Numerical analysis of folding and deployment dynamics of thin shell structures with localized folds

Thesis by

Fabricio Canales

In Partial Fulfillment of the Requirements for the

Degree of

Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY

Pasadena, California

2023

Defended May 1, 2023

ACKNOWLEDGEMENTS

I am grateful for my advisor, Professor Sergio Pellegrino, for his support and advice. He kept me on track and focused in relevant research topics. His continuous support helped me reach a happy conclusion to the PhD journey.

I am also thankful for the members of my PhD candidacy and PhD defense committee for their ideas and feedback. Thank you to Dr. Jane Bae, Dr. Dan Meiron, Dr. Peter Schroeder, and Dr. Michael Ortiz.

I also want to thank the Caltech staff for their help with administrative tasks which helped me focus on my research. Thank you to Kate Jackson, Paula Mark, Christine Ramirez and Jamie Meighen-Sei.

I also want to thank my fellow members of the Space Structures Laboratory for their ideas and support, including but not limited to Dr. Harshavardhan Reddy Narravula, Dr. Kanthasamy Ubamanyu and Dr. Michael Marshall.

Finally, I would like to thank my family, friends, and my wife Jacqueline Tawney.

ABSTRACT

This thesis focuses on the analysis of tape springs folded in the opposite sense and their dynamic deployment, and aims to use methods to reduce the computational cost of the analysis. The tape spring is a thin shell deployable structure that has features in common with other deployable structures. The deployment process of such structures can be difficult to predict, and the use of numerical models can be a more cost-effective alternative to experimental testing. Approaches to reduce the computational cost of the analysis of tape springs are investigated such as adaptive meshing and reduced order models. The thesis also presents an accurate analysis of tape spring deployment and a detailed study of the energies and the physics of the deployment. This is used to investigate the energy leak observed in previous tape spring deployment work. Overall, this thesis contributes to improving the efficiency and accuracy of the analysis of deployable structures, particularly tape springs, which can have significant applications in spacecraft technology.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] F. Canales and S. Pellegrino. "High-Fidelity Simulations of Thin-Shell Deployable Structures with Adaptive Meshing". In: *AIAA SCITECH 2022 Forum*. 2022, p. 1269. DOI: 10.2514/6.2022-1269.
F. Canales performed numerical analysis on tape spring opposite-sense bending and applied adaptive meshing techniques.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Overview

Deployable structures

Deployable structures have been used for spacecraft applications such as antennas and solar power. These structures can be packaged in a small volume during launch and deployed in space such that they cover large surface areas. Deployable structure concepts that rely on the release of stored strain energy during deployment do not require motors and have reduced number of components. Concepts of this type have increased packaging efficiency and lower complexity. However, the predictability of the deployment process is decreased in these concepts. Various approaches are used to predict the deployment.

Experimental testing

Gravity and air damping modify the testing conditions of deployable structures with respect to space. Methods used to solve this issue include using drop towers and reduced-gravity flights. Testing in drop towers involves releasing the structure near the top of the tower. The air drag can be reduced by producing a vacuum, although residual air pressure and therefore air drag is unavoidable. This approach requires structures that are either inexpensive or robust enough to survive the fall.

Another option is to use reduced-gravity flights. The deployable structure is contained inside an aircraft which follows a series of parabolic flight paths. By controlling the thrust and drag appropriately, weightlessness is achieved for a portion of the flight. The main issue is that air drag is still present inside the aircraft, which is particularly troublesome for thin structures commonly used for deployable structure concepts.

Experimental testing has a series of disadvantages which make it difficult to reproduce space applications. In addition, it can be very expensive, and performing parametric studies is difficult. The main alternative is the use of numerical models for predicting the deployment process.

Numerical modeling

The use of numerical models for analysis of structures is widespread. In particular, the Finite Element Method (FEM) is commonly used. Different element formulations have been developed such as truss, beam, plate, shell and continuum elements. Shell elements are usually the optimal choice for the analysis of thin deployable structures. Various shell elements are available in commercial FEM software and many more have been proposed in the literature.

Using FEM for the analysis of large complex structures requires using a fine element discretisation. This can result in a very high computational cost, making the analysis unfeasible. Various methods have been developed to improve the efficiency of the numerical analysis. Adaptive meshing involves using a non-uniform element discretisation through the structure. This method has been studied extensively for linear problems, but for nonlinear problems the optimal adaptive meshing approach remains uncertain due to the huge variability of problems. Other approaches involve using reduced order models. This involves using a simplified representation of the original high-fidelity model.

Numerical artifacts are present in the analysis of structures using FEM. The specific details of the element formulation will dictate the potential for undesirable numerical effects. Although these numerical artifacts can be avoided by applying different fixes, some of these corrections result in an even higher computational cost. Therefore, a careful selection of the element formulation can significantly reduce the computational cost by avoiding these numerical artifacts.

1.2 Objective

This work focuses on the analysis of tape springs in opposite-sense bending and dynamic deployment. In addition, the computational cost of the analysis will be reduced.

Tape springs are thin shell deployable structures that have various features common with other more complex deployable structures. Tape springs present regions of localized elastic deformation, denoted as folds. The folds both allow efficient packaging and greatly difficults the numerical analysis. The first area of research is improving the analysis efficiency for the tape spring opposite-sense bending problem. This is a challenging problem due to snap-through buckling.

A second area of research is the accurate analysis of the tape spring dynamic deployment. This would allow an improved understanding of the physics of the deployment, which can often be clouded by the presence of confounding variables such as damping in experimental testing and numerical artifacts in numerical models.

The third area of research is the use of techniques to reduce the cost of the dynamic deployment of tape springs. The physics understanding of the deployment can be used to define simplified models tailored for these specific problems. The physics features are common with other thin shell deployable structures, and thus these simplified models can be extended to more complex structures.

1.3 Layout of dissertation

This dissertation is composed by seven chapters, including this first chapter.

Chapter 2 introduces various examples of deployable structures in the literature. The tape spring is presented and various special features are illustrated. Approaches for finite element modeling are discussed, and previous work on the analysis of tape springs is reviewed.

Chapter 3 introduces the tape spring opposite-sense bending problem and solves it using an in-house finite element code. The in-house code is motivated by the need for careful managing of variables in reduced order models, which is not possible in commercial FEM software.

Chapter 4 presents two approaches for improving the efficiency of the tape spring opposite-sense bending problem. A reduced order modeling approach is presented and applied. Afterwards, a novel adaptive meshing procedure is presented, which is shown to be very effective for the analysis of tape spring bending.

Chapter 5 introduces the tape spring deployment problem and solves it using both the ABAQUS FEM software and the in-house finite element code. Issues with the ABAQUS model are highlighted. Afterwards, it is shown that the element formulation used for the in-house code can avoid these issues due to various special properties.

Chapter 6 presents an analysis of the energy leak in tape spring deployment. Unlike experimental testing, the in-house code can have truly zero damping and therefore it is used to shine light in the energy leak phenomenon. The physics of the deployment are investigated by analyzing the energy components within the tape spring.

Chapter 7 presents two approaches for improving the efficiency of the tape spring deployment problem. An adaptive meshing approach based on the fold position is used. Afterwards, the solution using implicit solvers is explored. A novel locking effect is investigated and a method for alleviating it is detailed.

Chapter 2

BACKGROUND

This chapter is split into three sections. The first section gives an overview of ultralight boom deployable structures. In the second section, the tape spring is introduced as a simple example of an ultralight deployable structure. In the third section, modeling approaches currently used to analyze tape springs are presented.

2.1 Ultralight boom deployable structures

Strain energy deployable booms

Booms that can self-deploy are an attractive option due to their reduced complexity and part count. Strain energy stored within the boom during packaging is often used to drive the deployment. The strain energy can be generated by folding a part of the structure.

An example are thin-walled composite tubes with tape spring hinges [35]. A tape spring is defined as a thin cylindrical shell with an open cross-section. The tape spring hinges are created by cutting a region out from the tube, as shown in Figure 2.1a. The folded configuration is achieved through bending, as shown in Figure 2.1b. A holding mechanism is included to keep the boom packaged during launch

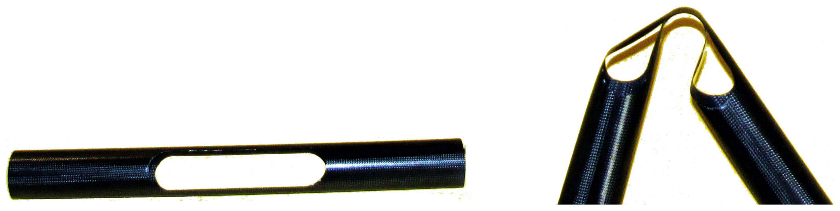


Figure 2.1: Cylindrical boom with a cutout hinge in (a) deployed and (b) partially folded configuration.

and then released in space.

Coilable booms

Instead of forming localized folds, booms can be flattened and wrapped around a hub. Such booms allow for very efficient packaging and thus are attractive for space applications.

Tape springs are very simple coilable booms and hence are used for small scale applications. Bi-stable tape springs have been manufactured using fiber reinforced materials [39]. These tape springs are stable in the fully rolled and fully unrolled configuration. The bi-stability avoids the requirement of a complex containment mechanism for packaging and deployment. A deployment mechanism for such tape springs is shown in Figure 2.2, used in a CubeSat design [28]. One disadvantage of tape springs is that in most cases they do not have equal bending stiffness in different directions, and therefore the ratio of stiffness over mass is not optimal.

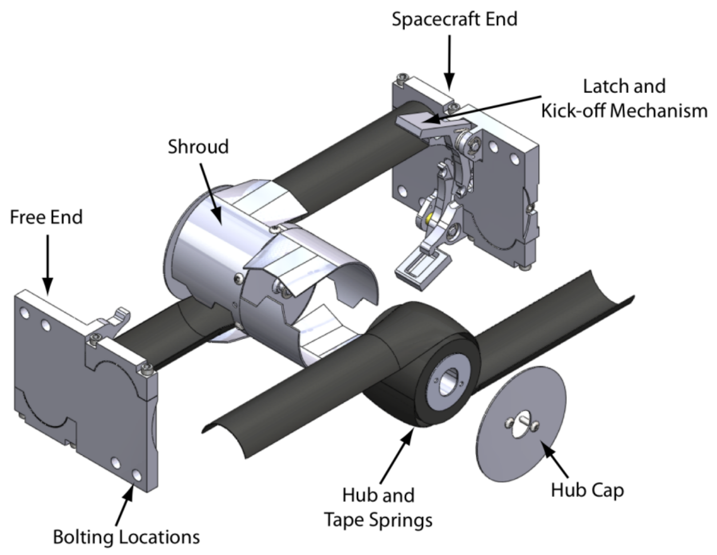


Figure 2.2: Exploded view of deployment mechanism for bi-stable tape springs [28].

A slightly more complex design is the Storable Tubular Extendible Member (STEM) [53]. While a tape springs only covers part of the cylinder formed by its radius of curvature, STEM booms have a cross-section where the angle subtended by the arc of cylinder is greater than 2π radians. This is illustrated in Figure 2.3a. A further modification is the bi-STEM, where two tape springs are overlapped during deployment, as illustrated in Figure 2.3b. This has the advantage of reducing the size of the transition region between the coiled and cylindrical parts. Another variation is the interlocked STEM boom, where matching tabs are formed on the strip edges and a roller mates the tabs during deployment. This is illustrated in Figure 2.3c. This provides a much larger torsional stiffness than the other STEM designs.

Another coilable booms design is the Triangular Rollable and Collapsible (TRAC) boom [38]. The design joins two tape springs along one edge, as shown in Figure 2.4. An advantage over tape springs is that by careful choice of the cross-section

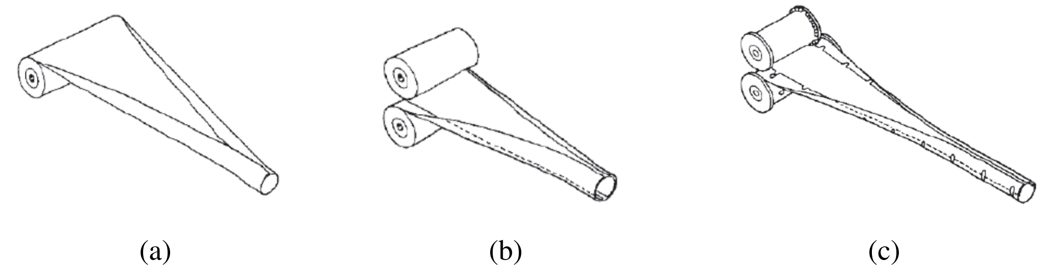


Figure 2.3: Cylindrical boom with a cutout hinge in (a) deployed and (b) partially folded configuration.

geometry parameters, similar bending stiffness in both directions can be achieved, therefore having optimal relative stiffness. These booms have been used in various CubeSat designs due to their low mass and high packaging efficiency.

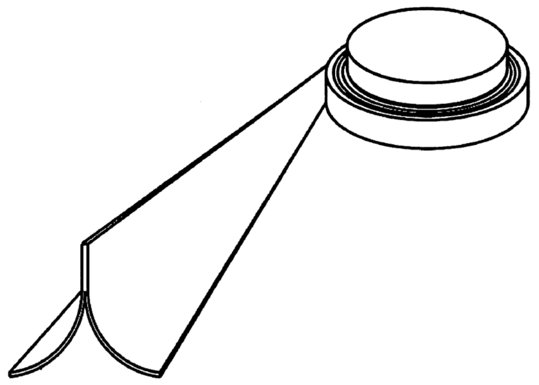


Figure 2.4: TRAC boom.

2.2 Tape springs

As seen in the previous section, tape springs are commonly used as a component of ultralight deployable structures. In this section, a more detailed overview of the tape spring structural response is presented.

By applying bending rotations at the ends, a localized fold appears via snap-through buckling. Therefore, they can store elastic energy during folding and release it when deployed. Many folded configurations can be achieved by creating folds at different locations, offering a wide range of possibilities for deployment. When a tape spring is subjected to opposite-sense bending moments it is deformed as shown in Figure 2.5. Initially the tape spring deforms smoothly as a beam. However, as the moment is increased the deformation becomes localized, and a fold forms at the center. Same-sense bending can also be applied on the tape spring ends. In this case, there is a combined flexion-torsion deformation, complicating the analysis.

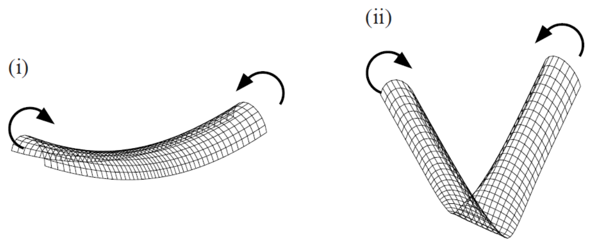


Figure 2.5: Opposite-sense bending [55]. (i) Smooth beam-like bending. (ii) Localized fold formation.

A testing apparatus to study this phenomenon was presented in [55]. One end of the tape spring is fixed longitudinally, while the other end is free to slide. This is illustrated in Figure 2.6.

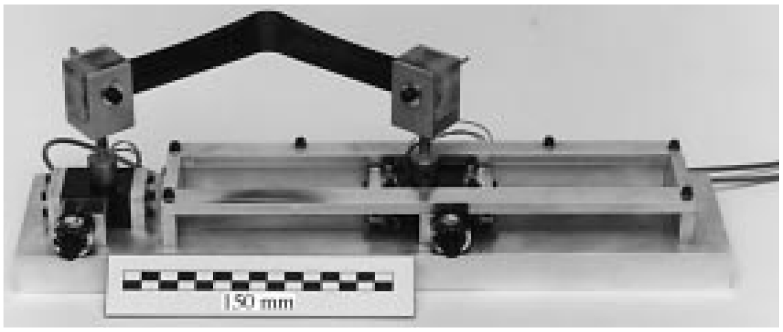


Figure 2.6: Tape spring bending experiment setup [55].

The tape spring opposite-sense bending process can be characterized by plotting the relation between the moment reaction at the ends and the rotation applied at the ends. A schematic diagram of this relationship is shown in Figure 2.7. For small end rotations the moment-rotation relationship is linear. As the rotation is increased, a peak moment, M^{max}, is reached as the cross-section near the center of the tape suddenly flattens. With further rotation increases, the deformation localizes in a small region, and a snap-through behavior occurs between points A and B. If the rotation is increased further, the moment stays almost constant.

The dynamic deployment of tape springs is also a problem of interest. For this purpose, a fold can be formed in the tape spring via opposite-sense bending and held in place. Afterwards, one end of the tape spring is released. Experiments involving the deployment of tape springs have been studied in [55], and frames of the deployment process were taken, as shown in Figure 2.8.

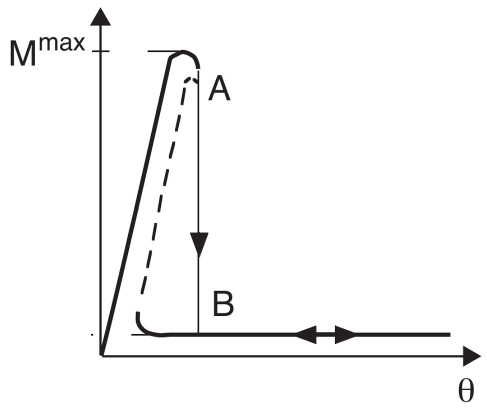


Figure 2.7: Moment-rotation schematic relationship.

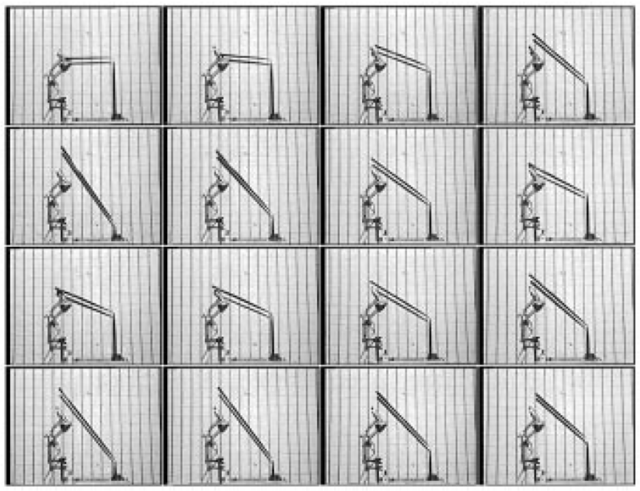


Figure 2.8: Experimental deployment sequence [55].

2.3 Finite element modeling

Element integration

Gaussian quadrature is used to evaluate integrals within the elements numerically. Using the appropriate number of gauss points for exact integration of the polynomial shape functions in linear problems is known as full integration. On the other hand, using fewer gauss points than what would be needed for full integration is known as reduced integration. The choice between integration methods is dependent on the presence of numerical artifacts known as shear locking and hourglassing. Shear locking involves an overly stiff behavior and can observed in elements with full integration under certain conditions [59]. Hourglassing involves an overly flexible behavior where deformation can occur without strain energy, being common in elements with reduced integration [5]. Additionally, using reduced integration is significantly less computationally expensive, and is therefore preferred. It must be noted that for nonlinear problems exact integration is not possible in general, even when using full integration.

The commercial finite element package ABAQUS will be used extensively in the current work. The shell finite elements available include S4 and S4R, which are 4-noded quadrilateral shells. The S4R element uses reduced integration, and can be prone to hourglassing. Therefore, it has different control methods to limit hourglassing [1]. On the other hand, the S4 element uses full integration, and can be prone to shear locking.

Hourglass control

Hourglassing appears due to the presence of zero-energy modes of deformation [1]. For example, consider a first-order element subjected to a bending moment. If reduced integration is used, only a single point in the center is used for integration. Therefore, the element strains and stresses are defined by the dotted lines. The deformed element has a different shape, but the dotted lines remain the same. This causes the strains and stresses at the single integration point to be zero. Therefore the element has deformed without any strain energy being generated.

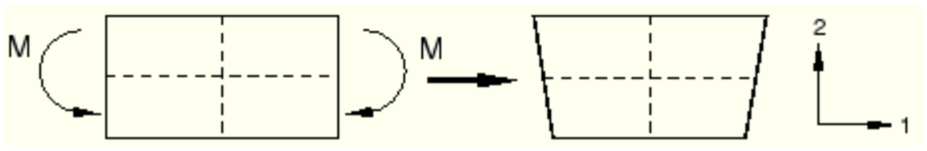


Figure 2.9: Deformation of an element with reduced integration and hourglassing present [1].

Despite the presence of hourglassing, the use of elements with reduced integration is preferred in computational structural mechanics due to the reduced computational cost. Due to this, many approaches to control hourglassing have been proposed. The most common approach is to control hourglassing by identifying the zero-energy or hourglassing modes of the element formulation. Afterwards, artificial nodal forces are applied to counteract the hourglass modes, as first proposed in [22]. This is the approach used in most finite element software, including ABAQUS [1]. For shell elements, the artificial nodal forces used are derived from the presentation given in [4]. Various hourglass control settings within the software specify the definition of the artificial nodal forces. The issue with this approach is that these artificial nodal forces produce work, denoted as artificial energy. If left unchecked, the artificial energy will add an excessive amount of energy to the system, resulting in non-physical results. The ABAQUS manual recommends keeping the ratio between artificial energy and strain energy below 5% [1].

A different approach is modifying the stiffness matrix by hourglass stabilization matrices [5]. This approach will also result in artificial energies being introduced in the system. Therefore, similar issues appear as when artificial nodal forces are introduced.

Implicit and explicit dynamic analyses

There are two main approaches available for dynamic analysis of structures, which differ in the method used for discretising time: implicit and explicit solvers. Implicit solvers allow using relatively large time steps and thus often result in much lower computational cost. However, implicit methods are inefficient with high frequency vibrations, contact and highly nonlinear effects. Explicit methods require very small time steps to be stable, but are efficient for highly nonlinear or impact problems.

Consider the following general structural dynamics equation without damping:

$$\mathbf{M}\ddot{\mathbf{u}}_n + \mathbf{f}^{int}(\mathbf{u}_n) = \mathbf{f}_n \tag{2.1}$$

where $\mathbf{M}$ is the mass matrix, $\mathbf{f}^{int}$ is the vector of internal forces, $\mathbf{f}$ is the vector of external forces, and $(\mathbf{u}_n, \dot{\mathbf{u}}_n, \ddot{\mathbf{u}}_n)$ are the displacement, velocity and acceleration vectors respectively. It is assumed that kinematic quantities at a previous time step $(\mathbf{u}_{n-1}, \dot{\mathbf{u}}_{n-1}, \ddot{\mathbf{u}}_{n-1})$ are known. From (2.1) the acceleration $\ddot{\mathbf{u}}_n$ can be obtained as the solution of a linear system if approximations for $\mathbf{u}_n, \dot{\mathbf{u}}_n$ are defined.

The explicit Newmark method [40], also known as explicit central difference scheme, defines approximations for $\mathbf{u}_n, \dot{\mathbf{u}}_n$ by the following equations:

$$\begin{cases} \dot{\mathbf{u}}_n = \dot{\mathbf{u}}_{n-1} + \dfrac{\Delta t}{2}\left(\ddot{\mathbf{u}}_{n-1} + \ddot{\mathbf{u}}_n\right) \\ \mathbf{u}_n = \mathbf{u}_{n-1} + \Delta t \dot{\mathbf{u}}_{n-1} + \dfrac{\Delta t^2}{2}\ddot{\mathbf{u}}_{n-1} \end{cases} \tag{2.2}$$

Combining (2.1) and (2.2) the structural solution can be advanced in time while requiring only solving a linear system at each time step. The explicit Newmark method is still used extensively, being the explicit solver of choice in the ABAQUS software. Recent advances in explicit solvers for structures have focused on increasing the order of accuracy while preserving stability [51].

The implicit Newmark method defines approximations as follows:

$$\begin{cases} \dot{\mathbf{u}}_n = \dot{\mathbf{u}}_{n-1} + \dfrac{\Delta t}{2}\left(\ddot{\mathbf{u}}_{n-1} + \ddot{\mathbf{u}}_n\right) \\ \mathbf{u}_n = \mathbf{u}_{n-1} + \Delta t \dot{\mathbf{u}}_{n-1} + \dfrac{\Delta t^2}{2}\left(\ddot{\mathbf{u}}_{n-1} + \ddot{\mathbf{u}}_n\right) \end{cases} \tag{2.3}$$

Combining (2.1) and (2.2) results in a nonlinear equation for $\ddot{\mathbf{u}}_n$ due to the nonlinearity introduced by the function $\mathbf{f}^{int}$. Therefore iterations are required to obtain the solution at every time step.

Implicit solvers can use larger time steps while preserving stability of the solution. However, using large time steps results in not being able to resolve high-frequency vibrations present in the structure, depending on the Nyquist frequency corresponding to the time step used. These vibrations may be an artifact of the finite element discretisation and thus it is desirable to dampen high-frequency oscillations in implicit solvers [31]. Many implicit methods have been proposed for this purpose, such as the Newmark-β method [40], the $HHT - \alpha$ method [24] and the $CH - \alpha$ method [14]. These methods can be considered particular cases of a more general formulation known as generalized$-\alpha$ method [20]. This method controls the dissipation via the spectral radius at infinity, allowing control on the dissipation at the high-frequency range. ABAQUS currently uses the $HHT - \alpha$ method as the implicit solver.

Shell element formulations

Classical shell theories can be classified depending on the thickness of the shell. The Kirchhoff-Love shell theory is accurate for thin shells, while the Reissner-Mindlin shell theory is more appropriate for thick shells. However, there are many more differences between these shell theories. To illustrate these differences, we define a three-dimensional shell body using a curvilinear coordinate system, as shown in Figure (2.10). The in-plane coordinates are ξ_α, where $\alpha = (1, 2)$. The shell thickness direction is ξ_3. The position vector $\mathbf{X}$ is defined as a function of the midsurface position $\mathbf{R}$ and the director vector $\mathbf{A}_3$ as [19]:

$$\mathbf{X}(\xi_1, \xi_2, \xi_3) = \mathbf{R}(\xi_1, \xi_2) + \xi_3 \mathbf{A}_3(\xi_1, \xi_2) \tag{2.4}$$

The Kirchhoff-Love shell theory considers that the director $\mathbf{A}_3$ is always perpendicular to the mid-surface. Therefore, the shell body can be described uniquely by the midsurface position $\mathbf{R}$, and only displacement variables are used. Moreover, since the director always follows the midsurface, the out-of-plane shear strains are

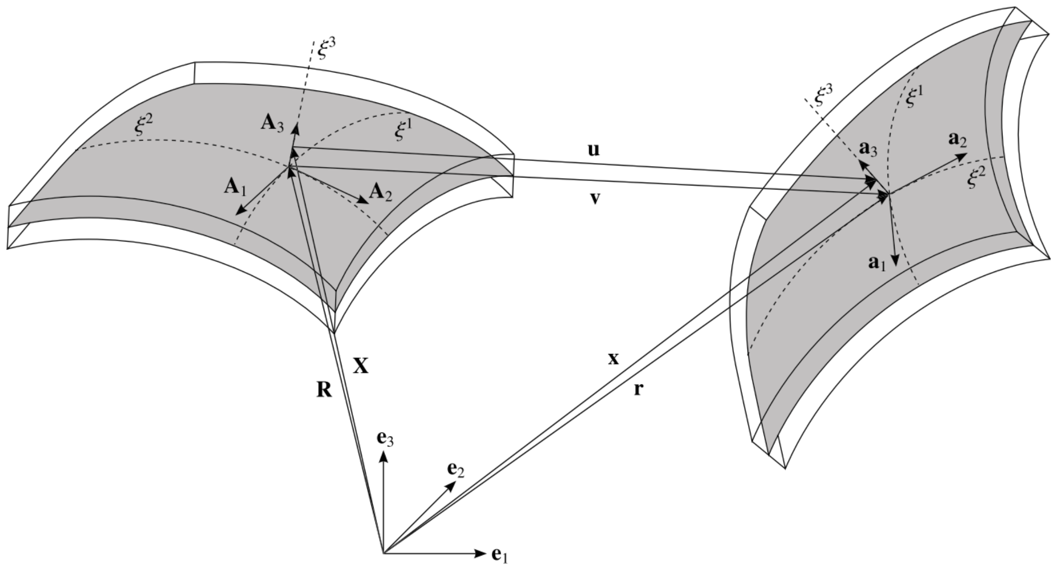


Figure 2.10: Reference and current configurations of shell body [42].

not present in the formulation. This has the advantage of eliminating shear locking at the formulation level. However, displacement-based shell finite elements using the Kirchhoff-Love shell theory require shape functions with C^1 continuity between elements. This has been proven to be very difficult to formulate using standard polynomial shape functions [61].

The Reissner-Mindlin shell theory considers that an additional rotation can be imposed in the director vector $\mathbf{A}_3$ such that it is not always perpendicular to the mid-surface. Therefore, this theory uses additional rotational degrees of freedom. This theory only requires C^0 continuity between elements and therefore they are compatible with standard lagrangian shape functions. Due to this, the Reissner-Mindlin shell theory is used routinely in commercial finite element software. However, the inclusion of rotational degrees of freedom has many drawbacks, including the presence of shear locking, complex large-rotation transformations and rotational inertias scaling in dynamic simulations. In addition, these additional degrees of freedom increase the computational cost with little benefit for thin shell analysis.

2.4 Tape spring analysis

The analysis of tape spring folding and deployment remains challenging due to the presence of different instabilities. The folds form suddenly, can move along the tape spring, split into different folds, or merge into a single fold. Numerical effects such as locking and hourglassing can distort the accuracy of the solution and prevent getting a true understanding of the phenomenon. Experiments can be similarly challenging due to the sensitivity of the structural behavior to instabilities.

Various simplified models for prediction of tape spring bending have been developed. A planar rod model has been presented in [7]. It was assumed that the cross-section remains circular, but with potentially different radii of curvature along the rod length. The model is based on 4 variables at each point of the rod: the 2 coordinates that define the position within a plane, the rotation of the cross-section and a variable related to the curvature of the cross-section. This model was further improved in [45] and extended for 3D motion in [44].

An experimental analysis of tape spring deployment is presented in [55]. The experimentally obtained fold position and angle are identified using a normalized fold position λ with the tape length L. Additionally, a fold angle θ is defined, as shown in Figure 2.11. A typical plot of the fold position λ and angle θ is shown in

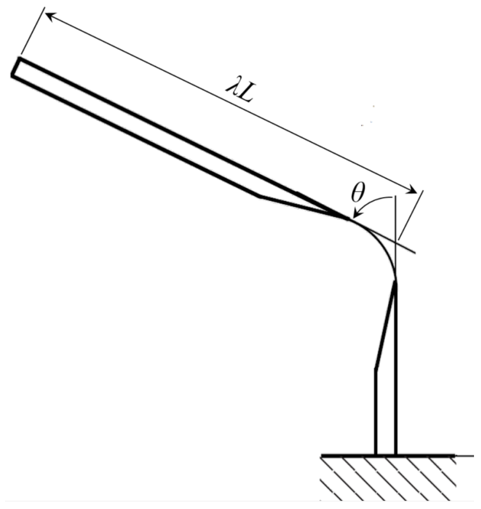


Figure 2.11: Fold geometry description.

Figure 2.12. Initially λ increases because the fold moves towards the support, while the fold angle θ simultaneously reduces as the tape spring starts to straighten. As the fold gets closer to the support, eventually there is a reversal in the fold velocity direction which occurs around $t = 0.12$ s. The fold then moves back through the tape spring while the fold angle begins to increase. However, the tape spring does not return to the original state.

2.5 Open-source finite element software

Although commercial finite element software is easy to use in predefined cases, customization is limited and integration with external tools can be very difficult. On the other hand, an open-source software has source code freely available and

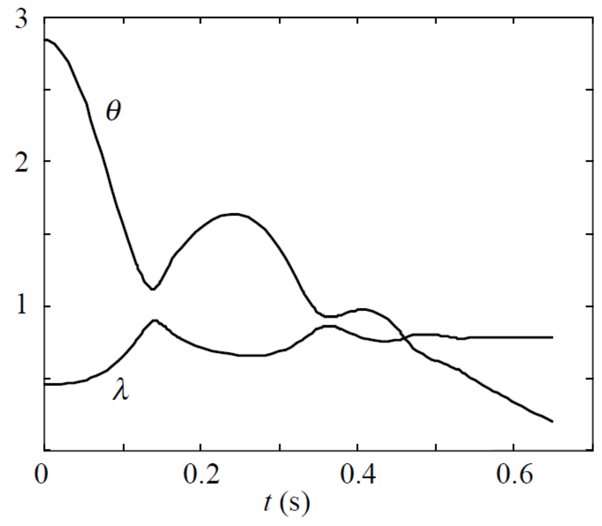


Figure 2.12: Typical tape spring deployment plots for fold position λ and angle θ [55].

can be modified according to the needs of the user. This makes them attractive for specific applications where precise control and modifications of the solution process is required, such as in reduced order modeling. There are currently many open-source software available for finite element analysis. In this section, a brief overview of current open-source codes for finite element analysis is given.

Calculix is an open-source package for solution of problems via the finite element method. It uses an input format similar to ABAQUS, and therefore the input data is compatible. Python language can be used for automating the creation of models. Although 4-node and 8-node shell elements are available, during calculations they are expanded into three-dimensional 8-node and 20-node brick elements respectively [11]. This expansion does not always result in correct answers, such as when rotational boundary conditions are applied.

FEniCS Project is a collection of software components aimed at the solution of partial differential equations. It was initiated in 2003 by a joint collaboration between the University of Chicago and Chalmers University of Technology. It uses Python language to easily express high-level finite element procedures and uses C++ for assembly and solution. Various external libraries are used for numerical linear algebra, meshing and distributed computing. It includes Von Karman shallow shell elements, where some higher order strain terms are neglected by assuming moderate rotations, and Naghdi shell elements. Reduced integration is available for the elements.

Elmer is a software for multiphysics problems developed by the CSC - IT Center for Science, which is owned by the Finnish state. It is written using C, C++ and Fortran90 languages. Although shell elements are available via an additional file, there is no documentation.

OpenSees is a software developed via sponsorship of the Pacific Earthquake Engineering Research Center, which is a research center located at the University of California, Berkeley. It is primarily written in C++ with various numerical libraries in Fortran. It includes MITC4 shell elements, where the strain components are interpolated to alleviate shear locking, and generalized conforming shell elements, where the conforming conditions are not applied exactly but in an integral sense.

Chapter 3

TAPE SPRING OPPOSITE-SENSE BENDING - SIMULATION

This chapter is concerned with developing an in-house code for the simulation of the tape spring opposite-sense bending. The physical details of the physical problem to simulate are introduced. This problem has been studied before in commercial finite element software [49], as well as with in-house code implementations of various finite element formulations [58]. Afterwards, a shell finite element formulation is presented following [30]. The in-house code allows the reduced order modeling methods used in the next chapter, which require manipulation of variables and precise control of the solution methods and is often not possible in commercial finite element software. The details of the finite element formulation used and comparison with the ABAQUS software are presented.

3.1 Problem description

A tape spring can be described as a cylindrical shell of length L, thickness t, radius of curvature R and angle subtended α. The XY plane of the coordinate system is parallel to the cross-section, where the X axis is parallel to the chords connecting the ends of the circular cross-section, as shown in Figure 3.1. The Z axis is the longitudinal axis of the tape spring. The center of the coordinate system is the center of the circular arc in the cross-section on one end of the tape spring.

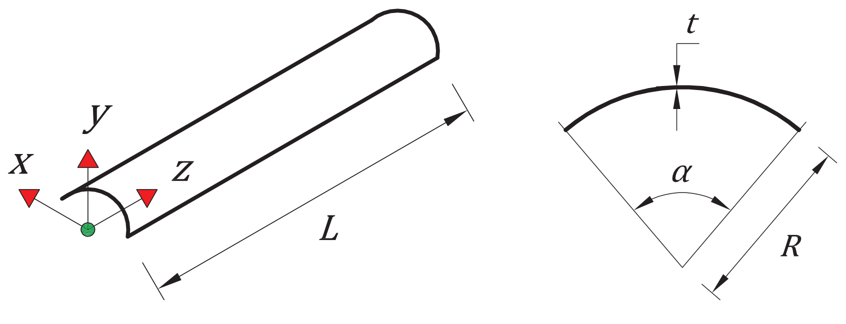


Figure 3.1: (a) Tape spring geometry, where the origin is represented by the green point. (b) Tape spring cross-section.

Opposite-sense bending is applied by rotating both ends of the tape spring around an axis parallel to the X axis. For this purpose, we identify the two cross-sections where boundary conditions will be applied in (3.2): The top and bottom cross-section are

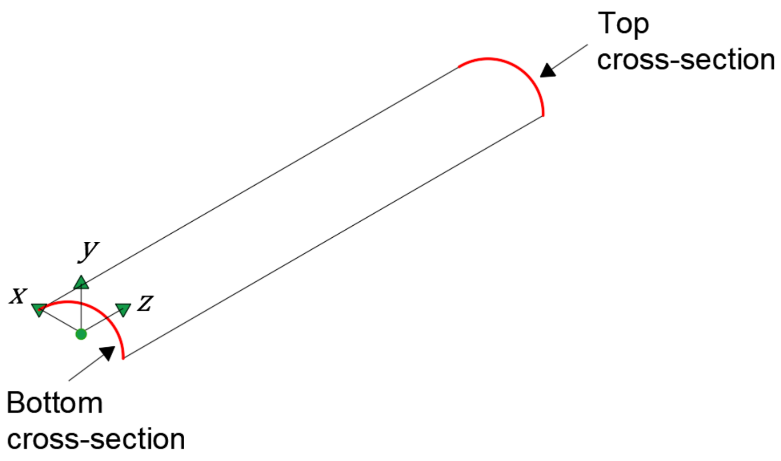


Figure 3.2: Boundary condition regions for opposite-sense bending of tape spring.

rotated in opposite directions with respect to the X axis. The bottom cross-section is completely fixed in translation, while the top cross-section can move longitudinally (along the Z axis) but is otherwise fixed in translation. The boundary conditions are indicated in (3.1), where nodes contained in a plane are indicated by just listing a single coordinate (plane equation).

Table 3.1: Boundary conditions for opposite-sense bending of tape spring.

Region	Location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = \phi_y = \phi_z = 0\ ,\ \phi_x = \theta$
Top cross-section	$z = L$	$u_x = u_y = \phi_y = \phi_z = 0\ ,\ \phi_x = -\theta$

.

3.2 Arc length method

As described in Chapter 2, there is a sudden localization and a snap-through when opposite-sense rotations are applied to a tape spring. This requires a special incremental approach in order to be able to trace the stable and unstable equilibrium paths. For this purpose, the Riks method can be used [52]. Continuation through limit and bifurcation points is possible by using the length of the equilibrium path as a control parameter. This requires adding to the standard equilibrium equations a constraint equation, which destroys the symmetric banded nature of the finite element equations. A modified Riks approach was presented in [15] which preserves the symmetric banded form of the matrices used while maintaining the capabilities of the Riks method. This will be the method used in the code implementation.

A brief description of the Riks method follows. We consider here a general quasi-static structural problem with geometrical nonlinearities but within the elastic regime. After discretisation, the problem is defined by the following nonlinear equation:

$$\mathbf{r}\left[\mathbf{u}\right] = \mathbf{f}\left[\mathbf{u}\right] - \mathbf{f}_{ext} = \mathbf{0} \tag{3.1}$$

where $\mathbf{r}$ is the residual vector, $\mathbf{u} \in \mathbb{R}^N$ is the displacement vector, $\mathbf{f} \in \mathbb{R}^N$ is the vector of internal forces within the structures and $\mathbf{f}_{ext} \in \mathbb{R}^N$ is the vector of external forces. The displacement vector $\mathbf{u}$ is the unknown in this equation. For the tape spring snap-through buckling problem there are no external forces, so $\mathbf{f}_{ext} = 0$. However, the rotations θ and $-\theta$ imposed at the tape spring ends act as the loading, such that the equation (3.1) is modified as follows:

$$\mathbf{r}\left[\mathbf{u}, \theta\right] = \mathbf{f}\left[\mathbf{u}, \theta\right] = \mathbf{0} \tag{3.2}$$

It is possible to trace an equilibrium path by progressively incrementing θ and solving the resulting nonlinear equation. However, this approach does not allow tracing the complete stable and unstable equilibrium paths. To see this, an example of the equilibrium path desired is shown in Figure 3.3, obtained for a tape spring with parameters $L = 300$ mm, $t = 0.1$ mm, $\alpha = 110°$, $R = 10$ mm, Young's modulus $E = 131$ GPa and Poisson's ratio $\nu = 0.3$. The plot contains both the stable and unstable equilibrium paths. If we choose to increment the rotation θ monotonically, we only trace part of this path, as shown in Figure 3.4. To trace both the stable and

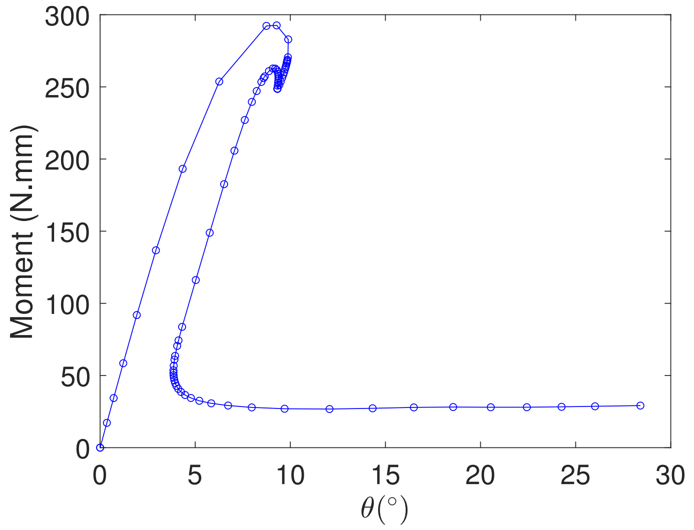


Figure 3.3: Stable and unstable equilibrium path for opposite-sense bending of tape spring obtained by the arc-length method.

unstable equilibrium paths it is required to simultaneously modify $\mathbf{u}$ and θ, such that both of them become variables of the problem. Given an initial state $\mathbf{u}_0$ and θ_0, the

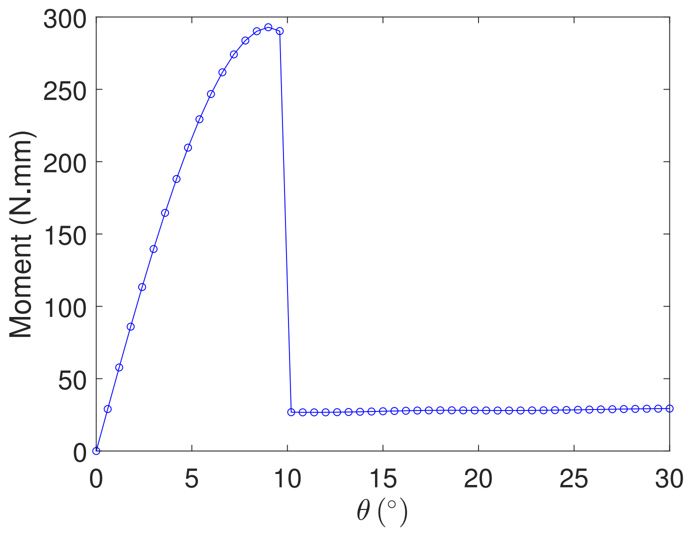


Figure 3.4: Part of equilibrium path for opposite-sense bending of tape spring.

variables are the increments $\Delta\mathbf{u}$ and $\Delta\theta$. The system is closed by constraining the total combined increment and adding this condition to (3.2):

$$\begin{cases} \mathbf{r}\left[\mathbf{u},\theta\right] = \mathbf{f}\left[\mathbf{u},\theta\right] = \mathbf{0} \\ \mathbf{u} = \mathbf{u}_0 + \Delta\mathbf{u} \\ \theta = \theta_0 + \Delta\theta \\ \|\Delta\mathbf{u}\|^2 + \lambda\left(\Delta\theta\right)^2 = \Delta s^2 \end{cases} \tag{3.3}$$

where Δs defines the step size taken during every increment and is chosen such that the equilibrium path is traced accurately, and λ defines a scaling factor between $\Delta\mathbf{u}$ and $\Delta\theta$.

The value of Δs can be chosen adaptively such that the equilibrium path is traced efficiently, resulting in the adaptive arc length method. The adaptive arc length method was used in Figure 3.3, where the value of the increment size Δs is reduced as the equilibrium path becomes more nonlinear. Additional details regarding the solution of this system of equations are given in [15]. This method is used in the remainder of this chapter to model the opposite-sense bending of the tape spring.

3.3 Isogeometric shell finite element

Isogeometric finite elements [27] use the same shape functions as the geometries defined in computer-aided design software, that is, Non-Uniform Rational B-Spline (NURBS) shape functions. These functions can model certain geometric features exactly, such as circular arcs. This makes them attractive for the analysis of elastic folds, which are almost (but not completely) cylinders with circular arc cross-section.

In this section, details from an in-house isogeometric shell finite element code are presented.

NURBS basis functions for surfaces

A brief overview of the bidimensional NURBS basis functions used in the finite element procedure and the B-spline functions which are used as building blocks is presented here. These functions are significantly different than the standard lagrangian shape functions used in commercial finite element software. More details are given in [27].

Univariate B-spline basis functions of degree p are constructed by defining a non-decreasing sequence of numerical values $\left\{\xi_1, ..., \xi_{n+p+1}\right\}$, each of which is called a knot. There are n basis function associated with this knot sequence, which can be computed using the Cox-de-Boor recursion formula:

$$N_{i,p}\left(\xi\right) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{3.4}$$

For the case $p = 0$ a piece-wise constant basis function is defined:

$$N_{i,0}\left(\xi\right) = \begin{cases} 1, & \text{if} \quad \xi_i \leq \xi < \xi_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{3.5}$$

Bidimensional B-spline basis functions are obtained as a tensor product of sets of univariate B-spline functions. On each dimension we use the variables $\left(\xi^1, \xi^2\right)$ respectively, the degrees of the functions $\left(p^1, p^2\right)$ and $\left(n^1, n^2\right)$ basis functions. The bidimensional B-spline basis functions are defined as:

$$B_{i,\mathbf{p}}\left(\boldsymbol{\xi}\right) = N_{i^1,p^1}\left(\xi^1\right) N_{i^2,p^2}\left(\xi^2\right) \tag{3.6}$$

where $i = i^1 + n^1(i^2 - 1)$, $\mathbf{p} = \left\{p^1, p^2\right\}$, $\boldsymbol{\xi} = \left(\xi^1, \xi^2\right)$ and $i^k = \left\{1, ..., n^k\right\}$. Bidimensional NURBS basis functions are defined as:

$$R_{i,\mathbf{p}}\left(\boldsymbol{\xi}\right) = \frac{w_i B_{i,\mathbf{p}}\left(\boldsymbol{\xi}\right)}{\sum_{i=1}^{n} w_i B_{i,\mathbf{p}}\left(\boldsymbol{\xi}\right)} \tag{3.7}$$

where $n = n^1 n^2$ and w_i are specified weights. NURBS sufaces are defined using the bidimensional NURBS basis functions as follows:

$$\mathbf{S}\left(\boldsymbol{\xi}\right) = \sum_{i=1}^{n} \boldsymbol{P}_i R_{i,\mathbf{p}}\left(\boldsymbol{\xi}\right) \tag{3.8}$$

where $\boldsymbol{P}_i \in R^3$ are control points which form the surface mesh. An example of a surface and corresponding control points is shown in 3.5. By carefully choosing the control points $\boldsymbol{P}_i$ and the weights w_i, various surfaces can be obtained. It must be noted that the control points $\mathbf{P}_i$ are not located on the surface $\mathbf{S}$ in general.

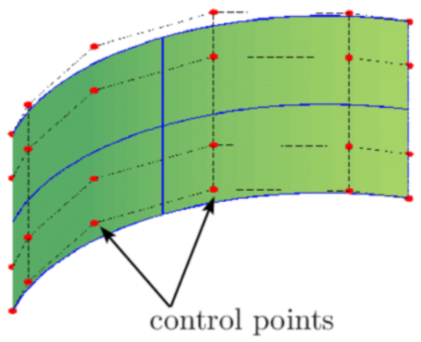


Figure 3.5: Surface mesh and corresponding control points.

Isogeometric concept

The isogeometric concept uses the NURBS basis functions for interpolation within finite elements. The knots define the finite element boundaries (not including the repeated knots). The control point coordinates $\mathbf{P}_i$ become the displacement variables, while the weights w_i remain fixed. To illustrate the isogeometric concept, the quadratic B-spline basis functions for a knot vector $[0, 0, 0, 1, 2, 3, 3, 3]$ are shown in Figure 3.6. For comparison, the quadratic lagrangian basis functions, used in standard finite elements, are shown in Figure 3.7. Some special features of the B-spline basis functions, and by extension of the NURBS basis functions [46], are as follows:

- They can have non-zero values over multiple elements
- They have smooth derivatives over multiple elements
- They are non-interpolatory at the element boundaries

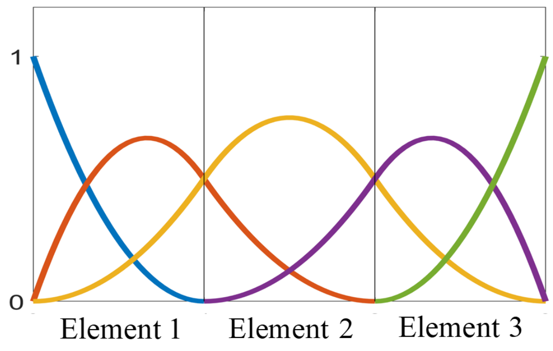


Figure 3.6: Quadratic B-spline basis functions.

Additionally, the continuity at the element boundaries can be controlled by adding repeated knots. In this manner, arbitrary degrees of continuity can be obtained.

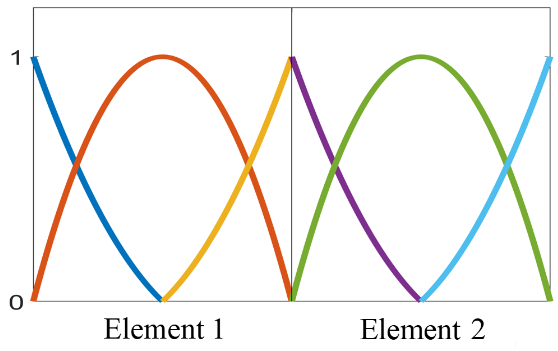


Figure 3.7: Quadratic lagrangian basis functions.

Shell formulation

An isogeometric shell element using Kirchhoff-love shell theory and C^1 NURBS shape functions has been presented in [30]. This element uses only displacement variables and therefore avoids the previously mentioned drawbacks of the Reissner-Mindlin shell theory. The C^1 continuity requirement is obtained via the NURBS shape functions. The shell three dimensional continuum is reduced to the midsurface of the shell. Transverse normal stresses and transverse shear strains are neglected, following the assumptions of the Kirchhoff-Love theory. The strain is linear through the thickness, with a constant part due to membrane effects and a linear part due to bending. The NURBS shape functions are used within the finite element method via standard procedures [3] to develop the shell element.

Rotational boundary conditions

The absence of rotational variables in the Kirchhoff-Love shell theory has many advantages. However, the application of rotational boundary conditions requires special treatment. To do so, we use the special properties of the NURBS curves and surfaces.

Although a NURBS curve in general does not pass through the control points, it can be shown that the first and last control point of the curve are interpolated exactly. Additionally, the tangents to a NURBS curve at the start and end of the curve are parallel to the lines that connect the two control points closest to the start and end respectively. In an analogous manner, the control points in the boundary of the NURBS surface are interpolated exactly. Also, the tangent to a NURBS surface at a location in the boundary of the surface is parallel to the line connecting the two points closest to the boundary location.

Therefore, to apply the clamped boundary condition at a boundary of a NURBS surface we fix the two rows of control points next to the boundary. Fixing the first row ensures that the boundary does not move (since the control points in the first row are exactly interpolated). Fixing the second row ensures that the boundary does not rotate (since the slope of the boundary is defined by the line connecting the control points of the two rows adjacent to the boundary).

To apply a non-zero rotation angle on a boundary, displacements are applied to the control points along the boundary. The rotation axis is defined as the geometric center of the boundary. With the rotation axis and angle defined, the position of the rotated boundary can be calculated and the control points are displaced appropriately.

Implementation

A geometrically nonlinear shell finite element code using the isogeometric elements previously described has been implemented in MATLAB. The code relies extensively on efficient vectorized algorithms [16] which are optimal for implementation in vector languages. This allows the implementation to be competitive with other coding language choices. Additionally, the computations leverage the GPU capabilities to obtain a fast execution time. The computer used to run the code in the numerical results of this thesis is a laptop with Intel i7-8750H CPU processor, 16 GB RAM and a GeForce GTX 1070 GPU.

3.4 Numerical results

The tape spring opposite-sense bending problem is used to verify the isogeometric shell code implementation. Quadratic degree NURBS functions in both directions are used for interpolation within each element. The tape spring considered has geometrical parameters $L = 300$ mm, $t = 0.1$ mm, $\alpha = 110°$ and $R = 10$ mm. An isotropic material is considered, with Young's modulus $E = 128.7$ GPa and Poisson's ratio $\nu = 0.3$. Results from ABAQUS finite element software are used to verify the implementation. In both the isogeometric shell code and ABAQUS the arc-length method is used to trace the complete equilibrium path accurately. The mesh used in ABAQUS is made of 4000 S4 elements, with 200 elements along the length and 20 elements along the arc cross-section.

Figure 3.8 shows the comparison between the isogeometric shell code and ABAQUS software for the tape spring opposite-sense bending. The moment reaction at the tape spring ends is computed by force balance for the isogeometric shell code. It can be observed that there is good agreement between both models, validating the shell code implementation.

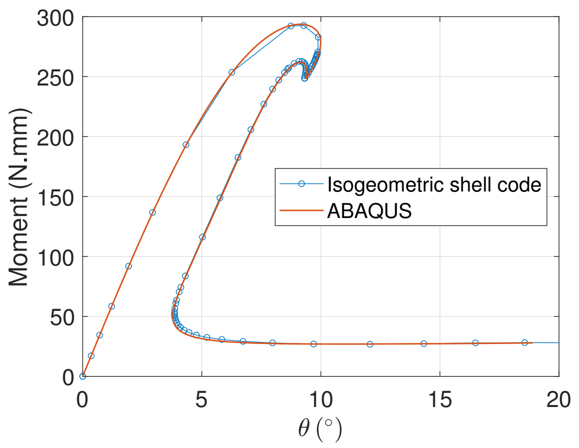


Figure 3.8: Moment-rotation plots for opposite-sense bending of tape springs for isogeometric shell code and ABAQUS.

3.5 Higher order NURBS functions

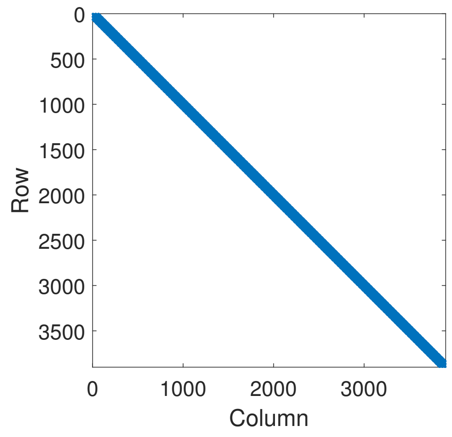
Using higher order NURBS functions increases the accuracy for a given number of degrees of freedom [6]. However, using higher-order NURBS functions in practice results in a high computational cost. Although reducing the number of degrees of freedom reduces the size of the linear system formed from the discretisation, the matrix is less sparse when higher-order functions are used. This increases the cost of solving the linear system. Additionally, the cost of assembling the matrix increases significantly due to the higher number of gauss points required.

To illustrate this point, the sparsity pattern of the stiffness matrix created by using quadratic NURBS functions and quartic NURBS functions with two different meshes will be shown. Table 3.2 details the number of finite elements and the degrees of freedom of the two meshes, as well as the density and the non-zero elements of the stiffness matrix for both degrees used.

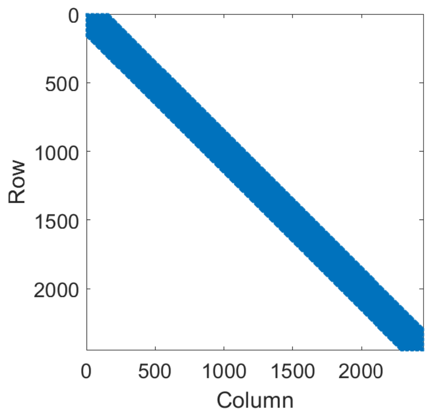
Table 3.2: Mesh and stiffness matrix parameters for different NURBS degrees.

Parameter	Quadratic NURBS	Quartic NURBS
Elements	1024	512
Degrees of freedom	3900	2448
Matrix density	0.017‰	0.078‰
Non-zero elements	255004	468859

It can be observed that despite having fewer degrees of freedom, the matrix built using quartic NURBS functions has more non-zero elements. The sparsity patterns for both cases are shown in Figure 3.9, where it can be seen that the matrix formed using quartic NURBS functions is significantly less sparse. This not only increases the computational cost of solving the linear system but also the cost of assembling the system.



(a)



(b)

Figure 3.9: Sparsity pattern of matrices (a) Using quadratic NURBS functions (b) Using quartic NURBS functions.

Although this analysis would suggest that using lower degrees is more efficient, there is a lower limit to the permissible NURBS degree. The Kirchhoff shell formulation computes curvatures via second derivatives of the displacements. Therefore, at least a quadratic degree is required, and this degree is used for the analysis of the tape spring opposite-sense bending.

3.6 Discussion

The opposite-sense bending of tape spring problem has been introduced. The presence of stable and unstable equilibrium paths is illustrated, and the necessity for the arc-length method is detailed. Afterwards, the isogeometric shell finite element formulation is presented. Various interesting properties of the isogeometric basis functions are shown. The implementation of the isogeometric shell code has been detailed.

Numerical results have been developed to validate the isogeometric shell code using the opposite-sense bending of tape springs as a test case. Results from the ABAQUS software are used as reference. The comparison indicates that there is good agreement between both models and the shell code implementation has been validated.

Chapter 4

TAPE SPRING OPPOSITE-SENSE BENDING - SPEEDUP

This chapter is concerned with the efficient analysis of the tape spring snap-through buckling problem by reducing the computational cost. It is of interest to obtain this speedup within a single simulation procedure. Therefore, no data from previous simulations, also known as off-line simulations, is used. This single-simulation approach is known as an on-the-fly procedure. The use of reduced order modeling techniques and adaptive meshing within an on-the-fly procedure will be studied. Since these methods require manipulation of variables often not available directly in commercial finite element software, an in-house finite element code will be developed. The details of the finite element formulation used are described below.

4.1 Reduced order modeling

In this section we illustrate the construction of a reduced order basis via the method of snapshots as presented in [57]. Afterwards, the state-of-the-art energy-conserving and weight sampling hyper-reduction method [21] is applied as a second layer of reduction. The ROM framework will be tested on the tape spring opposite-sense bending to evaluate the efficiency of the reduced order model approach. Due to various particular features of the tape spring opposite-sense bending problem, various novel modifications need to be introduced with respect to the framework presented in [21].

Reduced order basis

A quasi-static structural problem with geometrical nonlinearities but within the elastic regime is considered here. After discretisation, the problem is defined by the following nonlinear equation, denoted as the full-order model (FOM):

$$\mathbf{r}\left[\mathbf{u}\right] = \mathbf{f}\left[\mathbf{u}\right] - \mathbf{f}_{ext} = \mathbf{0} \tag{4.1}$$

where the displacement vector $\mathbf{u} \in \mathbb{R}^N$ is the unknown in this equation. Solving this system of dimension N is expensive, motivating a reduction of the dimension. The displacement vector can be approximated within a subspace of dimension $k << N$ as follows:

$$\mathbf{u} = \mathbf{V}\mathbf{q} \tag{4.2}$$

where $\mathbf{q} \in \mathbb{R}^k$ is a vector of reduced coordinates, and the matrix $\mathbf{V}$ defines a reduced order basis. Substituting (4.2) in (4.1) and constraining the residual to be orthogonal to the reduced order basis V:

$$\mathbf{r}\left[\mathbf{q}\right] = \mathbf{V}^T\mathbf{f}\left[\mathbf{Vq}\right] - \mathbf{V}^T\mathbf{f}_{ext} = \mathbf{0} \tag{4.3}$$

To solve this system of equations we use the Newton-Raphson iteration method [3]. We define an initial guess $\mathbf{q}_0$ and correct this solution by construct a sequence of increments $\Delta\mathbf{q}$ as:

$$\mathbf{J}\Delta\mathbf{q} = -\mathbf{r} \tag{4.4}$$

where $\mathbf{J} \in \mathbb{R}^{k\times k}$ is the jacobian of equation (4.3):

$$\mathbf{J} = \frac{\partial\mathbf{r}}{\partial\mathbf{q}} \tag{4.5}$$

Substituting (4.3) in (4.5):

$$\mathbf{J} = \mathbf{V}^T\mathbf{K}\mathbf{V} \tag{4.6}$$

where $\mathbf{K}$ is the stiffness matrix, defined by:

$$\mathbf{K} = \frac{\partial\mathbf{f}}{\partial\mathbf{u}} \tag{4.7}$$

Substituting (4.6) and (4.3) in (4.4):

$$\left(\mathbf{V}^T\mathbf{K}\mathbf{V}\right)\Delta\mathbf{q} = -\mathbf{V}^T\mathbf{f} + \mathbf{V}^T\mathbf{f}_{ext} \tag{4.8}$$

The cost of solving the nonlinear system of equations (4.8) of size k is less than the original system since $k << N$. However, constructing this system requires calculating $\mathbf{f}$ and $\mathbf{K}$ for computing the reduced quantities $\mathbf{V}^T\mathbf{f}$ and $\mathbf{V}^T\mathbf{KV}$ respectively. Therefore, the construction of this system still scales with the dimension of the FOM. This motivates a further layer of reduction, described in the following sub-section.

It must be noted that the displacement variables part of the boundary conditions are known and therefore do not require approximation. Therefore, in practice the reduced order basis is defined only for the degrees of freedom which are not part of the displacement boundary conditions. In other words, if the structural problem is defined in the region Ω and has displacement boundary conditions of the form:

$$\mathbf{u}|_{\Omega_1} = \bar{\mathbf{u}} \tag{4.9}$$

Then the reduced order basis approach is used in the region $\Omega - \Omega_1$, such that equation (4.2) becomes:

$$\mathbf{u}|_{\Omega-\Omega_1} = \mathbf{Vq} \tag{4.10}$$

where the operator $-$ in this context indicates a subtraction of sets.

Hyper-reduction scheme

The scheme described here will follow [21]. The vector of internal forces $\mathbf{f}$ can be expressed as a sum of components at the element level, following standard assembly procedures [3]:

$$\mathbf{f} = \sum_{e=1}^{n_e} \mathbf{R}_e^T \mathbf{f}_e \tag{4.11}$$

where n_e is the number of finite elements in the mesh, $\mathbf{f}_e \in \mathbb{R}^d$ is the vector of internal forces within an element e, $\mathbf{R}_e^T \in \mathbb{R}^{N\times d}$ is a boolean matrix with 0s and 1s that indexes $\mathbf{f}_e$ within the global internal force vector $\mathbf{f}$, and d is the number of degrees of freedom per finite element. Similarly, the stiffness matrix $\mathbf{K}$ can be expressed as a sum of components at the element level:

$$\mathbf{K} = \sum_{e=1}^{n_e} \mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e \tag{4.12}$$

where $\mathbf{K}_e \in \mathbb{R}^{d\times d}$ is the stiffness matrix of an element e. The reduced vector of internal forces $\mathbf{V}^T\mathbf{f}$, required to evaluate (4.3), can be expressed similar to (4.11):

$$\mathbf{V}^T\mathbf{f} = \sum_{e=1}^{n_e} \mathbf{V}^T \mathbf{R}_e^T \mathbf{f}_e \tag{4.13}$$

The reduced stiffness matrix $\mathbf{V}^T\mathbf{K}\mathbf{V}$, required to evaluate (4.8), can be expressed similar to (4.12):

$$\mathbf{V}^T\mathbf{K}\mathbf{V} = \sum_{e=1}^{n_e} \mathbf{V}^T \mathbf{R}_e^T \mathbf{K}_e \mathbf{R}_e \mathbf{V} \tag{4.14}$$

We observe that evaluating (4.13) and (4.14) scales with the number of total elements in the mesh n_e, being expensive computationally.

The hyper-reduction method is defined by approximating the vector $\mathbf{V}^T\mathbf{f}$ using only a subset of the original mesh, herein denoted as a reduced mesh, and scaling the element contributions with a weight. More specifically, given the original set of finite elements E_0 we identify a subset $\hat{E}$ of size $h << n_e$ and a set of weights ξ_e such that the weight is zero for elements not contained in the subset $\hat{E}$:

$$\begin{cases} \xi_e = 0 & e \notin \hat{E} \\ \xi_e \neq 0 & e \in \hat{E} \end{cases} \tag{4.15}$$

This is used to approximate the reduced internal forces vector (4.13) as follows:

$$\mathbf{V}^T\mathbf{f} = \sum_{e=1}^{n_e} \mathbf{V}^T \mathbf{R}_e^T \xi_e \mathbf{f}_e \tag{4.16}$$

where the terms of the sum are zero if $e \notin \hat{E}$ and can be ignored. Therefore this expression is equivalent to:

$$\mathbf{V}^T\mathbf{f} \approx \sum_{e\in\hat{\mathbf{E}}} \mathbf{V}^T\mathbf{R}\xi_e\mathbf{f}_e \tag{4.17}$$

Similarly, the reduced stiffness matrix (4.14) is approximated as follows:

$$\mathbf{V}^T\mathbf{K}\mathbf{V} \approx \sum_{e\in\hat{\mathbf{E}}} \mathbf{V}^T\mathbf{R}_e^T\mathbf{K}_e\mathbf{R}_e\mathbf{V} \tag{4.18}$$

In this manner, the computation of $\mathbf{V}^T\mathbf{f}$ now scales with the size h of the subset $\hat{E}$, much smaller than the size n_e of the original set of elements E_0. In order to find the mesh subset $\hat{E}$ and the weights ξ_e an optimization procedure is used [21] based on data acquired. For different training configurations $i \in \{1, 2, ..., n_t\}$, which can correspond to different load magnitudes or times, the internal forces will have different values, and are denoted as $\mathbf{f}_i$. We identify the reduced internal forces vectors at the element and global level as:

$$\mathbf{g}_{ie} = \mathbf{V}^T\mathbf{R}_e\mathbf{f}_{ie} \tag{4.19}$$

$$\mathbf{b}_i = \mathbf{V}^T\mathbf{f}_i \tag{4.20}$$

The reduced order basis is defined on the degrees of freedom not associated with displacement boundary conditions, that is, in the region $(\Omega - \Omega_1)$. Therefore the vectors $\mathbf{g}_{ie}$ and $\mathbf{b}_i$ are defined with degrees of freedom in the region $(\Omega - \Omega_1)$.

$$\mathbf{g}_{ie} = \mathbf{V}^T \left.(\mathbf{R}_e\mathbf{f}_{ie})\right|_{\Omega-\Omega_1} \tag{4.21}$$

$$\mathbf{b}_i = \mathbf{V}^T \left.\mathbf{f}_i\right|_{\Omega-\Omega_1} \tag{4.22}$$

Using these equations with (4.13) it is evident that:

$$\sum_{i=1}^{n_e} \mathbf{g}_{ie} = \mathbf{b}_i \tag{4.23}$$

The element and global contributions for different configurations are organized in block-matrix form as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_{11} & \cdots & \mathbf{g}_{1n_e} \\ \cdots & & \cdots \\ \mathbf{g}_{n_t1} & \cdots & \mathbf{g}_{n_tn_e} \end{bmatrix} \tag{4.24}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \dots \\ \mathbf{b}_{n_t} \end{bmatrix} \tag{4.25}$$

These matrices will be used in the optimization procedure for the weights ξ_e and the mesh subset $\hat{E}$. The weights ξ_e are grouped in a vector $\boldsymbol{\xi}$ and an error metric is defined as:

$$\mathbf{k} = ||\mathbf{G}\boldsymbol{\xi} - \mathbf{b}||_2 \tag{4.26}$$

We observe that choosing all the weights ξ_e equal to 1 corresponds to an error metric $\mathbf{k} = ||\mathbf{G}\boldsymbol{\xi} - \mathbf{b}||_2 = 0$ due to (4.23), (4.24) and (4.25). This indicates that the full mesh, where all the elements are correctly accounted for, has zero error metric. Choosing a different vector of weights $\boldsymbol{\xi}$ will define a mesh subset $\hat{E}$ for computations with error defined by (4.26).

Optimal values $\boldsymbol{\xi}_0$ are found via a minimization using the L_0 norm:

$$\boldsymbol{\xi}_0 = \arg\min_{\boldsymbol{\xi}\in\Phi} ||\boldsymbol{\xi}||_0 \tag{4.27}$$

where $\boldsymbol{\Phi}$ is the space of acceptable solutions. The use of L_0 minimizes the non-zero components of the weights vector, which is desirable since this minimizes the size of the subset $\hat{E}$. The space of acceptable solutions Φ is defined in [21] as:

$$\Phi = \{\xi \in \mathbb{R}^{n_e} : \mathbf{k} \leq \tau \, ||\mathbf{b}||_2 \, , \boldsymbol{\xi} \geq 0\} \tag{4.28}$$

The equations (4.27) and (4.28) define an optimization problem which finds a vector $\boldsymbol{\xi}$ with a minimal number of non-zero elements that bounds the error, as measured by $\mathbf{k}$, using a tolerance τ. The condition $\boldsymbol{\xi} \geq 0$ ensures that the element weights are positive.

The exact solution of the optimization problem requires an exhaustive search and is NP-hard [21]. Therefore, an algorithm that yields a suboptimal solution is used, namely the algorithm for solving the sparse non-negative least squares problem [32]. The procedure start with a guess of $\boldsymbol{\xi}$ as the zero vector $\boldsymbol{\xi} = \mathbf{0}$ and finds an optimal element to add to the subset $\hat{E}$, that is, the component of the vector $\boldsymbol{\xi}$ which should be non-zero. This is repeated iteratively by adding more and more elements until the error metric drops below the tolerance, or equivalently, the condition $\mathbf{k} \leq \tau \, ||\mathbf{b}||_2$ is satisfied.

Hyper-reduction scheme - Modifications

Various modifications are introduced with respect to the approach presented [21], since it is not possible to use the optimization procedure directly.

- For the tape spring opposite-sense bending problem, the displacement degrees of freedom within the region Ω_1 produce the ends rotation, and internal forces are produced in those degrees of freedom. On the other hand, the displacement degrees of freedom within the region $(\Omega - \Omega_1)$ not part of the displacement boundary conditions are free of external forces. Therefore at equilibrium we have:

  $$\begin{cases} \mathbf{f}_i|_{\Omega_1} \neq \mathbf{0} \\ \mathbf{f}_i|_{\Omega-\Omega_1} = \mathbf{0} \end{cases} \tag{4.29}$$

  Since the reduced order basis is only defined for the region $(\Omega - \Omega_1)$, following (4.22) we have that $\mathbf{b}_i = \mathbf{0}$ and therefore from (4.25) we get $\mathbf{b} = \mathbf{0}$. Since we start the optimization procedure with the zero vector $\boldsymbol{\xi} = \mathbf{0}$ then the error metric $\mathbf{k} = ||\mathbf{G}\boldsymbol{\xi} - \mathbf{b}||_2$ is zero trivially and therefore is not useful.

  A well-defined optimization procedure is introduced by including the internal forces where the displacement boundary conditions are defined, that is, $\mathbf{f}_i|_{\Omega_1}$. We define a new vector $\mathbf{b}_i^*$ as:

  $$\mathbf{b}_i^* = \begin{bmatrix} \mathbf{V}^T\, \mathbf{f}_i|_{\Omega-\Omega_1} \\ \mathbf{f}_i|_{\Omega_1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_i|_{\Omega_1} \end{bmatrix} \tag{4.30}$$

  We also define a new vector $\mathbf{g}_{ie}^*$ as:

  $$\mathbf{g}_{ie}^* = \begin{bmatrix} \mathbf{V}^T\, (\mathbf{R}_e\mathbf{f}_{ie})|_{\Omega-\Omega_1} \\ \mathbf{V}^T\, (\mathbf{R}_e\mathbf{f}_{ie})|_{\Omega_1} \end{bmatrix} \tag{4.31}$$

  The element and global contributions for different configurations are organized in block-matrix form as follows:

  $$\mathbf{G}^* = \begin{bmatrix} \mathbf{g}_{11}^* & \cdots & \mathbf{g}_{1n_e}^* \\ \ldots & & \ldots \\ \mathbf{g}_{n_t1}^* & \cdots & \mathbf{g}_{n_tn_e}^* \end{bmatrix} \tag{4.32}$$

  $$\mathbf{b}^* = \begin{bmatrix} \mathbf{b}_1^* \\ \ldots \\ \mathbf{b}_{n_t}^* \end{bmatrix} \tag{4.33}$$

  The new error metric is defined as:

  $$\mathbf{k}^* = ||\mathbf{G}^*\boldsymbol{\xi} - \mathbf{b}^*||_2 \tag{4.34}$$

  The space of acceptable solutions is defined as:

  $$\Phi = \{\xi \in \mathbb{R}^{n_e} : \mathbf{k}^* \leq \tau\, ||\mathbf{b}^*||_2\, , \boldsymbol{\xi} \geq 0\} \tag{4.35}$$

From (4.29), (4.30) and (4.33) we get that $\mathbf{b}^* \neq 0$. Therefore we can start the optimization procedure with a weight vector $\boldsymbol{\xi} = \mathbf{0}$ while from (4.34) we get $\mathbf{k}^* \neq 0$, that is, the error metric is non-zero (as desired).

- The condition $\boldsymbol{\xi} \geq 0$ was introduced in [21] as a way to guarantee the positive-definiteness of the reduced stiffness matrices $\mathbf{V}^T\mathbf{K}\mathbf{V}$, since the weights ξ_e are used in the approximation (4.18). However, the tape spring opposite-sense bending problem involves tracing stable and unstable equilibrium paths, for which the stiffness matrices may not necessarily be positive-definite.

  Therefore we drop this condition in the current hyper-reduction implementation. The space of acceptable solutions is defined as:

$$\Phi = \{\xi \in \mathbb{R}^{n_e} : \mathbf{k}^* \leq \tau \, \|\mathbf{b}^*\|_2\} \tag{4.36}$$

  The optimal values of the weights vector $\boldsymbol{\xi}_0$ are found via a minimization using (4.27).

Since we no longer need to guarantee that $\boldsymbol{\xi} \geq 0$, a modification of the algorithm presented in [21] is in order. The new algorithm is shown in Table 4.1, where we denote by $\mathbf{G}^*_{\hat{E}}$ the column-wise restriction of the matrix $\mathbf{G}^*$ according to the subset $\hat{E}$. Similarly, $\boldsymbol{\xi}_{\hat{E}}$ is the restriction of the vector $\boldsymbol{\xi}$ to the components indicated by the subset $\hat{E}$.

Table 4.1: Modified hyper-reduction optimization algorithm.

```
$\hat{E} \leftarrow \emptyset$
$Z \leftarrow \{1, 2, ..., n_e\}$
$\boldsymbol{\xi} \leftarrow \mathbf{0}$
while $\|\mathbf{G}^*\boldsymbol{\xi} - \mathbf{b}^*\|_2$ do
    $\mu \leftarrow \mathbf{G}^{*T}\left(\mathbf{b} - \mathbf{G}^*\boldsymbol{\xi}\right)^*$
    $\hat{E} \leftarrow \hat{E} \cup \{\text{maxIndex}(\mu)\}$
    $Z \leftarrow \{1, 2, ..., n_e\} - \hat{E}$
    $\boldsymbol{\xi}_{\hat{E}} \leftarrow \left(\mathbf{G}^{*T}_{\hat{E}}\mathbf{G}^*_{\hat{E}}\right)^{-1}\mathbf{G}^*_{\hat{E}}\mathbf{b}^*$
    $\boldsymbol{\xi}_{\hat{Z}} \leftarrow \mathbf{0}$
```

On-the-fly algorithm

We seek to construct a ROM without using previous simulation data. For this purpose, we initially use the FOM for N_F load steps of the static solver. Afterwards, we construct a ROM using the data from the previous load steps. The reduced order basis matrix $\mathbf{V}$ is computed using the method of snapshots [57]. The reduced mesh $\hat{E}$ and weight coefficients ξ_e are computed as described in the previous subsection.

The ROM obtained is used in subsequent load steps, and is adjusted based on the error observed (error computation is discussed below). The algorithm is described in Figure 4.1. If the ROM error is high, the FOM is used for the current load step, and the solution is used to reconstruct the ROM with the augmented data. If the ROM error is below a specified tolerance, the ROM solution is accepted, and the ROM is used for the next load step. Note that in this case the solution vector is not used to augment the ROM already constructed.

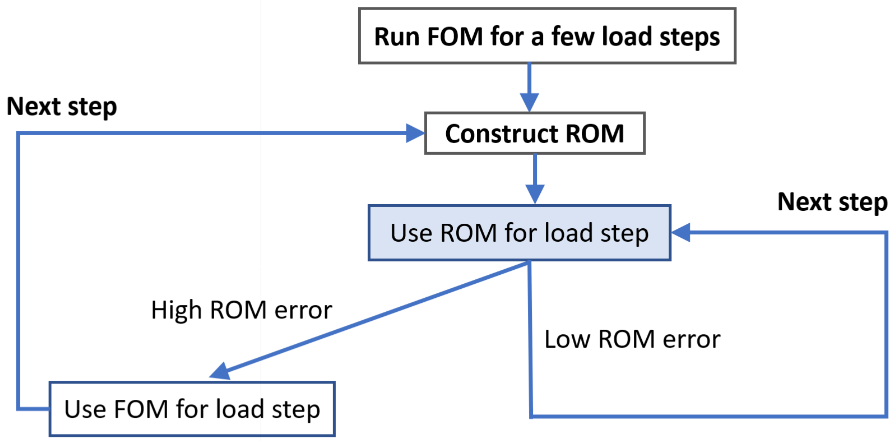


Figure 4.1: On-the-fly algorithm for ROM application.

The error incurred by the ROM can be computed exactly or approximated using various approaches, as presented in [10] and [9]. In this work we choose to compute this error as:

$$e_{ROM} = \frac{\left\|\mathbf{r}_{\Omega-\Omega_1}\right\|}{\|\mathbf{r}\|} \tag{4.37}$$

where $\mathbf{r}_{\Omega-\Omega_1}$ is the restriction of the residual vector $\mathbf{r}$ to the rows indexed by the active degrees of freedom, therefore not including the degrees of freedom where displacement boundary conditions are applied. This particular choice for the error metric (and not simply $\mathbf{r}$) is necessary because of the characteristics of the opposite-sense tape spring bending problem. When convergence is achieved $\left\|\mathbf{r}_{\Omega-\Omega_1}\right\| \approx \mathbf{0}$ while $\|\mathbf{r}\| \neq 0$. The error tolerance is denoted by δ_{ROM}, such that the ROM solution is accepted if $e_{ROM} < \delta_{ROM}$.

This error metric allows us to identify correctly when the ROM ceases to be accurate. However, the cost of computing $\|\mathbf{r}\|$ exactly scales with the dimension of the FOM, noticeably increasing the computational cost. Nevertheless, this cost is still significantly lower cost than computing the stiffness matrix $\mathbf{K}$ of the FOM. Therefore, this choice seeks to strike a balance between the high speed of the pure ROM approach and the accuracy obtained with the FOM.

Numerical results

We use the tape spring opposite-sense bending problem for application of the proposed ROM approach. An arc-length solver is used, using the rotation applied at the ends as the incremental variable. The total rotation applied is chosen to be high enough to observe the snap-through behavior. An adaptive arc length algorithm is used to trace the snap-through curve efficiently. An isotropic tape spring is considered in the following. It has Young's modulus $E = 131$ GPa and Poisson's ratio $\nu = 0.3$. The angle subtended by the cross-section is $\alpha = 110°$.

The FOM was used for the first $N_F = 14$ load steps and then the ROM is constructed, defining a reduced order basis and a subset of the mesh for computations. The reduced order basis constructed with the previous displacement snapshots and without compression is used. This choice does not affect computational efficiency due to the low amount of snapshots acquired at this point. The hyper-reduction method tolerance chosen is $\tau = 0.00025$. The initial mesh, shown in Figure 4.2, has 2048 elements. After hyper-reduction the subset $\hat{E}$ (reduced mesh) has 86 elements at the first step that the ROM is constructed. The reduced mesh is shown in Figure 4.2, where the tape spring 2D mesh is shown along the longitudinal and arc axes. It can be observed that the reduced mesh contains all the elements in the bottom and top ends of the tape spring as well as some internal elements.

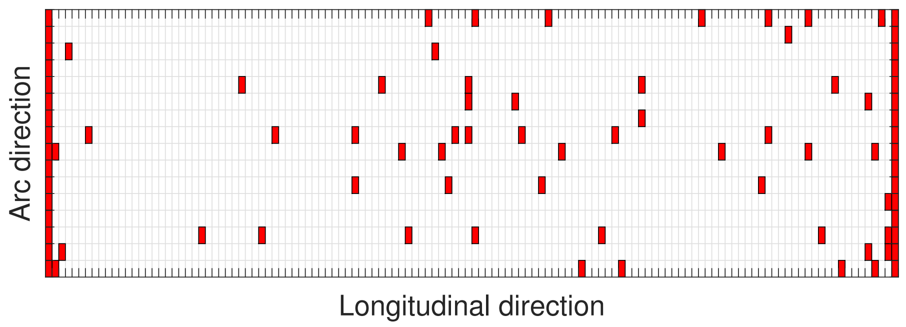


Figure 4.2: Reduced mesh for ROM application, where the initial mesh has 128 x 16 uniform finite elements and the red rectangles represent elements in the reduced mesh.

The ROM error tolerance was set to be $\delta_{ROM} = 0.008$. This value is such that accurate results could be obtained. Looser tolerances result in equilibrium paths that diverge significantly from the true solution. The results are shown in Figure 4.3. It

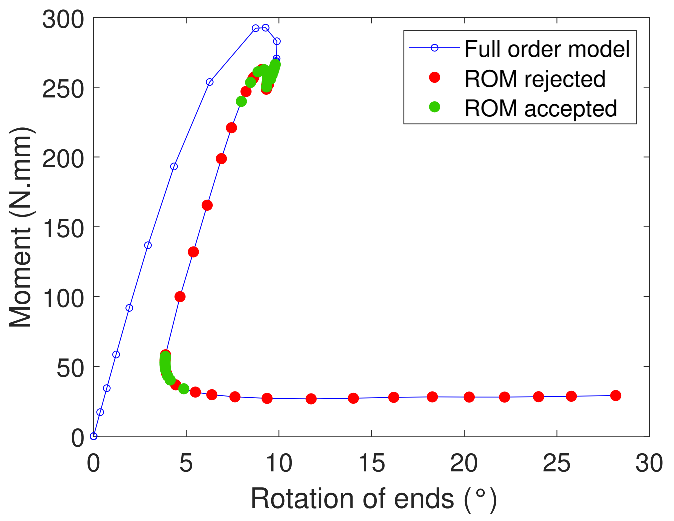


Figure 4.3: Opposite-sense bending of tape spring problem analyzed with on-the-fly reduced order basis and hyper-reduction construction.

can be observed that the ROM constructed is very effective around the regions where a small arc length is used. However, it does not perform well after the snap-through is completed and the fold forms (after the rotation of the ends increases beyond 10°). The CPU time comparisons are given in table 4.2. The ROM approach has a speedup factor of 1.2x compared to the full order model. The ROM is successful in obtaining an accurate solution in 27 out of 61 possible increment step. However, the ROM fails to be accurate in 34 out of 61 increment steps and therefore it is required to switch to the FOM, which significantly penalizes the computation time. In addition, the ROM does not seem useful after the snap-through is completed.

Table 4.2: CPU time for full and reduced order model.

Model	CPU time
Full order model	120 s
Reduced order model	95 s

To study whether the ROM failure after the fold forms is due to the hyper-reduction approximation, we repeat the computations but without using the hyper-reduction procedure. Therefore, the ROM approximation only involves the reduced order basis, and this is equivalent to applying an on-the-fly POD procedure. The results are shown in Figure 4.4. It can be observed that the ROM is still ineffective after the fold forms. The ROM is successful in obtaining an accurate solution in 23 out of 61 possible increment steps.

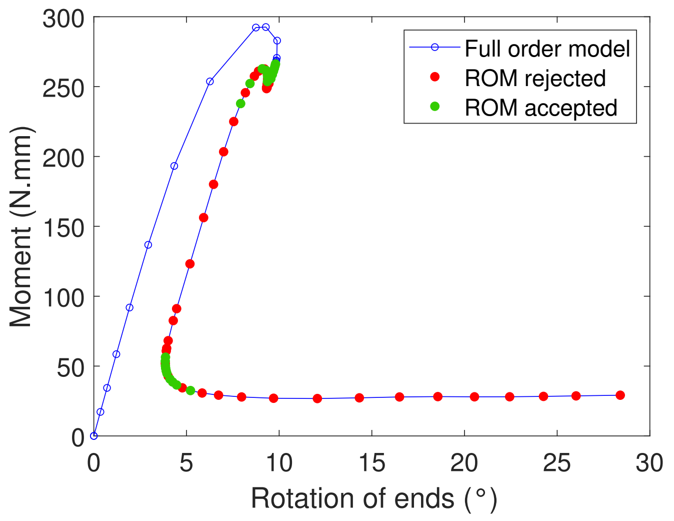


Figure 4.4: Opposite-sense bending of tape spring problem analyzed with on-the-fly reduced order basis and no hyper-reduction.

To investigate this issue further we plot the tape spring at various stages of the folding process, as shown in Figure 4.5. We observe that the straight parts mainly rotate as the fold angle increases, and the fold region size progressively increases. It is well known that of POD models have very limited ability to account for translation or rotation invariances [8]. Expert knowledge of the model can help identify proper mathematical procedures to deal with the invariances, but this strategy has limitations.

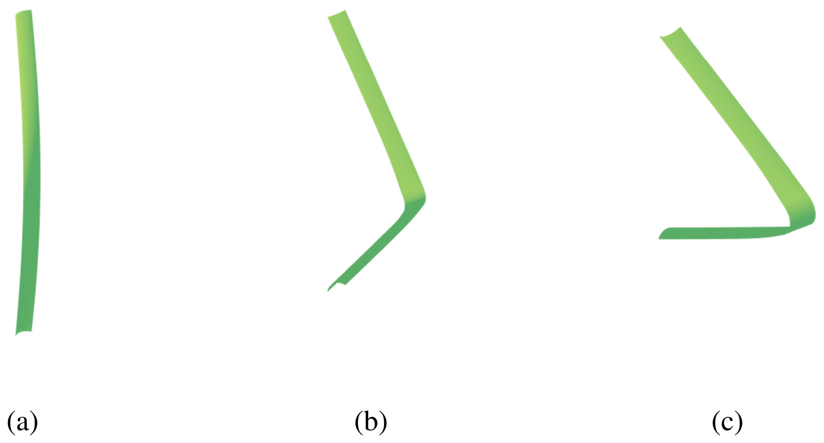


Figure 4.5: Tape spring during opposite-sense bending at different end rotations. (a) Ends rotated by 8° (b) Ends rotated by 40° (c) Ends rotated by 65°.

It is tempting to divide the structure by considering each straight part of the tape spring and the fold as different regions. The displacement approximation for the straight parts of the tape can be assumed to be in an arbitrary orientation with respect to the coordinate system. We can apply a rotation by an angle ψ to obtain the approximation oriented with the coordinate system:

$$\mathbf{u} = \mathbf{R}\left(\psi\right)\mathbf{V}\mathbf{q} \tag{4.38}$$

The angle ψ is unknown initially and therefore a new variable of the problem. The application of (4.38) for model reduction is complicated by the fact that different tape spring sections will have different orientations in general. Therefore the displacement approximations quickly become very involved as multiple superimposed rotations are required. This becomes not viable within an on-the-fly approach due to the high amount of data required before good approximations are possible. Note that (4.38) is a special case of the general approximation for autoencoders, which are neural networks used for reduced order modeling [43]:

$$\mathbf{u} = \mathbf{V}\left(\psi\right)\mathbf{q} \tag{4.39}$$

where $\mathbf{V}\left(\psi\right)\mathbf{q}$ indicates an arbitrary function dependence. For this case, neural networks working with large amounts of offline simulations are often used. Since an on-the-fly procedure is the goal, this approach is not pursued further.

4.2 Adaptive meshing procedure

In this section we detail another approach to reduce the computational cost with an on-the-fly procedure by using an adaptive meshing procedure. The presentation here is partially based on the work of the author given in [13]. This approach involves using a non-uniform distribution of finite elements by using a fine discretisation in regions where more precise interpolation is required. An example of a tape spring with a non-uniform mesh is shown in Figure 4.6.

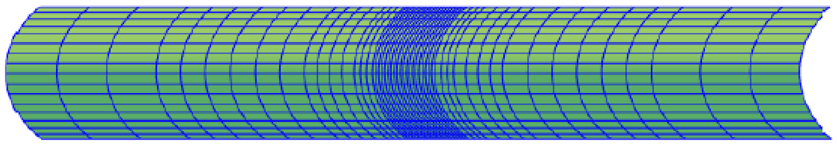


Figure 4.6: Tape spring with non-uniform mesh of finite elements.

The adaptive procedure is different than for standard finite elements due to the tensor product structure of NURBS shape functions. Refinement in the literature is commonly performed in every axis of a given element, which is not possible in general when using NURBS shape functions. Therefore, we propose a new adaptive meshing procedure which is suitable for meshes formed by tensor product. This procedure is described by specifying (i) the refinement indicator and (ii) the method for remeshing. Both are defined below.

Refinement indicator

The refinement indicator is the parameter used to decide whether an element needs to be refined or not. We propose using one refinement indicator for each direction in the local shell finite element coordinate system. Each refinement indicator r_i is defined using the bending strain κ_i and the element length h_i along the corresponding curvilinear coordinate axis i (local to the shell element):

$$r_i = \kappa_i h_i \tag{4.40}$$

It must be noted that the bending strains along each curvilinear coordinate axis are closely related to the curvature changes along each curvilinear axis. Subdivisions are added along a given element axis if the refinement indicator corresponding to that axis exceeds a threshold value r_0. In addition, we define an element length limit h_{th} at which no further refinement is carried out:

$$\text{IF} \quad \begin{cases} r_i > r_0 \\ h_i/2 > h_{th} \end{cases} \quad \text{REFINE along } i \text{ axis} \tag{4.41}$$

The indicator threshold value r_0 will be found by a calibrating procedure. The element length limit h_{th} is defined using the tape spring radius R and thickness t:

$$h_{th} = \sqrt{Rt} \tag{4.42}$$

This parameter is related to the edge effect in tape springs [12]. When there is a fold in a tape spring, within the fold area there is a transition along the arc curvilinear coordinate axis, between the free edges and the central uniform curvature region. To illustrate this region, a tape spring with a fold is shown in Figure 4.7. The red box contains part of the fold region, and the blue curve identifies points at a tape cross-section within the fold region. The fold region seems to have curvature in one direction (along the longitudinal axis) while having close to zero curvature along the other perpendicular axis. Therefore the points along the blue curve would be expected to form an approximately straight line. A plot of the Y coordinate along the blue curve, using the coordinate system from (3.1) is illustrated in Figure

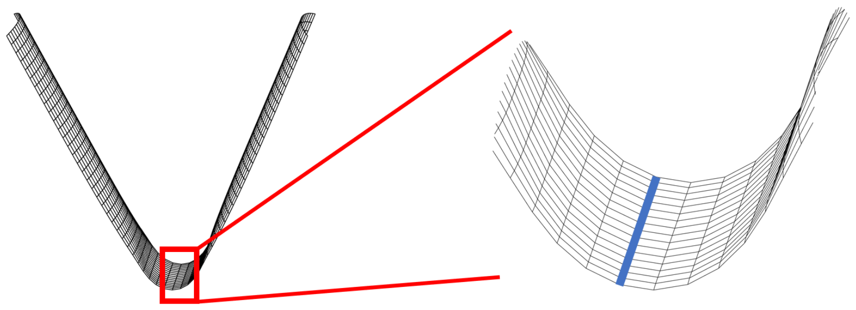


Figure 4.7: Bent tape spring with red box showing fold region and blue curve illustrating a cross-section of the tape spring.

4.8, where c is a non-dimensional coordinate along the blue curve (a tape spring cross-section), and the coordinate y is non-dimensionalized using the tape spring thickness t. It can be seen that the center region of the blue curve is almost straight, but the outer regions are not flat, and the deviations are significant with respect to the tape spring thickness t. Therefore, there is a transition between the outer region and the central, almost flat region. This transition region requires a fine mesh to be modeled accurately. The parameter h_{th} describes this transition length and is therefore used to control the mesh refinement.

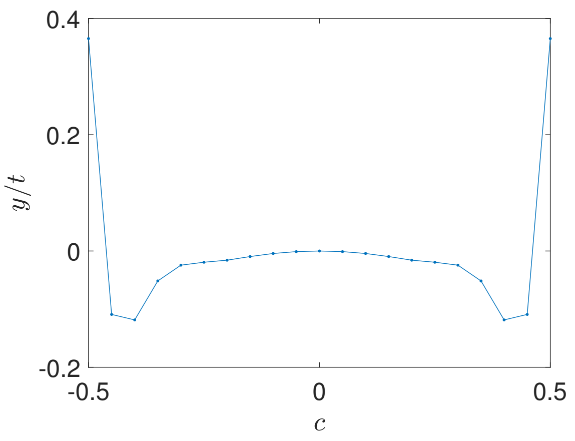


Figure 4.8: Z coordinate along a cross-section within the fold region.

Remeshing scheme

When the refinement indicator r_i exceeds the threshold value r_0, the element is marked for refinement. We refine the element along the axis i by introducing subdivisions. For example, consider the mesh in Figure 4.9a. If the top two elements are considered for refinement along the vertical axis, we introduce the subdivision shown in Figure 4.9b. If the same two elements are considered for refinement along the horizontal axis, we introduce subdivisions such as those shown in Figure 4.9c.

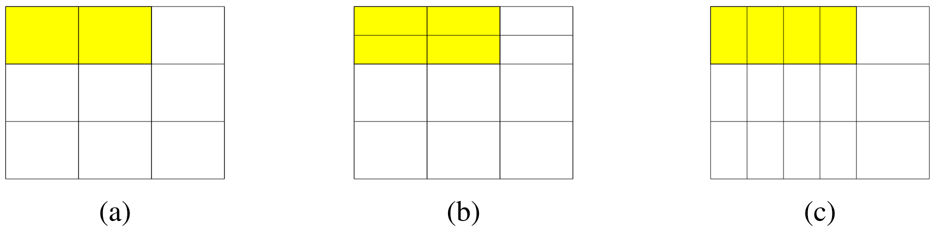


Figure 4.9: Refined meshes. (a) Initial mesh with elements marked for refinement in yellow. (b) Elements subdivided along vertical direction. (c) Elements subdivided along horizontal direction.

.

It must be noted that this type of remeshing (where the additional subdivision lines extend through the whole mesh) is necessary due to the grid structure of the NURBS shape functions used. However, due the particular nature of the folds in tape springs (which drive the adaptive meshing), the proposed adaptive meshing procedure is still efficient, because the fold region is a relatively small rectangular region.

After the finer mesh is defined, the displacement data in the coarse mesh is interpolated to the fine mesh using the NURBS shape functions. However, the interpolated displacement data does not in general satisfy the equilibrium equations defined with the fine mesh. Therefore, a new equilibrium is solved for using the Newton-Raphson method. This is done with the load level fixed at the value from the previous step. Due to the presence of snap-through behavior in tape springs, the standard Newton-Raphson solver might fail. Therefore, a backtracking line search [41] is added to the Newton-Raphson scheme to increase robustness.

Numerical results

Tape springs of various geometries are tested using the adaptive procedure described above. The refinement indicator threshold r_0 is found by a calibration procedure, evaluating the error of a coarse mesh by comparing with results from a fine mesh for a few load steps. We stop this procedure when the error of the coarse mesh exceeds 1%, and the maximum value of the refinement indicator r_i given in Eq. (4.40) is taken as the threshold. We observed that this value only changes very slightly when different tape spring geometries are used, so it is assumed constant. Using this method, the refinement indicator threshold is chosen to be $r_0 = 0.019$.

The Newton-Raphson solver is used, using the rotation applied at the ends as the incremental variable. The total rotation applied is chosen to be high enough to observe the snap-through behavior. The rotation increment at each step is fixed, and dependent on the tape spring geometry. The same step increment is used for results using a uniform mesh and the adaptive mesh procedure.

A geometrically nonlinear isogeometric shell element formulation [29] has been implemented in MATLAB using GPU commands and is used to carry out the analysis. Computations are performed using an Intel i7-8750H processor, 16 GB RAM and a GeForce GTX 1070 GPU. Results from ABAQUS/Standard 2018 finite element software using shell elements S4 and the arc-length method [17] are used for reference.

An isotropic tape spring is considered in the following. It has Young's modulus $E = 131\,\text{GPa}$ and Poisson's ratio $\nu = 0.3$. The angle subtended by the cross-section is $\alpha = 110^\circ$. Initially, we consider a tape spring with the geometrical parameters $L = 300\,\text{mm}$, $R = 10\,\text{mm}$ and $t = 0.1\,\text{mm}$. The initial mesh of the tape spring contains 4 elements along the length and 4 elements along the arc. Using Eq. (4.42), the element length limit for the adaptive procedure is $h_{th} = 1\,\text{mm}$.

The initial and final states of deformation of the tape spring are shown in Figure 4.10; the formation of a localized deformation at the center is evident. Using the adaptive procedure we obtain the results shown in Figure 4.11. Results from ABAQUS are also shown to ascertain the accuracy of the uniform fine mesh. The adaptive procedure results show good accuracy compared to results from ABAQUS and results from a fine mesh. Meshes used at different rotation angles are shown in Figure 4.12, as well as the uniform fine mesh used for comparison.

It is observed that after the snap-through behavior occurs (at around 10° of rotation of the ends) the mesh remains unchanged until the end, indicating that the bending strains do not increase further. Also, the number of elements along the arc is the same at the end of the adaptive procedure as with the uniform mesh. However, the number and distribution of elements along the length of the tape spring differs.

Influence of tape spring length, radius and thickness

We tested the adaptive procedure for tape springs of two different lengths: $L = 150$ mm, and $L = 600$ mm keeping the radius and thickness the same as before. We plot the results using the adaptive meshing method and the results obtained using a uniform fine mesh as a reference in Figure 4.13. Good agreement is observed in both cases. The steady-state moment is almost the same in both cases, which agrees

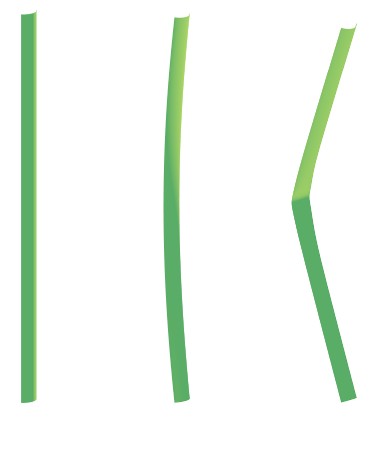


Figure 4.10: Tape spring with parameters $L = 300$ mm, $R = 10$ mm and $t = 0.1$ mm. (a) Initial state. (b) Deformed state before snap-through. (b) Final deformed state.

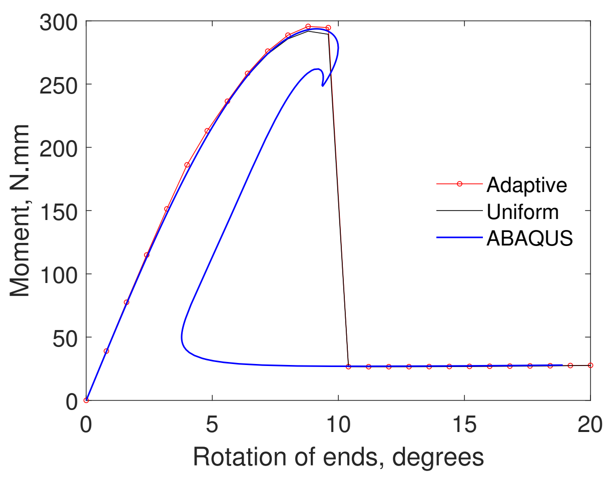


Figure 4.11: Moment-rotation plots for tape spring with parameters $L = 300$ mm, $R = 10$ mm and $t = 0.1$ mm.

with the analytical results from [55].

The speedup factor of the adaptive procedure compared to using a uniform mesh with similar accuracy is given in Table 4.3. This includes time spent remeshing and finding a new equilibrium solution after remeshing. With longer lengths, the fold that is created in the tape spring becomes a smaller part of the total structure and is more localized. This is shown in Figure 4.14, where the final deformation state for the two tape spring geometries are presented. Due to the more localized nature of the fold as the tape spring is longer, the adaptive procedure is more efficient.

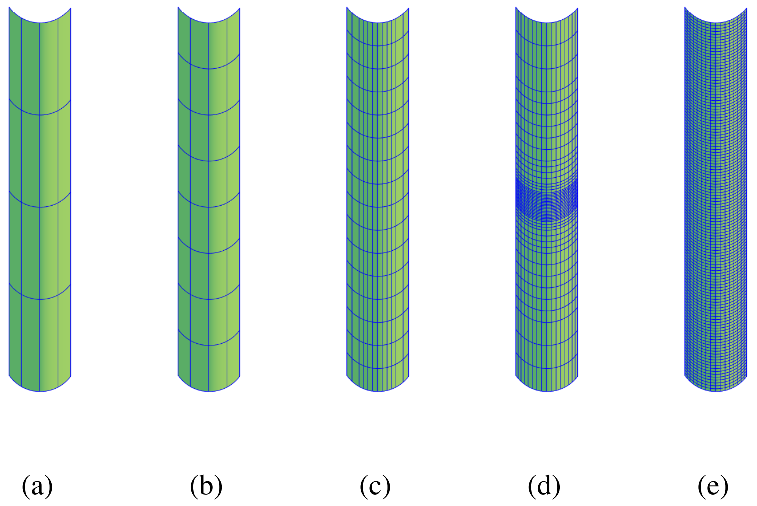


Figure 4.12: Meshes at different rotation angles from adaptive meshing simulation shown in Figure 4.11 and uniform mesh. (a) Initial mesh (16 elements). (b) 2.4° rotation (32 elements). (c) 5.6° rotation (192 elements). (d) 10.4° rotation (832 elements). (e) Uniform mesh (2048 elements).

.

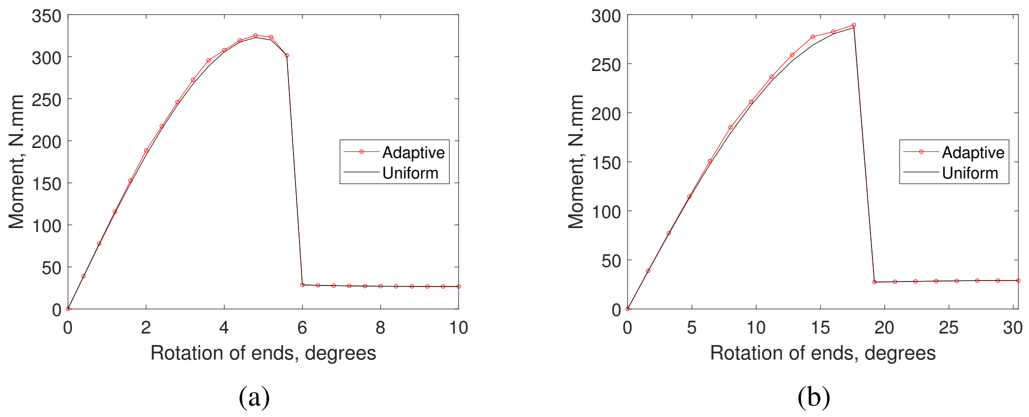


Figure 4.13: Moment-rotation plots for tape springs with parameters $R = 10\,\text{mm}$, $t = 0.1$ mm and different lengths. (a) $L = 150$ mm. (b) $L = 600$ mm.

.

We now consider tape springs with two sets of radii and thicknesses, given in Table 5.9. Using Eq. (4.42), the element length limit is $h_{th} = 1.2\,\text{mm}$ for the first case and $h_{th} = 1.4\,\text{mm}$ for the second case. The initial mesh is the same as before, with 4 elements along the length and 4 elements along the arc. Results are shown in Figure 4.15 for both the adaptive procedure and the uniform fine mesh. The speedup factor of the adaptive procedure compared to using a uniform mesh with similar accuracy

Table 4.3: Speedup factors for three tape springs with different lengths.

Properties	Speedup Factor
R = 10 mm, t = 0.1 mm, L = 150 mm	3.4
R = 10 mm, t = 0.1 mm, L = 300 mm	4.7
R = 10 mm, t = 0.1 mm, L = 600 mm	10.6

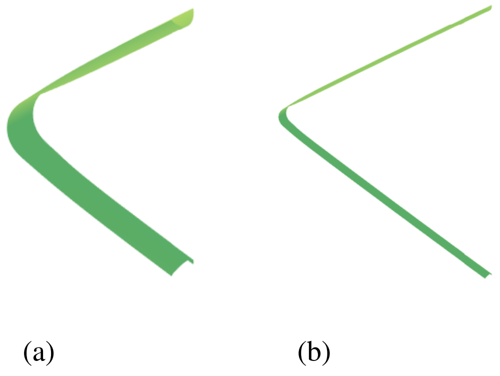


Figure 4.14: Tape spring with ends rotated 60° and parameters R = 10 mm and t = 0.1 mm. (a) L = 150 mm. (b) L = 600 mm.

is given in Table 4.4. It is observed that less accuracy is obtained than in previous cases, although the speedup factor is higher. This suggests that a smaller refinement indicator threshold r_0 should be used in order to get more accurate results while keeping good computational speed.

Table 4.4: Speedup factors for two tape spring geometries with different element length limit.

Properties	Speedup Factor
R = 15 mm, t = 0.1 mm, L = 300 mm	4.2
R = 20 mm, t = 0.1 mm, L = 300 mm	3.6

4.3 Comparison with energy-based refinement

The static analysis of hyperelastic structures is governed by the principle of minimum potential energy [48]. It has been proposed to use this principle to guide the adaptive refinement process [37]. When a mesh refinement is desired, we can choose it based on the reduction in the potential energy. Refinements which result in larger reductions of the potential energy are deemed more accurate.

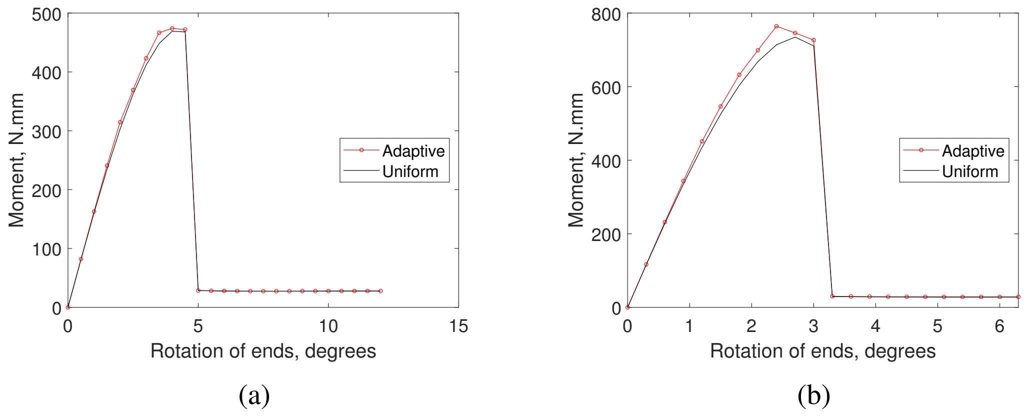


Figure 4.15: Moment-rotation plots for tape springs with $L = 300$ mm and variable element length limit. (a) $R = 15$ mm, $t = 0.1$ mm. (b) $R = 20$ mm, $t = 0.1$ mm.

.

Formulation of minimization problem

We consider a displacement field $\mathbf{u}$, a strain energy W, and tractions $\mathbf{T}$ applied on the traction boundary $\partial\Omega_2$. The total potential energy I of the hyperelastic structure is then given by:

$$I(\mathbf{u}) = W(\mathbf{u}) - \int_{\partial\Omega_2} \mathbf{T} \bullet \mathbf{u} dA \tag{4.43}$$

We will specialize this expression for the present problem of opposite-sense bending of tape springs. Since the element formulation uses only displacement variables, the rotations applied correspond to non-homogeneous essential boundary conditions $\bar{\mathbf{u}}$ in the domain Ω_1. Therefore, there are no external tractions applied on the structure, and the total potential energy of the structure reduces to:

$$\begin{gathered} I(\mathbf{u}) = W(\mathbf{u}) \\ \text{where} \quad \mathbf{u}|_{\Omega_1} = \bar{\mathbf{u}} \end{gathered} \tag{4.44}$$

We now introduce the finite element discretisation and invoke the principle of minimization of total potential energy. Therefore, we seek a displacement vector $\mathbf{u}$ contained within the finite element discretisation space V that minimizes the strain energy W while satisfying the boundary conditions in a region Ω_1:

$$\inf_{\substack{\mathbf{u} \in V \\ \mathbf{u}|_{\Omega_1} = \bar{\mathbf{u}}}} W(\mathbf{u}) \tag{4.45}$$

Numerical comparisons

We consider a tape spring with parameters $L = 300$ mm, $R = 10$ mm and $t = 0.1$ mm. and $\alpha = 110°$. We consider an initial mesh with 8 elements along the arc and 8 elements along the length. We perform the simulation up to a rotation of ends of 7.2°. Figure 4.16 shows the comparison of the results from this coarse mesh with a reference result using a fine mesh.

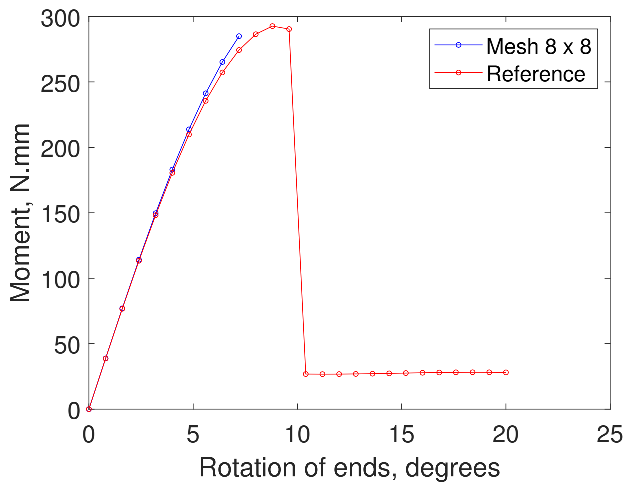


Figure 4.16: Moment-rotation results for 8 x 8 mesh.

Ideally, an adaptive procedure would refine the mesh appropriately in order to follow the reference result more closely. We will test the improvement of the solution when a single subdivision in the longitudinal direction is introduced (adding more elements along the length), as illustrated in Figure 4.17. For every possible longitudinal subdivision, a new mesh will be defined and studied.

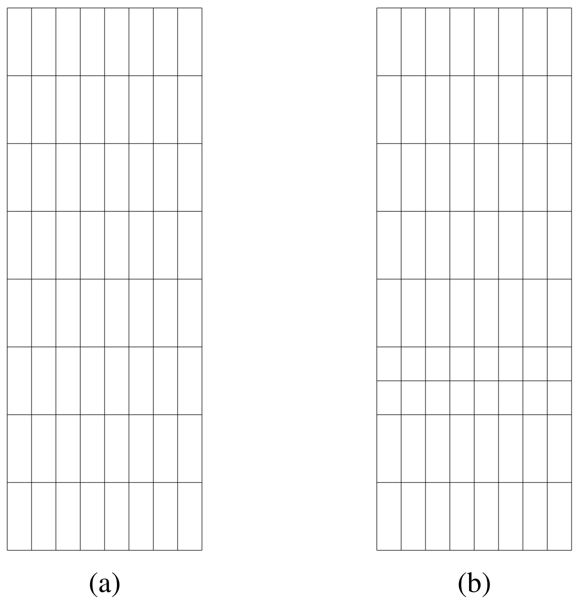


Figure 4.17: Meshes used during refinement, where the length of the tape spring spans in the vertical direction (a) Initial 8 x 8 mesh (b) One possible refined mesh.

After each new mesh is defined, a new equilibrium solution is found by iteration since the finite element discretization space changes. The new strain energy and moment at the same rotation of 7.2° are measured. The change in the strain energy and the change in the moment due to the refinement of the mesh for the 8 subdivisions possible are plotted in Figure 4.18. Note that two of the subdivisions give very similar results and thus the points overlap in the figure.

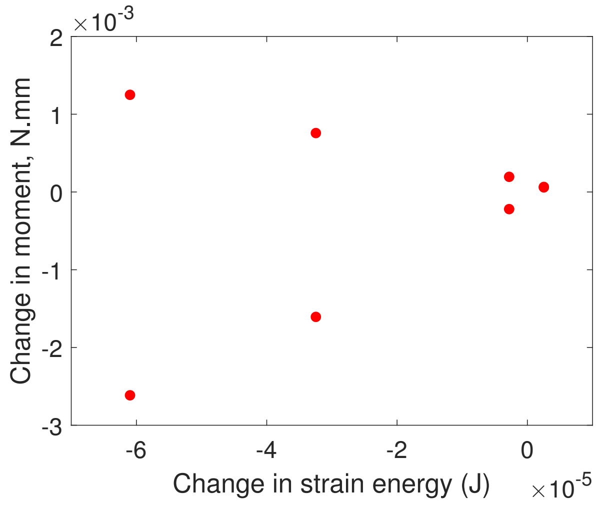


Figure 4.18: Comparison of changes in strain energy and moment for each of the 8 subdivisions possible with reduced integration.

Following the minimization in Eq. (4.45), the solution is improved by refinement when the change in the strain energy is negative. Additionally, since in Figure 4.16 the solution from the 8 x 8 mesh results in a moment higher than from the reference result, the solution is improved by refinement when the change in the moment is negative. We observe that the additional subdivision does not always correspond with an improvement of the moment (a negative change). In particular, one of the subdivisions which results in the largest strain energy decrease will increase the moment and thus the moment-rotation relationship becomes even more inaccurate. The two subdivisions corresponding to the largest strain energy decreases are shown in Figure 4.19, where the tape spring end fixed longitudinally is represented with a black boundary.

It can also be observed from Figure 4.18 that some of the subdivisions introduced result in a slight increase of the strain energy. This seems to be an artifact of the reduced integration used. When full integration is used, no increase of strain energy is observed as a result of refining the mesh (as expected). Figure 4.20 illustrates the changes in strain energy and moment for the same previous case, except that full

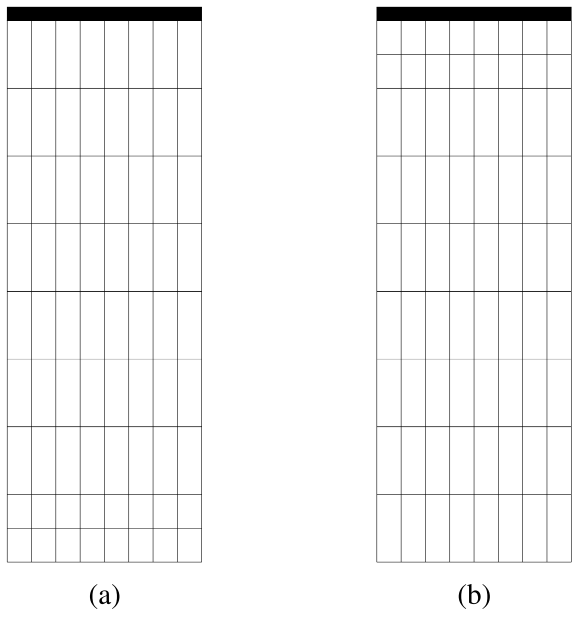


Figure 4.19: Refined meshes with largest strain energy decrease. The end fixed longitudinally is shown with a black boundary (a) Largest moment increase (less accurate) (b) Largest moment decrease (more accurate).

integration is used. Nevertheless, the same issue of large decreases in strain energy resulting in increases of the moment (less accurate) is observed.

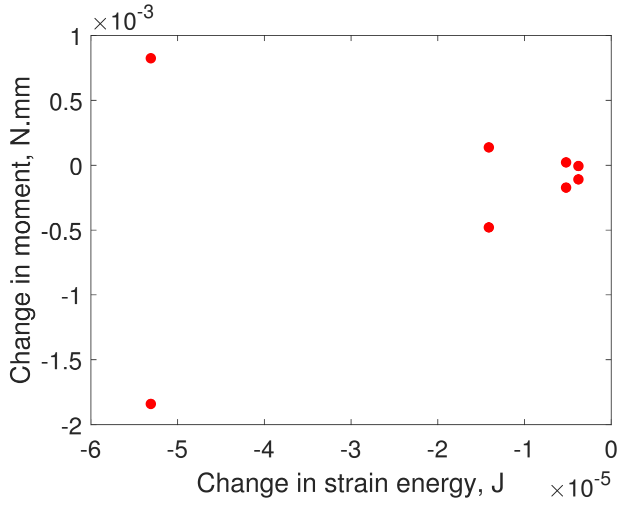


Figure 4.20: Comparison of changes in strain energy and moment for each of the 8 subdivisions possible with full integration.

Since evaluating the strain energy is costly, in practice the minimization is performed using a lower bound estimate. For this purpose, we define a ring of elements incident to the refined elements *e* as Star(e). We then only allow the displacements in the new mesh to differ from the displacements on the old mesh in Star(e). This is illustrated in Figure 4.21. The estimate obtained in this manner is a lower bound on the true strain energy [37].

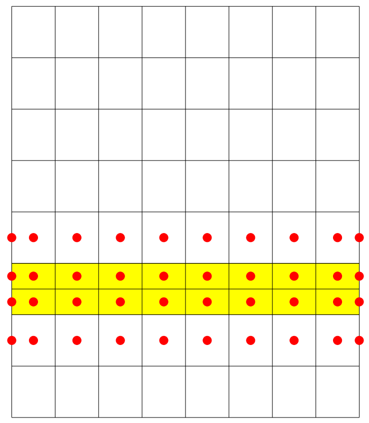


Figure 4.21: Mesh with a single subdivision added in yellow. Control points which are allowed to change given in red.

An attempt to use the minimization method after every increment was performed. The reduction in strain energy was estimated as indicated before. It turned out that in some meshes attempting to perform the jump, that is, when the fold forms suddenly, can be very difficult. Even though an individual mesh can be forced to perform the jump via the backtracking line search in the Newton-Raphson scheme, convergence of some of the refined meshes can be excessively difficult. The solver runs into problems as it tries to refine the mesh while avoiding non-convergence.

4.4 Discussion

The efficient analysis of tape spring opposite-sense bending problem has been investigated using reduced order modeling and adaptive meshing. The method is based on an energy-conserving sampling and weighting of internal forces used during the finite element assembly. An algorithm for applying this reduced order in an on-the-fly approach is presented. Numerical results indicate an speedup of $1.25x$ times with respect to the high-fidelity model. By analyzing the motion during the tape spring bending, it is observed that the translation and rotation invariances are an important part of the model simplification. However, the current reduced order model is incapable of dealing with such invariances efficiently.

A new adaptive meshing procedure for thin shell structures with localized folds has been presented. By using one refinement indicator per element axis, the refinement procedure can be oriented along a particular axis. A refinement limit specific for tape springs has also been introduced based on an estimate of the boundary layer effect of the fold. Test cases have analyzed the effects of various geometrical parameters.The proposed adaptive procedure performs well for the tape springs considered. It is particularly efficient when the tape spring is long relative to its cross-section dimensions.

The energy-based approach presented in [37] was also tested. It was shown that basing the successive refinements purely on the strain energy is not always a robust approach. In particular, the accuracy of the moment-rotation curve can be hampered even when the governing variational principle guides the adaptive procedure. Although this method can test many meshes efficiently, it requires performing equilibrium iterations around the snap-through jump with every mesh. For some of the meshes, achieving convergence can be very difficult. The proposed adaptive procedure is more efficient to apply since the refinement scheme is based on the current curvature state in the elements and it does not require dealing with multiple snap-through jumps.

Chapter 5

TAPE SPRING DEPLOYMENT - SIMULATION

In this chapter, the deployment of tape springs will be analyzed using both ABAQUS commercial software and an in-house finite element code. It will be shown that ABAQUS is very inefficient at analyzing this problem, and this is due to the element formulation used in ABAQUS. Following this, the same problem is tackled using the isogeometric shell code developed in chapter 3, which avoids issues encountered with ABAQUS.

5.1 Problem description

A tape spring is deformed such that a fold forms at the center. One end of the tape spring has a fixed support, while the other end of the tape spring is kept in equilibrium due to forces applied, as shown in Figure 5.1. Afterwards, one end is released and the tape spring deployment occurs. The intermediate folding steps can be defined in various ways, but the end result after folding should be the same. It must be noted that only the final state after folding is relevant for the accuracy of the subsequent dynamic deployment. Therefore, the specific folding procedure is not important.

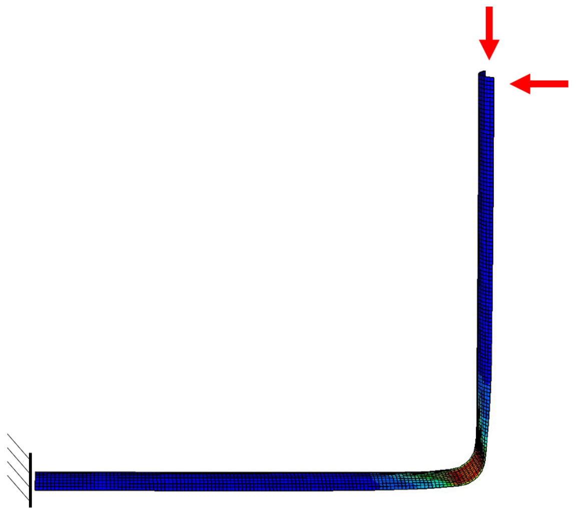


Figure 5.1: Tape spring with a fold at the center, with one end fixed by a support and the other end held kept in equilibrium due to forces applied.

The tape spring considered has geometrical parameters $L = 505$ mm, $t = 0.1$ mm,

$\alpha = 108°$, $R = 14.5$ mm. An isotropic material is considered with Young's modulus $E = 128.7$ GPa, Poisson's ratio $\nu = 0.275$ and density $\rho = 8200$ kg/m^3. These properties will be used for all the simulations of tape spring dynamic deployment.

5.2 ABAQUS simulation - Tape spring folding

An approach for the tape spring folding simulation performed in LS-DYNA finite element software was presented in [60]. A series of folding steps were performed using a dynamic solver, allowing enough time and introducing enough damping to minimize the kinetic energy present in the structure. We followed a similar approach for analysis in ABAQUS finite element software, but defining the fold steps in a slightly different way in order to minimize the artificial energy. We used the explicit solver in ABAQUS with time steps defined by the software. The linear and quadratic bulk viscosity parameters were kept at $b_1 = 0.06$ and $b_2 = 1.2$ respectively.

The mesh was constructed using S4R elements, which have 4 nodes per element, and 6 degrees of freedom per element (3 translations and 3 rotations). Reduced integration is used within these elements. The mesh was composed of 2 640 finite elements and 17 238 degrees of freedom, divided uniformly through the structure by defining an approximate global size.

Center cross-section flattening

The first step is to flatten the center cross-section. For this purpose, we fixed the middle point of the center cross-section, and we moved the points at the outer ends of the center cross-section such that they reach the same Y coordinate as the fixed middle point. The displacement can be computed geometrically as $u_{y0} = R\left(1 - \cos\left(\alpha/2\right)\right)$. The boundary conditions are indicated in Table 5.1, where nodes contained in a plane are indicated by just listing a single coordinate (plane equation). The regions at which boundary conditions are applied are shown in Figure 5.2. The displacement condition is applied as a smooth step in order to minimize generating stress waves, and it is applied over 0.8 s. Afterwards, there is no increase in displacement loads over 0.1 s in order to damp out any kinetic energy within the structure. Therefore, this step takes place over 0.9 s.

The final state of the tape spring is shown in Figure 5.3, where it can be observed that the center cross-section is successfully flattened.

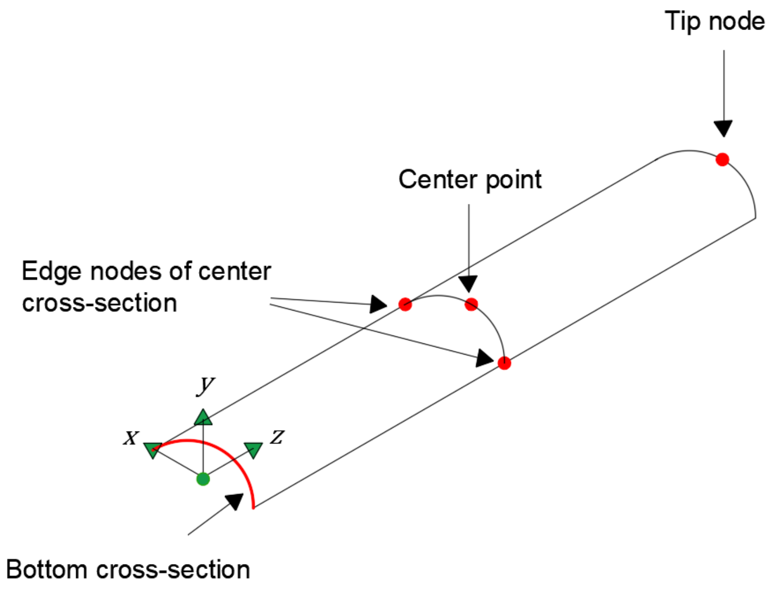


Figure 5.2: Location of boundary conditions for center cross-section flattening, relocation of free end and tip rotation.

Table 5.1: Boundary conditions for center cross-section flattening.

Region	Node location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = 0, \theta_x = \theta_y = \theta_z = 0$
Edge nodes	$x = \pm a/2, z = L/2$	$u_y = u_{y_0}$
Center point	$x = 0, z = L/2$	$u_x = u_y = u_z = 0$

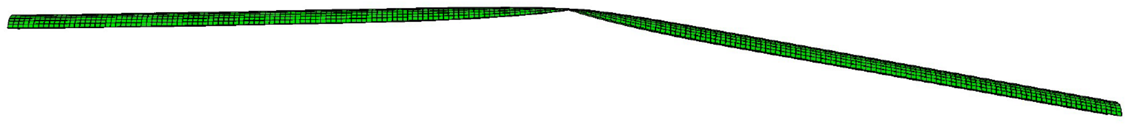


Figure 5.3: Tape spring after center cross-section flattening.

Free end motion

The free end at $z = L$ moves as a consequence of the center cross-section flattening. Therefore, we relocate this end such that it returns to the initial position. We keep the edge nodes and the center point of the center cross-section fixed, while we apply a displacement u_{y1} at one point in the free end along the Y axis. The displacement magnitude u_{y1} is such that the point returns to the original coordinate. The boundary conditions are indicated in the Table 5.2, where the regions at which the boundary conditions are applied are shown in Figure 5.2. The displacement condition is applied as a smooth step in order to minimize generating stress waves, and it is applied over 0.15 s. Afterwards, there is no increase in displacement loads over 0.05 s in order to damp out any kinetic energy.

Table 5.2: Boundary conditions for free end motion.

Region	Node location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = 0, \theta_x = \theta_y = \theta_z = 0$
Edge nodes	$x = \pm a/2, z = L/2$	$u_y = 0$
Center point	$x = 0, z = L/2$	$u_x = u_y = u_z = 0$
Tip node	$x = 0, z = L$	$u_y = u_{y1}$

The final state of the tape spring is shown in Figure 5.4, where it can be observed that the free end returns to the original position while the center cross-section remains flattened.

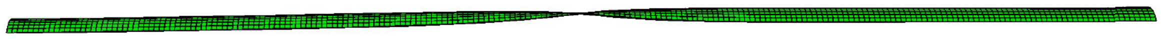


Figure 5.4: Tape spring after free end motion.

Tip rotation

To increase the fold angle one end of the tape spring was rotated with respect to the flattened location, forming a fold. The edge nodes and center point of the center cross-section remain fixed. It is desired to move the tip node such that the fold develops, forming an almost cylindrical shape. Therefore, the tip node was translated in the Y and Z axes following the geometrical path defined by coiling around a cylinder, which is given in [60] as follows:

$$\begin{cases} u_{y1} = R_c\left(1 - \cos\theta\right) + \left(\frac{L}{2} - R_c\theta\right)\sin\theta \\ u_{z1} = R_c \sin\theta + \left(\frac{L}{2} - R_c\theta\right)\cos\theta - \frac{L}{2} \end{cases} \tag{5.1}$$

where R_c is the radius of the coiling cylinder and is taken as $1.06R$. It must be noted that the radius of the fold has been shown to be slightly larger than the initial radius of the tape spring cross-section [56], and therefore we choose R_c slightly larger than R.

The boundary conditions are indicated in the Table 5.3, where the location of the boundary conditions are shown in Figure 5.2. The displacements are applied in tabular form over 1.6 s, and afterwards there is no increase in displacement loads over 0.2 s in order to damp out any kinetic energy. Application of a smooth step for the tip rotation results in very high artificial energy and thus it is not considered here. The angle θ goes from 0 to = 95.7°. The final state of the tape spring is shown in Figure 5.5, where a fold angle slightly larger than 90 degrees can be observed.

Table 5.3: Boundary conditions for tip rotation.

Region	Node location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = 0, \theta_x = \theta_y = \theta_z = 0$
Edge nodes	$x = \pm a/2, z = L/2$	$u_y = 0$
Center point	$x = 0, z = L/2$	$u_x = u_y = u_z = 0$
Tip node	$x = 0, z = L$	$u_y = u_{y2},\ u_z = u_{z2}$

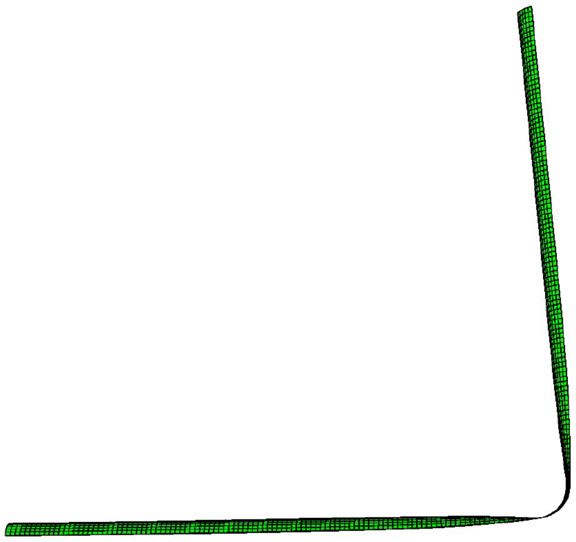


Figure 5.5: Tape spring after tip rotation.

Force release

The points held fixed at the center cross-section was released, which result in a tape spring held just at the ends with a fold at the center. The edge nodes and the center point of the center cross-section are held fixed. The whole cross-section at the free end is kept fixed. The boundary conditions are indicated in the Table 5.4, where the location of the boundary conditions are shown in Figure 5.6. This step takes place over 0.3 s. The final state of the tape spring is shown in Figure 5.7. The tape spring

Table 5.4: Boundary conditions for force release step.

Region	Node location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = 0, \theta_x = \theta_y = \theta_z = 0$
Top cross-section	$z = L$	$u_y = u_z = 0$

configuration changes only very slightly.

Energy history

The strain, kinetic, artificial and total energies of the folding simulation are shown in Figure 5.8. A necessary condition for a simulation is an almost constant total

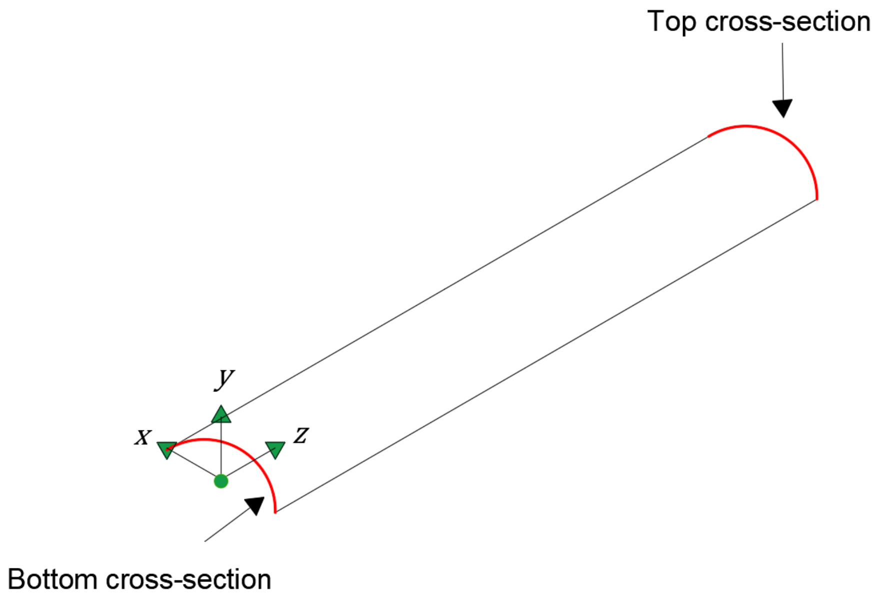


Figure 5.6: Location of boundary conditions for force release step.

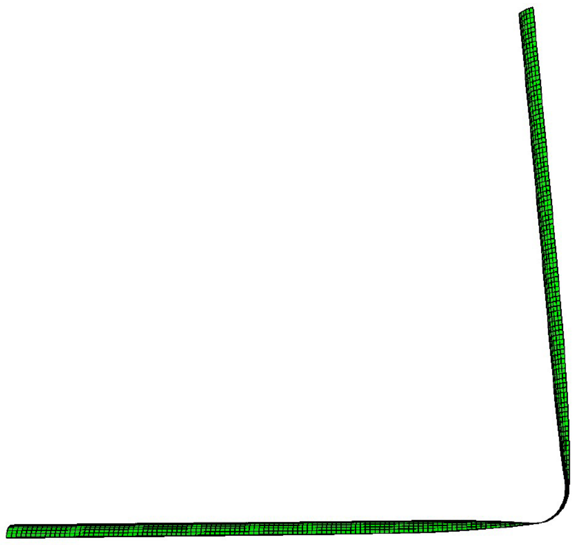


Figure 5.7: Tape spring after force release.

energy and an artificial energy below 5% of the strain energy. It can be seen that the total energy remains fairly constant, and the artificial energy is kept small [1]. At the final time step of the folding procedure the artificial energy is 4.8% of the strain energy, and thus it remains within what is generally considered an acceptable range. Although having long durations for the folding steps increases the computational cost, it was observed that reducing the time duration of the different folding steps created more artificial energy, possibly due to there being more kinetic energy in the system and thus there is a sudden motion introduced.

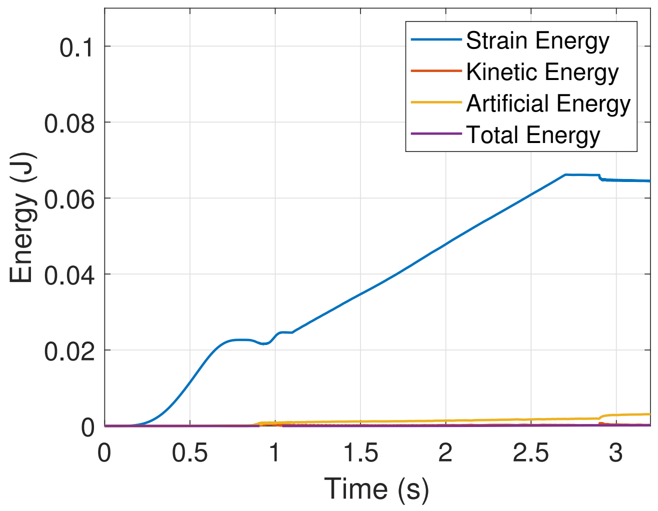


Figure 5.8: Energy history during tape spring folding in ABAQUS.

5.3 ABAQUS simulation - Tape spring deployment

Initial result

In this section, the tape spring, deformed by the folding steps previously described, is deployed. For this purpose, the points held fixed at one end of the tape spring were released. The boundary conditions are indicated in the Table 5.5, where the location of the boundary conditions are shown in Figure 5.2. The same mesh as before was used, with 17,238 degrees of freedom. This step takes place over 0.3 s. No damping is applied during the dynamic deployment.

Table 5.5: Boundary conditions for deployment.

Region	Node location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = 0, \theta_x = \theta_y = \theta_z = 0$

.

The energy history is shown in Figure 5.9, where the X axis indicates the time since the deployment started. It can be observed that the artificial energy is initially small but grows quickly. By the end of the deployment simulation the artificial energy dominates the analysis, being larger than the strain energy. This indicates that the results are not reliable.

Refinement of results

In order to reduce the artificial energy the ABAQUS manual suggests the following approaches [1]:

- Avoiding boundary conditions applied at a single point

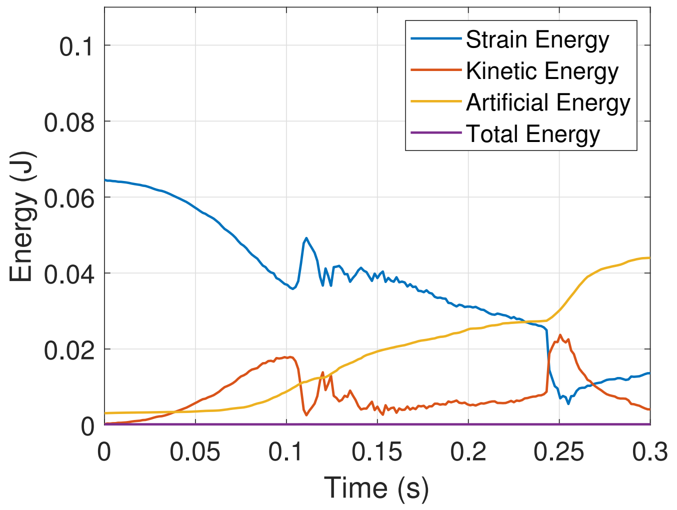


Figure 5.9: Energy history during tape spring deployment in ABAQUS with mesh of 17,238 DOFs.

- Changing the hourglass control
- Refining the mesh

Although boundary conditions are applied at a single point during the folding steps, the artificial energy is kept low. However, the artificial energy grows during deployment, and yet there are no boundary conditions applied at a single point. Therefore, the point boundary conditions are not the issue in the current simulation.

Refining the mesh is the last resort due to the high computational cost involved. Two additional refined meshes were used: one with 37,506 DOFs and another with 76,626 DOFs. The artificial energies during deployment are shown in Figure 5.10. It can be observed that the mesh refinement results in much lower artificial energy. However, the refined mesh with 76,626 DOFs requires a long simulation time, taking 6 days to complete with 4 CPUs in a desktop with Intel Xeon E5-2643 v3 processor and 128 GB of RAM. Therefore, using more refined meshes is not pursued further.

The energy history for the mesh with 76,626 DOFs is shown in Figure 5.11. Although the artificial energy remains low, by the end of the simulation the artificial energy is 10.9% of the strain energy. This is deemed as still too high according to the ABAQUS manual recommendations [1]. Although the results would be expected to be more accurate than before, they are still not ideal and more refined meshes are required.

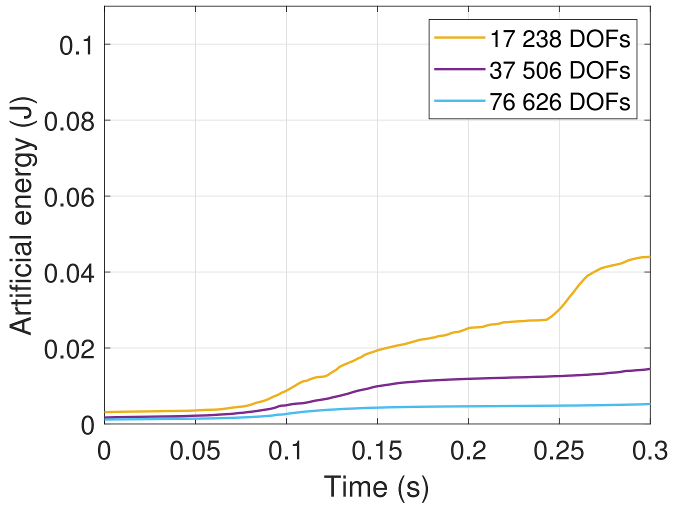


Figure 5.10: Artificial energy during tape spring deployment in ABAQUS for different meshes.

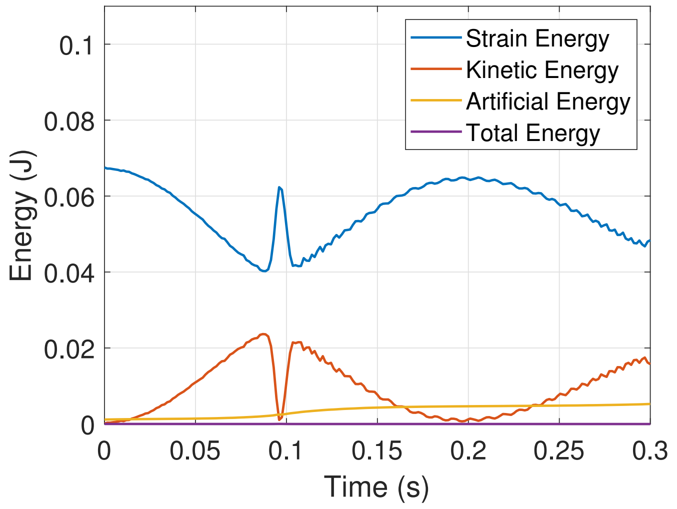


Figure 5.11: Energy history during tape spring deployment in ABAQUS with mesh of 76,626 DOFs.

Discussion

Although the ABAQUS software can simulate the tape spring folding accurately, it is very inefficient for simulation of the tape spring deployment. Large meshes are required solely to reduce the hourglassing effects, which greatly increase the computational cost. Therefore, in the next section we present an alternative element formulation which avoids hourglassing and shear locking effects simultaneously while maintaining low computational cost.

5.4 Hourglassing in isogeometric elements

Hourglass control involves applying artificial forces to reduce the effect of hourglass modes. This is the method of choice in ABAQUS software. Since this method was shown to be very inefficient for the analysis of tape spring deployment, we would like to eliminate hourglassing completely. This can be achieved with fully integrated elements, but this introduces shear locking as an undesirable numerical effect. Figure 5.12 illustrates the difference between full and reduced integration for linear elements with standard polynomial shape functions. In standard finite elements the

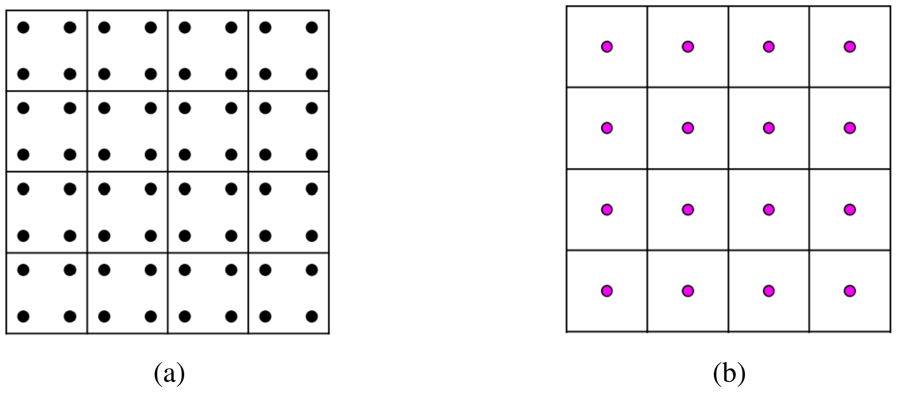


Figure 5.12: Integration rules for linear standard 2D elements (a) Full integration (b) Reduced integration.

.

dichotomy between full and reduced integration implies a choice between dealing with shear locking or hourglassing. However, this dichotomy appears due to the polynomial shape functions commonly used, and is not generally true for other shape functions.

In chapter 3 an isogeometric shell code was implemented, and various of the unique properties of the isogeometric elements were illustrated. The unique continuity of the shape functions used in isogeometric formulations allows non-uniform gauss integration schemes to be introduced [2]. A full integration rule and a special reduced rule are compared in Figure 5.13. The special rule is defined by using 4 integration points in the corner elements, 2 integration points for the remaining elements in the boundaries and 1 integration point for the elements not within the boundaries. Many other unique reduced integration rules for isogeometric elements have been devised.

Unlike for standard finite element, the reduced integration concept for isogeometric elements is not a specific rule but instead encompasses a set of rules. The hour-

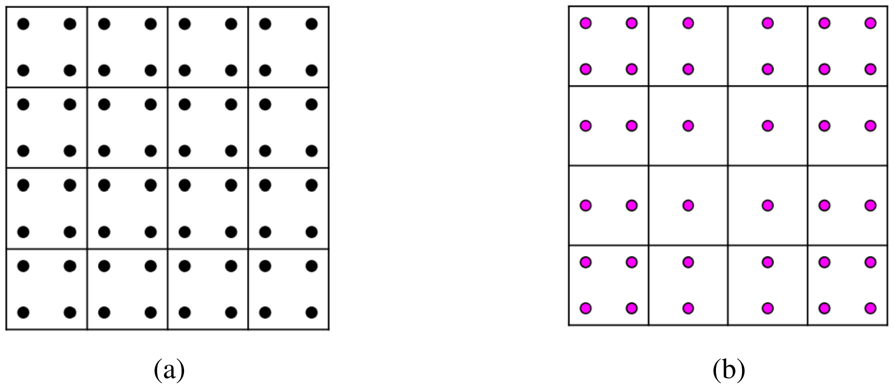


Figure 5.13: Integration rules for linear isogeometric 2D elements (a) Full integration (b) A reduced integration rule [26].

.

glassing and shear locking of each of these rules have been studied for quadratic and cubic isogeometric elements [54]. In particular, one reduced integration rule for quadratic isogeometric elements has been shown to avoid hourglassing entirely. This property has been demonstrated for practical nonlinear shell problems [26]. The rule is similar to the special integration rule mentioned before, but with one extra point in each direction. Shear locking is eliminated at the formulation level by using the Kirchhoff-Love shell theory.

5.5 Isogeometric shell code - Simulation procedure

General description

The tape spring deployment will be simulated using the isogeometric shell finite elements described in chapter 3. Before the deployment can take place, a fold must be generated in the tape spring. The folding steps differ slightly than those used in ABAQUS software since artificial energy is not present due to hourglassing not being a concern.

In the ABAQUS results from the previous section the quasi-static solution for the folded state was obtained using a dynamic solver and allowing enough time and damping for the kinetic energy to be reduced. However, a quasi-static solution can be achieved exactly by using static solvers for the folding steps, instead of a dynamic solver. Using static solvers reduces the computational cost compared to explicit dynamic solvers, but could result in failure to converge. Nevertheless, convergence was achieved for all the folding steps with the isogeometric shell elements.

Center cross-section flattening

In this step, the cross-section located at the center of the tape spring will be approximately flattened. A static solver is used in this step. Figure 5.14 shows an schematic of the boundary condition locations for this step. The bottom cross-section is fixed in translation and rotation, while the longitudinal centerline is fixed in translation. The center cross-section is flattened by moving the two nodes located at the edges along the Y axis until they meet the Y coordinate of the longitudinal centerline. In other words, we set $u_y = u_{y0} = R\,(1 - \cos(\alpha/2))$ for the edge nodes of the center cross-section. The boundary conditions are listed in Table 5.6.

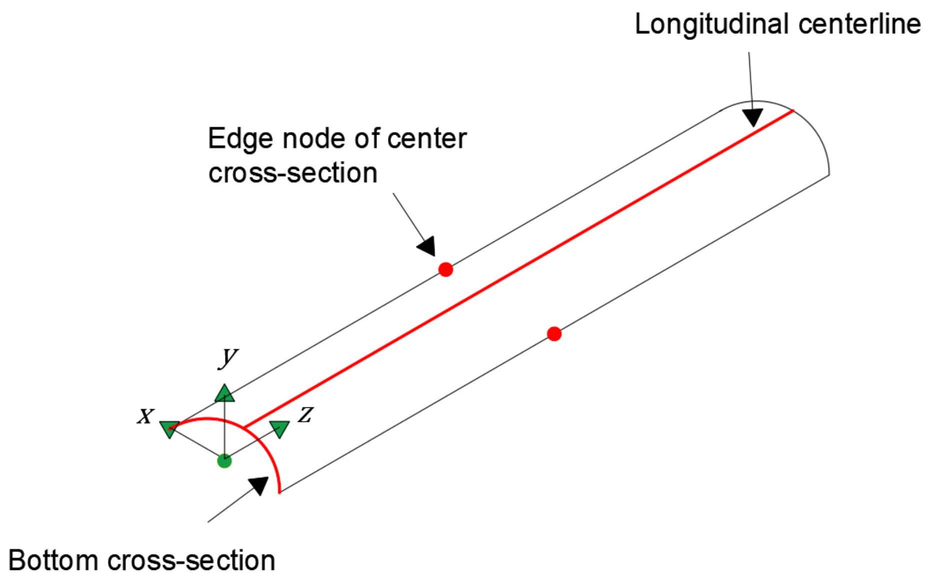


Figure 5.14: Location of boundary conditions for center cross-section flattening. The origin of the coordinate system is represented by the green point.

Table 5.6: Boundary conditions for center cross-section flattening.

Node location	Boundary condition
$z = 0$	$u_x = u_y = u_z = 0, \theta_x = \theta_y = \theta_z = 0$
$x = 0$	$u_x = u_y = u_z = 0$
$x = \pm a/2, z = L/2$	$u_y = u_{y_0}$

.

Tip rotation

In this step, one end of the tape spring was rotated with respect to the center of the tape spring, forming a fold. A static solver was used in this step. The boundary condition locations are identified in Figure 5.15. The center cross-section is fixed in translation, while the bottom cross-section is fixed in translation and rotation.

The node (or nodes) at the tip of the tape spring are translated in the Y and Z axis following the path of a tape spring being coiled around a cylinder [60] as follows:

$$\begin{cases} u_{y1} = R\,(1 - cos\theta) + \left(\frac{L}{2} - R\theta\right) \sin\theta \\ u_{z1} = R \sin\theta + \left(\frac{L}{2} - R\theta\right) \cos\theta - \frac{L}{2} \end{cases} \tag{5.2}$$

The boundary conditions are listed in Table 5.7.

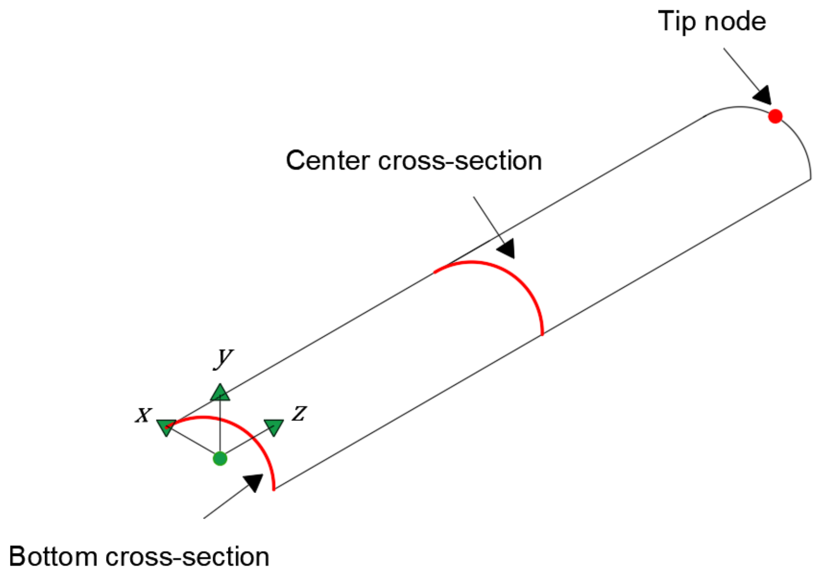


Figure 5.15: Location of boundary conditions for center cross-section flattening.

Table 5.7: Boundary conditions for tip rotation.

Region	Location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = \theta_x = \theta_y = \theta_z = 0$
Center cross-section	$z = L/2$	$u_x = u_y = u_z = 0$
Tip node	$x = 0, z = L$	$u_y = u_{y1},\ u_z = u_{z1}$

.

Force release

In this step, the center cross-section was released, allowing the formation of a fold while only holding the ends of the tape spring. A static solver was used in this step. Figure 5.15 shows a schematic of the boundary condition locations for this step. The tip node is fixed in translation, while the bottom cross-section is fixed in translation and rotation. The boundary conditions are listed in Table 5.8, where nodes contained in a plane are indicated by just listing a single coordinate (plane equation).

Table 5.8: Boundary conditions for force release step.

Region	Location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = \theta_x = \theta_y = \theta_z = 0$
Tip node	$x = 0, z = L$	$u_y = u_z = 0$

.

Deployment

In this step, one end of the tape spring was released, allowing the fold to move and the fold angle to change. The Newmark explicit dynamic solver was used [40]. Zero viscous damping was used during this step. The boundary condition locations are identified in Figure 5.15. Only the bottom cross-section is held fixed in translation and rotation. The boundary conditions are listed in Table 5.9.

Table 5.9: Boundary conditions for dynamic deployment.

Region	Location	Boundary condition
Bottom cross-section	$z = 0$	$u_x = u_y = u_z = \theta_x = \theta_y = \theta_z = 0$

.

5.6 Fold identification algorithm

It will be helpful to define the fold position and fold angle from 3D displacement data obtained via finite element simulations. The fold is identified using the following parameter, which is similar to the one used in [45]:

$$k(z) = \sqrt{\left(\Delta P_y(z)\right)^2 + \left(\Delta P_z(z)\right)^2} \tag{5.3}$$

where $\Delta P_i(z) = p_i(x = 0, y = R, z) - p_i(x = -a/2, y = 0, z)$ and $p_i(x_0, y_0, z_0)$ is the current position coordinate in the i axis of the point initially located in the position (x_0, y_0, z_0). A schematic of ΔP_y at a particular location z of the initial tape spring geometry is shown in Figure 5.16. The parameter k is equal to zero when the cross-section is completely flattened and lies a plane parallel to the XY plane. A typical plot of k/k_0 along the length coordinate z of a tape spring with a

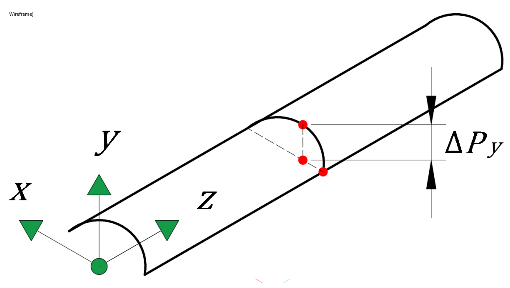


Figure 5.16: Schematic of ΔP_y.

fold is shown in Figure 5.17, where k_0 is the value obtained from Eq. 5.3 for the undeformed tape spring. It can be seen that there is a region with small but non-zero values of k, corresponding to the fold region. This is because the cross-section is not completely flat within the fold due to small bulges near the edges, as mentioned in [55]. We can identify the fold region by using a threshold for the parameter k/k_0. By

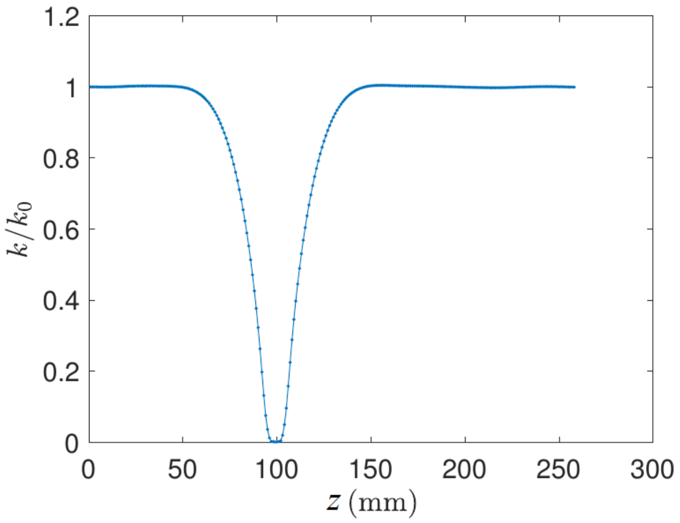


Figure 5.17: Typical plot for k/k_0.

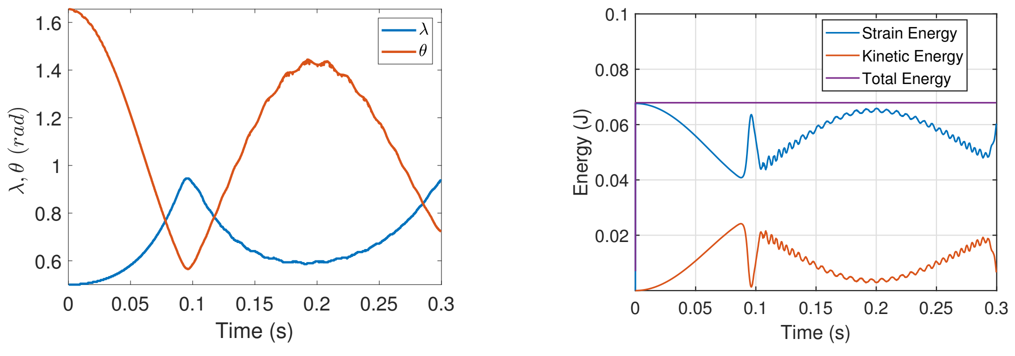
finding the center of the resulting fold region we can define the fold position along the longitudinal axis of the tape spring. The threshold is set to be $k/k_0 = 0.025$.

With the fold position along the tape spring defined by the previous procedure, the fold angle is identified by the angle between two lines that originate from the fold position. The first line is defined from the tape spring support (cross-section at $z = 0$) to the fold position, being normal to the support. To define the second line it is required to decide on a representative point close to the free end of the tape spring. Due to the oscillations present during deployment, this point is found by averaging 10 control point positions along the tape spring centerline.

5.7 Isogeometric shell code - Results

The fold position λ and fold angle θ are shown in Figure 5.18a. The tape spring starts with a fold at the middle ($\lambda = 0.5$) and a fold angle slightly higher than 90 degrees ($\theta = 1.67$ rad). The fold initially moves towards the bottom support while the tape spring starts to straighten and the fold angle is reduced. The fold approaches the support around $t = 0.1$ s (with λ close to 1) and then reflects, moving in the opposite direction back through the tape spring. The fold travels for some time and stops around $t = 0.2$ s close to the enter of the tape spring ($\lambda = 0.6$). Afterwards, there is another fold velocity reversal, and the fold travels back towards the support.

The energy evolution during deployment is shown in Figure 5.18b. Since no hourglassing correction is attempted, there is no artificial strain energy. It can be observed that the total energy remains constant. This is because no energy dissipation mechanism was introduced. However, the fold does not return the initial position, as illustrated in 5.18a. The implications of this result for tape spring deployment will be analyzed in the following section.



(a) (b)

Figure 5.18: Finite element results (a) Fold position λ and angle θ (b) Kinetic, strain, potential and total energies during deployment.

.

Chapter 6

TAPE SPRING DEPLOYMENT - PHYSICS ANALYSIS

In this chapter, the apparent energy leak reported in previous work of the literature is analyzed. First, a brief introduction to the analytical formulation presented in [55] is given. Although this formulation gives accurate evolution of kinematic variables, there is an apparent energy leak. This has motivated the use of various energy dissipation mechanisms in tape spring modeling. Afterwards, a simulated experiment approach is used to try to reproduce the energy leak when various simplifications are applied. The finite element code discussed in chapter 5 is used to analyze the energy history in different parts of the tape spring. It is shown that the energy leak is an artifact of the simplified model used in previous work. Furthermore, this energy analysis illustrates various physical features of the tape spring deployment. These features can help identify regions and quantities of interest for posterior reduced order modeling.

6.1 Analytical formulation

Evolution of kinematic variables

An analytical formulation is developed in [55] for folded tape springs using a two degrees of freedom model. Figure 6.1 illustrates the variables and coordinate systems used. The fold position λ and fold angle θ are taken as kinematic variables, where the fold is considered localized at a point B. There are 2 coordinate systems used to define position vectors: one centered in the support (point O) and another one centered in the fold (point B). The position of a generic point P in the straight part AB of the tape spring is described by a non-dimensional longitudinal coordinate ξ. The position vector of the point B is given by:

$$\mathbf{r}_P = (1 - \lambda)\, L\mathbf{e}_0 + (\xi - 1 + \lambda)\, L\mathbf{e}_1 \tag{6.1}$$

This expression is used to derive the equations of motion for the degrees of freedom λ and θ. The comparison of the analytical predictions with the experimental results is given in Figure 6.2. The analytical formulation has good agreement with experimental results regarding the fold position and fold angle evolution over time.

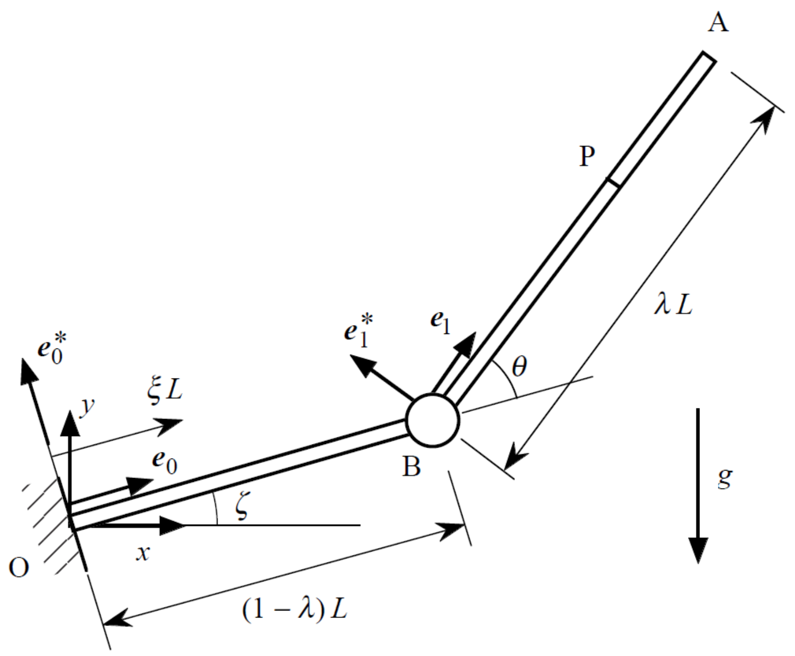


Figure 6.1: Kinematic variables and coordinate system used in [55].

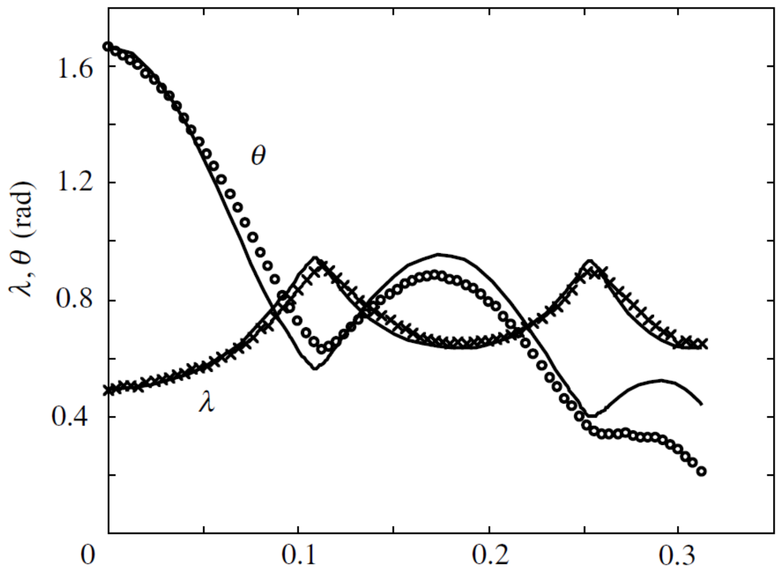


Figure 6.2: Comparison of experimental and analytical estimates. The circles and crosses represent experimental measurements and solid lines correspond to analytical predictions.

Energy formulas

Simplified expressions for the kinetic, strain and potential energy of the tape spring with a fold as a function of the kinematic variables λ and θ can be derived. As a first approximation, the strain energy can be considered to be concentrated on the fold.

An analytical expression for the strain energy in the fold region can be obtained by considering a region of constant transverse and longitudinal curvatures. Following the derivation in [55], the strain energy U is given by:

$$U = D(1+\nu)\alpha\theta \tag{6.2}$$

where $D = Et^3/12(1-\nu^2)$, E is the Young's modulus and ν is the Poisson's ratio. An analytical expression for the kinetic energy of the top straight part (which rotates as the fold angle changes) can be obtained as a function of the fold position λ and angle θ, as well as their time derivatives. Following [55], the kinetic energy T is given by:

$$T = \rho L^3 \left[\lambda\dot{\lambda}^2\left(1 - cos\theta\right) + \frac{1}{2}\lambda^2\dot{\lambda}\dot{\theta}\sin\theta + \frac{1}{6}\lambda^3\dot{\theta}^2\right] \tag{6.3}$$

Energy leak

The fold position and angle observed experimentally were used as input to evaluate the different energy components, and the total energy predicted is shown in Figure 6.3. A sudden decrease in the total energy predicted is observed around $t = 0.12$

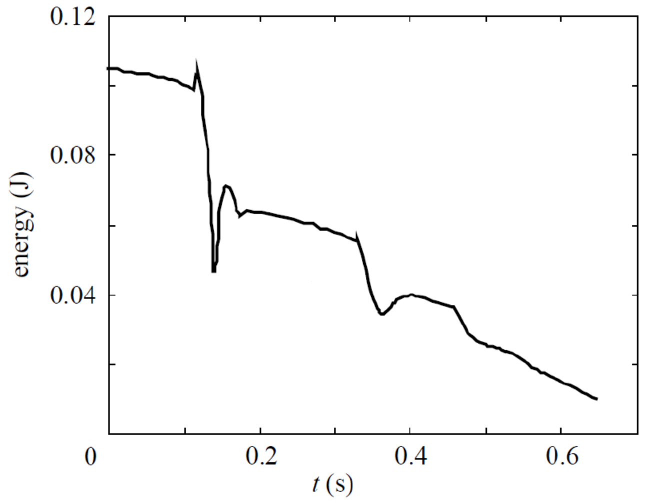


Figure 6.3: Total energy computed via analytical estimates using experimentally obtained λ and θ [55].

s and $t = 0.35$ s. This event corresponds with the fold reaching the support and reflecting. Afterwards, the total energy increases but does not reach the same level as before. This is the motivation for introducing damping in the analytical formulation for the dynamics of λ and θ.

It is hypothesized in [55] that energy is radiated through the support, accounting partially for the energy loss. Also, the authors note that the energy calculation involves various simplifications, which can result in false energy variation behaviors. This observed energy leak is the motivation for using damping or other mechanisms for energy loss in other numerical models for tape spring deployment presented in the literature. There is an apparent hysteresis effect present during quasi-static bending of tape springs [18], and structural damping has also been proposed as the source of energy leak. In [25], an analysis of tape spring hinges is presented. Numerical damping is used to represent the energy loss which is attributed to hysteresis damping. The energy leak can also be modeled via viscous damping or special boundary conditions. A finite element analysis of tape spring deployment using LS-DYNA has been presented in [60]. Dissipation effects were added using non-reflecting boundary conditions and nodal damping to improve slightly the agreement with experimental results from [55]. Composite tape spring hinges have been studied in ABAQUS [36]. Infinite elements were used at the support to account for dissipation at the boundaries. Bulk viscosity was also used as an additional dissipation mechanism. In summary, the main source of energy leak is currently unknown and the correct modeling procedure is uncertain.

6.2 Simulated experiment

General approach

In [55] the energy was computed using the previously discussed two degrees of freedom model as follows: the tape spring deployment was performed experimentally, the parameters λ and θ were measured from the experiment, and energies were computed using the analytical estimates from the two DOF model. The process is illustrated in Figure 6.4 This resulted in an energy leak from the experiments, which

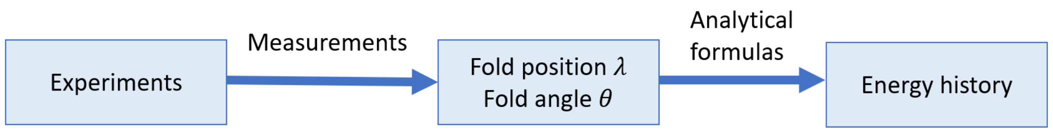


Figure 6.4: Process for computing total energy history in [55].

contrasts with the constant total energy observed in the simulations of chapter 5.

It is unclear if the energy leak occurs due to errors of the analytical estimates used for the energies, if there is intrinsic damping in the tape spring deployment experiment, or if both effects are present. We will separate both effects by removing any damping mechanisms and performing comparisons of energy histories for tape spring deployment with and without the analytical estimates for the energy. However, we cannot perform experiments without damping mechanisms, since those mechanisms are currently unknown. Therefore, we use finite element simulations to obtain "damping-free" deployment data. Afterwards, the high-dimensional data from the simulation can be used to obtain the true energy values via gauss integration over all the finite elements, as shown in Figure 6.5. Alternatively, the high-dimensional data can be used to measure the values of the fold position λ and fold angle θ, as shown in Figure 6.6. This will be denoted as a "simulated experiment", where we obtain measurements of λ and θ, just like in [55], and use them with the analytical formulas to obtain the energy history.

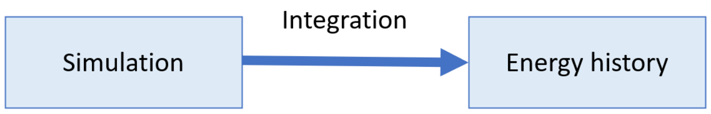


Figure 6.5: Process for computing total energy history using a high-fidelity finite element model.

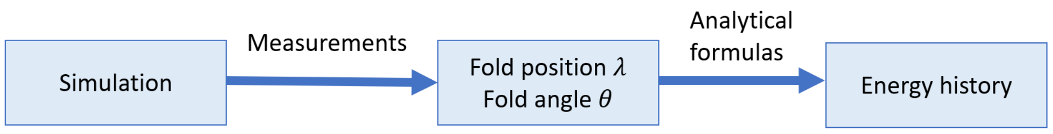


Figure 6.6: Process for computing total energy history in simulated experiment approach.

The energy results obtained directly from the high-fidelity finite element mesh and from the simulated experiment will be compared. This will provide an assessment of the accuracy of the analytical formulas presented in [55] for energy computations. Additionally, this detailed energy analysis will allow a deeper understanding of various physical features of the tape spring deployment.

Strain energy

The equation (6.2) is used to compute the strain energy. The fold angle θ identified from simulation is used to evaluate the analytical expression in the simulated experiment. The strain energy obtained from the finite element model is compared with the simulated experiment values in Figure 6.7a. It is observed that there is a large difference in the strain energy predicted. The main reason for the discrepancy is that the simulated experiment approach neglects various energy components from the tape spring:

- Strain energy of the transition region

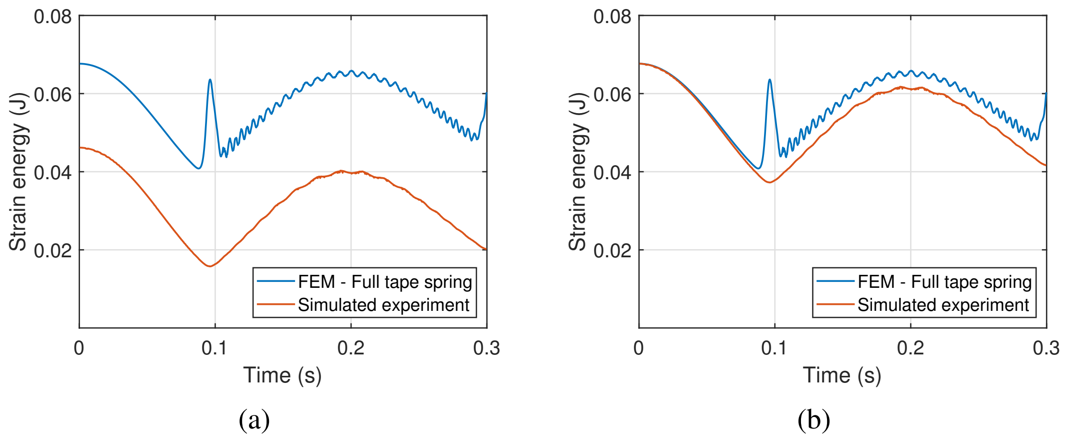


Figure 6.7: Comparison of strain energy variation during deployment from the finite element simulation and the simulated experiment (a) Raw values (b) After offset of simulated experiment values.

.

- Strain energy of the straight parts

This is due to the analytical model used in the simulated experiment. Accounting for the energy in the transition region analytically is complicated. However, as mentioned in [55], the transition region can be assumed to have constant strain energy. This would mean that the total strain energy would be off by a constant at all time, and hence applying a vertical offset, as shown in Figure 6.7b, would solve the discrepancy. However, there is still significant disagreement in the results. This is because when the fold gets close to the support, the transition region size changes. This fold-support interaction has been studied in [55], and the interaction approximately begins when the distance y from the support satisfies the following relation:

$$y < 1.5R\alpha^2 \tag{6.4}$$

Since the fold position can be "measured" from simulations, the time intervals where there is fold-support interaction can be identified. In Figure 6.8 these intervals are shown as shaded regions. It can be seen that even ignoring the time intervals where the transition region changes in size there is still discrepancy after the fold reflects off the support (after $t = 0.1$ s). Therefore, it is hypothesized that the strain energy of the straight parts, neglected during analytical calculations in the simulated experiment, is actually non-negligible. To verify this, we define a region centered around the fold which encompasses 25% of the tape spring length, denoted as the 25% region. Figure 6.9 illustrates this region for a particular fold position. The strain energy of the 25% region as predicted by the finite element model is shown

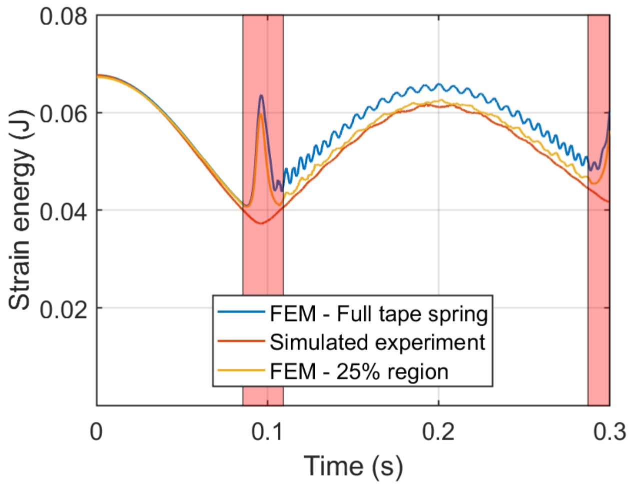


Figure 6.8: Strain energy variation, with regions of fold-support interaction shaded in red.

in Figure 6.8, where it can be observed that it matches closely with the simulated experiment results (after applying the offset). Therefore, the strain energy from the finite element model using the full tape spring is slightly higher than the simulated experiment results due to the contribution of the straight parts of the tape spring.

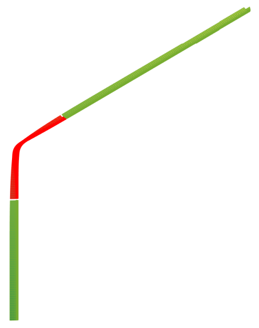


Figure 6.9: Region of tape spring centered on the fold for strain energy calculation with 25% of the length.

To study in detail the strain energy in the straight parts of the tape spring, the strain energy of each shell element is divided into membrane and bending terms. In addition, since the Kirchhoff-Love shell theory is used, for each term there are only 3 strain components: in the longitudinal axis, in the transverse (arc) direction, and due to an in-plane shear strain. Figure 6.10 shows the strain energy distribution in the straight parts of the tape spring at $t = 0.20$ s. It can be observed that the membrane strain in the longitudinal axis of the straight parts (which stretches/compresses the tape spring length-wise) is the main contributor to the strain energy of the straight parts and therefore to the discrepancy, accounting for 43 % of the strain energy discrepancy.

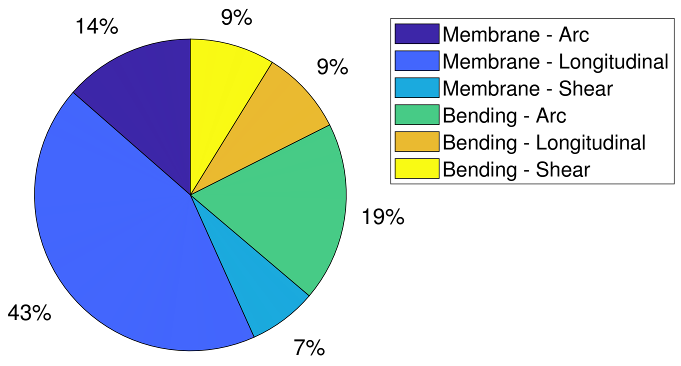


Figure 6.10: Breakdown of strain energy in straight parts of the tape spring at $t = 0.20\,\mathrm{s}$.

Therefore, in this section we have established that the analytical model does not correctly predict the strain energy, even after accounting for the transition region (as compared to the FEM simulation) since it neglects the membrane strain energy present in the straight parts of the tape spring.

Kinetic energy

Although the values of λ and θ can be "measured" from the simulation directly, the derivatives need to be calculated. In order to compute accurate derivatives it will be necessary to smoothen the data obtained from the fold identification algorithm. To do so, a cubic smoothing spline approach is used [50].

Given observations $\{x_i, y_i\}$ of a function f, estimates $\hat{f}(x_i)$ of the function at the observation points are defined by minimizing the following expression among the twice differentiable function space:

$$\sum_{i=1}^{n} \left\{y_i - \hat{f}(x_i)\right\} + w \int \hat{f}''(x)^2\,dx \qquad (6.5)$$

The smoothing parameter w controls the weight of the roughness penalty term. A value of $w = 5 \times 10^{-7}$ is used for fitting in the present work.

Figure 6.11 shows the raw and fitted data for the fold position λ and fold angle θ. The time derivatives of λ and θ can be computed from the smoothed data, and the kinetic energy can be evaluated using equation (6.3). Figure 6.12a shows the

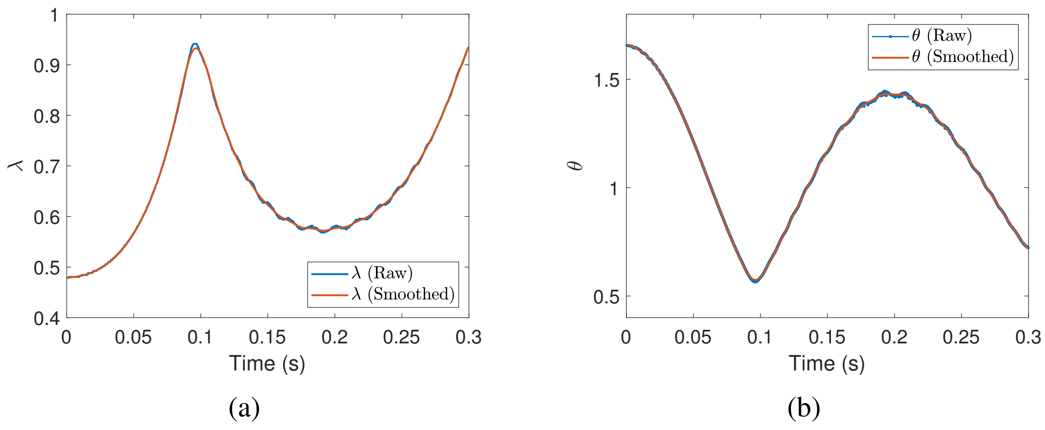


Figure 6.11: Data fitting of raw kinematic variables measurement from FEM simulation (a) Fold position λ (b) Fold angle θ.

.

kinetic energy variation during deployment for both the simulated experiment and the finite element model. It can be observed that initially there is good agreement.

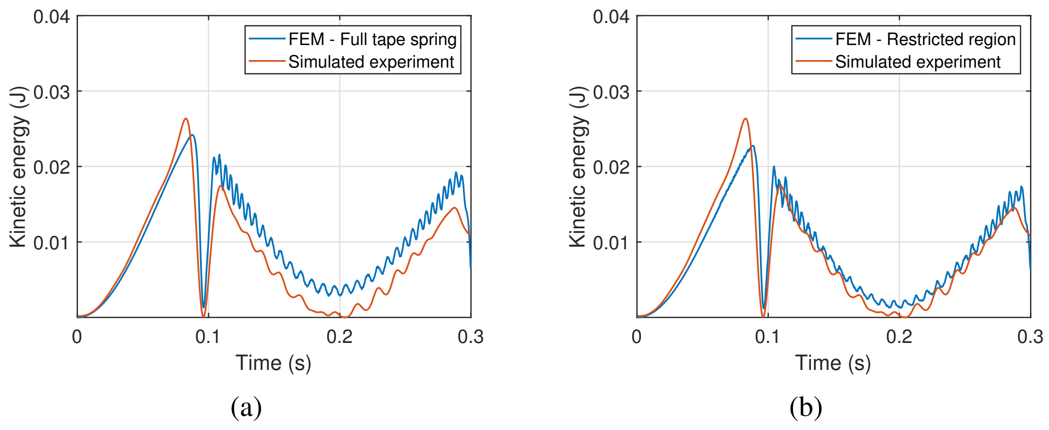


Figure 6.12: Comparison of kinetic energy during deployment predicted by the finite element model and the simulated experiment (a) FEM results use full tape model (b) FEM results restricted to top straight part of the tape spring and ignoring out-of-plane velocity components in the X axis.

However, when the fold approaches the support and then reflects (around $t = 0.1$ s), discrepancies arise. The FEM results for the full tape spring predicts higher kinetic energies consistently after the fold reflection. The reasons for this are as follows:

- Fold kinetic energy is neglected
- Small out-of-plane oscillations (in X axis)

The fold kinetic energy becomes particularly relevant due to the phenomenon of transverse fold oscillations, illustrated in Figure 6.13. A vertical red line is used for reference to help identify the oscillations, which occur within the plane where the tape spring centerline is located. It can be observed that the transverse oscillation results in minimal fold position and fold angle change. Therefore, the transverse fold oscillations cannot be accounted for in a model which uses only the fold position and fold angle as kinematic variables.

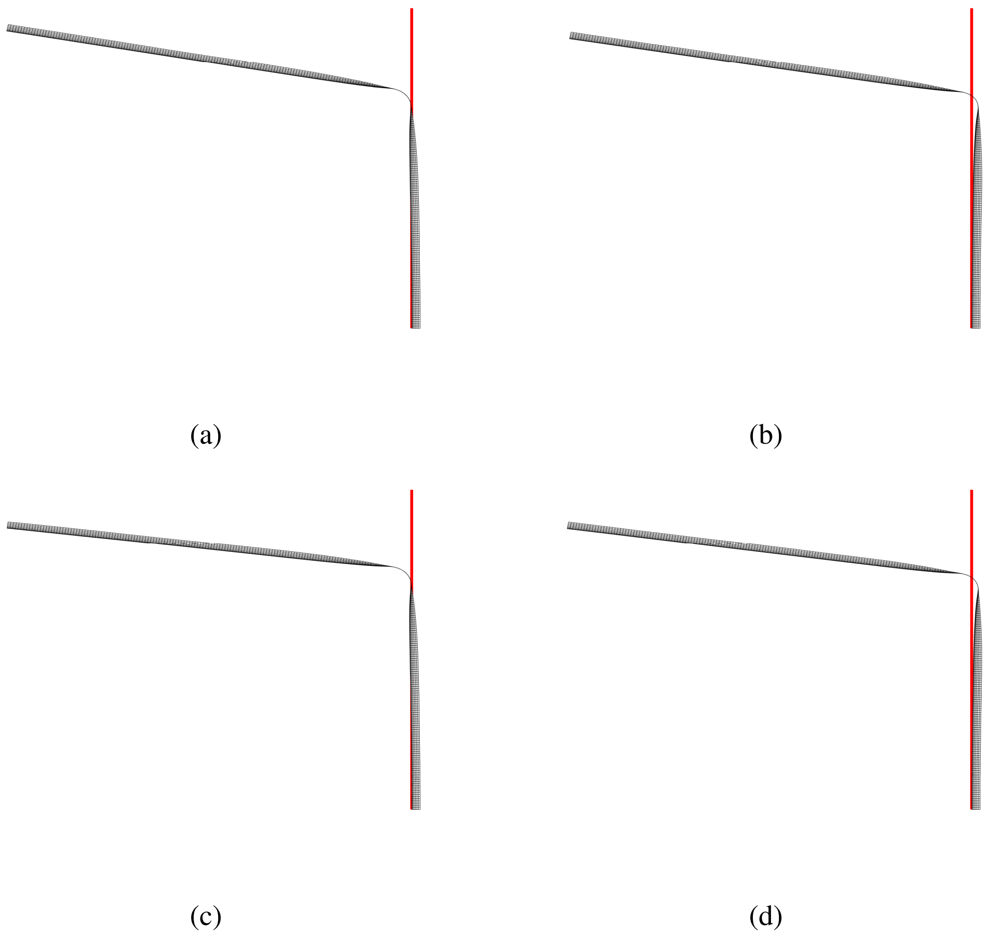


Figure 6.13: Tape spring configuration at two close times. The vertical red line is shown to illustrate the oscillations (a) $t = 0.174$ s (b) $t = 0.183$ s (c) $t = 0.191$ s (d) $t = 0.201$ s.

.

We can restrict the kinetic energy obtained from the simulation by neglecting the contribution due to the X axis velocity components and only considering the top straight part of the tape spring, neglecting the fold region. To define the fold region, we construct a region encompassing 6% of the length of the tape spring and centered on the fold position, and exclude it from computations. The region where the kinetic energy will be computed is shown in Figure 6.14. The resulting kinetic energy from the restricted FEM mesh is shown in Figure 6.12b. It can be observed that there is improved agreement between the analytical and simulation results after the fold reflects (after $t = 0.1$ s). However, the prediction of the peak in kinetic energy right before the fold reflection is slightly less accurate.

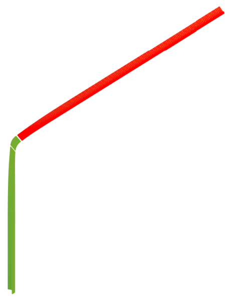


Figure 6.14: Region of tape spring centered on the fold for kinetic energy calculation (shown in red).

Therefore, in this section we have established that the analytical model does not correctly predict the kinetic energy (as compared to the FEM simulation) since it ignores the kinetic energy in the fold region and out-of-plane oscillations.

Total energy

By summing the energy components described previously the total energy predicted by the simulated experiment can be obtained. Figure 6.15 shows the total energy evolution over time. It can be observed that the total energy predicted decays after the fold reflection (after $t = 0.1$ s). Afterwards, the energy increases again but does not return to the initial value, and an energy leak is observed. This is the same behavior that was observed with experiments using the analytical formulation in [55]. As described in the previous section, this apparent energy dissipation is an artifact of the analytical model used. The energy is still present, but in the form of strain energy of the straight parts of the tape spring, kinetic energy of out-of-plane velocity components and kinetic energy due to transverse fold oscillations.

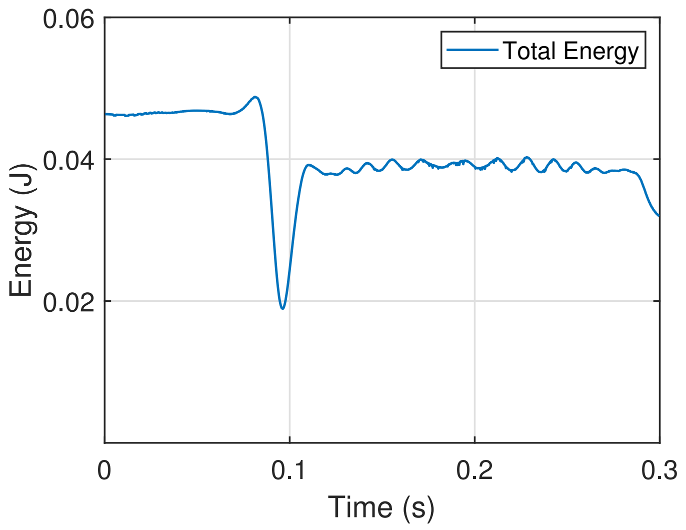


Figure 6.15: Total energy predicted by the simulated experiment.

6.3 Frequency analysis

Short-time Fourier transform

The transfer of energy from low to high-frequency dynamic components can be analyzed quantitatively by using the short-time Fourier transform (STFT). The STFT is used to describe the changes over time of the frequency content of a signal. Given a discrete time signal $x\left[n\right]$, the discrete STFT is expressed as:

$$X\left(t,\omega\right)=\sum_{n=-\infty}^{\infty}x\left[n\right]w\left[n-t\right]e^{-i\omega n} \tag{6.6}$$

where w is a window function which is nonzero only during a short period of time, usually a Gaussian window centered around zero. Therefore, the parameter t is the location of the center of the window w. This computation is repeated using different windows by modifying the parameter t.

The square of the magnitude of the STFT is used in a time-frequency representation of the signal known as a spectrogram:

$$\text{spectrogram}\left\{x\left[n\right]\right\}\left(t,\omega\right)=\left|X\left(t,\omega\right)\right|^2 \tag{6.7}$$

Application to finite element results

Among the kinematic quantities available during dynamic analysis, the velocity is considered the most relevant for frequency analysis due to the influence over the kinetic energy. Results from finite element analysis yield a velocity vector for every time step with components for every DOF. This is denoted in discrete form as $v_i\left[n\right]$, where n indicates the time step and i indicates the corresponding DOF. The STFT of every velocity component v_i is computed using equation (6.6):

$$V_i(t,\omega) = \sum_{n=-\infty}^{\infty} v_i[n]\, w[n-t]\, e^{-i\omega n} \tag{6.8}$$

To obtain a time-frequency representation that accounts for all the DOFs, the discrete STFT of every DOF is computed and the values are added together:

$$X(t,\omega) = \sum_{i=1}^{N} V_i(t,\omega) \tag{6.9}$$

where N is the number of degrees of freedom in the finite element model. Afterwards, the spectrogram representation is computed using equation (6.7).

Results

In order to obtain a better frequency resolution, the signal must be sampled over a longer period of time. For this purpose, the tape spring dynamic deployment was analyzed using an explicit solver as described in chapter 5 but over a period of time of 0.9 s instead of 0.3 s. The same material and geometric parameters were used. The energy evolution is shown in Figure 6.16. It can be observed that the tape spring energies oscillate initially with a period of around 0.2 s. However, after $t = 0.5$ s the oscillations become more complex. This is because the tape spring starts experiencing torsional deformation.

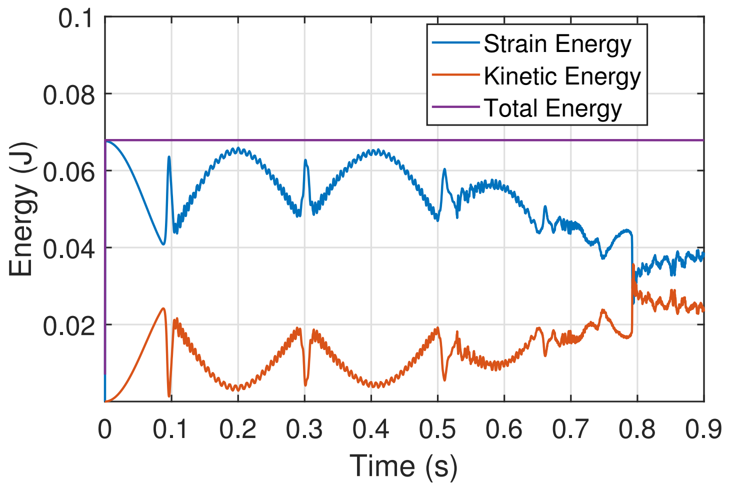


Figure 6.16: Kinetic, strain, potential and total energies during deployment of tape spring.

The spectrogram representation of the velocity vector was computed by using equations (6.7) - (6.9). The signal was divided into 8 sections of equal length, and a

50 % overlap between sections was defined. The sections were windowed using a Gaussian window. The resulting spectrogram is shown in Figure 6.17. It can be observed that initially the lower frequency components dominate the dynamics. However, over time the higher frequency content increases and cannot be neglected. This can be observed more clearly in Figure 6.18, which is a zoomed in plot of the

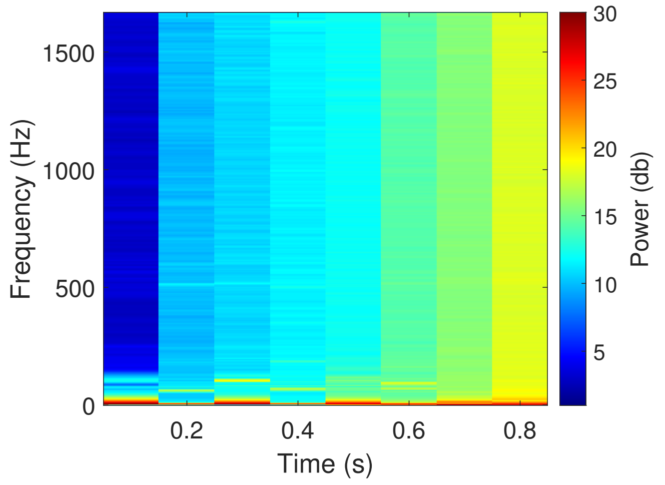


Figure 6.17: Spectrogram of the velocity vector which accounts for all the DOFs in the finite element model.

previous spectrogram. Initially the frequency content is mainly contained around 5 Hz, corresponding to the frequency of the main oscillation of the kinetic energy as observed from Figure 6.16 (with period of 0.2 s). However, as the time advances, the frequency content spreads out more evenly.

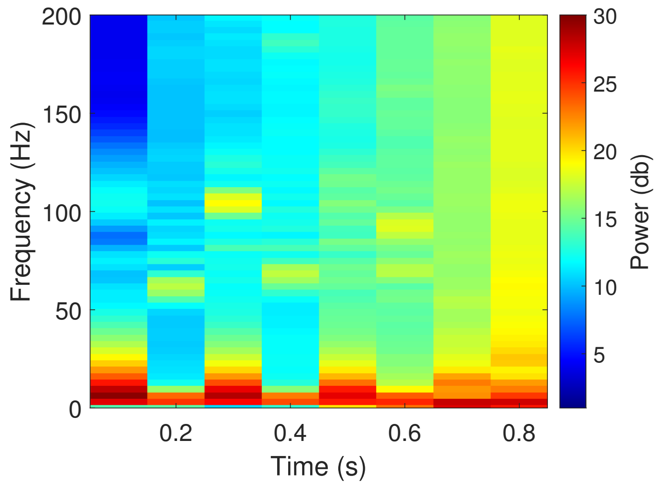


Figure 6.18: Zoomed in spectrogram of the velocity vector which accounts for all the DOFs in the finite element model.

6.4 Discussion

The finite element code for the analysis of tape spring deployment has been used to analyze the physics of the deployment. Results from the finite element code indicate there is no energy loss during the deployment. However, previous work showed an energy decay during tape spring deployment by using analytical estimates for the energies involved. It was unclear if this decay was due to damping within the experiments or inaccuracies of the analytical model. We used the finite element model to separate both effects due to its capability to perform the deployment without damping. A simulated experiment approach has been demonstrated.

Results indicate that the loss of energy is an artificial effect due to the simplfied analytical model used. In the high-fidelity finite element simulation the energy does not dissipate, but gets transformed into:

- Strain energy of the straight parts (mainly membrane strains along the tape spring length)
- Kinetic energy due to out-of-plane velocity components
- Kinetic energy due to transverse fold oscillation

Therefore, in order to get consistent energy during the deployment, a reduced order model for the tape spring deployment must be able to account for these effects. These results can be used to guide the construction of tape spring reduced order models.

Additionally, a spectral analysis of the velocity vector during tape spring deployment was performed using the short-time Fourier transform. It was observed that at the start of the deployment the lower frequency components dominate the analysis. However, as the deployment advances in time, the higher frequency components become more relevant. This indicates that for proper analysis of the tape spring deployment, the transfer of energy from low to high-frequency components must be accounted for.

Chapter 7

TAPE SPRING DEPLOYMENT - SPEEDING UP COMPUTATIONS

In this chapter, techniques for speeding up the dynamical analysis of the tape spring deployment problem are analyzed. The adaptive meshing procedure discussed in chapter 4, which was shown to be effective for speeding up opposite-sense bending of tape springs, is extended for dynamical analysis. Afterwards, the use of implicit solvers is analyzed. A novel locking behavior is shown to be present in implicit dynamical analysis of tape springs. A mixed formulation is extended for the dynamic case to alleviate the locking.

7.1 Adaptive meshing

Adaptive procedure

During deployment, the localized fold deformation requires a fine mesh for accurate analysis, and this fold moves along the tape spring length during the simulation. Therefore, using a uniform fine mesh results in inefficient simulations. A similar adaptive procedure as the one used in chapter 4 is used for the tape spring deployment problem. A significant difference between the opposite-sense bending and the deployment problems is that during opposite-sense bending the formation of the fold is part of the analysis, but during deployment the fold is already formed and moves within the tape spring. Therefore, it is more efficient to define a mesh which adapts based on the current fold position.

We define a non-uniform mesh as shown in Figure 7.1. The position of the fold is defined by the parameter p. A fine mesh centered around the fold is defined, with a width defined by the parameter d. Therefore, the non-dimensional parameters p/L and d/L define the structure of the mesh. The element size in the fine mesh is defined using a similar approach as in chapter 4 via the boundary layer effect size, as described by equation 4.42. The element size in the coarse mesh is defined as four times larger than the element size in the fine mesh.

At each time step the fold position is calculated using the same procedure as in chapter 6, using equation 5.3. If the position of the fold changes (as measured within the discretisation limits) with respect to the previous time step, the mesh is

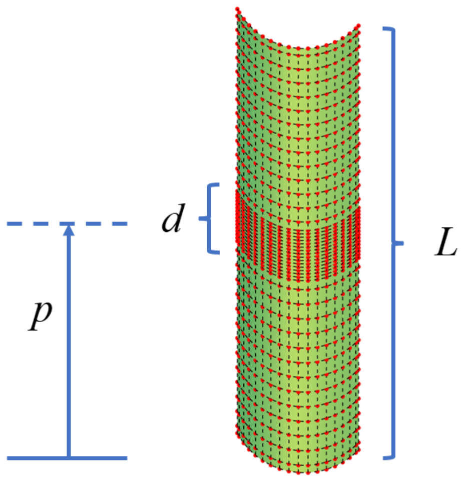


Figure 7.1: Adaptive mesh for dynamic simulations.

redefined based on the new fold position, and the kinematic variables are transferred to the new mesh. The pseudo-algorithm of the adaptive procedure is shown in (7.1).

Table 7.1: Adaptive meshing algorithm based on fold position.

```
For $N_t$ time steps
    Advance solution using explicit solver
    Compute new fold position
    If fold position changes
        Construct new mesh
        Transfer variables to new mesh
```

Transfer of kinematic variables

In static problems, after redefining a mesh the new values of the displacements are found by calculating a new equilibrium. This is possible because there is a unique solution for a given mesh and given loads applied. This was the approach used for adaptive meshing in chapter 4. However, in dynamic problems the correct approach is not evident. It was shown in [47] that if the mesh is redefined, the new solution becomes path-dependent on the meshes used. This is because the solution of a dynamic problem is dependent not only on the current mesh and the loads applied but also on all the previous kinematic states, which depends on the previous meshes used. Therefore, the new solution cannot be found as a result of an equilibrium step with just the new mesh and loads applied as inputs.

A solution for the transfer of kinematic variables between meshes in dynamic problems has been presented in [47], and the details are given below. We consider two meshes defined by interpolation functions ϕ_i and Φ_i. For each mesh we denote the displacement along a coordinate axis as u^* and U^*. The interpolation of the displacement for each mesh is defined as:

$$u^*(\mathbf{x}) = \sum_{i=1}^{n} \phi_i(\mathbf{x})\, u_i \tag{7.1}$$

$$U^*(\mathbf{x}) = \sum_{i=1}^{N} \Phi_i(\mathbf{x})\, U_i \tag{7.2}$$

where $\mathbf{x}$ is a position vector, and $(\mathbf{u}, \mathbf{U})$ are the vectors of nodal displacements for each of the meshes, which contain the values (u_i, U_i) respectively. We define auxiliary matrices for the mesh transfer as:

$$\mathbf{M}_{ij} = \int_{\Omega} \Phi_i \Phi_j \, d\Omega \tag{7.3}$$

$$\mathbf{m}_{ij} = \int_{\Omega} \Phi_i \phi_j \, d\Omega \tag{7.4}$$

where Ω is the domain where the mesh is defined. Given the current nodal displacements $\mathbf{u}$, the current shape functions ϕ_i and the shape functions of the new mesh Φ_i, the new nodal displacements $\mathbf{U}$ corresponding to the new mesh are given by:

$$\mathbf{U} = \mathbf{M}^{-1}\mathbf{m}\mathbf{u} \tag{7.5}$$

It is shown in [47] that for linear shape functions the transfer of variables reduces trivially to direct interpolation within the mesh. However, for more complex shape functions the solution is not obvious, and equation 7.5 must be used.

Numerical results

The adaptive meshing procedure discussed with the corresponding variable transfer is used to solve the tape spring deployment problem. The folding steps are performed as in chapter 5, and the dynamic deployment step is performed using the new approach with an explicit solver. The tape spring properties are the same as in chapter V. The fine mesh width parameter is set to $d/L = 0.12$, which results in using 25% of the number of degrees of freedom compared to a uniform mesh. The mesh at the initial state of the deployment is shown in Figure 7.2. The energy history is shown in Figure 7.3. It can be observed that the total energy oscillates significantly and does not behave correctly. Therefore, the adaptive mesh used is too coarse to simulate the deployment accurately.

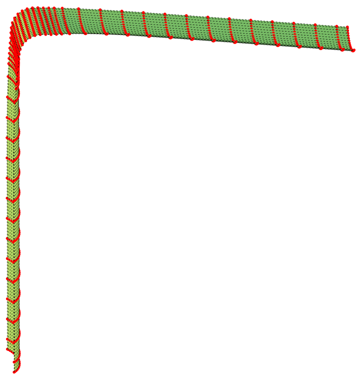


Figure 7.2: Adaptive mesh for tape spring deployment simulation with $d/L = 0.12$.

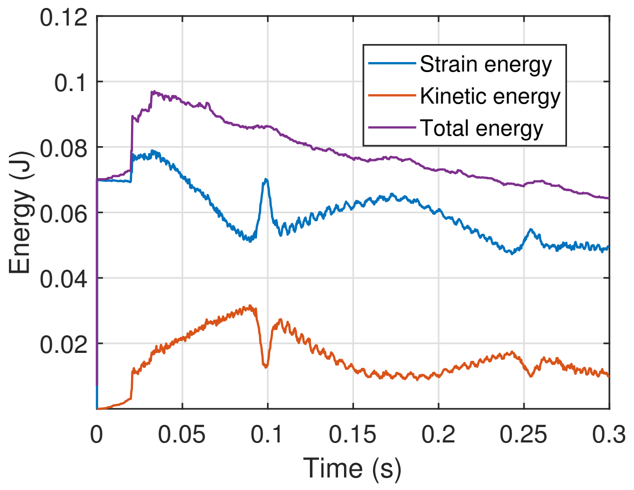


Figure 7.3: Energy history for tape spring deployment using adaptive meshing procedure.

The results are refined by increasing the width of the fine mesh region, setting $d/L = 0.3$, which results in using 49% of the number of degrees of freedom compared to a uniform mesh. The mesh at the initial state of the deployment is shown in Figure 7.4. The energy history is shown in Figure 7.5. It can be observed that the total energy does not oscillate. However, there is a decay in the total energy after the fold reflects ($t = 0.1$ s). This is an undesirable result, since we expect that the total energy remains constant due to the absence of damping. It is suspected that, despite their apparent geometric simplicity, the straight parts of the tape spring cannot be modeled accurately using coarse meshes. As shown in chapter 6, the dynamic buckling effects in the straight parts of the tape spring appear after the fold reflection ($t = 0.1$ s). Given that the present energy decay is also observed after ($t = 0.1$ s), causation is suspected. Resolving the dynamic buckling effects would require a fine mesh in a section of the straight parts of the tape spring.

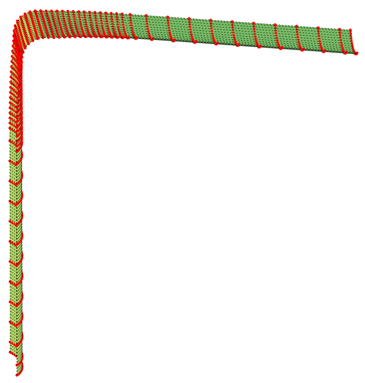


Figure 7.4: Adaptive mesh for tape spring deployment simulation with $d/L = 0.3$.

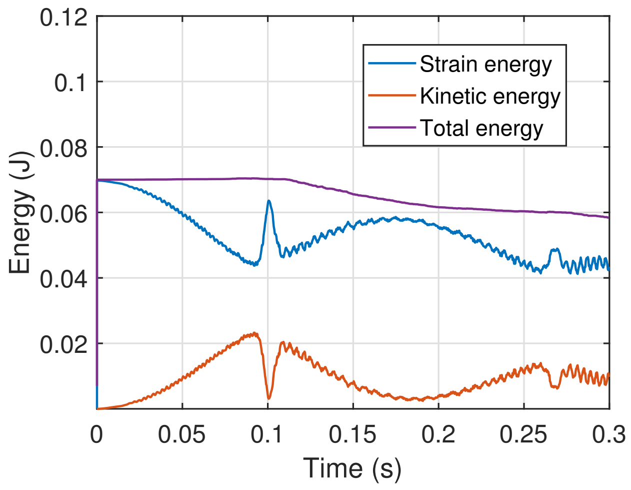


Figure 7.5: Energy history for tape spring deployment using adaptive meshing procedure.

Although more accurate results would be expected by increasing the fine mesh width parameter d/L, higher values result in less efficient computations since the fine mesh region encompasses most of the tape spring and the mesh becomes very similar to a uniform fine mesh.

7.2 Implicit solvers

Generalized alpha method

The tape spring deployment problem has been analyzed using explicit solvers in chapter 5. However, it is possible to use implicit solvers to compute the solution. This has the advantage of allowing the use of larger time steps, being potentially more computationally efficient. An important issue for implicit solvers within structural dynamics is the presence of high-frequency dynamic components in the system as artifacts of the discretisation. Since large time steps can be used in implicit solvers, the high-frequency motion is often not correctly resolved. The implicit solver must be able to dampen out the high-frequency components, otherwise the presence of the artificial high-frequency dynamics will difficult the convergence and greatly reduce the efficiency of the solver. We will use the generalized-alpha method [20] as the implicit solver due to its capability to dampen motion at high frequencies while reproducing low-frequency dynamics accurately. A brief description of the method is given below.

We recall that the standard structural dynamics equation without damping is given by:

$$\mathbf{M}\mathbf{a}^{i+1} + \mathbf{f}^{int}\left(\mathbf{u}^{i+1}\right) - \mathbf{F}^{i+1} = \mathbf{0} \tag{7.6}$$

where $\mathbf{M}$ is the mass matrix (assumed constant), $\mathbf{f}$ is the vector of internal forces, $\mathbf{F}$ is the vector of external forces, and $\left(\mathbf{u}^{i+1}, \mathbf{v}^{i+1}, \mathbf{a}^{i+1}\right)$ are the displacement, velocity and acceleration vectors respectively. The system is closed by adding the Newmark approximations [40], where (β, γ) are the Newmark parameters:

$$\begin{cases} \mathbf{u}^{i+1} = \mathbf{u}^{i} + \Delta t \mathbf{v}^{i} + (\Delta t)^2 \left(\left(\frac{1}{2} - \beta\right)\mathbf{a}^{i} + \beta \mathbf{a}^{i+1}\right) \\ \mathbf{v}^{i+1} = \mathbf{v}^{i} + \Delta t \left((1-\gamma)\,\mathbf{a}^{i} + \gamma \mathbf{a}^{i+1}\right) \end{cases} \tag{7.7}$$

The generalized-alpha method is based on modifying equation (7.6) by evaluating various quantities at different times as follows:

$$\mathbf{M}\mathbf{a}^{i+1-\alpha_m} + \mathbf{f}\left[\mathbf{u}^{i+1-\alpha_f}\right] - \mathbf{F}^{i+1-\alpha_f} = \mathbf{0} \tag{7.8}$$

where α_f, α_m are parameters of the generalized alpha method. The quantities evaluated at intermediate times are evaluated using linear interpolation as follows:

$$\begin{cases} \mathbf{u}^{i+1-\alpha_f} = (1-\alpha_f)\mathbf{u}^{i+1} + \alpha_f \mathbf{u}^{i} \\ \mathbf{a}^{i+1-\alpha_m} = (1-\alpha_m)\mathbf{a}^{i+1} + \alpha_m \mathbf{a}^{i} \end{cases} \tag{7.9}$$

The choice of α_f and α_m determines the numerical damping introduced in the model as well as various stability and accuracy properties of the method. It is shown in [20] that in order to have stability and second-order accuracy the variables $\left(\beta, \gamma, \alpha_f, \alpha_m\right)$ must be related in such a way that they can be expressed as a function of a single variable ρ_∞. This variable is related to the damping introduced for dynamic motion components with infinite frequency. Therefore, ρ_∞ directly controls the amount of numerical damping in the model, with higher values resulting in less damping.

Numerical results

The implicit solver was used for simulation of the tape spring deployment problem. The folding and deployment steps are the same as those described in chapter 5 for the isogeometric code with the explicit solver. The same tape spring properties are used. For the generalized alpha method, we use $\rho_\infty = 0.85$ and $\Delta t = 10^{-5}s$. The energy history is shown in Figure 7.6. The simulation stops slightly before $t = 0.1$ s due to failure of convergence.

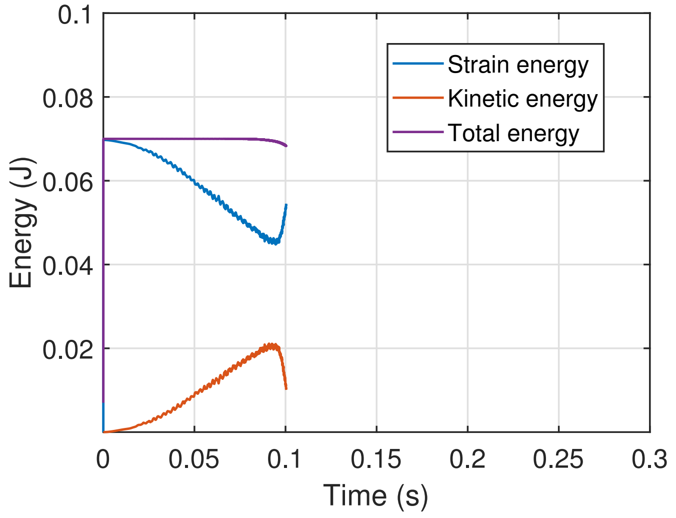


Figure 7.6: Energy history using implicit generalized alpha solver with $\rho_\infty = 0.85$ and $\Delta t = 10^{-5}$ s.

In order to improve the convergence behavior the time step can be reduced, although this increases the computational cost significantly. This is shown in Figure 7.7, where the time step was chosen as $\Delta t = 10^{-6}$ s. It can be seen that the simulation can be completed up to $t = 0.3$ s without convergence failure. However, there is a decay of the total energy of the system. This is due to the numerical damping introduced, which dampens out high-frequency motion in the system. Therefore, the high-frequency dynamic components during tape spring deployment contain a significant part of the energy in the system. Ideally the numerical damping deals with the high-frequency components which are artifacts of the discretisation and not the high-frequency components which are truly present and part of the deployment problem. Therefore, we seek to obtain constant total energy in the system by adjusting the solver parameters.

The convergence behavior can also be improved by reducing ρ_∞, which introduces higher damping of high-frequency motion. Although this significantly speeds up

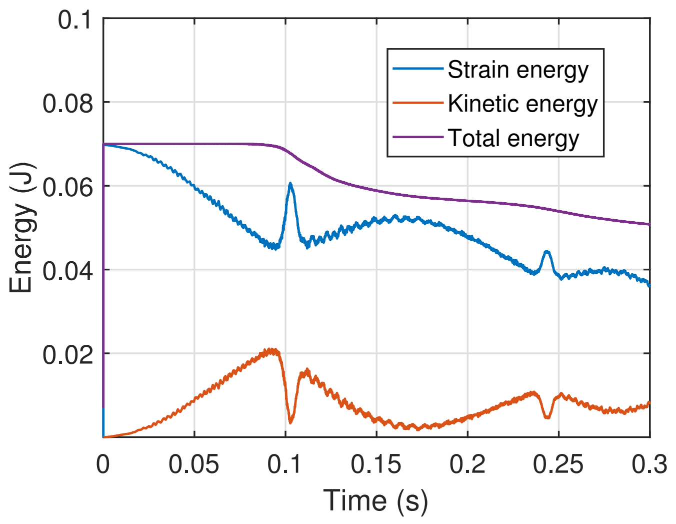


Figure 7.7: Energy history using implicit generalized alpha solver with $\rho_\infty = 0.85$ and $\Delta t = 10^{-6}$ s.

convergence and reduces the computational cost, the total energy decay is even larger. Therefore, this option is not ideal for our purposes. Instead we want to reduce ρ_∞ such that the energy decay is smaller. Since there will be more high-frequency components activated, the convergence will deteriorate. Therefore this should be accompanied with a smaller time step Δt. Unfortunately, this results in a very high computational cost. Although it is possible to switch to an explicit solver as it was done in chapter 5, it will be shown in the next section that the difficulties present in the implicit solver are related to a novel locking effect.

7.3 Interpolation locking

Introduction

The Newton-Raphson method has been used extensively for analyzing structures using the finite element method, as detailed in [3]. The equilibrium equations are solved by a sequence of fixed increments. This method can occasionally fail if a large step increment is used, requiring reducing the increment and increasing the computational cost. It is shown in [23] that the failures in convergence can be related to a locking phenomenon, caused by the difference between the true jacobian and the estimate used in the iterative process. The locking phenomenon, hereafter denoted as "interpolation locking", appears in slender structures, such as thin shells [34]. Unlike traditional locking effects, this locking does not cause a reduction in accuracy, but greatly increases the number of iterations for convergence in nonlinear problems solved by iteration. The locking effect appears due to the high flexuralaxial stiffness ratio and this requires taking smaller step increments as the structure gets thinner.

The importance of this locking phenomenon depends on the nonlinearity of the problem on the interpolation variables. In particular, displacement-based formulations can be very sensitive to interpolation locking. Mixed formulations interpolate both the stress and the displacements, and have been shown to greatly reduce the influence of the locking effect [33], although with increased computational cost. The Mixed Interpolation Method (MIP) has been shown to be an effective and efficient way to alleviate interpolation locking in thin structures. By using condensation of variables, this method avoids the additional cost of standard mixed formulations [23]. The interpolation locking has been mainly studied more for path-following static problems using the arc-length method. In dynamic problems the jacobian is modified by the addition of a mass matrix [3], and it could be argued that this addition makes the iterative process less prone to locking, but this is unclear.

In this section, the presence of interpolation locking in the solution of the tape spring dynamic deployment problem with implicit solvers is studied. It is shown that application of standard implicit solvers result in severe interpolation locking for the tape spring deployment simulation. Afterwards, the Mixed Interpolation Point (MIP) method is extended to the dynamic case as a method for alleviating the interpolation locking while maintaining computational efficiency.

Locking in static problems

The description of the locking phenomenon in this section will follow the presentation given in [23]. Using finite element discretization, the equilibrium of an elastic structure can be expressed by the following nonlinear equation:

$$\mathbf{r}\left[\mathbf{u}\right] = \mathbf{f}\left[\mathbf{u}\right] - \mathbf{F} = \mathbf{0} \tag{7.10}$$

where $\mathbf{r}$ is the residual vector, $\mathbf{u}$ is the displacement vector, $\mathbf{f}$ is the vector of internal forces and $\mathbf{F}$ is the vector of external forces. Given an initial guess $\mathbf{u}_0$, the nonlinear equation (7.10) is solved via the Newton-Raphson iteration method. The jacobian matrix of equation (7.10) is defined as:

$$\mathbf{J} = \frac{d\mathbf{r}}{d\mathbf{u}} \tag{7.11}$$

A sequence of increment estimates $\Delta\mathbf{u}_j$ are computed using the jacobian matrix $\mathbf{J}_j$:

$$\mathbf{J}_j \Delta\mathbf{u}_j = -\mathbf{r}_j \tag{7.12}$$

where

$$\mathbf{J}_j = \mathbf{J}\left[\mathbf{u}_j\right] \tag{7.13}$$

$$\Delta\mathbf{u}_j = \mathbf{u}_{j+1} - \mathbf{u}_j \tag{7.14}$$

$$\mathbf{r}_j = \mathbf{r}\left[\mathbf{u}_j\right] \tag{7.15}$$

We can define a secant jacobian matrix $\bar{\mathbf{J}}_j$ [23] as the average jacobian matrix along the increment path:

$$\bar{\mathbf{J}}_j = \int_0^1 \mathbf{J}\left[\mathbf{u}_j + t\left(\mathbf{u}_{j+1} - \mathbf{u}_j\right)\right] dt \tag{7.16}$$

From the mean value theorem and (7.11) it is evident that this matrix satisfies the following condition:

$$\mathbf{r}_{j+1} - \mathbf{r}_j = \bar{\mathbf{J}}_j\left(\mathbf{u}_{j+1} - \mathbf{u}_j\right) \tag{7.17}$$

Substituting (7.12) in (7.17):

$$\mathbf{r}_{j+1} = \left(\mathbf{I} - \bar{\mathbf{J}}_j\mathbf{J}_j^{-1}\right)\mathbf{r}_j \tag{7.18}$$

The iterative scheme given in (7.12) is guaranteed to converge if the norm of the residual $\mathbf{r}$ decreases at each iteration. Therefore from (7.18) we observe that the iterative scheme has guaranteed convergence if the following condition is satisfied:

$$\rho\left(\mathbf{I} - \bar{\mathbf{J}}_j\mathbf{J}_j^{-1}\right) < 1 \tag{7.19}$$

where $\rho(.)$ is the spectral radius of matrix (.). If the increment $\Delta\mathbf{u}_j$ is very small then the secant jacobian matrix through the increment step is almost the same as the tangent jacobian matrix at the initial point. Therefore if $\left\|\Delta\mathbf{u}_j\right\| \approx 0$ then $\bar{\mathbf{J}}_j$ and $\mathbf{J}_j$ are almost equal to each other and $\rho\left(\mathbf{I} - \bar{\mathbf{J}}_j\mathbf{J}_j^{-1}\right) \approx 0$. This case corresponds to fast convergence of the iterative scheme when a small increment is used.

However, for large increments the secant and tangent jacobian matrices can differ considerably such that the condition (7.21) may not be satisfied and convergence is not guaranteed. This will result in convergence difficulties, requiring use of a smaller increment. This phenomenon is denoted as interpolation locking in [23].

We can consider configuration-independent external forces, that is, external forces $\mathbf{F}$ which do not depend on the current displacement $\mathbf{u}$. Therefore, from (7.10) and (7.11) we can identify $\mathbf{J}$ as the classic tangent stiffness matrix $\mathbf{K}$, where:

$$\mathbf{K} = \frac{d\mathbf{f}}{d\mathbf{u}} \tag{7.20}$$

Similarly, the secant jacobian $\bar{\mathbf{J}}_j$ becomes the secant stiffness matrix $\bar{\mathbf{K}}_j$, that is, the average stiffness matrix through the step increment. Substituting the jacobians for tangent matrices in (7.19) we get:

$$\rho\left(\mathbf{I} - \bar{\mathbf{K}}_j \mathbf{K}_j^{-1}\right) < 1 \tag{7.21}$$

Similar conclusions drawn before for the secant and tangent jacobian matrixes can be stated for the secant and tangent stiffness matrixes.

Locking in dynamic problems

The interpolation locking has not been studied previously for dynamic problems. Therefore, in this section we extend the previously discussed locking effect for dynamic equations. The solution at the time increment $i+1$ for a dynamic system involves solving for the displacement vector $\mathbf{u}^{i+1}$, the velocity $\mathbf{v}^{i+1}$ and the acceleration $\mathbf{a}^{i+1}$. Following the generalized alpha time-stepping procedure [20] and considering no damping, the following balance equation is solved:

$$\mathbf{r}\left[\mathbf{u}^{i+1-\alpha_f}, \mathbf{a}^{i+1-\alpha_m}\right] = \mathbf{M}\mathbf{a}^{i+1-\alpha_m} + \mathbf{f}\left[\mathbf{u}^{i+1-\alpha_f}\right] - \mathbf{F}^{i+1-\alpha_f} = \mathbf{0} \tag{7.22}$$

where α_f, α_m are parameters of the generalized alpha method, and $\mathbf{M}$ is the mass matrix which is assumed constant. Configuration-independent structures are considered in the following. The system is closed by defining the kinematic quantities at intermediate times as:

$$\begin{cases} \mathbf{u}^{i+1-\alpha_f} = (1-\alpha_f)\mathbf{u}^{i+1} + \alpha_f \mathbf{u}^i \\ \mathbf{a}^{i+1-\alpha_m} = (1-\alpha_m)\mathbf{a}^{i+1} + \alpha_m \mathbf{a}^i \end{cases} \tag{7.23}$$

and adding the Newmark approximations (where β, γ are Newmark parameters):

$$\begin{cases} \mathbf{u}^{i+1} = \mathbf{u}^i + \Delta t \mathbf{v}^i + (\Delta t)^2 \left(\left(\frac{1}{2} - \beta\right)\mathbf{a}^i + \beta \mathbf{a}^{i+1}\right) \\ \mathbf{v}^{i+1} = \mathbf{v}^i + \Delta t\left((1-\gamma)\,\mathbf{a}^i + \gamma \mathbf{a}^{i+1}\right) \end{cases} \tag{7.24}$$

To solve equation (7.22) for $\mathbf{a}^{i+1}$ we define the jacobian matrix $\mathbf{J}$ as:

$$\mathbf{J} = \frac{d\mathbf{r}}{d\mathbf{a}^{i+1}} \tag{7.25}$$

We construct a sequence of estimates for the increments $\Delta \mathbf{a}_j^{i+1}$ as:

$$\mathbf{J}_j \Delta \mathbf{a}_j^{i+1} = -\mathbf{r}_j \tag{7.26}$$

where

$$\mathbf{J}_j = \mathbf{J}\left[\mathbf{u}_j^{i+1-\alpha_f}, \mathbf{a}_j^{i+1-\alpha_m}\right] \tag{7.27}$$

$$\Delta \mathbf{a}_j^{i+1} = \mathbf{a}_{j+1}^{i+1} - \mathbf{a}_j^{i+1} \tag{7.28}$$

$$\mathbf{r}_j = \mathbf{r}\left[\mathbf{u}_j^{i+1-\alpha_f}, \mathbf{a}_j^{i+1-\alpha_m}\right] \tag{7.29}$$

We can define a secant jacobian matrix using conditions similar to (7.17) and (7.16):

$$\mathbf{r}_{j+1} - \mathbf{r}_j = \bar{\mathbf{J}}_j \left(\mathbf{a}_{j+1} - \mathbf{a}_j\right) \tag{7.30}$$

$$\bar{\mathbf{J}}_j = \int_0^1 \mathbf{J}\left[\mathbf{a}_j + t\left(\mathbf{a}_{j+1} - \mathbf{a}_j\right)\right] dt \tag{7.31}$$

Due to the similarities of equations (7.26) and (7.12), the convergence of the iterative process is guaranteed if the condition (7.19) is satisfied. However, the jacobian $\mathbf{J}$ is different for the dynamic case. Using (7.25), (7.22), (7.23), (7.24) and (7.20) in (7.27), we obtain:

$$\mathbf{J}_j = (1 - \alpha_m)\,\mathbf{M} + \left(1 - \alpha_f\right)\beta\,(\Delta t)^2\,\mathbf{K}\left[\mathbf{u}_j^{i+1-\alpha_f}\right] \tag{7.32}$$

We observe that the jacobian for the dynamic case consists of the standard stiffness matrix but modified by the addition of the mass matrix. Therefore, the conditions in which interpolation locking appears, that is, when the secant and tangent jacobians are significantly different, are not the same as in static problems. Substituting (7.32) and (7.29) into (7.26):

$$\Delta \mathbf{a}_j^{i+1} = -\left((1 - \alpha_m)\,\mathbf{M} + \left(1 - \alpha_f\right)\beta\,(\Delta t)^2\,\mathbf{K}\left[\mathbf{u}_j^{i+1-\alpha_f}\right]\right)^{-1} \mathbf{r}\left[\mathbf{u}_j^{i+1-\alpha_f}, \mathbf{a}_j^{i+1-\alpha_m}\right] \tag{7.33}$$

Equation (7.33) is used to compute the increments at each iteration.

Mixed formulation framework

The strain-displacement matrix $\mathbf{B}_e$ of an element e is defined as:

$$\mathbf{B}_e = \frac{\partial \boldsymbol{\epsilon}_e}{\partial \mathbf{u}_e} \tag{7.34}$$

where $\boldsymbol{\epsilon}_e$ is the strain vector within the element and $\mathbf{u}_e$ are the element nodal displacements. The internal forces at the element level are obtained by gauss integration as [3]:

$$\mathbf{f}_e = \sum_k \mathbf{B}_e^{kT}\left[\mathbf{u}_e\right]\boldsymbol{\sigma}_e^k w^k \tag{7.35}$$

where the k is the gauss point index, $\mathbf{B}_e^k$ is the strain-displacement matrix of the element e evaluated at the gauss point k, $\boldsymbol{\sigma}_e^k$ are the stresses of the element e at the gauss point k, and w^k are the gauss weights.

Considering elastic materials, we have a constitutive equation which for each element uses an elastic matrix $\mathbf{C}_e^k$ and the strain vector $\boldsymbol{\epsilon}_e^k$, both evaluated at the gauss point k. The strains are a function of the nodal displacements of the element $\mathbf{u}_e$:

$$\boldsymbol{\sigma}_e^k = \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k \left[\mathbf{u}_e\right] \tag{7.36}$$

The standard procedure is to use (7.36) to evaluate (7.35), making the internal forces only a function of displacements:

$$\mathbf{f}_e \left[\mathbf{u}_e\right] = \sum_k \mathbf{B}_e^{kT} \left[\mathbf{u}_e\right] \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k \left[\mathbf{u}_e\right] w^k \tag{7.37}$$

By assembling over all the elements we obtain the global internal forces vector given as follows:

$$\mathbf{f}\left[\mathbf{u}\right] = \sum_e \sum_k \mathbf{V}_e \mathbf{B}_e^{kT} \left[\mathbf{u}\right] \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k \left[\mathbf{u}\right] w^k \tag{7.38}$$

where $\mathbf{V}_e$ is a boolean matrix with zeros and ones as entries that maps between element and global indexes. We get the stiffness matrix (7.20) by differentiating with respect to the global displacements and substituting (7.36) and (7.34):

$$\mathbf{K}\left[\mathbf{u}\right] = \sum_e \sum_k \mathbf{V}_e \left(\frac{\partial \mathbf{B}_e^{kT}}{\partial \mathbf{u}} \left[\mathbf{u}\right] \boldsymbol{\sigma}_e^k \left[\mathbf{u}\right] + \mathbf{B}_e^{kT} \left[\mathbf{u}\right] \mathbf{C}_e^k \mathbf{B}_e^k \left[\mathbf{u}\right] \right) \mathbf{V}_e^T w^k \tag{7.39}$$

Instead of following the standard approach, a mixed formulation can be developed by considering in (7.35) that the stresses $\boldsymbol{\sigma}$ are an independent variable. Therefore, the the internal forces of the element are now a function of displacements $\mathbf{u}_e$ and stresses $\boldsymbol{\sigma}_e$:

$$\hat{\mathbf{f}}_e \left[\mathbf{u}_e, \boldsymbol{\sigma}_e^k\right] = \sum_k \mathbf{B}_e^{kT} \left[\mathbf{u}_e\right] \boldsymbol{\sigma}_e^k w^k \tag{7.40}$$

By assembling over all the elements we obtain the global internal forces vector given as follows:

$$\hat{\mathbf{f}}\left[\mathbf{u}, \boldsymbol{\sigma}_e^k\right] = \sum_e \sum_k \mathbf{V}_e \mathbf{B}_e^{kT} \left[\mathbf{u}\right] \boldsymbol{\sigma}_e^k w^k \tag{7.41}$$

The notation used emphasizes that the vector $\mathbf{f}$ is a function of the global displacement vector $\mathbf{u}$ and the stresses $\boldsymbol{\sigma}_e^k$ at each gauss point of each element.

Using (7.41) instead of (7.37) during the assembly of the governing equations given in (7.22) adds the stresses of each element at each gauss point $\boldsymbol{\sigma}_e^k$ as new variables. Therefore, a new equation must be added, namely the constitutive equation (7.36) at the time increment $\left(i + 1 - \alpha_f\right)$ for each element and gauss point, to close the system:

$$\left\{ \begin{array}{c} \hat{\mathbf{r}} = \mathbf{M}\mathbf{a}^{i+1-\alpha_m} + \hat{\mathbf{f}}\left(\mathbf{u}^{i+1-\alpha_f}, \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}\right) - \mathbf{F}^{i+1-\alpha_f} = \mathbf{0} \\ \mathbf{g}_e^k = \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f} - \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k \left[\left(\mathbf{u}_e\right)^{i+1-\alpha_f}\right] = \mathbf{0} \end{array} \right. \tag{7.42}$$

where $\hat{\mathbf{r}}$ and $\mathbf{g}_e^k$ are the residuals of the system of equations, dependent on both displacements and stresses of the elements. The hat notation is used to differentiate from the residual (7.22) where the element stresses are not independent variables. The derivatives of (7.41) will be used in the following section:

$$\begin{cases} \dfrac{\partial \hat{\mathbf{f}}}{\partial \mathbf{u}} = \sum_e \sum_k \mathbf{V}_e \dfrac{\partial \mathbf{B}_e^{kT}}{\partial \mathbf{u}_e} \boldsymbol{\sigma}_e^k w^k \mathbf{V}_e^T \\ \dfrac{\partial \hat{\mathbf{f}}}{\partial \boldsymbol{\sigma}_e^k} = \mathbf{V}_e \mathbf{B}_e^{kT} w^k \end{cases} \tag{7.43}$$

We also define a stiffness matrix similar to (7.39) but where the stresses are computed independent from the displacements:

$$\hat{\mathbf{K}}\left[\mathbf{u}, \boldsymbol{\sigma}_e^k\right] = \sum_e \sum_k \mathbf{V}_e \left(\frac{\partial \mathbf{B}_e^{kT}}{\partial \mathbf{u}} [\mathbf{u}]\, \boldsymbol{\sigma}_e^k + \mathbf{B}_e^{kT} [\mathbf{u}]\, \mathbf{C}_e^k \mathbf{B}_e^k [\mathbf{u}] \right) \mathbf{V}_e^T w^k \tag{7.44}$$

The nonlinear system of equations (7.42) can be solved for $(\mathbf{u}^{i+1-\alpha_f}, \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f})$ using the Newton-Raphson method. We construct a sequence of estimates for the increments by solving the following system of equations:

$$\frac{\partial \hat{\mathbf{r}}}{\partial \mathbf{a}^{i+1}} \Delta \mathbf{a}_j^{i+1} + \sum_e \sum_k \frac{\partial \hat{\mathbf{r}}}{\partial \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}} \Delta \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} = -\hat{\mathbf{r}} \tag{7.45}$$

$$\frac{\partial \mathbf{g}_e^k}{\partial \mathbf{a}^{i+1}} \Delta \mathbf{a}_j^{i+1} + \frac{\partial \mathbf{g}_e^k}{\partial \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}} \Delta \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} = -\mathbf{g}_e^k \tag{7.46}$$

where $\Delta \mathbf{a}_j^{i+1}$ and $\Delta \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}$ are the increments, with their corresponding coefficients the terms of the jacobian. The increments satisfy the following equations:

$$\Delta \mathbf{a}_j^{i+1} = \mathbf{a}_{j+1}^{i+1} - \mathbf{a}_j^{i+1} \tag{7.47}$$

$$\Delta \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} = \left(\boldsymbol{\sigma}_e^k\right)_{j+1}^{i+1-\alpha_f} - \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} \tag{7.48}$$

The system of equations indicated in (7.45) and (7.46) can be solved at each iteration for $\Delta \mathbf{a}_j^{i+1}$ and $\Delta \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f}$. However, this system of equations is larger than the system given in (7.26) since the element stresses are additional variables. Therefore, the use of a mixed formulation has increased the computational cost.

Using (7.42), (7.23), (7.24), (7.34) and (7.43) we find the terms of the jacobian of the system as:

$$\frac{\partial \hat{\mathbf{r}}}{\partial \mathbf{a}^{i+1}} = (1-\alpha_m)\,\mathbf{M} + \left(1-\alpha_f\right) \beta\, (\Delta t)^2 \sum_e \sum_k \mathbf{V}_e \frac{\partial \mathbf{B}_e^{kT}}{\partial \mathbf{u}_e} \boldsymbol{\sigma}_e^k w^k \mathbf{V}_e^T \tag{7.49}$$

$$\frac{\partial \hat{\mathbf{r}}}{\partial \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}} = \mathbf{V}_e \mathbf{B}_e^{kT} w^k \tag{7.50}$$

$$\frac{\partial \mathbf{g}_e^k}{\partial \mathbf{a}^{i+1}} = -\left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \mathbf{C}_e^k \mathbf{B}_e^k \mathbf{V}_e^T \tag{7.51}$$

$$\frac{\partial \mathbf{g}_e^k}{\partial \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}} = \mathbf{I} \tag{7.52}$$

where these terms, being parts of the jacobian of the system, are evaluated at intermediate times and for the current iteration, as indicated by (7.27). The evaluation is not indicated explicitly to simplify the notation.

Condensation of variables

The system of equations for the mixed formulation defined by (7.45) and (7.46) can be simplified by condensing out one equation. Substituting (7.51), (7.52) and (7.42) in (7.46):

$$\Delta\left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} = \left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \mathbf{C}_e^k \mathbf{B}_e^k \mathbf{V}_e^T \Delta \mathbf{a}_j^{i+1} - \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f} + \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k\left[\left(\mathbf{u}_e\right)^{i+1-\alpha_f}\right] \tag{7.53}$$

Substituting (7.53), (7.49) and (7.50) into (7.45):

$$\begin{pmatrix} \left(1-\alpha_m\right)\mathbf{M} + \left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \sum_e \sum_k \mathbf{V}_e \dfrac{\partial \mathbf{B}_e^{kT}}{\partial \mathbf{u}_e} \boldsymbol{\sigma}_e^k w^k \mathbf{V}_e^T \\ + \sum_e \sum_k \left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \mathbf{V}_e \mathbf{B}_e^{kT} w^k \mathbf{C}_e^k \mathbf{B}_e^k \mathbf{V}_e^T \end{pmatrix} \Delta \mathbf{a}_j^{i+1} = \\ \sum_e \sum_k \mathbf{V}_e \mathbf{B}_e^{kT} w^k \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f} - \sum_e \sum_k \mathbf{V}_e \mathbf{B}_e^{kT} w^k \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k\left[\left(\mathbf{u}_e\right)^{i+1-\alpha_f}\right] - \hat{\mathbf{r}} \tag{7.54}$$

We identify the stiffness matrix with stresses computed independently as in (7.44). Substituting also (7.42):

$$\left(\left(1-\alpha_m\right)\mathbf{M} + \left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \hat{\mathbf{K}}\left[\mathbf{u}, \boldsymbol{\sigma}_e^k\right]\right)\Delta \mathbf{a}_j^{i+1} = \sum_e \sum_k \mathbf{V}_e \mathbf{B}_e^{kT} w^k \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f} \\ - \sum_e \sum_k \mathbf{V}_e \mathbf{B}_e^{kT} w^k \mathbf{C}_e^k \boldsymbol{\epsilon}_e^k\left[\left(\mathbf{u}_e\right)^{i+1-\alpha_f}\right] - \mathbf{M}\mathbf{a}^{i+1-\alpha_m} - \hat{\mathbf{f}}\left(\mathbf{u}^{i+1-\alpha_f}, \left(\boldsymbol{\sigma}_e^k\right)^{i+1-\alpha_f}\right) + \mathbf{F}^{i+1-\alpha_f} \tag{7.55}$$

Substituting (7.41) and (7.38):

$$\left(\left(1-\alpha_m\right)\mathbf{M} + \left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \hat{\mathbf{K}}\left[\mathbf{u}, \boldsymbol{\sigma}_e^k\right]\right)\Delta \mathbf{a}_j^{i+1} = -\mathbf{f}\left(\mathbf{u}^{i+1-\alpha_f}\right) - \mathbf{M}\mathbf{a}^{i+1-\alpha_m} + \mathbf{F}^{i+1-\alpha_f} \tag{7.56}$$

We identify on the right side the residual $\mathbf{r}$ of the dynamic problem considering stresses as a function of displacements, as in (7.22):

$$\left(\left(1-\alpha_m\right)\mathbf{M} + \left(1-\alpha_f\right)\beta\left(\Delta t\right)^2 \hat{\mathbf{K}}\left[\mathbf{u}, \boldsymbol{\sigma}_e^k\right]\right)\Delta \mathbf{a}_j^{i+1} = -\mathbf{r} \tag{7.57}$$

The terms must be evaluated at the intermediate times as indicated by (7.27) and (7.29) for the current iteration j. Indicating the evaluation explicitly:

$$\Delta \mathbf{a}_j^{i+1} = -\left((1-\alpha_m)\,\mathbf{M} + \left(1-\alpha_f\right)\beta\,(\Delta t)^2\,\hat{\mathbf{K}}\left[\mathbf{u}_j^{i+1-\alpha_f}, \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f}\right]\right)^{-1} \mathbf{r}\left[\mathbf{u}_j^{i+1-\alpha_f}, \mathbf{a}_j^{i+1-\alpha_m}\right] \tag{7.58}$$

This expression is used to compute the increment $\Delta \mathbf{a}_j^{i+1}$. It can be observed that this equation is very similar to the equation for the increment $\Delta \mathbf{a}_j^{i+1}$ in a standard formulation (not mixed), given by(7.33). The only difference is that the stiffness matrix is computed with element stresses being independent of displacement variables. Therefore, the condensation procedure has allowed using a mixed formulation with the same computational cost as standard formulations.

To compute $\Delta\left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f}$ we use (7.53), where the evaluation of the variables for the corresponding iteration is expressed explicitly as:

$$\Delta\left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} = \left(1-\alpha_f\right)\beta\,(\Delta t)^2\,\mathbf{C}_e^k\mathbf{B}_e^k\mathbf{V}_e^T\Delta\mathbf{a}_j^{i+1} - \left(\boldsymbol{\sigma}_e^k\right)_j^{i+1-\alpha_f} + \mathbf{C}_e^k\boldsymbol{\epsilon}_e^k\left[\mathbf{u}_j^{i+1-\alpha_f}\right] \tag{7.59}$$

Numerical results

In this section, we will study the efficacy of the MIP method in speeding up the solution of the tape spring deployment problem. An implicit dynamics solver with the generalized alpha method is used. This is necessary because the MIP is only applicable with an implicit solver. We use the same tape spring properties as in chapter 5.

The results without using the MIP method are shown in Figure 7.6, where $\rho_\infty = 0.85$ and $\Delta t = 10^{-5}s$. We now consider the same problem but using the MIP method. The energy history is shown in Figure 7.8. In this case, the simulation proceeds until the end ($t = 0.3$ s) despite using a relatively large time step $\Delta t = 10^{-5}$ s. We observe the results at a given time are the same as those obtained without using the MIP method, which is expected since the MIP method does not directly change the solution at every time step but only improves convergence. To analyze the convergence failure we plot the number of iterations required at every time increment with and without applying the MIP method, as shown in Figure 7.9. We observe that initially both approaches require the same number of iterations. However, after $t = 0.1s$ the number of iterations required when using the MIP method stays constant. On the other hand, the results when not using the MIP method require increasingly more iterations around $t = 0.1s$ and eventually too many iterations are required, producing convergence failure.

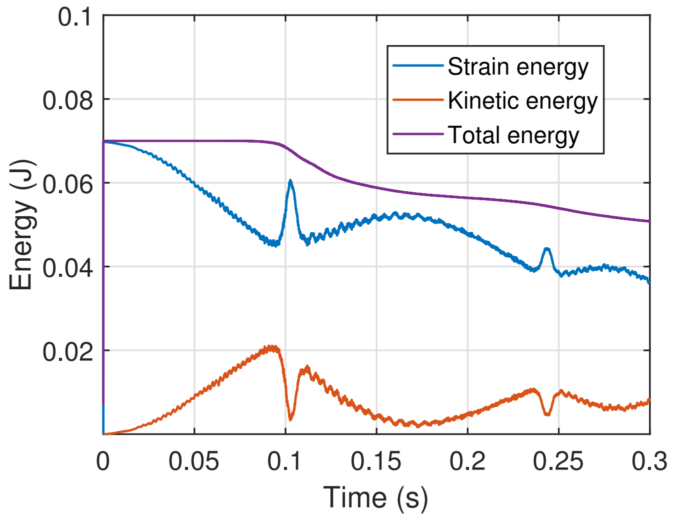


Figure 7.8: Energy history using MIP method with $\rho_\infty = 0.85$ and $\Delta t = 10^{-5}$ s.

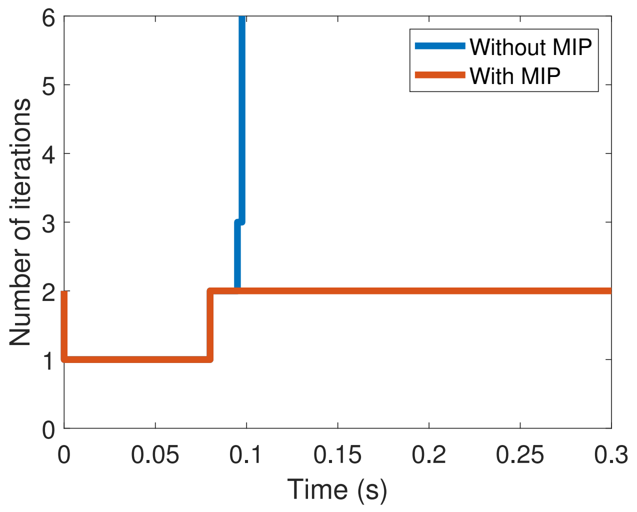


Figure 7.9: Number of iterations required for convergence with and without MIP.

To further study the improved convergence, we use the condition given in equation (7.19). The spectral radius at different time steps when MIP is not used is shown in Figure 7.10. It can be observed that the spectral radius is initially small but gradually increases, approaching values close to 1 right before $t = 0.1s$, and afterwards the simulation fails to converge. Therefore, interpolation locking is present in the simulation. The failure to convergence is correctly identified by the spectral radius condition. We now use the MIP method and recompute the spectral radius for this case, as shown in Figure 7.11. We observe that the spectral radius remains low and close to 0, which explains the low number of iterations for convergence. Therefore, the MIP method was successful in alleviating the interpolation locking and allowing convergence of the simulation.

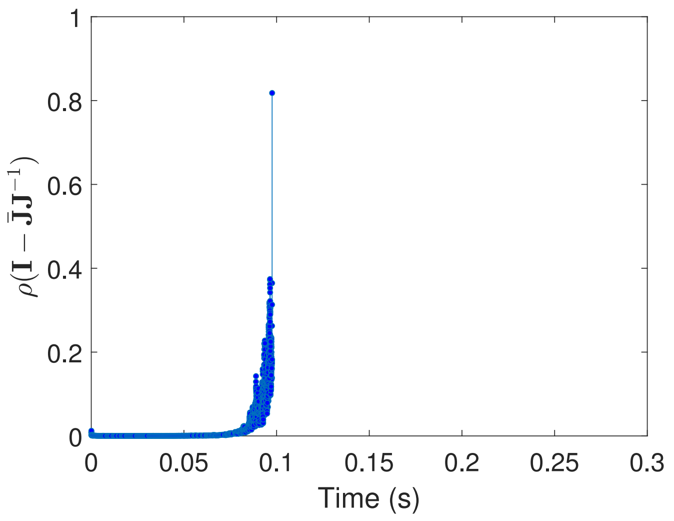


Figure 7.10: Spectral radius of matrix $(\mathbf{I} - \bar{\mathbf{J}}\mathbf{J}^{-1})$ when no MIP procedure is used.

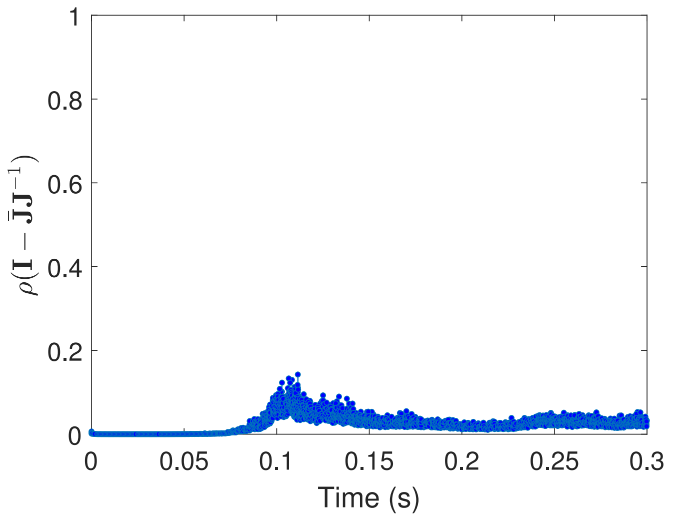


Figure 7.11: Spectral radius of of matrix $(\mathbf{I} - \bar{\mathbf{J}}\mathbf{J}^{-1})$ with MIP procedure.

The CPU times for the solution with and without MIP using ρ_∞ are shown in table 7.2. It can be observed that using MIP allows convergence of the solution until the end (denoted as a complete solution) while using a relatively large time step. Reducing the time step also allows a complete solution, but with significantly more computational cost. Additionally, the time taken for the simulation using MIP is very similar to the one not using MIP if ρ_∞ and Δt are kept fixed.

7.4 Discussion

An adaptive meshing procedure for dynamic analysis of tape spring deployment was developed. A fine mesh is defined around the fold position and a coarse mesh is used elsewhere. When this adaptive mesh is chosen to have 49% of the uniform fine

Table 7.2: CPU time for implicit solver with and without MIP.

ρ_∞	Δt (s)	MIP	Status	CPU time (s)
0.85	10^{-5}	No	Incomplete	5 207
0.85	10^{-6}	No	Complete	34 546
0.85	10^{-5}	Yes	Complete	5 367

mesh degrees of freedom, a decaying total energy is obtained. It is suspected that a coarse mesh in the straight parts of the tape spring is insufficient due to dynamic buckling effects, and this produces inaccurate results.

The generalized alpha implicit solver was used to solve the tape spring deployment problem. Numerical damping is introduced to dampen artificial high-frequency dynamic components. It was observed that the numerical damping significantly reduces the total energy of the system, which implies that the tape spring deployment intrinsically has high-frequency dynamic motion with significant energy.

The interpolation locking effect was presented and extended for dynamic problems. A mixed formulation framework was developed for dynamic problems as a method to potentially alleviate the interpolation locking. The computational efficiency was retained by using a condensation of variables known as Mixed Interpolation Point (MIP). Numerical results show that the interpolation locking effect is present in the tape spring deployment problem, and that the MIP procedure is effective at alleviating the locking effect by reducing the number of iterations to convergence.

Chapter 8

CONCLUSIONS AND FUTURE WORK

Deployable structures are used in various space applications. Experimental testing has a high cost and there are difficulties in reproducing space conditions. Therefore, numerical models are used for prediction of the deployment process.

This thesis had three research objectives. Firstly, to reduce the computational cost of the tape spring opposite-sense bending. Secondly, to perform an accurate numerical analysis of the tape spring deployment, allowing various physical effects to be investigated. Thirdly, to improve the computational efficiency of the analysis of tape spring deployment using various techniques.

8.1 Efficient analysis of tape spring opposite-sense bending

The tape spring opposite-sense bending problem was studied using an in-house finite element code, validated using ABAQUS finite element software. Afterwards, a reduced order model based on energy-conserving sampling and weighting of data was presented. An on-the-fly algorithm is introduced for application of the reduced model. This approach was used to model the tape spring opposite-sense bending problem. The speedup achieved is about $1.25x$ compared to the original high-fidelity model. It was shown that the reduced order model is incapable of dealing with translation and rotation invariances, which greatly decreases the effectiveness of this method.

A novel adaptive meshing procedure was then introduced. This procedure uses the curvature changes along each curvilinear coordinate of each finite element to dictate the refinement. A stopping condition based on the boundary layer effect on tape springs is presented. The adaptive meshing approach is used for the analysis of the tape spring opposite-sense bending. The method results in highly efficient computations, and the dependence on the tape spring geometry was explored. The computational speedup is in the order of $2x-4x$ depending on the specific geometry.

8.2 Modeling of tape spring deployment

The tape spring deployment problem was studied using ABAQUS finite element software. It was shown that the element formulation used is inadequate for analysis of tape spring deployment due to the hourglassing present, which results in an excessive amount of artificial energy. Although this can be alleviated by refining the mesh, the computational cost is increased excessively.

The in-house isogeometric shell code was shown to be capable of avoiding hourglassing due to the special properties of the shape functions. Using a special reduced integration rule, both hourglassing and shear locking are almost entirely avoided. This makes this formulation optimal for the analysis of tape spring deployment. The Newmark explicit solver was used to discretise the time. Numerical results are then presented for the energy history and the fold kinematics during deployment.

Afterwards, the energy leak present in tape spring deployment experiments was investigated. This effect is of relevance since it motivates the use of damping for tape spring deployment in the literature. Unlike experiments, the in-house shell code can have truly zero damping. This allows it to examine the effect of various simplifications used in analytical models from previous work while not being distorted by damping on experiments. After a detailed analysis of energy components within the tape spring, it is observed that the analytical models previously used are partially responsible for the energy leak. This analysis also demonstrates that the straight parts of the tape spring have significant strain energy after the fold reflection, and thus it may not be necessarily easier to model than the fold region. The kinetic energy of in-plane and out-of-plane oscillations is also shown to be significant.

Additionally, the transfer of energy from low to high-frequency dynamic components was analyzed using the short-time Fourier transform. It was observed that initially low-frequency components dominate the analysis, but over time there is a transfer of energy to high-frequency components. Therefore, the dynamics become more complex as the time advances.

8.3 Efficient analysis of tape spring deployment

The tape spring deployment problem was solved using an adaptive mesh. The mesh is defined based on the fold position, where a finer discretisation is used in the fold region and a coarse discretisation outside of this fold region. Since the transfer of variables between meshes is not straightforward in dynamic problems with special shape functions, a special transfer method is used. Numerical results are shown using an adaptive mesh with 49% of the degrees of freedom of the high-fidelity model. Accurate results are obtained up to the fold reflection. However, after the fold reverses direction the total energy slowly decreases, indicating that the mesh used is not sufficient for accurate analysis. Although an adaptive mesh with more degrees of freedom could be used, the computational efficiency becomes less attractive.

Afterwards, the solution using an implicit solver was explored. The generalized alpha implicit solver was used due to the capability of controlling damping in the high and low frequency range. Numerical results showed a decrease in the total energy, unlike the solution with an explicit solver. The results showed that a significant part of the energy in the deployment is contained in high-frequency components, which is difficult to solve correctly using implicit solvers. Although smaller time steps and moving the high-frequency threshold to even higher frequencies reduces the magnitude of the decrease in total energy, this significantly increases the computational cost.

The efficiency of using implicit solvers for tape spring deployment analysis was shown to be related to a novel locking effect known as interpolation locking. This locking effect does not directly reduce the accuracy but significantly increases the number of iterations for convergence. Therefore, this locking ultimately forces the use of larger damping to improve the convergence behavior, resulting in even more total energy decrease. Since mixed formulations alleviate this locking effect, a mixed formulation framework is presented and extended for dynamic problems. Afterwards, a condensation procedure known as Mixed Interpolation Point (MIP) is applied to maintain the same number of variables as the original formulation while taking advantage of the mixed formulation properties. Numerical results showed that after the fold reflection the MIP method significantly improves the computational cost. This was shown to be related to the presence of interpolation locking in the tape spring deployment when the MIP method is not used.

8.4 Future Work

To further develop the work in this thesis, the following steps can be taken:

- Analyze the dynamic deployment of more complicated thin-shell deployable structures
- Apply adaptive meshing and the MIP method for more complex structures
- Explore other reduced order modeling techniques to further reduce the computational cost

- Develop experiments to validate the computational results
- Study the damping present in experiments and implement an appropriate damping formulation in the simulation.

BIBLIOGRAPHY

[1] *ABAQUS User Manual*. Mar. 2021. URL: http://130.149.89.49:2080/v6.14/books/usb/default.htm.

[2] C. Adam, S. Bouabdallah, M. Zarroug, and H. Maitournam. "Improved numerical integration for locking treatment in isogeometric structural elements. Part II: Plates and shells". In: *Computer Methods in Applied Mechanics and Engineering* 284 (2015), pp. 106–137.

[3] K. J. Bathe. *Finite element procedures*. Prentice Hall, 2006.

[4] T. Belytschko, J. Lin, and T. Chen-Shyh. "Explicit algorithms for the nonlinear dynamics of shells". In: *Computer Methods in Applied Mechanics and Engineering* 42.2 (1984), pp. 225–251.

[5] T. Belytschko, J. Ong, W. Liu, and J. Kennedy. "Hourglass control in linear and nonlinear problems". In: *Computer Methods in Applied Mechanics and Engineering* 43.3 (1984), pp. 251–276.

[6] D. Benson, Y. Bazilevs, M-C Hsu, and T. Hughes. "A large deformation, rotation-free, isogeometric shell". In: *Computer Methods in Applied Mechanics and Engineering* 200.13-16 (2011), pp. 1367–1378.

[7] S. Bourgeois, B. Cochelin, F. Guinot, and E. Picault. "Buckling analysis of tape springs using a rod model with flexible cross-sections". In: *European Journal of Computational Mechanics* 21.3-6 (2012), pp. 184–194.

[8] S. Brunton and J. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2022.

[9] T. Bui-Thanh, K. Willcox, and O. Ghattas. "Model reduction for large-scale systems with high-dimensional parametric input space". In: *SIAM Journal on Scientific Computing* 30.6 (2008), pp. 3270–3288.

[10] T. Bui-Thanh, K. Willcox, and O. Ghattas. "Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications". In: *AIAA Journal* 46.10 (2008), pp. 2520–2529.

[11] *Calculix eight-node shell element*. Apr. 2023. URL: https://web.mit.edu/calculix_v2.7/CalculiX/ccx_2.7/doc/ccx/node39.html.

[12] C. Calladine and K. Seffen. "Folding the Carpenter's tape: boundary layer effects". In: *Journal of Applied Mechanics* 87.1 (2020).

[13] F. Canales and S. Pellegrino. "High-Fidelity Simulations of Thin-Shell Deployable Structures with Adaptive Meshing". In: *AIAA SCITECH 2022 Forum*. 2022, p. 1269. DOI: 10.2514/6.2022-1269.

[14] J. Chung and G. Hulbert. "A time integration algorithm for structural dynamics with improved numerical dissipation: the generalized-α method". In: *Journal of Applied Mechanics* 60.2 (1993), pp. 371–375.

[15] M. Crisfield. "A fast incremental/iterative solution procedure that handles "snap-through"". In: *Computational Methods in Nonlinear Structural and Solid Mechanics*. Elsevier, 1981, pp. 55–62.

[16] F. Cuvelier, C. Japhet, and G. Scarella. "An efficient way to assemble finite element matrices in vector languages". In: *BIT Numerical Mathematics* 56 (2016), pp. 833–864.

[17] R. De Borst, M. Crisfield, J. Remmers, and C. Verhoosel. *Nonlinear finite element analysis of solids and structures*. John Wiley & Sons, 2012.

[18] F. Dewalque, P. Rochus, and O. Brüls. "Importance of structural damping in the dynamic analysis of compliant deployable structures". In: *Acta Astronautica* 111 (2015), pp. 323–333.

[19] R. Echter, B. Oesterle, and M. Bischoff. "A hierarchic family of isogeometric shell finite elements". In: *Computer Methods in Applied Mechanics and Engineering* 254 (2013), pp. 170–180.

[20] S. Erlicher, L. Bonaventura, and O. Bursi. "The analysis of the generalized-α method for non-linear dynamic problems". In: *Computational Mechanics* 28.2 (2002), pp. 83–104.

[21] C. Farhat, P. Avery, T. Chapman, and J. Cortial. "Dimensional reduction of nonlinear finite element dynamic models with finite rotations and energy-based mesh sampling and weighting for computational efficiency". In: *International Journal for Numerical Methods in Engineering* 98.9 (2014), pp. 625–662.

[22] D. Flanagan and T. Belytschko. "A uniform strain hexahedron and quadrilateral with orthogonal hourglass control". In: *International Journal for Numerical Methods in engineering* 17.5 (1981), pp. 679–706.

[23] G. Garcea, G. Trunfio, and R. Casciaro. "Mixed formulation and locking in path-following nonlinear analysis". In: *Computer Methods in Applied Mechanics and Engineering* 165.1-4 (1998), pp. 247–272.

[24] H. Hilber, T. Hughes, and R. Taylor. "Improved numerical dissipation for time integration algorithms in structural dynamics". In: *Earthquake Engineering & Structural Dynamics* 5.3 (1977), pp. 283–292.

[25] S. Hoffait, O. Brüls, D. Granville, F. Cugnon, and G. Kerschen. "Dynamic analysis of the self-locking phenomenon in tape-spring hinges". In: *Acta Astronautica* 66.7-8 (2010), pp. 1125–1132.

[26] J. Hokkanen and D. Pedroso. "Quadrature rules for isogeometric shell formulations: study using a real-world application about metal forming". In: *Computer Methods in Applied Mechanics and Engineering* 363 (2020), p. 112904.

[27] T. Hughes, J. Cottrell, and Y. Bazilevs. “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement”. In: *Computer Methods in Applied Mechanics and Engineering* 194.39-41 (2005), pp. 4135–4195.

[28] S. Jeon and T. Murphey. “Design and analysis of a meter-class CubeSat boom with a motor-less deployment by bi-stable tape springs”. In: *52nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 19th AIAA/ASME/AHS Adaptive Structures Conference 13t*. 2011, p. 1731.

[29] J. Kiendl. “Isogeometric analysis and shape optimal design of shell structures”. PhD thesis. Technische Universität München, 2011.

[30] J. Kiendl, K. Bletzinger, J. Linhard, and R. Wüchner. “Isogeometric shell analysis with Kirchhoff–Love elements”. In: *Computer Methods in Applied Mechanics and Engineering* 198.49-52 (2009), pp. 3902–3914.

[31] S. Krenk. “Energy conservation in Newmark based time integration algorithms”. In: *Computer Methods in Applied Mechanics and Engineering* 195.44-47 (2006), pp. 6110–6124.

[32] C. Lawson and R. Hanson. *Solving least squares problems*. SIAM, 1995.

[33] D. Magisano, L. Leonetti, and G. Garcea. “Advantages of the mixed format in geometrically nonlinear analysis of beams and shells using solid finite elements”. In: *International Journal for Numerical Methods in Engineering* 109.9 (2017), pp. 1237–1262.

[34] D. Magisano, L. Leonetti, and G. Garcea. “How to improve efficiency and robustness of the Newton method in geometrically non-linear structural problem discretized via displacement-based finite elements”. In: *Computer Methods in Applied Mechanics and Engineering* 313 (2017), pp. 986–1005.

[35] H. Mallikarachchi. “Thin-walled composite deployable booms with tape-spring hinges”. PhD thesis. University of Cambridge, 2011.

[36] H. Mallikarachchi and S. Pellegrino. “Deployment dynamics of ultrathin composite booms with tape-spring hinges”. In: *Journal of Spacecraft and Rockets* 51.2 (2014), pp. 604–613.

[37] J. Mosler and M. Ortiz. “Variational h-adaption in finite deformation elasticity and plasticity”. In: *International Journal for Numerical Methods in Engineering* 72.5 (2007), pp. 505–523.

[38] T. Murphey and J. Banik. *Triangular rollable and collapsible boom*. US Patent 7,895,795. Mar. 2011.

[39] T. Murphey, S. Jeon, A. Biskner, and G. Sanford. “Deployable booms and antennas using bi-stable tape-springs”. In: *Small Satellite Conference*. 2011.

[40] N. Newmark. “A method of computation for structural dynamics”. In: *Journal of the Engineering Mechanics Division* 85.3 (1959), pp. 67–94.

[41] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[42] B. Oesterle, R. Sachse, E. Ramm, and M. Bischoff. “Hierarchic isogeometric large rotation shell elements including linearized transverse shear parametrization”. In: *Computer Methods in Applied Mechanics and Engineering* 321 (2017), pp. 383–405.

[43] T. Phillips, C. Heaney, P. Smith, and C. Pain. “An autoencoder-based reduced-order model for eigenvalue problems with application to neutron diffusion”. In: *International Journal for Numerical Methods in Engineering* 122.15 (2021), pp. 3780–3811.

[44] E. Picault, S. Bourgeois, B. Cochelin, and F. Guinot. “A rod model with thin-walled flexible cross-section: Extension to 3D motions and application to 3D foldings of tape springs”. In: *International Journal of Solids and Structures* 84 (2016), pp. 64–81.

[45] E. Picault, P. Marone-Hitz, S. Bourgeois, B. Cochelin, and F. Guinot. “A planar rod model with flexible cross-section for the folding and the dynamic deployment of tape springs: Improvements and comparisons with experiments”. In: *International Journal of Solids and Structures* 51.18 (2014), pp. 3226–3238.

[46] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 1996.

[47] R. Radovitzky and M. Ortiz. “Error estimation and adaptive meshing in strongly nonlinear dynamic problems”. In: *Computer Methods in Applied Mechanics and Engineering* 172.1-4 (1999), pp. 203–240.

[48] J. Reddy. *Theory and analysis of elastic plates and shells*. CRC press, 2006.

[49] N. Reddy and S. Pellegrino. “Time-efficient geometrically non-linear finite element simulations of thin shell deployable structures”. In: *AIAA Scitech 2021 Forum*. 2021, p. 1795.

[50] C. Reinsch. “Smoothing by spline functions”. In: *Numerische mathematik* 10.3 (1967), pp. 177–183.

[51] M. Rezaiee-Pajand and M. Karimi-Rad. “A family of second-order fully explicit time integration schemes”. In: *Computational and Applied Mathematics* 37 (2018), pp. 3431–3454.

[52] E. Riks. “An incremental approach to the solution of snapping and buckling problems”. In: *International Journal of Solids and Structures* 15.7 (1979), pp. 529–551.

[53] F. Rimrott and G. Fritzsche. “Fundamentals of stem mechanics”. In: *IUTAM-IASS Symposium on Deployable Structures: Theory and Applications: Proceedings of the IUTAM Symposium held in Cambridge, UK, 6–9 September 1998*. Springer. 2000, pp. 321–333.

[54] D. Schillinger, S. Hossain, and T. Hughes. “Reduced Bézier element quadrature rules for quadratic and cubic splines in isogeometric analysis”. In: *Computer Methods in Applied Mechanics and Engineering* 277 (2014), pp. 1–45.

[55] K. Seffen and S. Pellegrino. “Deployment dynamics of tape springs”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 455.1983 (1999), pp. 1003–1048.

[56] K.A. Seffen. “On the behavior of folded tape-springs”. In: *J. Appl. Mech.* 68.3 (2001), pp. 369–375.

[57] L. Sirovich. “Turbulence and the dynamics of coherent structures. I. Coherent structures”. In: *Quarterly of Applied Mathematics* 45.3 (1987), pp. 561–571.

[58] Z. Soltani and M. Santer. “Efficient geometrically-nonlinear analysis of tape spring flexures using a unified beam formulation”. In: *AIAA Scitech 2021 Forum*. 2021, p. 1149.

[59] H. Stolarski and T. Belytschko. “Shear and membrane locking in curved C0 elements”. In: *Computer Methods in Applied Mechanics and Engineering* 41.3 (1983), pp. 279–296.

[60] L. Wilson. “Analysis of Packaging and Deployment of Ultralight Space Structures”. PhD thesis. California Institute of Technology, 2017.

[61] O. Zienkiewicz, R. Taylor, and J. Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 2005.