Generative Design using Lindenmayer-Systems and Numerical Optimisation

by

Izak Johannes Joubert

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechanical) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. M.P. Venter

April 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: ...19 February 2020...............

Abstract

Generative Design using Lindenmayer-Systems and Numerical Optimisation

Thesis: MEng (Mechatronic)

2020

Traditional methods of generative design for structural design applications typically utilise finite element analysis or other resource intensive methods. A more efficient method of design generation is needed in order for generative design to be utilised for these applications.

A generative design method is presented, utilising reduced-order modelling, Lindenmayer-systems and numerical optimisation. Through the use of recursive design techniques, this method reduces the computational cost of generating structural designs and through numerical optimisation is capable of developing targeted designs, using a genetic algorithm.

This method was applied to generate soft robotic bending actuator designs. The actuator is assembled from 15 modular cells and targeted for maximum bending, vertical and horizontal extension. An idealised reduced-order model is developed for the modular cells and reduced the evaluation runtime of the designs by a factor of approximately 6600. In all target objectives, the generated designs produced comparable results to a full 3D finite element model.

Soft robotic bending actuator designs for grasping various different objects were also developed. Grasping was prescribed through curve fitting of the actuator to the object. All designs generated exhibited successful grasping of the objects of different sizes and positioned at different locations.

Uittreksel

Generatiewe ontwerp met behulp van Lindenmayer-stelsels en numeriese optimering

("Generative Design using Lindenmayer-Systems and Numerical Optimisation")

Tesis: MIng (Megatronies)

2020

Tradisionele generatiewe ontwerp metodes vir strukturele ontwerpe maak tradisioneel gebruik van eindige elementanalise of ander hulpbron intensiewe metodes. 'n Meer effektiewe metode van ontwerp generering is nodig om generatiewe ontwerp vir hierdie toepassings te gebruik.

'n Generatiewe ontwerp metode word ontwikkel met behulp van verminderde-orde modellering, Lindenmayer-stelsels en numeriese optimering. Deur gebruik te maak van rekursiewe ontwerp tegnieke verminder hierdie metode die berekenings koste van die generering van strukturele ontwerpe en deur numeriese optimering is dit in staat om doelgerigte ontwerpe te ontwikkel met behulp van 'n genetiese algoritme.

Hierdie metode is toegepas om sagte robot buig toestel ontwerpe te genereer. Die buig toestel is saamgestel uit 15 modulêre eenhede en is gemik op maksimum buig-, vertikale en horisontale verlenging te bekom. 'n Geïdealiseerde beperkte orde model word vir die modulêre eenhede ontwikkel wat die evalueringstyd van die ontwerpe verminder met 'n faktor van ongeveer 6600. In al die gevalle het die gegenereerde ontwerpe vergelykbare resultate gelewer met 'n volledige 3D-eindige elementmodel.

Sagte robot buig toestel ontwerpe vir die gryp van verskillende voorwerpe is ook ontwikkel. Gryping is voorgeskryf deur die liggaam van die buig toestel op die voorwerp te pas met behulp van krommepassing. Al die ontwerpe wat gegenereer is, het die voorwerpe, van verskillende groottes en op verskillende plekke geposisioneer, suksesvol aangegryp.

Acknowledgements

I would like to express my sincere gratitude to my fiancè Elsabè, you are my rock and my light and I would not have been able to finish this thesis without you.

Also, to my dear brother Jakob, for supporting me financially through my postgraduate studies. Jakes, thank you for everything!

Contents

List of Figures

Chapter 1

Introduction and Overview

1.1 Problem Statement

Traditionally, the design process for complex, non-trivial objects uses a trail-and-error approach. The exploration of the design space and alternative constructs occur during the concept phase of the object development, Krish (2011). As the process advances, the design of the object becomes more fixed and changes are rarely implemented after the concept phase. A thorough exploration is therefore required to ensure that an optimal design is selected. Numerous design domains posses several unique solutions, each capable of fulfilling the requirements of the design scenario.

Generative design is a method that makes use of automated processes without designer intervention A design problem is formulated in terms of a set of objectives and constraints. A computational process is then used to explore and evaluate potential designs. With advancements in computational and processing power, more powerful generative design generators can be developed, Lobos (2018), Shea *et al.* (2005). This allows a more thorough exploration of all the possible constructs of the object subsequently yielding novel or optimised designs. Generative design is an iterative process wherein initial designs are developed and refined until the final design is generated as illustrated in Figure 1.1 showing the generative design process of a chair, Schwab (2019).

Figure 1.1: Iterative process of a chair being designed using generative design, Schwab (2019)

Venkataraman and Haftka conclude that although computational power has increased drastically in accordance with Moore's law, the request for higher fidelity designs increase as a result, which in turn require more computational power, Venkataraman and Haftka (2004). According to Cheney *et al.* (2013), generative design for engineering structures rely heavily on finite element (FE) analysis, which is notoriously resource-intensive. Less resource-intensive techniques are thus required to exploit these methods for engineering applications. Methods that allow design changes to be implemented throughout the design process, will deliver an advantage in that the user can observe the process and deliver inputs if and where necessary to drive the process in a specific direction. A less resource-intensive technique will also be capable of increased design iterations, allowing for a more in-depth exploration of the entire design space by generating and evaluating more designs in comparison to a manual design approach.

A generative design method must therefore be developed for structural design applications that is efficient, accurate and fast in order to be a suitable alternative to traditional design methods. By developing a faster and more efficient method, and exhaustive search of the typically large design domains of structural applications can be executed. This will not only allow faster design times but also deliver novel designs.

1.2   Aim and Objectives

The aim of this research is to develop a generative design method (GDM) for structural design applications, that functions as an alternative to traditional resource-intensive design techniques. It will endeavour to:

1. Evaluate existing generative design methods in terms of their suitability for structural design applications;

2. Develop a GDM capable of generating adequate designs for soft robotic bending actuators;

3. Using the developed GMD, design a soft robotic bending actuator that exhibits a target behaviour.

1.3   Scope and Assumptions

Several assumptions are made that restrict the scope of this research:

1. A generalised approach will be followed during the development of the GDM. The method will be developed as a baseline tool and not targeted at any specific application.

2. The suitability of the method will be evaluated only in the context of soft robotic bending actuators.

3. The developed method will be applied to 2-dimensional soft robotic bending actuator designs.

4. The performance heuristics will be determined based on the chosen application.

5. A static simulation will be used to evaluate the generated designs. Dynamic effects will not be considered.

1.4   Project Layout

A literature review will be provided in Chapter 2. The current state of generative design as a whole will be discussed. A detailed exploration of generative design, specifically for soft robotics will follow. The chapter will also review numerical optimisation, including the optimisation tools that are available and will touch on the machine learning techniques that may apply to this research. Chapter 3 will employ the developed method to generate designs in order to evaluate the performance and capabilities thereof. The results and a discussion will follow. Chapter 4 will develop a reduced-order model for the soft robotic

application to reduce the dimensionality and complexity of the application as an optimisation technique. The GDM will be combined with the reduced-order model to generate soft robotic designs in Chapter 5. These designs will be evaluated based on a selected performance measure that will be developed for the specific application. Here, the suitability of the research will be determined based on the performance of the generated designs. The method will also be extended, theoretically to other applications in soft robotics. The research conclusion follows in Chapter 6.

Chapter 2

Literature Review

2.1 Generative Design

Generative design was first introduced by Gullichsen and Chang (1985) for architectural applications. It was based on a theory developed by Christopher Alexander, that all architectural forms are made up of interacting patterns created through generative rules. Gullichsen *et al.* applied this theory to generate drawing views of 3-dimensional architectural forms, using the Prolog programming language. Generative design has since found increased use in architecture, (Shea *et al.* (2005), Chien and Flemming (2002), Kasmarik *et al.*, Krish (2011)), computer science, (MacDonald *et al.*, Troiano and Birtolo (2014), Salge *et al.* (2018)) and engineering, (Nordin (2018), Marinov *et al.* (2019), Oh *et al.* (2019), Lobos (2018)). Generative design can be viewed as a manual process, using pen and paper, or an automated process using software tools. Engineering applications range from automotive to construction. Figure 2.1 illustrates several iterations of the process of designing the chassis of an unmanned aerial vehicle.

In its most basic form, design can be expressed as producing some output, based on a particular input. Designers are tasked with creating a product/object to fulfil some requirement/task. During the process of designing, the designer applies their knowledge of the particular design space and produces an output as a result of not only the requirements explicitly imposed on the output, but also the designer's prior knowledge. In essence, therefore, designing concerns the application of knowledge to produce some output, subject to a set of implicit and explicit incentives and constraints. The design process has been classified into two distinct categories, namely the rational model, Axten *et al.* (1973) and the action-centric model, Ralph (2010).
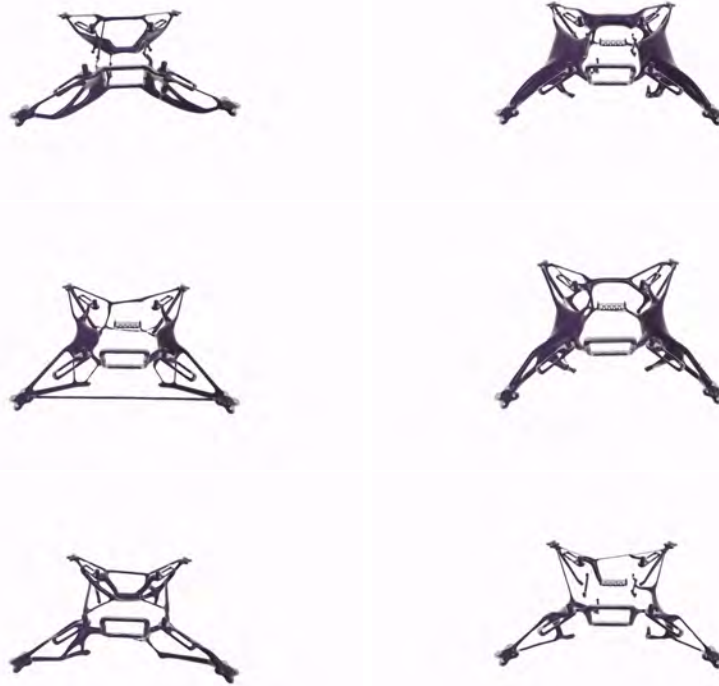
Figure 2.1: Several iterations in the automated design process of the chassis of an unmanned aerial vehicle using the Autodesk software package, Autodesk Inc.

The rational model describes the design process as an attempt to deliver an optimised output based on problem specific requirements and constraints and is a plan-driven process. The process can be viewed as a sequential flow of stages. Typically, these stages can be grouped into three phases namely a) the conceptual phase, where the requirements are finalised and an initial design is created, b) the production phase, where the design is produced and evaluated and c) the implementation phase wherein the design is concluded and implemented. This model has been criticised greatly on two accounts. Firstly, designers rarely follow this approach, (Cross *et al.* (1992), Ralph (2010), Schön (2017)) and secondly, this model enforces an inflexible design flow, which is contradictory to actual design processes (McCracken and Jackson (1982), Cross (2011)).

The action-centric model, in contrast, describes the design process as a creative technique, with improvisation and unstaged and coupled phases, that remain connected throughout, Ralph (2010). Here, the design "grows" as it develops and requirements are met informally and not necessarily in order. This model is substantiated through ample empirical evidence, supporting

this methodology as the one followed by most designers, Cross *et al.* (1992). Notwithstanding, the overall design process remains prevalent in the action-centric model, with most approaches having informal concept, production and implementation phases. The extent to which these phases are interconnected, however, differs substantially from the rational model. The action-centric model is viewed as a web, rather than a flow of processes (Truex *et al.* (2000), Beck *et al.* (2001)).

Design automation, therefore, concerns automating the process of design rather than arbitrarily generating designs. Through automation, new insights may be developed into the process of design itself. As an extension of design automation, generative design carries several benefits for designers, including providing a different perspective on the design, providing a more thorough exploration of the design and solution space and augmenting the design process. It would be impossible for pen-and-paper designers to execute a complete exploration of both the design and solution space within the time requirements imposed by modern scenarios. Generative design agents only require the constraints and requirements imposed on the design space to produce solutions.

ESP (encapsulation, syllabus and pandemonium) is an approach to generative design, proposed by Lessin *et al.* (2013), which includes a human-designed *syllabus*, which prescribes learning several simpler tasks to develop a complex behaviour, *encapsulation* that preserves these simple tasks for future recollection and *pandemonium*, a mechanism that refines and determines the better of two similar tasks within the increasingly complex behaviour. Also known as task decomposition, this method aims to reduce the complexity of generating solution by splitting the requirements into several smaller, simpler tasks, which are easier to achieve and then recombining these educated units to accomplish a more complex behaviour (Brooks (1986), Bay (1995), Celaya and Porta (1998)).

Generally, due to the complexity of a typical design domain, an encoding or representation is required to define the properties, parameters, constraints, etc. of the design space for computational manipulation. Defining a suitable encoding is difficult, especially in evolutionary computation (Schoenauer (1996)). Generative encoding, encodings that specify how to generate the body or phenotype of the design, have been presented as solutions to this problem, Schoenauer (1996), Bentley (2000). These encodings are called genotype-phenotype encodings and are akin to the DNA - physical attribute relationship of all biological life. The DNA or genotype of an organism will influence an observable change in the physiology or phenotype of the organism.

Generative encoding therefore provides a more compact encoding scheme for the typical design domain by re-using pieces of the genotype to construct

the phenotype. Rather than representing the entire genotype, this type of encoding only represents the developmental procedure of the genotype. For generative design, a generative encoding is applied to actively create designs from smaller parts rather than generating a full design from start to finish. The fundamental concept is recursion. By recursively developing a design from smaller parts, larger more complex designs are created at reduced computational cost, Hornby and Pollack (2002*a*).

Some generative encodings include cellular automata (CA), Lindenmayer systems (L-systems) and shape grammars (SGs).

Cellular Automata

Cellular automata have been documented and classified by Chopard (2012):

> Cellular automata (often termed CA) are an idealisation of a physical system in which space and time are discrete, and the physical quantities take only a finite set of values.

According to Wolfram (1984) CA are made up of simple units, individually capable of only rudimentary behaviours. However, whole CA can be extremely complex, serving as suitable models for a wide variety of biological pattern formations. They are constructed as an n-dimensional lattice or grid of cells with each cell possessing a finite number of states. The complexity of the CA is linked to the type of lattice, Kasmarik *et al.*. A CA evolves by applying a specific set of rules to each cell in the lattice, based on the state of each neighbouring cell, at each time step. A basic example is illustrated in Figure 2.2. The lattice is constructed as a display of the states of the 53 horizontal cells at each time step. Each cell possesses only two states, *on* and *off* represented by black and white respectively. The rule applicable to this example is as follows:

A cell will only switch on, in the subsequent time step, if the cells immediately to the left or right of that cell were switched on during the previous time step. Otherwise, the cell will switch off or remain switched off.

The starting condition for this time step is the single cell that is switched on as shown in Figure 2.2 at time step 1. This example illustrates the capacity of CA to provide a better understanding of a system's macroscopic structure and supports the statement made by Epstein (1999):

> Even perfect knowledge of individual decision rules does not always allow us to predict macroscopic structure. We get macro-surprises despite complete micro-knowledge.
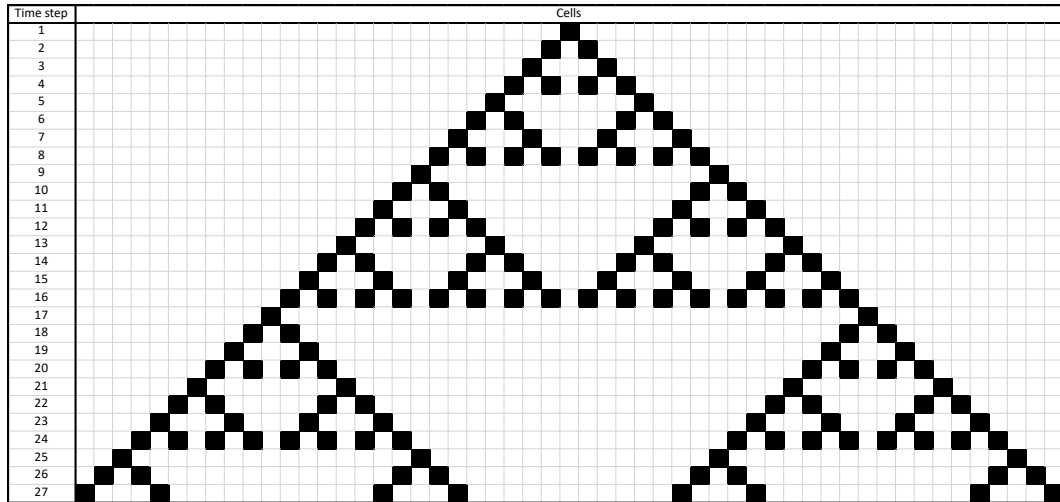
Figure 2.2: A cellular automata cell grid being developed through time. Each cell's state is determined by the states of the cells on either side of that cell during the previous time step

Ilachinski (2001) classified the applications of CA into four main areas namely computational engines, discreet dynamical system simulators, conceptual vehicles for exploring pattern formations and original models for fundamental physics. Engineering applications range from robotics, Beni and Wang (1993) and artificial intelligence, de Garis (1994) to fluid dynamics, Rothman (1987) and sewer network optimisation, Rothman (1987). The development of a CA is invariably context-specific and built upon cell states. The resulting structure of the cell lattice is function-driven as a result of the cell states.

Lindenmayer Systems

Lindenmayer systems (L-systems) is a type of formal grammar, a set of rules that describes how to form objects using a specific alphabet, developed as a mathematical approach to modelling morphogenetic (pattern-generating) processes proposed by Aristid Lindenmayer, Lindenmayer (1968). These systems were conceived to model the recursive nature of biological development, but has since been used in a variety of fields, the most prominent of which is perhaps (the field of) computer graphics, Rozenberg and Salomaa (2012). A complex object is created through the successive replacement of certain parts of the object based on specified rules, Françon (1997).

The simplest L-systems are deterministic and context-free and they are called DOL-systems. A formal definition for DOL-systems is provided by Françon (1997) and detailed in, Lindenmayer and Rozenberg (1972), Rozenberg and Salomaa (1974) as follows:

Let V denote an alphabet, V^* the set of all words [constructs of the alphabet] over V, and V^+ the set of all non-empty words over V. A *string OL-system* is an ordered triplet $G = <V, \omega, P>$ where V is the *alphabet* of the system, $\omega \in V^+$ is a non-empty word called the *axiom* and $P \subset V \times V^*$ is a finite *set of productions*. A production $(a, \chi) \in P$ is written as $a \longrightarrow \chi$. The letter a and the word χ are called the *predecessor* and the *successor* of this production, respectively. It is assumed that for any letter $a \in V$, there is at least one word $\chi \in V^*$ such that $a \longrightarrow \chi$. If no production is explicitly specified for a given predecessor $a \in V$, the *identity production* $a \longrightarrow a$ is assumed to belong to the set of productions P. An OL-system is *deterministic* (noted *DOL-system*) if and only if for each $a \in V$ there is exactly one $\chi \in V^*$ such that $a \longrightarrow \chi$.

More simply, an L-system consists of an alphabet, a set of productions or rules and a starting sequence or axiom. Within the alphabet, two subsets preside, namely variables and constants. The set of productions or rules prescribe the replacement of the variables within the L-system. Through iterative recursions, the L-system is developed into what is known as an L-string, made up of characters in the alphabet of said L-system.

A practical example of L-systems is the approximation of the Sierþinski triangle, Figure 2.3. The Sierþinski triangle can be approximated using the following L-system:

An L-string can be created by recursively applying the rules of the L-system to the current L-string. At each recursion, the rule is applied only to the applicable characters of the L-string, as it stands, at the current recursion. Instinctively, this will produce enlarged L-strings for higher recursions. For the L-system denoted in Table 2.1, the L-string is developed as follows:

Table 2.1: The L-system parameters for creating a visual representation of the Sierþinski triangle

Description	Value
Variables	(A, B $\in$ V)
Constants	(+, - $\in$ V)
Axiom	A
Rules	(A → B-A-B)
	(B → A+B+A)
Angle	60°

Axiom: A
Recursion 1:   B-A-B
Recursion 2:   A+B+A-B-A-B-A+B+A
Recursion 3:   B-A-B+A+B+A+B-A-B-A+B+A-B-A-B...
Recursion 4:   A+B+A-B-A-B-A+B+A+B-A-B+A+B+A...

Here, the axiom character A is replaced following the rule A → B-A-B in the first recursion. Then both B characters are replaced following the rule B → A+B+A as well as the A character being replaced again in the second recursion and so forth. This system also excellently illustrates the tendency of some L-strings to enlarge abruptly. The third recursion has a total of 53 characters and the fourth recursion a total of 161 characters.

The A and B characters represent a unit step forward, the + and - characters a left and right turn respectively at a 60°angle. If this system were to be illustrated, the use of a turtle interpretation may be employed. The turtle interpretation is discussed in Françon (1997) and detailed further in Hart *et al.* (1987). Simply put, if a turtle were to be placed on a beach and directed to walk in a specific fashion, the line of its footsteps in the sand would create a physical representation of this imposed limitation. Therefore, a turtle interpretation of the L-system for the Sierṗinski triangle can be illustrated as shown in Figure 2.3.

A relationship can thus be established between the L-string and its physical geometry, herein referred to as the L-creature. The L-string is the encoding of the L-creature creating the genotype-phenotype relationship wherein the genotype contains the information of the creature and the phenotype contains the actual observed properties such as geometry, development and behaviour. This relationship in itself provides somewhat of a reduced-order representation of the L-creature. By evaluating the L-string, an approximation of the performance of the L-creature may be gained.

L-strings have been used as genotypes for various objects including biological matter like algae, art, Konrády *et al.* (2016), music, Kim and Talib (2010) and even tables, Hornby and Pollack (2002*a*). Another possible application of
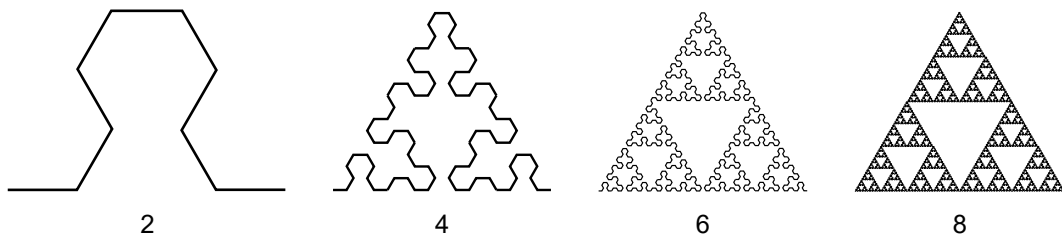

Figure 2.3: The development of an approximation of the Sierṗinski triangle, through the use of L-systems, for 2,4,6 and 8 recursions.
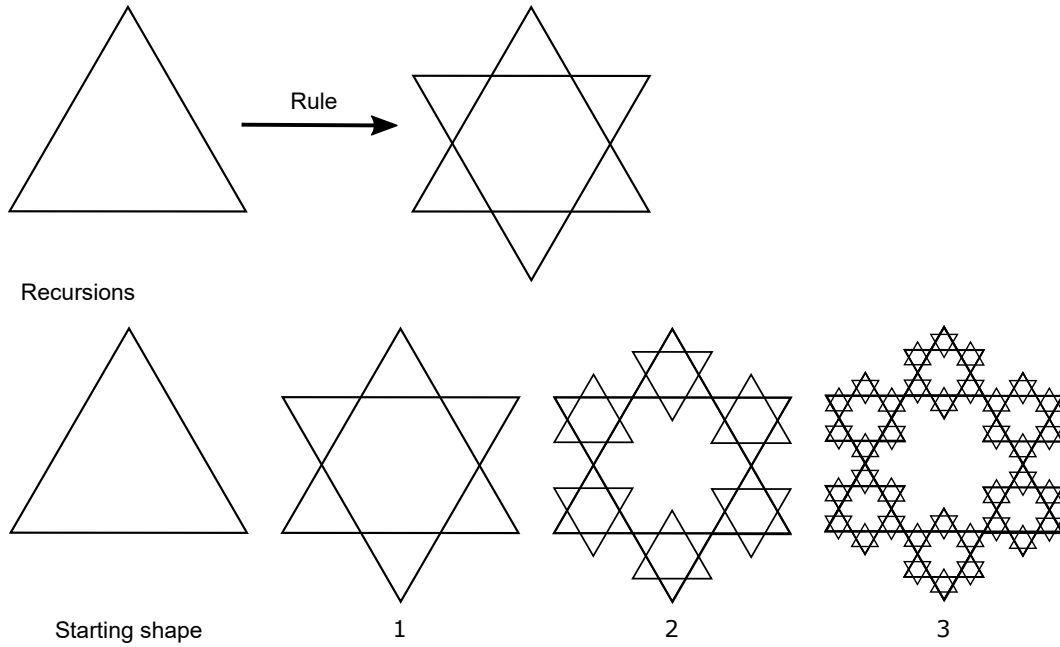
Figure 2.4: A snowflake shape being recursively generated using a shape grammar with a rule, specifying that each triangle is replaced with a hexagram

this generative approach lies in the field of soft robotics.

Shape Grammars

SGs have been documented in Stiny (1994). SGs specify a method of performing recursive shape generation, Tapia (1999) and also constitute a type of formal grammar similar to L-systems. The difference between shape grammars and L-systems lies in the fact that L-systems produce a string of characters that are encoded to some physical manifestation whilst shape grammars only use shapes. Initially developed for painting and sculpture applications (Stiny and Gips (1972)), they have since found increased use in architecture (Çağdaş (1996), Coutinho *et al.* (2013), Muslimin (2017), Benrós *et al.* (2012), Yue *et al.* (2012)) as they strictly adhere to specified forms. SGs are also notable in engineering, Cagan (2001). A simple example is illustrated in Figure 2.4. It illustrates the creation of a snowflake shape starting from a triangular shape. The recursion is specified to replace each triangle with a hexagram. The image illustrates three recursions.

Generative design aims to provide new design processes that are computationally cheaper and faster yet delivers novel, efficient and expandable designs, Shea *et al.* (2005). Using generative encodings, designs may be more efficiently represented while still providing enough insight and accuracy. These method

thus allow a less extensive investigation of all the generated designs to determine the most optimal. In-depth analysis is therefore reserved for optimal designs and computational time and cost is not wasted.

2.2 Soft Robotics

Soft robotics comprise the sub-field of robotics concerned with utilising compliant materials in the design and manufacturing of robotic devices, Wang *et al.* (2015), often emulating biological life, Trivedi *et al.* (2008). The term "soft" refers to the material used during fabrication of the robot, generally possessing hyperelastic properties and displaying a higher level of compliant behaviour in comparison with traditional robot materials.

Traditionally, robotic fabrication materials occupy a Young's modulus range of $10^9 - 10^{12}Pa$, Rus and Tolley (2015). These comprise of metals and hard plastics, commonly possessing rigid properties. Biological life conversely is often composed of materials in the Young's modulus range of $10^4 - 10^9Pa$, Rus and Tolley (2015), exhibiting elastic properties. Soft robotics aims to emulate these biological life form compositions and apply them in robotics.

Soft robots present the opportunity for increased robot-human interaction outside of carefully controlled environments and conditions. Rigid body robotic devices generally need to be restricted, with little to no human interaction allowed whilst the robotic device is in operation. These rigid body robots often move with such speed that a human-robot collision could be fatal. Due to the rigidity of the robot materials, no force from the impact can be absorbed by the material, whereas soft robot materials can absorb some of the impact by either bending or deformation, Martinez *et al.* (2014). Although it is possible to design rigid body robotic devices to include compliant behaviour (Stephens and Atkeson (2010), Shin *et al.* (2016)), this often requires additional sensors and increasingly complex control algorithms. Soft robots include passive compliance, as a property of the materials used to manufacture the robot. Therefore, it is seldom the case that additional sensors are required to realise a safe operating space for human interaction.

Soft robotic materials possess hyperelastic properties and it is often this property that is most attractive for soft robotic applications. Some manufacturing materials include elastomers and electroactive polymers (Trivedi *et al.* (2008), Shi *et al.* (2012)), shape-memory alloys (Lin *et al.* (2011), Laschi *et al.* (2012)), hydrogels (Higashi and Miki (2014), Duan *et al.* (2017)) and foam (Robertson and Paik (2017)).

Apart from safe human interaction, soft material compliance is also exploited in many soft robotic applications as a means to achieve locomotion (Meng *et al.* (2017), Robertson and Paik (2017), Park and Wood (2013), Laschi *et al.* (2012)). Due to the elastic deformation allowed by these materials, the soft robotic device can be designed to limit these deformations in some directions while allowing them to move freely in others and thereby realising the locomotion of the robotic device.

Soft robotic actuators are manufactured from soft materials such as silicon. Gorissen *et al.* (2017) provides a detailed review. These actuators are typically fluid powered and are commonly known as elastic inflatable actuators (EIAs). Fluid actuation carries the benefits of having a small footprint, as it does not require considerable supporting equipment to function, as well as being less invasive due to the fact that the fluid used for actuation can be biocompatible, Gorissen *et al.* (2017). The most common types of soft robotic actuators are expanding, contracting, twisting and bending actuators, see Figure 2.5.

Most soft robotic actuators are anisotropic elastic structures that use fluid actuation to achieve movement. Anisotropy may be achieved either through engineered asymmetric construction of the actuator geometry or by embedding alternate materials within the body of the soft robotic actuator. Complex soft robotic actuator movement may be achieved by combining these techniques within the same actuator (Polygerinos *et al.* (2015)).

Typically, bending in soft robotic actuators is achieved by constructing the actuators with an asymmetrical cross-section and an inflatable void. This asymmetry causes it to have a preferential bending direction. The uneven ma-
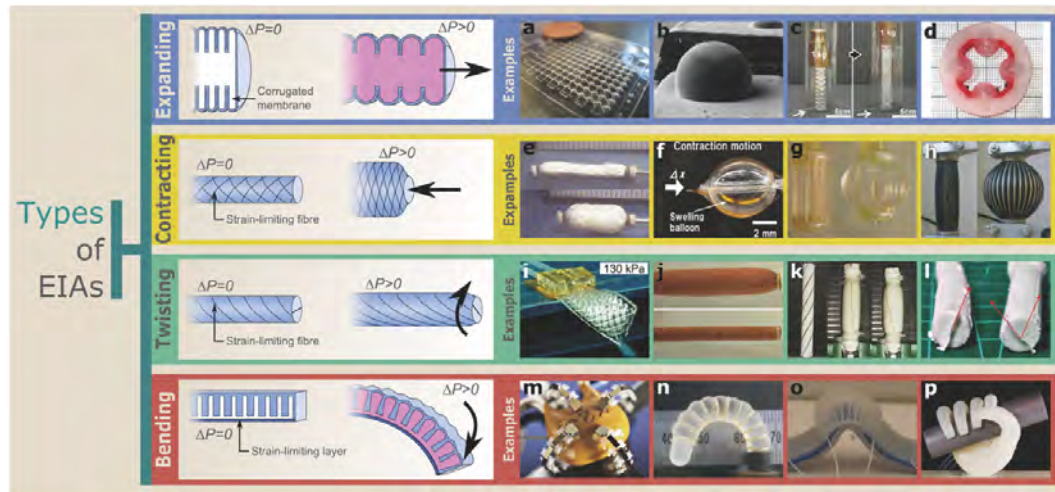

Figure 2.5: The four common types of soft robotic actuators are expanding, contracting, twisting and bending actuators, Gorissen *et al.* (2017)

terial distribution creates a strain limiting layer which, when inflated, cannot deform equally to the remaining material layers. This strain limiting layer therefore creates increased strain within the actuator and pulls it in a certain direction as shown in Figure 2.6. A strain limiting layer can also be realised in the soft robotic actuator using or embedding different, more rigid materials (Suzumori *et al.*, Hirai *et al.*, Jeong and Konishi (2006), Gorissen *et al.*).

Biomimetic and Soft Robot Design

Biomimetics or biomimicry involves the practice of emulating biological traits or mechanisms in technology Vincent *et al.* (2006). From an engineering perspective, this commonly involves developing technologies that perform or exhibit behaviours similar to biological organisms.

Due to the properties of the materials from which they are manufactured or fabricated, soft robots inherently possess the potential to mimic biological organisms, as most biological materials are soft materials. Considerable research regarding biomimetic and soft robotics has been conducted and delivered promising results (Trivedi *et al.* (2008), Laschi *et al.* (2012), Marchese *et al.* (2014), Kwon *et al.* (2008), Laschi and Cianchetti (2014), Must *et al.* (2015), Cho *et al.* (2009)).

Another interesting aspect is the embedding of behaviour in soft robots, Iida and Laschi (2011), Ellis *et al.* (2019). Embedded behaviours are inherent in biological lifeforms and several researchers have employed this technique in different scenarios with promising results (Kim *et al.* (2012*b*), Nishikawa
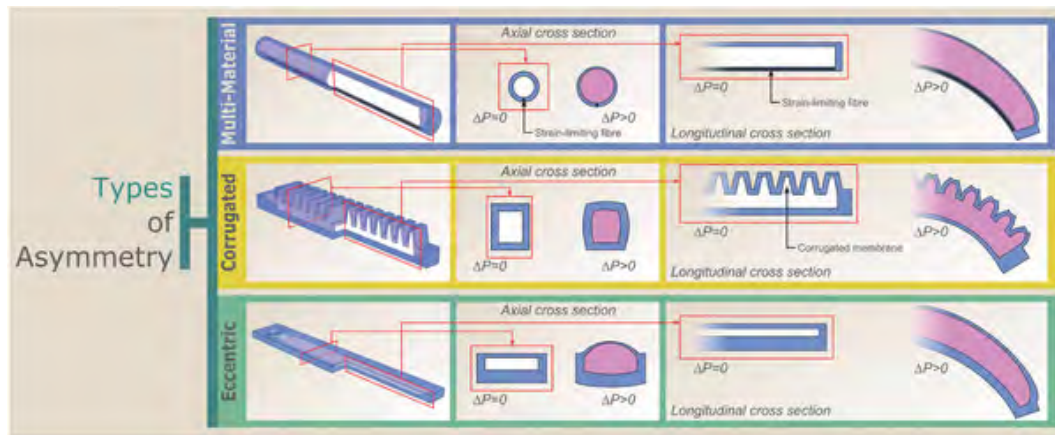

Figure 2.6: Soft robotic bending actuators can bend by creating asymmetry in the actuator cross-section resulting in a strain limiting layer/plane that pulls the actuator in a certain direction when inflated, Gorissen *et al.* (2017)
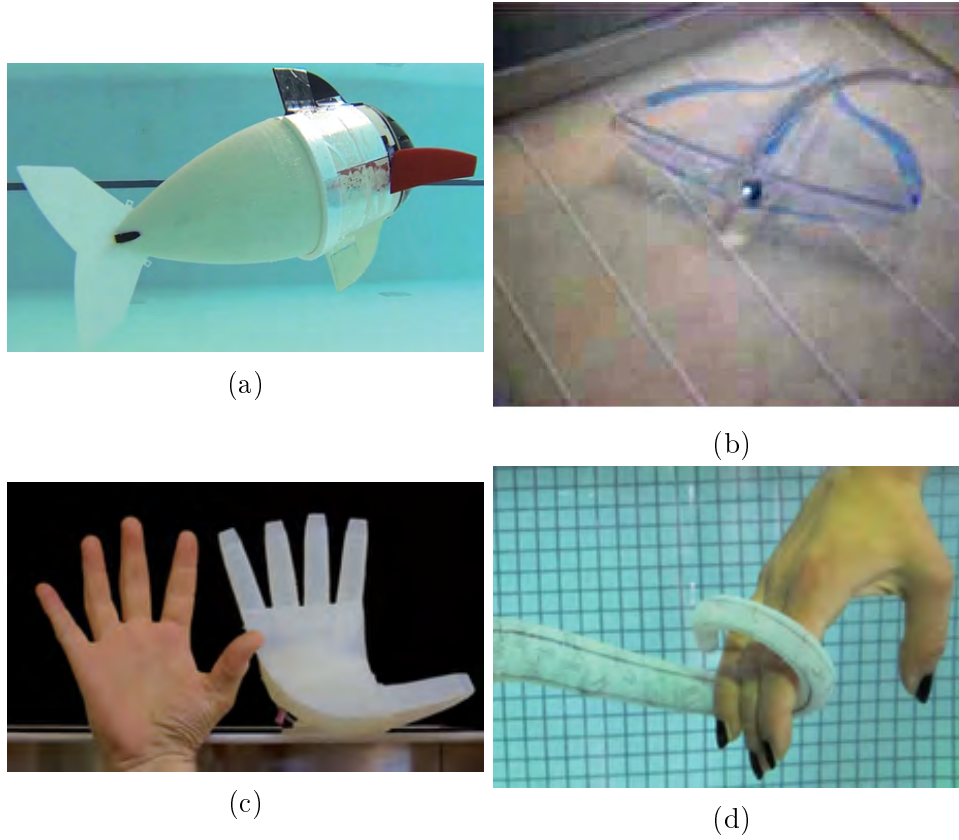
Figure 2.7: Examples of biomimicry in soft robotics. a) A soft robotic fish capable of rapid escape manoeuvres, b) A manta robot with a swimming speed of 100 mm/s, c) A synthetic soft robotic hand capable of dexterous movement, d) An octopus inspired manipulator. Rus and Tolley (2015).

et al. (2018), Che *et al.* (2019)). By embedding behaviours in soft robots, manipulations can be achieved without specific actions, the result of which is a naturally intelligent and compact design that eliminate the need for additional apparatus, Trimmer *et al.* (2012).

Generative Design in Soft Robotics

Generative design for robotics, and especially soft robotics, is a fairly new concept, Lipson (2014). Several factors have hindered this advancement, most prominently manufacturing and designing constraints. With the advent of additive manufacturing and the increase of computational capabilities, novel methods are being developed to generate and manufacture soft robotic designs, Pollack *et al.* (2001), Hornby and Pollack (2002*b*).

Generative design in soft robotics, utilising traditional FE software, is espe-

cially difficult due to the hyperelastic, non-homogeneous material properties of soft robots, Moseley *et al.* (2016). As such, they have been limited to specific materials, morphologies and scenarios, (Suzumori *et al.* (2007), Roche *et al.* (2014), Udupa *et al.* (2014), Polygerinos *et al.* (2015)). Soft material models are more complex than traditional robotic materials and, therefore, exacerbate the computational requirements for FE analysis, Renda *et al.* (2012).

Additionally, soft material models such as the Neo-Hookean and Mooney-Rivlin models are accurate only in low-strain environments as a result of using linear approximations of the strain invariants, Yeoh (1993). At higher strains, the accuracy of these material models decreases significantly, Kim *et al.* (2012*a*). The soft materials possess dynamic properties that are volatile and difficult to predict. In many cases, these difficulties manifest in the form of relative motions in the soft materials. When the material is actuated it may experience additional stresses and forces in unintended directions, Minh *et al.* (2012), thus contributing to the non-linearity thereof and increasing its complexity.

Nevertheless, generative design may be employed to create soft robotic devices, Cheney *et al.* (2013). A promising approach to generative design for soft robotics is reduced-order modelling, Chenevier *et al.* (2018). Several methods have been explored precisely for this purpose, including constant piecewise curvature (Runge *et al.* (2017)), volume-pressure relationships (Gilbertson *et al.* (2017)), and application of the Cosserat rod theory (Renda *et al.* (2014), Chenevier *et al.* (2018)). Real-time FE modelling has also been applied in conjunction with reduced-order modelling (Laschi *et al.* (2012), Duriez (2013), Largilliere *et al.* (2015), Roßmann *et al.* (2015), Coevoet *et al.* (2017), Bieze *et al.* (2018)), however, extreme simplicity is needed for these methods to function. Unfortunately, due to the extensive simplifying of the material models and soft robot geometries, results are often inaccurate, Chenevier *et al.* (2018).

Lindenmayer Systems and Soft Robotics

L-systems may be visualised using a turtle representation. This representation may prove useful in establishing a basis for a reduced-order model for various applications. Specifically, this approach can be applied to soft materials (Hornby and Pollack (2001*b*), Rieffel and Smith (2010)), and soft robotics, Rieffel and Smith (2012). A further implementation of this approach co-developed the geometry and neural controller of virtual creatures (Hornby and Pollack (2001*a*)), with increased success in comparison to previous attempts, Sims (1994), Komosiński *et al.*, Hiller and Lipson (2012).

In a generative design context, this approach has shown to deliver improved results in comparison with non-generative applications, Hornby and Pollack (2002*a*), who attributes this success to the modularisation inherent in L-systems. By generating designs using modules, design changes need only be implemented on a single module through which it will propagate to all instances of said module in each design. Say, for instance, generative design is employed to generate table designs. Should the table height be changed, this change need only be implemented in the table-leg module, whereafter, each table-leg module in the design will be altered inherently.

It has been argued that the only way to achieve sufficiently complex robotic geometries, relevant in engineering applications, is to use this type of modularisation approach, Hornby *et al.* (2001), Pollack *et al.* (2003). L-systems intrinsically possess modularisation utilising the L-system rules. Each rule can be viewed as a separate module that is recursively inserted into the design.

2.3   Numerical Optimisation

Generative design is an iterative approach at attempting to deliver an optimised result. Each generated design is evaluated against some predefined requirement, and subsequently improved to better meet these requirements. Therefore generative design constitutes a form of numerical optimisation. Numerical optimisation thus ensures that the generative design generator delivers optimal designs for the design domain by applying some optimisation scheme to the design domain. According to Nocedal and Wright (2006):

> Mathematically speaking, optimization is the minimization or maximization of a function subject to constraints on its variables.

Numerical optimisation is a technique used to establish the optimal solution to a mathematically definable problem. According to Arora (2006) optimisation can formally be classified as:

1. the *formulation* and

2. *solution* of a problem:

$$\underset{w.r.t\ x}{minimise}\ f(x), x = [x_1, x_2, \cdots, x_n]^T \in \mathbb{R} \tag{2.1}$$

subject to the constraints:

$$g_j(x) \leq 0,\ j = 1, 2, \cdots, m \tag{2.2}$$
$$h_j(x) = 0,\ j = 1, 2, \cdots, r \tag{2.3}$$

where $f(x), g_j(x)$ and $h_j(x)$ are scalar functions of the column vector $\boldsymbol{x} \in \mathbb{R}$. $\boldsymbol{x}$ denotes the *parameters* or *design variables* of the function; $f(x)$ denotes the *objective function*, which is the function of $\boldsymbol{x}$ that is to be minimised or maximised; $g_j(x)$ are the inequality constraints and $h_j(x)$ the equality constraints under which the *objective function* operates.

Numerical optimisation can be broadly classified into two categories namely, gradient and non-gradient based optimisation. Gradient based optimisation utilises gradients and line search to converge to an optimal solution and typically start from a single design point. Non-gradient based methods may initiate from numerous design points and are inspired from natural phenomenon. Genetic algorithms (GAs), simulated annealing and particle swarm optimisation (PSO) are some of the most common non-gradient based optimisation methods.

Although theoretically sophisticated, numerical optimisation, especially in the field of engineering, is inhibited by several factors (Arora (2006)) including:

1. expensive function evaluations e.g. structural FE analysis, multi-body system dynamics and computational fluid dynamics;

2. noisy data, either experimental or numerical;

3. discontinuity;

4. numerous local optima;

5. undefined function domains within the design domain; and

6. an overwhelming number of design variables.

By using reduced-order modelling, encodings and non-gradient based numerical optimisation in unison, this research hopes to overcome or eliminate these factors.

Genetic Algorithms

GAs were first introduced by John Holland in 1960 and later extended by his student David E. Goldberg, Goldberg and Holland (1988). According to The Math Works:

> The genetic algorithm is a method for solving both constrained and unconstrained optimisation problems that are based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions.

> At each step, the genetic algorithm... [performs a selection of] individuals... [to generate] the next generation. Over successive generations, the population "evolves" toward an optimal solution.

GAs are a stochastic global search method, Chipperfield *et al.* (1994) and population-based optimisation approach. Typically, a GA will produce an initial solution, whereafter it will be refined and distilled to eventually produce the final, optimised solution. Commonly, a GA consists of a population, some evolutionary analogous operators and a fitness evaluation that determines the "goodness" of the solution. A typical GA is initiated with a population of random approximated solutions to a specific problem or design query. These solutions are evaluated and ranked in order of best to worst, whereafter a selection process is performed to generate the next generation of approximate solutions. The process is repeated until a certain termination criterion is met and the final solution is produced.

Several selection heuristics have been developed, the most common of which is crossover (recombination) and mutation, Chipperfield *et al.* (1994). Other heuristics include elitism and randomised generation. Crossover is similar to traditional breeding, where an *offspring* solution(s) is produced as a result of the combination of two *parent* solutions. Several distinct criteria, to determine the composition of the offspring solution(s), have been developed. The simplest form is single-point crossover, wherein the offspring solution(s) is composed of some percentage of the first parent's composition and the remaining percentage of the second parent's composition. Consider the following binary strings:

$ParentA = 01011010$
$ParentB = 01000001$

Two offspring can be produced by combining the first 4 characters of Parent A and the remaining 4 characters from Parent B, for the first offspring, and vice-versa for the second offspring as shown:

$OffspringA = 01010001$
$OffspringB = 01001010$

Therefore the point of crossover was the third character and therein the single-point crossover. Naturally, there are multiple ways in which to create new strings using Parent A and Parent B, so also with crossover.

Mutation involves randomly replacing a single piece of a selected individual to produce a new solution. This heuristic is commonly applied, with a low probability, to ensure diversity. Instinctively, if the mutation probability is increased, the algorithm would defer to a primitive randomised search at

some point.

Elitism involves selecting the best solution for each generation to exist un-altered in the subsequent generation. This ensures that the quality of solutions will either remain constant or increase. It also provides better solutions to the crossover and mutation operators. Randomised generation may be employed to ensure that a comprehensive search of the entire solution space is conducted. Randomised search may serve as an escape tool for applications with numerous local optima, solutions that are good but not the optimum.

A measure of the "goodness" of the solution, also called fitness, is required for every application of a GA. Fitness will always be context-specific and must be prescribed/determined before the GA can commence. Solutions are evalu-ated during each generation of the GA's execution using an objective function to characterise the performance of the solution in the problem domain.

Particle Swarm Optimization

PSO is another population-based optimisation approach based on the simula-tion of social behaviour, Shi and Eberhart (1999). PSO was first developed by Eberhart *et al.* (1996) who, instead of using evolutionary operators, imple-mented a social behaviour on a group of particles. Each particle in the swarm flies within the problem domain with a dynamically adjusted velocity based on its own experience and that of its neighbours. In comparison to the GA, a particle is regarded as an individual and the state of the swarm of particles at each time step are the "generations". Each particle is a regarded as a point within the n-dimensional problem domain with:

$$S = X_1, X_2, \cdots, X_m \tag{2.4}$$

the swarm of size m and X_i the ith particle. Also let:

$$I = 1, 2, \cdots, m, \qquad D = 1, 2, \cdots, n \tag{2.5}$$

denote the indices of each particle in the swarm and the coordinate directions respectively. t will denote the iteration counter of the algorithm. Each particle can then be described as,

$$X_i^{(t)} = \langle \boldsymbol{x}_i^{(t)}, \boldsymbol{v}_i^{(t)}, \boldsymbol{p}_i^{(t)}, \mathrm{NB}_i^{(t)} \rangle, \quad i \in I \tag{2.6}$$

where

$$\boldsymbol{x}_i^{(t)} = [x_{i1}^{(t)}, x_{i2}^{(t)}, \cdots, x_{in}^{(t)}]^T \tag{2.7}$$

defines the current position of the particle within the problem domain,

$$\boldsymbol{v}_i^{(t)} = [v_{i1}^{(t)}, v_{i2}^{(t)}, \cdots, v_{in}^{(t)}]^T \tag{2.8}$$

defines the velocity of the particle,

$$\boldsymbol{p}_i^{(t)} = [p_{i1}^{(t)}, p_{i2}^{(t)}, \cdots, p_{in}^{(t)}]^T \tag{2.9}$$

is the best position thus far recorded for the particle within the problem domain i.e. the best solution found, in the problem domain, by the particle up until iteration t as defined by,

$$\boldsymbol{p}_i^{(t)} = \arg \min_{\{\boldsymbol{x}_i^{(k)},\ k \leq t\}} f(\boldsymbol{x}_i^{(k)}) \tag{2.10}$$

and,

$$\mathrm{NB}_i^{(t)} \subseteq I, \tag{2.11}$$

the *neighbourhood* of the particle.

Optimised solutions are obtained by biasing the velocity $(\boldsymbol{v}_i^{(t)})$ of the particle, using firstly $\boldsymbol{p}_i^{(t)}$. This biases the particle towards spaces within the problem domain that are promising. Each particle also exchanges information with its neighbourhood NB such that,

$$\boldsymbol{p}_{g_i}^{(t)} = \arg \min_{\{\boldsymbol{p}_j^{(t)},\ j \in NB_i^{(t)}\}} f(\boldsymbol{p}_j^{(t)}) \tag{2.12}$$

is used as the second biasing mechanism for the particles velocity.

For each iteration, each particle is updated as follows:

$$\begin{aligned}
v_{ij}^{(t+1)} &= \chi v_{ij}^{(t)} + \mathcal{C}_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \mathcal{C}_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \\
x_{ij}^{(t+1)} &= x_{ij}^{(t)} + v_{ij}^{(t+1)}
\end{aligned} \tag{2.13}$$

with $i \in I$ and $j \in D$. χ is the constriction coefficient or the inertia weight which is implemented to overcome the *swarm explosion effect* wherein the velocity of the particles would grow arbitrarily large. $\mathcal{C}_1$ and $\mathcal{C}_2$ follow continuous

uniform distributions based on user defined parameters c_1 and c_2, also known as the *acceleration constants*, as follows,

$$\mathcal{C}_1 \sim \mathcal{U}(0, c_1), \qquad \mathcal{C}_2 \sim \mathcal{U}(0, c_2) \tag{2.14}$$

The *acceleration constants* are implemented to control the acceleration magnitude for $\boldsymbol{p}_i^{(t)}$ and $\boldsymbol{p}_{g_i}^{(t)}$ respectively.

As with the GA, the PSO executes until reaching a predefined stopping criteria. What constitutes a promising space of the problem domain is also determined using a fitness function similar to the GA.

Simulated Annealing

Simulated annealing is analogous to metal annealing and based on the way that particles within the metal crystallise, Brooks and Morgan (1995). Annealing is an iterative process and can be described as follows. Consider a set of decreasing temperatures starting from some maximum T_0. At each temperature T, thermal equilibrium is reached and the probability of being in a state with energy E is given by the Boltzmann equation,

$$P(E = k) = \frac{1}{Z(T)} exp(-\frac{k}{k_b T}) \tag{2.15}$$

with $Z(T)$ a normalisation function and k_b the Boltzmann constant. As the temperature decreases, high energy states become less frequent and the range of the distribution focuses on states with lower energy. When the temperature reaches a low enough value, the system is said to "freeze" and the state with the minimum energy is reached. The Metropolis criterion (Metropolis *et al.* (1953)) is incorporated to influence the probability of remaining in a state as,

$$exp\{-\frac{(E - E_0)}{k_b T}\} \tag{2.16}$$

where E_0 is the energy of the current state and E is the energy of the new state. The Metropolis criterion is only implemented if $E \geq E_0$. If $E < E_0$ the system moves to the new state immediately.

Simulated annealing can be implemented in optimisation using the following general methodology:

1. Start at an initial state (T_0) with initial objective function parameters and the calculated initial function value (E).

2. Randomly select function parameters within a predefined neighbourhood of the initial state and calculate the function value (E_{new}).

3. Compare the function values. Let $\triangle E$ denote the difference between E_{new} and E.

4. If $E_{new} \geq E$ the Metropolis criterion is used to move to the new position E_{new} if and only if a uniformly distributed (over $[0, 1]$), random value U satisfies $U \leq exp(-\dfrac{\delta}{T})$ or equivalently $E_{new} \leq E_{old} - T log U$. If $E_{new} < E$ the system is moved regardless.

5. Steps 2 - 4 are repeated until equilibrium is reached at state T. The state is then lowered to a new temperature according to a annealing schedule. Steps 2-4 are once again repeated until equilibrium is reached.

6. Steps 2 - 5 are repeated until convergence is reached based on a predefined stopping criteria.

Curve Fitting

Curve fitting is another form of optimisation and concerns the process of determining the parameters of a function that has the best fit to a specific set of data points, that is to say, a curve that represents the data points in the most accurate manner as illustrated in Figure 2.8. Formally, curve fitting can be defined as minimising the distance measure of a given set of data points $\mathcal{D} = \{\vec{x_p}\}, p \in [1, P]$ and a general curve or surface $\mathcal{Z}(f)$ or:

$$\frac{1}{P} \sum_{p=1}^{P} dist(\vec{x_p}, \mathcal{Z}(f)) \rightarrow Minimum$$

Where the distance from the point $\vec{x_p}$ to the curve or surface $\mathcal{Z}(f)$ is defined as:

$$dist(\vec{x_p}, \mathcal{Z}(f)) = min\{\| \vec{x_p} - \vec{x_t} \| : f(\vec{x_p}) = 0\}$$

Another formalisation can be expressed as finding the function parameter vector $\boldsymbol{a}$ for a function,

$$f(x) = a_0 + a_1 x + a_2 x^1 + \cdots + a_n x^m \tag{2.17}$$

such that the residual vector e is minimised. The residual vector e $(n \times 1)$ is the difference between the data points of the fitted model,

$$f(x)_{fitted} = a_{0fitted} + a_{1fitted} x + a_{2fitted} x^1 + \cdots + a_{nfitted} x^m \tag{2.18}$$

and the original dataset:

$$
\begin{bmatrix}
x_1 & y_1 \\
x_2 & y_2 \\
x_3 & y_3 \\
\vdots & \vdots \\
x_n & y_n
\end{bmatrix}
-
\begin{bmatrix}
x_1 & y_{fitted_1} \\
x_2 & y_{fitted_2} \\
x_3 & y_{fitted_3} \\
\vdots & \vdots \\
x_n & y_{fitted_n}
\end{bmatrix}
\tag{2.19}
$$

The difference between these vectors can be calculated using various distance measures including Manhattan distance, chessboard distance and euclidean distance. Historically, using the euclidean distance as the measurement heuristic for curve fitting was unpractical due to the extreme computational cost, Faber and Fisher (2001). Approximations were developed at the cost of accuracy. Modern systems, however, can accommodate the computational overhead of using the euclidean distance and benefits from the accuracy and robustness it provides.
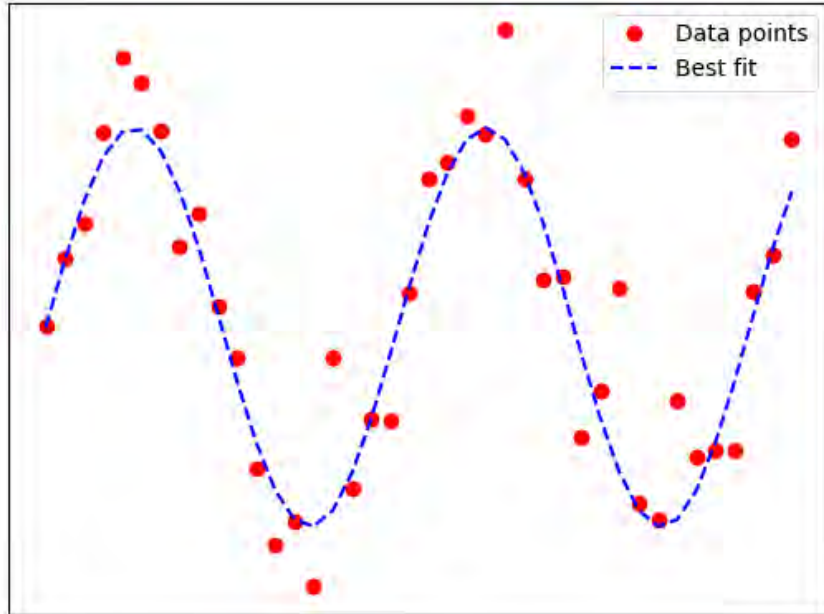

Figure 2.8: Curve fitting concerns determining the parameters of the curve (blue) that best describe the trend of the data (red).

Machine Learning Tools

Several machine learning tools are used for optimisation including Linear Regression, Decision Trees, Random Forests, Support Vector Machines (SVMs) and artificial neural networks (ANNs).

Linear Regression entails linear curve fitting to a dataset. Machine learning in general can be viewed as curve fitting where algorithms are trained to fit datasets to specified curves in order to enable them to predict unknown data points. Decision Trees are tree-like decision models that are based on conditional statements, if this then that. Random Forests are a combination of several Decision Trees who's outputs are average to determine the final output of the algorithm.

SVMs can formally be denoted as follows, given a set of N data points $\{x_k, y_k\}_{k=1}^{N}$, where $x_k \in \mathbb{R}^n$ and $y_k \in \mathbb{R}$ is the k-th data point, an SVM will estimate the function parameters as follows,

$$f(x) = \sum_{k=1}^{N} \alpha_k y_k \psi\left(x, x_k\right) + b \tag{2.20}$$

where α_k and b are positive real constants (Vapnik (2013)). $\psi(x, x_k)$ is the residual vector e and is dependant of the type of SVM as follows:

- $\psi(x, x_k) = x_k^T x$ for a linear SVM

- $\psi(x, x_k) = (x_k^T x + 1)^d$ for a polynomial SVM of degree d

- $\psi(x, x_k) = exp\{- \parallel x - x_k \parallel_2^2 /\sigma^2\}$ for a radial basis SVM ($\sigma = $ constant)

- $\psi(x, x_k) = tanh[\kappa x_k^T x + \theta]$ for a two layer neural SVM ($\kappa, \theta = $ constant)

ANNs are a form of machine learning that utilise a network of interconnected units, sometimes referred to as neurons, to perform computations using some form of logic, Gron (2017). Originally developed to mimic the workings of the brain, ANNs have a similar structure to that of the biological neural network, see Figure 2.9. ANNs are at the core of most reinforcement learning methods.

Reinforcement learning concerns the computational application of learning through interaction, Sutton and Barto (2018). Commonly, the subject to be taught, referred to as the learner or agent, must develop an understanding of its circumstances or environment and possible actions through a trial-and-error approach as illustrated in Figure 2.10. Typically, actions will have consequences in the form of penalties or rewards and it is the task of the learner to discover the action or action sequence that maximises the reward.
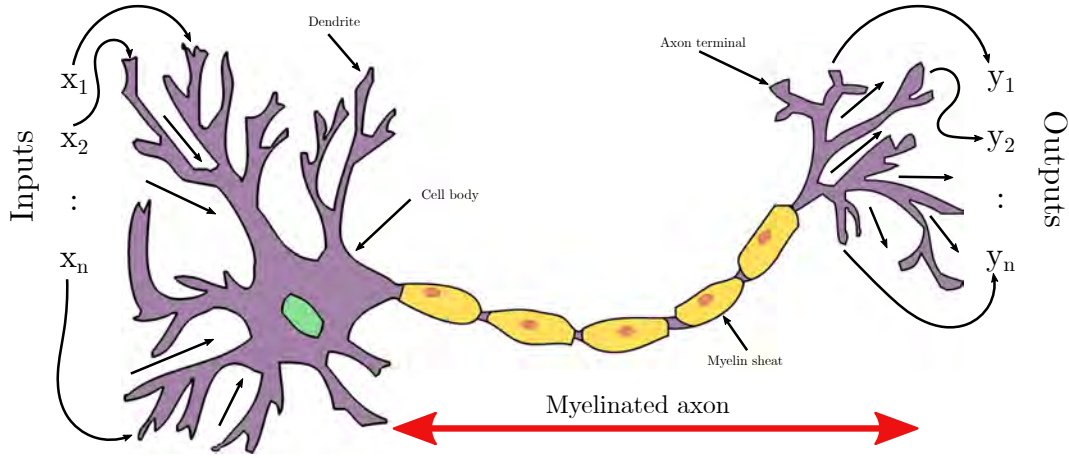
Figure 2.9: Originally developed to mimic biological neural networks, artificial neural networks function similarly producing outputs $y_1, y_2, \ldots, y_m$ from inputs $x_1, x_2, \ldots, x_m$ and logical computations.

Consider a chess player. When executing a move, the player must take into account the position of all the pieces on the board i.e the environment or state of the board and the effect or consequence that this move will have on both his own pieces and the pieces of it's opponent. If the move were to place his king in a compromising position, the consequence or "reward" of the move could be considered negative. Contrarily, if the move resulted in a victory or advantageous position, the "reward" is considered positive. The board is also placed into a new state as a result of the move. Through this process, a chess player can learn which moves deliver positive results and which moves deliver negative results and so improve their game.

This technique has been employed in soft robotics, typically in the application of control system development Gupta *et al.* (2016), Zhang *et al.* (2017), Thuruthel *et al.* (2019). The prominent methods of reinforcement learning is value-based and policy-based learning.

A value-based agent learns the reward maximising action sequence for its environment. It iteratively builds up a table of possible actions for each state within the environment and, through trial-and-error, learns the most optimal action sequence (Moni). It can be approximated as a randomised search through all the possible actions. Value-based learning can also be coupled with deep learning for environments with larger state spaces. The policy of an agent is the function that determines which decision the agent will make in a certain situation. It maps the environment states to the possible actions (Moni). Policy-based learning is the approach used when an agent learns the parameters of the policy function that maximises the rewards.

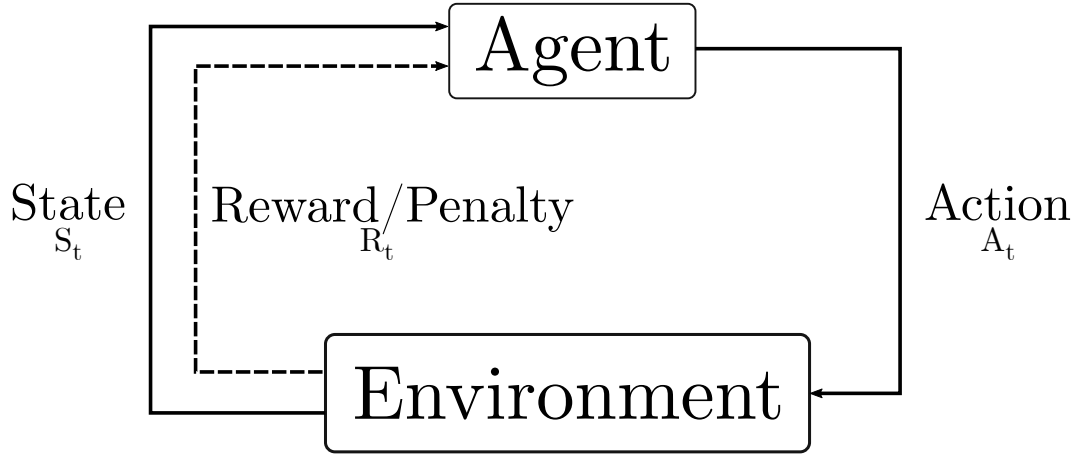A hybrid form of reinforcement learning is the actor-critic model. Here,

Figure 2.10: The learner or agent must develop an understanding of its circumstances or environment by taking some action A_t for a reward or penalty R_t that changes the environment or agent in some way S_t

both methods are employed in the form of a critic, which evaluates the action that was taken (value-based) and an actor, which learns the optimal policy function (policy-based). Simply put, the actor performs an action for which it receives feedback from the critic and updates its decision making appropriately. The critic also learns to provide better feedback.

Reinforcement learning can be employed using numerous forms of neural networks (Lin (1993), Coulom (2002)) including deep neural networks (Schmidhuber (2015), Neapolitan and Neapolitan (2018)), convolutional neural networks (Rusu *et al.*, Nagabandi *et al.*, Gu *et al.*), and recurrent neural networks, Hunt *et al.* (1992), Mayer *et al.* (2008).

Generative design has been coupled with machine learning to generate 2D floorplans (Ruiz-Montiel *et al.* (2013)), inverse molecular designs (Sanchez-Lengeling and Aspuru-Guzik (2018)) and art, Fischer and Herr (2001).

2.4   Summary

A generative design method for structural design applications demands a comprehensive, efficient and optimised algorithm to provide a suitable replacement for traditional FE and design methods. This method must be efficient, computationally inexpensive, generalisable and accurate.

Shape grammars are a specialised type of formal grammar for generative design. To sufficiently encode a design domain using only shapes requires a shape approximation for all objects in the design space. This becomes in-

creasingly difficult for structural applications wherein designs often consist of complex shapes and curves. Conversely the more general L-systems provide the means of encoding whole objects using single characters in the alphabet.

Genetic algorithms are inherently suited to optimization problems with categorical design variables, Fogel (1997). Contrarily, the original PSO algorithm is unsuited to categorical design problems as it requires continuous solution values in order to calculate the velocities of the particle, Strasser *et al.* (2016). Simulated annealing is computationally expensive and requires more iterations for convergence in comparison to other methods (Cisilino and Sensale (2002), Shan and Shuxia (2008)).

Machine learning tools were not chosen for the scope of this research as their implementation would necessitate a deeper understanding of the field. An erroneous implementation could lead to inaccurate results and thus machine learning was deemed outside the scope of this research.

As a result of this literature review, the following methods and techniques will be utilised in an attempt to deliver a generative design method for engineering applications. Firstly, reduced-order modelling will be adopted to reduce the complexity and dimensionality of the application. This will be employed as an optimisation technique in addition to an encoding tool to facilitate the development of a generative design algorithm. Secondly, Lindenmayer systems will be employed as a generative design method. Finally, a genetic algorithm will be utilised to optimise the generated designs for a specific form or function.

Curve fitting will be used to establish a proxy to evaluate the effectiveness of the developed soft robotic bending actuator designs in grabbing or grasping an object. For this purpose, the curve fitting will assist the soft robotic bending actuator in conforming to a designated "grasping" curve around the object.

Chapter 3

Generative Design with L-Systems: Preliminary Evaluation

To develop an understanding of the capabilities of L-systems and the capacity of generative design, a preliminary evaluation is undertaken. This evaluation will not directly answer the research question, however, it will provide a better understanding of the operation and properties of L-systems. In this regard, a Monte Carlo Method (MCM) will be employed to generate a large number of designs providing adequate data for analysis.

3.1   Problem Definition

This preliminary evaluation will involve the development of algae, the first application of Lindenmayer's original L-system. Here the word algae is used to describe the aquatic, autotrophic group of photosynthetic eukaryotic organisms widely known as the most common cause of pond scum. Figure 3.1 depicts three common microscopic growth patterns of freshwater brown algae.

A reduced-order model for the algae will be developed and utilised. This model will be simplistic, disregarding most biological properties of the algae except the instinctive need to acquire sustenance. This behaviour will be the basis of determining the performance or fitness of the generated algae. This evaluation will regard only static conditions and behaviours. The algae will be incapable of moving and not subject to gravitational or dynamic forces. The algae will be evaluated solely on its efficiency in gathering sustenance, based on its growth processes.

After developing a reduced-order model, an L-system will be designed for
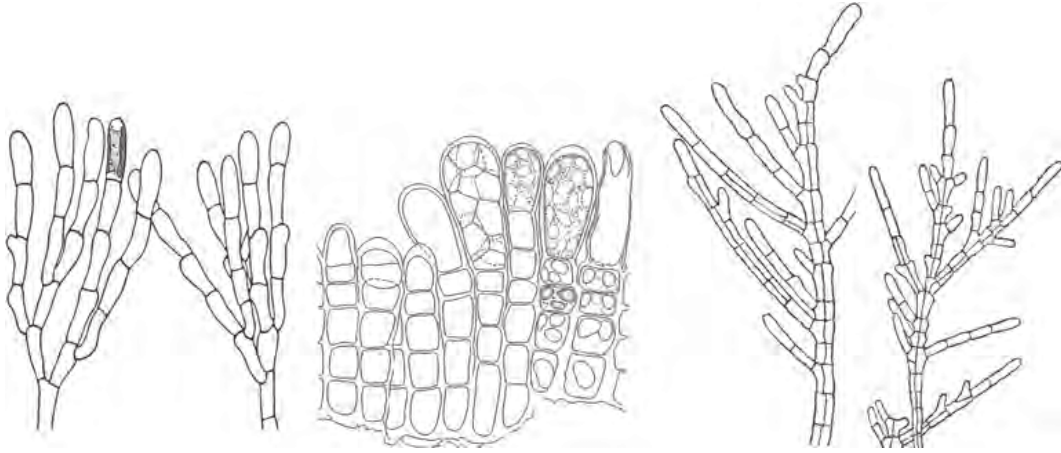
Figure 3.1: The three common microscopic growth patterns of freshwater
brown algae, Wehr (2015)

the algae, based on the reduced-order model. This L-system will be used to
generate the designs for the MCM.

The algae will require sustenance or nutrition to survive. The algae will be
capable of attaining sustenance by absorbing nutrients from the surrounding
environment. The entire organism will possess this ability. Consequently, ef-
fective nutrient absorption will be reduced if the surrounding environment is
occupied by multiple algae extremities. Overlapping extremities will also not
benefit doubly; instead, the nutrients will be divided equally between them.
Therefore, it will be regarded as an inefficiency if any extremities are within
each other's absorption range. These preliminary evaluations will regard the
entire environment as being uniformly and indefinitely dispersed with nutri-
ents. Therefore, the evaluation of the algae will solely be based on the overall
absorption area.

3.2 Design Methodology

For these evaluations, the algae will be simplified by disregarding the biological
factors that influence its growth structures. Therefore, it will be approximated
as single line segments, connected to form the overall body of the algae. Each
line segment will possess a specified absorption zone radiating outward, per-
pendicular to the line segment. This approximation is shown in Figure 3.2,
wherein the red lines represent the body of the algae and the blue zone the
absorption zone. Overlapping segments of the blue area will reduce the effec-
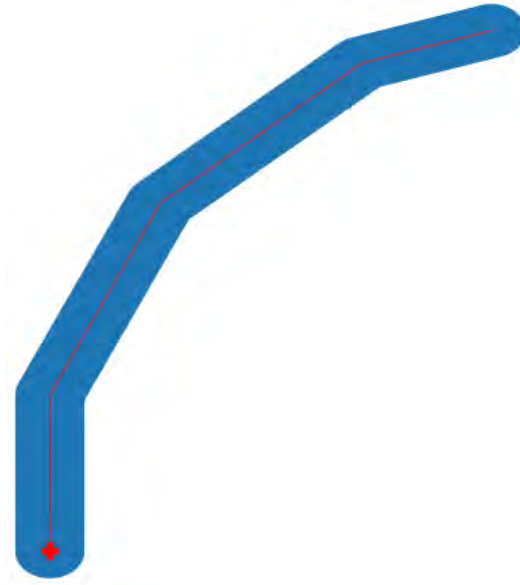tive absorption capacity of the algae.

Figure 3.2: A simple example of an algae L-creature illustrating the body of
the creature (in red), and the absorption range of the creature, (in blue). The
red cross denotes the origin of the algae

The objective of this evaluation is not to determine the optimal algae form
but rather to provide some insight into the characteristics and properties of
algae morphologies when using L-systems and reduced-order modelling.

Eight distinct scenarios will be regarded during this evaluation. Each sce-
nario will be a unique combination of each of the following specific instances
of the L-system. Two distinct groups are created by differentiating between
creatures that are capable of branching growth and those who are not. Branch-
ing allows the algae to develop forms that exhibit bifurcating growth i.e., the
algae grows in a tree-like shape. This advantage allows the algae to increase
its area of absorption whilst minimising the overall footprint of the algae body.

Furthermore, both an unbounded and bounded growing environment will
be explored. In the bounded environment, the diagonal distance of the algae
form's bounding box will be incorporated into the evaluation of the fitness of
the algae. An inverse relationship between the absorption area and the diago-
nal distance will be developed where smaller distances with higher absorption
areas are considered more efficient. This heuristic should encourage the algae
to develop smaller forms with maximised absorption areas.

Each algae instance will be subject to a penalty based on the distance that
the nutrients must travel to reach the origin. This penalty should directly
influence the algae to develop more condensed forms and may produce some

novel geometries.

These criteria are applied to evaluate the susceptibility of L-systems to a constrained environment as will often be the case in generative design for structural design applications. The performance of the L-system algorithm in terms of runtime duration should also provide a preliminary finding on the efficacy of L-systems in the capacity of generative design generator. Table 3.1 shows the L-systems developed to model the algae.

Table 3.1: The developed L-system to model algae for both the non-branching and branching cases

Description	Non-branching Value	Branching Value
Variables	X	X
Constants	F, +, -, _	F, +, -, [,], _
Axiom	FX	FX
Rules	X → ***** (TBD)	X → ***** (TBD)
	X → ***** (TBD)	X → ***** (TBD)
Angle	Variable	Variable
Maximum allowable recursions	5	5

The character encodings for the algae evaluations are shown in Table 3.2.

Table 3.2: The L-system character encodings for the algae evaluations

Character	Description
F	Unit step
+	Clockwise angle change
-	Anti-clockwise angle change
_	Placeholder character

The rule length for these L-systems will be fixed to a 5 character string. This should allow sufficient intricacy without negatively impacting the core principal of L-systems, which is recursion. The L-system must be limited in such a way as to force recursion, otherwise the use of L-systems would be futile. The "[" and "]" characters provide the L-system with branching capabilities. The "[" character signifies a state save, wherein the algorithm preserves the current state and location of the development of the algae while the "]" character returns the leading growth point to the last state that was preserved
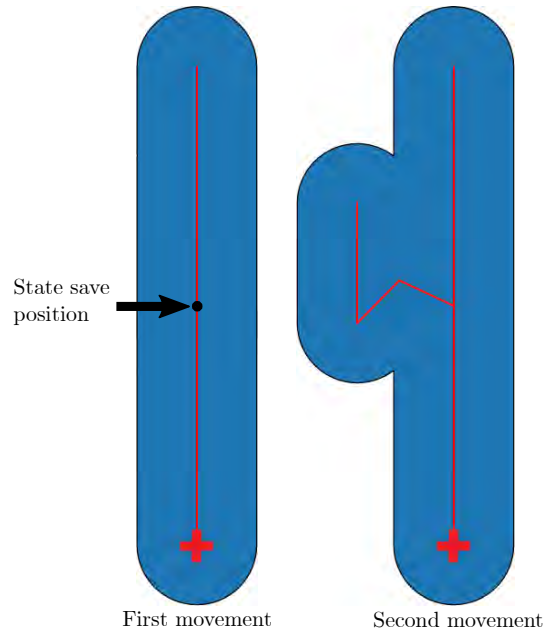
Figure 3.3: An example of a branching L-creature. During the first movement, the state of the growth of the algae is saved at the position indicated. Once the "|" character is encountered in the L-string, the leading growth point is returned to that position, makes an anti-clockwise angle change and continues growing.

resulting in branching growth as shown in Figure 3.3. The "_" character is meaningless in the L-system alphabet. It has no physical representation and will terminate algae growth if encountered within the L-string of the algae. Introducing this character into the L-system alphabet may produce interesting results due to its function as an early stopping mechanism.

In this L-system, two distinct rules will be generated for each variable, for each alga. To generate multiple distinct algae forms, the rules of the L-systems will not be prescribed. To gain a comprehensive understanding of L-systems and their capabilities, algae forms need to be generated across the entire design space. Only by conducting an exhaustive search can the ability of this particular L-system to model algae, be assessed. This exhaustive search will also provide the necessary insight to determine the capacity of L-systems to adapt to constrained environments. Allowing the rules to be generated at random for each alga, constitutes a randomised search, and in employing a MCM wherein an extremely large sample is generated, an exhaustive one.

The L-strings generated by these L-systems will be restricted to a maximum recursion allowance of five times, meaning that the L-string will be developed over five sequential recursive operations, wherein each "X" character in the L-string will be replaced by one of the generated rules. Should an L-string no

longer contain any "X" characters at any point during the recursion process, that L-string will be considered final and no further recursions will apply.

To display the generated algae, the unit step parameter is advantageous in keeping the generated graphics modest, thereby allowing mediocre computational hardware to render these forms. It should be noted that this is purely an aesthetic restriction. If desired, the algae can be generated and evaluated computationally in any scale as these computations do not utilise the visual form of the algae, only the measured properties that can be gained without drawing the forms. Therefore these evaluations are assumed to be scale-invariant. The specific software packages used, namely Python 3.6.8 and Shapely 1.6.4 do, however, incur some computational penalty for increased scales. However, this penalty is only applicable to the software and not to the L-systems.

3.3 Results and Discussion

The eight scenarios that were considered is shown in Table 3.3. A sample size of 100 000 algae was generated. The distribution of algae absorption area for these evaluations is right-skewed as illustrated in Figure 3.4. This is a prevalent trend in evolutionary systems (Brown *et al.* (1993), McShea (1994)) and supports the validity of the methodology of this evaluation in terms of evolutionary design generation. The probability density for each sample size remains consistent. The maximum achievable absorption area value for these evaluations is 1365 as calculated in Figure 3.7 and the algae form for this value is present in the data. Considering these factors, the sample size was deemed adequate. Table 3.4 summarises the algae evaluations that were not penalised based on nutrient travel distance, as well as the fittest algae in each category. Conversely, Table 3.5 summarises the penalised evaluations.

Table 3.3: A summary of the scenarios that were evaluated

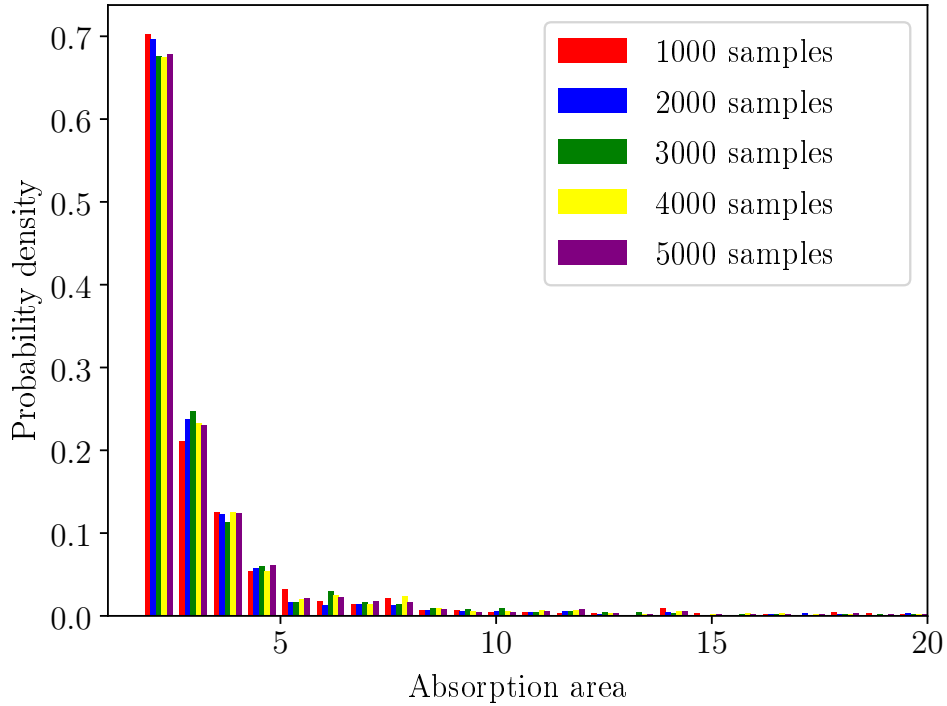| Scenario	Bounded	Branched	Penalized
1	No	No	No
2	Yes	No	No
3	No	Yes	No
4	Yes	Yes	No
5	No	No	Yes
6	Yes	No	Yes
7	No	Yes	Yes
8	Yes	Yes	Yes

Figure 3.4: A portion of the MCM data distribution illustrating the effect of sample size on the probability density of the data.

For scenario 1 the algae was allowed to grow unbounded, but was not provided branching capabilities. Logically, the fittest solution, as illustrated, is that of a single straight line. This configuration has no inefficiencies and maximises the absorption area of the algae. Several variations of this configuration also exhibit comparable performances, as shown in Figure 3.5, but are encumbered by slight angle variations, thereby, reducing the overall achievable absorption area of the algae due to the angle change characters occupying the algae's L-string.

The "straight line" configuration holds prominently in the results of the first scenario, however, some algae performed comparably without utilising this configuration, shown in Figure 3.6. This is attributed to the rewriting nature of L-systems. Consider three L-system rules of the algae evaluations, namely 'FFFFX', 'XXXXF' and 'FFFXX'. As shown in Figure 3.7, the L-system rule with the most 'F' characters, surprisingly, does not deliver an L-string with the same performance. Conversely, the L-system rule with the most 'X' characters or variables, delivered the L-string with the most 'F' characters. Since an L-string is developed through recursive replacement of the variable character with its corresponding rule, the L-system with the most variables in its

rules will deliver the longest L-string. This provides the L-system with the potential to develop longer L-strings with more influential characters such as the 'F' character.

Scenario 2 did not provide branching capabilities either, however, the algae were encouraged to develop more compact geometries. The algae with the highest absorption area would simply be a more compact form of the "straight line" configuration as shown. This coiling form maintains the overall absorption area of the "straight line" configuration i.e. the maximum achievable performance and tolerates the inefficiencies of overlap. Developing a bounded, non-overlapping algae, proved to be too complex for this particular L-system.

Scenarios 3 and 4 are similar to the first two, except for allowing branching of the algae forms. For scenario 3, this allowance did not have any influence, since the most optimal form is still the straight line. No restrictions were imposed upon this form and, therefore, no change was necessary. However, in scenario 4, branching was beneficial to develop algae that had less overlapping in comparison to the non-branching algae. As illustrated in Table 3.4, the branching property allowed the algae to grow in a rather bizarre form. Although unclear from the figure, this form has several branches propagating from the origin in various directions. This configuration is mirroring the coiling approach followed by the non-branching algae, however, it achieves this form more efficiently through simultaneous growth of extremities through branching.

When applying the nutrient transport penalty to these scenarios, only two results differ from the unpenalised cases. The alga in scenario 5 still achieves the highest performance, even when penalised, with the "straight line" configuration. Similarly, the alga in scenario 6 also maintains the form generated in the previous scenario. These configurations remain the optimal even under these constraints. The alga in scenario 7 follows a new approach by growing numerous extremities from its origin disregarding overlap merely attempting to cover as much area as possible, whilst minimising the penalty of nutrient transport. The alga in scenario 8 also grows numerous extremities from its origin, attempting to cover as much area as possible, whilst maintaining a small footprint in comparison with its unbounded counterpart.

Table 3.4: Representative results showing the most fit algae for each of the 4 unpenalised scenarios that were evaluated.
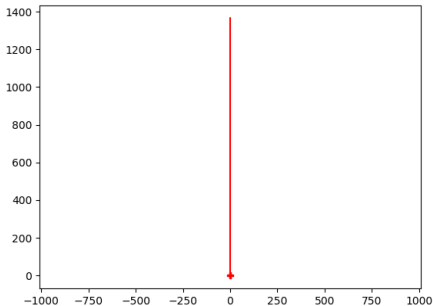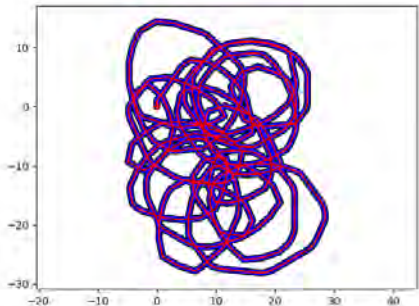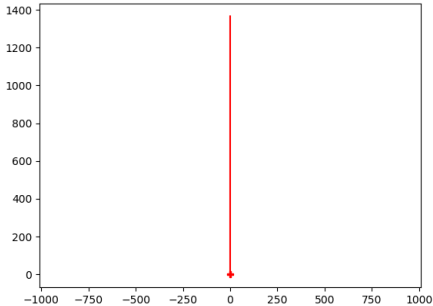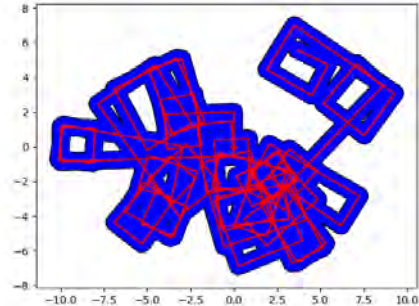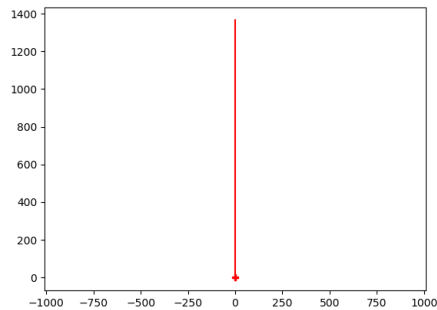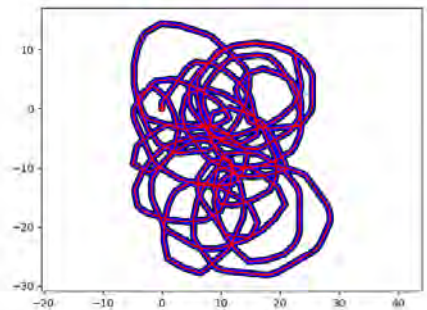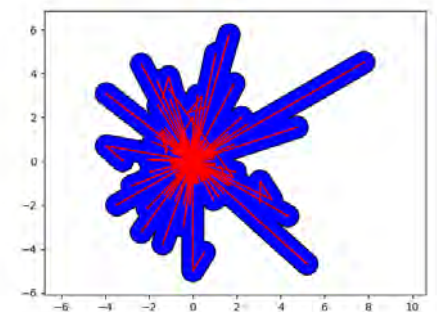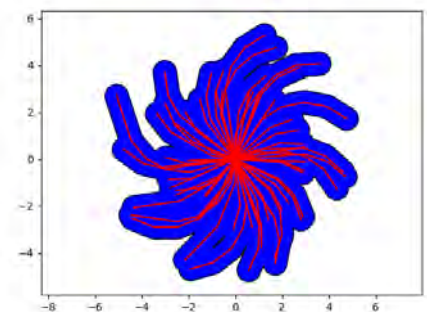
Environment type Creature type	Unbounded	Bounded
	Scenario 1	Scenario 2
Non-branching		
	Scenario 3	Scenario 4
Branching		

Table 3.5: Representative results showing the most fit algae for each of the 4 penalised scenarios that were evaluated.

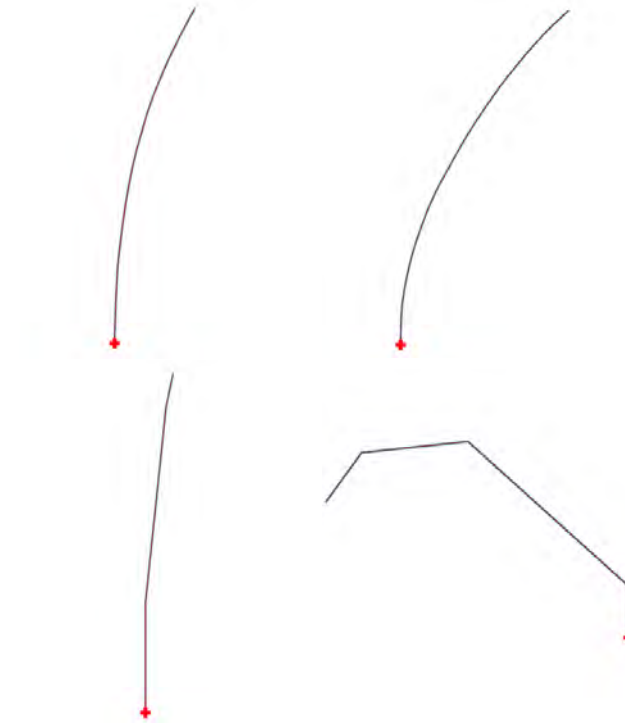| Environment type Creature type	Unbounded	Bounded
	Scenario 5	Scenario 6
Non-branching		
	Scenario 7	Scenario 8
Branching		

Figure 3.5: Several algae that possess comparable absorption areas to the most fit algae configuration.

These L-systems delivered intriguing results in accordance with the logical outcome of these evaluations. Firstly, the global optimum form ("straight line") was observed in the data. Therefore, the L-system is capable of optimisation without constraints. Secondly, the optimum configuration for each constrained scenario was also developed, exhibiting the capabilities of using L-system for constrained optimisation. These results corroborate the use of L-systems for generative design.
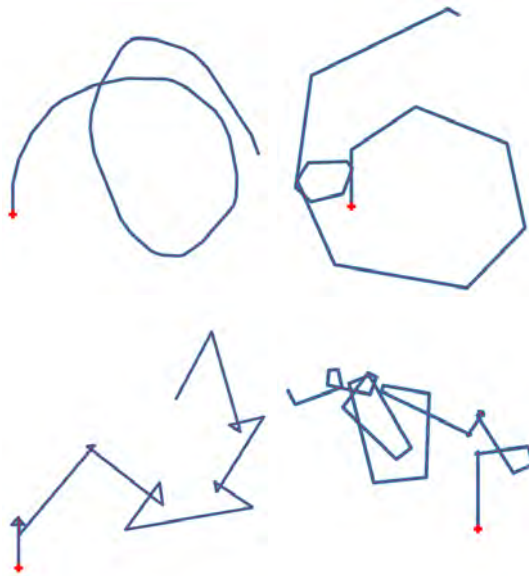
Figure 3.6: Algae that delivered comparable absorption areas to that of the "straight line" configuration algae
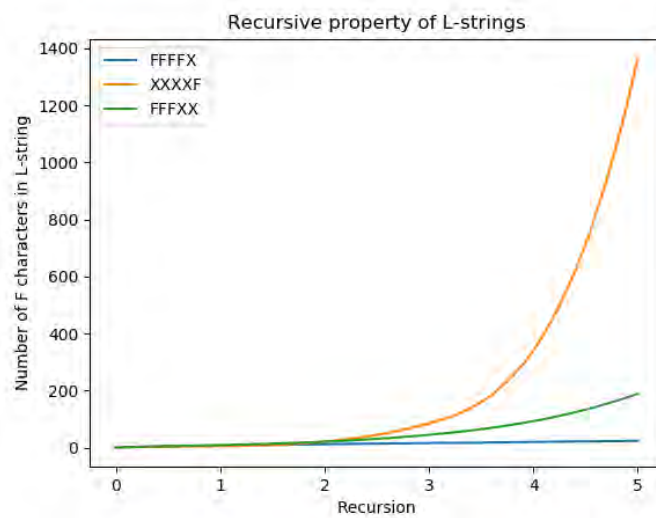

Figure 3.7: The L-string's 'F' character count for each rule after 5 recursions.

Chapter 4

Soft Robot Actuator Models

4.1   Introduction

To demonstrate the capabilities of the generative design technique developed herein for structural design applications, a generator will be developed to produce soft robotic bending actuator designs.

Chapter 2 identified the need to employ reduced-order modelling for generative design to increase the efficiency of the generator. Commonly, full-complexity models, which account for every aspect of the model during simulation i.e. material properties, dynamic properties, time-transient properties etc. produce excessive data, most of which may not be necessary for the generator's purposes, Ellis *et al.* (2019). Therefore, a reduced-order model will provide a simplified parametric model with fewer computational requirements, whilst maintaining the level of accuracy required by the generator to deliver viable designs. These reduced-order models are mostly context-specific and should be developed on a case-to-case basis.

Reduced-order modelling for soft robotic bending actuators are most commonly employed when developing a control system for the soft robot (Thieffry *et al.* (2018*a*), Thieffry *et al.* (2018*b*), Katzschmann *et al.* (2019)), to circumvent the complexity of material behaviour modelling. A reduced-order model must aim to capture these behaviours accurately, to correlate with experimental data, whilst minimising the computational resources needed to evaluate a model. Herein, reduced-order modelling is utilised to reduce the dimensionality of the design domain thus decreasing the computational requirements for the evaluations of the soft robotic actuator designs resulting in increased design iterations allowing for a more thorough design process.

Soft robotic bending actuators commonly achieve bending either through incorporation of a strain-limiting layer or by utilising an asymmetrical cross-

section to produce a preferential bending direction. Incorporation of a strain-limiting layer can be achieved by embedding a layer of different material, with less elasticity, within the soft robotic bending actuator. An asymmetrical cross-section can be achieved by increasing the volume of a layer of the soft robotic bending actuator, thereby creating the asymmetry discussed above. This method is commonly known as eccentric void asymmetry. Another method for asymmetry utilises corrugated membranes which expand more easily under pressure by adding folds on one side of the actuator cavity. To achieve the intricate actuator shapes necessary for this research, the strain-limiting layer must be extensively varied from segment to segment. Embedding a different material within the actuator becomes increasingly difficult. Therefore, an actuator geometry will be used that employs asymmetry to achieve bending.

A reduced-order model for the soft robotic bending actuator can be developed by quantifying the geometric and material changes that the actuator undergoes at a certain pressure. These parameters can then be incorporated into a centerline representation of the soft robotic bending actuator. The reduced-order model therefore does not account for the time-transient properties of the actuator during inflation but only considers the final inflated state of the actuator. The centerline representation can thus be viewed as a spline with piecewise polynomials consisting of the angle and length changes that the actuator undergoes.

4.2   Reduced-Order Modelling for Soft Robotic Actuators

The soft robotic bending actuator used in this research was developed by Ellis *et al.* (2019), who utilised a modular construction of fluid actuated soft material units to construct the actuator. Ellis endeavoured to create a computational tool for designing modular soft robotic bending actuators. By implementing optimisation and reduced-order modelling, he decreased the amount of iterations necessary to develop soft robotic actuators and decreased the evaluation runtime of the actuators by approximately 92% in comparison to the non-reduced-order model. The reduced-order model correlated closely with the fabricated actuator and delivered promising results. Ellis was also capable of developing targeted actuators that conform to specific forms or curves.

The reduced-order model for this research, is developed to accurately represent the 2D model of Ellis' fabricated, pneumatically actuated, soft robotic bending actuator. The actuator is fabricated from two different silicones namely Mold Star 15 and Smooth-Sil 950, Inc.. The modular approach fol-

lowed, allowed the development of various actuator configurations simply by changing the orientation of each segment with regards to its strain limiting layer. Figure 4.1a illustrates a single modular unit in the un-actuated and actuated states.
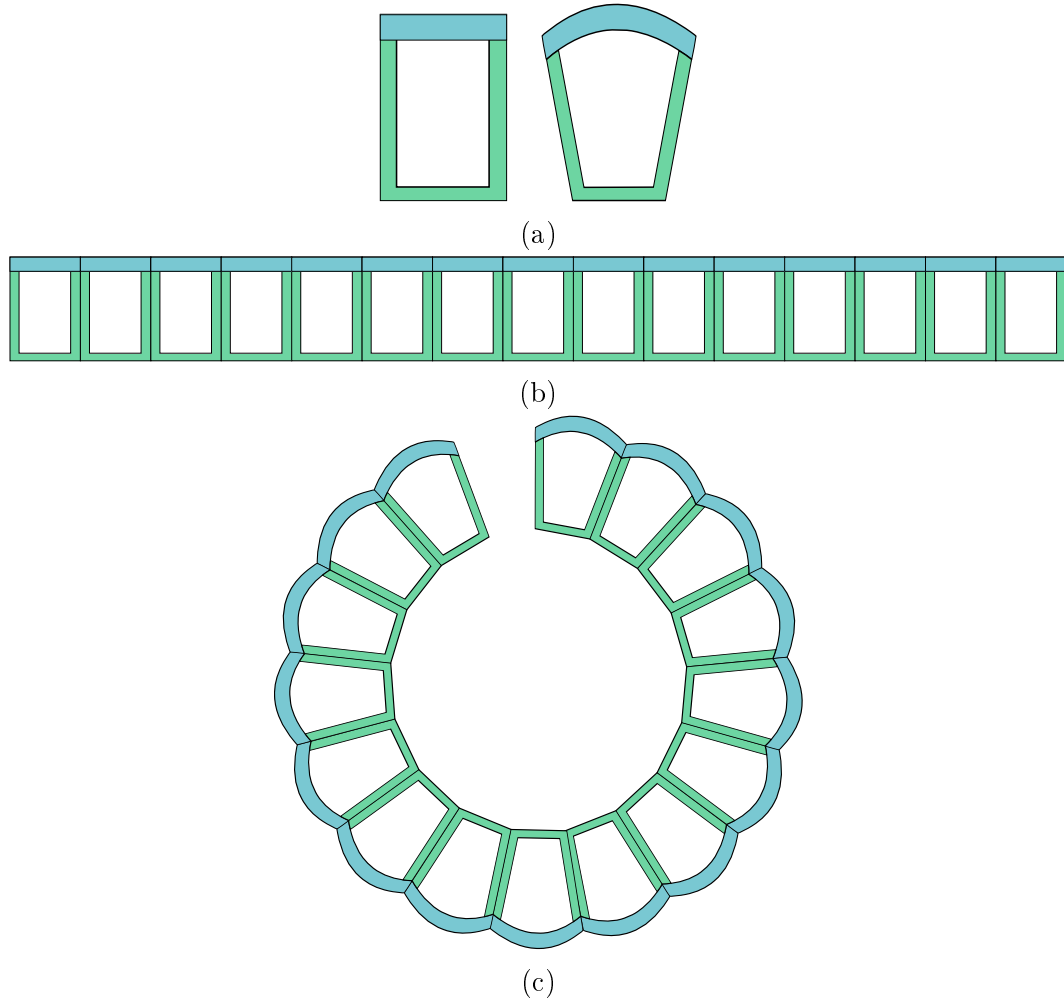

(a)

(b)

(c)

Figure 4.1: a) Cross-sectional view of a single parametric actuator module, fabricated from Mold Star 15 (green) and Smooth-Sil 950 (blue) shown with the strain limiting layer located on top of the module. b) Soft robotic bending actuator constructed from a series arrangement of actuator modules in an un-actuated state. c) Soft robotic bending actuator in the actuated state.

By combining these modular units in series, a soft robotic bending actuator can be developed, see Figure 4.1b. The orientation of each module can be varied between the top and bottom wall of the module, altering the preferential bending direction of that module. The orientation vector is defined as the vector denoting the position of the strain limiting layer and subsequently the

Table 4.1: Actuator shape objectives used for comparison between the developed models and the fabricated actuator

Target	Objective function
Maximum horizontal displacement	maximise X
Minimum horizontal displacement	minimise X
Maximum vertical displacement	maximise Y
Minimum radius of curvature	min $\sqrt{(X_f - X_i)^2 + (Y_f - Y_i)^2}$
Fit to Sin curve profile	min $e = a_{fitted}sin(\theta_{fitted}) + q_{fitted} - y$
Fit to Cos curve profile	min $e = a_{fitted}cos(\theta_{fitted}) + q_{fitted} - y$

preferential bending direction i.e. Up and Down. This vector will be denoted as follows $[\uparrow, \downarrow ...]$.

4.3   Development of Reduced-Order Model for Soft Robotic Bending Actuator

Using a GA to develop actuator shapes through optimisation of the orientation vector, several comparison objectives were executed as shown in Table 4.1. The objective as specified in the *Target* column applies to the tip position of the actuator. Ellis used the same objectives and therefore provides a comparison for the method developed herein.

The displacements and geometric transformation of each modular unit during actuation can be quantified. Through superficial measurements of Ellis' work, these values were determined as an 80% increase in horizontal centreline length and an angle change of 11° for each sidewall of the module as shown in Figure 4.2. These measurements will form the basis of the reduced-order model. As illustrated in the preliminary simulations in Chapter 3, an L-system may be represented as a line segment wherein the line segment undergoes the geometric transformation applicable to that extremity. A soft robotic bending actuator may also be approximated as a composition of line segments, where each segment represents a modular parametric unit. The length increase and angle adjustment can then be applied to the line segment to accurately represent a modular unit during the actuated phase.

A reduced-order model is developed for these soft robotic bending actuators in the form of a line segment composition as shown in Figure 4.3. Each line segment has a length increase similar to the measurements and an equivalent angle change to accurately reflect the geometric transformation of the

actuator segments.

In order to compare with previous research, a 15-segment actuator will be used. The actuator in Figure 4.3 possesses the following orientation vector $[\uparrow,\downarrow,\downarrow,\uparrow,\uparrow,\downarrow,\downarrow,\uparrow,\uparrow,\downarrow,\downarrow,\uparrow,\uparrow,\downarrow,\downarrow]$ indicating the position of the strain limiting layer for each segment. The orientation vector, therefore, becomes the object of optimisation to deliver specific shaped actuators. To that end, as with Ellis, a GA is employed to obtain actuator shapes for the same objectives as listed in Table 4.1.
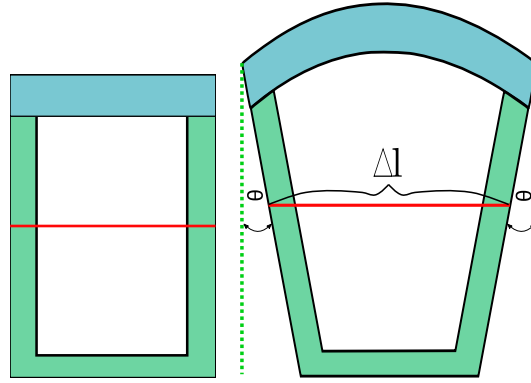

Figure 4.2: Quantified parametric actuator module illustrating the measurements taken to develop a reduced order model. $\triangle l$ is the length increase of the horizontal centreline and θ is the angle change applicable to the segment side walls.
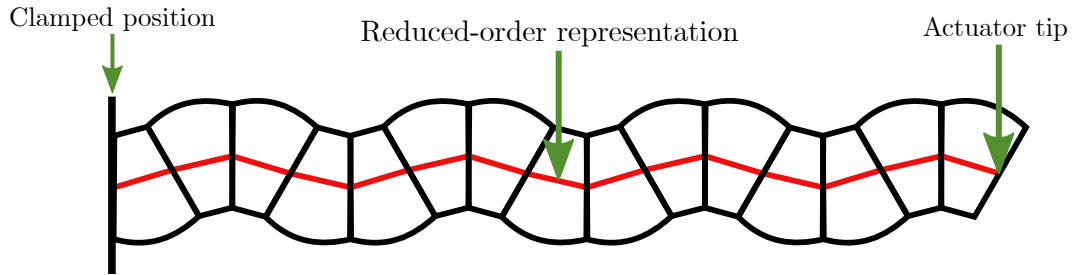

Figure 4.3: A reduced-order model (red) for the actuated soft robotic bending actuators (black) is illustrated as a continuous line segment from the clamped position to the tip of the actuator.

4.4 Results and Comparison to Previous Research

The optimisation of the orientation vector to deliver targeted actuator shapes is executed using a GA. This process is illustrated using a data flow diagram as shown in Figure 4.4. Each of the objectives in Table 4.1 were targeted and the optimal actuators were developed. The actuators generated using the reduced-order model correlated well with the non-reduced-order model of the fabricated actuator as illustrated in Figures 4.5, 4.6, 4.7 and 4.8.

For this evaluation, the termination criteria of the GA was based on stagnation. If no improved solution was found after a certain amount of iterations, the algorithm would terminate.

Actuators were also developed to conform to desired profiles specifically a Sin and Cos curve. Curve fitting was employed to optimise the shape of the actuator to fit the profile. The design is optimized to minimise the distance between the coordinates of the points of each line segment and the corresponding data points of a Sin or Cos function. This approach is discussed in more detail in Chapter 5. The actuators generated for these scenarios also correlate well with the non-reduced-order model of the fabricated actuator as shown in Figures 4.9 and 4.10.

The actuators developed herein achieved each target objective with improved correlation in comparison with previous research as illustrated in Figure 4.11 and Table 4.2. Table 4.2 illustrates the displacement of the tip of the soft robotic bending actuator from the relevant axis. The fabricated actuator's displacement is shown in the *Actual* column while the *Previous* column represents the displacement of the designs developed in previous research. The *Results* column represents the displacement of the soft robotic actuator tip for the designs developed in this research. This table represents the accuracy of the developed designs in comparison with the actual fabricated actuator.

Figure 4.11 illustrates the comparison between the obtained profiles of the modelled fabricated actuator (green), the results obtained in previous research (red and blue) and the results obtained in this research (grey). In some cases the desired profile is shown (black). The average evaluation runtime for each actuator design was reduced from approximately 45 seconds to 6.8 microseconds.

Table 4.2: The actuators developed using the reduced-order model show improved accuracy compared to previous research. The *Actual* column represents the fabricated actuator tip position, *Previous* is the previous research results and *Results* is this research's results

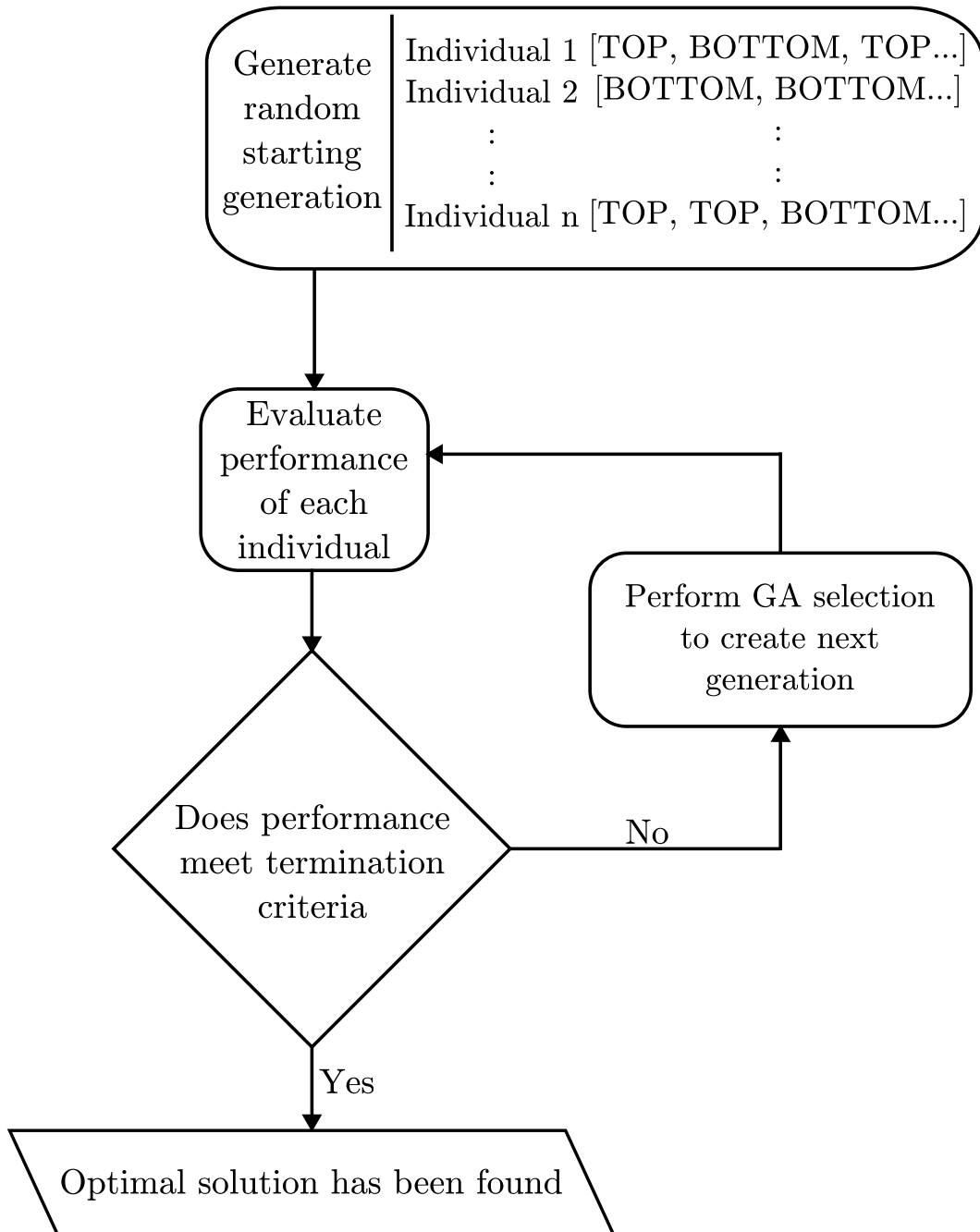| Target	Actual [mm]	Previous [mm]	Results [mm]
Maximum horizontal displacement	225.2	232.11	221.95
Minimum horizontal displacement	-101.22	-99.87	-97.77
Maximum vertical displacement	204.64	211.16	201.4
Minimum radius of curvature	10.39	30.81	16.51

Figure 4.4: The dataflow diagram for the optimisation of the reduced-order soft robotic bending actuators to deliver targeted shapes.
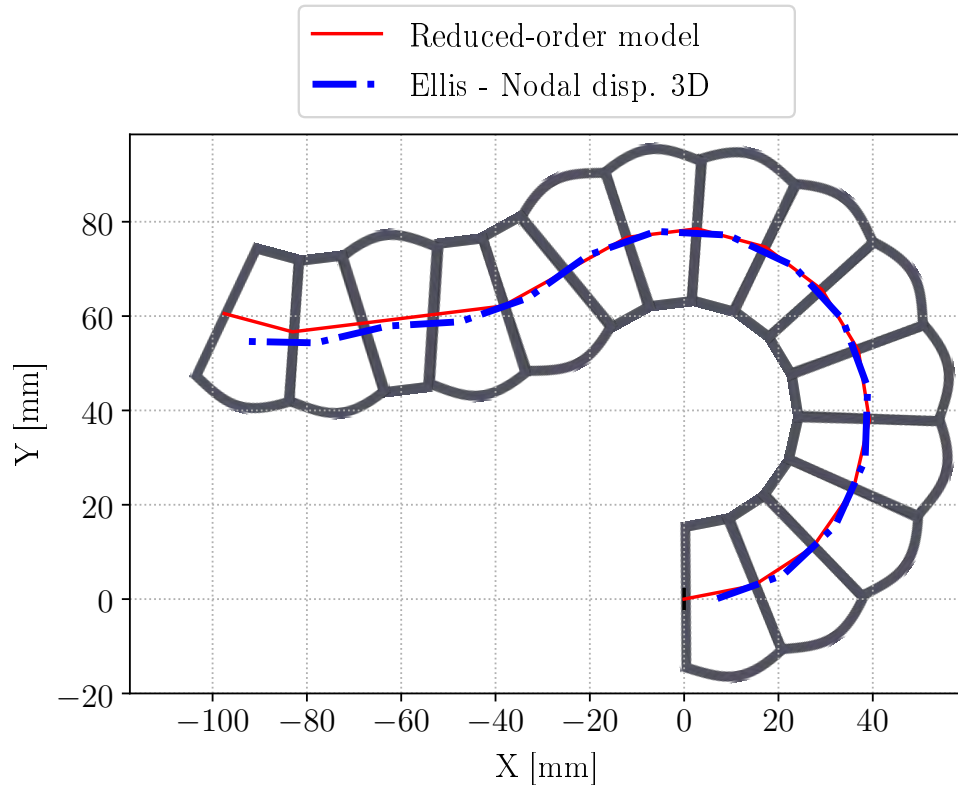
Figure 4.5: The reduced-order model (red) of the actuator design generated to minimise X correlates well with the fabricated actuator (blue)
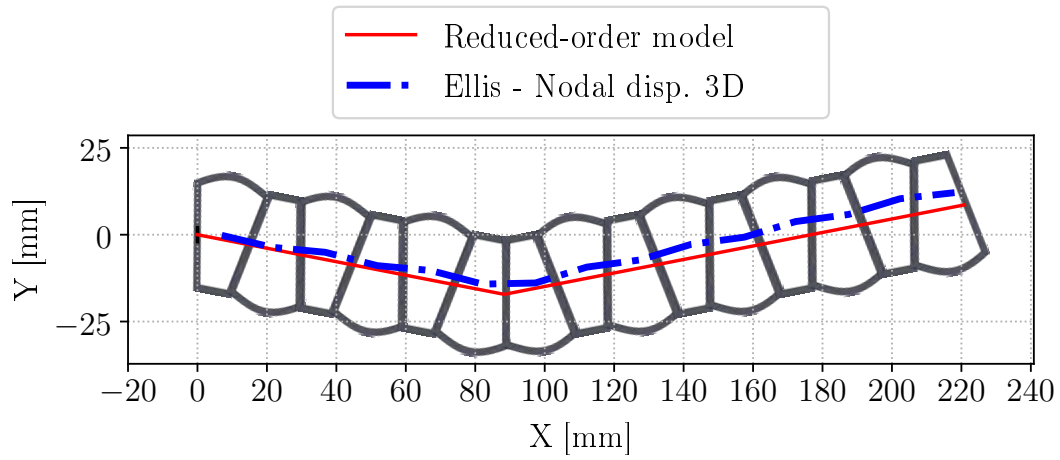

Figure 4.6: The reduced-order model (red) of the actuator design generated to maximise X correlates well with the fabricated actuator (blue)
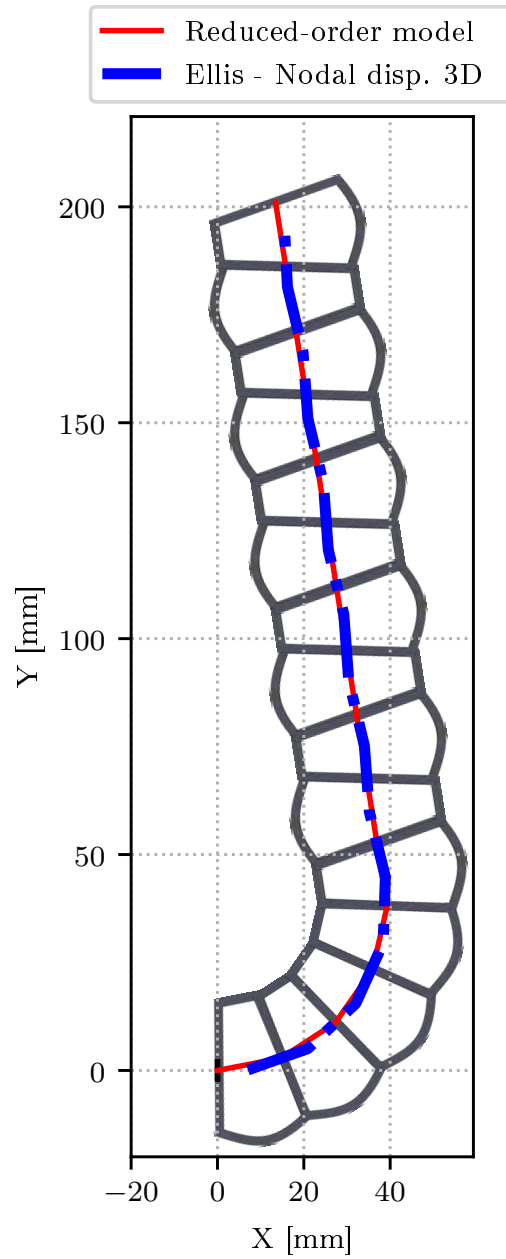
Figure 4.7: The reduced-order model (red) of the actuator design generated to maximise Y correlates well with the fabricated actuator (blue)

Figure 4.8: The reduced-order model (red) of the actuator design generated to minimise the radius of curvature correlates well with the fabricated actuator (blue)

Figure 4.9: The reduced-order model (red) of the actuator design generated to fit the Sin profile (black) correlates well with the fabricated actuator (blue)
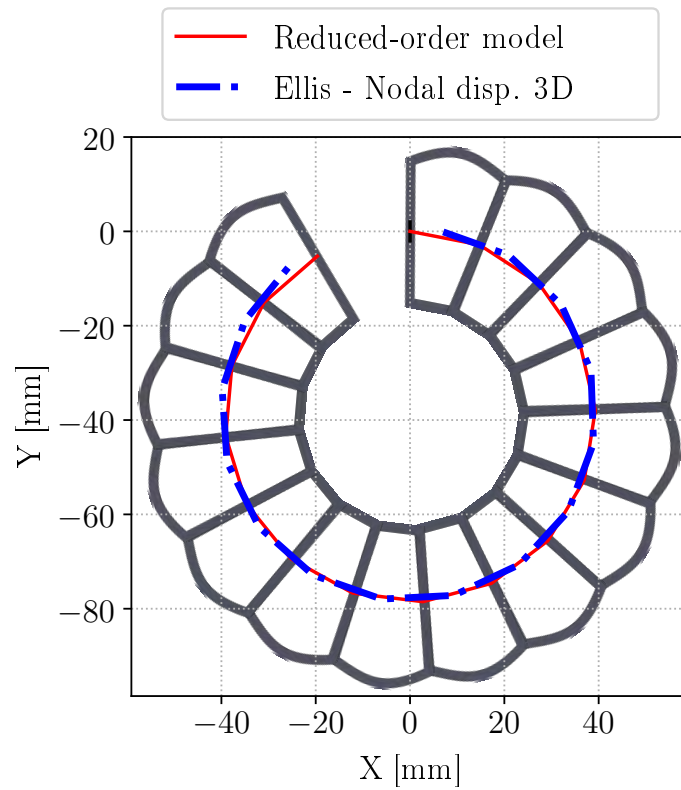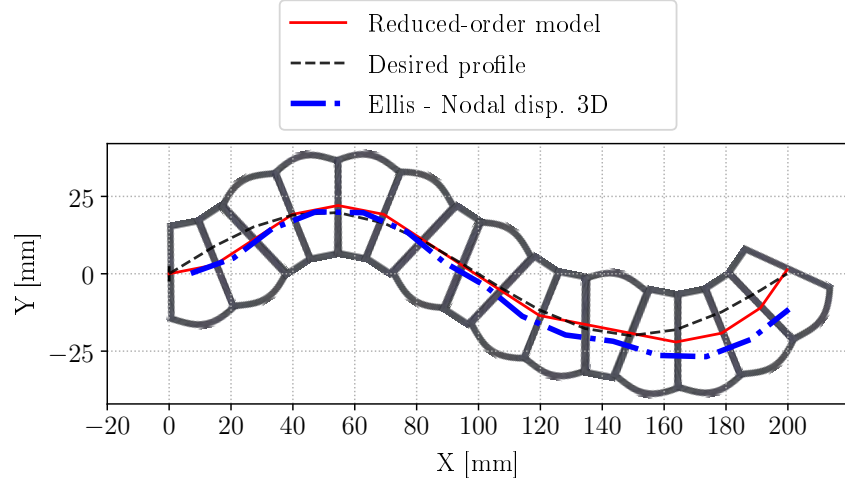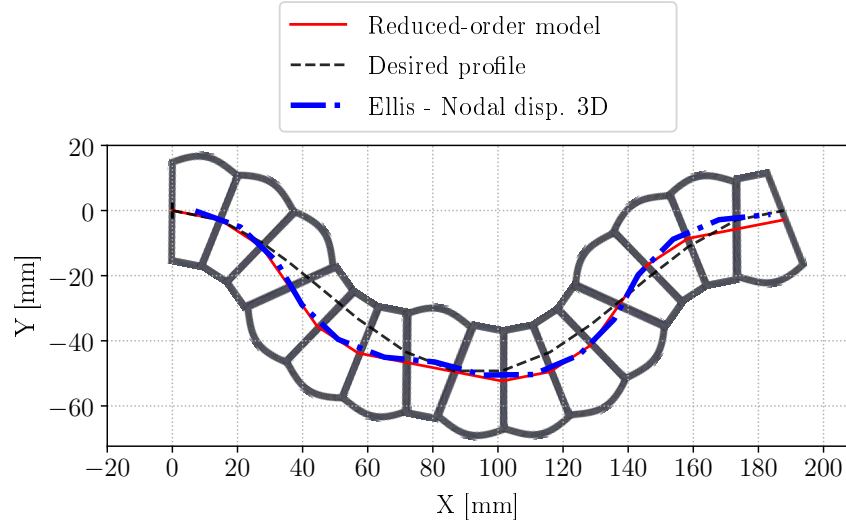

Figure 4.10: The reduced-order model (red) of the actuator design generated to to fit the Cos profile (black) correlates well with the fabricated actuator (blue)
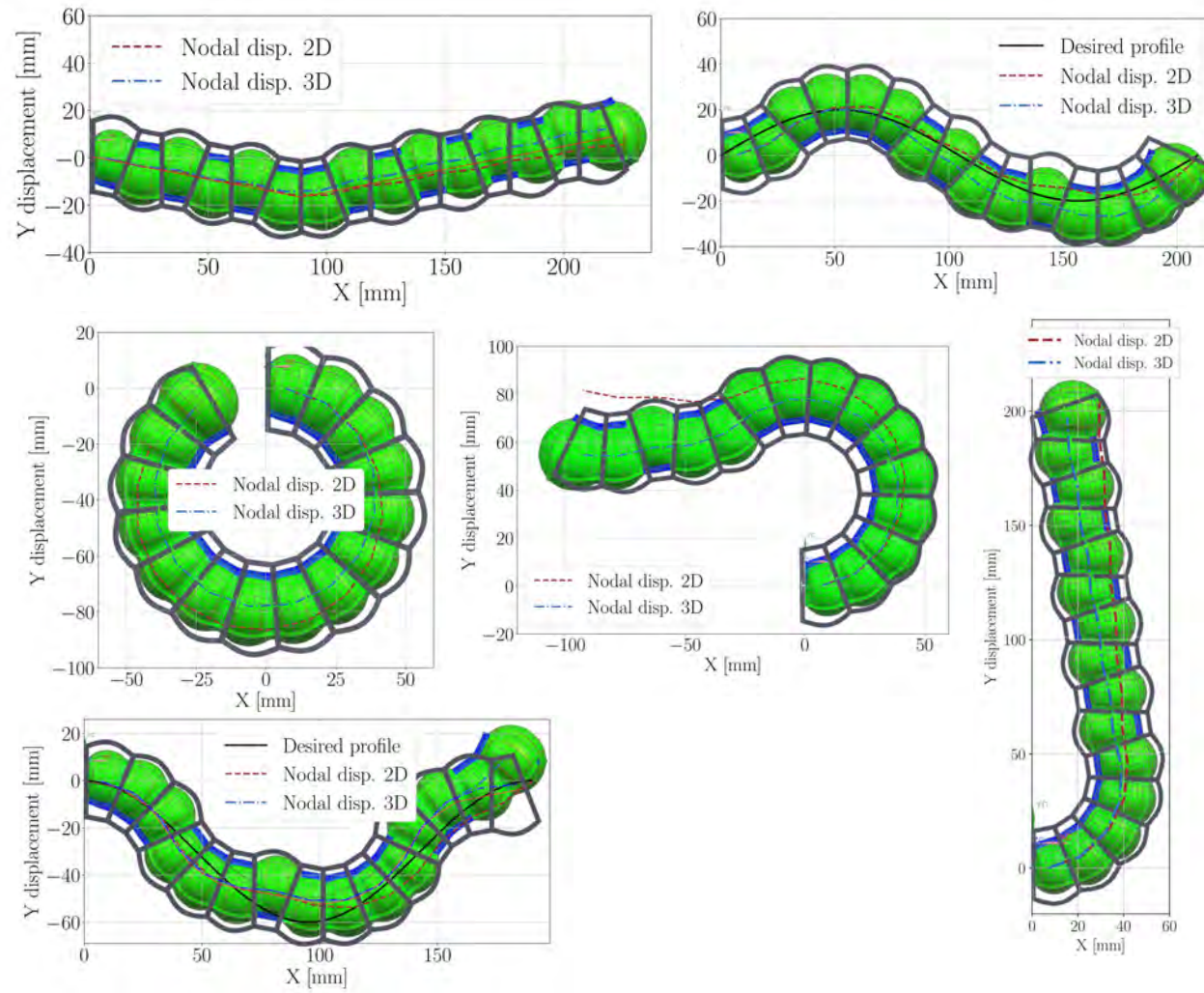
Figure 4.11: For each target objective the reduced-order model developed herein displayed increased accuracy to previous research

Chapter 5

Generating a New Bending Actuator Design

The technique developed in Chapter 4 possesses the capabilities to generate soft robotic bending actuator geometries for various targeted forms and curves with similar accuracies to FE methods. By optimising the orientation vector, the actuated shape of the actuator can be varied.

5.1 Generative Design Method for Soft Robotics

In order to employ L-systems to generate soft robotic bending actuator designs, an *Alphabet* must be created, consisting of *Variable* and *Constants* from which the L-system can generate rules.

Task decomposition is the practice of subdividing a complex task into several smaller, simpler tasks that can be learned and combined to achieve the more complex behaviour. To apply this approach here, the soft robotic bending actuator can be divided into several smaller sub-units, of two or three segments each, that form the core units of specific actuator functions such as bending, elongation and curving. For example, the actuator developed in Figure 4.8 can be summarised as some combination of two modular segments with bottom located strain limiting layers as illustrated in Figure 5.1.

These *sub-units* are selected by reducing several arbitrary designs to functionally similar core modules i.e. a curving design implements two modules with the same bending direction in conjunction therefore this constitutes a sub-unit. Sub-units are thus chosen to eliminate duplicate functionality. Having two sub-units that perform the same function would have decreased the diversity of the population. This forms part of the Pandemonium section of

ESP as discussed in Chapter 2.Therefore, an *Alphabet* for soft robotic bending actuators can be developed using this technique.
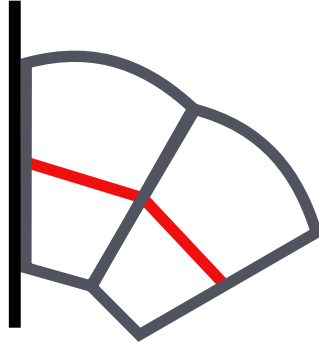
Figure 5.1: The core sub-unit for a bending actuator is constructed from two modular segments with bottom located strain limiting layers

This process involves generating high performance actuators for each actuator function, using the technique developed in Chapter 4, and reducing them to their most basic sub-unit, delivering a set as illustrated in Figure 5.2. Unique letters are then attributed to each sub-unit to develop the L-system alphabet.

Figure 5.2: The core sub-units for each actuator function generated by reducing high performance actuators to their most basic form

This L-system can then be employed similarly as in Chapter 3 to generate soft robotic bending actuator designs. These designs may be targeted by spec-

ifying a fitness evaluation as the aim of the GA optimisation. To demonstrate, soft robotic bending actuators will be developed to grasp objects in a 2 dimensional environment. As stated in Chapter 1, this will be a static simulation and will not incorporate dynamic aspects.

5.2   Problem Definition

To develop a suitable fitness metric for the GA, a proxy for actuator grasping must be developed. Considering an object in 2D space, grasping can be said to occur if the grasper envelops the object sufficiently. This envelopment can be represented by a curve denoting where the centreline of the grasper should lay to envelop the object, as shown in Figure 5.3. To this end, curve fitting was employed as, as shown in Figure 5.4. For this particular application the type of curve fitting is 1D continuous piecewise function curve fitting. These functions can be viewed as several discrete, continuous line segments, Jekel and Venter (2019), similar to the line segments of the reduced-order model for the soft robotic bending actuator. The coordinates where each line segment ends and the next one begins are called the breakpoints, as illustrated in Figure 5.5. Various techniques have been developed for piecewise function curve fitting (Muggeo (2003)) however the minimisation of the residual vector prevails.

Mathematically this problem can be formulated for a linear case as follows. Consider the ordered data points $x_1, x_2, ... x_n$ with dependant values of y such that $y(x)$:

$$
\begin{bmatrix}
x_1 & y_1 \\
x_2 & y_2 \\
x_3 & y_3 \\
\vdots & \vdots \\
x_n & y_n
\end{bmatrix}
\tag{5.1}
$$

with x_1 & y_1 being the first pair of data points. A piecewise linear function can thus be expressed as a set of linear functions for each data pair:

$$
\mathbf{y}(x) = \begin{cases}
\eta_1 + m_1 \left(x - b_1 \right) & b_1 < x \le b_2 \\
\eta_2 + m_2 \left(x - b_2 \right) & b_2 < x \le b_3 \\
\vdots & \vdots \\
\eta_{n_b-1} + m_{n_b-1} \left(x - b_{n_b-1} \right) & b_{n_b-1} < x \le b_{n_b}
\end{cases}
\tag{5.2}
$$

where b is the x locations of the n_b-number of breakpoints. Thus, there are $n_b - 1$ line segments. Equation 5.2 therefore represents a set of linear piecewise functions. When continuity is enforced over the line segments i.e. each line segment is connected to the previous line segment, each subsequent gradient
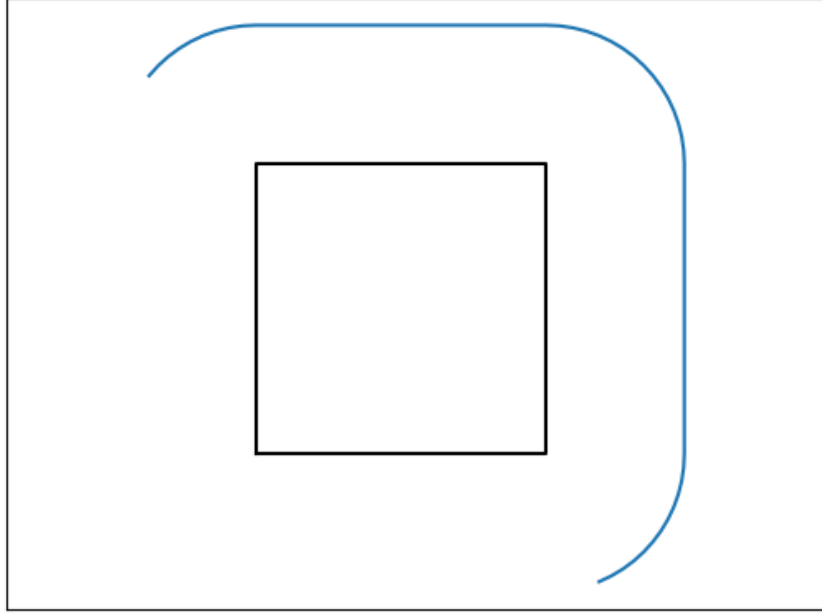
Figure 5.3: The curve (blue) that a grasper must follow to successfully grasp the object

becomes dependant upon the previous segments gradient. The set of linear piecewise functions are then reduced to:

$$
\mathbf{y}(x) = \begin{cases}
\beta_1 + \beta_2 (x - b_1) & b_1 \leq x \leq b_2 \\
\beta_1 + \beta_2 (x - b_1) + \beta_3 (x - b_2) & b_2 < x \leq b_3 \\
\vdots & \vdots \\
\beta_1 + \beta_2 (x - b_1) + \beta_3 (x - b_2) + \cdots + \beta_{n_b} (x - b_{n_b-1}) & b_{n-1} < x \leq b_{n_b}
\end{cases}
\tag{5.3}
$$

resulting in an equal number of model parameters β_{n_b} and breakpoints b_{n_b} reducing the problem to solving the set of linear equations:

$$
\mathbf{A}\beta = \mathbf{y}
\tag{5.4}
$$

where $\mathbf{A}$ is the $n \times n_b$ input data matrix $[x_n \times b_{n_b}]$, β is the $n_b \times 1$ vector of unknown parameters and $\mathbf{y}$ is the $n \times 1$ vector of y data points. The residual vector e $(n \times 1)$ can then be obtained as follows:

$$
e = \mathbf{A}\beta_{fitted} - \mathbf{y} \longrightarrow minimum
\tag{5.5}
$$

with
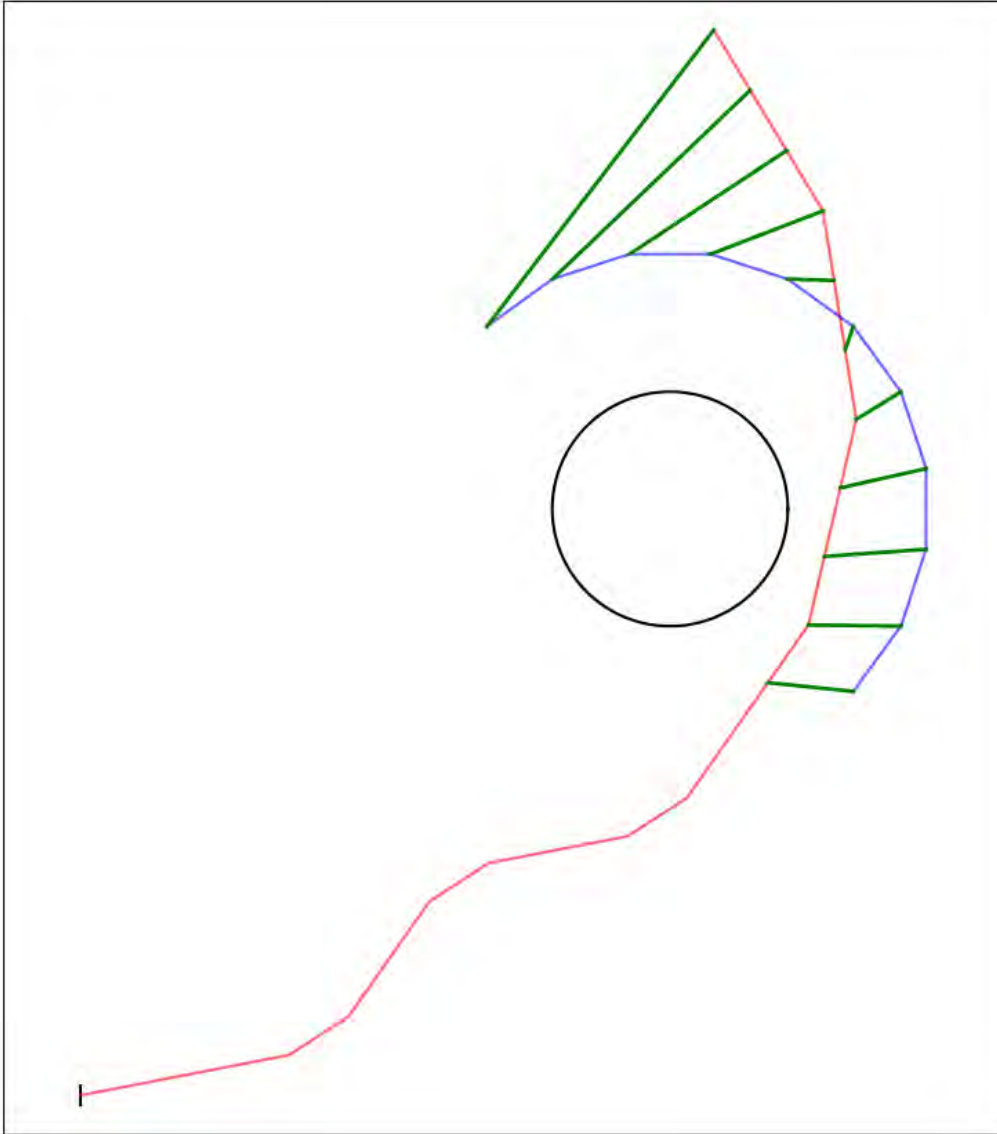
Figure 5.4: The reduced-order model (red) must minimise the distance (green) between the breakpoints and the curve (blue)

$$\beta_{fitted} = (\mathbf{A}_T\mathbf{A})^{-1}\mathbf{A}_T\mathbf{y} \tag{5.6}$$

For this research, the least-squares distance heuristic will be employed as follows:

$$\text{Sum-of-squares} = e^T e \tag{5.7}$$

The resulting positive scalar value will be utilised as the fitness of the actuator with the objective of the GA being to minimise this scalar value. An envelopment curve will be calculated for each object at each variation in scale.

Breakpoints

Figure 5.5: The breakpoints of the reduced-order model for the soft robotic bending actuator.

Actuator designs will then be generated and optimised to fit this curve.

5.3 Results and Discussion

The actuators were developed using the process depicted in Figure 5.6. Using similar methods as discussed in Chapters 3 and 4, grasping soft robotic bending actuator designs are generated using a combination of L-systems, reduced-order modelling and GA optimisation.
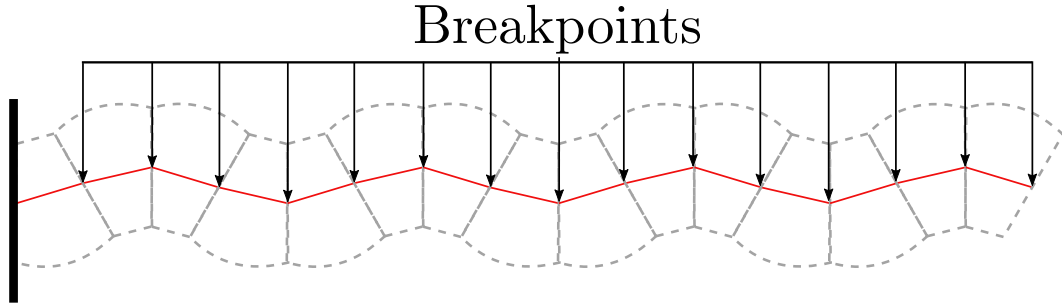
Firstly, a starting generation of actuators are created by generating random L-system rules for each actuator. These rules are comprised of the characters associated with the unique sub-units that were identified. Each actuator's L-string is then developed in 5 recursions. Once the L-string is fully developed and the actuator can be constructed, it is evaluated using the proxy as described above. The actuators are then ranked in ascending order based on the quality of the curve fitting and the GA performs selection to generate the subsequent generation.

The actuator that has the closest fit overall is always retained. At each generation, this actuator is used to evaluate the performance of the generation. If an actuator is developed that performs better, it is promoted to this position and the patience counter is reset. Otherwise the patience counter is incremented. Once the counter reached a prescribed maximum, the algorithm is terminated.

The method was successful in generating soft robotic bending actuators to grasp three uniquely shaped objects at three different scales. The resulting actuator designs are shown in Figure 5.7. The performance of the method for each of the nine cases is recorded in Table 5.1.
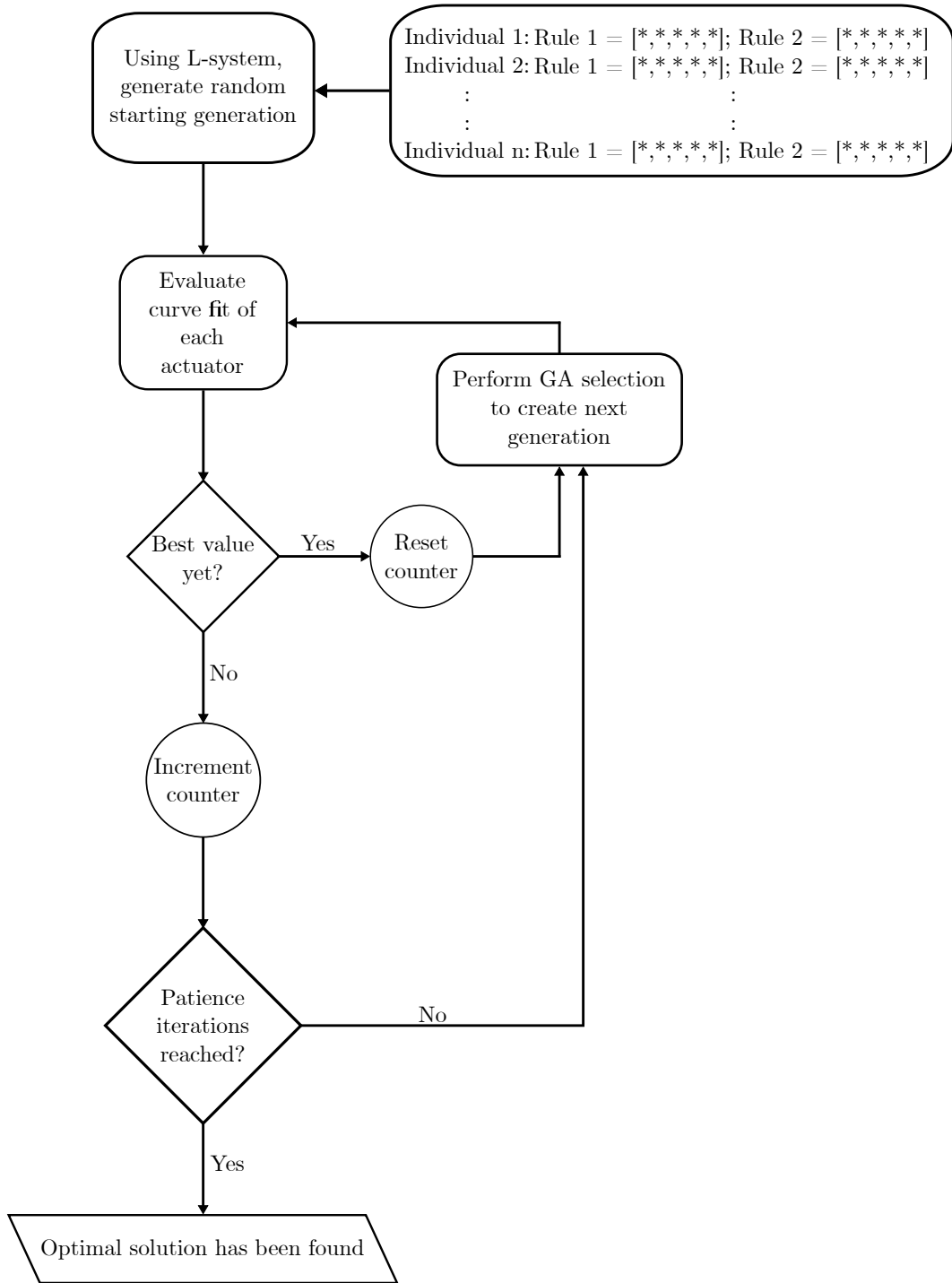
Figure 5.6: A data flow diagram for developing grasping soft robotic bending actuators.

Due to the reduced complexity of this method, generating designs for each of the different environments took, on average, just under three minutes. The longest design runtime was four minutes and eight seconds for the largest triangle size. These runtime reductions attest to the viability of the method to process increase design iterations allowing a more thorough design process.

The grasping proxy also performed suitably. For each scenario, the actuators were optimised to fit the curve of the object. This resulted in actuator shapes that enveloped the object to a sufficient degree to consider grasping successful.
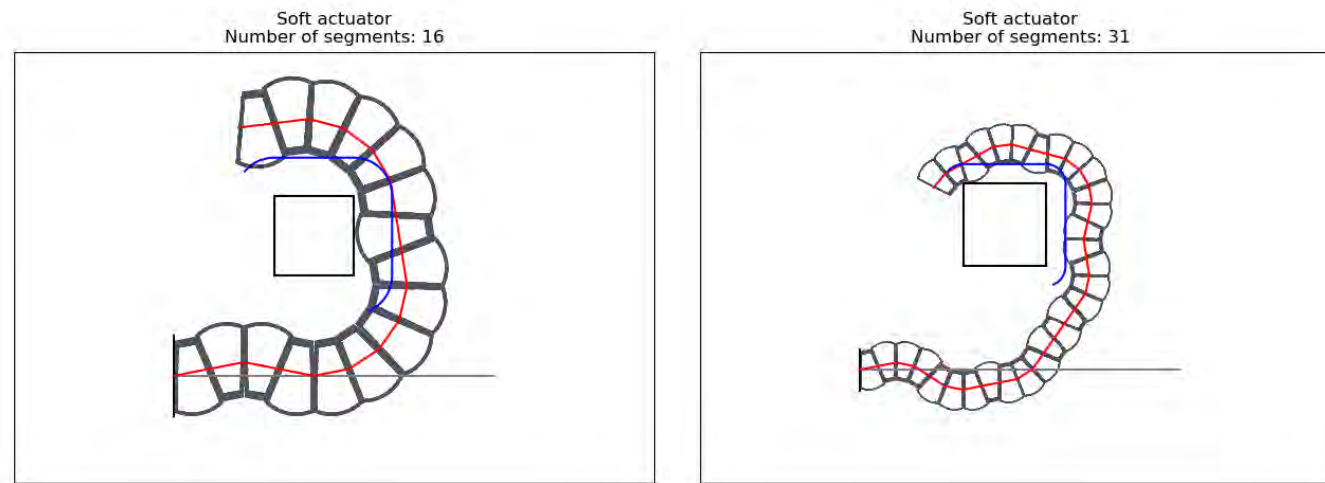
(a) The final actuator design for grasping a small square at a close distance.

(b) The final actuator design for grasping a medium square at an intermittent distance.
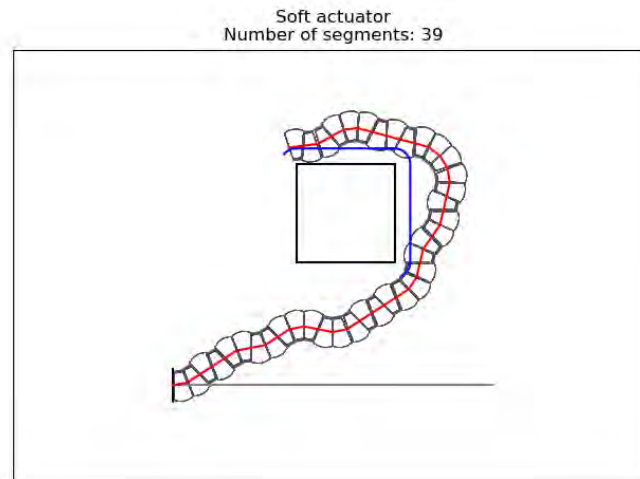
(c) The final actuator design for grasping a large square at a far distance.
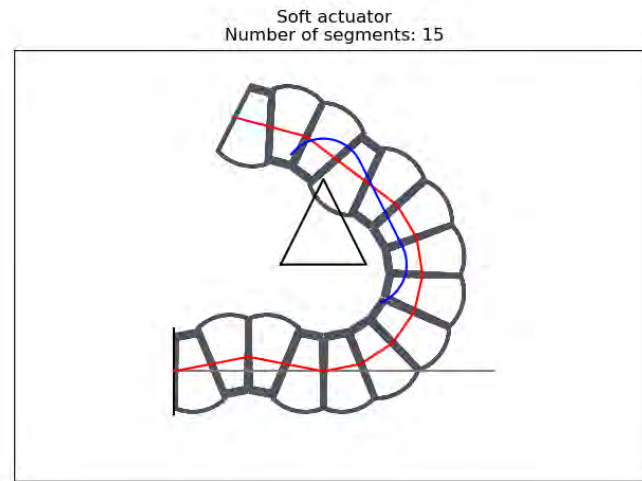
(d) The final actuator design for grasping a small triangle at a close distance.
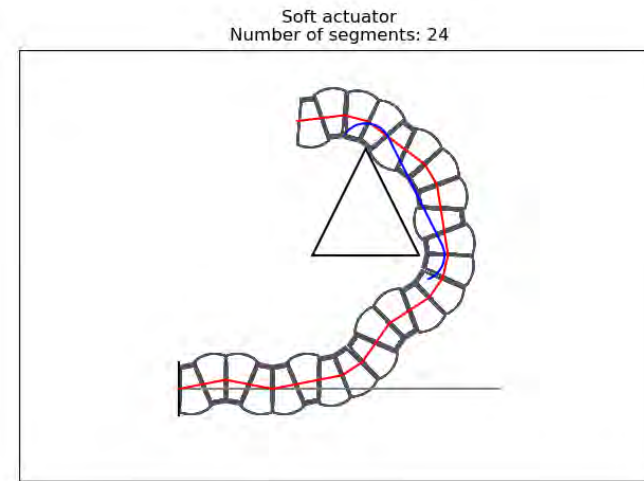
(e) The final actuator design for grasping a medium triangle at an intermittent distance.
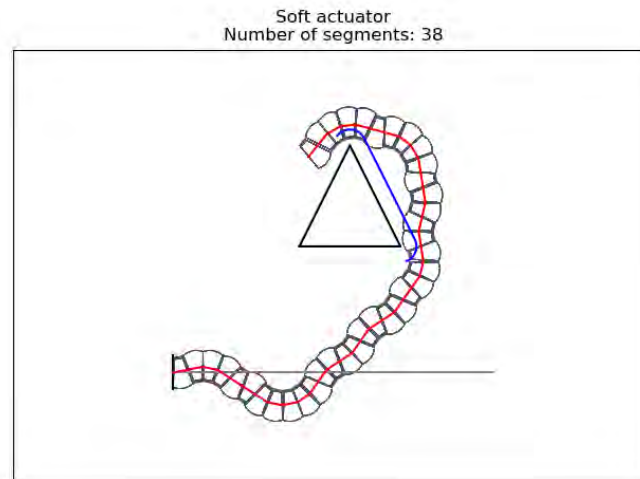
(f) The final actuator design for grasping a large triangle at a far distance.
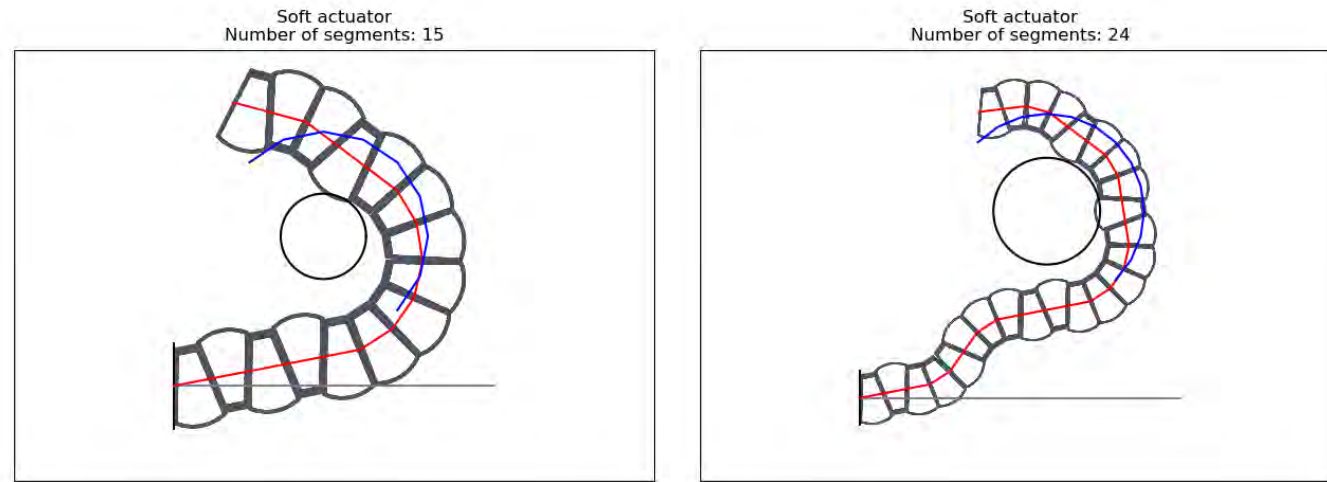
(g) The final actuator design for grasping a small circle at a close distance.

(h) The final actuator design for grasping a medium circle at an intermittent distance.
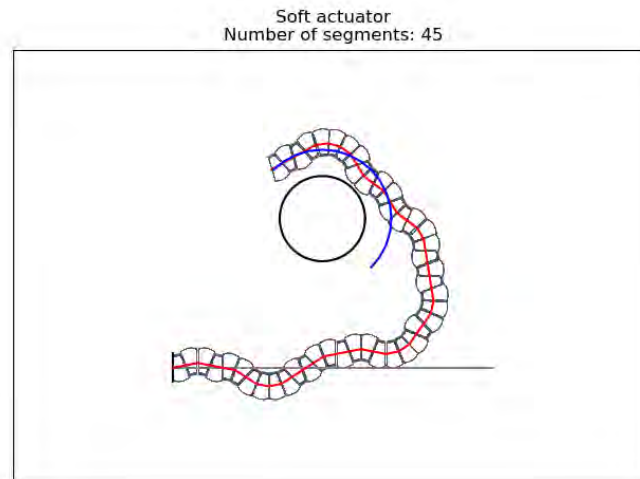
(i) The final actuator design for grasping a large circle at a far distance.

Figure 5.7: Generated actuator designs for grasping a square, triangular and circular object positioned at a near, intermediate and far distance from the origin

Table 5.1: The performance of the developed method for each of the test cases

| Object	Size	Distance	Number of designs evaluated	Evaluation runtime per individual
Square	Small	Close	56 000	0.00353 seconds
Square	Medium	Intermediate	29 000	0.00425 seconds
Square	Large	Far	42 750	0.00437 seconds
Triangle	Small	Close	38 500	0.00444 seconds
Triangle	Medium	Intermediate	50 000	0.00410 seconds
Triangle	Large	Far	56 000	0.00443 seconds
Circle	Small	Close	35 000	0.00371 seconds
Circle	Medium	Intermediate	42 250	0.00409 seconds
Circle	Large	Far	35 750	0.00438 seconds

Refer to Appendix A for computer hardware specifications.

Chapter 6

Extending the Generative Design Method

6.1 General Implementation

The method developed herein could be employed in a variety of design scenarios. The only requirement is the development of a domain representative L-system. Figure 6.2 illustrates the process that can be followed to apply this method to a different design application.

Firstly, an L-system alphabet must be developed for the design application. This step is context specific and will be determined by the design domain as well as the desired outcome of the generative design generator. A reduced-order model will be beneficial here but is not a mandatory requirement. The L-system alphabet must be developed to encode the characteristics of the design domain, as illustrated in Figure 6.1. This will include physical, material and dynamic (if applicable) properties e.g. an **iron** rod of **'X'** length and **'R'** radius with Young's modulus of **'Y'**. Therefore the L-system must take into account that this is a rigid rod, of a certain length and radius. This rod can be represented in the L-system alphabet by a character similar to the 'F' character used in this research.

Secondly, a fitness metric for the application must be developed. This fitness metric will depend on the desired outcome and must be tailored to each application. The fitness metric must then be numerically developed and incorporated into the GA to evaluate the generated designs. The designs can then be ranked based on the fitness metric be it minimisation or maximisation. The fitness metric must be complex enough to ensure that designs are non-trivial. By imposing constraints on the design generator, i.e. limiting the footprint or applying gravitational force, more novel designs may also be generated. The benefit of a reduced-order model is that these constraints can be incorporated
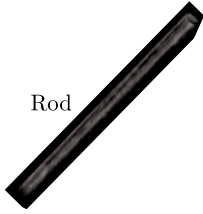
Object	Properties	L-system encoding	L-system properties
Rod	Rigid No elongation High strength	Requires an encoding that accounts for the rigidity of the rod. For dynamic systems the structural strength of the rod must also be included	The rod would have a fixed length and diameter in an L-system encoding. It would be capable of supporting other objects
Rope	Flexible Slight elongation Medium strength	Requires an encoding that accounts for the flexibility and instability of the rope. It can not be used as structural support	The rope would be a flexible segment capable of slight elongation. It would not be capable of supporting other objects
Spring	Slight bending Elongation & Compression High strength	Requires an encoding that accounts for the compression and elongation of the spring inlcuding the forces generated.	The spring would be an elastic member capable of elongation and compression.

Figure 6.1: An L-system encoding must capture all the properties of the object to accurately represent the object during the generative design process. All of the objects properties must be accounted for when developing the encoding.

into the L-system alphabet thereby removing the need to impose them on the generator during the GA optimisation.

6.2   Possible Applications

Further development of this method may allow more complex actuators to be developed and may even prove suitable to develop sub-assemblies of these designs. The following is a discussion on possible applications of the method developed herein.

Claw Assembly

Using the grasping actuator designs, an assembly could be created to perform similarly to a claw type attachment as shown in Figure 6.3a. Utilising symmetry conditions, three similar actuator designs can be combined to form a triangular claw as illustrated in Figure 6.3b. This developed assembly could
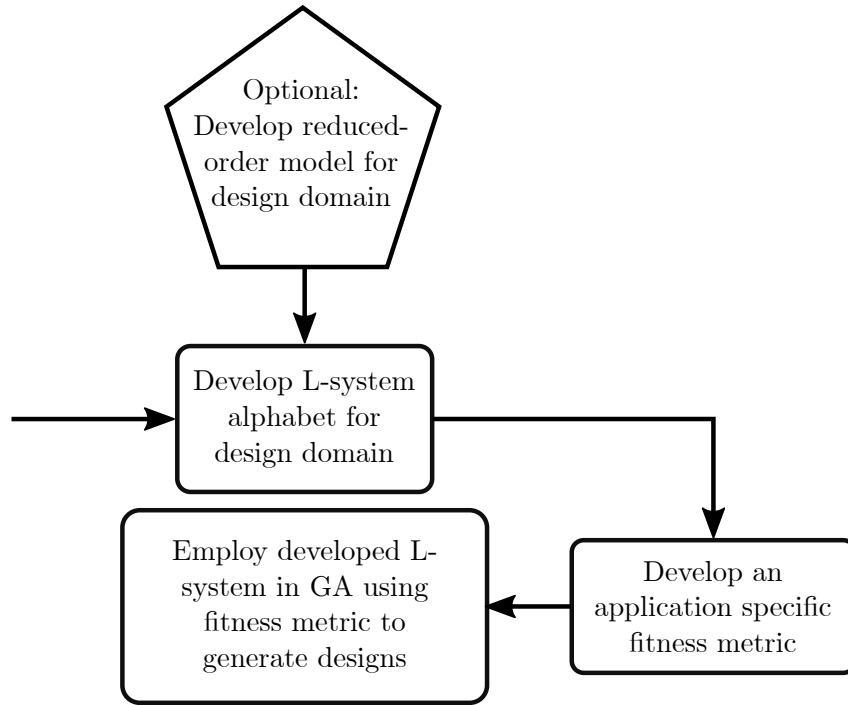
Figure 6.2: The generalised methodology for the method developed herein can be employed in a variety of design scenarios

then be used to lift and place fragile or soft objects by incorporating pressure feedback into the actuator assembly to control the amount of pressure supplied to the actuators. This assembly would also be advantageous in lifting irregularly shaped objects, due to the conforming nature of the soft material.

(a) Normal claw attachment

(b) Soft robotic claw attachment
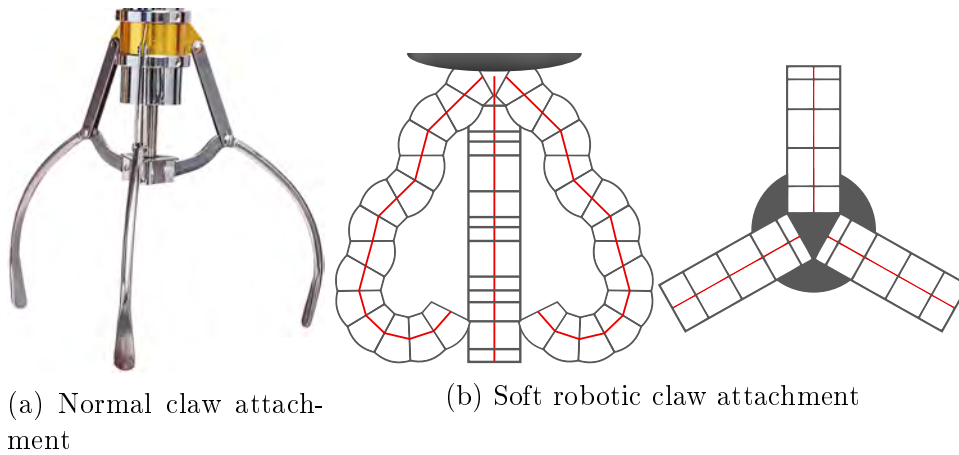
Figure 6.3: Claw assembly application for soft robotic bending actuator designs

Muscular Hydrostat

A further variant of the above mentioned application is to affix the claw assembly to a muscular hydrostat. Muscular hydrostats are biological appendages/limbs consisting mainly of muscles as opposed to a combination of muscle and bone. These limbs possess a constant volume and manoeuvre by the transference of fluid. Due to the constant volume property, decreasing of fluid in one position causes an increase of fluid in at least one other position thereby inducing movement of the limb. Examples include a human tongue, and elephant trunk and an octopus tentacle. For this particular application consider the octopus tentacle and its suckers. Here, the suckers may be replaced by the claw assemblies as described above to form a soft robotic tentacle capable of grabbing several smaller objects or wrapping around larger object, see Figure 6.4.
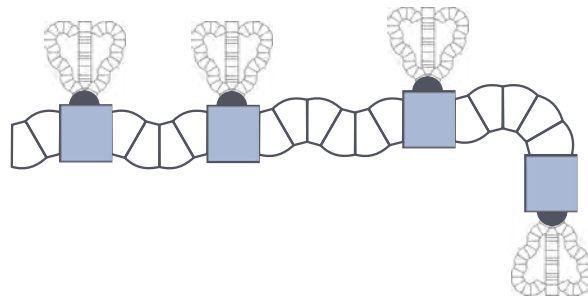
Figure 6.4: The claw assembly can be combined with a larger soft robotic bending actuator to form a soft robotic tentacle, capable of grabbing multiple objects.

Chapter 7

Conclusion

The aim of this research was to develop a generative design method for structural designs, that functions as a suitable alternative to traditional resource heavy techniques. To illustrate the capabilities of the method it was applied to generate designs for soft robotic bending actuators to perform targeted functions.

Lindenmayer-systems were identified as a suitable encoding for generative design after being employed in a preliminary evaluation to generate algae designs. The method was successful in generating designs, in both unconstrained and constrained environments, that exhibited logical behaviour in these scenarios. The tool proved capable of complex geometry generation using a genetic algorithm to optimise the desired algae geometries.

A reduced-order model was developed for a soft robotic bending actuator, similar to work done by Ellis *et al.* (2019). Utilising finite element analysis, he created a reduced-order model for a fabricated soft robotic bending actuator. The quantifiable geometric transformations of the fabricated soft robotic bending actuator were utilised to develop the reduced-order model herein. This model correlated well with the fabricated soft robotic bending actuator for various test cases and achieved evaluation runtime reductions in the order of 1×10^6 compared to Ellis.

The reduced-order model was then coupled with Lindenmayer-systems to develop a method to generate soft robotic bending actuator designs. An encoding for the reduced-order model was developed to use as the *Alphabet* of the L-system. This combination of techniques was applied to generate soft robotic bending actuators, that were optimised to achieve target shapes or perform specific functions by utilising curve fitting. The method proved capable of generating designs for a grasping type soft robotic bending actuator. It also proved capable of consistently providing these capable designs for a variety of objects and scales.

Chapter 8

Future Research

8.1 Generalisation

The method developed herein could possibly be adapted to any application. The only requirement would be to frame the design domain in terms of an L-system. A library could be created to implement this method as a general generative design tool using the following steps:

- Develop an alphabet for the application. The alphabet must comprise the encoding used to represent the design domain and the physical attribute for each character in the alphabet i.e. 'F' is a line segment of 1 unit.

- Determine the parameters of the GA that suits the application best. Some applications may benefit from a different selection method. This can also be automated similar to the technique employed in most machine learning algorithms for hyper-parameter tuning.

- Images and other illustrations can be added after design generation to provide a better visualisation of the generated designs.

Reduced-order modelling would be a beneficial inclusion but it is not a mandatory requirement. Reduced-order modelling will, however, decrease the computational requirements of the algorithm allowing more iterations of design generation.

8.2 Machine Learning

Machine Learning could be employed when developing the soft robotic bending actuator designs. Combinatorial machine learning is an active field of research. The use of machine learning for this purpose is twofold, Bengio *et al.* (2018).

Firstly, machine learning can be used to create approximations for the design problem that are less computationally expensive and secondly, machine learning can be utilised to develop an understanding of the design domain to gain experience in developing the best performing behaviour. Artificial neural networks can be used to generate the orientation vector used to develop a soft robotic bending actuator with each output node of the network corresponding to a single module in the soft robotic bending actuator.
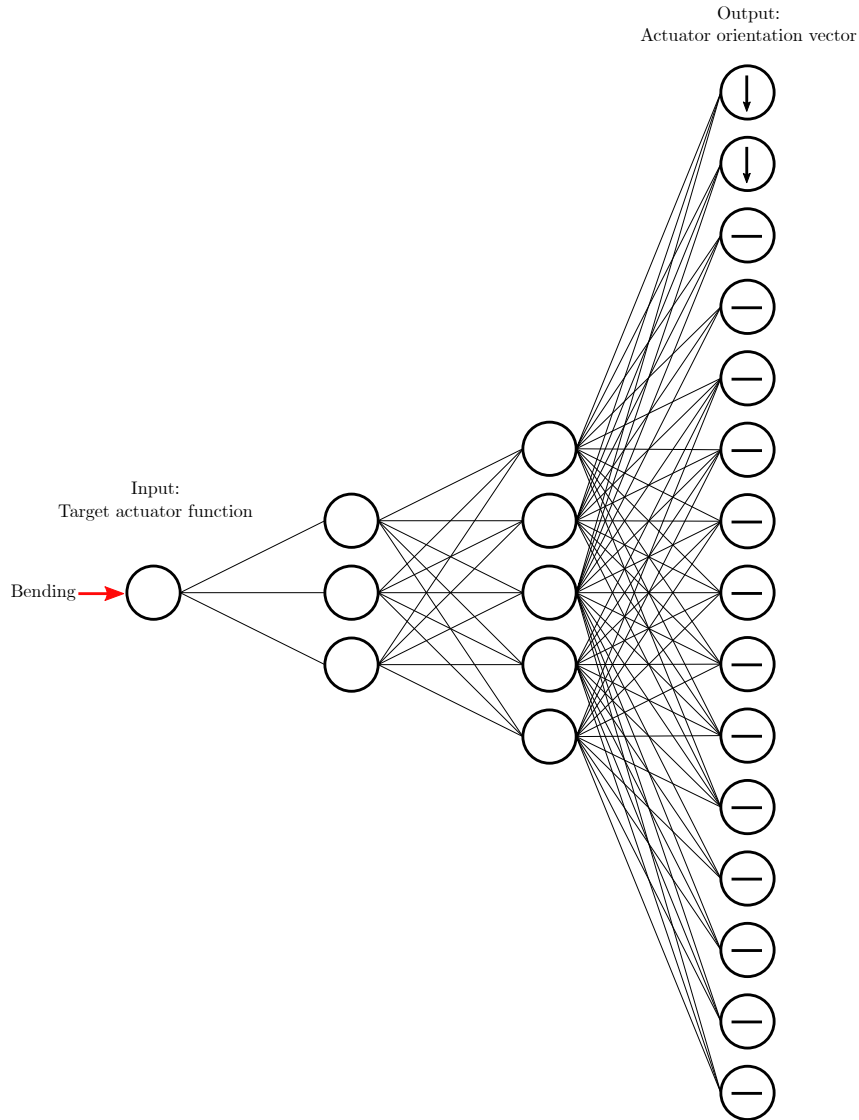
Figure 8.1: A combinatorial ANN that could be utilised to generate unique sub-units. The input is the target actuator function like bending or elongation and the output is the 15 segment orientation vector of the soft actuator

8.3 Alternative Grasping Proxies

Alternative grasping proxies may be explored for the grasping actuator design. Promising avenues include collision detection and contact algorithms. These could be employed to generate designs that slightly deform around the object resulting in a more firm grasp. The level of required deformation will depend on the weight and compressibility of the object. Only sufficient deformation must be ensured lest the object be damaged by the actuator.

Utilising these methodologies may incur a computational penalty but will deliver fabrication-ready actuator designs, capable of firmly grasping objects in real-world scenarios.

Appendices

Appendix A

Hardware specifications

The computational calculations and simulations were performed using the following hardware:

Parameter	Value
CPU	Intel Core i5-2400 @ 3.10 GHz
RAM	12 GB DDR3
OS	Windows 10 Pro 64-bit
HDD	Mushkin MKNSSDSR120GB
Language	Python 3.6

Bibliography

Arora, J.S. (2006). Jan A. Snyman, Practical Mathematical Optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms. *Structural and Multidisciplinary Optimization*. ISSN 1615-147X.

Autodesk Inc. (). What is Generative Design | Tools & Software | Autodesk. Available at: `https://www.autodesk.com/solutions/generative-design`

Axten, N., Newell, A. and Simon, H.A. (1973). Human Problem Solving. *Contemporary Sociology*. ISSN 00943061.

Bay, J.S. (1995). Design of the "Army-Ant" Cooperative Lifting Robot. *IEEE Robotics and Automation Magazine*. ISSN 10709932.

Beck, K., Beedle, M., Bennekum, A.V., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001). Manifesto for Agile Software Development.

Bengio, Y., Lodi, A. and Prouvost, A. (2018). Machine learning for combinatorial optimization: a methodological tour d'horizon. *arXiv preprint arXiv:1811.06128*.

Beni, G. and Wang, J. (1993). Swarm Intelligence in Cellular Robotic Systems. In: *Robots and Biological Systems: Towards a New Bionics?*, pp. 703–712. Springer Berlin Heidelberg.

Benrós, D., Duarte, J.P. and Hanna, S. (2012). A New Palladian Shape Grammar. *International Journal of Architectural Computing*. ISSN 1478-0771.

Bentley, P.J. (2000). Exploring Component-based Representations - The Secret of Creativity by Evolution? In: *Evolutionary Design and Manufacture*.

Bieze, T.M., Largilliere, F., Kruszewski, A., Zhang, Z., Merzouki, R. and Duriez, C. (2018). Finite element method-based kinematics and closed-loop control of soft, continuum manipulators. *Soft Robotics*. ISSN 21695180.

Brooks, R.A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal on Robotics and Automation*. ISSN 08824967.

Brooks, S.P. and Morgan, B.J.T. (1995). Optimization using simulated annealing. *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 44, no. 2, pp. 241–257.

Brown, J.H., Marquet, P.A. and Taper, M.L. (1993 oct). Evolution of body size: consequences of an energetic definition of fitness. *The American naturalist*, vol. 142, no. 4, pp. 573–84. ISSN 0003-0147.
Available at: http://www.ncbi.nlm.nih.gov/pubmed/19425961

Çağdaş, G. (1996 jan). A shape grammar model for designing row-houses. *Design Studies*, vol. 17, no. 1, pp. 35–51. ISSN 0142694X.
Available at: https : / / linkinghub . elsevier . com / retrieve / pii / 0142694X9500005C

Cagan, J. (2001). Engineering Shape Grammars: Where We Have Been and Where We Are Going. In: Antonsson, E.K. and Cagan, J. (eds.), *Formal Engineering Design Synthesis*, chap. 3, pp. 65–92. Cambridge University Press, Cambridge, UK. ISBN 0-521-79247-9.
Available at: http://books.google.com/books?id=m5DKOFSuqysC{&}lpg= PA65{&}ots = AZKotosLbz{&}dq = Engineeringshapegrammars{&}lr{&}pg = PP1{#}v=onepage{&}q=Engineeringshapegrammars{&}f=false

Celaya, E. and Porta, J.M. (1998). A Control Structure for the Locomotion of a Legged Robot on Difficult Terrain. *IEEE Robotics and Automation Magazine*. ISSN 10709932.

Che, K., Rouleau, M. and Meaud, J. (2019 oct). Temperature-tunable time-dependent snapping of viscoelastic metastructures with snap-through instabilities. *Extreme Mechanics Letters*, vol. 32, p. 100528. ISSN 23524316.

Chenevier, J., González, D., Aguado, J.V., Chinesta, F. and Cueto, E. (2018). Reduced-order modeling of soft robots. *PLoS ONE*. ISSN 19326203.

Cheney, N., MacCurdy, R., Clune, J. and Lipson, H. (2013). Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation - GECCO '13*. ISSN 19318499.

Chien, S.-F. and Flemming, U. (2002 jan). Design space navigation in generative design systems. *Automation in Construction*, vol. 11, no. 1, pp. 1–22. ISSN 09265805.
Available at: https : / / linkinghub . elsevier . com / retrieve / pii / S0926580500000844

Chipperfield, A., Fleming, P. and Pohlheim, H. (1994). Genetic Algorithm Toolbox for use with MATLAB. *Department of Automatic . . . .*

Cho, K.J., Koh, J.S., Kim, S., Chu, W.S., Hong, Y. and Ahn, S.H. (2009 jul). Review of manufacturing processes for soft biomimetic robots.

Chopard, B. (2012). Cellular automata modeling of physical systems. In: *Computational Complexity: Theory, Techniques, and Applications*. ISBN 9781461418009.

Cisilino, A.P. and Sensale, B. (2002). Application of a simulated annealing algorithm in the optimal placement of the source points in the method of the fundamental solutions. *Computational Mechanics*. ISSN 01787675.

Coevoet, E., Escande, A. and Duriez, C. (2017). Optimization-Based Inverse Model of Soft Robots with Contact Handling. *IEEE Robotics and Automation Letters*. ISSN 23773766.

Coulom, R. (2002). Reinforcement learning using neural networks, with applications to motor control.
Available at: `http://hal.univ-grenoble-alpes.fr/tel-00003985/`

Coutinho, F., Castro E Costa, E., Duarte, J.P. and Kruger, M. (2013). A shape grammar to generate Loggia Rucellai. In: *Open Systems - Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia, CAADRIA 2013*. ISBN 9789881902641.

Cross, N. (2011). The Design of Design: Essays from a Computer Scientist. *Design Studies*. ISSN 0142694X.

Cross, N., Dorst, K., Roozenburg, N., Technische Hogeschool Delft. Faculty of Industrial Design Engineering. and Workshop of Research in Design Thinking (1991 : Delft, N. (1992). *Research in design thinking : proceedings of a workshop meeting held at the Faculty of Industrial Design Engineering, Delft University oof Technology, the Netherlands, May 29-31, 1991*. Delft University Press. ISBN 9062757960.

de Garis, H. (1994 jun). An artificial brain ATR's CAM-Brain Project aims to build/evolve an artificial brain with a million neural net modules inside a trillion cell Cellular Automata Machine. *New Generation Computing*, vol. 12, no. 2, pp. 215–221. ISSN 02883635.

Duan, J., Liang, X., Zhu, K., Guo, J. and Zhang, L. (2017). Bilayer hydrogel actuators with tight interfacial adhesion fully constructed from natural polysaccharides. *Soft Matter*. ISSN 17446848.

Duriez, C. (2013). Control of elastic soft robots based on real-time finite element method. *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3982–3987. ISSN 10504729.
Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}M54a4cc7f1427726eab4M4aa52061377553{&}database=cpx`

Eberhart, R.C., Simpson, P.K. and Dobbins, R.W. (1996). Evolutionary computation implementations. *Computational Intelligence PC Tools*, pp. 212–226.

Ellis, D.R., Venter, M.P. and Venter, G. (2019 may). Computational design for inflated shape of a modular soft robotic actuator. In: *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, pp. 7–12. Institute of Electrical and Electronics Engineers Inc. ISBN 9781538692608.

Epstein, J.M. (1999). Agent-based computational models and generative social science. *Complexity*. ISSN 10990526.

Faber, P. and Fisher, R. (2001). Pros and cons of euclidean fitting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2191, pp. 414–420. ISSN 03029743. Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}M30a8781d150f8df4165M7dfa10178163171{&}database=cpx`

Fischer, T. and Herr, C. (2001). Teaching generative design. ... *of the 4th Conference on Generative Art*.

Fogel, D.B. (1997). The advantages of evolutionary computation. In: *BCEC*, pp. 1–11.

Françon, J. (1997). The algorithmic beauty of plants. *Plant Science*. ISSN 01689452.

Gilbertson, M.D., McDonald, G., Korinek, G., Van De Ven, J.D. and Kowalewski, T.M. (2017). Serially Actuated Locomotion for Soft Robots in Tube-Like Environments. *IEEE Robotics and Automation Letters*. ISSN 23773766.

Goldberg, D.E. and Holland, J.H. (1988). Genetic Algorithms and Machine Learning. *Machine Learning*. ISSN 15730565.

Gorissen, B., Reynaerts, D., Konishi, S., Yoshida, K., Kim, J.W. and De Volder, M. (2017). Elastic Inflatable Actuators for Soft Robotic Applications.

Gorissen, B., Volder, M.D., Chip, D.R.L.o.a. and undefined 2015 (). Pneumatically-actuated artificial cilia array for biomimetic fluid propulsion. *pubs.rsc.org*. Available at: `https://pubs.rsc.org/en/content/articlehtml/2015/lc/c5lc00775e`

Gron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st edn. O'Reilly Media, Inc. ISBN 1491962291, 9781491962299.

Gu, S., Holly, E., Lillicrap, T. and Levine, S. (). Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates. Tech. Rep.. `1610.00633v2`.

Gullichsen, E. and Chang, E. (1985). GENERATIVE DESIGN IN ARCHITECTURE USING AN EXPERT SYSTEM. In: *Proceedings - Graphics Interface*, pp. 425–433. Canadian Information Processing Soc. ISSN 07135424.

Gupta, A., Eppner, C., Levine, S. and Abbeel, P. (2016). Learning dexterous manipulation for a soft robotic hand from human demonstrations. In: *IEEE International Conference on Intelligent Robots and Systems*. ISBN 9781509037629. ISSN 21530866. `1603.06348`.

Hart, M., Abelson, H. and di Sessa, A. (1987). Turtle Geometry. *The Mathematical Gazette*. ISSN 00255572.

Higashi, K. and Miki, N. (2014). A self-swimming microbial robot using microfabricated nanofibrous hydrogel. *Sensors and Actuators, B: Chemical*. ISSN 09254005.

Hiller, J. and Lipson, H. (2012). Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*. ISSN 15523098.

Hirai, S., Cusin, P., Tanigawa, H., ..., T.M......I. and undefined 2000 (). Qualitative synthesis of deformable cylindrical actuators through constraint topology. *ieeexplore.ieee.org*.
Available at: https://ieeexplore.ieee.org/abstract/document/894604/

Hornby, G. and Pollack, J. (2001*a*). Body-Brain Co-evolution Using L-systems as a Generative Encoding. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*.

Hornby, G. and Pollack, J. (2002*a*). The advantages of generative grammatical encodings for physical design.

Hornby, G.S., Lipson, H. and Pollack, J.B. (2001). Evolution of generative design systems for modular physical robots. In: *Proceedings - IEEE International Conference on Robotics and Automation*. ISBN 0780365763. ISSN 10504729.

Hornby, G.S. and Pollack, J.B. (2001*b*). Evolving L-systems to generate virtual creatures. *Computers and Graphics (Pergamon)*. ISSN 00978493.

Hornby, G.S. and Pollack, J.B. (2002*b*). Creating high-level components with a generative representation for body-brain evolution. *Artificial life*, vol. 8, no. 3, pp. 223–246. ISSN 10645462.

Hunt, K., Sbarbaro, D., Żbikowski, R., Automatica, P.G. and 1992, U. (1992). Neural networks for control systems—a survey. *Automatica*, vol. 28, no. 6, pp. 1083–1112.
Available at: https://www.sciencedirect.com/science/article/pii/000510989290053I

Iida, F. and Laschi, C. (2011). Soft robotics: Challenges and perspectives. In: *Procedia Computer Science*, vol. 7, pp. 99–102. Elsevier B.V. ISSN 18770509.

Ilachinski, A. (2001). *Cellular Automata*.

Inc., S.-O. (). Smooth-On, Inc. | Mold Making & Casting Materials | Rubbers, Plastics, Foams & More!
Available at: https://www.smooth-on.com/

Jekel, C.F. and Venter, G. (2019). pwlf: A Python Library for Fitting 1D Continuous Piecewise Linear Functions.

Jeong, O.C. and Konishi, S. (2006 aug). All PDMS pneumatic microfinger with bidirectional motion and its application. *Journal of Microelectromechanical Systems*, vol. 15, no. 4, pp. 896–903. ISSN 10577157.

Kasmarik, K., Gu, N., Singh, V. and Merrick, K. (). A framework to integrate generative design techniques for enhancing design automation. Tech. Rep..
Available at: `https://www.researchgate.net/publication/41903982`

Katzschmann, R.K., Thieffry, M., Goury, O., Kruszewski, A., Guerra, T.-M., Duriez, C. and Rus, D. (2019 may). Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer. *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, pp. 717–724.
Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}M46494e0416b8a24b159M75a810178163167{&}database=cpx`

Kim, B., Lee, S.B., Lee, J., Cho, S., Park, H., Yeom, S. and Park, S.H. (2012*a*). A comparison among Neo-Hookean model, Mooney-Rivlin model, and Ogden model for Chloroprene rubber. *International Journal of Precision Engineering and Manufacturing*. ISSN 12298557.

Kim, L.C. and Talib, A.Z. (2010). A visual language framework for music rendering using L-System. *International Conference on Visualization, Imaging and Simulation - Proceedings*, pp. 47–52. ISSN 17926130.
Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}6e3d601314319e372M69ec2061377553{&}database=cpx`

Kim, S.W., Koh, J.S., Cho, M. and Cho, K.J. (2012 mar*b*). Soft morphing motion of flytrap robot using bending propagating actuation. *Journal of Institute of Control, Robotics and Systems*, vol. 18, no. 3, pp. 168–174. ISSN 19765622.

Komosiński, M., VII, A.R.-V.A.L. and undefined 2000 (). From directed to open-ended evolution in a complex simulation model. *books.google.com*.
Available at: `https://books.google.com/books?hl=en{&}lr={&}id=-xLGm7KFGy4C{&}oi=fnd{&}pg=PA293{&}dq=From+directed+to+open-ended+evolution{&}ots=-3QRI2nD8J{&}sig=7KREQ-fcKYTGv7odtn--HELKcp8`

Konrády, T., Štekerová, K. and Tesařová, B. (2016). L2 Designer. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9770 LNCS, pp. 83–100. Springer Verlag.
Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}M4ebaabc415840495727M6fb210178163171{&}database=cpx`

Krish, S. (2011 jan). A practical generative design method. *Computer-Aided Design*, vol. 43, no. 1, pp. 88–100. ISSN 00104485.
Available at: `https://linkinghub.elsevier.com/retrieve/pii/S0010448510001764`

Kwon, G.H., Park, J.Y., Kim, J.Y., Frisk, M.L., Beebe, D.J. and Lee, S.H. (2008 dec). Biomimetic soft multifunctional miniature aquabots. *Small*, vol. 4, no. 12, pp. 2148–2153. ISSN 16136810.

Largilliere, F., Verona, V., Coevoet, E., Sanz-Lopez, M., Dequidt, J. and Duriez, C. (2015). Real-time control of soft-robots using asynchronous finite element modeling. In: *Proceedings - IEEE International Conference on Robotics and Automation*. ISSN 10504729.

Laschi, C. and Cianchetti, M. (2014). Soft robotics: New perspectives for robot bodyware and control. *Frontiers in Bioengineering and Biotechnology*, vol. 2, no. JAN. ISSN 22964185.

Laschi, C., Cianchetti, M., Mazzolai, B., Margheri, L., Follador, M. and Dario, P. (2012). Soft robot arm inspired by the octopus. *Advanced Robotics*, vol. 26, no. 7, pp. 709–727. ISSN 01691864.

Lessin, D., Fussell, D. and Miikkulainen, R. (2013). Open-ended behavioral complexity for evolved virtual creatures. In: *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*. ISBN 9781450319638.

Lin, H.T., Leisk, G.G. and Trimmer, B. (2011). GoQBot: A caterpillar-inspired soft-bodied rolling robot. *Bioinspiration and Biomimetics*, vol. 6, no. 2. ISSN 17483182. `79958131392`.

Lin, L.-J. (1993). Reinforcement learning for robots using neural networks. *ProQuest Dissertations and Theses*.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*. ISSN 10958541.

Lindenmayer, A. and Rozenberg, G. (1972). Developmental systems and languages. In: *Proceedings of the Annual ACM Symposium on Theory of Computing*. ISSN 07378017.

Lipson, H. (2014 mar). Challenges and Opportunities for Design, Simulation, and Fabrication of Soft Robots.

Lobos, A. (2018). Finding balance in generative product design. In: *Proceedings of NordDesign: Design in the Era of Digitalization, NordDesign 2018*. ISBN 9789176851852.

MacDonald, S., Szafron, D., Schaeffer, J., Anvik, J., Bromling, S. and Tan, K. (). Generative design patterns. In: *Proceedings 17th IEEE International Conference on Automated Software Engineering,*, pp. 23–34. IEEE Comput. Soc. ISBN 0-7695-1736-6.
Available at: `http://ieeexplore.ieee.org/document/1114991/`

Marchese, A.D., Onal, C.D. and Rus, D. (2014 mar). Autonomous Soft Robotic Fish Capable of Escape Maneuvers Using Fluidic Elastomer Actuators. *Soft Robotics*, vol. 1, no. 1, pp. 75–87. ISSN 21695172.

Marinov, M., Amagliani, M., Barback, T., Flower, J., Barley, S., Furuta, S., Charrot, P., Henley, I., Santhanam, N., Finnigan, G.T., Meshkat, S., Hallet, J., Sapun, M. and Wolski, P. (2019 oct). Generative Design Conversion to Editable and Watertight Boundary Representation. *Computer-Aided Design*, vol. 115, pp. 194–205. ISSN 00104485.
Available at: `https : / / linkinghub . elsevier . com / retrieve / pii / S0010448519301873`

Martinez, R.V., Glavan, A.C., Keplinger, C., Oyetibo, A.I. and Whitesides, G.M. (2014). Soft actuators and robots that are resistant to mechanical damage. *Advanced Functional Materials*, vol. 24, no. 20, pp. 3003–3010. ISSN 16163028. `0706.1062v1`.

Mayer, H., Gomez, F., Wierstra, D., Nagy, I., Knoll, A. and Schmidhuber, J. (2008 sep). A system for robotic heart surgery that learns to tie knots using recurrent neural networks. *Advanced Robotics*, vol. 22, no. 13-14, pp. 1521–1537. ISSN 01691864.

McCracken, D.D. and Jackson, M.A. (1982). Life cycle concept considered harmful. *ACM SIGSOFT Software Engineering Notes*. ISSN 01635948.

McShea, D.W. (1994 dec). MECHANISMS OF LARGE-SCALE EVOLUTIONARY TRENDS. *Evolution*, vol. 48, no. 6, pp. 1747–1763. ISSN 00143820.
Available at: `http://doi.wiley.com/10.1111/j.1558-5646.1994.tb02211.x`

Meng, C., Xu, W., Li, H., Zhang, H. and Xu, D. (2017). A new design of cellular soft continuum manipulator based on beehive-inspired modular structure. *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417707380.
Available at: `https://doi.org/10.1177/1729881417707380`

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. and Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*. ISSN 00219606.

Minh, T.V., Kamers, B., Ramon, H. and Brussel, H.V. (2012). Modeling and control of a pneumatic artificial muscle manipulator joint — Part I: Modeling of a pneumatic artificial muscle manipulator joint with accounting for creep effect. *Mechatronics*, vol. 22, no. 7, pp. 923–933. ISSN 0957-4158.
Available at: `http : / / www . sciencedirect . com / science / article / pii / S0957415812000967`

Moni, R. (). Reinforcement Learning algorithms — an intuitive overview.
Available at: `https://medium.com/@SmartLabAI/reinforcement-learning-algorithms-an-intuitive-overview-904e2dff5bbc`

Moseley, P., Florez, J.M., Sonar, H.A., Agarwal, G., Curtin, W. and Paik, J. (2016 jun). Modeling, Design, and Development of Soft Pneumatic Actuators with Finite Element Method. *Advanced Engineering Materials*, vol. 18, no. 6, pp. 978–988. ISSN 15272648.

Muggeo, V.M.R. (2003). Estimating regression models with unknown break-points. *Statistics in Medicine*. ISSN 02776715.

Muslimin, R. (2017). EthnoComputation: An inductive shape grammar on Toraja glyph. In: *Communications in Computer and Information Science*. ISBN 9789811051968. ISSN 18650929.

Must, I., Kaasik, F., Põldsalu, I., Mihkels, L., Johanson, U., Punning, A. and Aabloo, A. (2015 jan). Ionic and capacitive artificial muscle for biomimetic soft robotics. *Advanced Engineering Materials*, vol. 17, no. 1, pp. 84–94. ISSN 15272648.

Nagabandi, A., Kahn, G., Fearing, R.S. and Levine, S. (). Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. Tech. Rep.. `1708.02596v2`.
Available at: `https://sites.google.com/view/mbmf`

Neapolitan, R.E. and Neapolitan, R.E. (2018). Neural Networks and Deep Learning. In: *Artificial Intelligence*.

Nishikawa, S., Arai, Y., Niiyama, R. and Kuniyoshi, Y. (2018 apr). Coordinated Use of Structure-Integrated Bistable Actuation Modules for Agile Locomotion. *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1018–1024. ISSN 23773766.

Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

Nordin, A. (2018 feb). Challenges in the industrial implementation of generative design systems: An exploratory study. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 32, no. 1, pp. 16–31. ISSN 0890-0604.
Available at: `https://www.cambridge.org/core/product/identifier/S0890060416000536/type/journal{_}article`

Oh, S., Jung, Y., Kim, S., Lee, I. and Kang, N. (2019 nov). Deep Generative Design: Integration of Topology Optimization and Generative Models. *Journal of Mechanical Design*, vol. 141, no. 11. ISSN 1050-0472.
Available at: `https://asmedigitalcollection.asme.org/mechanicaldesign/article/doi/10.1115/1.4044229/955342/Deep-Generative-Design-Integration-of-Topology`

Park, Y.-L. and Wood, R.J. (2013). Smart pneumatic artificial muscle actuator with embedded microfluidic sensing. In: *SENSORS, 2013 IEEE*, pp. 1–4. IEEE.

Pollack, J.B., Hornby, G.S., Lipson, H. and Funes, P. (2003). Computer creativity in the automatic design of robots. *Leonardo*. ISSN 0024094X.

Pollack, J.B., Lipson, H., Hornby, G. and Funes, P. (2001). Three generations of automatically designed robots. *Artificial Life*, vol. 7, no. 3, pp. 215–223. ISSN 10645462.

Polygerinos, P., Wang, Z., Galloway, K.C., Wood, R.J. and Walsh, C.J. (2015 nov). Soft robotic glove for combined assistance and at-home rehabilitation. In: *Robotics and Autonomous Systems*, vol. 73, pp. 135–143. Elsevier. ISSN 09218890.

Ralph, P. (2010). Comparing two software design process theories. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. ISBN 3642133347. ISSN 03029743.

Renda, F., Cianchetti, M., Giorelli, M., Arienti, A. and Laschi, C. (2012). A 3D steady-state model of a tendon-driven continuum soft manipulator inspired by the octopus arm. *Bioinspiration and Biomimetics*, vol. 7, no. 2. ISSN 17483182.

Renda, F., Giorelli, M., Calisti, M., Cianchetti, M. and Laschi, C. (2014). Dynamic model of a multibending soft robot arm driven by cables. *IEEE Transactions on Robotics*. ISSN 15523098.

Rieffel, J. and Smith, S. (2010). A face-encoding grammar for the generation of tetrahedral-mesh soft bodies. In: *Artificial Life XII: Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2010*. ISBN 9780262290753.

Rieffel, J. and Smith, S. (2012). Growing and evolving soft robots with a face-encoding tetrahedral grammar. In: *GECCO'12 - Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion*. ISBN 9781450311786.

Robertson, M.A. and Paik, J. (2017). New soft robots really suck: Vacuum-powered systems empower diverse capabilities. *Sci. Robot.*, vol. 2, p. eaan6357.

Roche, E.T., Wohlfarth, R., Overvelde, J.T.B., Vasilyev, N.V., Pigula, F.A., Mooney, D.J., Bertoldi, K. and Walsh, C.J. (2014 feb). A bioinspired soft actuated material. *Advanced Materials*, vol. 26, no. 8, pp. 1200–1206. ISSN 09359648.

Roßmann, J., Schluse, M., Rast, M., Guiffo Kaigom, E. and Cichon, T. (2015 jan). Simulation technology for soft robotics applications. *Soft Robotics: Transferring Theory to Application*, pp. 100–119.
Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}596044315d5655846fM750f10178163176{&}database=cpx`

Rothman, D.H. (1987). Cellular-automaton fluids: A model for flow in porous media. In: *1987 SEG Annual Meeting*, pp. 297–300. Society of Exploration Geophysicists. ISSN 00168033.

Rozenberg, G. and Salomaa, A. (1974). The Mathematical Theory of L Systems. *DAIMI Report Series*. ISSN 0105-8517.

Rozenberg, G. and Salomaa, A. (2012). *Lindenmayer systems: impacts on theoretical computer science, computer graphics, and developmental biology*. Springer Science & Business Media.

Ruiz-Montiel, M., Boned, J., Gavilanes, J., Jiménez, E., Mandow, L. and Pérez-De-La-Cruz, J.L. (2013). Design with shape grammars and reinforcement learning. *Advanced Engineering Informatics*. ISSN 14740346.

Runge, G., Wiese, M., Gunther, L. and Raatz, A. (2017). A framework for the kinematic modeling of soft material robots combining finite element analysis and piecewise constant curvature kinematics. In: *2017 3rd International Conference on Control, Automation and Robotics, ICCAR 2017*. ISBN 9781509060870.

Rus, D. and Tolley, M.T. (2015 may). Design, fabrication and control of soft robots. *Nature*, vol. 521, no. 7553, pp. 467–475. ISSN 0028-0836.
Available at: http://www.nature.com/articles/nature14543

Rusu, A.A., Večerík, M., Rothörl, T., Heess, N., Pascanu, R. and Hadsell, R. (). Sim-to-Real Robot Learning from Pixels with Progressive Nets. Tech. Rep.. 1610.04286v2.

Salge, C., Green, M.C., Canaan, R. and Togelius, J. (2018). Generative design in minecraft (GDMC). In: *Proceedings of the 13th International Conference on the Foundations of Digital Games - FDG '18*, pp. 1–10. ACM Press, New York, New York, USA. ISBN 9781450365710.
Available at: http://dl.acm.org/citation.cfm?doid=3235765.3235814

Sanchez-Lengeling, B. and Aspuru-Guzik, A. (2018 jul). Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, vol. 361, no. 6400, pp. 360 LP – 365.
Available at: http://science.sciencemag.org/content/361/6400/360.abstract

Schmidhuber, J. (2015). Deep Learning in neural networks: An overview. 1404.7828.

Schoenauer, M. (1996). Shape Representations and Evolution Schemes. *Evolutionary Programming*, vol. 5.

Schön, D.A. (2017). *The reflective practitioner: How professionals think in action*. ISBN 9781351883160.

Schwab, K. (2019). This is first commercial chair made using generative design.
Available at: https://www.fastcompany.com/90334218/this-is-the-first-commercial-product-made-using-generative-design

Shan, H.B. and Shuxia, L. (2008). The comparison between genetic simulated annealing algorithm and ant colony optimization algorithm for ASP. In: *2008 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2008*. ISBN 9781424421084.

Shea, K., Aish, R. and Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. In: *Automation in Construction*, vol. 14, pp. 253–264. Elsevier. ISSN 09265805.

Shi, L., Guo, S., Li, M., Mao, S., Xiao, N., Gao, B., Song, Z. and Asaka, K. (2012). A novel soft biomimetic microrobot with two motion attitudes. *Sensors (Switzerland)*, vol. 12, no. 12, pp. 16732–16758. ISSN 14248220.

Shi, Y. and Eberhart, R.C. (1999). Empirical study of particle swarm optimization. In: *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*.

Shin, D., Tanaka, A., Kim, N. and Khatib, O. (2016). A Centrifugal Force-Based Configuration-Independent High-Torque-Density Passive Brake for Human-Friendly Robots. *IEEE/ASME Transactions on Mechatronics*. ISSN 10834435.

Sims, K. (1994). Evolving virtual creatures. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994*. ISBN 0897916670.

Stephens, B.J. and Atkeson, C.G. (2010). Dynamic balance force control for compliant humanoid robots. In: *IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings*, pp. 1248–1255. ISBN 9781424466757. ISSN 2153-0858.

Stiny, G. (1994 dec). Shape Rules: Closure, Continuity, and Emergence. *Environment and Planning B: Planning and Design*, vol. 21, no. 7, pp. S49–S78. ISSN 0265-8135.

Stiny, G. and Gips, J. (1972). Shape grammars and the generative specification of painting and sculpture. *Information Processing 71 Proceedings of the IFIP Congress 1971. Volume 2*. ISSN 0717-6163. `arXiv:1011.1669v3`.

Strasser, S., Goodman, R., Sheppard, J. and Butcher, S. (2016). A new discrete particle swarm optimization algorithm. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 53–60.

Sutton, R.S. and Barto, A.G. (2018). *Reinforcement Learning: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press. ISBN 9780262039246.
Available at: `https://books.google.co.za/books?id=sWV0DwAAQBAJ`

Suzumori, K., Endo, S., Kanda, T., Kato, N. and Suzuki, H. (2007 apr). A Bending Pneumatic Rubber Actuator Realizing Soft-bodied Manta Swimming Robot. In: *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4975–4980. IEEE. ISBN 1-4244-0601-3. ISSN 1050-4729.

Suzumori, K., Maeda, T., transactions on ..., H.W.....A. and undefined 1997 (). Fiberless flexible microactuator designed by finite-element method. *ieeexplore.ieee.org*.
Available at: `https://ieeexplore.ieee.org/abstract/document/653052/`

Tapia, M. (1999). A visual implementation of a shape grammar system. *Environment and Planning B: Planning and Design*. ISSN 02658135.

The Math Works (). What Is the Genetic Algorithm? - MATLAB & Simulink.
Available at: `https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html`

Thieffry, M., Kruszewski, A., Duriez, C. and Guerra, T.-m. (2018*a*). Control Design
for Soft Robots based on Reduced Order Model. *IEEE Robotics and Automation
Letters*. ISSN 2377-3766.

Thieffry, M., Kruszewski, A., Guerra, T.-M. and Duriez, C. (2018 nov*b*). Reduced
Order Control of Soft Robots with Guaranteed Stability. *2018 European Control
Conference, ECC 2018*, pp. 635–640.
Available at: `https://www.engineeringvillage.com/share/document.url?mid=cpx{_}fcdfed016875f6e3ecM72131017816339{&}database=cpx`

Thuruthel, T.G., Falotico, E., Renda, F. and Laschi, C. (2019). Model-Based Rein-
forcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipula-
tors. *IEEE Transactions on Robotics*. ISSN 15523098.

Trimmer, B.A., Lin, H.-T., Baryshyan, A., Leisk, G.G. and Kaplan, D.L. (2012 jun).
Towards a biomorphic soft robot: Design constraints and solutions. In: *2012
4th IEEE RAS & EMBS International Conference on Biomedical Robotics and
Biomechatronics (BioRob)*, pp. 599–605. IEEE. ISBN 978-1-4577-1200-5.
Available at: `http://ieeexplore.ieee.org/document/6290698/`

Trivedi, D., Rahn, C.D., Kier, W.M. and Walker, I.D. (2008 jan). Soft Robotics:
Biological Inspiration, State of the Art, and Future Research. *Applied Bionics and
Biomechanics*, vol. 5, no. 3, pp. 99–117. ISSN 1176-2322.
Available at: `https://doaj.org/article/365a8eb90e9843f3953673cab0358791`

Troiano, L. and Birtolo, C. (2014 feb). Genetic algorithms supporting generative
design of user interfaces: Examples. *Information Sciences*, vol. 259, pp. 433–451.
ISSN 00200255.
Available at: `https://linkinghub.elsevier.com/retrieve/pii/S0020025512000242`

Truex, D., Baskerville, R. and Travis, J. (2000). Amethodical systems development:
The deferred meaning of systems development methods. *Accounting, Management
and Information Technologies*. ISSN 09598022.

Udupa, G., Sreedharan, P., Dinesh, P.S. and Kim, D. (2014). Asymmetric bellow
flexible pneumatic actuator for miniature robotic soft gripper. *Journal of Robotics*,
vol. 2014. ISSN 16879619.

Vapnik, V. (2013). *The nature of statistical learning theory*. Springer science &
business media.

Venkataraman, S. and Haftka, R.T. (2004). Structural optimization complexity:
what has moore's law done for us? *Structural and Multidisciplinary Optimization*,
vol. 28, no. 6, pp. 375–387.

Vincent, J.F., Bogatyreva, O.A., Bogatyrev, N.R., Bowyer, A. and Pahl, A.K. (2006).
Biomimetics: Its practice and theory.

Wang, Z., Chen, M.Z. and Yi, J. (2015). Soft robotics for engineers. *HKIE Transactions Hong Kong Institution of Engineers*. ISSN 1023697X.

Wehr, J.D. (2015). Brown Algae. In: *Freshwater Algae of North America: Ecology and Classification*. ISBN 9780123858771.

Wolfram, S. (1984). Cellular automata as models of complexity. *Nature*. ISSN 00280836.

Yeoh, O.H. (1993). Some forms of the strain energy function for rubber. *Rubber Chemistry and Technology*, vol. 66, no. 5, pp. 754–771. ISSN 00359475.

Yue, K., Krishnamurti, R. and Grobler, F. (2012). Estimating the interior layout of buildings using a shape grammar to capture building style. *Journal of Computing in Civil Engineering*. ISSN 08873801.

Zhang, H., Cao, R., Zilberstein, S., Wu, F. and Chen, X. (2017). Toward effective soft robot control via reinforcement learning. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. ISBN 9783319652887. ISSN 16113349.