APPLICATIONS OF PARAMETERIZED L-SYSTEMS FOR PRELIMINARY STRUCTURAL DESIGN AND OPTIMIZATION

A Thesis

by

MADALYN K. MIKKELSEN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Darren J. Hartl
Committee Members,	Dimitris Lagoudas
	Richard Malak
Head of Department,	Rodney Bowersox

August 2020

Major Subject: Aerospace Engineering

ABSTRACT

Preliminary design is a complicated problem that is often solved using topology optimization. In this work, a heuristic approach to topology optimization is considered. This approach involves the coupling of a genetic optimization with a parallel rewriting system, known as an L-System. This approach encodes design variables into a string of characters that are then coupled with an interpreter to develop a structure in a given domain. By considering a heuristic bio-inspired approach over more traditional density and level set topology approaches, we are able to avoid numerical issues and rapid increasing design space dimensionality associated with complicated multi objective problems. In this work a new interpreter for L-Systems, called the Spatial Interpretation for the Design of Reconfigurable Structures (SPIDRS), is applied to several design problems. This work seeks to show SPIDRS as a powerful preliminary design tool for difficult problems that lack traditional engineering intuition.

First, a morphing airfoil inspired by the rotor blade of the UH-60 is considered. SPIDRS is utilized to determine the internal structural layout as well as the placement of actuators to facilitate morphing to meet a shape objective while minimizing mass. A coupled fluid structure interaction (FSI) evaluation is performed using the finite element method (Abaqus) and vortex lattice method (XFOIL). The resulting final shape of the airfoil, as determined by the FSI evaluation, and the mass are used as objectives in a genetic optimization

Second, a set of origami fold design problems are considered. SPIDRS is first validated using the well established square twist pattern and the results are compared to previous work in the literature. SPIDRS is then used with an arbitrary continuous kinematic objective to determine its ability to evolve towards an unknown solution pattern. The standard square twist pattern is also utilized to evaluate the efficacy of altering the original SPIDRS production rules for use specifically in origami.

DEDICATION

To my parents and grandparents. Thank you for your unending support and dedication to my education. You worked your entire lives to make this possible and I will never be able to show how grateful I am for you.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Professor Darren Hartl and Dimitris Lagoudas of the Department of Aerospace Engineering and Professor Richard Malak of the Department of Mechanical Engineering.

The work for Chapter 3 was completed with the assistance of Patrick Walgren, Michayal Mathew, and Brent Bielefeldt of Texas A&M University. The sample TOSCA results were graciously provided by Basak Abut. The results for Chapter 4 were collected in collaboration with Dr. Andrew Gillman and Dr. Phillip Buskohl of the Air Force Research Laboratory Materials and Manufacturing Directorate, and Kazuko Fuchi of the University of Dayton Research Institute.

All other work conducted for the thesis (or) dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a the TAMU/AFRL Data-Enabled Discovery and Design of Materials Program.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Motivation

As engineering problems increase in complexity with the introduction of smart materials and multiphysical objective problems, the traditional intuition of engineers becomes less applicable to design. For example, the structural layout of a static aircraft wing is a well studied problem and there are developed methods for designing these wings through a combination of prior knowledge and computational analysis methods. However, when the idea of optimizing the wing shape to perform well for an electromagnetic objective (i.e. be stealthy under radar detection), as well as the original aerodynamic and structural objectives, the intuition of design engineers is less established. This example demonstrates the need for preliminary design processes that are capable of working with limited human intuition, as problems become inherently multidisciplinary.

Biological systems are considered the best examples of solutions to these complex multidisciplinary problems, as they have been observed to optimize and adapt to the varying challenges continuously faced by organisms on Earth. Some examples of complex multidisciplinary topological layouts found in nature include leaf venation and the root systems of plants [5], venation of insect wings [6], and the circulatory systems of animals [7]. Engineers have long taken inspiration from nature to design their own versions of these systems and solutions. One of the earliest and most recognizable examples of this being Leonardo da Vinci's flying machines based on biomimicry of avian flight in the 15th century. More recently in the field of topology optimization, bio-inspired branching structures have been utilized to generate structural topologies for complex engineering problems [8, 9, 10].

The bio-inspired branching methods depend on a sequence of encoded genes and an interpreter, to decode the genes and execute a sequences of functions to generate a structure. The genes are encoded using a biological system called the Lidenmayer System [11, 12]. Previously, a method called Turtle interpretation has been used to interpret genes and build topologies by growing struc-

tures from a given point within a boundary. This method was developed with the intent to generate various types of images computationally [4, 13, 14]. This method has proven to be effective at developing interesting engineering structures [15, 16, 17], but has also shown issues with the limited nature of the Turtle graphics with respect to the assignment segment angles and lengths during the interpretation of the genes [18]. Notably, Turtle graphics also result in many "hanging" (i.e. non-load bearing) branches, which are unnecessary for structural design and often require removal via post processing.

The short comings of this popular methods for interpretation of L-Systems has inspired a new interpretation scheme called Spatial Interpretation for the Design of Reconfigurable Structures (SPIDRS) [18]. In this work, we will examine two areas of topology optimization, morphing airfoils and origami fold patterns, that could benefit from implementing an optimization scheme including SPIDRS. These two areas were chosen due to the lack of prior work existing for these problems, as well as the complicated objectives associated with successful designs. The current state of these fields are discussed below.

1.2 Literature Review

1.2.1 Topology Optimization

Topology optimization is the design of material distribution in a given domain subject to constraints and boundary conditions in order to maximize specific performance conditions of the system. This method is generally utilized during the preliminary stage of design and then refined to generate final products, thus making topology optimization extremely influential on the performance of the final design [19]. Density based methods, including Solid Isotropic Material with Penalization (SIMP), are the most popular approach to topology optimization, and consider a domain of finite elements and work to minimize an objective by determining whether a given element consists of solid material or is void [20, 21]. This method is easy to implement, but has difficulty with high dimensionality associated with the high resolution of elements often required for accuracy [19]. Density based methods also suffer from checker boarding, where the formation of

adjacent solid-void elements are arranged in a checker board pattern resulting in non-physical solutions [19, 20], and mesh dependencies, where different topologies result from the same design domain depending on the level of discretization.

The level set method is another well known topology optimization technique. Level set represents structural boundaries as the zero-level curve of a scalar level set function [19, 22]. This method is popular due to its convenient treatment of topological changes, which allows the boundaries of the structure to be modified by using optimization conditions to control the output of the level set function. The level set function is generally parameterized using the finite element method (FEM) or radial basis functions (RBFs) [22]. The accuracy of the local structural response is dictated by how the level set function is mapped to the structural model, meaning more accurate results often have a significantly higher computational expense [23]. As such, the dimensionality of the design space increases rapidly with the number of parameters used to define the level set function. Aditionally, this method also suffers from poor convergence rates and convergence to local minima [24].

There are also less popular methods used to approach topology optimization, like evolutionary structural optimization (ESO) and the ground structure method (GSM). ESO is based on the idea that optimal structures are composed of members that experience equivalent stress, and material is gradually removed from the design domain based on heuristic criteria like stress in the elements [25]. GSM represents the initial domain of the structure as a series of structural members that connect every member in a set of nodes to every other member in that set. The topology is then determined by varying the cross-sectional area of each member in the structure [26], and removing members with zero cross sectional area from the structure. This process is very good at finding solutions to truss problems, but the ultimate quality of the result is dependent on the initial locations and number of nodes included in the structural domain. Increasing the number of nodes results in an increase in design variables therefore increasing computational time needed to evaluate the initially dense configurations [26]. These methods also require intuition about the connectivity of the nodes and initial members to consider, making them a poor fit for the targeted problems in this

work.

Gradient optimization methods have been used extensively in order to find Pareto-optimal solutions using the methods mentioned above [20, 24, 26]. However, gradient optimization methods tend to have difficulties when problems contain multiple local minima, discontinuous design spaces, or discrete variables [27]. All three of these criteria are likely to be encountered in novel multidisciplinary design, which necessitates an alternative to account for these challenges. Therefore a non-gradient optimization approach should be utilized. Non-gradients methods utilize function evaluations of objectives to converge to a solution rather than gradients. A common form of non-gradient optimizer is the genetic algorithm (GA). The GA functions by utilizing the principles of natural selection to construct an optimization process that requires minimal information about the problem [27]. Similar to the genes that control a living being's observerable traits, the GA uses pseudo-chromosomal representations of design parameters to create designs. Generally, a GA is initialized with a random population of designs, those designs are then evaluated to determine a fitness value and then they are modified through reproduction, crossover, and mutation. Multidisciplinary design is well suited to utilize the GA, due to the inherent multi objective nature of the problems. The GA's stochastic nature is well suited to prevent convergence to local minima and is able to handle discrete variables, unlike most gradient based optimizers.

Recently, a class of bio-inspired topological representation methodologies was developed as an alternative to the more rigorous methods presented previously. These methods use the Lindenmayer System (a biological model), which, when coupled with an interpreter, creates and executes a series of operations to develop the topology in stages. A GA is used to govern the evolution of the L-System encoding, which in turn governs the topology of the structure[28]. This method imitates natural selection by choosing designs with high performance scores and modifying the genes of each design via crossover or mutation to arrive at a group of Pareto optimal designs. This process couples a biologically inspired topology creation method with the GA to present a truly biologically inspired design approach. These L-System based topology design approaches include two classes of processes, those inspired by cell division and those inspired by branching structures.

Kobayashi et al., for example, is inspired by cell division and operates by iteratively subdividing the structural domain [15]. This method has been used to optimize a flapping wing mechanism [29, 30] and aircraft wing structures [31, 32, 33].

The branching class of L-System approaches rely on an interpretation scheme to replicate simple biological structures, like the veins in leaves, circulatory systems in animals, and branch growth in plants. Traditionally, these methods have utilized a vector-base graphical method for interpretation known as turtle graphics [34]. Turtle graphics constructs straight 2D line segments in a given design domain and is based on geometry, meaning the turtle operator tracks its geographical location in the design domain at all times. This method has been utilized to create structural topologies [16, 17, 18], but the formulation of turtle graphics limits the modeling capability of the L-System[18]. As a response to the limitations of the turtle graphics interpreter, the Spatial Interpretation for the Development of Reconfigurable Structures (SPIDRS) was developed by Bielefeldt et al [35] to utilize a graph based approach, enabling the interpretation of the L-System as a function of the nodes, edges, and faces of the structure instead of being based on geometry. SPIDRS has previously been utilized to design compliant mechanisms [18], morphing airfoils [36, 37], and tailorable stiffness structures [38].

1.2.2 Morphing Airfoils

The idea of enabling shape change on the aerodynamic surfaces of airplanes is as old as manned flight itself. The first manned aircraft, the Wright Flier, enabled roll control through the change of twist in the wing using cables controlled by the pilot [39]. The desire for increases in cruise speed and payloads caused aircraft to shift towards rigid aerodynamic surfaces, but the idea of morphing never completely left the aircraft industry. Current aircraft are designed as a compromise between several high level mission objectives, like performance at cruise, take-off, and landing, resulting in modern aircraft utilizing a static lifting surface shape coupled several methods to alter properties through the different flight conditions, such as flaps and slats.

This approach has been challenged by researchers who have proposed a solution that avoids the initial compromise between objectives and instead allows aircraft to fly optimally at all conditions

by morphing between flight conditions. Morphing of aircraft wings can generate changes in twist, span, geometry, and camber [39]. In this work the camber morphing of airfoils is considered specifically. Changes in camber work to adjust the outer mold line of the airfoil, which has been shown to improve the lift/drag ratio in changing flight conditions [40], and can serve to increase performance in the transonic regime [41].

Over the last three decades, several of camber morphing concepts have been presented and tested. Champanile discusses two types of structures that allow for camber morphing in airfoils, mechatronic solutions and structronic solutions [42]. Mechatronic solutions are presented as internal structures where mechanisms like hinges or linear bearings are used to give the internal structure the degrees of freedom required to morph. Some examples of mechatronic solutions include traditional high-lift devices and the multi-section variable camber wing presented by Poonsong [43]. These solutions do not rely on compliant mechanisms or regions in structure to achieve the desired shape change unlike structronic solutions, which are classified as solutions that use the distributed flexibility of the structure to achieve shape change.

Solutions that rely on distributed compliance of the structure or specific members of the structure have become very popular in academia. One of the first examples of these compliant designs is the 'fish-bone' morphing airfoil. This type of airfoil is generally built using a spine that is somewhat compliant with many ribs protruding from this central spine to support the skin, then the airfoil is morphed by changing the shape of the central spine. In the last decade this method has been very popular and used with many types of actuators including piezoelectric composites [44, 45] and linear actuators [45, 46, 47]. The belt-rib concept is another example, where a compliant structural member (the rib) is coupled with an actuator [42, 48]. Several designs also utilize the buckling of internal structural members to produce bi-stable camber morphing [49, 50], and there are even designs that rely solely on the compliance of the structure to deform under specific flow conditions [51, 52]. Both mechatronic and structronic solutions rely on actuators to achieve camber morphing. Researchers have used a variety of traditional (electric and pneumatic [53, 54, 55, 56]) and SMART materials (piezoelectrics and shape memory alloys [57, 3, 58]) actuators for this pur-

pose.

In many of these design problems, the structural layout is predetermined and the problem presented is an optimization of actuator placement or structural parameters such as spar thickness or skin thickness [57, 58, 3]. In others, simple triangulation meshing is utilized to represent topology generation [49]. L-System cellular division based design has specifically been applied to morphing aerodynamic surfaces by Kobayashi et al. [31] and Kolonay et al. [33, 32]. However in all of these applications, camber morphing is not considered. Branching L-Systems that utilize turtle graphics have not been implemented in camber morphing airfoil design, but the SPIDRS interpretation scheme has been used. Bielefeldt et al. presents the first attempt at camber morphing via L-System optimization, but is only applicable to supersonic diamond shaped airfoils [36]. The first attempt at camber morphing a traditional subsonic airfoil is presented by Mikkelsen et al. [37], but this work produced undesirable results due to skin buckling during morphing.

Overall, there has been no overarching system for the design of morphing airfoils. The actuation concepts vary widely and the implementations even more so. In this work, a system to approach the design optimization of a morphing airfoil is presented using the SPIDRS interpretation scheme due to its ability to generalize to many problems in the morphing airfoil domain. The overarching objective is a system by which preliminary design can be efficiently and reliably conducted for camber morphing airfoils without the requirement of extensive design intuition to establish the problem boundary or objectives.

1.2.3 Origami

Origami is the Japanese art of paper folding. Originally, the purpose of this art form was purely for recreation or art, but has since evolved into a well studied field of mathematics and engineering [59]. Engineers have utilized origami inspired design principles to create a variety of systems including deployable space structures [60, 61], biomedical applications [62], and metamaterials [63]. Origami facilitates novels methods of fabrication, assembly, storage, and even morphing [64].

The design of origami structures provides the opportunity to design materials and structures that

control shape, energy, and multi physical properties. However, efficient design tools are needed to design these structures computationally, due to the vast design space available to map 2D patterns to complex 3D structures [65]. The most common approach in origami design is to define a target 3D surface and then insert material and fold lines to 'unfold' the 3D surface back to a 2D sheet. Tachi developed an unfolding method by considering tucking-based folding approach where the target surface is discretized into panels and projected into 2D. The resulting panels are then assembled with infinitesimal thickness material between the panels that can be tucked away when folded [66]. Lang proposed a tree-based approach to reach a target shape, where a stick figure representation of the final shape is used to decompose the fold pattern into 2D. Demaine et al. developed methods for generating orthogonal folded structures used a simple universal hinge pattern [67], but this method only applies to structures with orthogonal elements. Pereza-Hernandez et al. eventually expanded on the tucked folding approach of Tachi by introducing smooth folds and accounting for material thickness [68]. The common feature of all of these methods is their deductive design nature, meaning they build the crease patterns on the geometric analysis of the desired shape. As a consequence, these methods are better suited for mapping optimal shapes to fold patterns than finding the optimal shapes to meet physical objectives.

The alternative to deductive origami design, is abductive design. In this approach, a function evaluation is utilized to evaluate and improve fold patterns iteratively. This enables design to satisfy geometric and functional requirements rather than just accurately defining targeted shapes. The ice cracking technique is one such method. Ice cracking operates in the 2D domain by placing nodes and then evolving a topology using an encoded array from a genetic algorithm [69]. Ice cracking guarantees design foldability by basing the encoding scheme off of several well known origami foldability theorems, and can be implemented to guarantee flat foldability as well, but the encoding does not allow for the creation of any new nodes thus limiting the topology to some subdomain containing the prescribed initial nodes.

A lot of abductive origami design work has also been completed by a the RXAS group at the Air Force Research Laboratory. This work originates with Fuchi and Diaz utilizing a ground structure

based approach to origami design coupled with rigid origami mechanics to achieve targeted macroscopic objectives [70]. The group then started to explore modified frame elements to enable higher fidelity origami modeling. Using this higher fidelity modeling, the ground structure approach was coupled with the new modified frame elements and fold patterns optimized by tuning the stiffness of the folds in the original ground structure [71], enabling true abductive origami design optimization. Sequenced folding was then utilized for the discovery of auxetic structures [**?**]. Finally, the tool was formalize by Gillman et al. [1] and utilized in the discovery of sequenced origami fold patterns [65]. This work has also been applied to multi physical electromagnetic problems as well by Sessions et al. [72], where a coupled origami and EM design approach is presented. However, in all of this work, the abductive method is limited by the necessity of a ground structure for topology creation. The initial ground structure approach with tunable stiffness parameters is effective at producing simple bench marking designs like the Chomper, but unable to match more difficult cases like the well known Square Twist pattern [65], which requires a coupled twisting and upward motion to achieve the folded configuration.

In this work the SPIDRS design approach will be applied to the abductive origami design problem in conjuction with the non-linear truss based solver developed by Gillman et al. [1]. This method should enable the origami design process to be facilitated quickly and without the need for a ground structure, therefore reducing the required prior knowledge of the solution.

1.3 Thesis Summary

The overarching goal of this work is to demonstrate the SPIDRS interpretation scheme as a tool to facilitate the design of preliminary topological layouts for difficult engineering problems. This methodology requires three distinct parts: i) topology generation, ii) functional evaluation, iii) genetic optimization. In this work, the SPIDRS interpretation scheme is implemented and adapted for various problems in order to generate structural topologies. Function evaluations are performed using various physical analysis tools and then utilized to inform a genetic optimization process towards the evolution of optimal topologies. This process is demonstrated graphically in Figure 1.1.

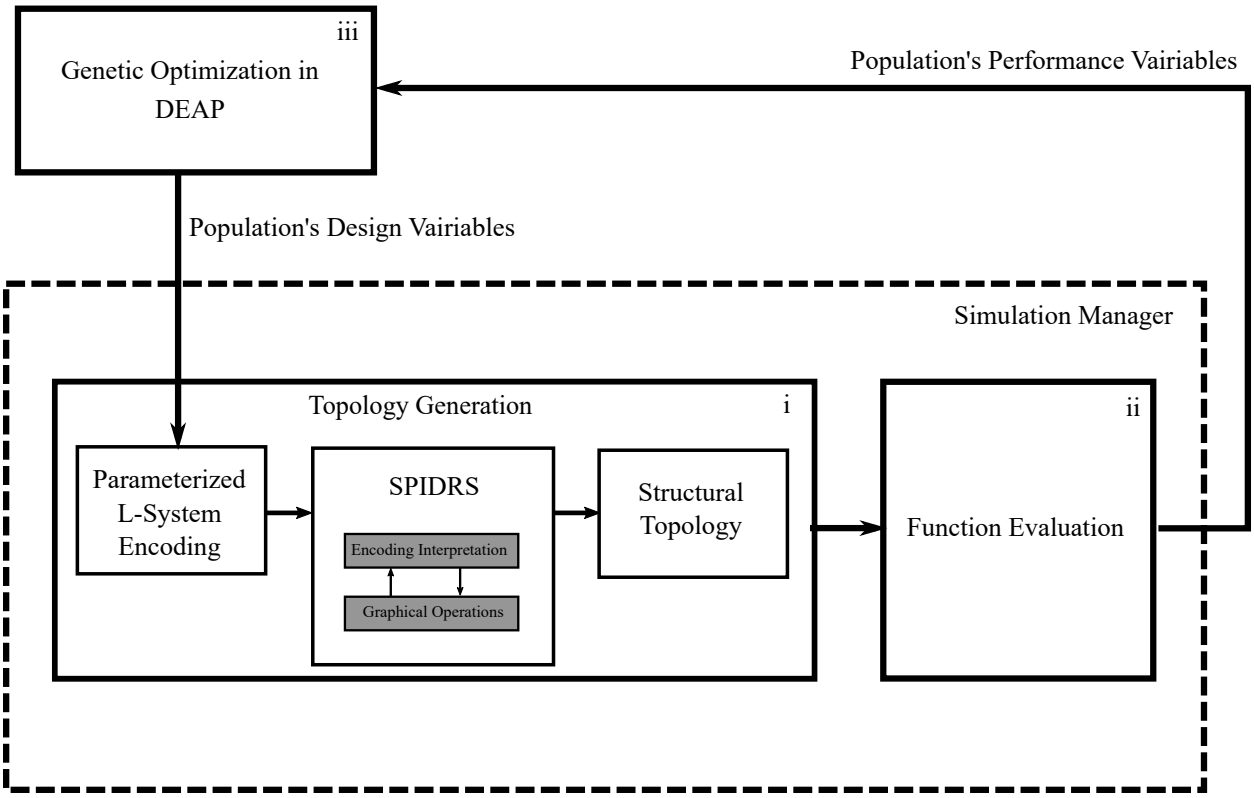


Figure 1.1: Topology design and optimization approach using SPIDRS.

This work utilizes tools built developed by previous researchers, but assembles then in a novel configuration to perform the function evaluations. In this thesis two novel design problems will be considered to evaluate the effectiveness of this approach. The presentation of this work is organized as follows:

- Chapter 2 provides an overview of the algorithms and computational tools used in this work to facilitate topology creation and functional evaluations. This chapter includes a descriptions of the SPIDRS algorithm, a non-linear truss based origami folding code, and overviews of the computational methods and software used to facilitate this work.

- Chapter 3 discusses the application of this design method to the structural topology optimization of a morphing airfoil. This section describes the problem setup, the implementation of the genetic optimizer, the structural and aerodynamic modeling, as well as the results of the optimization.

- Chapter 4 discusses the application of this design method to the topology optimization of origami fold patterns for kinematic objectives. First, the well known square twist pattern is considered and compared to previous work as a verification activity. Second, a continuous kinematic response objective without a known solution is presented. The SPIDRS design process is applied to this problem, an optimization is performed, and the results are reported.

- Chapter 5 summarizes the results of this work, presents the conclusions to be drawn from this work, and discusses possible future work.

2. OVERVIEW OF EXPERIMENTAL AND COMPUTATIONAL ENGINEERING TOOLS

In this work, the SPIDRS algorithm is utilized in conjunction with several different analysis tools to model physical reactions of interest. This process involves coupling specific tools with the SPIDRS code. In this chapter, the tools utilized to generate topologies are discussed in Section 2.1 and Section 2.2. Various tools utilized to model the physical responses of these topologies in a given problem are then described by Sections 2.3 through 2.6.

2.1 Parameterized Lindenmeyer System

A parallel rewriting system that uses a set of production rules to govern the evolution of a string of characters ($\boldsymbol{\omega}$) was developed by Aristid Lindenmayer in 1968, with the intent to provide a mathematical description of the development of simple organisms[11]. These Lindenmayer Systems (L-Systems) are composed of an alphabet, an axiom, and a set of production rules [12]. The alphabet consists of constant characters that are used to define actions in the topology drawing algorithm and/or assign material functionalities. The axiom is a string of N variable characters that provides the foundation for the full topology string of a structure. The production rules are a set of rules which are applied to the axiom recursively to create a new string of characters. The recursive application production rules leads to the increasing possible complexity of each level of recursion. In this application, L-Systems are used to generate strings of characters that encode topological information to be interpreted later.

Here we will consider an example of a simple L-System from Prusinkiewicz & Lindermayer [4]. There is a binary alphabet $\mathbf{V} = \{A, B\}$, an axiom $\boldsymbol{\omega^0} = B$, and production rules $\boldsymbol{P} = \{A \rightarrow AB, B \rightarrow A\}$. The production rules are applied to the axiom recursively, resulting in a series of developmental stages of the string of characters. The axiom itself represents that zeroth stage, i.e. $\boldsymbol{\omega^0} = B$. Each subsequent stage is developed by simultaneously applying the production rules to each character. The first stage, $\boldsymbol{\omega^1} = A$, is obtained by substituting B with A as defined by the production rules, $\boldsymbol{P}$. Similarly, applying the production rules to $\boldsymbol{\omega^1}$ results in $\boldsymbol{\omega^2} = AB$

$$\boldsymbol{\omega^0} = B$$
$$\boldsymbol{\omega^1} = A$$
$$\boldsymbol{\omega^2} = AB$$
$$\boldsymbol{\omega^3} = ABA$$
$$\boldsymbol{\omega^4} = ABAAB$$
$$\boldsymbol{\omega^5} = ABAABABA$$
$$\boldsymbol{\omega^6} = ABAABABAABAAB$$

Table 2.1: Example of L-System from Prusinkiewicz & Lindermayer [4]

by substituting A with AB. The first six steps of this process are shown below in Table 2.1. The resulting string has no inherent geometric representation and therefore must be paired with a graphical interpreter to generate structural topologies. A graphical interpreter for this purpose is presented in Section 2.2.

To increase the modeling capability of the L-System, an addition of numerical parameters was introduced by Lindenmayer to allow for variable angle and distance definitions [12]. In this work, a parameterized L-System is consider consisting of the following: 1) an alphabet, 2) a set of formal parameters, $\boldsymbol{\Sigma}$, 3) a set of production rules, $\boldsymbol{P}$, and 4) an axiom, $\boldsymbol{\omega_0}$. The alphabet for this system is composed of six variable character, $\boldsymbol{\alpha} = \{A, B, C, D, E\}$ and $\boldsymbol{\beta} = \{-, +\}$, and two constant characters, $\boldsymbol{\gamma} = \{[,]\}$. The first set of four variable characters, $\boldsymbol{\alpha}$, describe operations which govern the creation of topology and/or material assignments. The second set of two variable character, $\boldsymbol{\beta}$ indicate the movement of the operator inside the given topology. The last set of constant characters, $\boldsymbol{\gamma}$, signify beginning and end of markers which are utilized to traverse the topology. The set of formal parameters, $\boldsymbol{\Sigma}$, are associated with the variable characters of the alphabet such that characters in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are defined as $\alpha_i = \alpha_i(\sigma_{\alpha_1}, \sigma_{\alpha_2})$ and $\beta_i = \beta_i(\sigma_{\beta_1})$, where $\sigma_{\alpha_1}, \sigma_{\alpha_2}$, and σ_{β_1} are real numbers from $\boldsymbol{\Sigma}$. The meaning of $\sigma_{\alpha_1}, \sigma_{\alpha_2}$, and σ_{β_1} take on different meanings depending on which variable character they are associated with. The axiom consists of two variable letter characters ($\boldsymbol{\omega_0} = \omega_1, \omega_2, \omega_i \in \boldsymbol{\alpha}$) and is used to initiate the recursive development of the final command string.

This particular system considers a set of five production rules, $\boldsymbol{P}$. Each production rule consists

of ten characters and is written as [1] $P_i : \alpha_i(\sigma_{\alpha_1}, \sigma_{\alpha_2}) \rightarrow \lambda_1^i\lambda_2^i(\sigma_{\alpha_1}, \sigma_{\alpha_2})_2\lambda_3^i(\sigma_{\alpha_1}, \sigma_{\alpha_2})_3\lambda_4^i(\sigma_{\beta_1})_4\lambda_5^i$ ... $\lambda_6^i\lambda_7^i(\sigma_{\alpha_1}, \sigma_{\alpha_2})_7\lambda_8^i(\sigma_{\alpha_1}, \sigma_{\alpha_2})_8\lambda_9^i(\sigma_{\beta_1})_9\lambda_{10}^i$, where $i = 1, \ldots, 5$ for all $\alpha_i \in \boldsymbol{\alpha}$, and $\lambda_1^i\lambda_2^i \ldots \lambda_{10}^i$ is a string of ten characters such that $\lambda_j^i \in \Lambda_j$ and $\Lambda_1 = \{[, \}, \Lambda_2 = \boldsymbol{\alpha}, \Lambda_3 = \boldsymbol{\alpha}, \Lambda_4 = \beta, \Lambda_6 = \{[, \}, \Lambda_7 = \boldsymbol{\alpha}, \Lambda_8 = \boldsymbol{\alpha}, \Lambda_9 = \beta$. Assignments for Λ_5 and Λ_{10} are made such that if the begin branch indicator, "[", is assigned in Λ_1 and Λ_6, it is closed Given the definition of the production rule above, and the need for each character in α and $beta$ to have two parameters and one parameter, respectively, each production rule requires 10 independent parameters. The 10 parameters plus 8 alphabet characters mean that 18 independent variables are required to define each production rule. Since there are five production rules and a two character axiom, this means that the entire L-System requires 92 independent variables.

2.2 Spatial Interpretation for the Design of Reconfigurable Structures

A tool is required to create two dimensional closed cell topologies within a given domain. To achieve this, the Spatial Interpretation for the Design of Reconfigurable Structures (SPIDRS) algorithm is utilized. This algorithm both interprets instructions from a parametrized L-System and performs graphical operations towards the creation of structure [35].

The SPIDRS algorithm operates on a half edge data structure, which represents the planar graph, I', in a way that is easy to manipulate and traverse [18]. This structure contains record of the nodes, edges and faces of the graph, I', and follow three rules. First, the graph must be simple, meaning that no nodes are connected to themselves via an edge, and no two edges connect the same two nodes. Second, edges are defined by the idea that each edge is composed of two half edges that are oriented in opposite directions and each belong to two distinct faces. Third, faces are defined as directed walks, meaning a face is an enclosed set of half-edges which are oriented such that one can start at a given node, traverse around the graph based on the orientation of the half-edges in the face, and return to the original node. This structure is also capable of carrying attribute information relating to specific nodes, edges, and faces, like material assignments.

In this algorithm, the graph, is defined by the nodes, N, edges, E, faces, F, and mate-

[1] Parameters belonging to λ_i^j are independent of any other parameters, i.e. $(\sigma_{\alpha_1}, \sigma_{\alpha_2})_i \neq (\sigma_{\alpha_1}, \sigma_{\alpha_2})_{i+1}$

rial (i.e. attribute information), M. For example, consider the graph, I', shown in Figure 2.1. The set of nodes, edges, and faces making up I' are defined as $N(I') = \{1, 2, 3, 4\}$, $E(I') = \{e_{12}, e_{23}, e_{34}, e_{41}\}$, and $F(I') = [1, 2, 3, 4]$.

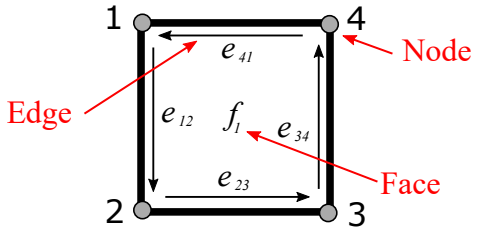


Figure 2.1: Example graph represented using half-edge data structure

A half-edge data structure is used in computational geometry to describe polygonal subdivisions due to ease of which the topological information of a planar graph. For instance, we will consider the graph, I', depicted in Figure 2.2 and observe what occurs when a subdivision is made in the graph between nodes 6 and 4. Before subdivison, the faces and materials of I' are described as $F(I') = \{f_1, f_2\} = \{[1, 2, 3, 6], [3, 4, 5, 6]\}$ and $M(I') = \{[S_1, S_2, S_3, S_4], [S_5, S_6, S_7, S_3]\}$. Notice that the material assignment is consistent for both half edges between nodes 3 and 6. Now consider the graph after subdivison occurs in f_2 between nodes 6 and 4. The faces and materials of I' are now defined as $F(I') = \{f1, f2, f3\} = \{[1, 2, 3, 6], [3, 4, 6], [4, 5, 6]\}$ and $M(I') = \{[S_1, S_2, S_3, S_4], [S_5, S_8, S_4], [S_6, S_7, S_8]\}$. This subdivision results in the creation of a new face, f_3, and a new pair of half edges, e_{46} and e_{64}. The material set is also updated to reflect the changes to I' regarding the new face and half edges written, in order to stay consistent. This half edge data structure is applicable only to convex graphical subdomains, because if the subdomain is not convex there is a possibility for newly created edges to intersect with preexisting non-convex edges.

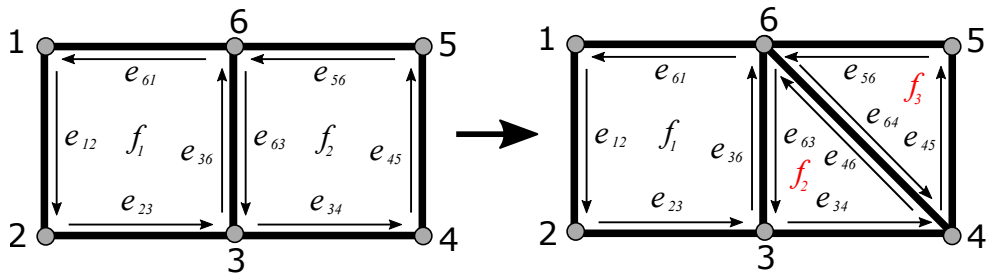


Figure 2.2: Example of how the half-edge data structure is updated to modify topology

2.2.0.1 *Move-Integer*

The move-integer command is represented in the L-System encoding as $A(\sigma_{\alpha_1})$. This command operates by moving the SPIDR by $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes in the current face, where N is total number of nodes in the current face. In this action, the SPIDR location is updated, but the topology of the graph is not modified (i.e. F and M are unchanged). An example of the move-integer command, $A(0.72)$, is shown below in Figure 2.3. In this example, assume that the SPIDR begins at node 1 and receives the command, $A(0.72)$. Since $N = 4$, $\lfloor N \times \sigma_{\alpha_1} \rfloor = \lfloor N \times 0.72 \rfloor = 2$, which causes the SPIDR to advance two nodes to node 3.

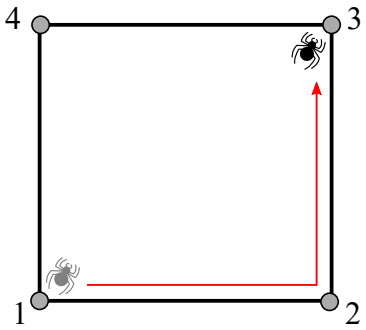


Figure 2.3: Example of a move-integer operation $A(0.72)$

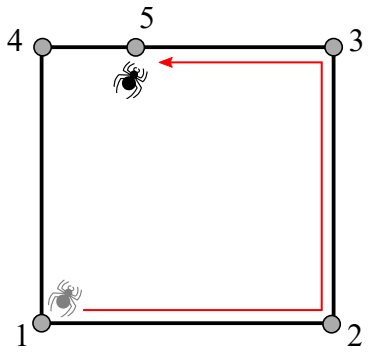


Figure 2.4: Example of a move-real operation $B(0.7)$

2.2.0.2 Move-Real Operation

The move-real command is represented in the L-System encoding as $B(\sigma_{\alpha_1})$. This command operates by the moving the SPIDR by $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes in the current face, and then to a newly created node $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor$ in between the current node and next node. An example of the move-integer command, $B(0.5)$, is shown below in Figure 2.4. In this example, assume that the SPIDR begins at node 1 and receives the command, $B(0.7)$. Since $N = 4$, $\lfloor N \times \sigma_{\alpha_1} \rfloor = \lfloor N \times 0.7 \rfloor = 2$, which causes the SPIDR to advance two nodes to node 3. Next, node 5 will be created between nodes 3 and 4 by moving the SPIDR $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor = 2.8 - 2 = 0.8$ of the distance between them. Since this action created a new node, the topology of the structure must be updated to reflect this change and is modified by updating F and M as follows:

$$F = \{[1, 2, 3, 4]\} \rightarrow \{[1, 2, 3, 5, 4]\},$$

$$M = \{[S_1, S_2, S_3, S_4]\} \rightarrow \{[S_1, S_2, S_5, S_3, S_4]\}.$$

The addition of the new node requires the addition of a material assignment to ensure that the edge and material descriptions remain consistent. The material assigned to the new edge is consistent with the preexisting edge, i.e. $S_2 = S_5$. Notably, this command is restricted such that $0.1 \leq N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor \geq 0.9$. Commands that result in $0.1 < N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor < 0.9$ are set to 0.1 and 0.9 respectively.

2.2.0.3 *Create-Integer Operation*

The create-integer command is represented in the L-System encoding as $C(\sigma_{alpha_1}, \sigma_{alpha_2})$. This command operates by moving the SPIDR by $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes in the current face before constructing and edge back to it's starting location. The σ_{alpha_2} parameter determines what material the newly constructed edge will be assigned. An example of the create-integer command, $C(0.51, 0.2)$, is shown below in Figure 2.5. In this example, again assume that the SPIDR begins at node one when receiving the command. Since N = 4, the SPIDR advances $\lfloor N \times \sigma_{\alpha_1} \rfloor = \lfloor 4 \times 0.51 \rfloor = 2$ nodes forward to node three, similarly to the move-int command. Next, the SPIDR creates a new edge from the current node, node three, to the original node, node one. This edge is assigned the material associated σ_{α_2}. Since this action subdivided a prexsisting face, the topology of the structure must be updated to reflect this change and is modified by updating F and M as follows:

$$F = \{[1, 2, 3, 4]\} \rightarrow \{[1, 2, 3][1, 3, 4]\},$$

$$M = \{[S_1, S_2, S_3, S_4]\} \rightarrow \{[S_1, S_2, S_5][S_5, S_3, S_4]\}.$$

After the subdivision occurs, the SPIDR is defined to be along its original half edge. In this example, the SPIDR originates from node one, meaning its initially along edge e_{12} and as such will stay in the new face which contains edge e_{12}. This command will not perform any action if the two nodes between which a new edge will be created are colinear. This constraint is enforced to prevent overlapping faces or material assignments.

2.2.0.4 *Create-Real Operation*

The create-real command is represented in the L-System encoding as $D(\sigma_{\alpha_1}, \sigma_{\alpha_2})$. This command operates by moving the SPIDR by $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes in the current face, then to a newly created node $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor$ in between the current node and next node, and then creating a new edge between the newly created node and the original node and returning the SPIDR to the original node. The σ_{alpha_2} parameter determines what material the newly constructed edge will be

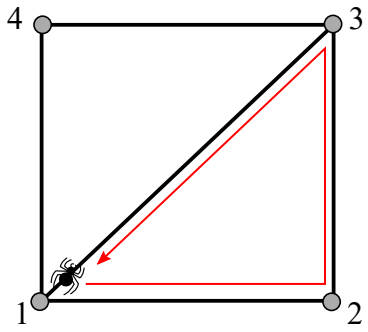


Figure 2.5: Example of a create-integer operation $C(0.51, 0.2)$

assigned. An example of the create-real command, $D0.7, 0.2$ is shown belown in Figure 2.6. In this example, again assume that the SPIDR begins at node one when receiving the command. Since N = 4, the SPIDR advances $\lfloor N \times \sigma_{\alpha_1} \rfloor = \lfloor 4 \times 0.7 \rfloor = 2$ nodes forward to node three. Next, node 5 will be created between nodes 3 and 4 by moving the SPIDR $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor = 2.8 - 2 = 0.8$ of the distance between them, similarly to the move-real command. Lastly, the SPIDR will create an edge between the current node, node 5, and the original node, node 1, and return to the original node. The material of the newly created edge is a function of σ_{alpha_2}. Since this action subdivided added a node to the topology and divided a preexisting face, the topology of the structure must be updated to reflect this change and is modified by updating F and M as follows:

$$F = \{[1, 2, 3, 4]\} \rightarrow \{[1, 2, 3, 5][1, 5, 4]\},$$

As described before, the SPIDR is defined to be along its original half edge after this operation is performed. This action will also subdivide the applicable half edges (i.e. $e_{34} \rightarrow [e_{35}, e_{54}]$) and their applicable twins (i.e. $e_{43} \rightarrow [e_{53}, e_{45}]$). This command is also restricted by the ability to create nodes within a certain range of other nodes and the creation of collinear nodes.

2.2.0.5 Change Material Operation

The change material command is represented in the L-System encoding as $E(\sigma_{\alpha_1})$. This command operates by changing the material of the current half edge to a material specified by σ_{α_1}, and

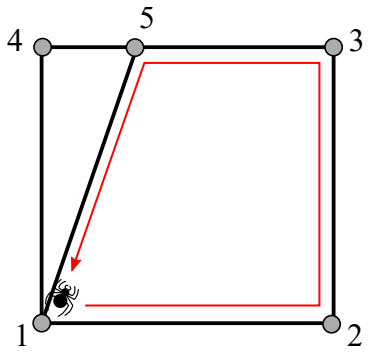


Figure 2.6: Example of a create-real operation $D(0.7, 0.2)$

then moving the SPIDR by one node. An example of the command, $E(0.63)$, is shown below in Figure 2.7. In this example, the SPIDR starts at node one, on the half edge e_{12}, with the material S_1 (shown as black lines). The SPIDR changes the material of the current half-edge e_{12} to H_1, which is the material associated with a σ_{α_1} value of 0.63. The material H_1 is shown in blue in the figure. The SPIDR also advances forward one node to node 2. It is important to note that the material assignment for the twin of the modified half edge, e_{21}, will also be updated to maintain material consistency in the graph. This command does not change the face set F.

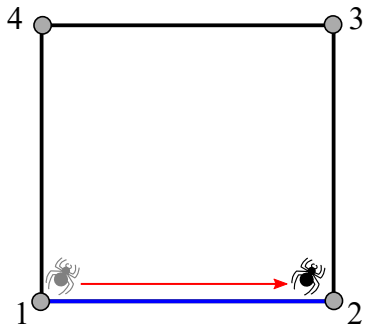


Figure 2.7: Example of a change material operation $E(0.63)$

2.2.0.6 Turning Operations

There are two turning commands defined in the L-System encoding, $+(\sigma_{\beta_1})$ and $-(\sigma_{\beta_1})$. These commands are defined for clockwise and counterclockwise motion, respectively. Each of these

commands are based on the number of faces, M, the current node is associated with. The SPIDR will move in $\lfloor M \times \sigma_{\beta_1} \rfloor$ in the direction prescribed by the command. An example of both the clockwise and counter clockwise commands are shown in Figure 2.8. First we will consider Figure 2.8a where a clockwise command, +(0.9), is depicted. In this example, the SPIDR is positioned at node 5 along half edge, e_{45}. Since node 5 is shared between two faces, M = 2, making $\lfloor 2 \times 0.9 \rfloor = 1$, the SPIDR proceeds to the next counter clockwise face. The SPIDR is now located in $F = [5, 3, 4]$ along half edge e_{53}. An example of a counter clockwise turning operation, $-(0.5)$ is depicted in Figure 2.8b. In this example, the SPIDR again originates at node 5 along edge e_{54}. In this case, node 5 is associated with three faces, making $M = 3$. Therefore, the SPIDR advances $\lfloor 3 \times 0.5 \rfloor = 1$ face in the counter clockwise direction. The SPIDR's new location is updated from $F = [5, 4, 1] \rightarrow F = [1, 2, 5]$, and is now positioned along the half edge e_{51}. When this command is associated with a node that is only present in a single face, the command is ignored.

(a) $+(0.9)$ (b) $-(0.5)$

Figure 2.8: Examples of turning operations

2.2.0.7 Bracket Operations

Bracket operations are represented in the L-System encoding as [and]. These commands represent the beginning and end, respectively, of a particular string of SPIDR commands. These

commands operate in conjunction with one another by the open bracket saving a particular location of the SPIDR agent, and the close bracket eventually returning the SPIDR to the most recently saved location (with out the modification of the topology). Notably, any operations that are encoded into the command string between these two operators will be performed by the SPIDR agent, meaning that these bracket operators can be nested. Saved locations are also continuously updated to correspond to the current topology, in order to prevent face subdivision or new node creations from invalidation the previously saved location.

2.3 Truss-based Nonlinear Mechanical Analysis of Origami Structures

In order to evaluate a kinematic objective of folded origami patterns, a tool is required to simulate the folding process. A modified nonlinear truss model that was derived for small small strain/larger behavior rotation is used. This model was selected for its ability to balance efficiency and accuracy as described by Gillman et al. [1]. This model is based off of a positional finite element truss model and is modified to include a torsional spring around the elements to represent the fold stiffness between adjacent facets. This section will outline the minimum energy principles and their application to the modified truss element, the linearization of these non-linear equations, the application of boundary conditions, and the numerical analysis tool used to solve the resulting system of equations. This model was first developed and described by Gillman et al. [1].

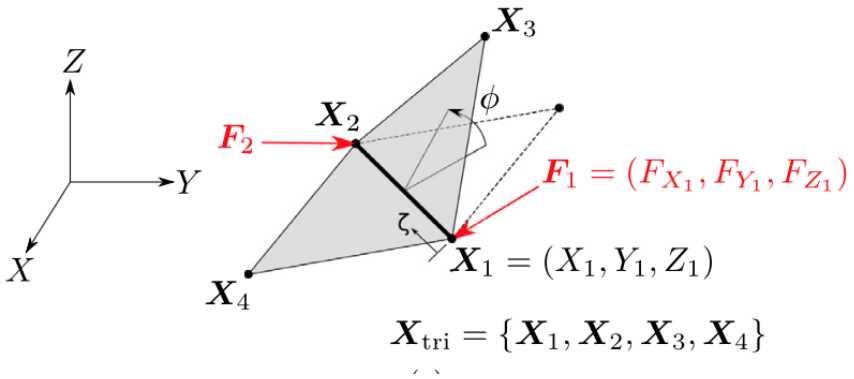


Figure 2.9: Schematic of modified truss element[1]

The schematic of a single element of the modified truss is shown in Figure 2.9, where the fold angle is depicted as the relative angle between the two triangular facets. The total energy, Π, of the element is defined as the total energy difference between potential, U_t, and external P, as shown in Equation 2.1. The potential energy of the truss element is given by Equation 2.2, where E is the Young's modulus of the truss, A is the cross sectional area of the truss, G is the torsional spring constant (per unit length),and ζ is the non-dimensional integration length dimension along the axial

direction of the truss. The energy per unit length of the truss is represented by u_t and the energy per unit length along the hinge is represented by u_h.

$$\Pi = U_t - P \tag{2.1}$$

$$U_t = l_0 \int_0^1 \frac{EA}{2}\epsilon(\boldsymbol{X}_1, \boldsymbol{X}_2)^2 + \frac{G}{2}\tilde{\phi}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4)^2 d\zeta = l_0 \int_0^1 u_t + u_h d\zeta \tag{2.2}$$

The deformation in the element is captured by the axial strain, ϵ, and the rotation in the spring is captured by, $\tilde{\phi}$. Both ϵ and $\tilde{\phi}$ are formulated in terms of the global position (Equations 2.3 and 2.4) and therefore naturally capture the non-linear geometry of the motion, despite the assumption of a linear constitutive model for both axial and torsional deformation.

$$\epsilon = \frac{(|\boldsymbol{X}_2 - \boldsymbol{X}_1| - l_0)}{l_0} = \frac{\sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2} - l_0}{l_0} \tag{2.3}$$

$$\tilde{\phi}(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4) = \phi(\boldsymbol{X}_1, \boldsymbol{X}_2, \boldsymbol{X}_3, \boldsymbol{X}_4) - \phi_0 \tag{2.4}$$

A penalty function is used to enforce local contact and avoid singularities. This penalty function is defined as $p(\phi) = C((\frac{\phi}{\pi})^B) + 1$, where C and B are constants. Equation 2.5 shows the potential energy equation (Equation 2.2) with the addition of the penalty function. The constant B is related to the rate of fold stiffness increase near the closed state, while C effects the nominal amount of stiffening. Noticeably the penalty is a function of ϕ and not $\tilde{\phi}$, meaning that the penalty is a function of only fold angle between the two facets.

$$U_t^P =_0 \int_0^1 u_t + p(\phi)u_h d\zeta \tag{2.5}$$

After the introduction of the penalty term, Equation 2.1 becomes $\Pi = U_t^P - P$. Finally, the external energy, P, is defined in Equation 2.6. In this equation, the forces at the end of the truss element are multiplied by the displacement of the global nodes. In both Equations 2.2 and 2.6, i in X_i, Y_i, and $Z_i (i \in [1, 4])$ are the local node numbers.

$$P = F_{X_1}(X_1 - X_1^0) + F_{X_2}(X_2 - X_2^0) + F_{Y_1}(Y_1 - Y_1^0) + F_{Y_2}(Y_2 - Y_2^0)$$

$$+ \mathrm{F}_{Z_1}(Z_1 - Z_1^0) + F_{Z_2}(Z_2 - Z_2^0) \tag{2.6}$$

The principle of minimum potential energy is utilized to determine the equilibrium state of the structure and results in the following system of nonlinear equations shown in Equation 2.7. To solve this system of nonlinear equations numerically, a linearization of the residual $R_i(X_{tri})$ is performed using a Taylor's series expansion and yields Equations 2.8, where $X_{tri} = \{X_1, Y_1, Z_1, X_2, Y_2, Z_2, X_3, Y_3, Z_3, X_4, Y_4, Z_4\}$ is the set of global coordinates for the local nodes that define the fold angle ϕ and the index of i iterates through all of the components in the set $X_t ri$.

$$\frac{\partial \Pi}{\partial X_i} = l_0 \int_0^1 \frac{\partial u_t}{\partial X_i} + (Gp(\phi)\tilde{\phi} + G\frac{\tilde{\phi}^2}{2}\frac{\partial p(\phi)}{\partial \phi})\frac{\partial \phi}{\partial X_i} d\zeta - F_{X_i} \tag{2.7}$$

$$R_i(X_{tri}) = \frac{\partial \Pi}{\partial X_i} = R_i(X_{tri}, F_i) = f(X_{tri} - F_i) = 0$$

$$R_i(X_{tri}) \approx R_i(X_{tri}^0) + \Delta R_i(X_{tri}^0)\Delta X_{tri} = 0 \tag{2.8}$$

The tangent term is defined as $K_{ik} = \Delta R_i(X_{tri}^0)$ and expanded below in Equation 2.9. This system of equations is solved iteratively until equilibrium is achieved using the Newton-Raphson method. This system of equations is solved for each applied load step until either the entire load is applied to the structure, or numerical convergence cannot be reached.

$$K_{ik} = f_{i,k} = l_0 \textstyle\int_0^1 \frac{\partial u_t}{\partial X_k \partial X_i} + Gp(\phi)(\frac{\partial \phi}{\partial X_k}\frac{\partial \phi}{\partial X_i} + \tilde{\phi}\frac{\partial^2 \phi}{\partial X_k \partial X_i})$$

$$+G\frac{\partial^2 p(\phi)}{\partial \phi^2}\frac{\tilde{\phi}}{2}(\frac{\partial \phi}{\partial X_k}\frac{\partial \phi}{\partial x_i} + \tilde{\phi}\frac{\partial^2 \phi}{\partial X_k \partial X_i}) + 2G\tilde{\phi}\frac{\partial p(\phi)}{\partial \phi}(\frac{\partial \phi}{\partial X_k}\frac{\partial \phi}{\partial X_i})d\zeta \tag{2.9}$$

2.4 Finite Element Analysis

Finite element analysis (FEA) is a method utilized to solve for stresses and displacements in structure. These models are utilized in order to solve problems that would otherwise be difficult to obtain exact solutions for, due to complex geometries or boundary conditions. This form of computational analysis is also much faster and cheaper than traditional prototyping, therefore making it an ideal tool for preliminary design and optimization. In this work, FEA is completed in Chapter 3 using Abaqus, a commercial FEA software produced by Dassault Systems [73].

FEA works by discretizing a structure into elements based on a mesh applied to the structure. The displacements are then solved for at each of the vertices (nodes) in the mesh and assembled to return the global displacements. Each element is assigned material properties and experiences the forces applied to itself by its neighboring elements. The material properties and the shape of the elements determine stiffness. The stiffness of each element is calculated and assembled into a large global stiffness matrix, k. The boundary conditions are then utilized to describe a global vector, F, in which the limited or imposed degrees of freedom are considered. The linear FEA problem is defined by the equation $F = ku$, where u is a vector of nodal displacements. In Abaqus, this equation is solved using by incrementing the applied loads through 'time' (i.e. load steps) and running the Newton-Raphson method to approximate u during each load step. For more detailed information on FEA, reference Reddy[74]. For more specific information on Abaqus/CAE, reference the Abaqus User Manual [73].

2.5 Aerodynamic Analysis

In order to determine the shape of an object under flow, like an airfoil, the structural implications of the aerodynamics forces need to be taken into consideration. In order to accomplish this, a tool is necessary to approximate the aerodynamic properties of the given structure under flow. In this work, a simple pressure distribution of the fluid onto the structure of interest is required. To accomplish this, an aerodynamic analysis tool, called XFOIL is utilized.

The 2D aerodynamic analysis was performed using the XFOIL panel method code. XFOIL's

higher order panel method relies on a linear vorticity stream function to solve for the flow field and streamlines along the boundary of the airfoil. Since the linear vorticity theory assumes inviscid flow, the code also utilizes a viscous boundary layer component to predict drag coefficients and improve fidelity compared to purely inviscid implementations. The wake and boundary layer are described using a two-equation lagged dissipation integral boundary formulation with an e^n transition model. This viscous boundary layer model was presented and described in detail by Drela and Giles in 1987 [75].

XFOIL is intended for use in application with small to moderate boundary layer thickness (low Reynolds numbers) and angles of attack up to stall. XFOIL has been used previously as a low fidelity aerodynamic analysis tool to evaluate morphing airfoil concepts in similar flow regimes in work completed by Woods et al [46].

2.6 Genetic Algorithm

As previously mention in Section 1.2.1, preliminary design requires the coupling of a topology creation method with an optimizer. In this work, a genetic optimizer is utilized due to its ability to handle difficult design domains with non-continuous design parameters and multiple objectives. Henceforth the genetic algorithm will be discussed in terms of the methods and process utilized for optimization in Sections 3 and 4. In this work, all optimizations are performed using a single set of optimization functions to perform initialization, selection, crossover, and mutation.

To initialize the GA, an initial population of size N is created using the uniform random number function from the NUMPY Python package. The uniform random number function works by returning a random number between given bounds with a uniform distribution between the upper and lower bound. In this application the function is bounded from 0 to 1 for all applications. Next, these arrays of numbers are mapped to the respective SPIDRS operational characters described in Section 2.2. The L-System is then utilized to encode the genes as described in Section 2.1. Lastly, the encoded genes are interpreted using the rules described in Section 2.2 leading to the creation of structures to evaluate in the GA.

The next step of the GA is to evaluate the fitness of each individual in the population. The

nature of this evaluation is dependent on the problem being considered, and for the purposed of this chapter can be considered as a black box. The fitness values are then utilized to perform selection between the current population and the offspring using the well known NSGA2 algorithm. NSGA2 is a non-dominated sorting algorithm for problems with multiple objectives [28].

The methods used to evolve the population are crossover and mutation. These functions operate on the selected population to create a new population of a similar size, which will henceforth be reference to as the offspring. Crossover is performed on two individuals in the population at a given time using the a bounded simulated binary crossover. The bounded simulated binary crossover is similar to the one described by Deb and Agrawal [76] and allows the user to assign a variable that defines the probability of the offspring similarity to the parent. In this application, the parameters used for the simulated binary boundary function are: a lower bound of 0, an upper bound of 1, and a crowding degree value of 20.0 (meaning offspring should appear relatively similar to parent). Notably, crossover is not performed on all individuals in the population in an attempt to preserve good characteristics from previous generations. In this application, the crossover probability for any given pair of individuals is 90%.

For mutation, a real-parameter mutation operation was conducted with a mutation probability of 1/n, n being the length of the design variable. This operator serves to maintain diversity in the evolving population. Mutation is performed on specific genes in individual when the mutation probability is met. The new value of the gene is selected from a bounded polynomial distribution from 0 to 1. The crossover, selection, and mutation operations are all similar to those utilized by Deb at al. to develop the NSGA2 algorithm [28].

The Distributed Evolutionary Algorithms in Python (DEAP) is a software framework implemented in the Python programming language that allows users to rapidly build evolutionary computational systems to apply the methods described above. DEAP acts as a framework to provide sophisticated and customizable evolutionary computational systems while maintaining also prioritizing code clarity and compactness [77]. These goals are accomplished by implementing two main structures: the creator and the toolbox. The creator module allows the creation of classes

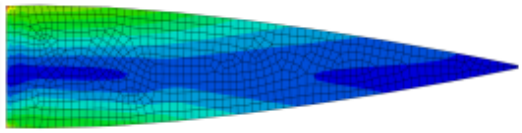
for the individual genotype. These classes allow for updates through out the evaluation and enable the algorithm to evaluate genotypes of various data types such as lists, dictionaries, or trees. The toolbox acts as a container for the tools that are used as operators in the evolutionary computational system. Each operator in the toolbox is populated manually by the user with selected tools. For example, if a selection algorithm is required in the evolutionary computational system, the user will register a "selection" tool into the toolbox and assign it to use the NSGA-ii algorithm for selection. DEAP offers many options for the selections algorithm, so the user could then decided to later adjust the selection tool by assigning a tournament style selection algorithm if needed without altering the main body of the code.

3. STRUCTURAL TOPOLOGY OPTIMIZATION OF A MORPHING AIRFOIL

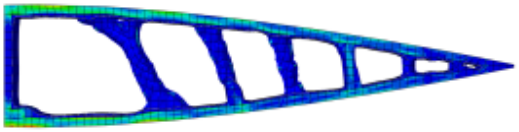
3.1 Introduction

Helicopters are one application in which camber morphing airfoils are extremely useful. Unlike most planes, helicopter blades are largely static without camber altering control surfaces. The airfoil section used in these rotor blades ends up being a utilitarian design that is not optimized for any single phase of the flight path, but instead is a compromise that can operate well under several flight conditions. Mission performance can be significantly increased by implementing a camber morphing rotor blade section to morph between optimal airfoil shapes during the different stages of a mission. Traditionally the design of a morphing rotor blade is conducted using a combination of density based topology optimization methods coupled with engineers prior knowledge of airfoil strcutural design to parameterize a shape that can later undergo sizing optimization. An example of this process is shown below in Figure 3.1.

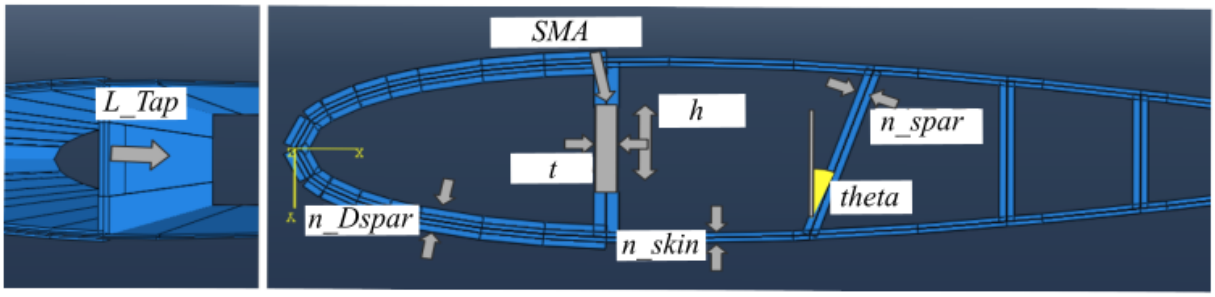
In this work, students attempted to optimize the internal structure of a rotor blade section to facilitate blade twist along the span. Their first step used the TOSCA structural topology optimization package in Abaqus and applied it to a meshed solid region with the OML of the section. This optimization iteratively deletes elements that experience low stress compared to other elements in the mesh until the desired volume fraction is achieved or no elements can be deleted without adverse side effects in the remaining elements. The resulting topology of this process is shown in Figure 3.1b. The topology is then verified by repeating this process with several different meshes in order to confirm the results, since this method's solutions are prone to mesh dependency. Finally, a person is required to evaluate the resulting topologies and determine parameters that can be mapped to a predetermined structural model. This method requires the user to subjectively map the results to the structural optimization domain and does not consider the placement of actuators in the domain. In Figure 3.1c, we can observed how the topological information from Figure 3.1b is transformed into structural parameters that will later be optimized.



(a) Meshed solid airfoil with stress contour



(b) Resulting optimal topoglogy using TOSCA in Abaqus with stress contour.



(c) Resulting parameterized topology dervied from b

Figure 3.1: Density based topology optimization on helicopter rotor airfoil

The SPIDRS design methodology is an alternative to this traditional approach and offers the ability to evaluate many different topologies as well as the placement of actuators in the airfoil. In order to apply SPIDRS to this design problem, a process was generated to create topologies inside of an initial airfoil and then simulate morphing under aerodynamic loading. The initial airfoil shape and final target shape were predetermined and corresponded to optimal airfoils under specific flight conditions (i.e. hover and cruise). To generate internal topologies, the SPIDRS algorithm is altered to accommodate non-linear domains (c.f. Appendix A) and then used to create the internal structure and place actuators to morph the airfoil. The resulting designs are then evaluated to determine the torsional stiffness of the airfoil and how well the shape change matched the final targeted airfoil. This process is shown with the specific function evaluation method in Figure 3.2.

The UH-60 rotor blade was the motivation for this problem. The target shapes and internal structures derived inspiration from the UH-60, but are unlikely to match exactly due to the proprietary nature of the UH-60 blade's internal structure and exact outer mold line.

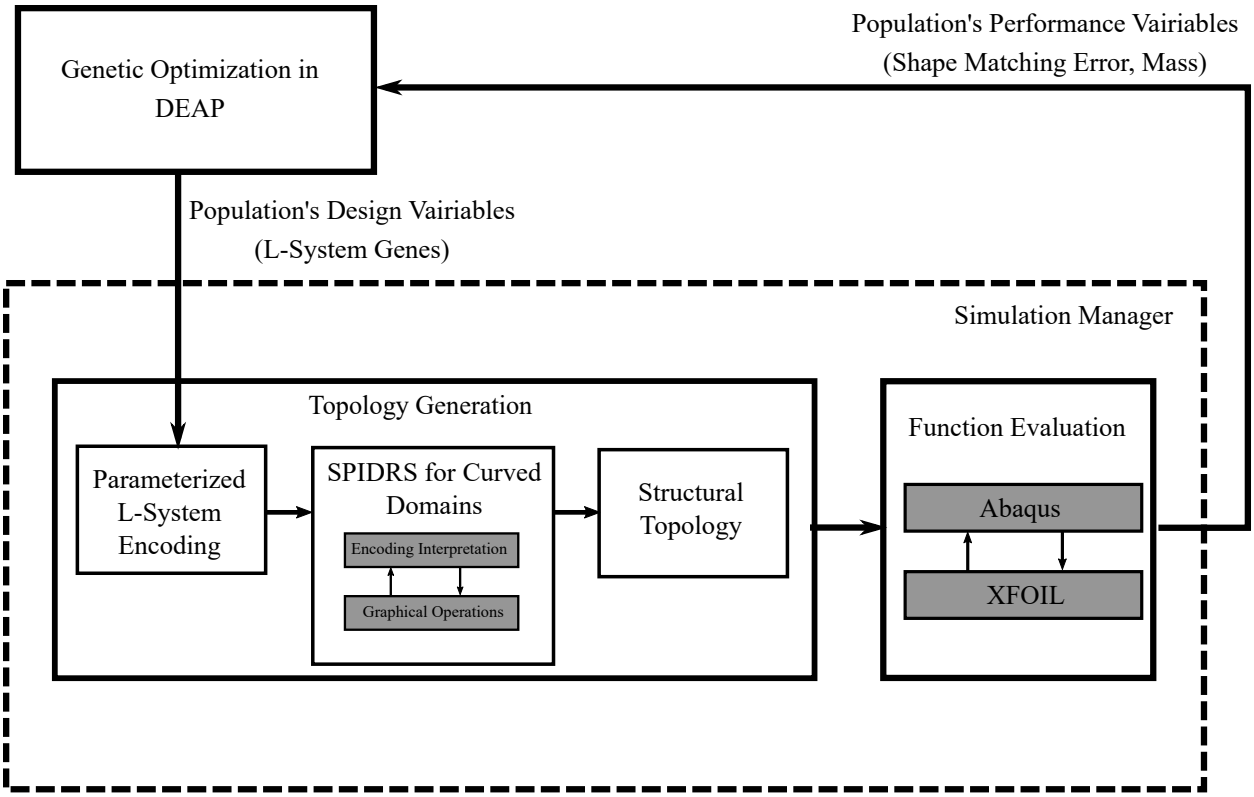


Figure 3.2: Topology design and optimization approach using an updated SPIDRS interpretation scheme and coupled aerodynamic and structural functional evaluations

3.2 Problem Setup

In order to apply L-System based design to the UH-60 blade, a schematic of the original internal structure was needed. However, due to the proprietary nature of the UH-60, the internal structure of the S-76 blade was used to inform the structure [2]. This rotor blade is dominated by a very stiff spar near the leading edge and supported by a Nomex honeycomb core aft of the spar, as shown in Figure 3.3. The Nomex honeycomb core prevents actuators from being inserted internally to the airfoil, so we consider using SPIDRS to generate internal structures to replace the structural support offered by the Nomex and enabling camber morphing by placing actuators in this domain. The stiff spar near the leading edge is maintained, but the shape was adapted to a D-Spar to allow for a simpler SPIDRS domain geometry.

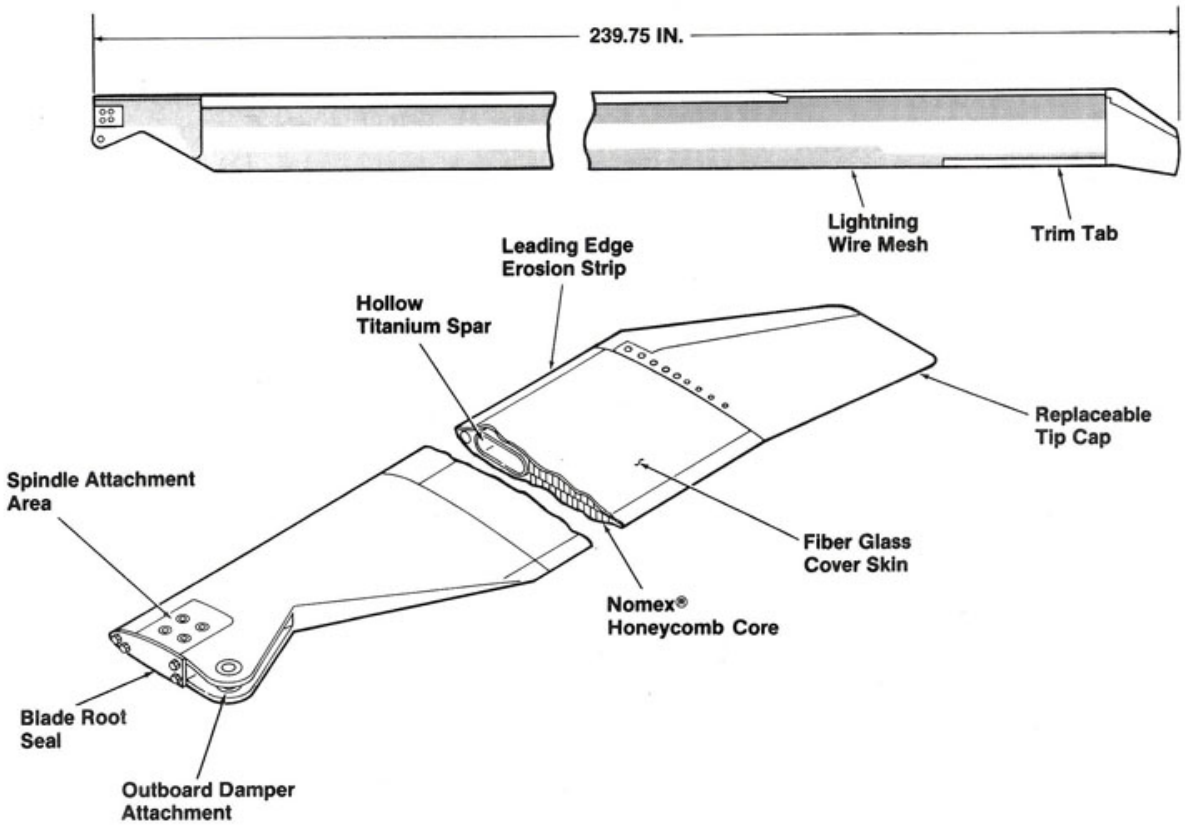


Figure 3.3: S-76 Main Rotor Details [2]

The blade cross section being considered in this problem has a chord of 0.57 meters and is constructed using carbon fiber reinforced composites for the structural members and skin. The initial OML of the airfoil is given by a class shape transformation (CST) function, with the parameters listed in Table 3.2. Actuation is provided by shape memory alloy plates embedded into the wing structure. An internal shear spar is located at 30% chord and the leading edge of the wing is dominated by a D-spar. Aft of the shear spar, the skin is defined by the outer mold line, but the internal structure is empty. This empty domain is pictured in Figure 3.4 and is called the SPIDRS design domain. In this area, the internal structural member and actuators are generated by the SPIDRS algorithm described in Section 2.2. The material assignments for all of the sections of the airfoil are shown in Table 3.1.

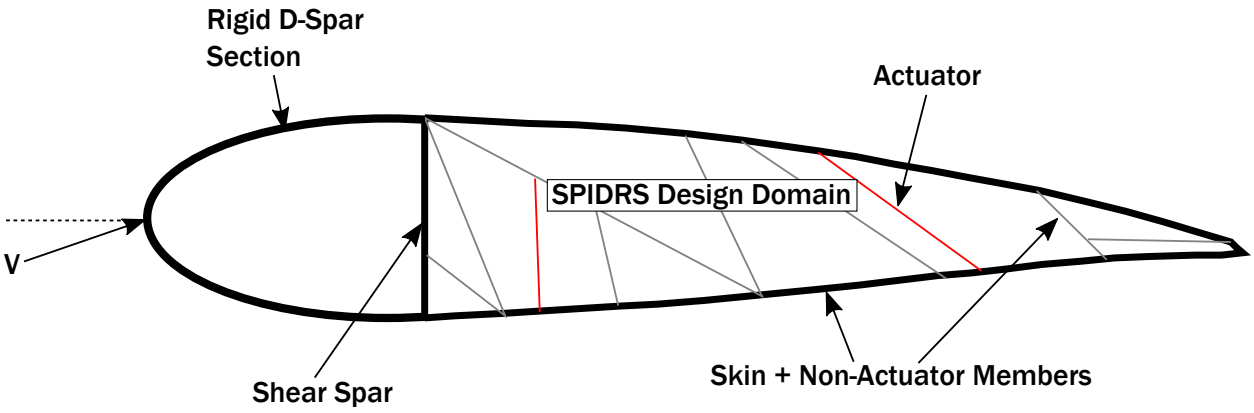


Figure 3.4: Problem setup for camber morphing airfoil generated with SPIDRS

Part	Material	Layup	Ply Thickness (mm)
D-Spar	Graphite Epoxy	$[0_2/\pm 45/90_2/\pm 45/0_2]_s$	0.2
Shear Spar	Graphite Epoxy	$[0_2/\pm 45/90_2/\pm 45/0_2]_s$	0.2
Skin	Graphite Epoxy	$[0_2/\pm 45/\pm 45]_s$	0.2
Non-Actuator Members	Graphite Epoxy	$[0_2/\pm 45/\pm 45]_s$	0.2
Actuators	SMA	-	2.0

Table 3.1: Class shape transformation parameters for initial and target airfoil shapes

The goal of this problem is to morph the airfoil to a target airfoil shape from an intial shape. The shapes of both the final target shape and the initial shape are given by the sets of CST parameters listed in Table 3.2 and is shown in Figure 3.5. The final shape is aerodynamically favorable during cruise and experiences both aerodynamic and structural loads due to morphing. The error between the final morphed shape of the blade under load and the target shape is the objective of this problem. Additionally, the internal structure of the airfoil must be able to provide a sufficient amount torsional rigidity for the wing to eventually be scaled into a 3D model and enough static rigidity to maintain its initial shape under aerodynamic loading.

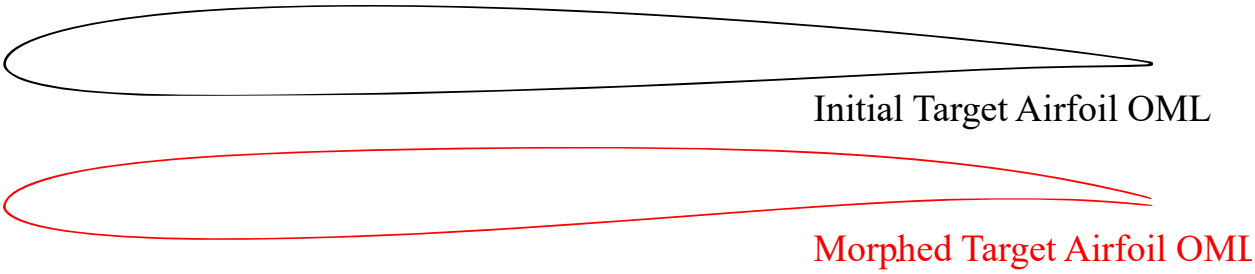


Figure 3.5: Target shapes for the initial and morphed conditions as defined by the CST parameters in Table 3.2.

3.2.1 Structural Analysis

In order to evaluate the structural properties of the 2D airfoil section outlined above, a common finite element model was created. This section details the steps taken to create this model and the methods utilized. First, the OML of the airfoil was sketched onto a 2D plane using a spline sketching tool in Abaqus. This spline is constructed using a set number of non-dimensional points along the chord, and then transformed into the problem domain. Notably, the trailed edge of this airfoil is left open in the Abaqus model to prevent skin buckling during actuation, which has previous occurred in similar problems [37]. The spline is also amended by including any nodes from the SPIDRS topology that have been created along the OML. This inclusion later allows for the effective assignments of materials, sections, loads, and boundary conditions. Second, the internal topology is obtained from SPIDRS and sketched on the same 2D plane. The result is a completely 2D sketch of the airfoil which is then extruded 0.02 m (2 cm) along the span. This sketch consists of shells, meaning the the structures created during this phase of model must eventually be meshed using shell elements. This type of element reduces the computational time required by the FEA, because they do not consider stress in the direction perpendicular the shell surface. This assumption is valid for shapes with very little relative thickness, like the skin and actuator materials being utilized in this model.

Next, the materials for each part are generated with their respective material properties and assigned. The graphite epoxy lamina considered has the following properties: $E_{11} = 142e^9$ Pa,

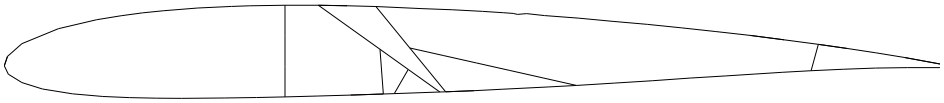
CST Parameters				
	Initial		Target	
Coefficient	Upper	Lower	Upper	Lower
c_1	0.170605	0.134677	0.170837	0.134860
c_2	0.163512	0.036237	0.136972	0.063118
c_3	0.141582	0.110329	0.168583	0.082946
c_4	0.1841999	-0.013120	0.371892	-0.013120

Table 3.2: Class shape transformation parameters for initial and target airfoil shapes

$E_{22} = 0.979e^9$ Pa, $\nu_{12} = 0.42$, $G_{12} = G_{21} = 5.93e^9$, and $/rho = 1580$ kg/m^3. This lamina is assembled into the various composite layups listed in Table 3.1. The lamina's thickness is 0.0002 m (0.2 mm). The layups utilized are all balanced and symmetric to avoid unpredictable mechanical couplings in the material that could effect the morphing.

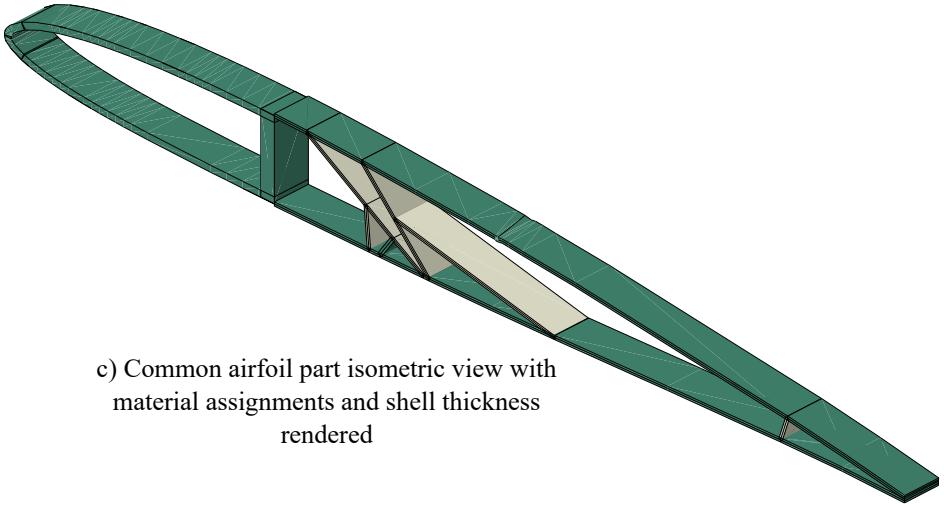
The material assigned to the actuator is intended to mimic a piece of shape memory alloy. However, full modeling of SMAs in Abaqus can be computational expensive, so an alternative approach is utilized. The actuator material is assigned a similar Young's Modulus ($E = 74.24e^9$ Pa) and density ($\rho = 6440.0$ kg/m^3) as the SMA, and actuated to a comparable actuation strain by setting the thermal coefficient and heating the material to achieve the desired strain. In this case, the actuation material has a 2% strain imposed during actuation by setting the thermal coefficient, α, to 0.02, and applying a thermal field with a magnitude value of 1.

The result of this process is a 3D shell part onto which loads and boundary conditions can be placed. The shell part is shown in Figure 3.6a. Depending on the analysis being done, the method of load application is different. However, for both the torsional rigidity analysis and the pressure displacement analysis, the meshing of the part is similar. The part is meshed using S4R shell elements with a global seed size of 0.01. The resulting mesh is shown below in Figure 3.7. Generally an extruded shell part would be meshed one with one element of depth, but in this application the part is meshed with two elements in the span wise direction. This mesh was chosen to allow for better elements aspect ratios in the smaller internal structural members.



a) Common airfoil part sideview

b) Common airfoil part sideview with materials assignments and shell thickness rendered



c) Common airfoil part isometric view with material assignments and shell thickness rendered

Figure 3.6: Common structural model in Abaqus.

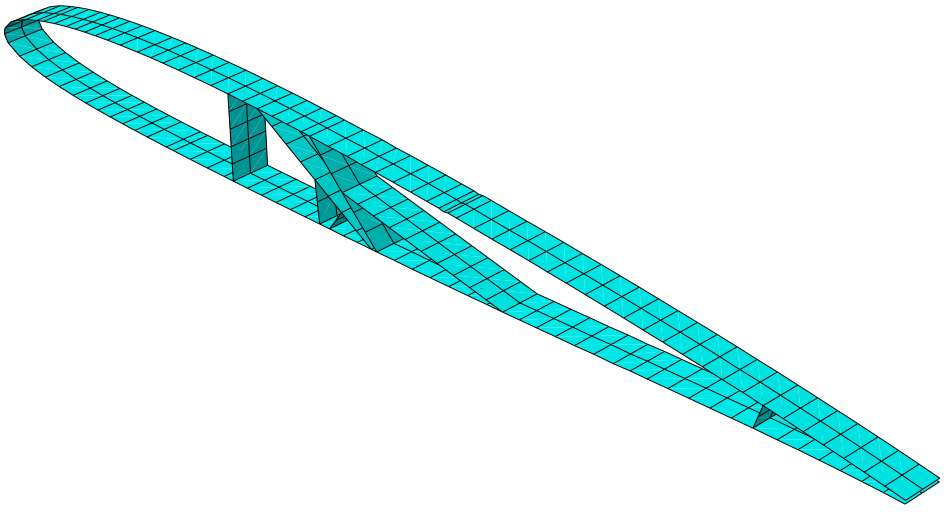


Figure 3.7: Mesh of common structural model in Abaqus.

3.2.1.1 Torsional Rigidity

In order to find the torsional rigidity of the airfoil, Equation 3.1 is utilized. T represents the the torque applied to a structure, θ represents the angle by which the structure rotates under a given T, L is the length along the axis of rotation, and GJ is the torsional rigidity of the structure.

$$\frac{T}{\theta} = \frac{GJ}{L} \tag{3.1}$$

To model this testing processing Abaqus, a infinitely rigid part is created and tied to one of the span wise edges in the common structural model. The other span wise edge is encastred to prevent any translations or rotations. Then, a moment is applied to the rigid part, which imparts the moment onto the structure. This application of the moment is meant to mimic the internal moment that would be created along the span of a full rotor blade model that was being twisted. Finally, the resulting change of angle is measured on the rigid part and utilized to calculate the torsional rigidity of the part. The boundary conditions for this processes are shown below in Figure 3.8.

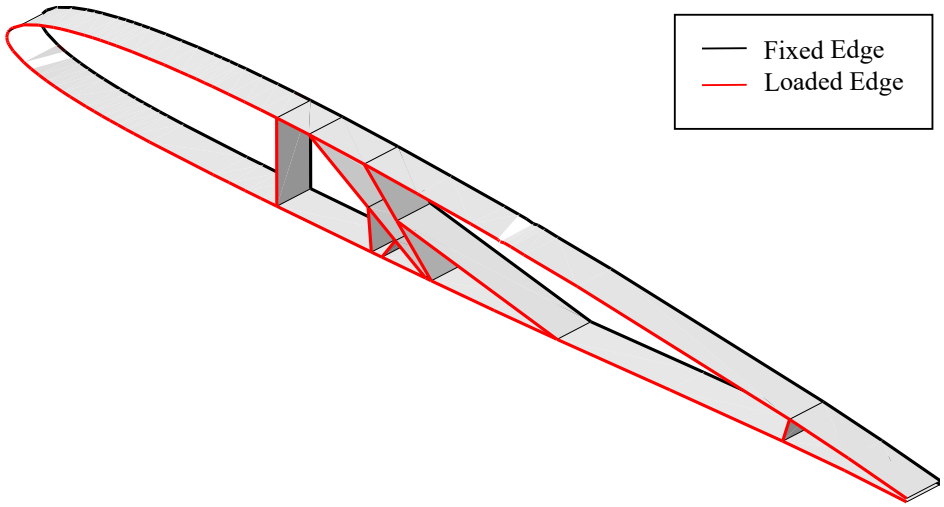


Figure 3.8: Boundary conditions applied to structural model in Abaqus.

3.2.1.2 Outer Mold Line Displacement

In order to model the displacement of the OML, pressure loads and actuation loads are considered. This function evaluation is completed by applying a pressure load to the common structural model described above as shown in Figure 3.10. These pressure distributions are generated by the aerodynamic evaluation using XFOIL (c.f. Section 2.5). The pressure load is applied at the span wise edges and the center line of the airfoil. In addition to the pressure loads, a thermal load is applied to the entire structure. The thermal load is applied via a thermal field in Abaqus and has a magnitude of 0 or 1, where 0 represents 0% actuation and 1 represents 100% actuation. This thermal field is what induces morphing in the structure.

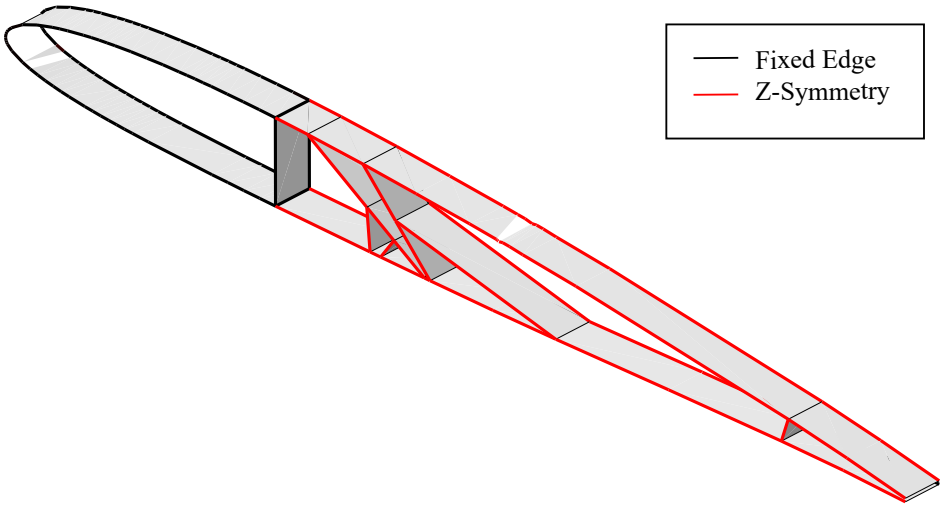


Figure 3.9: Boundary conditions applied to structural model in Abaqus.

There are two boundary conditions applied to the airfoil. First, the leaded edge D-Spar and Shear spar are encastered to prevent and translations or rotations. This condition is to ensure that each design is fixed similarly in space regardless of the internal structure generated by SPIDRS. Second, the span wise edges have a Z-symmetry condition imposed. This symmetry is used, because this small spanwise section is theoretically a part of a much larger wing or rotor. These

boundary conditions are displaced graphically in Figure 3.9.

After analysis is completed, the resulting output database of the model is post processed in order to find out the displacements and stresses in the design. Post processing works by querying elements that have been previously assigned to sections in the model based on geometric location or material properties. The displacement is queried for all nodes on the OML surface of the airfoil and saved as a Python dictionary for evaluation of the shape objective. The Von Mises stress is queried for each element in its respective material section and the maximum value for each material is saved. This stress is later used to check the stress constraint in the optimization.

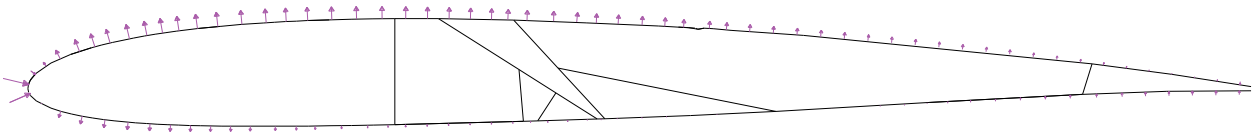


Figure 3.10: Pressure load applied to structural model in Abaqus.

3.2.2 Aerodynamic Analysis

In order to evaluate the aerodynamic loads on on the 2D airfoil section, an aerodynamic anlaysis was performed using the XFOIL panel method code in conjunction with the AeroPy Python package. XFOIL has been used previously as a low fidelity aerodynamic analysis tool to evaluate morphing airfoil concepts in similar flow regimes in work completed by Woods et al [46]. In our application, the AeroPy module is also utilized to facilitate easy data handling and transformations [**?**]. Here, the steps taken to find the pressure distribution for the airfoil as well as the parameters used, will be described in detail.

First, the shape of the airfoil is described by listing the upper and lower coordinates in order from trailing edge to leading edge and leading edge to trailing edge respectively. The python XFOIL Module developed by Leal[**?**] is then used to normalize these coordinates along the chord and rotate the airfoil as appropriate for usage in the XFOIL program. Second, the normalized

and rotated coordinates are saved to a text file for the XFOIL software to later read from. Third, the XFOIL module is used to submit the command to open the XFOIL program with the correct aerodynamic parameters and file names. In this work, the Reynolds number considered was approximately $3.2e6$ and was calculated based on the dynamic viscosity of air at 2000 m, with a free stream velocity of 81 m/s, and a chord of 0.57 m. The angle of attack utilized in the XFOIL evaluation is 0 $\deg$. From the XFOIL program, the pressure, lift, and drag coefficients are returned, but not the forces imparted on the airfoil itself. Fourth and finally, from the pressure coefficients, the geometry of the airfoil section, and the aerodynamic parameters, the pressure forces are calculated using the XFOIL module. The result is an array of pressure values that can then be applied to the structural model as shown in Figure 3.10.

3.2.3 Fluid Structure Interaction

In this problem, the structure needs to undergo an evaluation to determine the shape of the airfoil under aerodynamic loading for both the initial and morphed configurations. This evaluation must consider both the structural deformation caused by the aerodynamic pressure load and the changes in the pressure load due to shape changes caused by structural deformation or actuation. A coupled fluid structure interaction (FSI) analysis is utilized to concurrently solve for the boundary conditions on the airfoil and the final OML of the morphed configuration. An FSI analysis operates by coupling a method for structural and aerodynamic evaluations by sharing information about the updated configurations between the two evaluations. In this work, a loosely coupled FSI analysis in Abaqus and XFOIL is utilized. The data flow for this method is outline in Figure 3.11.

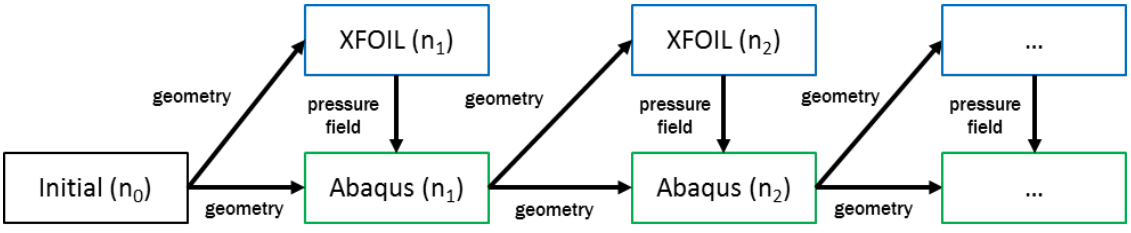


Figure 3.11: Loosely coupled fluid structure interaction scheme [3]

In this coupling, the initial airfoil shape is utilized to determine the initial pressure distribution. Next, the resulting deformed geometry is utilized to find the new pressure distribution and to create an updated structural model that reflects the deformed shape. This process is iterated until geometric convergence is achieve or the maximum iterations occurs. In this work, geometric convergence is defined by two points at the trailing edge of the airfoil. Therefore, convergence is reached when the tip of the trailing edge converges to a single location in space ($\pm 0.001m$) for two consecutive solutions.

In an FSI analysis, there is always the possibility that the aerodynamic and structural solutions are not able to find a concurrent solution for both the aerodynamic and structural models. To account for this, a limit on the number of iterations between the structural and aerodynamic analysis is implemented. In this work, an upper limit of 15 iterations is used. This simple numeric upper limit is used in other similar work [49] and guarantees a manageable run time for the coupled evaluation.

3.2.4 Genetic Optimization Framework

To set up the genetic optimization, the objectives and constraints of the problem need to be formalized. Previously, we have noted that the resulting airfoil design should be capable of maintain the prescribed initial shape under aerodynamic loads, should change shape when actuated to match the targeted airfoil shape, and should not lack torsional rigidity. In order to evaluate these requirements, a process was developed to describe the data flow required to evaluate each design for the GA. This process is shown in Figure 3.12.

Notably, this flowchart does not depict an FSI evaluation to determine that the airfoil design is capable of maintain the initial prescribed shape. To decrease the computational cost of this problem, the airfoils are assumed to match the initial sufficiently well if they pass the torsional rigidity check. This assumption is made, because member that provide torsional rigidity should also provide stiffness in the airfoil to prevent major shape change due solely to the aerodynamic loads.

This geometric mean square error is calculated by comparing the OML of the morphed airfoil

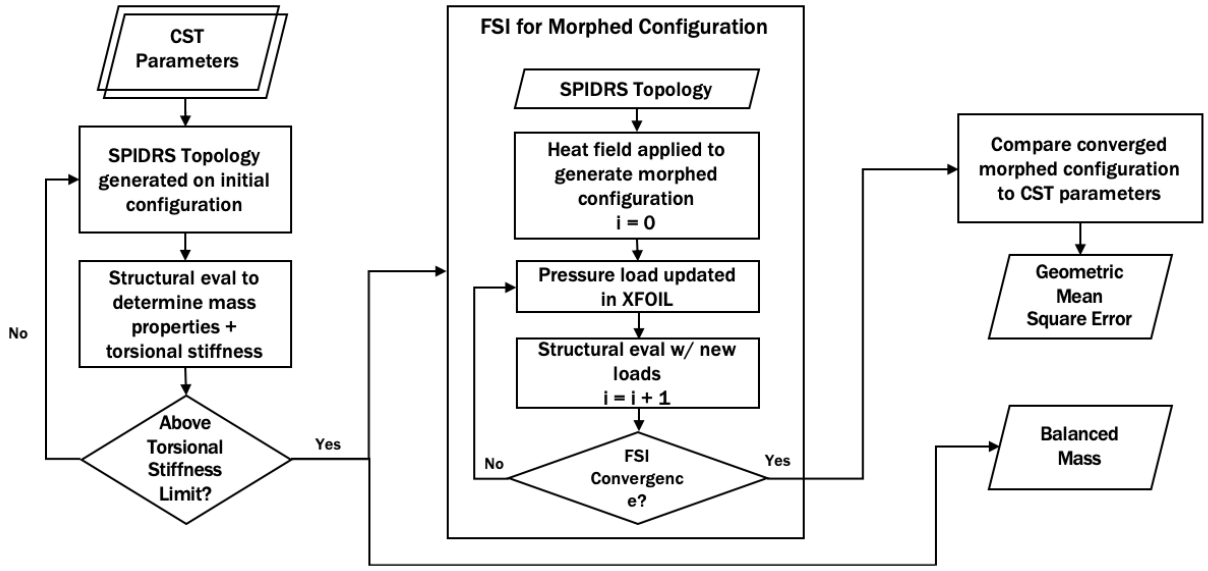


Figure 3.12: Flowchart describing the data flow for any single design evaluation

under aerodynamic loading to the objective airfoil shape. To ensure consistency, the x-coordinates of the list containing the deformed OML are used to calculate a list of objective y-coordinates using a polynomial defined by the CST parameters. The square of the difference between each y-coordinate is summed to represent the total shape error of the deformed configuration, and the average of this value is taken as the mean square error.

Mass is considered as the second objective and is necessary to prevent the genetic algorithm from populating solely with actuating members to provide stiffness. Without this objective, there is no penalty for utilizing actuating members for structural stiffness, despite their much higher density and cost of implementation.

The problem is constrained in order to penalize designs that violate stress constraints or without sufficient torsional rigidity. If a design fails a constraint, a penalty value of 1000 is assigned to the individual for the shape objective in the GA.

Using the methods, objectives, and constraints outlined above, an optimization is performed with 100 members per generation and 100 generations. This totals to 10,000 function evaluations

for the optimization. A large population size was utilized to ensure genetic diversity in the first generations to prevent premature convergence to local minima.

Design Problem Statement	
Minimize:	Shape Error and Mass
by varying:	2 axiom characters,
	5 rule assignments (18 genes each)
subject to:	$GJ \geq 100000GPA/m$,
	$\sigma^i_{VM} < \sigma^i_{crit}$
NSGA-II Parameters	
100 members for 100 generations	
$P_{cross} = 0.9, \eta_{cross} = 50$	
$P_{mutation} = 1/92, \eta_{mutation} = 50$	

Table 3.3: Genetic optimization problem statement and NSGA-II parameters

The genetic optimization is conducted using the DEAP package (c.f. 2.6) to manage generation, selection, mutation, and crossover. A uniform random number generator is used for the generation of each genome in the original population. Selection is performed using the NSGA2 algorithm (c.f. 2.6 for background on NSGA2). The original generation is evaluated and then mating and mutation are performed. In this application, mating is performed using the cxSimulatedBinaryBounded tool in DEAP. This mating process executes a simulated binary crossover that modfies in-place the input individuals, which is utilized due to it's similarity to the cross over presented in the original NSGA2 work. Mutation is performed using the mutPolynomialBounded tool in DEAP. This mutation scheme was also selected due to it's original implementation in NSGA2. The specific parameters for the crossover and mutation tools are listed in Table 3.3.

3.3 Results

The optimization returned 48 Pareto optimal solution. Their objective values and associated genes are listed in Table 3.4 and are plotted in Figure 3.13. The results shown in blue indicate all of the viable designs from the optimization and the red dots show the Pareto optimal solutions, with a red line connecting these solutions to approximate the Pareto frontier.

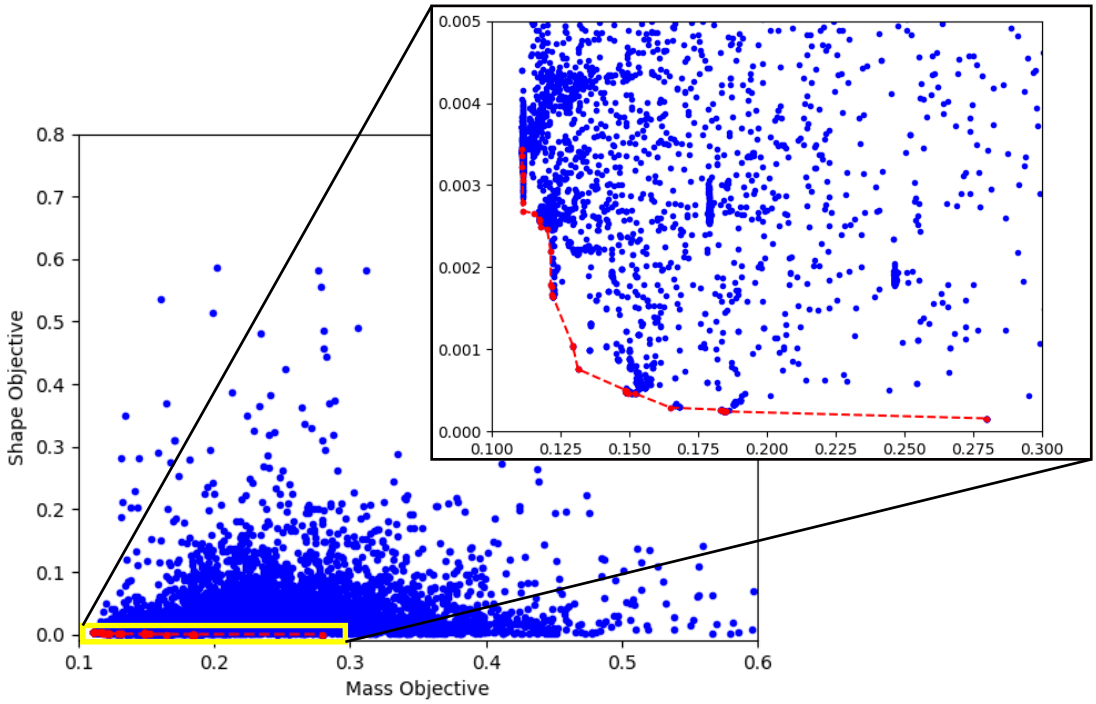
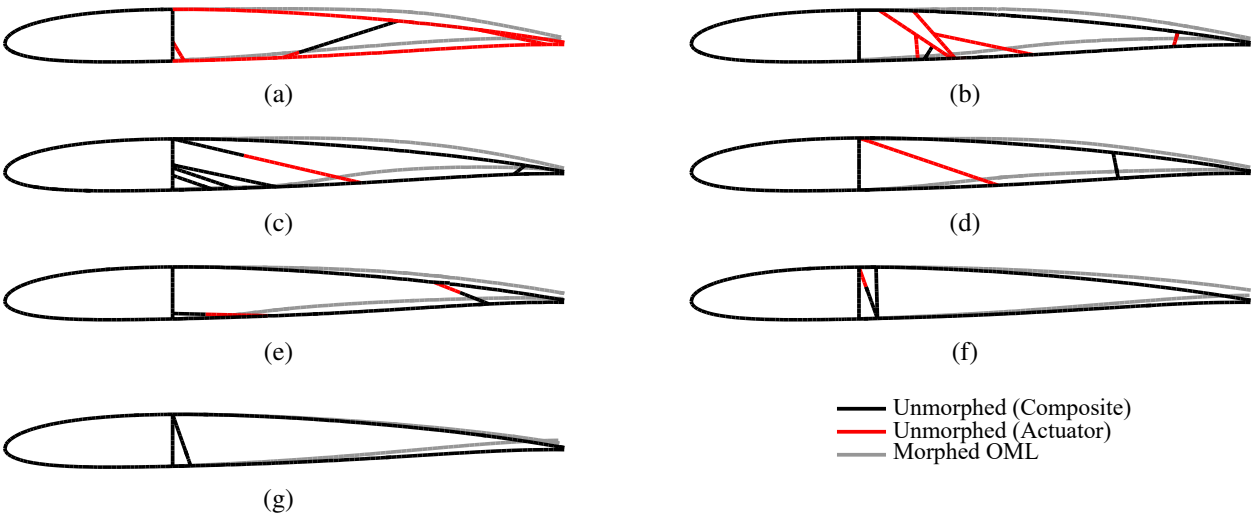


Figure 3.13: Pareto front for morphing airfoil design optimization.

Upon examination of the results, it is clear that the Pareto front is segregated into several clusters that produce similar masses and shape performance values. Representative examples of these clusters are shown in Figure 3.14. Interestingly, there is only one design on the pareto front that includes actuator material on the OML of the airfoil, design a (Table 3.4 item 48). This design also happens to be the best performer in terms of the shape matching objective and matches the target airfoil with an objective value of 0.000158 as shown in Figure 3.15.



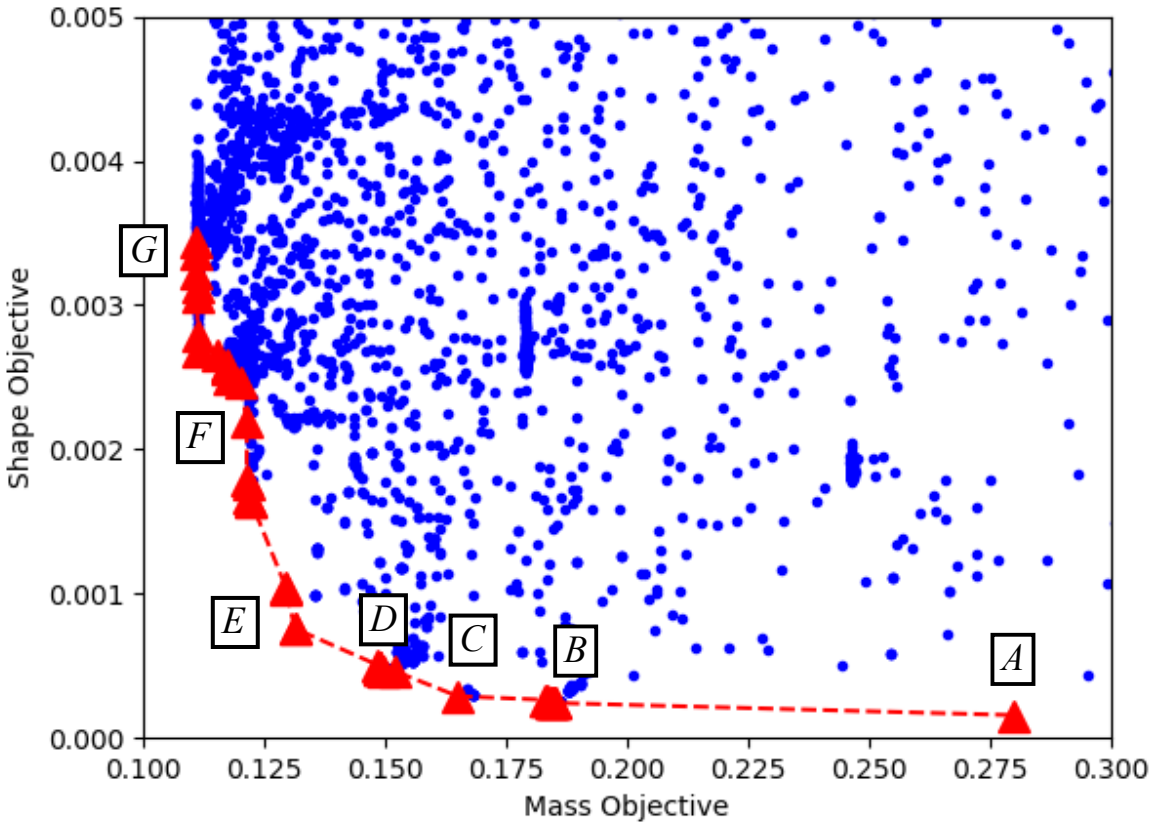


Figure 3.14: Pareto front with 7 representative designs from areas of the Pareto front

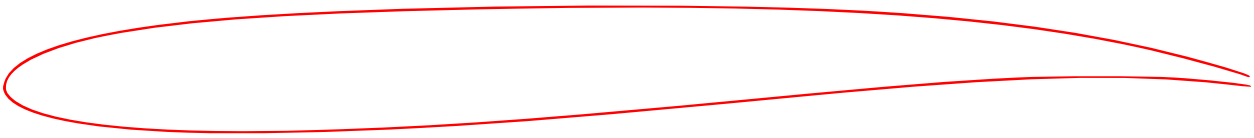
Figure 3.14b is representative of a cluster of 11 solutions (Table 3.4 items 38-47) that have similar geometries and similar shape objectives values of approximately 0.0002. In this cluster, all of the designs have the same number of structural and actuator members that have a very similar topological layout. The nodal locations between these designs tend to vary slightly though, which cause small differences in the length or angle of structural members. The differences between these designs are a result of the real number parameters used by the SPIDRS algorithm to create the topologies.

Figure 3.14c is representative of a single solution (Table 3.4 item 37) on the Pareto front. This design is the most similar topology to traditional morphing airfoil design, in that it places several spars throughout the structure and uses a single actuator to operate as a tendon to deform the airfoil. On the other hand, Figure 3.14d represents the extreme version of this type of design. In this cluster of 12 designs (Table 3.4 items 25-36) we notice very similar mas objectives and shape objectives of roughly 0.00048. These designs operate similarly to item 37, in that they contain a single actuator acting as a tendon to contract and deform the airfoil, but exhibit much less structure. As a result, they are obviously lighter but also the resulting shape is not as close to the targeted morphed condition.

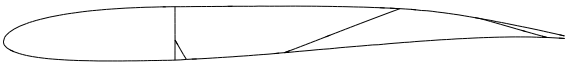
Figure 3.14e is representative of the solution closest to the utopia point on the Pareto front (Table 3.4 item 24). This design has very little structure, but is able to meet some shape requirements by distributing the actuator material through the structural members that do exist. There are two other points near e on the Pareto front that feature similar topologies, but have different location of the actuation material (Table 3.4 item 22-23). Up to this point, all of the designs on the Pareto have featured a small structural connection in the trailing edge of the airfoil. This makes sense for this problem implementation, because the trailing edge of this airfoil is left open in the structural model to prevent skin buckling upon actuation.

Figures 3.14f and 3.14g represent the rest of the designs on the Pareto front. These two designs represent the topologies utilized by the remaining 21 Pareto optimal designs. These designs vary much more significantly in terms of shape objective due to different placements of actuation ma-

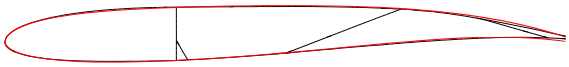
terial, but the mass stays relatively similar. Most of the deformation experienced by these designs is a result of the aerodynamic loading and not the actuation material.



(a) Target morphed airfoil shape

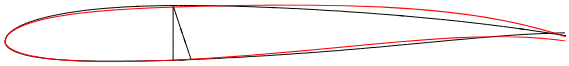


(b) Final morphed airfoil with best shape fitting objective



(c) Target airfoil overlaid onto best shape fitting design

(d) Final morphed airfoil with lowest mass



(e) Target airfoil overlaid onto lowest mass design

Figure 3.15: Best shape and mass objective airfoils in comparison to the target morphed shape

By comparing the best shape matching objective to the worst in Figure 3.15, we can see that this process was able to find a very good shape match and very light designs. What this also shows is that the shape objective matching for this problem is likely much more important than the mass objective, because the ideal mass objective problem will return an empty airfoil design.

Pareto Optimal Solutions					
#	Mass Objective (kg/0.02m)	Shape Objective	#	Mass Objective (kg/0.02m)	Shape Objective
1	0.1109	0.003440	25	0.1486	0.000496
2	0.1109	0.003361	26	0.1487	0.000491
3	0.1109	0.003217	27	0.1487	0.000490
4	0.1111	0.003129	28	0.1487	0.000489
5	0.1111	0.003059	29	0.1487	0.000488
6	0.1112	0.002794	30	0.1488	0.000487
7	0.1112	0.002686	31	0.1488	0.000485
8	0.1153	0.002655	32	0.1488	0.000484
9	0.1173	0.002583	33	0.1488	0.000482
10	0.1173	0.002568	34	0.1489	0.000481
11	0.1174	0.002562	35	0.1490	0.000473
12	0.1178	0.002487	36	0.1494	0.000461
13	0.1201	0.002468	37	0.1521	0.000459
14	0.1214	0.002189	38	0.1650	0.000289
15	0.1214	0.001787	39	0.1832	0.000264
16	0.1218	0.001757	40	0.1833	0.000632
17	0.1219	0.001664	41	0.1841	0.000252
18	0.1223	0.001642	42	0.1843	0.000250
19	0.1223	0.001633	43	0.1843	0.000248
20	0.1295	0.0010429	44	0.1846	0.000247
21	0.1295	0.001034	45	0.1847	0.000246
22	0.1312	0.000763	46	0.1848	0.000244
23	0.1312	0.000759	47	0.1849	0.000243
24	0.1484	0.000505	48	0.2798	0.000158

Table 3.4: Objective values for Pareto optimal results.

4. ORIGAMI FOLD PATTERN DESIGN AND OPTIMIZATION FOR KINEMATIC OBJECTIVES

4.1 Introduction

In order to apply L-System based design to aductive origami design problems, a process was generated to create topologies to represent origami fold patterns and then simulate the resulting folded shape. In this process fold patterns are generated in a specific design domain according to the rules describe in Sections 2.1 and 2.2. The fold patterns are then rotated and/or translated, if any symmetry conditions are present in the problem that can simplify the design domain. Finally, the fold pattern is evaluated by simulating the fold process as described in Section 2.3. This process is shown graphically in Figure 4.1. The user is able to change to problem being evaluated by changing the initial design domain of the problem, the boundary conditions applied to the pattern during the folding simulation, and/or the objective being evaluated.

This work first considers the well establish square twist pattern, which has previously been used for abductive origami design, as a validation problem. After validation this process is applied to a general continuous kinematic response objective. Here, the goal is to discover origami fold patterns that enables an objective node in the fold pattern to follow a targeted kinematic response throughout the folding process. The targeted response can be chosen somewhat arbitrarily, because the overarching objective of the work is to valid this process as a design tool for these types of problems. The specifics kinematics of these objectives will be discussed explicitly in their respective sections.

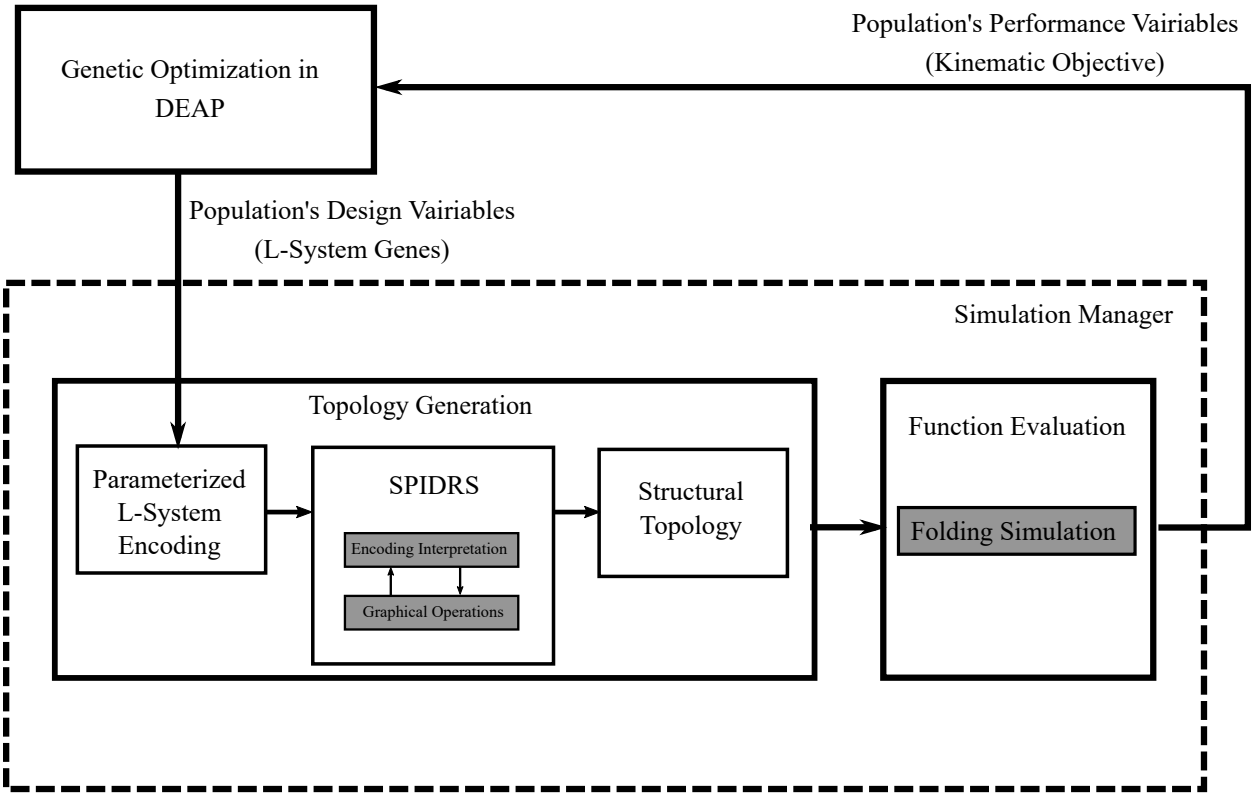
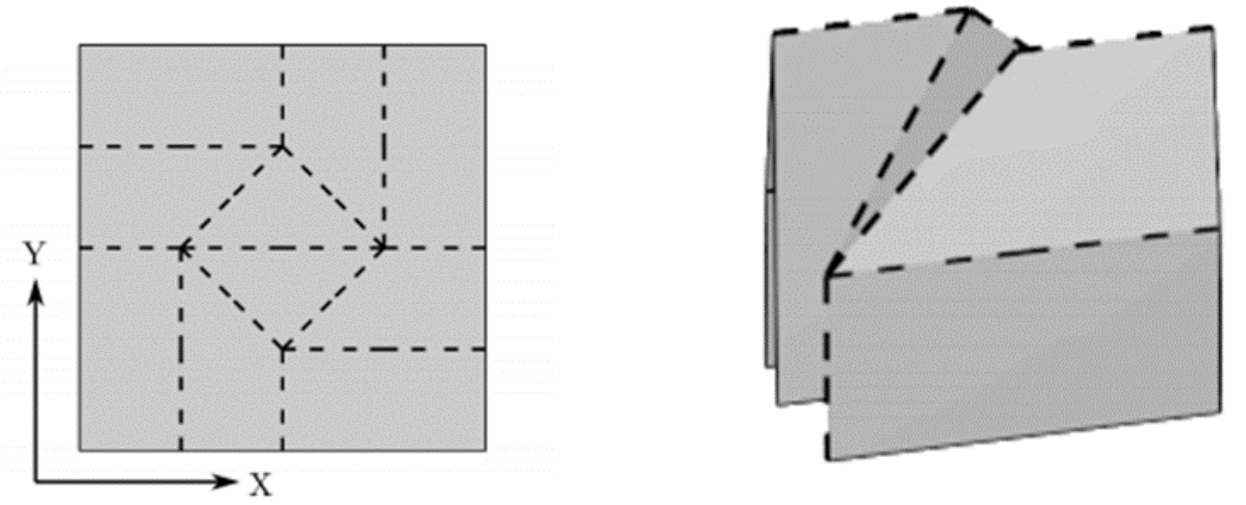


Figure 4.1: Fold pattern generation and simulation

4.2 Process Validation with Establish Fold Pattern

4.2.1 General Problem Setup

The square twist pattern was selected for evaluation due to the abundance of literature detailing its kinematic folding process [78, 79, 80], as well as previous work utilizing the truss based folding simulation (c.f. Section 2.3) to simulate this pattern [81, 1]. The ideal square twist pattern is shown in Figure 4.2a. This pattern requires an inward twisting and folding motion to achieve the flat foldable configuration shown in Figure 4.2b. The square twist is a well known pattern, but displays a relatively complex kinematic motion to achieve the flat fordable configuration and has not been modeled accurately using ground structure or ice cracking methods, making it an ideal benchmark for this novel design process. As previously mentioned, the setup of any given origami design problem is posed by determining the initial design domain and the boundary conditions and meta-parameters for the folding simulation. The details of these setups are described below.



(a) Square twist fold pattern

(b) Flat foldable configuration of the square twist fold pattern

Figure 4.2: Square twist origami pattern in its flat and folded configuration

4.2.1.1 Initial Design Domain and Symmetry

The design domain for the square twist pattern is the first quadrant of a square graph, as shown in Figure 4.3. This initial design space has a domain and range of $[0, 1]$ so that it can be easily scaled to any other length as needed. The initial set of nodes (N), edges (E), and faces (F) considered in the SPIDRS algorithm are

$$N = \{1 : (0, 0), 2 : (0.5, 0), 3 : (1, 0), 4 : (1, 1), 5 : (0, 1), 6 : (0, 0.5)\},$$

$$E = [[1, 2, 3], [3, 4], [4, 5], [5, 6, 1]],$$

$$F = [[1, 2, 3, 4, 5, 6]].$$

Nodes 2 and 6 are included to ensure consistent application of boundary conditions in the folding simulation. The material assignments from the SPIDRS algorithm are not utilized and all generated edges are considered active fold lines.

The initial domain is rotated counter clockwise about the origin to populate the three remaining quadrants. Due to the rotationally symmetric nature of the square twist pattern, it is logical to simplify the problem to a single repeatable unit cell of the pattern. This choice of design domain decreases computational complexity and is an important step for solving any problem that exhibits known symmetries in the solution. This process is demonstrated visually in Figure 4.3.

The entire pattern is then triangulated, using Delaunay triangulation[1], to make it compatible with the assumptions of the folding simulation. Any new edges are recorded and treated as inactive fold lines by the simulation.

The selection of the initial domain and symmetry conditions are based largely on the kinematic objective that is being optimized and the boundary conditions required to facilitate that. The application of boundary conditions in the folding simulation must be consistent across various designs and necessitates the existence of nodes at specific points. In this problem, nodes are added

[1] a triangulation scheme to maximize the minimum angle of all angles of the resulting triangles [**?**]

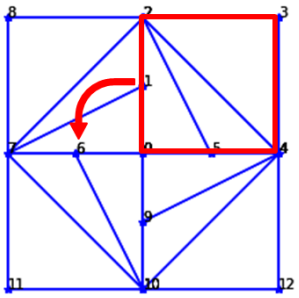


Figure 4.3: Population of origami fold pattern using rotational symmetry about the origin of the SPIDRS graph (outlined in red)

at $(0.0, 0.5)$ and $(0.5, 0.0)$ to facilitate boundary condition application in the folding simulation. Additionally, the rotational symmetry is selected to assist in facilitating boundary conditions by allowing the fold pattern to have a central anchor point at the origin.

4.2.1.2 Boundary Conditions and Folding Simulation Parameters

The boundary conditions and loading for the square twist problem are shown below in Figure 4.4. To run the folding simulation, the center node is fixed, and two nodes are constrained to move along the x-axis. These boundary conditions are needed to prevent rotation about the central node in the folding simulation, since moments are not constrained in the truss element. The load applied to the design in the folding simulation is a prescribed displacement on the two outer most nodes on the x-axis. The displacement is in the X and Z direction and applied in an arc. The input displacement in incrementally applied through 400 load steps in the folding simulation as described in Section 2.3. In this example, the input displacements are determined based on the known loading of the square twist pattern. The displacement arc is defined by a series of displacement steps, where $x_1 = (1 - cos(\theta)) * d$, $x_2 = (cos(\theta) - 1) * d$ and $z = -sin(\theta) * d$. The θ ranges from $[0, 0.7\pi]$ and has a moment arm d that is equivalent to $1/4$ of the total length scale of the design.

In Figure 4.4, four nodes are indicated as output nodes. The objective of the folding optimization is determined as a measure of the amount of actuation permitted in the prescribed output direction. For the square twist pattern, the objective indicates that the interior nodes should move

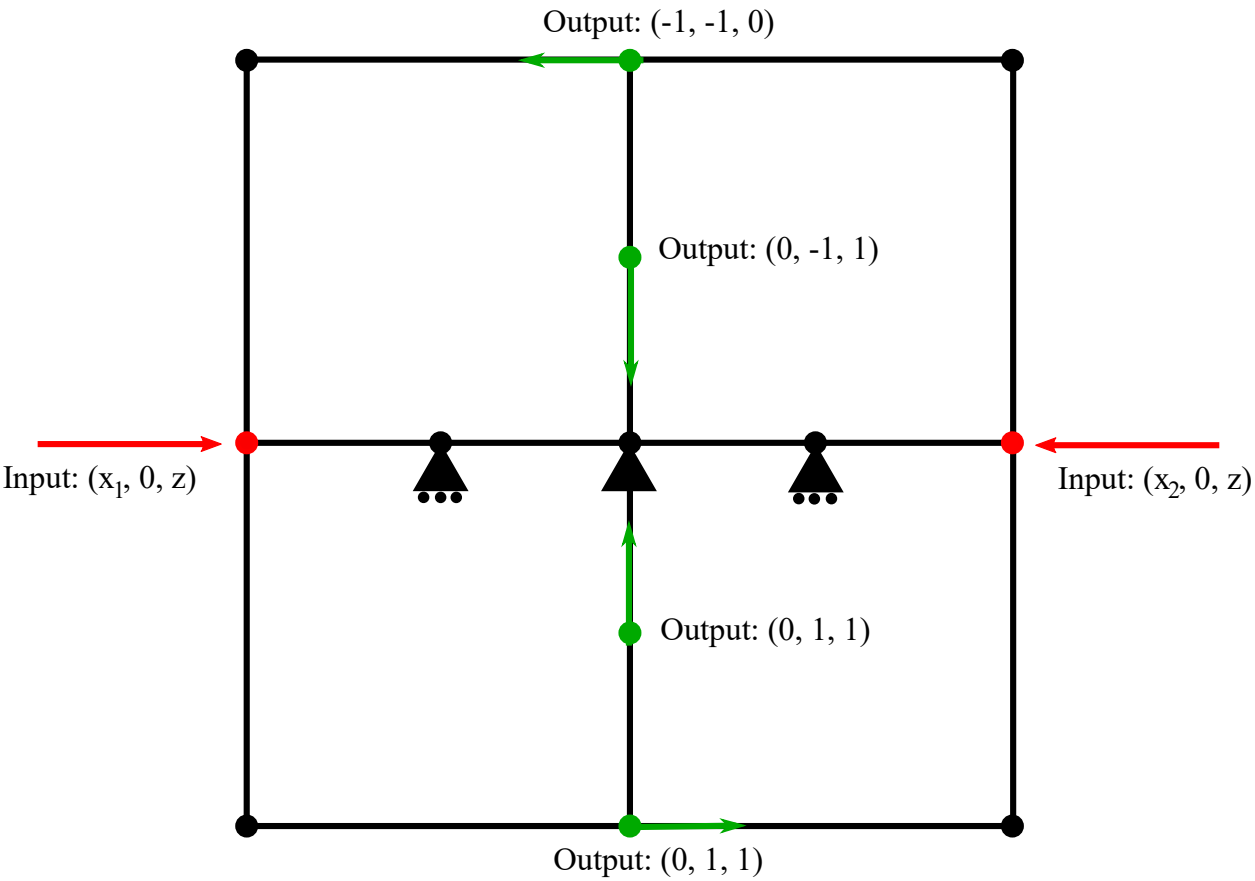


Figure 4.4: Boundary conditions, input, and output directions for the square twist problem

up out of plane and in towards the center. Further, the outer nodes should provide the twisting motion required. The design that enables the greatest actuation in the indicated direction of the output direction vectors (shown in Figure 4.4) is considered the optimal design.

In the folding simulation, several parameters are necessary to describe the material and penalties. The Young's Modulus, E in Equation 2.2 is $1e12$ Pa. The spring constant, G, is different for 'normal' and 'stiff' fold lines, with stiff fold lines being the product of triangulation and not the topology creation process. G is equal to $1e1$ and $1e4$ for normal and stiff folds, respectively. For the penalty function presented in Equation 2.5, C is equal to 1.0, and B is equal to 10.0.

The constraint on triangulation mesh quality has been relaxed from 0.2 to 0.1 to allow for more viable designs, as the SPIDRS topology creation methods has a tendency to create very small

polygonal subdomains which could not otherwise pass the meshing constraint. The results of this relaxation can cause designs to experience more deformation than preferred along the stiff foldlines during the folding simulation process.

This implementation utilized the Newton-Raphson method to solve each non-linear iteration of the imposed displacement. The displacement is applied over 400 loading steps, with a maximum of 100 Newton-Raphson iterations per loading step allowed. The tolerance for convergence in each load step is $1e{-}8$.

4.2.1.3 Genetic Optimization Framework

The design objective is to find the optimal fold pattern to facilitate actuation in the specified directions. In this framework, the objective is evaluated using the folding simulation tool described in Section 2.3, which acts through tuning the fold stiffness. Here, the fold stiffness is modeled as a torsional spring with stiffness, G. Lines generated from the SPIDRS topology in the initial graph are considered active and given small values of G.

Lines generated from the Delaunay triangulation process are considered inactive folds and assigned large values of G. The inactive folds also form a coarse discretization of facets to capture facet bending in the design. The fold stiffness for active folds is $G_{active} = 10^{\alpha_1}$, where $\alpha_1 = 1$, and the fold stiffness for inactive folds is $G_{inactive} = 10^{\alpha_2}$, where $\alpha_2 = 4$.

The output nodes are prescribed by the the constant vector $\boldsymbol{c}$ associated with the selected output degrees of freedom and the entries to this vector correspond with the vectors shown in Figure 4.4. The formal objective is obtained by taking the product of the transverse of the vector $\boldsymbol{c}$ with the global nodal displacements of the objective nodes, $\boldsymbol{u}$. The physical objective is to maximize this product, so the formal objective is multiplied by -1 to reframe the optimization statement as a minimization. There are 78 genes that are varied to find the minimized objective and are related to the 2 character axiom, ω_0, and 4 production rules, P (c.f. Section 2.2). Notably, the change material operation (c.f. Section 2.2.0.5) is not used in this optimization and therefore the 18 genes associated with it are left out. The initial optimization is not subject to any constraints and is run for 10 generations with a population of 240 individual designs.

Design Problem Statement	
Minimize:	$-\boldsymbol{c}^T\boldsymbol{u}$
by varying:	2 axiom characters,
	4 rule assignments (18 genes each)
NSGA-II Parameters	
240 members for 10 generations	
$P_{cross} = 0.9, \eta_{cross} = 50$	
$P_{mutation} = 1/92, \eta_{mutation} = 50$	

Table 4.1: Genetic optimization problem statement and NSGA-II parameters

4.2.2 Specific Problem Extensions

4.2.2.1 Enforced Bisection

Many well known flat foldable origami patterns have observable symmetry and have foldlines which only bisect faces, including the square twist pattern. These patterns have emerged naturally due to the ability of foldline which bisect faces to meet both Kawasaki and Makawa's theorems for flat foldability [69]. This knowledge has been utilized to create ground structure based approaches to generating origami fold patterns that incorporated bisection into the original ground structure [81], and develop topologies creation approaches that follow these rules inherently in their encoding [69]. However, this type of approach inherently limits the design space of the optimization to designs in the preexisting ground structure as discussed in Section 1.2.3. In alternative methods of origami creation, the a priori knowledge of bisection is generally not utilized due to the black box nature of many topology creation algorithms. Due to the basis of SPIDRS being in 5 simple production rules, it is easy to adapt the relevant production rules to specialize the algorithm to the problem in question by enforcing bisection during new structure creation.

In order to dictate bisection in the design process, the move real (c.f. Section 2.2.0.2) and the create real (c.f. Section 2.2.0.4) operations are altered. To do this, in both commands, α_1 is used to dictate the first part of the motion, where the operator proceeds $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes around the current face. In the second part of the command, the distance the node proceeds is altered from

$N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor$ to 0.5. This change allows SPIDRS to traverse the design space as before, while ensuring that any new nodes or edges are only generated at bisecting locations between nodes.

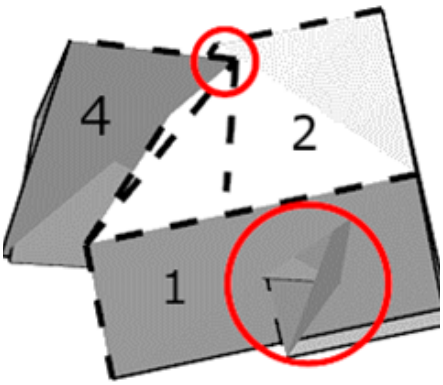
To test the hypothesis that patterns with imposed bisection will outperform patterns without bisection, two optimizations are run with the two different sets of production rules, P. The formal optimization statement is identical for both optimizations as shown in Table 4.1. The difference between the two optimizations is that the set of production rules are varied.

4.2.2.2 Self-Intersection Detection

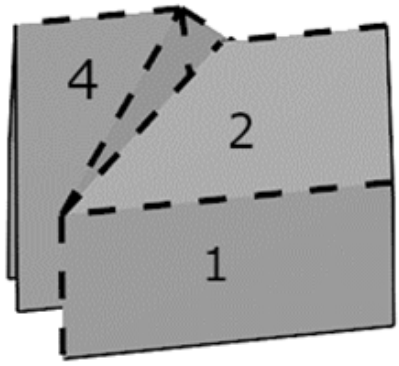
Previous implementations of origami folding simulations using the method discussed in Section 2.3 have not tracked global self intersections [81]. In fact, the global self intersection problem is generally not addressed during fold simulation and is addressed only as a post processing step due to the computational expense of implementing full contact modeling. However, ignoring the possibility of self intersection prevents true optimization due to the ability of patterns to be over-folded and outperform physically realizable patterns.

As an example, Figure 4.5 shows the folded configuration of a square twist pattern. In Figure 4.5a, the pattern is folded to completion in the folding simulation and experiences self intersections in faces 1 and 2 (circled for emphasis). In Figure 4.5b, the same pattern is displayed at the load step before self intersection occurs. What is notable about these two configurations of the same designs, is that the design that experiences self intersection outperforms the design that does not experience self intersection in the kinematic objective posed for this problem. These non-realizable foldings with better objective valuea are prone to bias the optimizer towards non-physical solutions. When the optimal designs are post processed, they will not outperform the optimal design from a genetic optimization that was able to account for these intersections during the folding simulation.

In order to test this hypothesis, a fast and reliable form of self-intersection detection needs to be implemented in the folding simulation code. While full contact modeling is out of the question due to computational expense and scope of the project, a quick geometric intersection check is capable of meeting these requirements. Möller's Algorithm for triangle triangle intersection in



(a) Design folded through all load steps



(b) Design folded until intersection detected

Figure 4.5: Actuated square twist origami pattern with and without fold intersection checking implemented

Design Problem Statement	
Minimize:	$-\boldsymbol{c}^T\boldsymbol{u}$
by varying:	2 axiom characters,
	4 rule assignments (18 genes each)
subject to:	$\boldsymbol{M} = 0$
NSGA-II Parameters	
240 members for 10 generations	
$P_{cross} = 0.9, \eta_{cross} = 50$	
$P_{mutation} = 1/92, \eta_{mutation} = 50$	

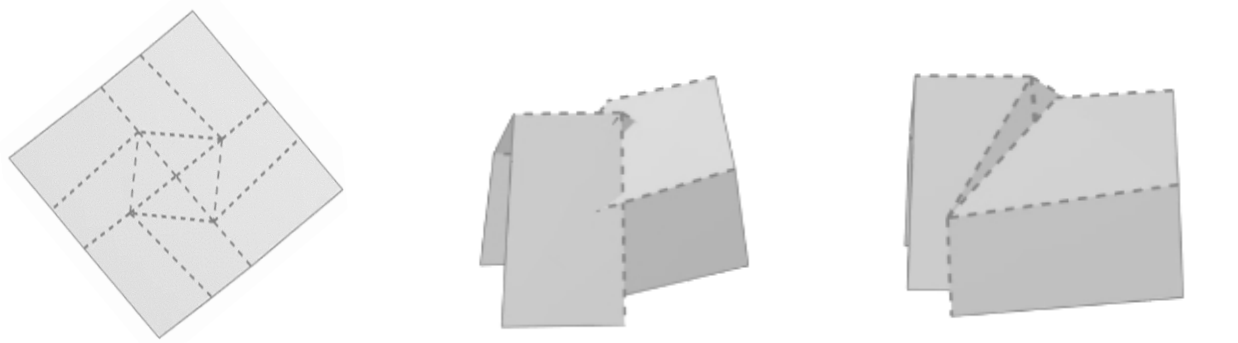
Table 4.2: Genetic optimization problem statement with self-intersection constraint

three dimensions [**?**] was implemented to perform this geometry check at the end of each loading step in the folding simulation (c.f. Appendix B). Due to the element shape of the truss-based linear code already having a triangular shape requirement, all were guaranteed to be discretized into a triangular mesh making this a practical implementation. The resulting new formal optimization statement with the self-intersection constraint is shown in Table 4.2, where $\boldsymbol{M}$ is a value associated with triangle-triangle intersection check the indicates if intersection has occured (1) or not (0).

4.2.3 Results

4.2.3.1 Self-Intersection Detection

In order to test the hypothesis that co-current self intersection detection will perform better than without intersection checking, two square twist optimizations are conducted with the same initial design space and folding simulation parameters. In the first optimization, no self intersection detection was performed and the minimum objective of each generation is shown by the blue line in Figure 4.7. In the second optimization, self intersection detection was performed during the folding simulations and the objective was determine to be the objective of the design at the loading step before self intersection occurs. The minimum objective of each generation is shown by the red line in Figure 4.7. From these results, one might conclude that the optimization without a self-intersection constraint outperforms the optimization with the self-intersection constraint, but the objectives that were determined without the self-intersection constraint are likely non-physically realizable.



(a) Optimal fold pattern

(b) Fully folded design

(c) Partially folded design, prior to self-intersection

Figure 4.6: Results of genetic optimization with and without self-intersection constraints before and after post processing to physically viable designs.

To demonstrate this concept, the a fold pattern without the self-intersection constraint is shown in Figure 4.6a. The fully folded design is shown in Figure 4.5b and the partially folded design is shown in Figure 4.6c. The fully folded design has an objective value of -2.149, while the partially folded design has an objective value of -2.121. However, the fully folded design is not physically realizable due to the self-intersections that occur.

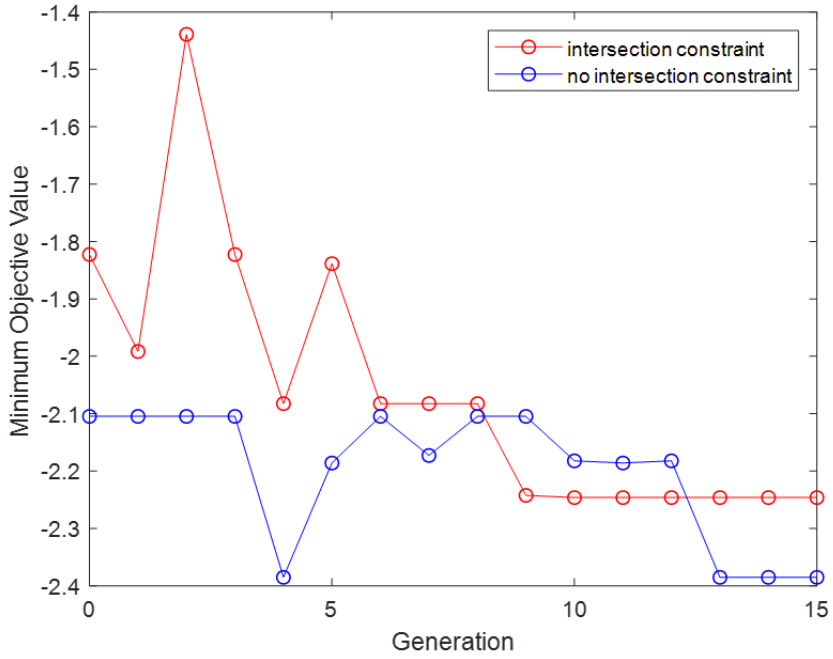


Figure 4.7: Results of square twist genetic optimization with no post processing

In Figure 4.8, the results of the optimization without the self-intersection constraint are post processed to determine where self-intersection occurs and the objectives at this point are re-plotted. From this graph, it's clear to see that imposing the self-intersection constraint improves the results of the optimization process.

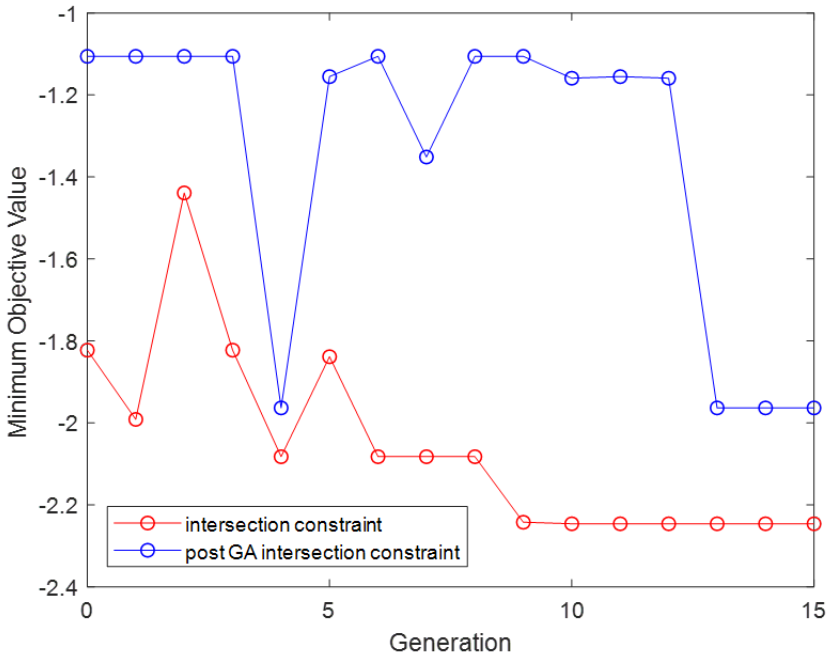


Figure 4.8: Results of square twist genetic optimization with all designs only folded to physically viable states (i.e. no self intersection in optimal designs)

4.2.3.2 *Enforced Bisection*

The optimization proposed in Table 4.2 was run with the alter bisection production rules and compared to the original SPIDRS production rules. The results of these optimizations are shown in Figure 4.9. The altered production (bisection constraint) rules are able to converge to a perfect square twist pattern very quickly, while the original (no bisection constraint) takes longer and does not reach a perfect objective. This is expected, as the optimization without bisection constraints is very unlikely to place an exact bisecting fold line. Instead, the optimization finds a fold line very close to bisection and utilizes this design to evolve, eventually getting stuck in a local minimum.

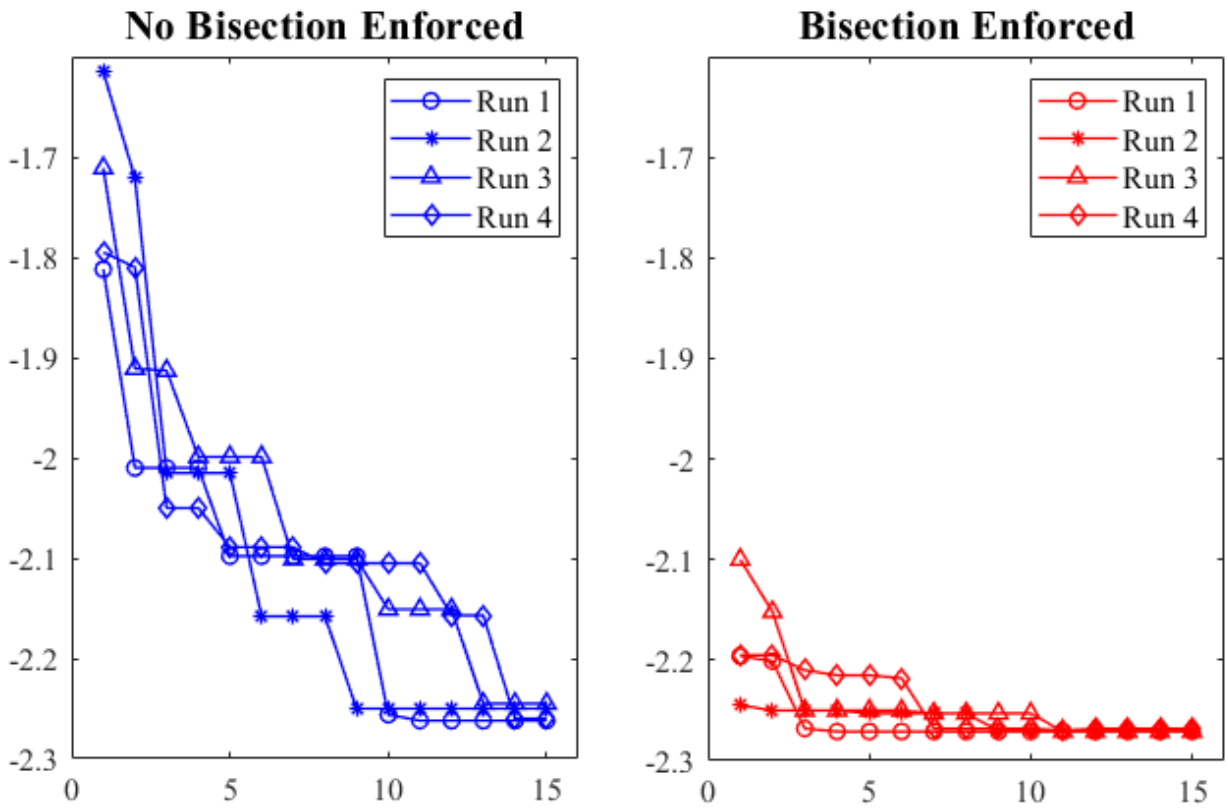


Figure 4.9: Results of twist genetic optimization with enforced bisection and no enforced bisection

This result points to the conclusion that bisection enforcement is useful for increasing convergence speed for designs in which bisection is favorable. However, we can also see that without the bisection constraint, the optimization framework laid out by SPIDRS is very good at finding the general pattern without interference. While the exact square twist was not found, the design without the bisection constraint would undoubtedly have been equally useful in preliminary design.

4.3 Continuous Kinematic Response Objective

4.3.1 Problem Setup

Ideally, the SPIDRS design process should be applicable to any arbitrary objective. To test this, an arbitrary kinematic objective was devised. In this objective, a single node is targeted to follow a 2D line in space during the folding process. This type of objective could be useful in any scenario with limited initial volume that requires a prescribed movement, like the deployment of a structure into space.

4.3.1.1 Initial Graph and Symmetry

The specific objective targeted in this section is a node that starts at $(0.0, 0.0, 0.0)$ and moves incrementally along the vector $< 1, 1, 0 >$ to the point $(0.5, 0.5, 0.0)$. This motion is depicted by the green arrow labeled 'Output Path' in Figure 4.10. The initial graph considered is the simplest possible graph, as the goal is to determine a fold pattern to meet the kinematic objective without any a priori knowledge of the solution. Also, due to the lack of intuition of the solution, no symmetry conditions are applied. Without a strong intuition of symmetry in the final solution, a symmetry condition will likely serve very litter purpose and would bias the pattern by placing a fold line at the bisection point in all designs.

The graph for this objective is a square in the XY plane with the lower left corner situated at the origin. This space has a domain and range of $[0, 1]$ so that it can be easily scaled to any other length as needed. The initial parameters utilized by SPIDRS are described as the nodes, $N = \{1 : (0.0, 0.0), 2 : (1.0, 0.0), 3 : (1.0, 1.0), 4 : (0.0, 1.0)\}$, edges, $E = [[1, 2], [2, 3], [3, 4], [4, 1]]$, and faces, $F = [[1, 2, 3, 4]]$. The material assignments from the SPIDRS algorithm are not utilized and all generated edges are considered active fold lines. As a result, the design variables associated with the change material production rule (c.f. 2.2.0.5) are removed and only 78 total design variables are necessary.

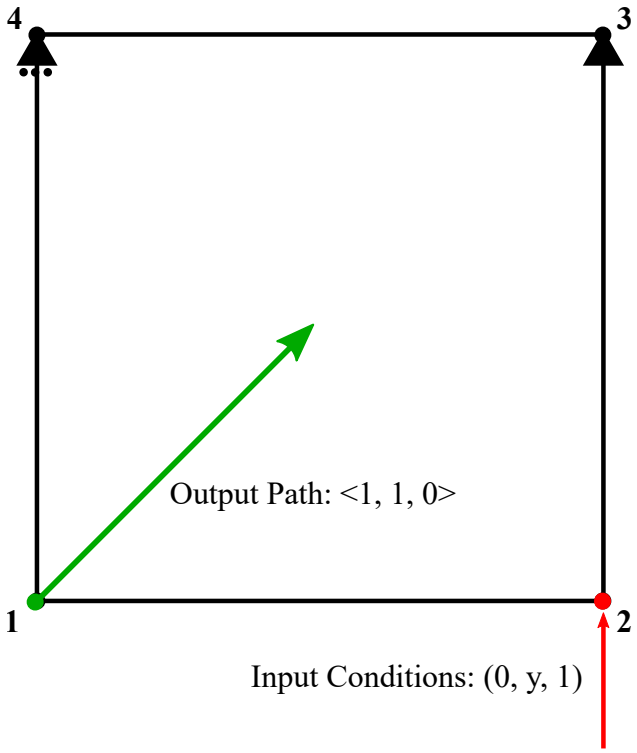


Figure 4.10: Boundary conditions, input, and output directions for the continuous kinematic response objective

4.3.1.2 Boundary Conditions and Folding Simulation Parameters

The boundary conditions on the design are also very simple in order to minimize the need for human intuition to solve the given problem. The boundary conditions are shown graphically in Figure 4.10. The objective node, node 1, is completely free of boundary conditions and therefore able to move in all 3D dimensions during the folding process. Node 2, is prescribed a displacement in the Y direction. The displacement is only applied in the Y direction and starts at $Y = 0.0$ and moves incrementally to $Y = 0.5$. This displacement is applied over 500 load steps in the folding simulation. The Z component of the loaded node is also constrained at $Z = 0$, while the X component of this node is free in space. This constraint was used to prevent rigid body rotation about the pinned edge ($[3, 4]$) in the folding simulation code. Node 3, is completely fixed and serves as the grounding point of the structure. Lastly, node 4 is fixed in the Y and Z directions,

but given freedom in X. This condition was prescribed to allow for contraction of the design if necessary along the upper pinned edge.

In the folding simulation, several parameters are necessary to describe the material and penalties. The Young's Modulus, E in Equation 2.2 is $1e12$ Pa. The spring constant, G, is different for 'normal' and 'stiff' fold lines, with stiff fold lines being the product of triangulation and not the topology creation process. G is equal to $1e1$ and $1e4$ for normal and stiff folds, respectively. For the penalty function presented in Equation 2.5, C is equal to 1.0, and B is equal to 10.0.

The constraint on triangulation mesh quality has been relaxed from 0.2 to 0.1 to allow for more viable designs, as the SPIDRS topology creation methods has a tendency to create very small polygonal subdomains which could not otherwise pass the meshing constraint. The results of this relaxation can cause designs to experience more deformation than preferred along the stiff fold lines during the folding simulation process.

This implementation utilized the Newton-Raphson method to solve each non-linear iteration of the imposed displacement. The displacement is applied over 500 loading steps, with a maximum of 50 Newton-Raphson iterations per loading step allowed. The tolerance for convergence in each load step is $1e{-}8$.

4.3.1.3 Genetic Optimization Framework

The design objective is to move node 1 from $(0.0, 0.0, 0.0)$ to $(0.5, 0.5, 0.0)$ along the $< 1.0, 1.0, 0.0 >$ vector incrementally. To evaluate this objective, the folding simulation tool is run with the parameters described above, which acts through tuning the fold stiffness. Here, the fold stiffness is modeled as a torsional spring with stiffness, G. Lines generated from the SPIDRS topology in the initial graph are considered active and given small values of G.

Lines generated from the Delaunay triangulation process are considered inactive folds and assigned large values of G. The inactive folds also form a coarse discretization of facets to capture facet bending in the design. The fold stiffness for active folds is $G_{active} = 10^{\alpha_1}$, where $\alpha_1 = 1$, and the fold stiffness for inactive folds is $G_{inactive} = 10^{\alpha_2}$, where $\alpha_2 = 3$.

The location of the output node, node 1, is saved during each loading step in the folding sim-

ulation. This collection of locations is considered the path of the node during folding and the Hausdorff distance[2], d_H, is taken between this array and an ideal path that increments from $0, 0$ to $0.5, 0.5$ in 400 steps. Four hundred steps are used, because this is the number of load steps in the folding simulation.

Design Problem Statement	
Minimize:	d_H
by varying:	2 axiom characters,
	4 rule assignments (18 genes each)
subject to:	$q > 0.1$
	$max(\|\alpha(n_{load})\|) > 0$
NSGA-II Parameters	
1000 members for 100 generations	
$P_{cross} = 0.9, \eta_{cross} = 50$	
$P_{mutation} = 1/92, \eta_{mutation} = 50$	

Table 4.3: Genetic optimization problem statement with no constraints

The formal optimization statement for this problem is shown in Table 4.3. Here, the Hausdorff distance between the objective node and a target path is minimize by varying the genes used by the Parameterized L-System. Notably, material properties are not utilized in this optimization, which reduces the required design parameters from 92 to 78. This optimization is performed on a population of 1000 individuals for 100 generations. The population size is selected to be relatively larger to ensure genetic diversity, because this process may result in many infeasible designs due to triangulation constraints and/or convergence issues in the folding simulation.

There are two constraints on this optimization. The first constraint is a mesh quality constraint to ensure that the resulting mesh from Delaunay triangulation does not produce poor quality elements which could compromise the effectiveness of the folding simulation code. The second constraint is that the maximum fold angle during the final loading step must be non-zero. This

[2] Hausdorff distance is a measure of how far two subsets of a metric space are from each other and is a defined by the longest distance one can travel between two points in two different sets [**?**].

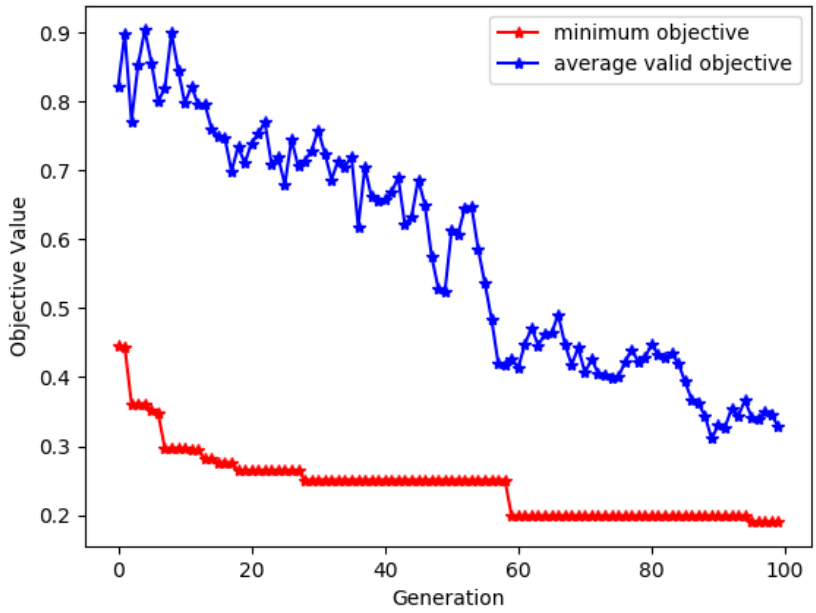
constraint is implemented to ensure that designs considered in the optimization are folding and not experiencing only fact bending. Designs without this constraint can perform well under the objective, but do not actually experience any folding, since the load is applied via a displacement condition and there is no objective considering folding energy.

4.3.2 Results

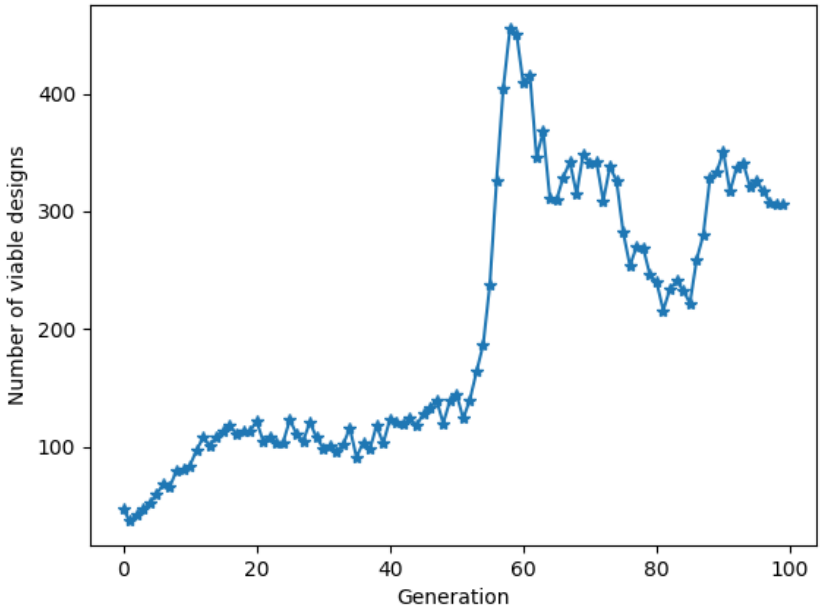
The resulting objective for the optimization is shown in Figure 4.11a. The minimum objective quickly decreases over the first 30 generations but then hits a local minima, then mutates and decreases again at 59 generations, where the patterns is again repeated at 94 generations. One reason for the convergence to local minima, may be the lack of genetic diversity in the population. Due to the optimization constraints, over 90% of the initial population is filtered out. The number of viable designs per generation is shown in Figure 4.11b. The percent of viable designs does slowly increase, but one can assume this is a result of the crossover occuring between already viable designs being selected by the genetic algorithm, meaning that this rise in viability does not introduce genetic diversity.

The resulting best design is shown in Figure 4.12a in its flat configuration. The folded configuration is shown in two different orientations in Figure 4.12b and 4.12c, which depict and isometric view and an XY planar view of the folded pattern respectively. In the XY view, we can note that the location of nodes 3 and 4 are identical to their relative location in flat configuration. The triangular section formed on the right side of the design is initially folded up and under by the imposed y-displacement puling the objective node in and to the right of its original location. The folding process is difficult to visualize, but the path of the objective node is traced in 3D space in Figure 4.13. There is also a distinct second stage of folding where the fold line parallel to the bottom edge of the design is folded 180 degrees back to the flat configuration. This fold coincides with the rounded, out of plane, hump visible in Figure 4.13a and 4.13c.

There is also a small amount of facet bending visible in this best design that occurs in the triangle on the upper left side of the flat configuration. In Figure 4.12b, on the left side at node 4, a difference in shading is visible and denotes deformation that occurs in the facet without the



(a) Optimization objective values

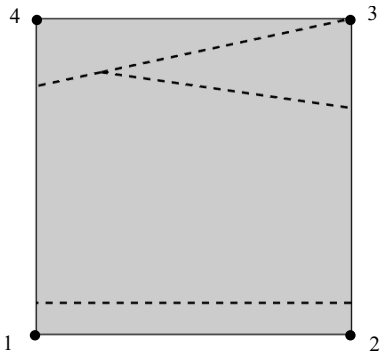


(b) Number of viables designs per generation

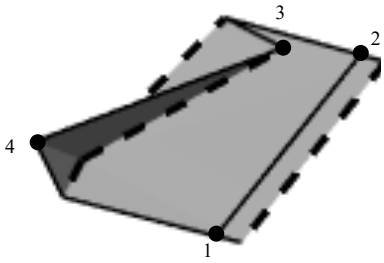
Figure 4.11: Optimization Results

presence of a fold line. This deformation occurs to allow for the complete flat folding of the triangle in the upper right side of the design. This small deformation is the only facet bending that occurs in this optimal design.

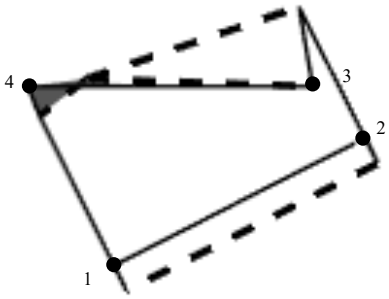
Overall, the path depicted in Figure 4.13 does not follow the linear path describe by the objective. However, this path does perform well in the XY plane near the end of the folding process. Overall, this solution would likely not be considered to follow this line, but the success of the optimization process implementation is observable. Possibly with a much larger initial population or by utilizing a process that has higher design viability, a better objective could be reached.



(a) Origami pattern

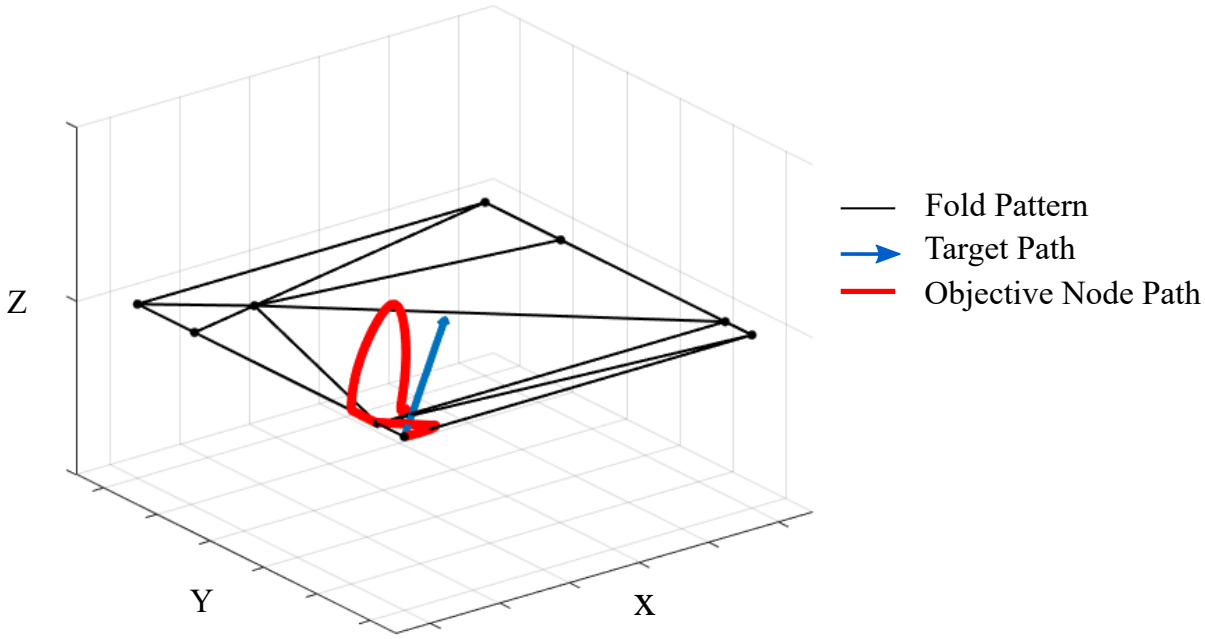


(b) Iso view of the folded configuration

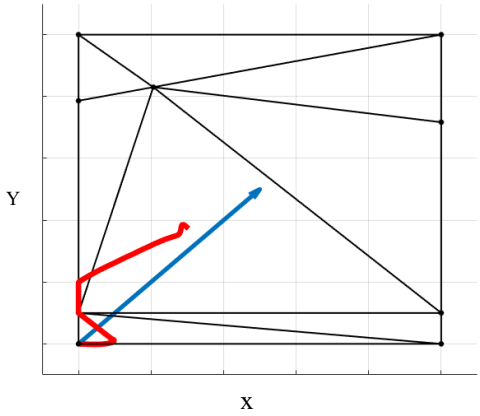


(c) XY view of the folded configuration

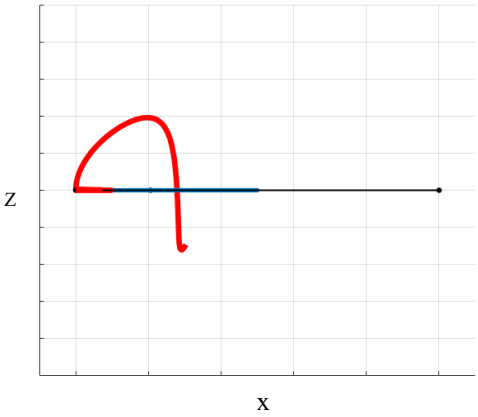
Figure 4.12: Best design for the continuous kinematic objective



(a) Iso view of the objective node path



(b) XY view of the objective node path



(c) XZ view of the objective node path

Figure 4.13: Objective path during folding for best design

5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this work, we have applied the SPIDRS design methodology to two different classes of problems in order to evaluate the effectiveness of the SPIDRS interpreter. The first problem required the design of the internal structural layout for a camber morphing airfoil. The second problem required the generation of origami fold patterns to meet kinematic objectives. In both of these applications, a similar approach is taken in which the SPIDRS algorithm is coupled with a analysis tool to evaluate a physical objective and an optimization is performed using this information. This process is shown schematically for each problem in Figures 3.2 and 4.1.

The results of the camber morphing airfoil problem produced a populated Pareto front that contained designs capable of minimizing the multiple objectives used in the optimization, shape and mass. The inclusion of the SPIDRS algorithm allows this method to produce a variety of topological designs that represent compromises between the two objectives with various actuator placements, giving engineers a strong preliminary point for refining the design of non-intuitive morphing structure. This method is also able to prevent designers from being boxed into a certain design space by avoiding the requirement for a priori knowledge of the actuator placement, which is required by more traditional methods. In the future, this method could be easily adapted to design camber morphing airfoils for a variety of physical objectives. For example, the problem can easily be altered by changing the method of actuation or changing the objective of the optimization to increase the C_l/C_d ratio.

The application of SPIDRS to the abductive origami design process was very successful when compared to well known solutions, but was unable to generate designs to meet the continuous kinematic folding response objective. These results are indicative of the incompatibility between the SPIDRS design methodology and the abductive origami problem posed. In order to effective simulate the folding of origami computationally, the loading path and boundary conditions must

be well defined. These conditions are easy to obtain through observation of the known solution for square twist and allow the folding simulation to serve as a robust tool by which to evaluate the objective and inform the optimizer. On the other hand, with the continuous kinematic response objective, there is no known solution available by which loading paths and boundary conditions can be observed. Arbitrary conditions are utilized to determine if the abductive design process is robust enough to find a solution to the continuous kinematic response objective, but the process is shown to be unreliable. With arbitrary loading conditions, that are not obtained from a preexisting solution, the kinematic folding simulation tool struggles to converge for simple topologies, resulting in very low design viability and genetic diversity in the GA.

The application of boundary conditions is especially difficult for origami folding, due to the relative nature of origami folding objectives. The objective of origami design is often to achieve a certain final shape, but the success of that shape in the real world is not dependent on it's location in a global reference frame but rather in a local reference frame that is able to track the facets relatively only to one another. Another difficulty of folding origami is that in real life, the boundary conditions and load paths can often be sequenced. Deciding on the sequencing for an objective with an unknown solution is difficult and using SPIDRS design for problems with predefined sequencing from the known solution is trivial.

Overall, we can observe that the SPIDRS design methodology was easily adapted to both classes of problems, but requires robust analysis tools that do not require intuition of the solution to implement function analysis. The creation of topology was also generalized to graphs with non-linear functions on the boundaries and adapted to provide bisection. In addition to these contributions relating to the SPIDRS algorithm, a useful implementation triangle intersection detection was utilized for the first time in origami folding simulation and could later prove valuable for origami design.

5.2 Future Work

There are several paths in which future work should be conducted on this problem. The first direction includes model validation and improvement on the currently implemented processes described in this work and are listed below:

- Perform model validation with a bench top model of a morphing airfoil.
- Perform a morphing airfoil optimization with an updated aerodynamic objective to validate the process for several types of objectives.
- Update the morphing airfoil optimization scheme with an uncoupled FSI aerodynamic model to reduce computational cost and confirm accuracy.
- Adapt the 3D extension for SPIDRS into the morphing airfoil problem to allow for the optimization of entire lifting surfaces.
- Utilize the SPIDRS approach for well defined origami problems, like the Chomper pattern or the Waterbomb.

The second path for future work involves utilizing the latest SPIDRS advancements to implement sizing optimizations into any of the problems or the re-frame the problems as 3D optimizations. The third and final path of exploration should include the application of SPIDRS topology design into new ares of engineering or included new functional analysis methods, like electromagnetic or thermal analysis methods.

REFERENCES

[1] A. Gillman, K. Fuchi, and P. R. Buskohl, "Truss-based nonlinear mechanical analysis for origami structures exhibiting bifurcation and limit point instabilities," *International Journal of Solids and Structures*, vol. 147, pp. 80–93, Aug. 2018.

[2] "Sikorsky Archives | S-76."

[3] P. B. Leal, M. A. Savi, and D. J. Hartl, "Aero-structural optimization of shape memory alloy-based wing morphing via a class/shape transformation approach," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 232, no. 15, pp. 2745–2759, 2018.

[4] P. Prusinkiewicz, "Graphical applications of L-systems," vol. 86, pp. 247–253, 1984.

[5] A. Roth-Nebelsick, D. Uhl, V. Mosbrugger, and H. Kerp, "Evolution and Function of Leaf Venation Architecture: A Review," *Annals of Botany*, vol. 87, pp. 553–566, May 2001.

[6] J. Stark, J. Bonacum, J. Remsen, and R. DeSalle, "THE EVOLUTION AND DEVELOPMENT OF DIPTERAN WING VEINS: A Systematic Approach," *Annual Review of Entomology*, vol. 44, no. 1, pp. 97–129, 1999.

[7] M. LaBarbera, "Principles of design of fluid transport systems in zoology," *Science*, vol. 249, pp. 992–1000, Aug. 1990.

[8] R. Dudley, *The Biomechanics of Insect Flight: Form, Function, Evolution*. Princeton University Press, Sept. 2002. Google-Books-ID: hTIMhD9BF1kC.

[9] A. Bejan, *Shape and Structure, from Engineering to Nature*. Cambridge University Press, Oct. 2000. Google-Books-ID: L4US10yjFeEC.

[10] S. B. Carroll, "Chance and necessity: the evolution of morphological complexity and diversity," *Nature*, vol. 409, pp. 1102–1109, Feb. 2001.

[11] A. Lindenmayer, "Mathematical models for cellular interactions in development II. Simple and branching filaments with two-sided inputs," *Journal of Theoretical Biology*, vol. 18, pp. 300–315, Mar. 1968.

[12] A. Lindenmayer, "Mathematical models for cellular interactions in development I. Filaments with one-sided inputs," *Journal of Theoretical Biology*, vol. 18, pp. 280–299, Mar. 1968.

[13] P. Prusinkiewicz, "Applications of L-systems to computer imagery," in *Graph-Grammars and Their Application to Computer Science* (G. Goos, J. Hartmanis, D. Barstow, W. Brauer, P. Brinch Hansen, D. Gries, D. Luckham, C. Moler, A. Pnueli, G. Seegmüller, J. Stoer, N. Wirth, H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, eds.), vol. 291, pp. 534–548, Berlin, Heidelberg: Springer Berlin Heidelberg, 1987. Series Title: Lecture Notes in Computer Science.

[14] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants*. Springer Science & Business Media, Dec. 2012. Google-Books-ID: 4F7lBwAAQBAJ.

[15] M. H. Kobayashi, "On a biologically inspired topology optimization method," *Communications in Nonlinear Science and Numerical Simulation*, vol. 15, pp. 787–802, Mar. 2010.

[16] D. J. Hartl, G. W. Reich, and P. S. Beran, "Additive Topological Optimization of Muscular-Skeletal Structures via Genetic L-System Programming," in *24th AIAA/AHS Adaptive Structures Conference*, (San Diego, California, USA), American Institute of Aeronautics and Astronautics, Jan. 2016.

[17] D. J. Hartl, B. Bielefeldt, G. W. Reich, and P. S. Beran, "Multi-fidelity Analysis and Experimental Characterization of Muscular-Skeletal Structures Optimized via Genetic Programming," in *25th AIAA/AHS Adaptive Structures Conference*, (Grapevine, Texas), American Institute of Aeronautics and Astronautics, Jan. 2017.

[18] B. R. Bielefeldt, G. W. Reich, P. S. Beran, and D. J. Hartl, "Development and validation of a genetic L-System programming framework for topology optimization of multifunctional structures," *Computers & Structures*, vol. 218, pp. 152–169, July 2019.

[19] J. D. Deaton and R. V. Grandhi, "A survey of structural and multidisciplinary continuum topology optimization: post 2000," *Structural and Multidisciplinary Optimization*, vol. 49, pp. 1–38, Jan. 2014.

[20] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural optimization*, vol. 1, pp. 193–202, Dec. 1989.

[21] H. A. Eschenauer and N. Olhoff, "Topology optimization of continuum structures: A review," *Applied Mechanics Reviews*, vol. 54, no. 4, p. 331, 2001.

[22] M. Wang, X. Wang, and D. Guo, "A level set method for structural topology optimization," *Computer Methods in Applied Mechanics and Engineering*, vol. 192, pp. 227–246, Jan. 2003.

[23] S.-H. Ha and S. Cho, "Level Set Based Topological Shape Optimization of Geometrically Nonlinear Structures Using Unstructured Mesh," *Comput. Struct.*, vol. 86, pp. 1447–1455, July 2008.

[24] N. P. van Dijk, K. Maute, M. Langelaar, and F. van Keulen, "Level-set methods for structural topology optimization: a review," *Structural and Multidisciplinary Optimization*, vol. 48, pp. 437–472, Sept. 2013.

[25] Y. M. Xie and G. P. Steven, "Basic Evolutionary Structural Optimization," in *Evolutionary Structural Optimization* (Y. M. Xie and G. P. Steven, eds.), pp. 12–29, London: Springer, 1997.

[26] M. P. Bendsøe, A. Ben-Tal, and J. Zowe, "Optimization methods for truss geometry and topology design," *Structural Optimization*, vol. 7, pp. 141–159, Apr. 1994.

[27] K. Deb, "An introduction to genetic algorithms," p. 23.

[28] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI* (M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 849–858, Springer, 2000.

[29] B. Stanford, P. Beran, and M. Kobayashi, "Aeroelastic Optimization of Flapping Wing Venation: A Cellular Division Approach," *AIAA Journal*, vol. 50, no. 4, pp. 938–951, 2012.

[30] B. Stanford, P. Beran, and M. Kobayashi, "Simultaneous Topology Optimization of Membrane Wings and Their Compliant Flapping Mechanisms," *AIAA Journal*, vol. 51, no. 6, pp. 1431–1441, 2013.

[31] M. Kobayashi, R. Kolonay, G. Reich, A. LeBon, and H.-T. Pedro, "On a Cellular Division Model for Multi-Disciplinary Optimization," in *51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference
 18th AIAA/ASME/AHS Adaptive Structures Conference
 12th*, (Orlando, Florida), American Institute of Aeronautics and Astronautics, Apr. 2010.

[32] R. Kolonay and M. Kobayashi, "Topology, Shape, and Sizing Optimization of Aircraft Lifting Surfaces Using a Cellular Division Method," in *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*, (Fort Worth, Texas), American Institute of Aeronautics and Astronautics, Sept. 2010.

[33] R. M. Kolonay and M. H. Kobayashi, "Optimization of Aircraft Lifting Surfaces Using a Cellular Division Method," *Journal of Aircraft*, vol. 52, no. 6, pp. 2051–2063, 2015.

[34] H. Abelson and A. A. DiSessa, *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. MIT Press, 1986. Google-Books-ID: 3geYp44hJVcC.

[35] B. R. Bielefeldt, E. Akleman, G. W. Reich, P. S. Beran, and D. J. Hartl, "L-System-Generated Mechanism Topology Optimization Using Graph-Based Interpretation," *Journal of Mechanisms and Robotics*, vol. 11, Apr. 2019.

[36] B. R. Bielefeldt, D. J. Hartl, J. D. Hodson, G. W. Reich, P. S. Beran, A. M. Pankonien, and J. D. Deaton, "Graph-Based Interpretation of L-System Encodings Toward Aeroelastic Topology Optimization of a Morphing Airfoil in Supersonic Flow," American Society of Mechanical Engineers Digital Collection, Dec. 2019.

[37] M. Mikkelsen, M. Mathew, P. Walgren, B. Bielefeldt, P. B. C. Leal, D. Hartl, and A. Arrieta, "Morphing Airfoil Design via L-System Generated Topology Optimization," American Society of Mechanical Engineers Digital Collection, Dec. 2019.

[38] R. M. Ward, B. R. Bielefeldt, and D. J. Hartl, "Design of tailorable stiffness structures using L-system topology optimization," in *Behavior and Mechanics of Multifunctional Materials IX*, vol. 11377, p. 113770R, International Society for Optics and Photonics, Apr. 2020.

[39] S. Barbarino, O. Bilgen, R. M. Ajaj, M. I. Friswell, and D. J. Inman, "A Review of Morphing Aircraft," *Journal of Intelligent Material Systems and Structures*, vol. 22, pp. 823–877, June 2011.

[40] J. Spillman, "The use of variable camber to reduce drag, weight and costs of transport aircraft," *The Aeronautical Journal (1968)*, vol. 96, pp. 1–9, 192.

[41] E. Stanewsky, "Adaptive wing and flow control technology," *Progress in Aerospace Sciences*, vol. 37, pp. 583–667, Oct. 2001.

[42] L. F. Campanile and D. Sachau, "The Belt-Rib Concept: A Structronic Approach to Variable Camber," *Journal of Intelligent Material Systems and Structures*, vol. 11, pp. 215–224, Mar. 2000. Publisher: SAGE Publications Ltd STM.

[43] P. Poonsong, *Design and Analysis of a Multi-Section Variable Camber Wing*. Thesis, Mar. 2004. Accepted: 2004-05-31T20:15:18Z.

[44] I. K. Kuder, A. F. Arrieta, W. E. Raither, and P. Ermanni, "Variable stiffness material and structural concepts for morphing applications," *Progress in Aerospace Sciences*, vol. 63, pp. 33–55, Nov. 2013.

[45] B. K. S. Woods, O. Bilgen, and M. I. Friswell, "Wind tunnel testing of the fish bone active camber morphing concept," *Journal of Intelligent Material Systems and Structures*, vol. 25, 2014.

[46] B. K. S. Woods, I. Dayyani, and M. I. Friswell, "Fluid/Structure-Interaction Analysis of the Fish-Bone-Active-Camber Morphing Concept," *Journal of Aircraft*, vol. 52, pp. 307–319, Jan. 2015.

[47] B. K. S. Woods and M. I. Friswell, "Multi-objective geometry optimization of the Fish Bone Active Camber morphing airfoil," *Journal of Intelligent Material Systems and Structures*, vol. 27, 2016.

[48] L. Campanile, "Lightweight Shape-Adaptable Airfoils: A New Challenge for an Old Dream," *Adaptive Structures*, May 2007.

[49] I. K. Kuder, U. Fasel, P. Ermanni, and A. F. Arrieta, "Concurrent design of a morphing aerofoil with variable stiffness bi-stable laminates," *Smart Materials and Structures*, vol. 25, p. 115001, Sept. 2016.

[50] S. Gano, V. Perez, J. Renaud, S. Batill, and B. Sanders, "Multilevel Variable Fidelity Optimization of a Morphing Unmanned Aerial Vehicle," in *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, (Palm Springs, California), American Institute of Aeronautics and Astronautics, Apr. 2004.

[51] D. Bornengo, F. Scarpa, and C. Remillat, "Evaluation of hexagonal chiral structure for morphing airfoil concept," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 219, pp. 185–192, Mar. 2005. Publisher: IMECHE.

[52] L. Shili, G. Wenjie, and L. Shujun, "Optimal Design of Compliant Trailing Edge for Shape Changing," *Chinese Journal of Aeronautics*, vol. 21, pp. 187–192, Apr. 2008.

[53] R. Decamp and R. Hardy, "Mission adaptive wing advanced research concepts," in *11th Atmospheric Flight Mechanics Conference*, (Seattle,WA,U.S.A.), American Institute of Aeronautics and Astronautics, Aug. 1984.

[54] K. Bonnema and S. Smith, "AFTI/F-111 Mission Adaptive Wing flight research program," in *4th Flight Test Conference*, (San Diego,CA,U.S.A.), American Institute of Aeronautics and Astronautics, May 1988.

[55] S. B. Smith and D. W. Nelson, "Determination of the aerodynamic characteristics of the mission adaptive wing," *Journal of Aircraft*, vol. 27, pp. 950–958, Nov. 1990.

[56] F. Austin, M. Siclari, W. Van Nostrand, G. N. Weisensel, V. Kottamasu, and G. Volpe, "Comparison of smart wing concepts for transonic cruise drag reduction," in *Smart Structures and Materials 1997: Industrial and Commercial Operations of Smart Structures Technologies*, vol. 3044, (San Diego, California), 1997.

[57] P. B. Leal, M. A. Savi, and D. J. Hartl, "Aero-structural optimization of SMA-based wing morphing via a Class/Shape Transformation approach," 2010.

[58] P. B. Leal, R. Petterson, and D. J. Hartl, "Design optimization toward a shape memory alloy-based bio-inspired morphing wing," in *25th AIAA/AHS Adaptive Structures Conference*, p. 0054, 2017.

[59] E. A. Peraza-Hernandez, D. J. Hartl, R. J. M. Jr, and D. C. Lagoudas, "Origami-inspired active structures: a synthesis and review," *Smart Materials and Structures*, vol. 23, p. 094001, Aug. 2014.

[60] M. Schenk, A. D. Viquerat, K. A. Seffen, and S. D. Guest, "Review of Inflatable Booms for Deployable Space Structures: Packing and Rigidization," *Journal of Spacecraft and Rockets*, vol. 51, pp. 762–778, May 2014.

[61] S. A. Zirbel, R. J. Lang, M. W. Thomson, D. A. Sigel, P. E. Walkemeyer, B. P. Trease, S. P. Magleby, and L. L. Howell, "Accommodating Thickness in Origami-Based Deployable Arrays1," *Journal of Mechanical Design*, vol. 135, p. 111005, Nov. 2013.

[62] K. Kuribayashi, K. Tsuchiya, Z. You, D. Tomus, M. Umemoto, T. Ito, and M. Sasaki, "Self-deployable origami stent grafts as a biomedical application of Ni-rich TiNi shape memory alloy foil," *Materials Science and Engineering: A*, vol. 419, pp. 131–137, Mar. 2006.

[63] E. Boatti, N. Vasios, and K. Bertoldi, "Origami Metamaterials for Tunable Thermal Expansion," *Advanced Materials*, vol. 29, no. 26, p. 1700360, 2017. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/adma.201700360.

[64] N. Turner, B. Goodwine, and M. Sen, "A review of origami applications in mechanical engineering," *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 230, pp. 2345–2362, Aug. 2016.

[65] A. S. Gillman, K. Fuchi, and P. R. Buskohl, "Discovering Sequenced Origami Folding Through Nonlinear Mechanics and Topology Optimization," *Journal of Mechanical Design*, vol. 141, Apr. 2019. Publisher: American Society of Mechanical Engineers Digital Collection.

[66] T. Tachi, "Generalization of Rigid-Foldable Quadrilateral-Mesh Origami," *Journal of the International Association for Shell and Spatial Structures*, vol. 50, pp. 173–179, Dec. 2009. Publisher: International Association for Shell and Spatial Structures (IASS).

[67] E. D. Demaine and J. O'Rourke, *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007. Google-Books-ID: ycYLAQAAQBAJ.

[68] E. A. Peraza Hernandez, D. J. Hartl, and D. C. Lagoudas, "Design and simulation of origami structures with smooth folds," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, p. 20160716, Apr. 2017.

[69] W. Li, *A NEW DESIGN METHOD FRAMEWORK FOR OPEN ORIGAMI DESIGN PROBLEMS*. PhD thesis, Texas A&M University, College Staion, TX, 2014.

[70] K. Fuchi and A. R. Diaz, "Origami Design by Topology Optimization," *Journal of Mechanical Design*, vol. 135, Nov. 2013. Publisher: American Society of Mechanical Engineers Digital Collection.

[71] K. Fuchi, P. R. Buskohl, G. Bazzan, M. F. Durstock, G. W. Reich, R. A. Vaia, and J. J. Joo, "Origami Actuator Design and Networking Through Crease Topology Optimization," *Journal of Mechanical Design*, vol. 137, p. 091401, Sept. 2015.

[72] D. Sessions, G. Huff, J. Ruff, K. Fuchi, A. Cook, A. Gillman, A. Pankonien, and P. Buskohl, "Coupled Structural-Electromagnetic Analysis of Origami-Inspired Adaptive Structures," in

2019 International Applied Computational Electromagnetics Society Symposium (ACES), pp. 1–2, Apr. 2019.

[73] *Abaqus Analysis User's Manual*, vol. Version 6.14-2. Woodlands Hills, CA: Dassault Systèmes of America Corp., 2007.

[74] J. N. Reddy and J. N. Reddy, *An Introduction to the Finite Element Method*. McGraw-Hill, 1993. Google-Books-ID: bXnIQgAACAAJ.

[75] M. Drela, "XFOIL: An Analysis and Design System for Low Reynolds Number Airfoils," in *Low Reynolds Number Aerodynamics* (T. J. Mueller, ed.), (Berlin, Heidelberg), pp. 1–12, Springer Berlin Heidelberg, 1989.

[76] K. Deb and R. B. Agrawal, "Simulated Binary Crossover for Continuous Search Space," tech. rep., 1994.

[77] F.-A. Fortin, F.-M. D. Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary Algorithms Made Easy," *Journal of Machine Learning Research*, vol. 13, no. Jul, pp. 2171–2175, 2012.

[78] H. C. Greenberg, M. L. Gong, S. P. Magleby, and L. L. Howell, "Identifying links between origami and compliant mechanisms," *Mechanical Sciences*, vol. 2, pp. 217–225, Dec. 2011.

[79] K. Miura, T. Kawasaki, T. Tachi, R. Uehara, R. J. Lang, and P. Wang-Iverson, *Origami6*. American Mathematical Soc., Dec. 2015. Google-Books-ID: mthYCwAAQBAJ.

[80] J. L. Silverberg, J.-H. Na, A. A. Evans, B. Liu, T. C. Hull, C. D. Santangelo, R. J. Lang, R. C. Hayward, and I. Cohen, "Origami structures with a critical transition to bistability arising from hidden degrees of freedom," *Nature Materials*, vol. 14, pp. 389–393, Apr. 2015. Number: 4 Publisher: Nature Publishing Group.

[81] A. Gillman, K. Fuchi, G. Bazzan, E. J. Alyanak, and P. R. Buskohl, "Discovering Origami Fold Patterns With Optimal Actuation Through Nonlinear Mechanics Analysis," American Society of Mechanical Engineers Digital Collection, Nov. 2017.

APPENDIX A

AUGMENTATION OF SPIDRS FOR GRAPHS BOUNDED BY NON-LINEAR FUNCTIONS

The SPIDRS interpreter (c.f. 2.2) was augmented in order to be applied to the graph presented in Section 3. This argumentation is required due to the previous assumption that all connections between nodes could be represented by linear functions. This assumption assumption only effects the creation of new nodes, as it interpolates the location of the new node along a line between the two preexisting nodes. It is important to note that this adaptation does not invalidate the planar graph assumption, meaning that half-edge data structure is still valid for the augmented version of the SPIDRS interpreter. There are only two operations that cause node creation, therefore requiring modification, Move-Real (c.f. 2.2.0.2) and Create-Real (2.2.0.2). In this application, SPIDRS is only augmented to create nodes using non-linear functions to described outer boundaries of the initial graph (i.e. all interior operations will remain linear and follow previous described procedures).

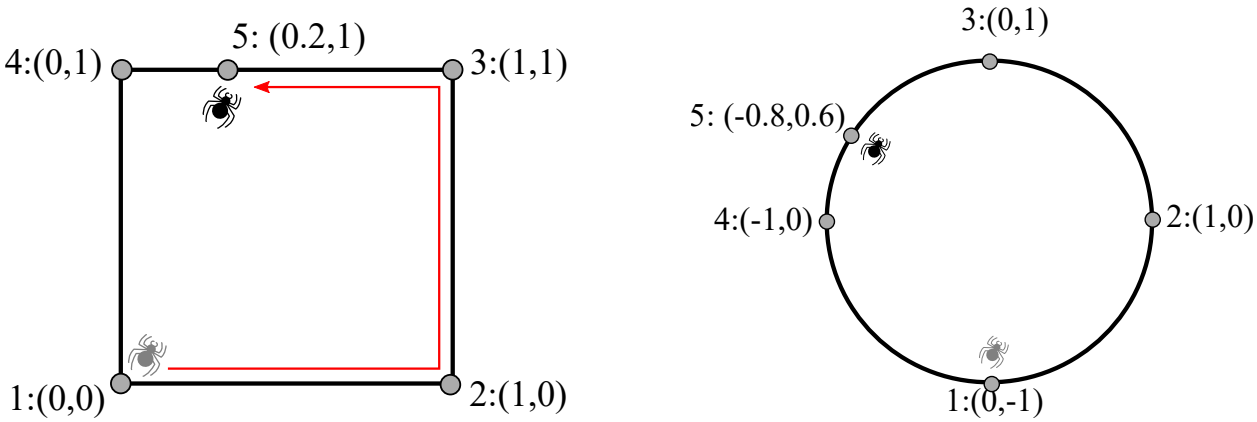
In both operations the creation of the new node is dictated by the following procedures. First, the SPIDRS operator proceeds $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes forward in the current face. Next, the SPIDR moves by the ratio $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor = n$ of the length of the current edge. To find the coordinates associated with this location, a linear interpolation is utilized as shown here, $x_i = x_0 + (x_f - x_0) \times n$ and $y_i = y_0 + (y_f - y_0) \times n$. To apply this process to a non-linear edge, the arc length of the function is taken and multiplied by the ratio of the current edge the node should move, as shown here $n \times \int_{x_f}^{x_i} s(x) = \int_{x_0}^{x_i} s(x)$. This value is then set equal to the arc length function and can be solved to find the x_i and y_i values. However, solving these arc length functions can be complicated for difficult shape equations, and for shapes like an airfoil, the arc length can be approximated by the x-coordinate. In the following examples, this specific approximation is utilized.

A.1 Move-Real

Similarly to the original implementation, the move-real command is represented in the L-System encoding as $B(\sigma_{\alpha_1})$. This command operates in the same way by the moving the SPIDR by $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes in the current face, and then to a newly created node $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor$ in between the current node and next node.

An example of the original and augmented move-integer commands, $B(0.7)$, are shown below in Figure A.1. In both examples, assume that the SPIDR begins at node 1 and receives the command, $B(0.7)$. Since $N = 4$, $\lfloor N \times \sigma_{\alpha_1} \rfloor = \lfloor N \times 0.7 \rfloor = 2$, which causes the SPIDR to advance two nodes to node 3. Next, node 5 will be created between nodes 3 and 4 by moving the SPIDR $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor = 2.8 - 2 = 0.8$ of the distance between them. In Figure A.1(a) a square graph is considered and a linear interpolation between nodes 3 and 4 is conducted. This interpolation results in $x_5 = x_3 + (x_4 - x_3) \times 0.8 = 1.0 + (0.0 - 1.0) \times 0.8 = 0.2$ and $y_5 = y_3 + (y_4 - y_3) \times 0.8 = 1.0 + (1.0 - 1.0) \times 0.8 = 1.0$. In Figure A.1(b) a circular graph is considered with a radius of 1.0 and centered at the origin. In this example, x_5 is determined using the same linear interpolation as before (i.e. $x_5 = x_3 + (x_4 - x_3) \times 0.8 = 0.0 + (-1.0 - 0.0) \times 0.8 = -0.8$), but the y-coordinate is now a function of the x-coordinate (i.e. $y_5 = f(x_5)$). In this specific example, a circle with the equation $x^2 + y^2 = 1$ is considered, so we are able to determine the the y-coordinate is $y_5 = \sqrt{1 - (-0.8)^2} = \pm 0.6$. Due to the initial edge definitions, it is determined that this operation is occurring on the upper edge of the curved graph, therefore $y_5 = 0.6$.

In the final augmented implementation, the function that determines $y_n(x_n)$ is a polynomial which utilizes the CST parameters to define the shape of the airfoil. This augmentation is only applied to the creation of nodes on the outer boundary of the SPIDRS graph. This is quickly determined by tracking the outer boundary of the graph with the first face in $\boldsymbol{F}$. The upper edge of the airfoil is defined as the first edge in the edge list and the lower edge of the airfoil is defined as the second edge in the edge list, $\boldsymbol{E}$.



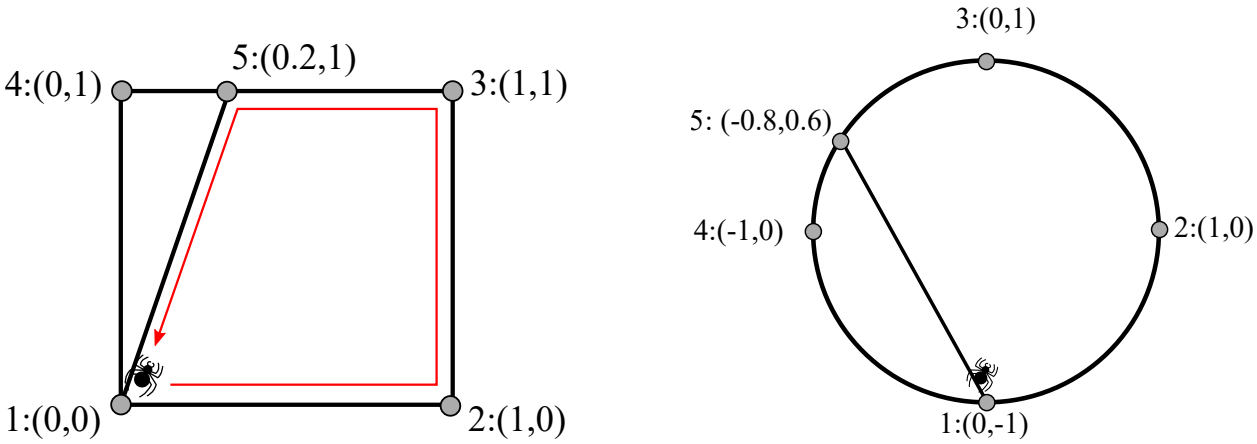
(a) Move-Real (b) Augmented Move-Real

Figure A.1: Examples of original and augmented Move-Real command B(0.7).

A.2 Create-Real

Again, the create-real command is represented in the L-System encoding as $D(\sigma_{\alpha_1}, \sigma_{\alpha_2})$. This command operates by moving the SPIDR by $\lfloor N \times \sigma_{\alpha_1} \rfloor$ nodes in the current face, then to a newly created node $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor$ in between the current node and next node, and then creating a new edge between the newly created node and the original node and returning the SPIDR to the original node. The σ_{alpha_2} parameter determines what material the newly constructed edge will be assigned.

An example of the original and augmented create-real commands, $D0.7, 0.2$ are shown belown in Figure 2.6. In this example, again assume that the SPIDR begins at node one when receiving the command. Since N = 4, the SPIDR advances $\lfloor N \times \sigma_{\alpha_1} \rfloor = \lfloor 4 \times 0.7 \rfloor = 2$ nodes forward to node three. Next, node 5 will be created between nodes 3 and 4 by moving the SPIDR $N \times \sigma_{\alpha_1} - \lfloor N \times \sigma_{\alpha_1} \rfloor = 2.8 - 2 = 0.8$ of the distance between them, similarly to the move-real command. Lastly, the SPIDR will create an edge between the current node, node 5, and the original node, node 1, and return to the original node. The material of the newly created edge is a function of σ_{alpha_2}. In Figure A.2(a) the new node's coordinates are generated using linear interpolation between the nodes of interest, identically to the original move-real command. This results in a node created at $(0.2, 1.0)$



(a) Create-Real (b) Augmented Create-Real

Figure A.2: Examples of original and augmented Move-Real command B(0.7).

and a new edge being formed between this new node and the original node, located at $(0.0, 0.0)$. In Figure A.2(b), the new node's x-coordinate is generated again using a linear interpolation between the two nodes of interest, (i.e. $x_5 = x_3 + (x_4 - x_3) \times 0.8 = 0.0 + (-1.0 - 0.0) \times 0.8 = -0.8$). The y-coordinate is now a function of the x-value and is found identically to the process described for the augmented move-real command. Then a new edge is created from the new node located at $(-0.8, 0.6)$ to the original node located at $(0.0, -1.0)$.

The process of finding the coordinates for the new nodes is identical between both the move-real and create-real commands. In the final implementation of the augmented create-real command the function used to determine the y-coordinate of newly created nodes is a polynomial which utilized the CST parameters to define the shape and size of the airfoil. This augmentation is only applied to nodes that are created on the curved boundary of the graph and does not apply to interior members of the graph.

APPENDIX B

TRIANGLE-TRIANGLE INTERSECTION DETECTION

Intersection detection was performed using the Möller's algorithm to detect intersections between two triangles in three dimensional space [**?**]. This algorithm was developed to aid in the computational efficiency of rendering hardware, and is uniquely applicable to the truss-based origami folding tool described in Section 2 due to its triangular base element.

This algorithm considers two triangles T_1 and T_2, which both consist of three vertices, $V_0^i, V_1^i, V_2^i, i \in [1, 2]$, and are located in the planes π_1 and π_2. The equation of the plane, $\pi_2 : N_2\dot{X} + d_2 = 0$, is computed, where X is any point on the plane. N_2 is the normal vector to the plane ($N_2 = (V_1^2 - V_0^2) \times (V_2^2 - V_0^2)$), and d_2 is the distance from the origin ($d_2 = -N_2\dot{V}_0^2$).

Next, the distance between the vertices, $d_{v_i^2}$,of T_1 and π_2 are computed,

$$d_{v_i^1} = N_2\dot{V}_i^1 + d_2, i = 0, 1, 2.$$

Overlap is rejected if no points of T_1 are on the plane, i.e. all $d_{v_i^2} \neq 0, i = 0, 1$, and 2, and all $d_{v_i^2}$ have the same sign. If these criteria are met, T_1 lies on one side of the plane π_2 and overlap is impossible. This process is then repeated for T_2 and π_1.

If all $d_{v_i^2} = 0, i = 0, 1$, and 2, the triangles are known to be co-planar. When this occurs, the triangles are projected to the axis aligned plane which maximizes their area (i.e. the axis of the largest component of the normal vector). After projection, a simple two-dimensional triangle-triangle overlap test is performed. First, test for an intersection between the closed edges of T_1 and T_2. If the edges of the projected triangles intersect, the original triangles intersect. Otherwise, T_1 must be tested to see if T_2 is totally contained with in its boundary and vice versa.

If the two planes π_1 and π_2 intersect, but are not co-planar, they are intersected by a line, L. The line of intersection is given by $L = O + tD$, where $D = N_1 \times N_2$ is the direction of the line

and O is a point on the line. Due to the previous calculations, we know that the triangles T_1 and T_2 intersect the line, L, however it's possible that these triangle still do not intersect each other. To determine this, a scalar interval that represents the intersection between T_i and L is computed. First, the vertices are projected onto L, giving

$$p_{v_i^1} = D\dot{(}V_i^1 - O).$$

Next, the line parameter value, t_1, for $B = V_0^1\bar{V}_1^1 L = O + t_1 D$ is computed. By letting K_i^1 denote the projection of V_i^1 onto pi_2, it can be noted that $\Delta V_0^1 B K_0^1$ is similar to $V_1^1 B K_1^1$, therefore

$$t_1 = p_{V_0^1} + (p_{V_1^1} - p_{V_0^1})\frac{d_{V_0^1}}{d_{V_0^1} - d_{V_1^1}}.$$

t_2 is calculated similarly and the interval for T_1 on L is between t_1 and t_2. The interval for T_2 is calculated the exact same way. If the two intervals overlap, then the triangles intersect.

There are many known optimization of this process, but due to the relatively small number of intersection detection evaluations required for this work, they are not implemented.