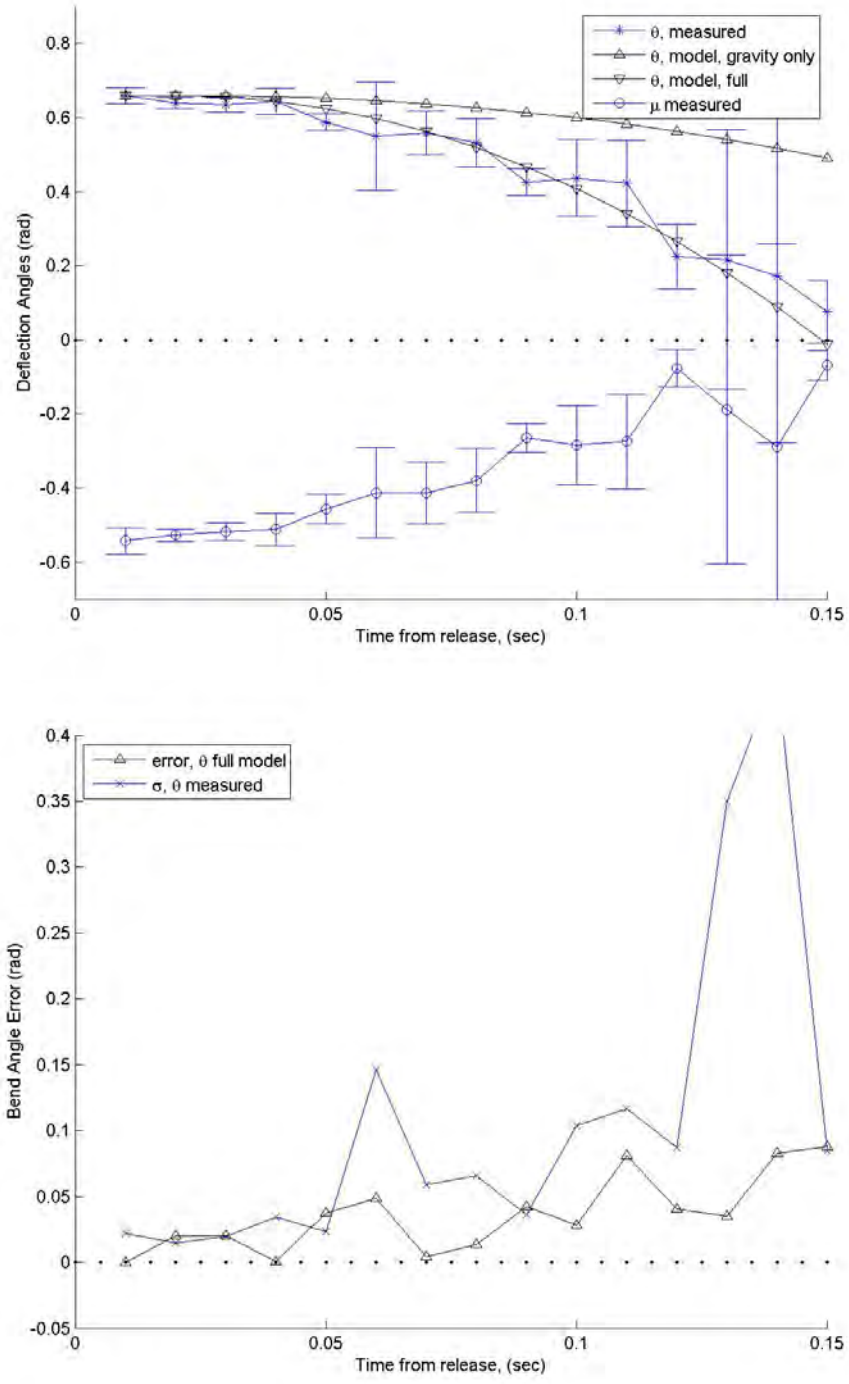

Figure 4.13: Block 4, Set 3, 39 Count, 30° Skew, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

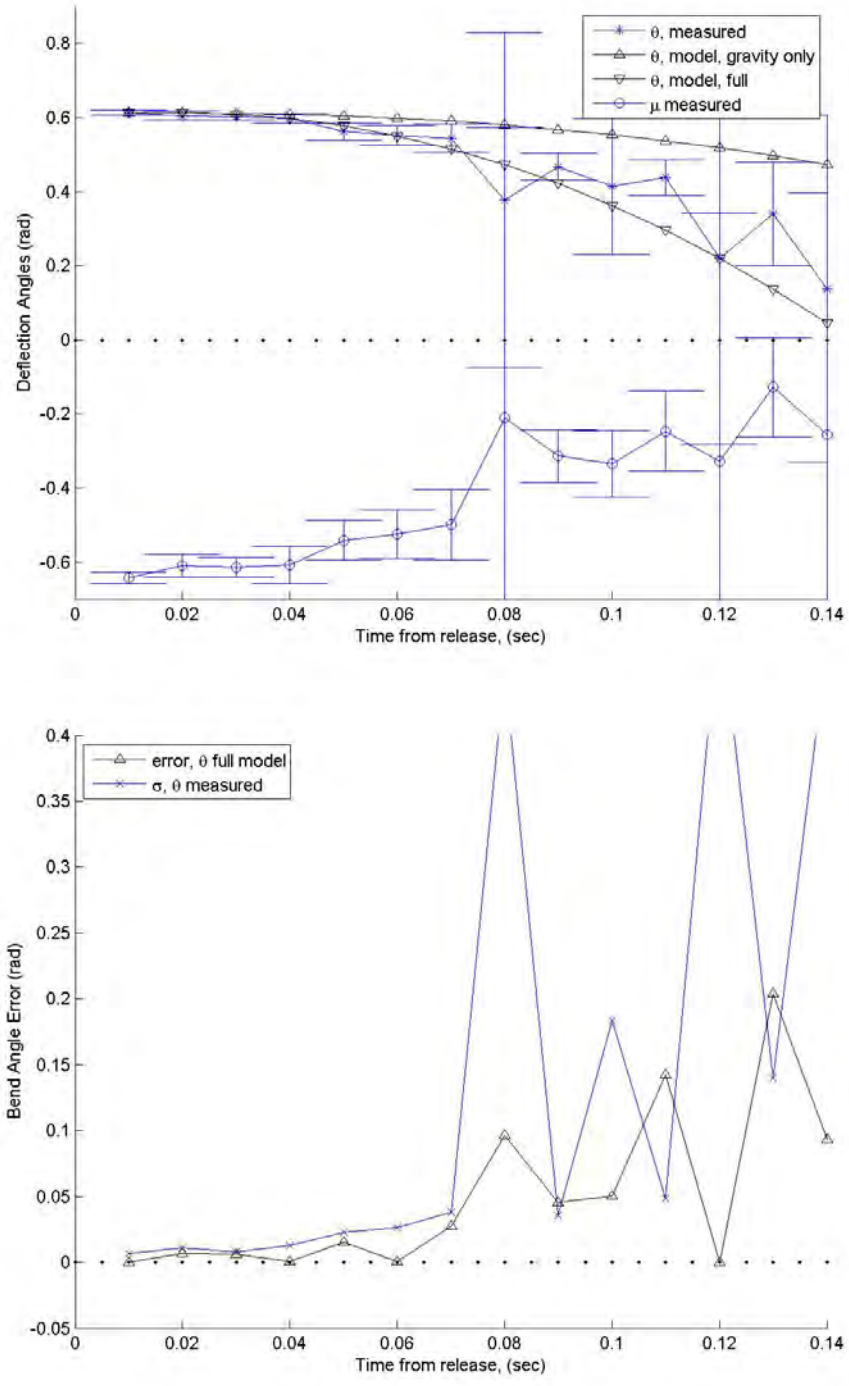

Figure 4.14: Block 4, Set 4, 31 Count, 45° Skew, Top: Deflection angles, both measured and model-predicted. Bottom: Bend angle θ error and data standard deviation.

4.5 Proposed Optimal Model

4.5.1 Model Formulation, Initial Version. From the analysis above, several changes to the existing model are proposed. This model is a formulation of the hinge reaction torque $\tau_{hinge,opt}$, shown in Equations (4.1),(4.2), which would replace Equation (3.14). In these equations, the existing variables ν, E, θ, μ, t, α and a all retain their meanings from above, while $[c_0, c_1, \ldots c_8]$ are constants to be determined by the optimization algorithm. (For convenience, the c_i constants may be collected into a vector $\vec{c} = [c_0, c_1, \ldots c_i, \ldots c_8]$.) The optimization algorithm will choose the $\vec{c}$ to minimize the error between this model and the measured data. After converging, the optimization algorithm will return a $\vec{c}$ which corresponds to the $\tau_{hinge,opt}$ model shape with the smallest error. In this proposed model, K_{opt} accomplishes a similar purpose to Flexural Rigidity, D, but takes into account the additional geometry parameters of arc width a.

$$\tau_{hinge,opt} = (1-\nu)K_{opt}\alpha(c_0 + c_1\theta + c_2 \sin\mu) \tag{4.1}$$

$$K_{opt} = c_3 t + c_4 t^2 + c_5 t^3 + c_6 t^4 + c_7 a + c_8 t^3 a \tag{4.2}$$

As this a first attempt at modeling such an unrestricted problem, the above guess at a model will not be completely correct. However, this first model will be instructive as to which terms are most important. Because larger coefficients are more important to the model than smaller ones, the relative magnitudes of the final coefficents will indicate which terms were correctly guessed to be critical, and which were not (and thus may be safely neglected). Subsequent models then, may cull unimportant terms to produce a minimum complexity model with no loss in accuracy.

Indeed, some of the above terms in (Equation 4.1) are expect to result in small c_i values, but are included to test the hypothesis. For instance, c_6 is expected to

be negligable because the Seffen-Pellegrino Model is already affected too much by thickness in the results above.

4.5.2 Modeling Procedure. To produce the desired $\vec{c}$ coefficients, the model in Equation (4.1) was added to the analysis code, then provided to the optimization toolbox in MATLAB. The toolbox contains a method 'nlinfit' which applies a Levenberg-Marquardt Least-Squares Regression Algorithm (LMA) to optimize a supplied model, vector of parameters and target data. The model is Equation (4.1) and (4.2) above. The parameters are the c_1-c_9 coefficients, measured θ profiles, measured μ profiles, tape spring properties, and other physical constants. Lastly, the target data is the measured θ profile over the time interval from release (t_r) to t_f when $\theta = 0$. Inside the algorithm, the "best" fit is found by minimizing the least-squares error J, calculated by a least-squares comparison of the model and target data (here, the measured experimental data). In this case, J is an error measure of the predicted θ profile compared to the measured θ profile, shown in Equation (4.3) below. J is identically equal to the sum of the squares of the residuals, so when $J = 0$ the model fits perfectly, and when J is very large, the model fits poorly. Note that this formulation is not linear with respect to either time or $\vec{c}$.

$$J = \sum_{t=t0}^{t_f} \left(\theta_{measured} - \theta_{model,new}\right)^2 \tag{4.3}$$

4.5.3 Model Formulation Changes. Due to limitations in the optimization algorithm, the initial model formulation in Equations (4.1),(4.2) was numerically unstable, and did not generate a solution. Therefore, a new model was recreated from Equation (3.12). This model still generates an expression for $\tau_{hinge,opt}$, and is shown below in Equation (4.4). (Note: Although Equation (4.4) and Equation (4.1) may be shown equivalent by an appropriate choice of the $\vec{c}$ values, the equivalence is neither interesting nor relevant to the problem.) This new version of the model in Equation (4.4). The biggest difference is the new model is now linear with respect

to $\vec{c}$. Linearity allows the variations to be independent and allows the entire model to be computationally less complex. Another difference is that Flexural Rigidity D dominates the coefficients, and accounts for several orders of magnitude. This effectively camoflages differences that the model was designed to show. So, D is multiplied by every coefficient, and the remaining terms are instead judged by their effect on J, the smallest error a given model allows.

This new model is shown below in Equation (4.4). It is now linear, and therefore numerically stable. The optimization algorithm produces the $\vec{c}$ shown in Table 4.1. The value row is the simple values of each coefficient output by 'nlinfit', while the sensitivities are more complicated. The sensitivity was found by removing each c_i term from the model, performing an optimization run, then recording the difference of the new error J and the old error as the sensitivity for c_i. This means that the sensitivity is a lower bound of each coefficient's importance to the model.

Overall, this optimized model is generally much closer (in a least-squares sense) to the $\theta_{measured}$ path than the Seffen-Pellegrino model. To demonstrate the improvement of the optimized model, a representative data set is shown in Figure 4.15. Although useful to verify model behavior, the error function J is the primary measure of quality for any particular model version. The result of optimizing Equation (4.4) is shown in Figure 4.15 and produced an error of $J = 0.2113$.

$$\tau_{hinge,opt} = -(1-\nu)D\left(c_1 + c_2\alpha + c_3\theta + c_4\mu + c_5 a + \frac{c_6 a}{t^3} + \frac{c_7}{t} + \frac{c_8}{t^2} + \frac{c_9}{t^3}\right) \quad (4.4)$$

Table 4.1: Optimized Model Results. These values are the result of optimizing Equation (4.4) against all measured data sets. The resultant error was $J = 0.2113$

Coefficient	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
Value	-13.3	225.1	0.27	0.25	-2.49	0.38	-0.85	0.26	-12.26
Sensitivity	0.00	0.00	0.2146	0.0604	0.00	0.00	0.0101	0.0168	0.00

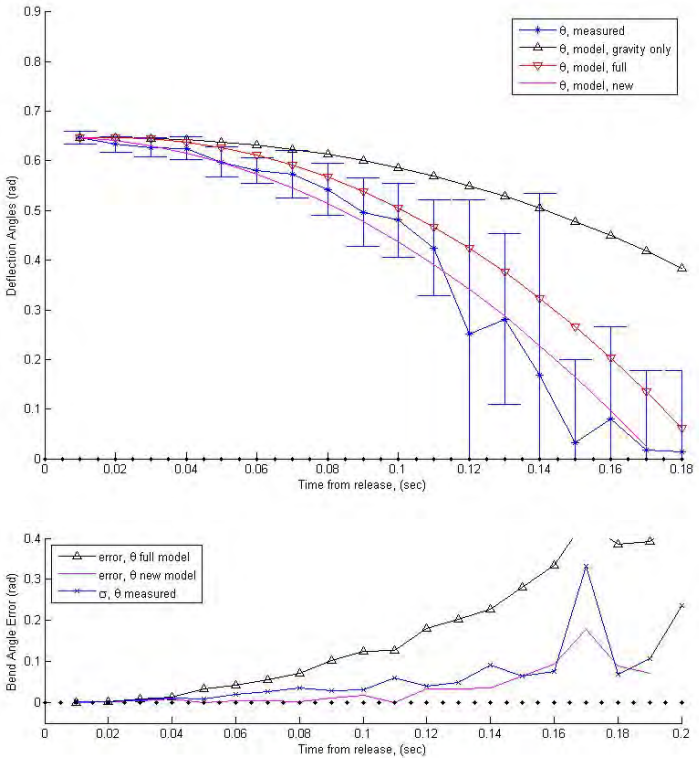

Figure 4.15: Block 2, Set 3. Top: Deflection angles, both measured and model-predicted. This plot also includes the optimized model generated by Equation (4.4). Bottom: Bend angle θ error and data standard deviation.

4.5.4 Model Reduction. The next step is to eliminate unnecessary terms from Equation (4.4) to produce the reduced form Equation (4.5). The process for this was a series of successive trial and error tests. Every term was tested by removal, optimization, then comparing the growth in error. If the resultant error did not grow, then that term was redundant. This process was repeated until no term in the reduced model was redundant. (Note: because of coupling, removal of a single term may be compensated by growth in other terms. This causes the error function to converge to a different equally optimal point in the solution space.) By inspection, it may be seen that the terms eliminated were literally redundant in that they could be expressed by the c_5 term remaining in the function. In contrast, the c_1 and c_2 terms were individually redundant, but became necessary when the c_5 and c_9 terms were removed from Equation (4.4). Unfortunately then, this reduction process did not highlight any physical shortcomings in the model. Furthermore, Equation (4.5) will still describe any bias or other systematic error present in the data.

$$\tau_{hinge,opt} = -(1-\nu)D\left(c_1 + c_2\alpha + c_3\theta + c_4\mu + \frac{c_6 a}{t^3} + \frac{c_7}{t} + \frac{c_8}{t^2}\right) \tag{4.5}$$

Table 4.2: Optimized Model Results. These values are the result of reducing Equation (4.5), and then optimizing against all data sets. The resultant error was still $J = 0.2113$

Coefficient	c_1	c_2	c_3	c_4	c_5	c_6	c_7
Value	1.5064	-1.5048	0.2796	0.2596	-0.0004	-0.8505	0.2626
Sensitivity	0.0181	0.1083	0.2146	0.0604	0.0080	0.0102	0.0168

4.5.5 Analysis of Optimized Model Results. Because the terms in the above model are not normalized, the coefficients hold no special significance. In fact, in this form the coefficients may be unduly sensitive to variations in hinge geometry or deformation angle because the coefficient multiplies a larger number. Nevertheless, these values are provided for any future comparison with these results. In contrast, the resultant sensitivities in Figure 4.2 above are, in a mathematical sense, a rating of

relative importance among each term. Not surprisingly then, the largest individual sensitivity is in response to the $c_3\theta$ term. This indicates that the physical tape spring response is indeed dependant upon bend angle, in disagreement with the Seffen-Pellegrino model. Because the error has a non-trivial sensitivity to this term, future models are suggested to include a term dependent on θ. For the same reason, future models should have a μ although it is likely to be smaller.

Unfortunately, the remaining terms have several issues which cloud similar interpretations. Although tape spring geometry clearly affects the hinge torque response, this thesis only contains two independent data points: tape spring number three, and tape spring number four. Even worse, both tape springs have very similar geometry and provide only a very small baseline to extrapolate towards larger or smaller geometries. The two tape springs' thickness and subtended angle both differed only by about 20%; whereas the values of θ and μ vary from about 0 to 1. The small number of tape springs, the similiarity between the tested tape springs, and geometry measurement error suggest that the results of the c_1, c_2, c_5, c_6, and c_7 terms should not be extrapolated to tape spring behavior in general without a large range of data to fit them.

Yet another caution is that neither the $\frac{aD}{t}$, $\frac{D}{t^2}$, nor $\frac{D}{t^3}$ terms were removed during the model reduction. After canceling out the t^3 contained within Flexural Rigidity, those same three terms become t^2, t and 1. The data from Block 2 (Section 4.2 above) does indicate errors with respect to thickness. However, neither Plate theory nor Beam theory predict moments that depend on t^2, t and 1. The presence of all three terms in the empirical model suggest that the optimized model does not match existing plate theory, and therefore that multiple thinner tape spring are not equivalent to a thicker monolithic tape spring.

4.5.6 Curve Fitting Conclusion. Applying an optimized model to the data sets has been productive. Although the model fits this particular data set well,

its small baselines lead to coefficients which are unlikely to extrapolate in general. The model and coefficient dependence upon thickness will be particularly inaccurate, because this thesis did not include any monolithic tape springs. Nevertheless, the optimized models do indicate affects of both bend angle θ and skew angle μ, and imply that future models should include these angles.

V. Conclusions and Future Work

5.1 Conclusions

The four different variables investigated here produced three significant variations where the existing model lacks accuracy, while highlighting a parameter where the model is already sufficiently accurate, and one paramater which is inconclusive. The areas of improvement are moment dependence on bend angle θ, on tape spring shape, and on tape spring thickness. The areas where the model is already accurate are moment dependence on load. The proposed changes changes were then implemented in an optimized curve fit model, which indicated a strong dependence on tape spring bend angle (θ) and skew angle μ. However, the optimized curve-fit model was inconclusive about the remaining parameters: width a and subtended angle (α).

Compared to the experimental data gathered, the Seffen-Pellegrino model is an accurate first order approximation, but may not be close enough to model tape spring movement in microgravity with acceptable accuracy. In addition, the current photogrammetry system used to conduct these experiments shows significant drawbacks when measuring fast, flexible structures. However given the ability of existing and proven tape spring designs, the potential for increased mission ability is clear. Better models and predictions of tape spring behavior would enable lighter and more complex structures. These structures in turn could accomplish broader missions with greater reliability for less cost.

5.2 Recommendations for Future Work

5.2.1 Near Term Recomendations. The biggest issue with this work is the error sources. The measured data contained too many sources whose errors are too large to ignore. Some of the model features may even be a result entirely of these noise sources. Fortunately, the data points to several probable causes. The first is inaccuracies of VICON measurement, and dropped frames. The cause is not clear, but it manifests in dropping 50-70% of frames in later trials– even though the

first trial or two of a run are frame complete. Along these lines, an experimental apparatus which produced consistent deformation of given conditions would improve irregularities and other noise in the data.

Once the measured data is satisfactory, more accurate measurements of the mass, tape springs, moment of inertia of the free end pipe, and additional weights would further increase the accuracy in post-processing.

Another possible source of error is in the data processing. Because of the difficulty of picking the instant at which release occurs, the error in initial conditions also contributes to the remaining path error.

A wider variation of design parameters, and a larger number of tapesprings would improve the baseline of the optimized model. This improved baseline would increase the accuracy of its predictions to large design envelopes.

There are of course several tests to validate the assumptions the experiments start with. One option is to obtain a tape spring of the same geometry as the composite steel tape springs (i.e. instead of four #4 tape springs combined, one tape spring with quadruple the thickness, but same subtended angle, curvature, arc length, and chord length). This would provide a tapespring that matches the model, and will provide more accurate and appropriate data. A second option would be to obtain a stiffer tape spring with enough force to run deployments perpendicular to, or against gravity, instead of with gravity. This would provide more comparisons of how much gravity affects deployment dynamics.

5.2.2 Medium Term. Once the analysis is free of spurious noise, the problem remains that the tape spring dynamics are inherently noisy, and so the model must incorporate this. Something as simple as Gaussian noise added to the measured angles may suffice to mimic the observed variations.

Another option is to change the VICON setup. Although the free end pipe adds a useful amount of inertia, the weight and momentum it imparts prevents testing

of lighter tape springs, particular the size and thickness appropriate for actual use on satellites. A lighter-weight mount for the reflectors would enable testing of this category of tape springs.

Yet another possibility is measurement by a higher frequency (or higher accuracy) measurement method. Along with improving the flexure methods of the tape spring, these offer distinct possibilities for reducing error, and obtaining a much more accurate model than the experimental hardware used in this paper could achieve.

5.2.3 Far Term. The most valuable avenue will likely be with more complex structures. Constructing, modeling and measuring multi-joint deployment dynamics such as [26] has the potential to unlock much larger scale missions. This paper has shown that joints behave independently, suggesting the models for such structures will be relatively simple, and accurate enough to support flight development.

Appendix A. Equipment Properties

Table A.1: Tape Spring Properties

Tape Spring # (Steel)	1	2	3	4	5	6
Thickness (mm)	0.09	0.10	0.10	0.11	0.08	0.09
Angle (°)	16.0	21.5	29.6	23.6	14.1	25.8
Curvature (mm^{-1})	0.022	0.020	0.021	0.013	0.010	0.018
Radius (mm)	44.6	50.0	48.4	76.6	100.0	56.1
Arc Width (mm)	12.44	18.78	25.03	31.63	24.66	25.27
Chord Width (mm)	12.02	17.39	22.15	28.88	22.40	22.60
Length (mm)	203.2	203.2	203.2	203.2	215.9	159.0
Mass (g)	1.8	2.8	4.3	7.0	4.5	3.8
Material	Steel	Steel	Steel	Steel	Steel	Steel
Density (g/mm)	0.009	0.014	0.021	0.034	0.021	0.024
Flexural Rigidity (D)	17.11	23.47	23.47	31.24	12.02	17.11

Tape Spring #(CFRP)	7	8	9	10	11	12
Thickness (mm)	0.090	0.210	0.130	0.130	0.280	0.420
Angle (°)	19.18	66.14	55.06	50.11	71.02	72.35
Curvature (mm^{-1})	0.024	0.057	0.049	0.044	0.054	0.014
Arc Width (mm)	14.126	20.310	19.650	20.100	23.130	89.600
Chord Width (mm)	18.860	13.590	14.080	15.070	14.880	56.500
Length (mm)	159.00	460.40	279.40	279.40	495.30	NaN
Mass (g)	1.000	3.300	1.300	1.400	5.000	25.000
Material	CFRP	CFRP	CFRP	CFRP	CFRP	CFRP
Density (g/mm)	0.006	0.007	0.005	0.005	0.010	NaN

Table A.2: PVC Pipe Properties

Pipe # (PVC)	1	2	3	4
Mass (g)	82.4	61.0	97.1	179.0
Length (mm)	223.8	254.0	223.8	279.4
Inner Diameter (mm)	15.71	15.71	20.03	34.62
Outer Diameter (mm)	21.38	21.38	26.75	42.08
Thickness (mm)	2.87	2.87	3.31	3.69
Linear Density (g/mm)	0.37	0.24	0.43	0.64
Volumetric Density (g/mm^2)	0.00223	0.00145	0.00176	0.00143
Moment of Inertia (g/mm^2)	7.82e+006	7.09e+006	9.15e+006	2.34e+007

```
% this file contains the measurements of the tapesprings, and
% generates the LaTeX tables (directly pastable) for the paper.

clc
clear

% will print n column separators, and a newline (LaTeX syntax)
% second argument 'i' is ignored (matlab is picky...)
printncol=@(n,i) eval('for i=1:n; fprintf('' &''); end; fprintf('' \\\\
    \n'');');

%% Tapespring Physical Properties
% ts == tapespring
ts_ct_steel =6;
ts_ct_cfrp = 6;
ts_dx_cfrp = (ts_ct_steel+1):(ts_ct_steel+ts_ct_cfrp);
ts_type_count = ts_ct_steel + ts_ct_cfrp;
ts(ts_type_count).depth=0; % partial pre-allocation

ts(1).chord=12.02;
ts(1).depth=1.65;
ts(1).arc_length=12.44;
ts(1).thick_total=0.13;
ts(1).thick = 0.09;
ts(1).length= 203.2; %mm
ts(1).mass = 1.8; %g
ts(1).material='Steel';
ts(1).descr='';
% 205 GPa =2.05 e11 Pa = 2.05 e11 N/m^2 = 2.05 e11 g/mm
ts(1).E = 2.05e11;  % g/mm^2
ts(1).nu = 0.290;

ts(2).chord=17.39;
ts(2).depth=3.12;
```

```
ts(2).arc_length=18.78;
ts(2).thick_total=0.11;
ts(2).thick = 0.1; % dummy variable
ts(2).length= 203.2; %mm
ts(2).mass = 2.8; %g
ts(2).material='Steel';
ts(2).descr='';
ts(2).E = 2e11; % N/mm^2
ts(2).nu = 0.290;

ts(3).chord=22.15;
ts(3).depth=5.36;
ts(3).arc_length=25.03;
ts(3).thick_total=0.14;
ts(3).thick = 0.10;
ts(3).length= 203.2; %mm
ts(3).mass = 4.3; %g
ts(3).material='Steel';
ts(3).descr='';
ts(3).E = 2.05e11; % N/mm^2
ts(3).nu = 0.290;

ts(4).chord=28.88;
ts(4).depth=5.65;
ts(4).arc_length=31.63;
ts(4).thick_total=0.165;
ts(4).thick = .11;
ts(4).length= 203.2; %mm
ts(4).mass = 7.0; %g
ts(4).material='Steel';
ts(4).descr='';
ts(4).E = 2.05e11; % N/mm^2
ts(4).nu = 0.290;
```

```
ts(5).chord=22.4;
ts(5).depth=2.54;
ts(5).arc_length=24.66;
ts(5).thick_total=.14;
ts(5).thick = .08;
ts(5).length= 215.9; %mm
ts(5).mass = 4.5; %g
ts(5).material='Steel';
ts(5).descr='';
ts(5).E = 2e11; % N/mm^2
ts(5).nu = 0.290;


ts(6).chord=22.6;
ts(6).depth=4.75;
ts(6).arc_length=25.27;
ts(6).thick_total=.14;
ts(6).thick = .09;
ts(6).length= 159; %mm
ts(6).mass = 3.8; %g
ts(6).material='Steel';
ts(6).descr='';
ts(6).E = 2e11; % N/mm^2
ts(6).nu = 0.290;

ts(7).chord=18.860; %mm
ts(7).depth=4.4496;
ts(7).arc_length=14.1260;
ts(7).thick_total=.14;
ts(7).thick = .09;
ts(7).length= 159; %mm
ts(7).mass = 1.0; %g
ts(7).material='CFRP';
ts(7).descr='1-ply ? glass [\\pm45 deg]';
ts(7).descr='';
```

```
ts(7).E = 2e11; % N/mm^2
ts(7).nu = 0.290;

% CFRP #: 8,9,10,11
% 5/8" ==  15.875 mm
lw=8; sw=9; sb=10; lb=11;

ts(lw).chord=13.59; %mm
ts(lw).depth=6.42;
ts(lw).arc_length=20.31;
ts(lw).thick_total=0.21;
ts(lw).thick = 0.21;
ts(lw).length= 460.4; %mm, 18 1/8"
ts(lw).mass = 3.3; %g
ts(lw).material='CFRP';
ts(lw).descr=' 2-ply, S-2 UT, T300 PW with 8552 matrix: [0 UT, \\pm 45
    PW]';
ts(lw).E = 2e5; % N/mm^2
ts(lw).nu = 0.290;

ts(sw).chord=14.08; %mm
ts(sw).depth=5.62;
ts(sw).arc_length=19.65;
ts(sw).thick_total=0.13;
ts(sw).thick = 0.13;
ts(sw).length= 279.4; %mm, 11"
ts(sw).mass = 1.3; %g
ts(sw).material='CFRP';
ts(sw).descr=' 1-ply, S-2 glass with 8552 matrix: [\\pm45 PW]';
ts(sw).E = 2e5; % N/mm^2
ts(sw).nu = 0.290;

ts(sb).chord=15.07; %mm
ts(sb).depth=5.63;
```

```
ts(sb).arc_length=20.1;
ts(sb).thick_total= 0.13;
ts(sb).thick = 0.13;
ts(sb).length= 279.4; %mm, 11"
ts(sb).mass = 1.4; %g
ts(sb).material='CFRP';
ts(sb).descr=' 1-ply, T300 with 8552 matrix: [\\pm 45 PW] (bi-stable)';
ts(sb).E = 2e5; % N/mm^2
ts(sb).nu = 0.290;

ts(lb).chord= 14.88; %mm
ts(lb).depth= 7.4;
ts(lb).arc_length= 23.13;
ts(lb).thick_total=0.28;
ts(lb).thick = 0.28;
ts(lb).length= 495.3; %mm, 19 1/2
ts(lb).mass = 5.0; %g
ts(lb).material='CFRP';
ts(lb).descr=' 3-ply, S-2 glass with 8552 matrix: [\\pm45 PW, 0 UT, \\
    pm45 PW]';
ts(lb).E = 2e5; % N/mm^2
ts(lb).nu = 0.290;

% 2.25" = 57.15 mm
ts(12).chord=56.5;
ts(12).depth=28.03;
ts(12).arc_length=89.6;
ts(12).thick_total=0.42;
ts(12).thick = 0.420;
ts(12).length= 457.2; %mm
ts(12).mass = 25.0; %g
ts(12).material='CFRP';
ts(12).descr=' 2-ply, IM7 with RS-36 matrix: [\\pm 45 PW, \\pm 45 PW] (
    bi-stable)';
```

```
ts(12).E = 2e5;
ts(12).nu = .3;

for i=1:ts_type_count

    % https://en.wikipedia.org/wiki/Sagitta_%28geometry%29
    ts(i).radius = (ts(i).depth^2+ts(i).chord^2)/(2*ts(i).depth);
    ts(i).curvature = 1/(ts(i).radius);

    % http://mathcentral.uregina.ca/QQ/database/QQ.09.09/h/darryl1.html
    ts(i).curv_error = ts(i).chord - 2*ts(i).radius*sin(ts(i).arc_length
        /(2*ts(i).radius));
    % == +- 2 % (approx)

    % this works correctly, but the inaccuracy of measurements dominates
        the
    % radius (thus the curvature) calculations
    % ts(i).radius_check = fzero( @(r) 2*r*sin(ts(i).arc_length/(2*r))-
        ts(i).chord, ts(i).radius ) ;
    ts(i).alpha_rad = ts(i).arc_length*ts(i).curvature;
    ts(i).alpha_deg = rad2deg(ts(i).alpha_rad);

    ro = ts(i).radius+ts(i).thick/2;
    ri = ts(i).radius-ts(i).thick/2;
    % locate the neutral axis, distance from span midpoint
    % via [http://mathworld.wolfram.com/CircularSector.html]
    %  Beyer, W. H. (Ed.). CRC Standard Mathematical Tables, 28th ed.
        Boca Raton, FL: CRC Press, p. 125, 1987.
    Ai = ri^2/2*ts(i).alpha_rad;
    Ao = ro^2/2*ts(i).alpha_rad;
    xi = 4*ri/(3*ts(i).alpha_rad)*sin(ts(i).alpha_rad/2);
    xo = 4*ro/(3*ts(i).alpha_rad)*sin(ts(i).alpha_rad/2);

    % xd = ((Ao) * (xo) - (Ai)*(xi))/((Ad))
```

```
    %   Ad = Ao - Ai
    ts(i).neutral_offset= ((Ao)*(xo) -(Ai)*(xi))/((Ao-Ai));
    % finally subtract this location from the already calculated radius:
    % i.e. location from the center of the arc.
    ts(i).neutral_offset= ts(i).radius - ts(i).neutral_offset;

    % check this figure using the centroid of a circle arc from here:
    %http://www.engineering.com/Library/ArticlesPage/tabid/85/
        articleType/ArticleView/articleId/109/Centroids-of-Common-Shapes.
        aspx
    % r_arc = r * sin(theta)/theta

    ts(i).density_length=ts(i).mass/ts(i).length;

    % flexural rigidity:
    % D = (E t^3)/(12(1-nu))
    ts(i).flex_rigid = (ts(i).E*ts(i).thick^3)/(12*(1-ts(i).nu));

end

fprintf('%% Tape Spring properties (Body)\n');
fprintf('\\begin{table}[h!b!p!]\n');
fprintf('\\caption{Tape Spring Properties}\n');
fprintf('\\label{tbl:tape_spring_prop_body}\n');
fprintf('%%\\renewcommand*\\arraystretch{0.9}\n');
fprintf('\\begin{tabular*}{1.0\\textwidth}{l r r r r r r r}   \n');

fprintf('\\# & Thickness&    Angle& Curvature&   Width&   Chord&
    Radius&   Depth \\\\ \n');
fprintf('   &           t& $\\alpha$& $\\kappa$&       a&       c&
          R&       d \\\\ \n');
fprintf('   &        (mm)&    (deg)& (mm$^{-1}$)&    (mm)&    (mm)&
    (mm)&    (mm) \\\\ \n');
fprintf('Steel '); printncol(7);
```

```
fprintf('\\hline \n');
for i=1:ts_ct_steel
  fprintf('%2d&     % 6.2f&    % 6.1f&', i, ts(i).thick, ts(i).
     alpha_deg);
  fprintf('    % 6.3f& % 6.2f& % 6.2f&', ts(i).curvature, ts(i).
     arc_length, ts(i).chord);
  fprintf(' % 7.2f& % 6.2f \\\\ \n', ts(i).radius, ts(i).depth);
end
printncol(7);
fprintf('CFRP ');printncol(7);

fprintf('\\hline \n');
for i=ts_dx_cfrp
  fprintf('%2d&     % 6.3f&     % 6.1f&    % 6.3f& % 6.2f& % 6.2f&',
     i, ts(i).thick, ts(i).alpha_deg, ts(i).curvature, ts(i).arc_length
     , ts(i).chord);
  fprintf('  % 6.2f& % 6.2f \\\\ \n', ts(i).radius, ts(i).depth);
end

fprintf('\\end{tabular*}\n');
fprintf('\\end{table}\n\n\n\\n');



fprintf('%%Tape Spring properties (Appx)\n');
fprintf('\\begin{table}[h!b!p!]\n');
fprintf(['\\caption{Tape spring Properties.  These measurements were
   performed on specific samples, and thus some' ...
      'lengths may be different than the hinges used in experiments
         .}\n']);
fprintf('\\label{tbl:tape_spring_prop_appx}\n');
fprintf('%%\\renewcommand*\\arraystretch{0.9}\n');
fprintf('\\begin{tabular*}{1.0\\textwidth}{l r r r r r r r}   \n');
%fprintf('\\#&    Thickness&   Angle   &   Curvature&   Width&  Chord&
    Radius&   Depth&  neut. offs.&\n');
```

```
%fprintf('  &          t&     $alpha$&     $kappa$&      a&      c&
          R&       d&              &\n');
%fprintf('  &       (cm)&      (deg)&          (-)&    (cm)&   (cm)&
       (cm)&    (cm)&          (cm)&\n');

fprintf(' Tape Spring \\# (Steel)& ');
for i=1:ts_ct_steel
   fprintf('      %2d&', i);
end
fprintf(' \\\\ \n \\hline ');
fprintf('      ' \n Thickness (mm)&        ');
for i=1:ts_ct_steel
   fprintf(' % 6.2f&', ts(i).thick);
end
fprintf(' \\\\ \n Angle ($^\\circ$)&       ');
for i=1:ts_ct_steel
   fprintf(' % 6.1f&', ts(i).alpha_deg);
end
fprintf(' \\\\ \n Curvature (mm$^{-1}$)& ');
for i=1:ts_ct_steel
   fprintf(' % 6.3f&', ts(i).curvature);
end
fprintf(' \\\\ \n Radius (mm)&           ');
for i=1:ts_ct_steel
   fprintf(' % 6.1f&', ts(i).radius);
end
fprintf(' \\\\ \n Arc Width (mm)&        ');
for i=1:ts_ct_steel
   fprintf(' % 6.2f&', ts(i).arc_length);
end
fprintf(' \\\\ \n Chord Width (mm)&      ');
for i=1:ts_ct_steel
   fprintf(' % 6.2f&', ts(i).chord);
end
```

```
    fprintf(' \\\\ \n Length (mm)&           ');
286 for i=1:ts_ct_steel
        fprintf(' % 6.1f&', ts(i).length);
    end
    fprintf(' \\\\ \n Mass (g)&              ');
    for i=1:ts_ct_steel
291     fprintf(' % 6.1f&', ts(i).mass);
    end
    fprintf(' \\\\ \n Material&              ');
    for i=1:ts_ct_steel
        fprintf('  %s&', ts(i).material);
296 end
    fprintf(' \\\\ \n Density (g/mm) &       ');
    for i=1:ts_ct_steel
        fprintf(' % 6.3f&', ts(i).density_length);
    end
301 fprintf(' \\\\ \n Flexural Rigidity (D)& ');
    for i=1:ts_ct_steel
        fprintf(' % 6.2f&', ts(i).flex_rigid);
    end
    fprintf(' \\\\ \n');
306 printncol(7);

    fprintf(' Tape Spring \\# (CFRP)&     ');
    for i=ts_dx_cfrp
        fprintf('      %2d&', i);
311 end
    fprintf(' \\\\ \n \\hline ');
    fprintf(' \n Thickness (mm)&         ');
    for i=ts_dx_cfrp
        fprintf(' % 6.2f&', ts(i).thick);
316 end
    fprintf(' \\\\ \n Angle ($^\\circ$)&      ');
    for i=ts_dx_cfrp
```

```
    fprintf(' %6.1f&', ts(i).alpha_deg);
end
fprintf(' \\\\ \n Curvature (mm$^{-1}$)& ');
for i=ts_dx_cfrp
    fprintf(' %6.3f&', ts(i).curvature);
end
fprintf(' \\\\ \n Radius (mm)&          ');
for i=ts_dx_cfrp
    fprintf(' %6.1f&', ts(i).radius);
end
fprintf(' \\\\ \n Arc Width (mm)&       ');
for i=ts_dx_cfrp
    fprintf(' %6.2f&', ts(i).arc_length);
end
fprintf(' \\\\ \n Chord Width (mm)&     ');
for i=ts_dx_cfrp
    fprintf(' %6.2f&', ts(i).chord);
end
fprintf(' \\\\ \n Length (mm)&          ');
for i=ts_dx_cfrp
    fprintf(' %6.1f&', ts(i).length);
end
fprintf(' \\\\ \n Mass (g)&             ');
for i=ts_dx_cfrp
    fprintf(' %6.2f&', ts(i).mass);
end
fprintf(' \\\\ \n Material&             ');
for i=ts_dx_cfrp
    fprintf('    %s&', ts(i).material);
end
fprintf(' \\\\ \n Density (g/mm) &      ');
for i=ts_dx_cfrp
    fprintf(' %6.3f&', ts(i).density_length);
end
```

```
    fprintf(' \\\\ \n Flexural Rigidity (D)& ');
    for i=ts_dx_cfrp
       fprintf(' % 6.2f&', ts(i).flex_rigid);
356 end
    fprintf(' \\\\ \n');
    fprintf('\\end{tabular*}\n');
    fprintf('\\end{table}\n\n\n\n');
    clear ri ro xi xo Ao Ai
361

    %% PVC pipe physical properties
    pvc_type_count =4;
    pvc(pvc_type_count).thick_avg=0; % partial pre-allocation
366
    pvc(1).thick_list=[2.75 2.93 2.87 2.94 2.90 2.89 2.87 2.86 2.80]';
    pvc(1).inner_dia_list=[15.68 15.69 15.72 15.70 15.73 15.69 15.73 15.70
        15.74]';
    pvc(1).outer_dia_list=[21.36 21.48 21.31 21.37 21.41 21.31 21.36 21.39
        21.40]';
    pvc(1).length=223.84; % mm
371 pvc(1).mass = [82.4]';%[119]';

    pvc(2).thick_list=pvc(1).thick_list;
    pvc(2).inner_dia_list=pvc(1).inner_dia_list;
    pvc(2).outer_dia_list=pvc(1).outer_dia_list;
376 pvc(2).length=254; % mm
    pvc(2).mass = [61]';

    pvc(3).thick_list=[3.31 3.28 3.28 3.33 3.38 3.29]';
    pvc(3).inner_dia_list=[20.08 19.69 20.18 19.97 19.98 20.18 20.11]';
381 pvc(3).outer_dia_list=[26.71 26.79 26.70 26.68 26.82 26.81 26.77]';
    pvc(3).length=223.84; %mm
    pvc(3).mass=[97.2 97.1 97.0]';
```

```
pvc(4).thick_list=[3.75 3.72 3.67 3.63 3.64 3.76 3.63]';
pvc(4).inner_dia_list=[34.44 34.37 35.00 34.50 35.03 34.36]';
pvc(4).outer_dia_list=[42.35 42.33 42.04 41.84 41.83 42.07]';
pvc(4).length=279.4; %mm
pvc(4).mass=[179.1 179.0 178.9]';

for i= 1:pvc_type_count
    pvc(i).thick_avg=mean(pvc(i).thick_list);
    pvc(i).thick_stddev=std(pvc(i).thick_list);
    pvc(i).inner_dia_avg=mean(pvc(i).inner_dia_list);
    pvc(i).inner_dia_stddev=std(pvc(i).inner_dia_list);
    pvc(i).outer_dia_avg=mean(pvc(i).outer_dia_list);
    pvc(i).outer_dia_stddev=std(pvc(i).outer_dia_list);
    pvc(i).length_avg= mean(pvc(i).length);
    pvc(i).length_stddev= std(pvc(i).length);
    pvc(i).mass_avg=mean(pvc(i).mass);
    pvc(i).mass_stddev=std(pvc(i).mass);

    pvc(i).volume = pi*pvc(i).length_avg*((pvc(i).outer_dia_avg/2)^2-(
        pvc(i).inner_dia_avg/2)^2);
    % should be 1.3-1.45 g/cc
    pvc(i).density_volume = pvc(i).mass_avg/pvc(i).volume;
    pvc(i).density_length = pvc(i).mass_avg/pvc(i).length;

    % pipe end to hinge center:
    r = 69.85;
    % g * mm^2
    %pvc(i).MOI = pvc(i).mass_avg*(pvc(i).length_avg+r)^2+  1/12*pvc(i)
        .mass_avg*(pvc(i).length_avg)^2;
    pvc(i).MOI = pvc(i).mass_avg*(pvc(i).length_avg+r)^2+  1/12*pvc(i).
        mass_avg*(pvc(i).length_avg)^2+(36.6)*(r+30)^2;
end

pipe_range = 1:pvc_type_count;
```

```
fprintf('%% PVC Pipe properties\n');
fprintf('%% NOTE: pipe 1 weight includes screws, reflectors and clamps
    \n');
fprintf('\\begin{table}[h!b!p!]\n');
fprintf('\\caption{PVC Pipe Properties}\n');
fprintf('\\label{tbl:pipe_prop_appx}\n');
fprintf('%%\\renewcommand*\\arraystretch{0.9}\n');
fprintf('\\begin{tabular*}{1.0\\textwidth}{l | r r r r r }   \n');

fprintf(' Pipe \\# (PVC)&                    ');
for i=pipe_range
   fprintf('      %2d&', i);
end
fprintf('\n \\\\ \\hline \n Mass (g)&                        ');
for i= pipe_range
    fprintf('    % 6.1f&', pvc(i).mass_avg);
end
fprintf('\\\\ \n Length (mm)&                     ');
for i= pipe_range
    fprintf('    % 6.1f&', pvc(i).length);
end
fprintf('\\\\ \n Inner Diameter (mm)&             ');
for i= pipe_range
    fprintf('    % 6.2f&', pvc(i).inner_dia_avg);
end
fprintf('\\\\ \n Outer Diameter (mm)&             ');
for i= pipe_range
    fprintf('    % 6.2f&', pvc(i).outer_dia_avg);
end
fprintf('\\\\ \n Thickness (mm) &                 ');
for i= pipe_range
    fprintf('    % 6.2f&', pvc(i).thick_avg);
end
fprintf('\\\\ \n Linear Density (g/mm)&           ');
```

```
    for i= pipe_range
        fprintf('    % 6.2f&', pvc(i).density_length);
451 end
    fprintf('\\\\ \n Volumetric Density &            ');
    for i= pipe_range
        fprintf('    % 6.5f&', pvc(i).density_volume);
    end
456 fprintf('\\\\ \n (g/mm$^2$)&    & & & &  ');
    fprintf('\\\\ \n Moment of Inertia &             ');
    for i= pipe_range
        fprintf('    % 10.3g&', pvc(i).MOI);
    end
461 fprintf('\\\\ \n (g/mm$^2$)&    & & & &  ');
    fprintf(' \\\\ \n');
    fprintf('\\end{tabular*}\n');
    fprintf('\\end{table}\n\n\n');


466 mats.ts = ts;
    mats.pvc = pvc;

    procDir = 'preproc_data/';
    save([procDir 'materials.mat'], 'mats');
471

    clear ts pvc procDir
```

../model/measurements.m

```
    function [debug, blocks] = TapeSpringAnalysis_Import_Data()
    % This files reads the data needed to determine the joint coordinate
  3 % system, which it then saves

        global g raw proc

        [g, raw, proc] = load_globals();
  8     load_dependencies();
```

```
% initialize empty return values
debug=[];
for i=1:proc.block_count
    blocks(i).set_count =0;
end

%% ===== Validation Data =====
fprintf('\n\n ====== ====== Starting Data Import Module: ======
    ======= \n');
% debugged, but not necessary yet.
fprintf(' >> loading previous data\n');
[blocks,~] = load_proc_data(proc.dir,proc.ext);

% Automatic file loading:
search_path=strcat(raw.dir,'*',raw.ext);
fileSets=getFileList(search_path,raw.dir, raw.ext);
filesToProcess = 1:length(fileSets);
%filesToProcess = [9:10]; % specify here which files to process

for setIndex=filesToProcess
    % get the data from the requested file fit to 'dataColumnCount'
        columns
    fprintf('[%d]  ', setIndex);
    fileSets(setIndex).data = getFileData(fileSets(setIndex).fname,
        raw.dataColumnCount);
    curSet = fileSets(setIndex);
    if size(curSet.data)<[proc.initial_value_size raw.
        dataColumnCount]
        fprintf('        !! dataSet is too small:  Length= %d < %d
            \n', dslen, initial_value_size);
        continue;
    end
```

```
38          % Clean Data, begining and ending trail, double sampling
            % remove placeholder first line and start up lines
            curSet = cullData(curSet);
            if any(size(curSet.data)<[proc.initial_value_size raw.
                dataColumnCount])
                fprintf('        !! culled too much of this file.  Skipping
                    :');
43              fprintf('    [%d, %d]\n',size(curSet.data,1), size(curSet.
                    data,2));
                continue;
            end

            % split this file/dataset into multiple trials,
48          % discarding the long intervals of zero data:
            [blocks,curSet] = splitData(curSet, blocks);
            %plot_fileset(curSet, [6:8,12:14]);
        end

53      %plot_trial(blocks, [1], [4], [4]);
        write_proc_data(blocks, proc);
    end

    function load_dependencies()
58  % This function loads any libraries necessary for subsequent code
    % Also, it checks if the libraries are already added or not,
    %     so the .m search path isn't polluted.

        if ~exist('dcm2q','file')
63          addpath('quaternions/');
            fprintf('Adding Quaternion Library to Matlab Path.\n');
        end

    end
68
```

```
function [g, raw, proc] = load_globals()
%  centralize all global settings.  If a variable isn't needed in
%  more than one function, it shouldn't be global (or here)

73      global g raw proc
        clear g raw proc

        g.ffign=1;
        g.mfign=1;
78
        raw.iter_ctr=0;
        raw.max_iters=50;
        raw.sample_rate = 100; % Hz
        raw.dt = 1/raw.sample_rate; % sec
83      raw.DataValues=14; %15 if the cammera trigger is recorded, 14
            otherwise
        raw.dataColumnCount = raw.DataValues;
        raw.dir='raw_data/';
        raw.ext='.dat';
        raw.acceptable_loss_rate = 0.2; % 20% frame loss rate
88
        proc.default_reference_side=2; %the movement is with respect to
            this side
        proc.initial_value_size=80; % number of indices to average for
            initial value
        proc.dir='preproc_data/';
        proc.ext='.mat';
93      proc.block_count = 4;

        % displacements larger than this trigger a trial
        proc.trial.start_thresh=20.0;
        % when the values are less than this (abs) value, (for duration)
            the trial ends.
98      proc.trial.end_thresh=20.0;
```

```
        % body must stay below the threshold for this many seconds to end a
            trial.
        proc.trial.end_dur=0.8;
        % if a body moves less than this amount, it's classified as a base.
        proc.choose_base_thresh=7.0;
        % if displ value goes below this number, the trial is reject,
        % because the hinge probably failed.
        proc.trial.reject_low_thresh = [-1000, -800, -100];

        return;
end

    function plot_trial(blocks, bnums, snums, tnums)
        global g raw proc

        time_mask_start=1;   % start plotting at this time index
        time_mask_stop=inf;  % stop plotting at this time index
        plot_density = 25;   % plot every 'plot_density ' indices of data

        if (bnums > proc.block_count)
            fprintf('  !!Block number too big. Unable to plot.\n');
            return;
        elseif snums > blocks(bnums).set_count
            fprintf('  !!Set number too large. Unable to plot.\n');
            return;
        elseif tnums > blocks(bnums).sets(snums).trial_count
            fprintf('  !!Trial number too large. Unable to plot.\n');
            return;
        end

        % ==== Plot Deflections (mean-zeroed) =====
        mfign=559; % i.e. Mother FIGure Number
        for block_index = bnums
            for set_index = snums
```

```
for trial_index = tnums
    tr = blocks(block_index).sets(set_index).trials(
        trial_index);

    mfign=mfign+1;
    figure(mfign);
    id_str=sprintf(' [Block %d, Set %d, Trial %d]',
        block_index, set_index, trial_index);
    set(mfign, 'Name', id_str);
    mask=max(time_mask_start,1):min(time_mask_stop, tr.
        length);
    subplot(1,2,1);
    plot(tr.time(mask),tr.free.displs(mask,1),'k-x', ...
         tr.time(mask),tr.free.displs(mask,2),'k-+', ...
         tr.time(mask),tr.free.displs(mask,3),'k-o', ...
         tr.time(mask),tr.base.displs(mask,1),'r-x', ...
         tr.time(mask),tr.base.displs(mask,2),'r-+', ...
         tr.time(mask),tr.base.displs(mask,3),'r-o');
    legend('Free Body, u_1', 'Free Body, u_2', 'Free Body,
        u_3', ...
            'Fixed Body, u_1', 'Fixed Body, u_2', 'Fixed
                Body, u_3');
    xlabel('time (sec)');
    ylabel('mm');
    title('Displacements (Inertial, Initial-Zero)');
    %text(mean(t(:,1)),min(Pose1(:,1))-30,'X');
    %text(mean(t(:,1)),min(Pose1(:,2))-30,'Y');
    %text(mean(t(:,1)),min(Pose1(:,3))-30,'Z, Vicon
        Coordinates');

    subplot(1,2,2);
    plot( tr.time(mask),rad2deg(tr.free.angles(mask,1)),'k-
        x', ...
```

```
158             tr.time(mask),rad2deg(tr.free.angles(mask,2)),'k
                    -+', ...
                tr.time(mask),rad2deg(tr.free.angles(mask,3)),'k-
                    o', ...
                tr.time(mask),rad2deg(tr.base.angles(mask,1)),'r-
                    x', ...
                tr.time(mask),rad2deg(tr.base.angles(mask,2)),'r
                    -+', ...
                tr.time(mask),rad2deg(tr.base.angles(mask,3)),'r-
                    o');
163         legend('Free Body, \theta_1=\gamma','Free Body, \
                theta_2=\theta', ...
                    'Free Body, \theta_3=\mu','Fixed Body, \theta_1'
                        , ...
                    'Fixed Body, \theta_2','Fixed Body, \theta_3');
            xlabel('time (sec)');
            ylabel('rad');
168         title('VICON Euler angles (Inertial, Absolute)');

            view_dir=[-26,24];
            %==== Plot Tip Deflections (relative)
            % Plot position in inertial frame.
173         % !! BRILLIANT.  Keep this plot.
            mfign=mfign+10;figure(mfign);clf;
            trial_num=1;
            plot3(tr.base.displs(:,1),tr.base.displs(:,2),tr.base.
                displs(:,3),'b-',...
                    tr.free.displs(:,1),tr.free.displs(:,2),tr.free.
                        displs(:,3),'r-.');
178         daspect([1,1,1]); hold on;
            legend('Base Body','Free Body', 'Location', 'NorthEast
                ');
            xlabel('x');ylabel('y');zlabel('z');
            title('Body Displacements in Inertial Frame.');
```

```
set(mfign, 'Name', id_str);

% ==== Plot absolute Positions, deflections
% draw the body axis for every "plot_density"th frame
mfign=mfign+1;figure(mfign);clf;
subplot(2,1,1);
daspect([1 1 1]); hold on;
frame_mask = max(time_mask_start,1):plot_density:min(
    time_mask_stop, tr.length);
for frameIndex = frame_mask
    % original lines of code f/Jennings
    %AxisDraw(TForm_2(eye(3),[0,0,0],Pose1(frameIndex
        ,4:6)),Pose1(frameIndex,1:3),50);
    %AxisDraw(TForm_2(eye(3),[0,0,0],Pose2(frameIndex
        ,4:6)),Pose2(frameIndex,1:3),50);

    AxisDraw(q2dcm(tr.free.q(frameIndex,:)')*eye(3),
        ...
            tr.free.displs(frameIndex,:)+tr.free.u0
                ,32);
    AxisDraw(q2dcm(tr.base.q(frameIndex,:)')*eye(3),
        ...
            tr.base.displs(frameIndex,:)+tr.base.u0
                ,32);
end
legend('i^\wedge','j^\wedge','k^\wedge', 'Location', '
    SouthEast');
xlabel('x');ylabel('y');zlabel('z');
hold off;
view(view_dir);
title_str=sprintf('Displacement and Orientation (
    Absolute, Inertial Frame)');
set(mfign, 'Name', id_str);
title(title_str);
```

```
208             %mfign=mfign+1;figure(mfign);clf;
                subplot(2,1,2);
                daspect([1,1,1]); hold on;
                xlabel('x_1');ylabel('y_1');zlabel('z_1');
                frame_mask = max(time_mask_start,1):plot_density:min(
                    time_mask_stop, tr.length);
213             for frameIndex= frame_mask
                    % original code:
                    %AxisDraw(TForm_1(TForm_2(eye(3),[0,0,0],Pose2(
                        frameIndex,4:6)),[0,0,0],Pose1(frameIndex,4:6))
                        ,...
                    %           Pose2_1(frameIndex,1:3),50);
                    AxisDraw(q2dcm(tr.dq(frameIndex,:)')*eye(3), tr.
                        free.displs(frameIndex,:), 50);
218             end
                view(view_dir);
                title_str=sprintf('Body_2 in Body_1 frame (Displacement
                     only)');
                set(mfign, 'Name', id_str);
                title(title_str);
223
                % checked on combined motion,
                % noticably higher error due to small baseline of
                    markers

                % find the deployed body position
228             %Pose2_1_base_all(datasetIndex,1:3)=mean(Pose2_1(:,1:3)
                    ,1);
                % disp(mean(Pose2_1(:,1:3),1))
                % pause

                drawnow;
233         end
```

```
        end
    end

end

function AxisDraw(DCM, center, Scale)

% the numeric residual actually allows for the inverse to work at
    theta2=90
% still should check for special cases at some point

    plot3( Scale*([0,DCM(1,1)])+center(1), Scale*([0,DCM(2,1)])+center
        (2), Scale*([0,DCM(3,1)])+center(3), 'r-', ...
            Scale*([0,DCM(1,2)])+center(1), Scale*([0,DCM(2,2)])+center
                (2), Scale*([0,DCM(3,2)])+center(3), 'g-', ...
            Scale*([0,DCM(1,3)])+center(1), Scale*([0,DCM(2,3)])+center
                (2), Scale*([0,DCM(3,3)])+center(3), 'b-');

    % originial code
    % AxisDraw=@(DCM, Cent, Scale) ...
%     plot3( Scale*([0,DCM(1,1)])+Cent(1), Scale*([0,DCM(1,2)])+Cent(2),
    Scale*([0,DCM(1,3)])+Cent(3), 'r-',...
%           Scale*([0,DCM(2,1)])+Cent(1), Scale*([0,DCM(2,2)])+Cent(2),
    Scale*([0,DCM(2,3)])+Cent(3), 'g-',...
%           Scale*([0,DCM(3,1)])+Cent(1), Scale*([0,DCM(3,2)])+Cent(2),
    Scale*([0,DCM(3,3)])+Cent(3), 'b-');
end

function [curSet] = cullData(curSet)
    data = curSet.data;

    % Clean Data, begining and ending trail, double sampling
    % remove placeholder first line and start up lines
```

```
        data(1:19,:)=[];   % data(1:2,:)=[];   %% data(1:MAGIC_NUMBER_00,:)
            =[];

        %length(data)
263     %firstIndex = data(1,1)
        %% Remove constant values at begining and end
        % this is either buggy, or removes different constant
        % values.
        MAGIC_NUMBER_00=5;
268     temp1=find(all([any(diff(data(:,3:8)),2) ,...
                        any(diff(data(:,9:14)),2)],2),1,'first');
        temp2=find(all([any(diff(data(:,3:8)),2) ,...
                        any(diff(data(:,9:14)),2)],2),1,'last');
        data([1:(temp1+MAGIC_NUMBER_00),(temp2-MAGIC_NUMBER_00):end],:)=[];
273     temp3=find(any([all(data(:,3:8)==0,2),all(data(:,9:14)==0,2)],2));
        data(temp3,:)=[];
        clear temp1 temp2 temp3
        %firstIndex = data(1,1)
        %lastIndex = data(end,1)
278     %length(data)
        %return;

        % NEED to do bad fit detection
        % Remove samples with duplicate times
283     % (currently redundant)
        [~,goodIndices]=unique(data(:,1),'first');
        data=data(goodIndices,:);
        curSet.data = data;
        %clear goodIndices
288 end

    function [blocks,fileSet] = splitData(fileSet, blocks)
    % Splits the data in a file into specific trials within a set.
    % Data is stored in this pattern:
```

```
293 %     blocks(i).sets(j).trials(k).displacement(time,dimension)
    %     blocks(i).sets(j).material

      global g raw proc
      trial_start_threshold = proc.trial.start_thresh;
298   trial_end_threshold = proc.trial.end_thresh;
      trial_end_duration = proc.trial.end_dur;

      % these are hand-picked from looking at data... could add a
      % better detection algorithm
303   % this isn't really used anymore...
      trial_duration_offset=12*raw.sample_rate;  % time in seconds *
          sample_rate

      initial_interval = [1:proc.initial_value_size];
      dslen = length(fileSet.data); % dataset length
308
      time=(fileSet.data(:,1)-fileSet.data(1,1))/raw.sample_rate;
      if proc.trial.start_thresh < proc.trial.end_thresh
          proc.trial.start_thresh = proc.trial.end_thresh ;
      end
313
      %==== zero mean the displacements ====
      if proc.block_count < fileSet.block_num
          fprintf('    !! file is not in a recognized block. Aborting
              split.\n');
          return;
318   end

      %==== zero mean the displacements ====
      % calculate mean values for each data column:
      initial_values = mean(fileSet.data(initial_interval,:));
323   % except for these columns ...
      initial_values([1:2])=0;
```

```
initial_values([1:2,3:5,9:11])=0;

% now zero the remaining columns:
fileSet.data(:,:) = fileSet.data(:,:) - repmat(initial_values,
    dslen, 1);

%  side 1 is the reference.
max_side_1 = max(abs(fileSet.data(:,[6:8])));
ReferenceSide=proc.default_reference_side;
if max(max_side_1) < proc.choose_base_thresh
    ReferenceSide=1;
else
    ReferenceSide=2;
end
% DEBUG DEBUG DEBUG
%ReferenceSide
%max_side_1
clear max_side_1

% this leaves free side at 3:8, and base at 9:14
% [positions, angles]
if ReferenceSide==2
    fileSet.data(:,3:14)=fileSet.data(:,[3:8, 9:14]);
else
    fileSet.data(:,3:14)=fileSet.data(:,[9:14, 3:8]);
end

% ===== split off each trial =====
trial_found_count=0;
trial_start_index=0;
trial_end_index=0;

% this finds the first index where a displacement is further from
% the starting average than the threshold
```

```
358     % (free end displacement is columns [9:14])
        search_offset=1;
        [~,index] = find(abs(fileSet.data(:,6:8)') > trial_start_threshold,
            1);

        % loop while we have data left
363     raw.iter_ctr=1;
        while length(index) > 0
          trial_found_count = trial_found_count +1;

          %fprintf('  ... found a value exceeding threshold. splitting out
              a trial.\n');
368       % block off rough times before and after— this becomes a trial
          dslen = length(fileSet.data); % dataset length
          trial_start_coarse= index;
          trial_end_coarse= min(trial_start_coarse+trial_duration_offset,
              dslen);
          % local time
373
          trial_start_index = trial_start_coarse;
          %need to overhaul this block...the max may not be important...
          % % find a max within this window, and start just before that...
          %[max_values, max_indices] = max(abs(data(trial_start_coarse:
              trial_end_coarse,3:5)),[],1);
378       % [tr.max_value, max_val_col] = max(max_values);
          %tr.max_index = trial_start_offset;  % because we're discarding
              everything
          %                                    % before the max.
          % trial_start_index = max_indices(max_val_col)+trial_start_coarse
              ;
          % clear max_values max_vol_col
383
          % ... until the disturbances die down...
          [~, trial_end_index] = find_trial(fileSet.data(:,6:8), index);
```

```
    if -1 == trial_end_index
        fprintf('  !! Failed to find the trial end.  Adjust search
            window.\n');
388 end

    % correct:
    %max_time = time(max_indices(max_val_col)+trial_start_coarse-1)-
        time(1)

393 % this trial, simple data
    tr=[];
    tr.date = fileSet.date;
    tr.t0 = time(trial_start_index);
    tr.time = time(trial_start_index:trial_end_index)-tr.t0;
398 tr.base.displs = fileSet.data(trial_start_index:trial_end_index
        ,12:14);
    tr.base.u0 = initial_values(12:14);
    tr.base.angles = fileSet.data(trial_start_index:trial_end_index
        ,9:11);
    tr.base.a0 = initial_values(9:11);
    tr.free.displs = fileSet.data(trial_start_index:trial_end_index
        ,6:8);
403 tr.free.u0 = initial_values(6:8);
    tr.free.angles = fileSet.data(trial_start_index:trial_end_index
        ,3:5);
    tr.free.a0 = initial_values(3:5);
    tr.size = size(tr.base.angles);
    tr.length = trial_end_index-trial_start_index+1;
408

    % add quaternion representation of angles
    tr.free.q = zeros(tr.length, 4);
    tr.base.q = zeros(tr.length, 4);
    tr.dq = zeros(tr.length, 4);
413 for frameIndex = 1:tr.length
```

```
        tr.free.q(frameIndex,:) =dcm2q(xform_g2b(eye(3), zeros(1,3),
            tr.free.angles(frameIndex,:)));
        tr.base.q(frameIndex,:) =dcm2q(xform_g2b(eye(3), zeros(1,3),
            tr.base.angles(frameIndex,:)));
    end
    % rotation of free side relative to bose:
    % dq = qmult(qconj(qbase),qfree)
    tr.dq = qmult(qconj(tr.base.q'), tr.free.q')';

    % if trial exhibits snap-through, reject
    accept_status = reject_trial(tr, proc);
    if false== accept_status
        fprintf('        !!  Snap-through.');
    end

    avg_dt = (tr.time(end)-tr.time(1))/tr.length;
    debug.avg_dt(raw.iter_ctr) = avg_dt;
    if avg_dt > raw.dt/(1 -raw.acceptable_loss_rate)
        %accept_status = false;
        fprintf('        !! High Frame Loss.');
        fprintf('      average step size: %g ', debug.avg_dt(raw.
            iter_ctr));
    end

    if false == accept_status
        fprintf('      Trial #%d Rejected.\n',trial_found_count);
    else
        % DEBUG DEBUG DEBUG
        fprintf('      >>> detected trial %d start...',
            trial_found_count);
        %fprintf(' i={%8d-%8d of %8d} ',trial_start_coarse,
            trial_end_coarse, dslen);
        %fprintf(' (%g @%g sec)',time(trial_end_coarse)-time(
            trial_start_coarse), time(trial_start_coarse));
```

```
        %fprintf('\n');
443     %fprintf('        >>> shrinking trial %d: ',
            trial_found_count);
        %fprintf('i={%8d-%8d of %8d} ',trial_start_index,
            trial_end_index, dslen);
        fprintf('(%g-%g ',time(trial_start_index), time(
            trial_end_index));
        fprintf('(%g)) ', time(trial_end_index)-time(
            trial_start_index));
        fprintf('\n');
448     % DEBUG DEBUG DEBUG

        % merge this trial into the data tree
        block_index=fileSet.block_num;
        set_index=fileSet.set_num;
453
        if ((blocks(block_index).set_count < set_index) || (size(
            blocks(block_index).sets(set_index).trial_count,2) < 1))
            fprintf('          >> New set: adding first trial: b%d s
                %d t%d\n', block_index, set_index, 1);
            % new set.  populate all the fields
            blocks(block_index).sets(set_index).trial_count = 1;
458         blocks(block_index).sets(set_index).trials(1) = tr;
            blocks(block_index).sets(set_index).hinge_num = fileSet.
                hinge_num;
            blocks(block_index).sets(set_index).pipe_num = fileSet.
                pipe_num;
            blocks(block_index).sets(set_index).multiplicity =
                fileSet.multiplicity;
            blocks(block_index).set_count = max( blocks(block_index).
                set_count,set_index);
463     else
            % existing set...
            %fprintf('    >>> existing set ...');
```

```
            ctrials = blocks(block_index).sets(set_index).trials;
468         trial_found = false;
            for trial_index = 1:blocks(block_index).sets(set_index).
                trial_count
                % check if this trial already exists
                if (tr.date == ctrials(trial_index).date) && (tr.t0
                    == ctrials(trial_index).t0)
                    % if duplicate, discard, and start the next trial
                        .
473                 %fprintf('  duplicate trial: skipping: b%d s%d t%
                        d\n', ...
                    %        block_index, set_index, trial_index);
                    trial_found = true;
                    break;
                end
478         end
            if ~trial_found
                % could not find trial in existing data.
                trial_index = trial_index+1;
                fprintf('          >>> new trial: adding@ b%d s%d t%d
                    \n', block_index, set_index, trial_index);
483             blocks(block_index).sets(set_index).trial_count =
                    trial_index;
                blocks(block_index).sets(set_index).trials(
                    trial_index) = tr;
            end
            clear ctrials trial_found
        end
488     end

        % record where search has progressed to...
        search_offset = trial_end_index+0.5 *raw.sample_rate;
```

```
493         % Find the next displacement index larger than threshold
            [~,index] = find(abs(fileSet.data(search_offset:end,6:8)') > proc
                .trial.start_thresh, 1);
            index = index+search_offset;

            raw.iter_ctr= raw.iter_ctr+1;
498         if raw.iter_ctr > raw.max_iters
                fprintf('!! triggered iteration guard: >25\n');
                return;
            end
        end
503     %fprintf('  ... finished splitting out trials.\n');
    end

    function [beginIndex, endIndex] = find_trial( data, startIndex)
    % finds when the given data columns 'data' all fall below some
        threshold 'thresh'
508
        global g raw proc
        sample_rate = raw.sample_rate;
        thresh= proc.trial.end_thresh;
        duration= proc.trial.end_dur;
513     tryIndex = startIndex;
        beginIndex=-1;
        endIndex = -1; % i.e. default-to-error

        %% the easy way...
518     % [~,tryIndex] = find(abs(data(tryIndex:end,:)') > thresh, 1, 'last
            ');
        % if max(tryIndex)+0.1*sample_rate < size(data,1)
        %     fprintf('found tryIndex the easy way: %d:\n ',tryIndex);
        %     endIndex = tryIndex;
        %     return;
523     % else
```

```
        %     tryIndex=1;
        % end

        % the hard way....
528     while tryIndex < size(data,1)
            % find the next time signal falls below threshold
            [~,candidateIndex] = find(abs(data(tryIndex:end,:)') < thresh,
                1);
            candidateIndex = max(tryIndex) + candidateIndex;
            % then find how long this lull lasts:
533         [~,nextIndex] = find(abs(data(candidateIndex:end,:)') > thresh,
                1);
            nextIndex = min(nextIndex);
            %db_tryIndex = tryIndex
            %db_candidateIndex = candidateIndex
            %db_nextIndex = nextIndex
538
            if (length(nextIndex) == 0 ) || ( nextIndex > duration*
                sample_rate )
                %this candidate index passes. accept.
                %fprintf('    this candidate index passes. accept...\n');
                endIndex = tryIndex;
543             return;
            else
                %fprintf('    this candidate fails: %d \n', nextIndex);
                tryIndex = tryIndex + nextIndex -1 + sample_rate*0.2;
            end
548     end

    end



553
    function MyData = getFileData(filepath, column_count)
```

```
        fid = fopen(filepath,'r');
        if -1==fid
            fprintf('could not open file: "%s"\n                    in: "%s"\n
                ', filepath,pwd);
            return;
        end
        MyData = fscanf(fid,'%g',[column_count,inf]).';
        fclose(fid);
        fprintf('  >> Loaded "%s".  Processing ...\n', filepath);
    end

    function [datasets] = getFileList(search_path, raw_path, data_extension
        )
        fprintf('... Searching for raw data files to load... (*%s)\n',
            data_extension);
        search_list = ls(search_path);
        good_set_num=0;
        bad_set_num=0;
        datasets = [];
        % automatically load each file, and check for metadata in filename
        % if formatted, add to dataset to process next. Otherwise
        % reject and move on.
        for fileIndex = 1:size(search_list,1)
            [vars, var_count] =sscanf(search_list(fileIndex,:), 'data_b%d_s
                %d_h%d_p%d_m%d_%d_%s.dat', inf);

            if var_count == 7
                good_set_num= good_set_num+1;
                datasets(good_set_num).fname = strcat(raw_path, search_list
                    (fileIndex,:));
                datasets(good_set_num).hinge_num = vars(3);
                datasets(good_set_num).pipe_num = vars(4);
                % the set number combines the block, and set within
                % that block.  The hundreds digit is the block, the
```

```
            % tens,ones indicate the set.
            datasets(good_set_num).block_num = vars(1);
            datasets(good_set_num).set_num = vars(2);
            datasets(good_set_num).multiplicity=vars(5);
            datasets(good_set_num).date = vars(6);
            datasets(good_set_num).file_num = fileIndex;

            % debug
            %title_str=fprintf('    .. Found good file: %d: %s \n',
                fileIndex, datasets(good_set_num).fname);
        else
            % discard this file entry
            bad_set_num=bad_set_num+1;
            fprintf('    !! Found Bad data file: %s:', search_list(
                fileIndex,:));
            fprintf(' (Badly formatted file name.)\n');
        end
    end
    fprintf('>>> Finished scanning.  Processed %d/%d files.\n', ...
            good_set_num, good_set_num+bad_set_num);
end

function plot_fileset(fSet,ind)
    %id_str=sprintf(' [Block %d, Set %d, Trial %d]', block_index,
        set_index, trial_index);
    %set(mfign, 'Name', id_str);
    global g

    figure(g.ffign);
    fdata=fSet.data;
    time=(fdata(:,1)-fdata(1,1))/100;
    plot(time(:), fdata(:,ind(1)),'k-x', ...
         time(:), fdata(:,ind(2)),'k-+', ...
         time(:), fdata(:,ind(3)),'k-o', ...
```

```
            time(:), fdata(:,ind(4)),'r-x', ...
            time(:), fdata(:,ind(5)),'r-+', ...
            time(:), fdata(:,ind(6)),'r-o');
618     legend('Free Body, u_1', 'Free Body, u_2', 'Free Body, u_3', ...
              'Fixed Body, u_1', 'Fixed Body, u_2', 'Fixed Body, u_3');
        name_str=sprintf('[File #%d, Block %d, Set %d ]', ...
                         fSet.file_num, fSet.block_num, fSet.set_num );

623     set(g.ffign, 'Name', name_str);
        xlabel('time (sec)');
        ylabel('mm');
        title(['Displacements (Inertial, Initial-Zero)',name_str]);

628     g.ffign=g.ffign+1;
        drawnow;
    end

    function accept = reject_trial(trial, proc)
633
        % dummy value
        accept = true;
        %min_displ= min(min(trial.free.displs));
        if any(proc.trial.reject_low_thresh > min(trial.free.displs))
638         %fprintf('!! Trial is below threshold. rejecting.\n');
            accept=false;
        end
    end


643
    % was TForm_1
    function [gdata] = xform_b2g( data, offset, angles)
    % transform from TForm_2 is for taking euler coordinates into global
        (?)
    % body to inertial transform: 1-2-3 (-theta)
```

```
% ----
% Data = data to xform?
% offset = vector to displace each entry by offest (additive)
% Theta = rotation angles

    % R_1(theta(1)), R_2(theta(2)), R_3(theta(3)),
    c = cos(angles);
    s = sin(-angles);
    gdata = ([ c(2)*c(3), c(2)*s(3), -s(2) ; ...
                -c(1)*s(3)+s(1)*s(2)*c(3), c(1)*c(3)+s(1)*s(2)*s(3), s
                    (1)*c(2) ; ...
                s(1)*s(3)+c(1)*s(2)*c(3), -s(1)*c(3)+c(1)*s(2)*s(3), c
                    (1)*c(2); ...
              ]*(data'))';

    % original formulation: 1-2-3
    % theta=angles;
    % gdata = (([1,0,0;0,cos(theta(1)),-sin(theta(1));0,sin(theta(1)),
        cos(theta(1))]*...
    %           [cos(theta(2)),0,sin(theta(2));0,1,0;-sin(theta(2)),0,
        cos(theta(2))]*...
    %           [cos(theta(3)),-sin(theta(3)),0;sin(theta(3)),cos(
        theta(3)),0;0,0,1])*...
    %          (data')+repmat(offset(:),1,size(data,1)))';
end

function [ bdata ] = xform_g2b(data, offset, angles)
% Reference
% inertial to body transform: 3-2-1
%xform_g2b=(([cos(Theta(3)),--sin(Theta(3)),0;-sin(Theta(3)),cos(Theta
    (3)),0;0,0,1]*...
%               [cos(Theta(2)),0,---sin(Theta(2));0,1,0;--sin(Theta(2)),0,
    cos(Theta(2))]*...
```

```
%               [1,0,0;0,cos(Theta(1)),--sin(Theta(1));0,-sin(Theta(1)),
    cos(Theta(1))])*...
%               (Data.')+repmat(X(:),1,size(Data,1))).';

    % (body) = R_1(-theta(1))*R_2(-theta(2))*R_3(theta(3)) (inertial)
678     c = cos(angles);
        s = sin(-angles);
        dcm = ([ c(2)*c(3), c(2)*s(3), -s(2) ; ...
                   -c(1)*s(3)+s(1)*s(2)*c(3), c(1)*c(3)+s(1)*s(2)*s(3), s
                       (1)*c(2) ; ...
                   s(1)*s(3)+c(1)*s(2)*c(3), -s(1)*c(3)+c(1)*s(2)*s(3), c
                       (1)*c(2)]);
683     bdata = dcm*data;
end
```

../model/TapeSpringAnalysis_Import_Data.m

```
function [blocks, mats ]= load_proc_data( proc_dir, proc_ext)
% load all existing data block files
% data is stored in this pattern: block(i).set(j).trial(k).field

 5      fprintf(' >> loading preprocessed data ');
        block_count=4;
        for block_num = 1:block_count
            search_file = sprintf('%spreprocessed_b%1d%s', proc_dir,
                block_num, proc_ext);

10          if 2 == exist(search_file,'file')
          %fprintf('     .. loading %s\n', search_file);
                % yes, this syntax is broken.  deal.
                % (the return value from 'load(...)' is assigned
                % *under* the variable you assign it to...
15              blocks(block_num) = load(search_file,'sets','set_count');
                fprintf('. ');
            else
```

```
            % initialize empty return values
            blocks(block_num).set_count=0;
            blocks(block_num).sets=[];
        end
    end


    search_file = sprintf('%smaterials%s', proc_dir, proc_ext);
    if 2 == exist(search_file,'file')
        %fprintf('    .. loading %s\n', search_file);
        % yes, this syntax is broken.  deal.
        % (the return value from 'load(...)' is assigned
        % *under* the variable you assign it to...
        load(search_file,'mats');
        fprintf('. ');
    else
        % initialize empty return values
        fprintf('!!Could not load materials file.\n');
    end
    fprintf('    Done.\n');
end
```

../model/load_proc_data.m

```
function [debug, blocks, mats]= TapeSpringAnalysis_Dynamics
% This files reads the data needed to determine the joint coordinate
% system, which it then saves
    global g proc anly
    [debug, g, proc, anly]=load_globals();
    load_dependencies();

    mfign=22;
    fprintf('\n\n ====== ====== Starting Data Modeling: ====== ====== \
        n');
    [blocks, mats] = load_proc_data(proc.dir,proc.ext);
```

```
11
    % this trial is invalid.  Delete.
    % (in the slap-dash manner possible)
    blocks(3).sets(2).trial_count=20;
    blocks(3).sets(2).trials(21:end)=[];
16
    for block_index = 1:proc.block_count
        for set_index = 1:blocks(block_index).set_count
            blocks(block_index).sets(set_index).t_stop=0.5; % secs
        end
21  end

    % these chop plots to this length of time, for clarity in thesis
    % ts == time_stop
    blocks(1).sets(3).ts=0.18;
26  blocks(1).sets(4).ts=0.16;

    blocks(2).sets(1).ts=0.22;
    blocks(2).sets(2).ts=0.20;
    blocks(2).sets(3).ts=0.18;
31  blocks(2).sets(4).ts=0.16;

    blocks(3).sets(1).ts=0.16;
    blocks(3).sets(2).ts=0.18;
    blocks(3).sets(3).ts=0.22;
36  blocks(3).sets(4).ts=0.22;

    blocks(4).sets(1).ts=0.16;
    blocks(4).sets(2).ts=0.15;
    blocks(4).sets(3).ts=0.15;
41  blocks(4).sets(4).ts=0.14;
    anly.max_ts_index = floor(0.22*proc.sample_rate);
    g.blocks = blocks;
    g.mats = mats;
```

```
46      set_count =0;
        si=0;
        set_mean=[];
        % NaN values are ignored by the fit algorithm
        % (hardcoded.  Properly done, this would dynamically grow....
51      set_fit_data=NaN(anly.max_ts_index, anly.max_mean_count);
        %  set_fit_data holds all data sets as a column vector.  This
        %  combination concatenates each data set after the previous,
        %  padding with NaN values so that each is a constant length.

56      %plot_trial(blocks, 4, 4, 31);
        for block_index = 1:proc.block_count
        %for block_index = 1
            fprintf('Block %d: \n',block_index);
            for set_index = 1:blocks(block_index).set_count
61          %for set_index  = 3 % DEBUG
                if 0 == size(blocks(block_index).sets(set_index).
                    trial_count,1)
                    continue;
                end
                fprintf('  Set %d: \n', set_index);
66              % temporary, hopefully.
                blocks(block_index).sets(set_index).block_num = block_index
                    ;
                blocks(block_index).sets(set_index).set_num = set_index;

                %blocks(block_index).sets(set_index).
71              init_size=10; % this number is arbitrary. needs to be >0
                set_count = set_count+1;
                si = set_count;
                set_mean(si).blockn = block_index;
                set_mean(si).setn = set_index;
```

```
76          set_mean(si).pipen = blocks(block_index).sets(set_index).
                pipe_num;
            set_mean(si).tapen = blocks(block_index).sets(set_index).
                hinge_num;
            set_mean(si).multiplicity = blocks(block_index).sets(
                set_index).multiplicity;
            set_mean(si).angles=zeros(init_size,3);
            set_mean(si).sample_count=zeros(init_size,1);  % element
                wise count, b/c of dropped frames.
81          set_mean(si).length=size(set_mean(si).angles,1);
            M2 = zeros(init_size,3); % intermediate tally for variance
                calculation.
            for trial_index = 1:blocks(block_index).sets(set_index).
                trial_count;
            %for trial_index = [1:8]; % DEBUG
                id_str=sprintf('[Block %d, Set %d, Trial %d]',
                    block_index, set_index, trial_index);
86              tr = blocks(block_index).sets(set_index).trials(
                    trial_index);
                if 10 > tr.length
                    fprintf('      Sampling %s .... ',id_str);
                    fprintf('      !! bad trial:short.  Len=%d \n',
                        length(tr.free.angles));
                    continue;
91              end
                if 0.25 > max(tr.free.angles(:,2))
                    fprintf('      Sampling %s .... ',id_str);
                    fprintf('      !! bad trial:bad pull.  Max Val=%g \
                        n', max(tr.free.angles(:,2)));
                    continue;
96              end
                tr.rate = tr.length/(tr.time(end)-tr.time(1));
                debug.rate(trial_index) = tr.rate;
```

```
% detect integration interval start:
% maximum smoothed theta_2 value
smoothed_angle = smooth(tr.free.angles(:,2),7);
[max_val, max_ind] = max(smoothed_angle);
[ivl_start_ind]= find( proc.peak.slop < abs(
    smoothed_angle(max_ind:end)-max_val), 1, 'first');
ivl_start_ind = ivl_start_ind + max_ind;
clear max_val max_ind;

% detect integration interval end: first local (
    smoothed) minima
th_d_exp=diff(smoothed_angle(ivl_start_ind:tr.length));
[ivl_stop_ind]= find( th_d_exp > 0, 1, 'first');
if length(ivl_stop_ind) == 0
    % If no valley is found:
    ivl_stop_ind = length(tr.time);
else
    % (normal case)
    % offset to where 'find' started from
    ivl_stop_ind=ivl_stop_ind +ivl_start_ind+1;
    % make sure the index doesn't overflow the data
    ivl_stop_ind=min(ivl_stop_ind, length(tr.time));
end

show_indiv_plots=false;
if show_indiv_plots == true
    % ===== plot theta comparison =====
    % DEBUG!! DEBUG!! DEBUG!!
    %dbfign=1; figure(dbfign);
    mfign=mfign+1; figure(mfign);
    set(mfign, 'Name', id_str);
    plot(tr.time, tr.free.angles(:,2),'k-+', ...
         tr.time, smoothed_angle, 'b-x', ...
```

```
131                 ... %tr.time(ivl_start_ind:(tr.length-1)),
                        th_d_exp, 'r-x', ...
            tr.time(ivl_start_ind), tr.free.angles(
                ivl_start_ind,2), 'r-o', ...
                 tr.time(ivl_stop_ind), tr.free.angles(
                    ivl_stop_ind,2), 'r-o');
            %axis([10.3, 10.8, -0.2, 0.8]);
            xlabel('Time from Trial Start (sec)');
136         ylabel('Angular Position (rad)');
            legend('\theta_2, measured','\theta_2, smoothed','
                Location','SouthWest');
            % DEBUG!!  DEBUG!!  DEBUG!!
        end

141     mod_ivl = ivl_start_ind:ivl_stop_ind;
        %fprintf('    Sampling %s ....    ',id_str);
        %fprintf('   time: %6.3f - %6.3fg (%6.3f) sec \n', ...
        %         tr.time(mod_ivl(1)), tr.time(mod_ivl(end)),
            ...
        %         tr.time(mod_ivl(end)) -tr.time(mod_ivl(1)));
146
        % ===== calculate running average, standard deviation
            =====
        % for algorithm, see:
        %  http://www.jstor.org/pss/1266577
        % "Note on a Method for Calculating Corrected Sums of
            Squares and Products", B. P. Welford
151     %
        % this munging is necessary because the data drops
        % some time frames.  So only write/average the
        % times we have...
        % wr_ind == write indices
156     wr_ind = round((tr.time(mod_ivl)-tr.time(mod_ivl(1)))/
            proc.dt)+1;
```

```
rd_ind = mod_ivl;    % read indices
Anp1(wr_ind,:) = tr.free.angles( rd_ind,:); % Anp1 ==
    A_{n+1}

%sz_ivl = size(mod_ivl)
%wr_min_max = [wr_ind(1), wr_ind(end)]
%sz_anp1 = size(Anp1)
%sz_avg = size(set_mean(si).angles)
%lengths_cmp = [wr_ind(end), set_mean(si).length]

%formula for incremental average: A_n+1 = A_n + (v_n+1-
    A_n)/n+1
if wr_ind(end) > set_mean(si).length
    % if this trial has more data than the average,
        lengthen the average.
    %fprintf('    >>> expanding An\n');
    set_mean(si).angles((set_mean(si).length+1):wr_ind(
        end),:)=0;
    set_mean(si).sample_count((set_mean(si).length+1):
        wr_ind(end)) = 0;
    M2((set_mean(si).length+1):wr_ind(end),:) = 0;
else
    % is the average is already big enough, but
    % need to expand the current increment.
    %fprintf('    >>> good size. expand Anp1.\n');
    Anp1(end:(set_mean(si).length),:)=0;
end
An = set_mean(si).angles; % A_n
set_mean(si).sample_count(wr_ind) = set_mean(si).
    sample_count(wr_ind) +([wr_ind]>0);
set_mean(si).length = size(An,1);
delta(wr_ind,:) = Anp1(wr_ind,:) - set_mean(si).angles(
    wr_ind,:);
```

```
set_mean(si).angles(wr_ind,:) = An(wr_ind,:) + (Anp1(
    wr_ind,:)-An(wr_ind,:))./repmat(set_mean(si).
    sample_count(wr_ind),1,3);
if max(set_mean(si).sample_count) > 1
    M2(wr_ind,:) = M2(wr_ind,:) + delta(wr_ind,:).*(
        Anp1(wr_ind,:) - set_mean(si).angles(wr_ind,:));
end

%sz_an = size(An)
%sz_avg = size(set_mean(si).angles)
%sz_n = size(set_mean(si).sample_count)
%sz_m2 = size(M2)

% sigma^2
set_mean(si).variance = M2./repmat(set_mean(si).
    sample_count,1,3);
clear Anp1 delta
%fprintf('Averaging finished: trial count: %d \n',
    trial_index);
%fprintf('                    size= %d \n',  size(An,1)
    );

continue;

% DEBUG!
% %===== plot theta, theta_d, theta_dd =====
% mfign=mfign+1; figure(mfign);
% set(mfign, 'Name', id_str);
% subplot(3,1,1);
% plot(tr.time(mod_ivl), tr.free.angles(mod_ivl,2),'k-o
    ', ...
%       tr.time(mod_ivl), tr.th_mdl, 'b-x');
% title('Angular Position (rad)');
% legend('\theta_2, measured','\theta_2, modeled');
```

```
                %%axis(dblims);
211             %% dblims(3:4)=[-100 +300];
                % subplot(3,1,2);
                % plot( tr.time(mod_ivl), tr.thd,'b-x');
                % title('Angular Velocity (rad/s)');
                % legend('\omega_2, model');
216             %%axis(dblims);
                % subplot(3,1,3);
                % plot( tr.time(mod_ivl), tr.thdd(mod_ivl),'b-x');
                % title('Angular Acceleration (rad/s/s)');
                % legend('\alpha_2, model');
221             %%axis(dblims);
            end

            % add this set's average values to the dataset we will
            % give to the curve-fit algorithm.
226         fit_end_index = floor(blocks(block_index).sets(set_index).
                ts*proc.sample_rate);
            set_mean(si).end_index = fit_end_index;
            set_fit_data(1:fit_end_index, set_count) = set_mean(si).
                angles(1:fit_end_index,2);

        end
231 end

% ===== setup curve fit =====
fprintf(' >> Fitting model to measured data....\n');
%coeffs = ones(9,1); % full model (v3)
236 coeffs = ones(7,1); % culled model (v4)
    %from previous runs of this...
%coeffs=[-10.5531   11.0605    0.2629    0.2602    0.1914
    0.0095]';
params.mean = set_mean;
params.mean_count = set_count;
```

```
241

    %set_count
    %should_be = 22*14 % 308
    set_fit_data=reshape(set_fit_data, anly.max_ts_index*anly.
        max_mean_count, 1);
246 %size(set_fit_data)
    %reshape(set_fit_data, max_ts_index, max_set_count)
    %initial_coeffs = coeffs'

    % ===== run the actual curve fit =====
251 % nlinfit uses the Levenberg-Marquardt algorithm [1] for nonlinear
        least squares to compute non-robust fits.
    % : provide theta and mu, but only compare against theta
    [coeffs, residuals, jacobian] = nlinfit(params, set_fit_data,
        @theta_model_fit_aggregate, coeffs);
    resultant_coefficients=coeffs'
    coeff_percent=coeffs'./max(coeffs)*100;
256 J = nansum((residuals).^2);
    fprintf(' J=%10.10f\n', J);
    fprintf(' Jerr=%10.10f\n', J-0.2112795996);
    jacobian

261 return;

    % ===== plot the results ======
    for meani = 1:set_count
        id_str=sprintf('[Mean %d, Block %d, Set %d]', meani, set_mean(
            meani).blockn, set_mean(meani).setn);
266     fprintf('  Plotting %s \n',id_str);
        si = meani;
        block_index = set_mean(meani).blockn;
        set_index = set_mean(meani).setn;
```

```
271         [set_mean(meani).time, set_mean(meani).theta_model_g, set_mean(
                meani).theta_model_all, set_mean(meani).theta_model_opt] =
                ...
                 theta_model( set_mean(si), mats, coeffs);
            set_mean(meani).deviation = sqrt(set_mean(meani).variance);

            % ===== plot thetas comparison =====
276         mfign=mfign+1; figure(mfign); clf(mfign);
            plot_name=sprintf('b%d_s%d_theta', block_index, set_index);
            set(mfign, 'Name', plot_name);
            set(mfign, 'Position', [1000, 400, 900,600]);
            % TODO: plot( set_mean(si).time, set_mean(si).
                theta_model_continuous);
281         hold on; % ======
            fit_mask = 1:set_mean(si).end_index;
            errorbar(set_mean(si).time, set_mean(si).angles(:,2), set_mean(
                si).deviation(:,2), 'b-*');
            plot( set_mean(si).time, set_mean(si).theta_model_g,'k-^');
            plot( set_mean(si).time, set_mean(si).theta_model_all,'k-v');
286         plot( set_mean(si).time(fit_mask), set_mean(si).theta_model_opt
                (fit_mask),'m-');
            ts = blocks(block_index).sets(set_index).ts;
            if block_index == 4
                errorbar(set_mean(si).time, set_mean(si).angles(:,3),
                    set_mean(si).deviation(:,3), 'b-o');
                legend('\theta, measured', '\theta, model, gravity only',
                    ...
291                     '\theta, model, full', '\theta, model, opt', ...
                        '\mu measured', ...
                        'Location','NorthEast');
                axis([0.0, ts, -0.7, 0.9]);
            else
296             legend('\theta, measured', '\theta, model, gravity only',
                    ...
```

```
            '\theta, model, full', '\theta, model, optimal', ...
            'Location','NorthEast');
        if block_index == 1
            axis([0.0, ts, 0, 0.9]);
        elseif block_index == 2
            axis([0.0, ts, 0, 0.9]);
        elseif block_index == 3
            axis([0.0, ts, 0, 1.25]);
        end
    end
    i=0:0.005:max(set_mean(si).time);
    plot( i, 0.*i, 'k.');
    hold off; % ======
            %title([id_str, ' Angular Position (rad)']);
    xlabel('Time from release, (sec)');ylabel('Deflection Angles (
        rad)');

    %% HARDCODED :(
    % ensure all trials across a block have identical limits.
    mfign=mfign+1; figure(mfign); clf(mfign);
    plot_name=sprintf('b%d_s%d_error', block_index, set_index);
    set(mfign, 'Name', plot_name);
    set(mfign, 'Position', [1000, 50, 900,260]);
    err_mask=1:22;
    error_seffpell = abs((set_mean(si).angles(:,2)-set_mean(si).
        theta_model_all));
    error_opt = abs((set_mean(si).angles(fit_mask,2)-set_mean(si).
        theta_model_opt(fit_mask)));
    lsq = nansum((set_mean(si).angles(fit_mask,2)-set_mean(si).
        theta_model_opt(fit_mask)).^2);
    %fprintf('     ?J = %g \n', lsq);
    hold on;
    plot( set_mean(si).time(err_mask), abs(error_seffpell(err_mask)
        ),'k-^');
```

```
326         plot( set_mean(si).time(fit_mask), abs(error_opt(fit_mask)),'m-
                ');
            plot( set_mean(si).time(err_mask), abs(set_mean(si).deviation(
                err_mask,2)), 'b-x');
            plot( i, 0.*i, 'k.');
            hold off;

331         legend('error, \theta full model', ...
                    ...%'error, \theta optimal model', ...
                    '\sigma, \theta measured', ...
                    'Location','NorthWest');
            xlabel('Time from release, (sec)'); ylabel('Bend Angle Error (
                rad)');
336         axis([0.0, 0.21, -0.05, 0.4]);
            tf = blocks(block_index).sets(set_index).t_stop;
            if block_index == 3
                axis([0.0, ts, -0.05, 0.4]);
            else
341             axis([0.0, ts, -0.05, 0.4]);
            end

        end

346 end

    function [time, theta_g, theta_all, theta_opt] = theta_model( mean,
        mats, coeffs)
    % Note: size(theta) is assumed =[n, 3]

351     block_num = mean.blockn;
        set_num = mean.setn;
        pipe_num = mean.pipen;
        ts_num  = mean.tapen;
        mult = mean.multiplicity;
```

```
356     th_exp = mean.angles;

        global g proc anly
        acc_grav = 9800; %mm/s
        pmass = mats.pvc(pipe_num).mass_avg;
361     pradius = 254; %mm = 10"; a rough guess atm.

        pmoi = mats.pvc(pipe_num).MOI; % g/mm2
        time = [(1:size(th_exp,1))*proc.dt]';
        interval = time(1:(end-1)); % == ivl
366
        % hard code the additional weights...
        addl_mass = [0,19,51,83];
        l = 304.8;
        if block_num == 3
371         pmoi = pmoi + addl_mass(set_num)*l^2;
            pmass = pmass + addl_mass(set_num);
        end

        % ====== ====== model the acceleration due to gravity ===== ======
376     % theta double-dot == $\ddot \theta$ == thdd_g = g *mm/s^2* mm/ (g*
            mm^2) = 1/s^2
        % 2D version
        %thdd_g = -pmass*acc_grav/pmoi*pradius*(cos(th_exp(:,2)));
        % 3D version
        thdd_g = -pmass*acc_grav/pmoi*pradius*(cos(th_exp(:,2))).*cos(
            th_exp(:,3));
381
        % ====== ====== model the hinge torque ===== ======
        % theta_double-dot (due to tapespring) = thdd_ts
        nu = mats.ts(ts_num).nu;
        %ky0=mats.ts(ts_num).curvature;
386     E=mats.ts(ts_num).E;
        t=mats.ts(ts_num).thick*mult;
```

```
D = E*t^3/((1 - nu^2)*12);
alpha = mats.ts(ts_num).alpha_rad;
torque_hinge= -(1 - nu)*D*alpha;
thdd_ts = torque_hinge/pmoi;
theta_double_dot_g = thdd_g;
theta_double_dot_all = thdd_g + thdd_ts;

theta_dot_0= 0;
theta_0 = th_exp(1,2);
% ====== ====== integrate just the gravity contribution ===== ======
% [Y] = ode2(ODEFUN, TSPAN, Y0)
% the integrators cheat by simply indexing existing:
% i.o.w. a simple integrator
% data by time ('floor((t)/incr') is this indexing
theta_ddot_g_func=@(t,theta_dot) theta_double_dot_g(floor((t)/proc.dt));
theta_dot_g = [theta_dot_0; ode2(theta_ddot_g_func, interval, theta_dot_0 )];
theta_dot_g_func=@(t,theta) theta_dot_g(floor(((t))/proc.dt)+1);
theta_g = [theta_0; ode2(theta_dot_g_func, interval, theta_0 )];

% ====== ====== integrate the total contribution ===== ======
% [Y] = ode2(ODEFUN, TSPAN, Y0)
% the integrators cheat by simply indexing existing:
% i.o.w. a simple integrator
% data by time ('floor((t)/incr') is this indexing
theta_ddot_all_func=@(t,theta_dot) theta_double_dot_all(floor((t)/proc.dt));
theta_dot_all = [theta_dot_0; ode2(theta_ddot_all_func, interval, theta_dot_0 )];
theta_dot_all_func=@(t,theta) theta_dot_all(floor(((t))/proc.dt)+1);
```

```
    theta_all = [theta_0; ode2(theta_dot_all_func, interval, theta_0 )
        ];

    theta_opt = theta_model_fit_single( coeffs, mean);

    %% DEBUG
    % sz_time = size(time)
    % sz_interval = size(interval)
    % sz_theta = size(th_exp)
    % sz_thdd = size(theta_double_dot_all)
    % sz_thd = size(theta_dot_all)
    % sz_th_opt = size(theta_opt)

end

function theta_predicted = theta_model_fit_aggregate( coeffs, params)
% (function signature is mandated by the nlinfit function)
% This function mediates between the linear fitting toolbox and the
% custom model.

% each theta profile (averaged over a set)  is separate calculated
% and integrated, then passed back to this function, where it is
% combined into one big result vector, and returned to the optimizer.
% The optimizer imposes the single-column vector limit on output.

    global g proc anly

    theta_predicted = NaN(anly.max_ts_index*anly.max_mean_count,1);
    for mi=1:params.mean_count
        wr_slice = [1:params.mean(mi).end_index]+(mi-1)*anly.
            max_ts_index;
        theta_predicted(wr_slice, 1)= theta_model_fit_single(coeffs,
            params.mean(mi));
    end
```

```
446
        % this is an aggregate of all modeled trials
        g.surf = reshape(theta_predicted, anly.max_ts_index, anly.
            max_mean_count);
    end

451 function theta_predicted = theta_model_fit_single (coeffs, set_mean)

        global g proc anly
        %unpack paramaters
        block_num = set_mean.blockn;
456     set_num = set_mean.setn;
        pipe_num =  g.blocks(block_num).sets(set_num).pipe_num;
        ts_num  = g.blocks(block_num).sets(set_num).hinge_num;
        mult = g.blocks(block_num).sets(set_num).multiplicity;
        end_index = set_mean.end_index;
461     theta= set_mean.angles(1:end_index,2);
        mu = set_mean.angles(1:end_index,3);

        % normalize the coeffs:
        weights= ones(length(coeffs),1);
466     %weights(2)= 225.1278 ;
        coeffs = coeffs.*weights;

        acc_grav = 9800; %mm/s
        pmass = g.mats.pvc(pipe_num).mass_avg;
471     pradius = 254; %mm = 10"; a rough guess atm.

        pmoi = g.mats.pvc(pipe_num).MOI; % g/mm2
        time = [(1:size(theta,1))*proc.dt]';
        interval = time(1:(end-1)); % == ivl
476

        % hard code the additional weights...
        addl_mass = [0,19,51,83];
```

```
l = 304.8;
if block_num == 3
    pmoi = pmoi + addl_mass(set_num)*l^2;
    pmass = pmass + addl_mass(set_num);
end

% ====== ====== model the acceleration due to gravity ===== ======
% theta double-dot == $\ddot \theta$ == thdd_g = g *mm/s^2* mm/ (g*
    mm^2) = 1/s^2
% 3D version
thdd_g = -pmass*acc_grav/pmoi*pradius*(cos(theta)).*cos(mu);

% ====== ====== calculate the new hinge torque ===== ======
% theta_double-dot (due to tapespring) = thdd_ts
nu = g.mats.ts(ts_num).nu;
E=g.mats.ts(ts_num).E;
t=g.mats.ts(ts_num).thick*mult;
alpha = g.mats.ts(ts_num).alpha_rad;
a = g.mats.ts(ts_num).arc_length;
D = E*t^3/((1 - nu^2)*12);
c=coeffs;

%% this version works, but is incomplete: (i.e. gimped.)
%torque_hinge_new= -alpha*(c(1)*D + c(2).*theta*D + c(3).*sin(mu)*D
    );

%% Model ver. 3
%% results for this model, on b1,s3:
% resultant_coefficients =
%   -13.3496  225.1278    0.2799    0.2766   -2.4878    0.3872
    -0.8450    0.2614  -12.2594
%torque_hinge_new= -(1-nu)*D*(c(1) + c(2)*alpha + c(3).*theta + c
    (4).*mu ...
```

```
%                           + c(5)*a + c(6)/t^3*a + c(7)/t + c(8)/t^2 +
    c(9)/t^3 );


% % Model ver. 4
torque_hinge_new= -(1-nu)*D*(c(1)+ c(2)*alpha + c(3).*theta + c(4)
    .*mu ...
                                + c(5)*a/t^3  + c(6)/t + c(7)/t^2 );

thdd_model_opt = torque_hinge_new/pmoi;
theta_double_dot_opt = thdd_g + thdd_model_opt;

theta_dot_0= 0;
theta_0 = theta(1);

% ====== ====== integrate the new model ===== ======
% [Y] = ode2(ODEFUN, TSPAN, Y0)
% the integrators cheat by simply indexing existing:
% i.o.w. a simple integrator
% data by time ('floor((t)/incr') is this indexing
theta_ddot_opt_func=@(t,theta_dot) theta_double_dot_opt(floor((t)/
    proc.dt));
theta_dot_opt = ode2(theta_ddot_opt_func, interval, theta_dot_0 );
theta_dot_opt(end+1)=theta_dot_opt(end);
theta_dot_opt_func=@(t,theta) theta_dot_opt(floor(((t))/proc.dt)+1)
    ;
theta_opt = ode2(theta_dot_opt_func, interval, theta_0 );
theta_opt(end+1)=NaN;%theta_opt(end);
theta_predicted=theta_opt;

% % DEBUG
%sz_time = size(time)
%sz_interval = size(interval)
%sz_theta = size(theta)
```

```
        %sz_thdd_new = size(theta_double_dot_opt)
        %sz_thd_new = size(theta_dot_new)
        %sz_th_new = size(theta_new)

    end

    function load_dependencies()
        if ~exist('dcm2q','file')
            addpath('quaternions/');
            fprintf('Adding Quaternion Library to Matlab Path.\n');
        end
        if ~exist('ode2','file')
            addpath('ode/');
            fprintf('Adding Integration Library to Matlab Path.\n');
        end
    end

    function [debug, g, proc, anly]=load_globals()
    %  centralize all global settings.  If a variable isn't needed in
    %  more than one function, it shouldn't be global (or here)
        global g proc anly
        clear g proc anlyq;

        debug = [];

        g=[];

        proc=[];
        proc.sample_rate=100;
        proc.dt = 1/proc.sample_rate;
        proc.default_reference_side=2; %the movement is with respect to
            this side
        proc.initial_value_size=80; % number of indices to average for
            initial value
```

```
        proc.dir='preproc_data/';
571     proc.ext='.mat';
        proc.block_count = 4;

        % displacements larger than this trigger a trial
        proc.trial.start_thresh=20.0;
576     % when the values are less than this (abs) value, (for duration)
            the trial ends.
        proc.trial.end_thresh=20.0;
        % body must stay below the threshold for this many seconds to end a
             trial.
        proc.trial.end_dur=0.8;
        % if a body moves less than this amount,it's classified as a base.
581     proc.choose_base_thresh=7.0;
        % if displ value goes below this number, the trial is reject,
        % because the hinge probably failed.
        proc.trial.reject_low_thresh = [-100, -80, -100];

586     proc.peak.slop=0.006;

        anly.time_mask_start=1;   % start plotting at this time index
        anly.time_mask_stop=inf;  % stop plotting at this time index
        anly.plot_density = 25;   % plot every 'plot_density ' indices of
            data
591     anly.max_mean_count =14;

    end

    function plot_trial(blocks, bnums, snums, tnums)
596     global g proc anly

        time_mask_start=1;   % start plotting at this time index
        time_mask_stop=inf;  % stop plotting at this time index
        plot_density = 25;   % plot every 'plot_density ' indices of data
```

```
601
    if (bnums > proc.block_count)
        fprintf('  !!Block number too big. Unable to plot.\n');
        return;
    elseif snums > blocks(bnums).set_count
606     fprintf('  !!Set number too large. Unable to plot.\n');
        return;
    elseif tnums > blocks(bnums).sets(snums).trial_count
        fprintf('  !!Trial number too large. Unable to plot.\n');
        return;
611 end

    % ==== Plot Deflections (mean-zeroed) =====
    mfign=559; % i.e. Mother FIGure Number
    for block_index = bnums
616     for set_index = snums
            for trial_index = tnums
                tr = blocks(block_index).sets(set_index).trials(
                    trial_index);

                mfign=mfign+1;
621             figure(mfign);
                id_str=sprintf('[Block %d, Set %d, Trial %d]',
                    block_index, set_index, trial_index);
                set(mfign, 'Name', id_str);
                mask=max(time_mask_start,1):min(time_mask_stop, tr.
                    length);
                subplot(1,2,1);
626             plot(tr.time(mask),tr.free.displs(mask,1),'k-x', ...
                     tr.time(mask),tr.free.displs(mask,2),'k-+', ...
                     tr.time(mask),tr.free.displs(mask,3),'k-o', ...
                     tr.time(mask),tr.base.displs(mask,1),'r-x', ...
                     tr.time(mask),tr.base.displs(mask,2),'r-+', ...
631                  tr.time(mask),tr.base.displs(mask,3),'r-o');
```

```
legend('Free Body, u_1', 'Free Body, u_2', 'Free Body,
    u_3', ...
        'Fixed Body, u_1', 'Fixed Body, u_2', 'Fixed
            Body, u_3');
xlabel('time (sec)');
ylabel('mm');
title('Displacements (Inertial, Initial-Zero)');
%text(mean(t(:,1)),min(Pose1(:,1))-30,'X');
%text(mean(t(:,1)),min(Pose1(:,2))-30,'Y');
%text(mean(t(:,1)),min(Pose1(:,3))-30,'Z, Vicon
    Coordinates');

subplot(1,2,2);
plot( tr.time(mask),rad2deg(tr.free.angles(mask,1)),'k-
    x', ...
       tr.time(mask),rad2deg(tr.free.angles(mask,2)),'k
           -+', ...
       tr.time(mask),rad2deg(tr.free.angles(mask,3)),'k-
           o', ...
       tr.time(mask),rad2deg(tr.base.angles(mask,1)),'r-
           x', ...
       tr.time(mask),rad2deg(tr.base.angles(mask,2)),'r
           -+', ...
       tr.time(mask),rad2deg(tr.base.angles(mask,3)),'r-
           o');
legend('Free Body, \theta_1=\gamma','Free Body, \
    theta_2=\theta', ...
        'Free Body, \theta_3=\mu','Fixed Body, \theta_1'
            , ...
        'Fixed Body, \theta_2','Fixed Body, \theta_3');
xlabel('time (sec)');
ylabel('deg');
title('VICON Euler angles (Inertial, Absolute)');
```

```
view_dir=[-26,24];
% ==== Plot Tip Deflections (relative)
% Plot position in inertial frame.
% !! BRILLIANT.  Keep this plot.
mfign=mfign+10;figure(mfign);clf;
trial_num=1;
plot3(tr.base.displs(:,1),tr.base.displs(:,2),tr.base.
    displs(:,3),'b-',...
       tr.free.displs(:,1),tr.free.displs(:,2),tr.free.
           displs(:,3),'r-.');
daspect([1,1,1]); hold on;
legend('Base Body','Free Body', 'Location', 'NorthEast
    ');
xlabel('x');ylabel('y');zlabel('z');
title('Body Displacements in Inertial Frame.');
set(mfign, 'Name', id_str);

% ==== Plot absolute Positions, deflections
% draw the body axis for every "plot_density"th frame
mfign=mfign+1;figure(mfign);clf;
subplot(2,1,1);
daspect([1 1 1]); hold on;
frame_mask = max(time_mask_start,1):plot_density:min(
    time_mask_stop, tr.length);
for frameIndex = frame_mask
    % original lines of code f/Jennings
    %AxisDraw(TForm_2(eye(3),[0,0,0],Pose1(frameIndex
        ,4:6)),Pose1(frameIndex,1:3),50);
    %AxisDraw(TForm_2(eye(3),[0,0,0],Pose2(frameIndex
        ,4:6)),Pose2(frameIndex,1:3),50);

    AxisDraw(q2dcm(tr.free.q(frameIndex,:)')*eye(3),
        ...
```

```
                    tr.free.displs(frameIndex,:)+tr.free.u0
                        ,32);
        AxisDraw(q2dcm(tr.base.q(frameIndex,:)')*eye(3),
            ...
                    tr.base.displs(frameIndex,:)+tr.base.u0
                        ,32);
    end
    legend('i^\wedge','j^\wedge','k^\wedge', 'Location', '
        SouthEast');
    xlabel('x');ylabel('y');zlabel('z');
    hold off;
    view(view_dir);
    title_str=sprintf('Displacement and Orientation (
        Absolute, Inertial Frame)');
    set(mfign, 'Name', id_str);
    title(title_str);

    %mfign=mfign+1;figure(mfign);clf;
    subplot(2,1,2);
    daspect([1,1,1]); hold on;
    xlabel('x_1');ylabel('y_1');zlabel('z_1');
    frame_mask = max(time_mask_start,1):plot_density:min(
        time_mask_stop, tr.length);
    for frameIndex= frame_mask
        % original code:
        %AxisDraw(TForm_1(TForm_2(eye(3),[0,0,0],Pose2(
            frameIndex,4:6)),[0,0,0],Pose1(frameIndex,4:6))
            ,...
        %           Pose2_1(frameIndex,1:3),50);
        AxisDraw(q2dcm(tr.dq(frameIndex,:)')*eye(3), tr.
            free.displs(frameIndex,:), 50);
    end
    view(view_dir);
```

```
                title_str=sprintf('Body_2 in Body_1 frame (Displacement
                    only)');
706             set(mfign, 'Name', id_str);
                title(title_str);

                % checked on combined motion,
                % noticably higher error due to small baseline of
                    markers
711
                % find the deployed body position
                %Pose2_1_base_all(datasetIndex,1:3)=mean(Pose2_1(:,1:3)
                    ,1);
                % disp(mean(Pose2_1(:,1:3),1))
                % pause
716
                drawnow;
            end
        end
    end
721 end

function AxisDraw(DCM,center,Scale)

% the numeric residual actually allows for the inverse to work at
    theta2=90
726 % still should check for special cases at some point

    plot3( Scale*([0,DCM(1,1)])+center(1),Scale*([0,DCM(2,1)])+center
        (2),Scale*([0,DCM(3,1)])+center(3),'r-',...
           Scale*([0,DCM(1,2)])+center(1),Scale*([0,DCM(2,2)])+center
               (2),Scale*([0,DCM(3,2)])+center(3),'g-',...
           Scale*([0,DCM(1,3)])+center(1),Scale*([0,DCM(2,3)])+center
               (2),Scale*([0,DCM(3,3)])+center(3),'b-');
731
```

```
    % originial code
    % AxisDraw=@(DCM,Cent,Scale) ...
%     plot3( Scale*([0,DCM(1,1)])+Cent(1),Scale*([0,DCM(1,2)])+Cent(2),
    Scale*([0,DCM(1,3)])+Cent(3),'r-',...
%           Scale*([0,DCM(2,1)])+Cent(1),Scale*([0,DCM(2,2)])+Cent(2),
    Scale*([0,DCM(2,3)])+Cent(3),'g-',...
%           Scale*([0,DCM(3,1)])+Cent(1),Scale*([0,DCM(3,2)])+Cent(2),
    Scale*([0,DCM(3,3)])+Cent(3),'b-');
end


function [gdata] = xform_b2g( data, offset, angles)
% transform from TForm_2 is for taking euler coordinates into global
    (?)
% body to inertial transform: 1-2-3 (-theta)
% ----
% Data = data to xform?
% offset = vector to displace each entry by offest (additive)
% Theta = rotation angles

    % R_1(theta(1)), R_2(theta(2)), R_3(theta(3)),
    c = cos(angles);
    s = sin(-angles);
    gdata = ([ c(2)*c(3), c(2)*s(3), -s(2) ; ...
              -c(1)*s(3)+s(1)*s(2)*c(3), c(1)*c(3)+s(1)*s(2)*s(3), s
                  (1)*c(2) ; ...
               s(1)*s(3)+c(1)*s(2)*c(3), -s(1)*c(3)+c(1)*s(2)*s(3), c
                  (1)*c(2); ...
             ]*(data'))';

    % original formulation: 1-2-3
    % theta=angles;
    % gdata = (([1,0,0;0,cos(theta(1)),-sin(theta(1));0,sin(theta(1)),
        cos(theta(1))]*...
```

```
        %               [cos(theta(2)),0,sin(theta(2));0,1,0;-sin(theta(2)),0,
            cos(theta(2))]*...
        %               [cos(theta(3)),-sin(theta(3)),0;sin(theta(3)),cos(
            theta(3)),0;0,0,1])*...
761     %           (data')+repmat(offset(:),1,size(data,1))).';
    end

    function [ bdata ] = xform_g2b(data, offset, angles)
    % Reference
766 % inertial to body transform: 3-2-1
    %xform_g2b=(([cos(Theta(3)),--sin(Theta(3)),0;-sin(Theta(3)),cos(Theta
        (3)),0;0,0,1]*...
    %               [cos(Theta(2)),0,---sin(Theta(2));0,1,0;--sin(Theta(2)),0,
        cos(Theta(2))]*...
    %               [1,0,0;0,cos(Theta(1)),--sin(Theta(1));0,-sin(Theta(1)),
        cos(Theta(1))])*...
    %               (Data.')+repmat(X(:),1,size(Data,1))).';
771

        % (body) = R_1(-theta(1))*R_2(-theta(2))*R_3(theta(3)) (inertial)
        c = cos(angles);
        s = sin(-angles);
        dcm = ([ c(2)*c(3), c(2)*s(3), -s(2) ; ...
776                 -c(1)*s(3)+s(1)*s(2)*c(3), c(1)*c(3)+s(1)*s(2)*s(3), s
                        (1)*c(2) ; ...
                    s(1)*s(3)+c(1)*s(2)*c(3), -s(1)*c(3)+c(1)*s(2)*s(3), c
                        (1)*c(2)]);
        bdata = dcm*data;
    end
```

../model/TapeSpringAnalysis_Dynamics.m

```
function blstat(blocks)
% This reads the block information struct, and prints out statistics

    global g proc anly
    [debug, g, proc, anly]=load_globals();

    %mfign=22;

    %plot_trial(blocks, 1, 3, [1 2]);
    % ==== Plot Deflections (mean-zeroed) =====
    for block_index = 1:proc.block_count
        fprintf('    Block %d \n', block_index);
        for set_index = 1:blocks(block_index).set_count
            %blocks(block_index).sets(set_index).
            fprintf('        Set %d: (%d) \n', set_index, blocks(
                block_index).sets(set_index).trial_count);
            %for trial_index = 1:blocks(block_index).sets(set_index).
                trial_count;
                %fprintf('            Trial %d \n', trial_index);
            %end
        end
    end

end

function [debug, g, proc, anly]=load_globals()
%  centralize all global settings.  If a variable isn't needed in
%  more than one function, it shouldn't be global (or here)
    global g proc anly
    clear g proc anlyq;

    debug = [];

    g=[];
```

```
    if ~exist('proc','var')
        proc=[];
        proc.sample_rate=100;
        proc.dt = 1/proc.sample_rate;
        proc.default_reference_side=2; %the movement is with respect to
            this side
        proc.initial_value_size=80; % number of indices to average for
            initial value
        proc.dir='preproc_data/';
        proc.ext='.mat';
        proc.block_count = 4;
    end

end
```

../model/blstat.m

```
function savefigs
% SAVEFIGS       save all existing figures
%    [] = SAVEFIGS();  Saves and writes all existing figures to jpg
%    files for the thesis.
%



    target_dir = '../williams_thesis/Figures/';
    for i=1:1000
        if ishandle(i)
            % write figure
            %fname = sprintf('%s%s', target_dir, get(i, 'Name'));

            plotname = get(i, 'Name');

            figure(i);
            set(i, 'Name', '');
```

```
            title('');

            fname = sprintf('%s%s', target_dir, plotname);
            saveas(i, fname, 'jpg');

            % then close
            close(i);


        end
    end

end
```

../model/savefigs.m

Bibliography

1. Farrand, H., "Rule," U.S. PATENT 1402589, DEC 1922, http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-bool.html&r=7&f=G&l=50&co1=AND&d=PALL&s1=1402589&OS=1402589&RS=1402589.

2. Northrop Grumman, Space Technology, Astro Aerospace, "40-LB LINEAR ACTUATOR BI-STEM," JUL 2004, http://www.as.northropgrumman.com/products/aa_stem/assets/DS-414-40lbBI-STEM.pdf.

3. Northrop Grumman, Space Technology, Astro Aerospace, "Canister AstroMast," http://www.as.northropgrumman.com/products/aa_astromast/assets/DS-303-CannisterAstroMast.pdf, JUL 2004.

4. ATK-Goleta, "Coilable Boom Systems," WWW, DEC 2011, http://www.atk.com/capabilities_multiple/documents/coilable.pdf.

5. Northrop Grumman, Space Technology, Astro Aerospace, "Astromesh Reflector," JUL 2004, http://www.as.northropgrumman.com/products/aa_thuraya/assets/DS-409-AstroMeshReflector.pdf.

6. H.W.Stoll, "Systematic design of Deployable Space Structures," *AIAA Conference on Large Space Platforms: Toward Permanent Manned Occupancy of Space, 2nd*, AIAA, San Diego, CA, Feb 1981, p. 1.

7. G.E.Weeks, "Dynamic Analysis of a Deployable Space Structure," *26th Structures, Structural Dynamics, and Materials Conference*, AIAA, Orlando, FL, April 1985, p. 43.

8. Cassapakis, C. and Thomas, M., "Inflatable Structures Technology Development Overview," *AIAA 1995 Space Programs and Technologies Conference*, AIAA, Huntsville, AL, September 1995, pp. 1–11.

9. Helms, S. K., *Development and Testing of an Inflatable, Rigidizable Space Structure Experiment*, Master's thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, March 2006.

10. Mansfield, E., "Large-deflexion torsion and flexure of initially curved strips," *Proceedings of the Royal Society, London. Series A.*, Vol. 334, 1973, pp. 279–298.

11. Soykasap, O., Pellegrino, S., Howard, P., and Notter, M., "Folding Large Antenna Tape Spring," *Journal of Spacecraft and Rockets*, Vol. 45, No. 3, May-June 2008, pp. 560–567.

12. Seffen, K., You, Z., and Pellegrino, S., "Folding and deployment of curved tape springs," *Internation Journal of Mechanical Sciences*, Vol. 42, 2000, pp. 2055–2073.

13. ISIS Inc., "Deployable UHF and VHF antennas," `http://www.isispace.nl/brochures/ISIS_AntS_Brochure_v.7.11.pdf`, Feb 2012.

14. Seffen, K., "On the Behavior of Folded Tape-Springs," *Journal of Applied Mechanics*, Vol. 68, 2001, pp. 369–375.

15. Walker, S. J. and Aglietti, G., "A Study into the Dynamics of Three Dimensional Tape Spring Folds," *44th AIAA/ASME/ASCE/AHS Structures, Structural Dynamics, and Materials Conference*, AIAA/ASME/ASCE/AHS, Norfolk, VA, April 2003, pp. 1–8.

16. Yee, J. and Pellegrino, S., "Composite Tube Hinges." *Journal of Aerospace Engineering*, Vol. 18, No. Oct, 2005, pp. 224–231.

17. F.Guinot, S.Bourgeois, B.Cochelin, C.Hochard, L.Blanchard, and A.Allezy, "Hybrid Tape-Spring for Deployable Hexapod," *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Palm Springs, California, May 2009, pp. 1–10.

18. Murphey, T. W., "A Material Structural Performance Index for Strain Based Deployable Trusses," *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Palm Springs, California, April 2004, pp. 1–15.

19. Heald, J. C., Montminy, S., Marcoux, E., and Potvin, M.-J., "Ground Characterisation of Tape-spring Deployment Mechanisms," *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Palm Springs, California, May 2009, pp. 1–11.

20. Adams, D. S. and Mobrem, M., "MARSIS Antenna Flight Deployment Anomaly and Resolution," *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Newport, Rhode Island, May 2006, pp. 1–9.

21. Marks, G., Reilly, M., , and Huff, R., "The Lightweight Deployable Antenna for the MARSIS Experiment on the Mars Express Spacecraft," *Proceedings of the 36th Aerospace Mechanisms Symposium*, April 2002, pp. 183–196.

22. Mallikarachi, H. and Pellegrino, S., "Design and Validation of Thin-Walled Composite Deployable Booms with Tape-Spring Hinges," *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Denver, Colorado, April 2011, pp. 1–18.

23. Olson, G. M., Murphey, T., and Thomas, G., "Free Deployment Dynamics of a Z-Folded Solar Array," *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Denver, Colorado, April 2011, pp. 1–13.

24. Seffen, K. and S, P., "Deployment Dynamics of Tapesprings," *Proceedings of the Royal Society of London Series A-Mathematical Physical and Engineering Sciences*, Vol. A, No. 334, March 1999, pp. 1003–1048.

25. Soykasap, O., "Analysis of Tape Spring Hinges," *International Journal of Mechanical Sciences*, Vol. 49, 2006, pp. 853–860.

26. Pollard, E. L. and Murphey, T. W., "Development of Deployable Elastic Composite Shape Memory Alloy Reinforced (DECSMAR) Structures," *47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, AIAA/ASME/ASCE/AHS/ASC, Newport, Rhode Island, May 2006, pp. 1–17.

27. VICON Motion Systems, "Vicon, Los Angeles," `http://www.vicon.com/`, DEC 2011.

28. Jennings, A., Black, J., Allen, C., Pace, T. H., and Trinh, A. T., "Empirically Based Modeling of Tape Spring Hinge Deployment for Space Structures," *AIAA Space 2011 Conference & Exposition*, AIAA, Long Beach, CA, September 2011, pp. 1–17.

29. The Mathworks, Inc, *MATLAB - Documentation*, R2012a ed., MAR 2012, `http://www.mathworks.com/help/techdoc/`.

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704–0188

1. REPORT DATE (DD–MM–YYYY)	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22-03-2012	Master's Thesis	August 2010-March 2012

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Empirical Characterization of Unconstrained Tape Spring Deployment Dynamics	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Daniel McClintic Williams	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENY) 2950 Hobson Way WPAFB OH 45433-7765	8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GSS/ENY/12-M07
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

Satellite cost and mission capability are very sensitive to mass requirements. Mass or volume savings in one component will either lower total spacecraft mass, or provide greater margin for design error in other components. Solar arrays and antennas in particular are often driven by launch vehicle volume constraints instead of mission needs. Deployable structures provide more on-orbit area for both solar arrays and antennas, while occupying less space during launch. These structures therefore, help address these constraints and have been proven as efficient solutions for many years. Tape springs are one example in this category of structures which offer greater reliability for less mass and for a variety of configurations.

This thesis compares a large number of trials on bending tape springs to characterize the behavior with respect to the tape spring geometry (radius, subtended angle, thickness and width) and load conditions (inertia, bend angle and skew angle). These tape spring trials measure representative unconstrained deployment trajectories to provide a representative dataset for comparison against previous and future models. Error trends are identified with respect to bend angle, skew angle, tape spring width, and tape spring thickness. Then an optimized model which characterizes these trends is fitted to the experimental data. This optimized model supports the hinge dependence on bend angle and skew angle, but is inconclusive with respect to the remaining design parameters.

15. SUBJECT TERMS
tape-springs, tape spring, gossamer structures, bistable structures, self-rigidizable

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	167	Dr. Jonathan Black
U	U	U			19b. TELEPHONE NUMBER (Include Area Code) (937)255-3636, ext 4578 Jonathan.black@afit.edu