December 2021

Developing Design and Analysis Framework for Hybrid Mechanical-Digital Control of Soft Robots: from Mechanics-Based Motion Sequencing to Physical Reservoir Computing

Priyanka B. Bhovad
Clemson University, priyanka.bhovad@gmail.com

Developing design and analysis framework for hybrid mechanical-digital control of soft robots: from mechanics-based motion sequencing to physical reservoir computing

---

A Dissertation Document
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

---

by
Priyanka Bhovad
December 2021

Accepted by:
Dr. Suyi Li, Committee Chair
Dr. Ardalan Vahidi
Dr. Ian Walker
Dr. Phanindra Tallapragada

Abstract

The recent advances in the field of soft robotics have made autonomous soft robots working in unstructured dynamic environments a close reality. These soft robots can potentially collaborate with humans without causing any harm, they can handle fragile objects safely, perform delicate surgeries inside body, etc. In our research we focus on origami based compliant mechanisms, that can be used as soft robotic skeleton. Origami mechanisms are inherently compliant, lightweight, compact, and possess unique mechanical properties such as– multi-stability, nonlinear dynamics, etc. Researchers have shown that multi-stable mechanisms have applications in motion-sequencing applications. Additionally, the nonlinear dynamic properties of origami and other soft, compliant mechanisms are shown to be useful for 'morphological computation' in which the body of the robot itself takes part in performing complex computations required for its control.

In our research we demonstrate the motion-sequencing capability of multi-stable mechanisms through the example of bistable Kresling origami robot that is capable of peristaltic locomotion. Through careful theoretical analysis and thorough experiments we show that we can harness multi-stability embedded in the origami robotic skeleton for generating actuation cycle of a peristaltic-like locomotion gait. The salient feature of this compliant robot is that we need only a single linear actuator to control the total length of the robot, and the snap-through actions generated during this motion autonomously change the individual segment lengths that lead to earthworm-like peristaltic locomotion gait. In effect, the motion-sequencing is hard-coded or embedded in the origami robot skeleton. This approach is expected to reduce the control requirement drastically as the robotic skeleton itself takes part in performing low-level control tasks.

The soft robots that work in dynamic environments should be able to sense their surrounding and adapt their behavior autonomously to perform given tasks successfully. Thus, hard-coding a certain behavior as in motion-sequencing is not a viable option anymore. This led us to explore

Physical Reservoir Computing (PRC), a computational framework that uses a physical body with nonlinear properties as a 'dynamic reservoir' for performing complex computations. The compliant robot 'trained' using this framework should be able to sense its surroundings and respond to them autonomously via an extensive network of sensor-actuator network embedded in robotic skeleton. We show for the first time through extensive numerical analysis that origami mechanisms can work as physical reservoirs. We also successfully demonstrate the emulation task using a Miura-ori based reservoir. The results of this work will pave the way for intelligently designed origami-based robots with *embodied intelligence*. These next generation of soft robots will be able to coordinate and modulate their activities autonomously such as switching locomotion gait and resisting external disturbances while navigating through unstructured environments.

Dedicated to the loving memory of Gokhale kaka and kaku...

Acknowledgments

I would like to thank my advisor, Prof. Suyi Li for his extremely valuable guidance. I thank him for his patience, all the help he provided throughout my research, and most importantly his confidence in me and my abilities. His insights and in-depth knowledge helped me in making a consistent progress towards my research problem. I would like to thank my parents, my brothers and my husband for their constant support and belief in me. Without their encouragement my doctoral research would not have been possible. I thank all my friends at Clemson who helped me through tough times, and made life at Clemson enjoyable. I thank all my lab-mates for their valuable suggestions and company, especially Joshua Kauffman, for helping me design the origami robot and conducting extensive experiments. I would also like to thank my committee members for their support throughout this process. Lastly, I would like to thank Department of Mechanical Engineering at Clemson University for offering me this great opportunity.

Go Tigers!

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, robotics has entered into all aspects of our everyday life. Robots have become a ubiquitous and necessary, helping us with basic tasks such as cleaning and cooking to safely transporting us for long distances and manufacturing amenities and products in mass quantities necessary to live our lives comfortably. The current robots for the most part are non-collaborative in nature; since they are made from rigid parts, complex, bulky but very accurately operating electro-mechanical systems. In the age of advanced human-robot collaboration, the advantage of soft robots is clear. The soft robots can be integrated safely in an unstructured dynamic environment without causing any harm to nearby humans and things. In future, soft robots can be safely deployed for disaster rescue, rehabilitation assistance, manipulation of fragile/ delicate objects, minimally-invasive surgery, robotic wearables/exoskeletons, space/deep ocean exploration, remote monitoring in hazardous environment, etc.

One major obstacle in the widespread implementation of soft robots is their *control*. They are composed of soft and compliant materials that possess highly non-linear, large deformation range accompanied with virtually infinite degrees of freedom. Due to their complex non-linear dynamics, the accurate control of soft robots is very difficult and they usually need an array of soft, flexible sensors and actuators to accurately control and determine their trajectory. Additionally, the electronic hardware used for control and actuation of soft robots is largely *rigid*, which makes untethered control of soft robots a difficult task.

Keeping the complexity in implementation of soft robotic control in mind, first we will review the state-of-the-art of autonomous, untethered soft robots with special focus on soft robotic

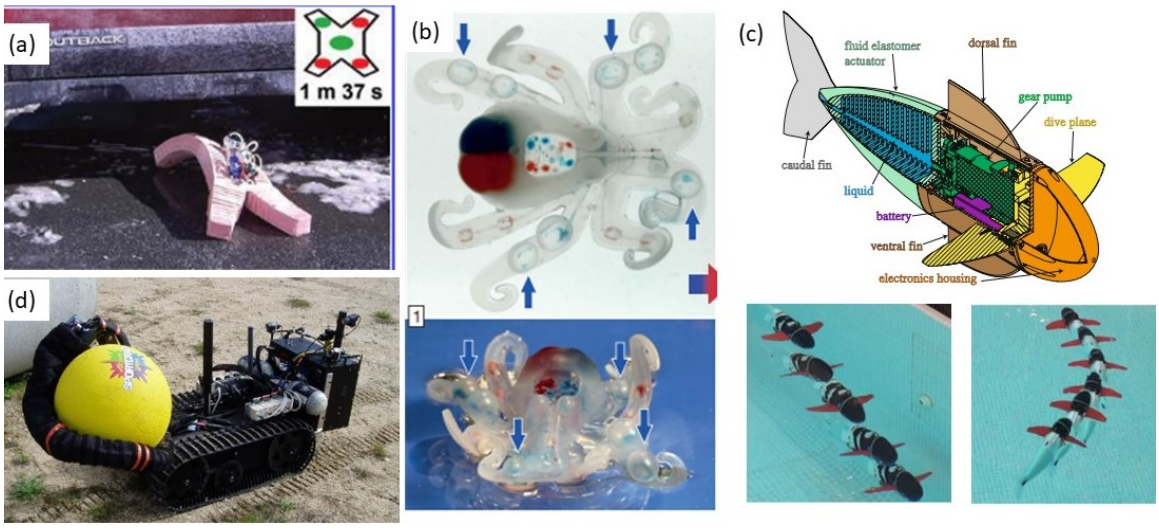

Figure 1.1: Pneumatically/hydraulically actuated autonomous soft robots. (a) Resilient quadruped robot [151], (b) octopus inspired octobot [163], (c) soft robotic fish capable of turning and diving [67], (d) continuum robotic OctArm [44]

locomotion. The focus here is to summarize emerging and widespread actuation and control technologies that are strong candidates for future implementation in autonomous, embodied soft robots and the field of morphological computation. Then, we will review origami mechanisms and Physical Reservoir Computing (PRC) from the point of view of their application to soft robotic control. Finally, we will define the objective of this research work and pose research questions that connect these topics together.

1.1 Autonomous Soft Robots

Pneumatic/hydraulic actuation has proven to be the most widespread method to design completely soft and untethered robots. One major category of these robots is composed of a layer of soft silicone-rubber like material with pre-defined internal cavities attached to a stiffer layer. When the cavities are inflated with pressurized fluid the difference in the stiffness between the layers causes the robot body to bend in various shapes depending on their internal structure. For example, Tolley et al. [151] developed a resilient, untethered quadruped robot with PneuNet architecture, on-board mini-air compressors and battery pack that can sustain limited exposure to fire, snow, water and execute various gaits. Wehner et al. [163] developed an entirely soft, autonomous, octopus inspired octobot which houses an onboard mono-propellant reactor with 50% aqueous hydrogen peroxide

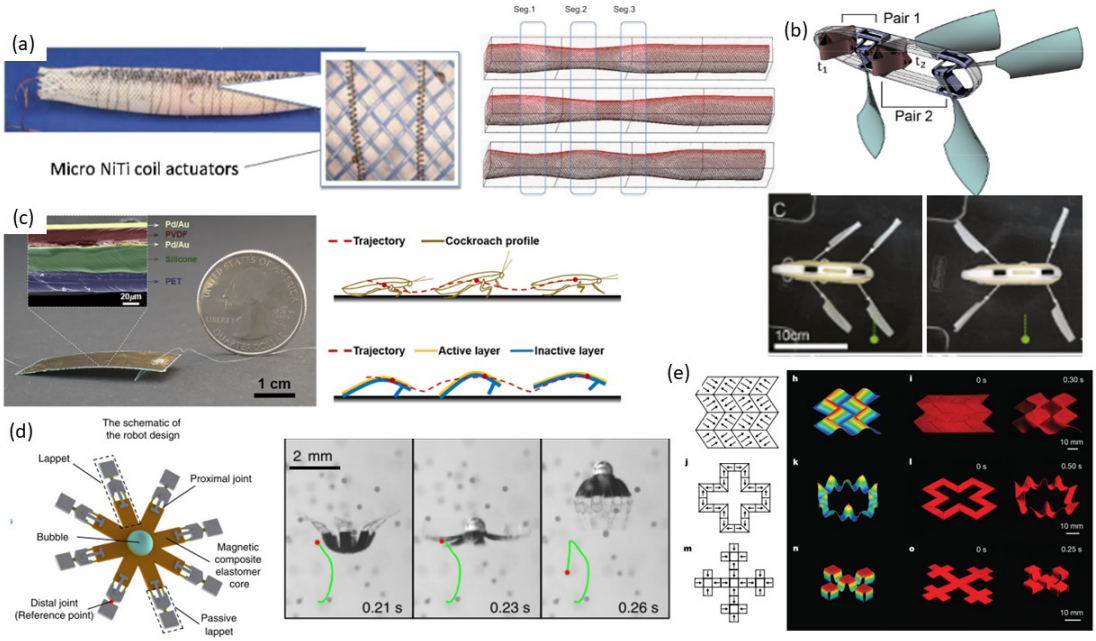

Figure 1.2: Smart Material actuated autonomous soft robots. (a) Earthworm-inspired Meshworm [143], (b) Bistability embedded swimming robot [12], (c) Cockroach inspired low profile robot [164], (d) a millimeter scale jellyfish-like robot [131], (e) 3D printed ferromagnetic soft material[70]

(H_2O_2) as fuel and platinum as catalyst. Fluidic elastomer actuators have enabled the control of an untethered autonomous soft robotic fish [67]. The propulsive forward swimming motion of the soft tail is achieved by alternate actuation of both sides of the fluidic actuator. Yaw control is achieved by retaining more average volume in one fluidic actuator half than the other and pitch control is achieved by adjusting the attack angle of the dive planes.

Another category of pneumatically actuated soft robots uses pneumatic air muscles, also called McKibben actuators/ Pneumatic artificial muscles (PAMs), which are akin to muscular hydrostats. OctArm [44, 161] is a continuum manipulator composed of high strain extensor actuators divided into multiple sections with rigid end-plates. Each section possesses 3 DoFs (two-axis bending and extension) and the total number of DoFs for OctArm is $3n$, where n is number of sections present. The OctArm can be mounted on a mobile platform and controlled wirelessly via joystick control. Additionally, draw wire encoders are used to provide the shape feedback for effective operation.

Smart materials are also used to design completely soft, untethered robots. Due to their low-profile nature, and in-built compliance they are an attractive option for soft robot actuation. For example, Meshworm [143] is an earthworm-inspired peristaltic crawling soft robot with braided

mesh-tube structure that is actuated by Nickel-Titanium (NiTi) coil actuators spread throughout the body. Under actuation, the braided mesh-tube structure provides antagonistic action of radial contraction and longitudinal extension. The robot uses two longitudinal muscles composed of NiTi coil spring actuators to incorporate steering capability. Researchers have also used Shape Memory Polymer (SMP) muscles to produce snap-through instabilities in bistable beam element and actuate an untethered, soft swimming robot with directional propulsion [12]. A fast and robust insect-scale soft robot is based on a curved piezoelectric Polyvinylidene difluoride (PVDF) unimorph structure. PVDF can produce periodic extension and contraction by the piezoelectric effect under an AC driving voltage to change the shape of the robot [164]. Ren et al. have developed a millimeter scale jellyfish-like robot that can generate multiple swimming gaits using its magnetic composite elastomer lappets under magnetic actuation [131]. 3D printing of programmed ferromagnetic domains in soft materials have enabled fast transformations between complex 3D shapes via magnetic actuation[70].

So far we have seen that pneumatic/hydraulic actuation is the most popular method to design and actuate completely soft, untethered robots, followed by smart materials. The major advantage of soft robots over their rigid bodied counterparts is their inherent compliance and variable stiffness; which makes tasks that were hitherto unavailable or difficult for traditional 'hard' robots look plausible. For example, manipulation and grasping of delicate objects, safe interaction with humans, and mimicking behavior of biological systems. The biological/ nature inspired design of these soft robots is however still lacking the control efficiency and intelligence we see in the natural systems. The control strategies and the actuation techniques used in these soft robots are still reminiscent of the principles used in traditional robotics. Most of these robots still use rigid internal or external electronic components to provide digital control architecture. The control actions/ commands are computed in an external digital controller/hardware. The compliant robotic structure merely responds to the actuation input applied and generates desired pre-programmed behavior. The performance of these soft robots is still limited to a narrow operation regime. Additionally, it is difficult to adapt to changing environmental conditions and the decision-making process is still limited to the external *brain* / processing unit.

There is a growing trend in soft robotics community to use all mechanical components, soft materials and/or systems that possess geometric non-linearity, and non-monotonous energy and force-displacement curves. The goal here is to create soft control systems with mechanical logic gates to minimize the presence of or to even get rid of rigid electronic components in the soft robotic

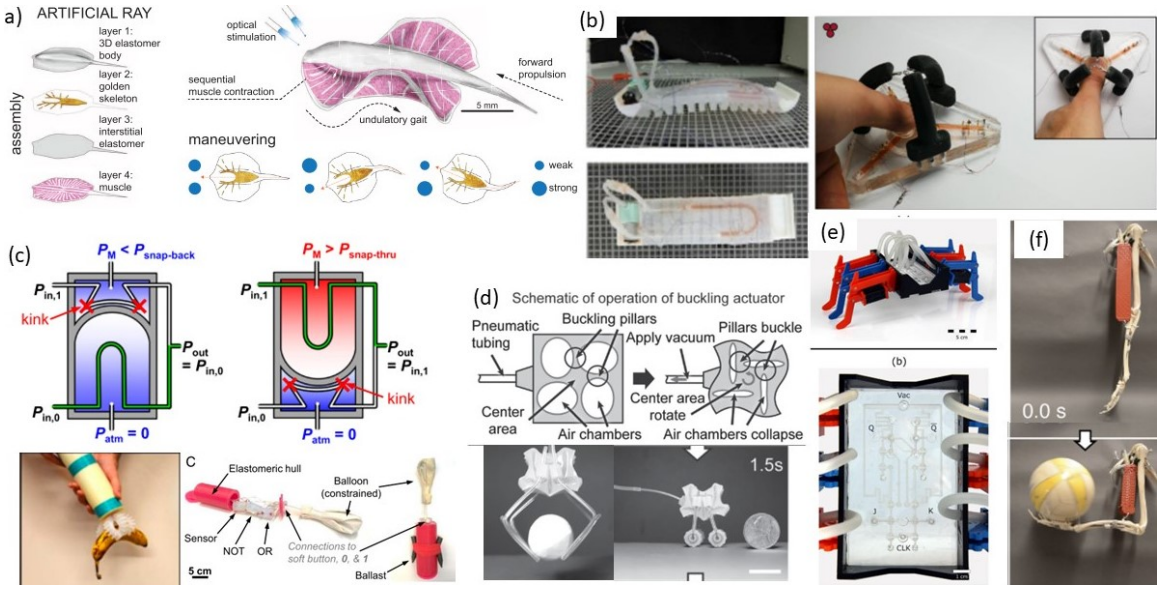

Figure 1.3: Soft controllers for autonomous soft robots. (a) Artificial phototactic ray [120], (b) Soft matter computer [40], (c) Bistable valve powered digital logic [126], (d) Vacuum powered buckling actuator [167], (e) soft hexapod robot for extreme environments[94], (f) Vacuum-actuated pneumatic muscle [168]

skeleton and/or digital processors. Such mechanical machines can still perform autonomously and under extremely harsh conditions where traditional electronic devices cannot be used. Additionally, researchers are looking at the ways to embed certain intelligence in the soft robots that will enable them to autonomously/ internally process information, perform computations and respond to the changing environmental conditions actively by generating the required control inputs.

In one interesting example, researchers designed a soft-robotic ray [120] inspired by stingrays and skates. The integrated sensory-motor system of the robotic ray is composed of four-layers: a three-dimensional elastomer [polydimethylsiloxane (PDMS)] body, a chemically neutral skeleton fabricated by means of thermal evaporation of gold through a custom designed shadow mask; a thin interstitial elastomer layer obtained by spin-coating,and a layer of optically responsive, live, rat cardiomyocytes generated via microcontact printing of fibronectin. When immersed in a 37°C Tyrode's physiological salt solution containing glucose as energy reservoir, and upon optical stimulation, the fabricated ray is propelled via the undulatory motion of its fins. In another example, researchers developed a soft matter computer (SMC) using a pattern of Conductive Fluid Receptors (CFRs) and insulating liquid as an input. This SMC can be directly integrated in robotic body to program peristaltic-like locomotion gait, reflexes for soft robotic gripper, and behavior switching in 2-DoF

soft actuator [40].

In an effort to design completely soft digital logic gates for soft robot embodiment, the researchers have started leveraging the non-linear material and geometric properties of soft and compliant mechanisms. For example, multi-stable and bistable mechanisms, inflatable fluidic actuators, and compliant buckling beams possess non-monotonous energy and force-displacement relationship. Under displacement/volume control, they undergo buckling/snap-through actions at constant total length/volume, that are accompanied by large geometric deformations, volume/pressure changes and spike in force exerted. Snap-through actions also cause the equilibrium path of multi-stable systems/ serially connected inflatable fluidic segments to switch from one branch to another. Under special conditions, the extension and contraction curves are different and result in a hysteresis-kind of loop that can be used to create periodic actuation cycles or for hardware sequencing. In the following paragraphs we focus on mechanisms that utilize bistability, multi-stability or buckling to design soft control for soft robots.

The Whitesides group designed a pneumatically activated bistable valve [126, 147, 125] that is composed of two cylindrical chambers separated by a hemispherical membrane. The air-supply tubes passing through both the chambers are connected to a common output. The valve has two stable states, unactuated and actuated. Chamber-1 is kept at atmospheric pressure while the pressure in the chamber-2 (P_M) is controlled via external application. When $P_M \leq P_{snap}$, the air supply tube of chamber-2 is kinked. Once the pressure in chamber-2 is increased above the snap-through pressure i.e. $P_M \geq P_{snap}$, the membrane snaps and kinks the air supply tube in chamber-1. So the output toggles between P_{atm} and P_M creating binary logic device and it is used to create complex logic gates. The soft logic gates are applied to a human-operated, completely soft gripper with toggle button, a semi-autonomous submersible robot, a soft ring oscillator for periodic actuation of soft robots.

Vacuum-operated buckling elastomeric actuators [167] are composed of a network of air chambers and elastomeric pillars. The application of vacuum collapses the air chambers inside the actuators, buckles the pillars and leads to central axis rotation. This feature is used to design soft grippers, swimmers and walkers. Mahon et al. developed a soft robot for extreme environments that combines fluidic switches and vacuum-actuated muscle to design electronic-free control architecture [94]. The soft robot uses fluidic switches, or transistors to create a logic for generating walk and grasp states. The legs are actuated with vacuum-actuated muscle-inspired pneumatic structures

that buckle under application of vacuum to generate linear motion [168].

Bifurcation based embodied logic [59] was demonstrated in a mechanical logic module fabricated by Direct ink writing–an extrusion-based 3D printing technique. This logic module can produce AND, OR, and NAND output in response to chemical inputs (water, toluene), depending on which structures are placed inside the module (hydrogel valve,PDMS-GF15 bistable unit, etc). Chalvet et al [11] designed a planar digital microrobot composed of four bistable modules that is used for precise positioning of its end-effector. Due to its multi-stability the microrobot doesn't need energy to maintain its position, but just for switching states of different modules.

Overvelde et al. designed a soft actuator that consists of serially connected inflatable fluidic segments (composed of latex tubes with stiffer outer braids) that undergo snap-through instabilities under volume controlled actuation[117]. These instabilities arise when individual segments with highly non-linear and non-monotonous pressure-volume relationship are connected together and lead to rapid change in the length and volume of the individual segments at constant total volume. The equilibrium path followed by such systems under inflation and deflation is different and makes a hysteresis-like loop that can be used for various applications [97]. This principle is also used for designing a pneumatically actuated, autonomous, quadruped compliant robot with embedded actuator sequencing [42]. They showed that such embedded hardware sequencing can reduce the input to a single fluidic supply tube and completely eliminate the internal valves with electrical tethers.

These non-traditional control strategies integrate the soft robot morphology, actuation and control architecture to develop *mechanically intelligent* soft robots. This new paradigm in soft robotics research can lead to the design of completely soft autonomous, untethered robots. These robots can potentially sense the environment around them and react/ or take decisions dynamically, leading to truly *intelligent* soft robots. Inspired by these advances we introduce two novel methods to *embed* hybrid mechanical-digital control in soft robotic skeleton, namely multi-stability for motion-sequencing and Physical Reservoir Computing (PRC) with origami. The soft robots in this research work are designed with origami and compliant mechanisms. We describe the research motivation and questions in detail in the final section of this chapter.

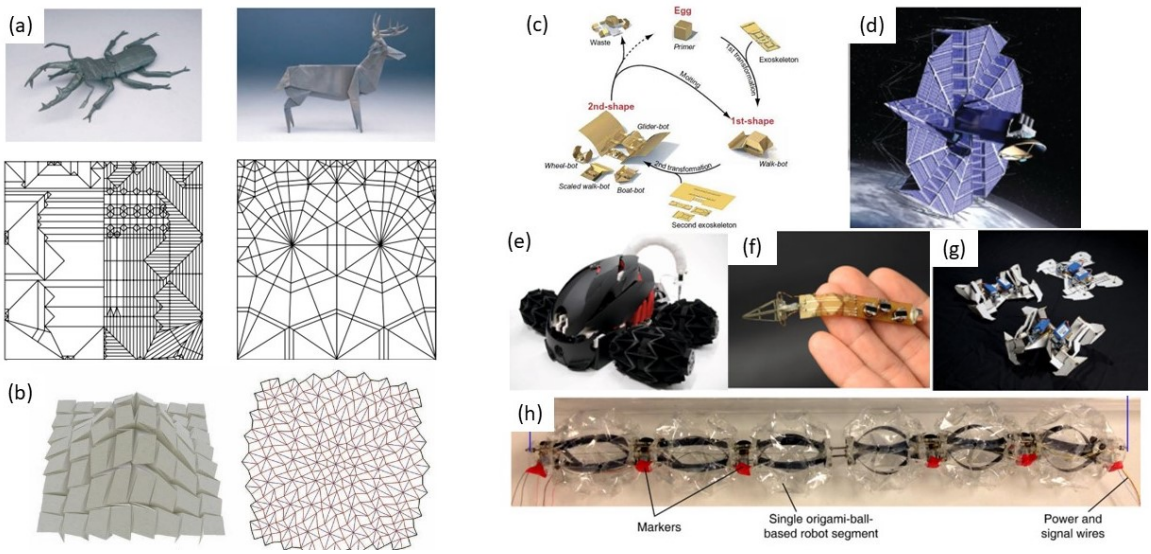

Figure 1.4: Origami applications in engineering. (a) Origami by Robert Lang, (b) Freeform Origami Tessellations by Generalizing Resch's Patterns[149], (c) Robotic origami metamorphosis [100], (d) Deployable solar panel [174], (e) origami wheels for performance on unstructured terrain, (f) wireless-powered gripper for medical applications [7], (g) self-assembling robot for search and rescue [28], (h)Origami-based earthworm-like locomotion robots [26]

1.2 Origami Mechanics for Soft Robotic Control

Origami is an ancient art of paper folding, in which a flat sheet of material is folded at pre-defined places called creases to create complex 3-D structures and mechanisms. The recent advances in computational mathematics and algorithms have helped origami evolve into a sophisticated art-form that boasts a deep connection with geometry and mathematics. In 2000s, researchers started studying ways to design automated/ computational geometric folding algorithms [73, 18, 95] for virtually any object. That led to the design of freeform origami ie. origami patterns or tessellations to create any 3-D shape [148, 19]. Structural engineers also started studying origami for applications in deployable mechanisms [174, 140, 32, 54, 72, 45]. In the ideal/ mathematical origami flat sheet of material is of zero thickness, but real origami is composed of materials with finite thickness. So researchers started to modify ideal origami for creating compatible thick origami designs [174, 13, 174]. Additional efforts are underway to study the folding mechanics via truss-frame based formulations that can capture the non-linear deformations and folding motion accurately[138, 30, 86].

Origami mechanisms are inherently lightweight, compact, and compliant, which makes them ideal candidate for designing soft robot skeletons. The origami mechanisms can exhibit many

unique properties such as – multi-stability, non-linear stiffness, negative Poisson's ratio and multi-transformability – which are not found in nature [139, 84]. Thus, origami has found multitude of applications in engineering and architecture. Origami-inspired and origami-derived designs have started to appear in spacecraft solar panels [174], biomedical robots [63], vibration isolation applications [134, 136], energy harvesting applications [171], self-folding robots [133, 99, 28], crawling robots [118, 26, 5, 130],programmed pop-ups in kirigami shells [129], DNA origami[162], mechanical logic gates [152, 112], shape morphing [80], programmable meta-materials [32, 116, 22, 144, 84, 66, 111], etc.

As explained in previous section, the multi-stable origami mechanisms also undergo buckling/ snap-through actions. Some bistable and multi-stable origami patterns are – square twist [145], Kresling [60] and its variant Flexigami [110] and reentrant Tachi-Miura Polyhedron (TMP) origami [169], creased conical surface[77], waterbomb base [46], origami hyperbolic paraboloid or hypar [34], and generic origami metasheet[160]. The pop-through defects created in the miura-ori meta-materials were applied to design re-programmable meta-materials with varying modulus [144]. Researchers have implemented mechanical logic gates via differential humidity-driven folding of waterbomb base origami [152]. In another example, researchers used magnetically actuated bistable Kresling to design Schmitt Trigger based logic gates [112].

We propose to use the non-linear mechanics of folding to design embedded control for soft origami robots. We started by examining the performance of fluidic origami cellular structures, since pneumatic actuation is most popular method for actuation in soft robotics. In the study presented in Chapter 2 we developed the truss-based lattice-framework for analyzing origami mechanisms. Using the correct stiffness formulation is one of the most important aspect for comparing the theoretical formulation with the experimental results. Next, we studied origami mechanisms that possess bistability and multi-stability, as they are ideal for design of soft robotic skeletons. Chapter 3 describes how we can hard-code/embed a periodic actuation cycle in a Kresling origami based multi-stable crawling robot for peristaltic-like locomotion gait generation. Next section introduces the concept of physical reservoir computing i.e. performing complex dynamic computations with physical bodies. In chapter 5 we show that origami meta-materials can be used as dynamic reservoirs and we propose design of soft origami robots with embedded physical reservoir computing power.

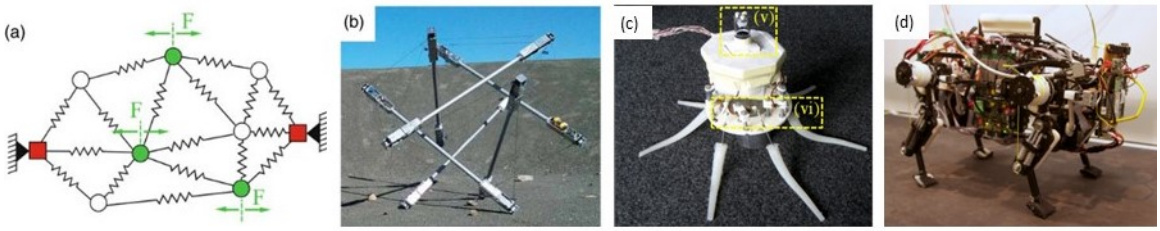

Figure 1.5: Physical Reservoir Computing for Soft Robotic Control. (a) Generic mass-spring-damper reservoir [49], (b) Tensegrity robot [9], (c) Octopus-inspired soft robot [14], (d) Compliant quadruped robot [17]

1.3 Physical Reservoir Computing for Soft Robotic Control

The natural world is a great source of inspiration for the field of soft robotics [98, 76]. For example, researchers have developed soft, compliant robots that have mimicked – locomotion of octopus [14],grasping of elephant trunk [47],flying of insects [90], swimming of jellyfish [64, 131] and fish [67], crawling of snakes [130],insects [164] and earthworms [143], etc. These biological systems all sport a a soft, compliant, variable stiffness skeleton with non-linear dynamic properties, and a complex, highly interconnected network of sensors (neurons) and actuators (muscles) spread throughout the body. The complex interplay between the brain, the sensor-actuator network and the body enables an animal to interact with the environment, process information, take appropriate decisions, and perform various tasks efficiently. Thus, the morphology of a body affects its actuation, control, and ultimately the "brain's" decision-making process. Therein lies the success of biological systems brought about by millions of years of evolution.

Similar to the biological systems soft, compliant robots possess virtually infinite degrees of freedom, non-linear dynamic properties with large non-linear deformation patterns, and an array of actuators and sometimes sensors to facilitate control. The current state-of-the-art soft robots however lack the high level of interactivity/information processing that exists between the robot actuation-control system and its state/sensory system. Traditionally in robotics every effort is made to reduce the effect of higher-order geometric and material non-linearities. But taking cues from the workings of natural systems, an increasing number of researchers are now embracing the *undesirable* nonlinear dynamics of soft and compliant bodies as computational resource to effectively create an embodied intelligence and control [121, 122, 9, 49, 108, 150]. Thus, a new computational paradigm called *morphological computation*[121, 122, 49, 9] has emerged in which the physical body of the

robot itself takes part in performing low level control tasks, such as co-ordination and modulation of locomotion gait simplifying the central controller architecture.

Reservoir computing (RC) is a computational framework based on artificial recurrent neural networks (RNNs) [56, 57, 93, 92, 91, 89, 142, 150, 106]. RNNs are extensively applied to the problems involving time-series prediction such as stock market and weather forecasting, robot gait and/or manipulator motion planning, robotic control, text and speech recognition, etc. A recurrent neural network is an artificial neural network where the output of the current time step depends on the output of the previous time step, in addition to the current input. Since RNNs involve forward as well as the back-propagation of input data, training them became a very difficult task. To circumvent these difficulties, the concept of a fixed recurrent neural network was independently introduced by Jaeger [56] as Echo State Networks (ESNs) and Maass [93] as Liquid State Machines (LSMs). In the foreword of a book called 'Reservoir computing' Prof. Jaeger credits Kirby and Day [71] as RC pathfinders [107]. In the RC setting, the neural network is randomly generated with fixed interconnections and input weights, and only the output readout weights are updated using simple techniques such as linear or ridge regression. The network dynamics are governed by non-linear dynamical equations, which transforms the input data stream into a high-dimensional non-linear state-space that can capture the non-linearities and time-dependent information in the input stream. This fixed RNN is called a *reservoir* as it remains fixed throughout the computation and this field of research has become known as *Reservoir Computing*.

Paul [121, 122] investigated the link between morphology and control for biped locomotion and used morphology as a basis to develop tensegrity robots with reduced control requirements. Taking this concept even further, Hauser et al. [49, 50] proved that a generic network of spring-mass-damper system with non-linear spring and damper properties can be used as a morphological computation device. In contrast, Caluwaerts et al. [10, 9] showed that geometric non-linearity of tensegrity structures can be the basis of a dynamic physical reservoir without the need for non-linear springs and dampers. Researchers have shown that physical dynamical systems can be used as computational reservoirs to perform complex computation tasks such as approximating non-linear dynamical systems [10, 9, 49, 50, 108, 150], pattern generation [10, 9, 49, 50, 108, 150], generating and controlling robot locomotion gait [9, 150, 17, 1, 14], and even for speech recognition [29] and machine learning[109, 150, 106, 102], etc. This paradigm is known as *Physical Reservoir Computing* (PRC).

We propose the use of origami meta-materials as dynamic physical reservoir, to leverage the non-linear geometry and mechanics of folding. In Chapter 5 we demonstrate the reservoir computing power of origami through multiple numerical experiments. We also demonstrate that origami reservoir can perform emulation task through a proof-of-concept experiment.

1.4 Research Objective and Questions

In this section, we will formally define the research objective and frame the research questions that will be answered throughout this research work.

Research Objective:

Developing design and analysis framework for hybrid mechanical-digital control of soft robots: from mechanics-based motion sequencing to physical reservoir computing

Inspired by the advances in novel techniques for autonomous soft robotic control, we propose the use of non-linear material and geometric properties of origami and compliant mechanisms to create embedded actuation and control of soft robotic locomotion. We examine and compare two different approaches for designing soft robots with embedded intelligence and computational power. The research work is divided into two major topics:

1. Harness multi-stability of origami and compliant mechanisms for mechanics-based motion sequencing in soft robots.

   In this approach the motion-sequencing is hard-coded or embedded in the origami robot skeleton. This approach is expected to reduce the control requirement drastically as the robotic skeleton itself takes part in performing low-level control tasks.

   However, such hard-coding does have its limitations. For example, we cannot add bending capability and completely different mechanism has to be added to switch the gait from straight line motion to bending motion. The robot cannot sense and react to the external disturbance in such type of control and any malfunction leads to failure.

2. Develop framework for the design of Physical Reservoir Computing embedded soft origami robots with distributed sensor and actuator network

To overcome the short-comings from hard-coding/ embedding particular kind of behavior in the soft robot, we examine PRC based approach for soft robot design. In this approach, a distributed network of actuator and sensors is used instead of minimizing their use. Multiple behaviors can be embedded in the robot simply by changing the output readout weights for corresponding behavior. The training process generates robust closed-loop behavior, so that small external disturbances can be resisted and robot performs well even in presence of noise. Since, the robot is composed of a network of actuators and sensors, it can sense the surrounding environment and can be trained to respond to the external inputs by switching or modulating the behavior.

Research Questions:

We addressed following questions through the course of our research work:

1. How to examine the actuation performance of fluidic origami cellular structure through theoretical analysis, and validate FEA and experimental results?

   Methods:

   - Formulate Miura-ori design and rigid-folding kinematics and use virtual work principle to calculate the final/steady-state folding angle at a given internal pressure.
   - Develop a quasi-static equivalent truss-frame model to discretize the continuous fluidic origami cellular structure into a network of pin-jointed stretchable truss elements with torsional spring stiffness assigned at the folding and facet bending creases.
   - Develop equivalent stiffness parameter formulation to correctly estimate truss stretching stiffness, crease folding stiffness and facet bending stiffness from 3D printed extension and contraction fluidic origami prototypes.
   - Perform parametric analysis to study the correlation between actuator performance metrics (free stroke and blocked force) and underlying Miura-ori design.

2. How to design Kresling origami based multi-stable robotic driving module?

   Methods:

- Design a generalized Kresling origami pattern to accommodate non-zero material thickness and increase the design space available for tailoring the kinematics of peristalsis crawling.
- Experimentally characterize the bistability of Kresling segment and multi-stability of the compliant driving module composed of two serially connected Kresling segments.

3. How to design compliant mechanism based multi-stable robotic driving module?

   Methods:

   - Use Pseudo Rigid Body Modeling (PRBM) to synthesize the compliant multi-stable mechanisms.
   - Experimentally characterize the bistability and tri-stability of compliant segments and multi-stability of the compliant driving module composed of two serially connected tri-stable and bi-stable segments.

4. How to exploit the multi-stability embedded in Kresling origami/ compliant mechanism based robotic skeleton to generate coordinated locomotion gait?

   Methods:

   - Formulate a potential energy-based optimization algorithm to identify the equilibrium deformation trajectories of multi-stable mechanisms.
   - Apply the optimization algorithm to the Kresling origami/ compliant mechanism based driving module to generate a deterministic actuation cycle.
   - Design passive foldable anchors that utilize the deformation of individual segments of the driving module to imitate earthworm *setae* that anchor to the surrounding environment and move the robot body forward.
   - Combine the actuation cycle with foldable anchors, to embed peristaltic-like crawling locomotion gait in the soft robot, that needs a single linear actuator and no digital controllers to control the individual segment deformation.
   - Design the experimental setup and fabricate the compliant robot working on these principles, to validate the peristaltic-like locomotion induced by multi-stability.

- Perform parametric analysis to uncover the correlations between peristalsis gait length and Kresling origami design.

5. Prove that origami meta-materials can be used for Physical Reservoir Computing (PRC) through extensive numerical simulations.

   Methods:

   - Develop lattice-framework based algorithm to analyze the dynamics of origami. This algorithm needs to work for any origami for which initial crease pattern is given. Implement gravitational, damping and external/ actuation forces and boundary conditions/ constraints.
   - Design the reservoir computing framework for an origami based reservoir.
   - Perform benchmark reservoir computing tasks (e.g. emulation, pattern generation, output modulation) to prove that origami can be used as dynamic physical reservoir.
   - Perform parametric analysis to uncover the correlations between reservoir computing performance and reservoir design parameters (underlying origami design, material parameters, and actuator-sensor distribution).

6. Demonstrate origami meta-materials can be used for Physical Reservoir Computing (PRC) through proof-of-concept experiments.

   Methods:

   - Develop experimental setup to perform benchmark reservoir computing task of emulation.
   - Perform multiple experiments with origami reservoir.
   - Perform additional analysis with techniques such as multiplexing on the data gathered from emulation task to improve the performance of reservoir.

7. How to design PRC embedded origami crawling and quadruped soft robots?

   Methods:

   - Enhance the dynamic model to simulate ground reactions and friction forces.

- Design origami pattern for earthworm-inspired crawling robot and the quadruped robot with optimum actuator-sensor network distribution.
- Implement PRC for forward locomotion gait generation in the soft robots and study the performance of the robot under closed loop.

Chapter 2

Actuation performance of fluidic origami cellular structure: Theoretical investigation

Abstract

Motivated by the sophisticated geometries in origami folding and the fluidic actuation principle in nastic plant movements, the concept of fluidic origami cellular structure was proposed for versatile morphing and actuation. The idea is to assemble tubular Miura-ori modules into a cellular architecture, and apply fluidic pressure to induce folding and hence actuation. Despite the promising potentials, the actuation capabilities of fluidic origami, such as free stroke and block force, are not elucidated. In particular, the effects of the thick facet material stiffness and pressure-sealing end caps are not understood. These gaps in our knowledge prevent the practical implementations of fluidic origami. Therefore, We constructed CAD models of the fluidic origami modules based on realistic design parameters to ensure that they can be fabricated via commercially accessible 3D printers while remaining pressure proof. In this study, we use the equivalent truss frame model to examine the actuation performance of fluidic origami actuators. We then comparing the results of theoretical analysis with the results obtained from FEA and experiments to reveal the influences of end caps and thick facet material stiffness. We also perform parametric analysis to study the

correlation between actuator performance metrics (free stroke and blocked force) and underlying Miura-ori design.

2.1 Introduction

Over the past several decades, the art of origami paper folding has been transformed into a design and fabrication framework for developing different sheet materials into sophisticated 3D shapes [73, 18, 95]. As a result, we are witnessing the rapid emergence of origami-inspired engineering applications spanning from the large-scale deployable aerospace structures [174, 140], kinetic architectures [32, 13], and self-reconfigurable robots [28, 99] [8,9] to the small-scale biomedical devices [63], mechanical meta-materials [139, 116, 22], and DNA machines [162]. These applications leverage the folding-induced shape reconfiguration to achieve their target performance, which can be tailored with an exceptionally large freedom by carefully designing the underlying crease patterns[148, 19]. This advantage is especially evident if the crease pattern is rigid-foldable, so that one can analyze the shape reconfiguration by treating the origami facets as rigid panels revolving around hinge-like crease lines – essentially a 3D linkage mechanism.

The versatile shape reconfiguration capability of origami is especially appealing for embedded actuation because folding can be used to program sophisticated actuation motion paths. Therefore, a fluidic origami cellular structure concept (referred simply as "fluidic origami" hereafter) was proposed by the authors via combining the origami geometry and the actuation principles of plant nastic movements [80]. Fluidic origami is essentially an assembly of tubular modules consisting of carefully designed, rigid-foldable Miura-ori sheets (Figure 1). When these modules are pressurized either pneumatically or hydraulically, they fold and generate effective actuation motion until the maximum internal volume allowed by rigid-folding is reached. Such a distributed, pressure activated actuation shares similar working principles to the rapid nastic movements in plants like Mimosa pudica, which can selectively manipulate the turgor pressure in its motor cells to create an internal bending moment for leaf folding [36, 20] (Figure 2.1(a)). Compared to other plant-inspired, pressure activated cellular structure concepts such as PACS [43], pressure adaptive honeycombs [158], and topologically optimized trailing edge [156, 83]; the fluidic origami has some unique advantages. Firstly, folding is a fundamentally three-dimensional shape reconfiguration that can enable complex motions like a combined extension and shear [22]. Secondly, the pressurized fluidic origami mod-

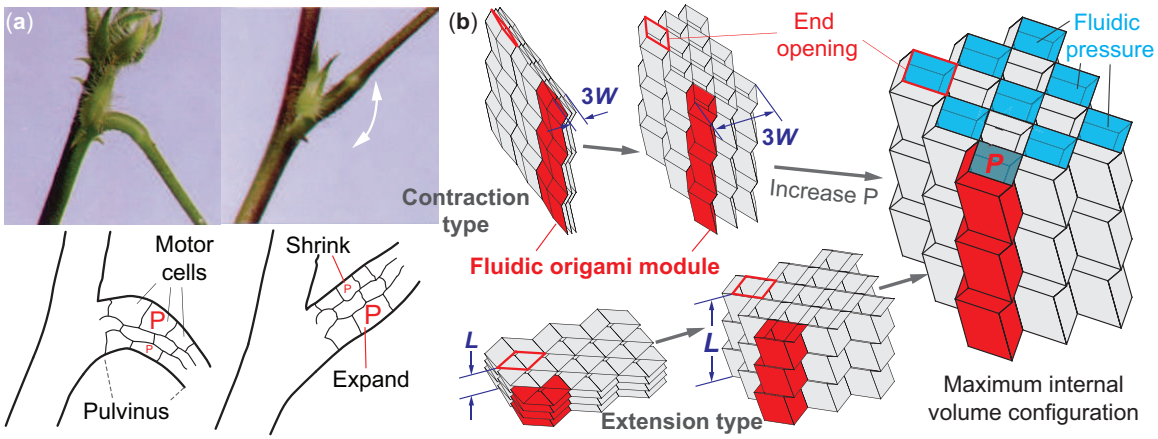

Figure 2.1: The concept of plant-inspired fluidic origami cellular structure. (a) The nastic movements in plants and (b) fluidic origami share the similar working principle: That is, the embedded and distributed pressurization in the cellular structure can enable versatile actuation motion. In the mimosa plant shown in (a), increase or decrease the turgor pressure can expand or shrink motor cells to bend the leaves. (Image adopted from [154] with permissions) In the fluidic origami shown in (b), an increasing internal fluidic pressure can induce large amplitude folding and generate actuation motion in both L and W directions. Note that the folding kinematics shown in this figure is based on idealistic model assuming rigid-folding, and the opening at the two ends of the fluidic origami modules changes their shapes significantly

ules can be seamlessly integrated with non-pressurized origami sheets for large-scale and efficient motions. Finally, the morphing and actuation capability of fluidic origami can be complemented by other adaptive functions such as stiffness adaptation [80] and pressure-dependent multi-stability [82]. The stiffness adaptation, which can be achieved near instantaneously via simple on/off valve control, can reduce the power requirements for actuation without sacrificing the external load bearing capacity. Meanwhile, the multi-stability can significantly amplify the actuation speed and magnitude similar to the impulsive trap closing motions in Venus flytrap [38]. Therefore, fluidic origami has great potentials to advance state-of-the-art of many engineering systems that require embedded actuation such as morphing airframe [157] and soft robots [75].

Despite these promising potentials, the actuation capabilities of fluidic origami such as free stroke and block force are not yet fully elucidated. Especially, previous studies by the authors relied on idealistic models assuming that 1) the origami facets have zero thickness, 2) the crease lines act like simple hinges, and 3) the tubular modules are sealed by ideal end caps that can perfectly accommodate the shape changes of their end openings. Indeed, similar assumptions have been made by many other studies on origami-inspired structures and materials. These idealistic models can reveal the working principles without unnecessary complexities, but they inevitably neglected

many important factors in practical implementations. In particular, additive manufacturing or 3D printing is considered as one of the most viable approaches to fabricate the complex geometries of multi-cellular fluidic origami while maintaining internal pressure sealing. Thus the finite thickness and compliance of the 3D printed facet and crease materials must be considered in order to examine the realistic actuation performance of fluidic origami. Furthermore, a realistic pressure-sealing end cap at the end openings of fluidic origami module can be incompatible with the folding-induced shape changes (Figure 2.1). Therefore, end caps can reduce the actuation performance and such negative effects must be analyzed carefully.

Therefore, the objective of this study is to conduct a holistic investigation of the actuation performance of the fluidic origami by incorporating realistic considerations in its design and fabrication. This study is conducted based on the CAD models of fluidic origami modules featuring finite facet material thickness and flat end caps. Various design variables, such as the material thinning along the crease lines, are carefully chosen to ensure that the fluidic origami modules can be fabricated via commercially accessible 3D printing machines. Two types of modules are designed based on their initial folding configurations, one contracts along longitudinal L direction under pressurization, and the other extends. Based on these CAD designs, we examine the free stroke and block force of fluidic origami module using two different approaches. The first approach relies on the simplified models used in authors' previous publication [80] and many other studies on origami-inspired engineering applications; and the second approach uses the more comprehensive finite element simulations. Comparing the results from these two different approaches can reveal the correlations between the fluidic origami actuation performance and many practical design considerations including Miura-ori geometry, facet thickness, material stiffness, and end caps. And understanding these correlations will eventually enable us to identify the optimal fluidic origami design. Therefore, this study can provide the practical guidelines for implementing fluidic origami as an active and adaptive structure.

The rest of this chapter is organized as follows. Section 2.2 discusses the design and kinematics of the fluidic origami module. Section 2.3 and 2.4 contains the in-depth analysis of its free stroke and block force performance, respectively. These two sections also include parametric analyses revealing the correlation between the actuation performance and the underlying Miura-ori design. Section 2.5 concludes this chapter with summaries and discussions.

2.2 Miura-ori design and folding kinematics

The backbone of a fluidic origami module consists of two identical Miura-ori strips connected along their zig-zag crease lines (Figure 2.2(a)). The design, stacking, and folding kinematics of the Miura-ori sheets have been extensively discussed in previous studies [139, 22, 21]. Here we only provide a brief overview for clarity. Miura-ori design is determined by three variables that remain unchanged regardless of folding: they are the crease length a, b and the sector angle γ between them. To describe the amount of folding, a folding angle θ is introduced as half of the dihedral angle between adjacent facets of the two connected Miura-ori strips (Figure 2.2(a)). According to rigid-folding kinematics that assumes rigid facets, hinge-like crease lines, and ideal end caps, the folding angle can take any values from 0° to 90° (Figure 2.2(b)). $\theta = 0°$ indicates that the Miura-ori is flat, and $\theta = 90°$ means it is fully-folded. The correlations among the external length L, width W, enclosed internal volume V, and the folding angle θ are strongly nonlinear as follows [79],

$$\begin{aligned} L &= \frac{2nb\cos\theta\sin\gamma}{\sqrt{1-\sin^2\gamma\sin^2\theta}} \\ W &= 2a\sin\theta\sin\gamma \\ V &= na^2b\sin^2\gamma\sin(2\theta) \end{aligned} \tag{2.1}$$

where n is the number of Miura-ori unit cells – the most elementary geometric identity – along the length of a tubular fluidic origami module. For example, the module shown in Figure 2 has three unit cells so $n = 3$. Clearly, the maximum volume always occurs when $\theta = 45°$ regardless of the Miura-ori design; and this is the critical, locking configuration at which further pressurization could not induce more actuation motion. Therefore, we can define two types of fluidic origami modules based on their initial resting folding angle θ. The first type, which features a resting folding angle less than 45°, slightly contracts in length L but significantly expands in width W under internal pressurization until the maximum volume configuration is reached (Figure 2.1(b)). For clarity, it will be referred as the "contraction type" hereafter. The second type of fluidic origami module has a resting folding angle bigger than 45°, so it extends significantly in length L but slightly contracts in width W (Figure 2.1(b)). We will refer it as the "extension type".

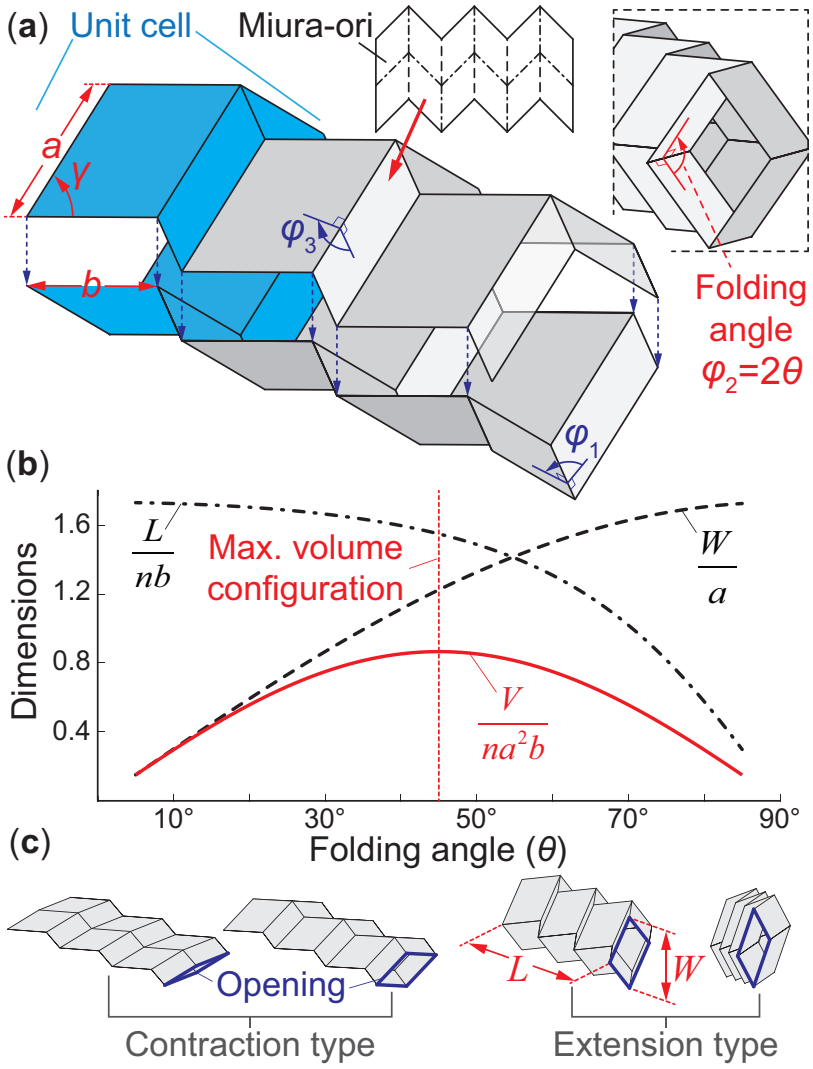

Figure 2.2: Design and ideal rigid-folding kinematics of a fluidic origami module. (a) A module consists of two identical Miura-ori strips connected along their crease lines. The most elementary unit cell and its important design parameters are highlighted. This particular module has three unit cells. The inserted figure at the top right illustrates the definition of folding angle θ. (b) The relationships among the folding angle, normalized length, width, and the internal enclosed volume according to the rigid-folding kinematics. In this figure$\gamma = 60°$. (c) Folded fluidic origami modules corresponding to $\theta = 5°, 25°, 65°, and 85°$ from left to right. Notice that the end opening of the fluidic origami exhibits significant shape changes

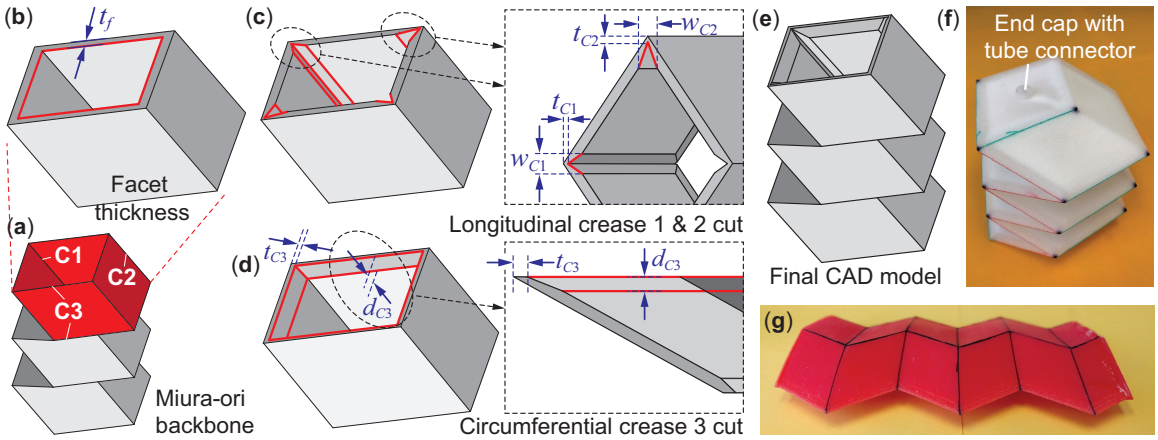

Figure 2.3: CAD design and 3D printing of the fluidic origami modules. (a, b) The facet thickness is modelled by an inward offsetting from the Miura-ori backbone geometry. (c, d) V-grooves are added along the crease lines to facilitate proper folding. (e) A finished CAD model for the fluidic origami module that includes the facet thickness and all crease thinning. This model was then sent directly to the 3D printers. (f) SLS printed, extension type of fluidic origami module using PEBA materials. (g) FDM printed contraction type of fluidic origami using TPU materials. Flat end caps are printed separately and glued to provide proper pressure sealing.

Table 2.1: Design parameters of the two fluidic origami prototypes shown in Figure 2.3 and Parametric studies

Design	**Parameters**	Extension	Contraction	Parametric study
Miura-ori backbone geometry	a	62.5 mm	46.9 mm	31 mm
	b	50 mm	37.5 mm	25 mm
	γ	60°	60°	60°
	θ^0	70.5°	16.1°	78°
	n	3	3	3
Additional design variables	t_f	4 mm	3 mm	4 mm
	t_{c1}	2 mm	1.5 mm	1 mm
	t_{c2}	1 mm	4.5 mm	1 mm
	t_{c3}	2 mm	2.3 mm	1 mm
	w_{c1}	8 mm	7.5 mm	4 mm
	w_{c2}	8 mm	1.5 mm	4 mm
	d_{c3}	3 mm	2.6 mm	2 mm

2.3 Free Stroke Performance of the Fluidic Origami

Free stroke and block force are the two most commonly used performance metrics for actuators. In this study, the free stroke S of the fluidic origami module is defined as the ratio of its averaged extension or contraction due to internal pressure over its initial dimensions, while no external loads are applied. For effective actuation, we examine the longitudinal deformation of the extension type of fluidic origami and transverse deformation of the contraction type. That is, $S_x = X_{final}/X_{initial} - 1$, where $X = L$ for the extension type and $X = W$ for the contraction type. The free stroke of fluidic origami is a function of internal pressure, and its magnitude increases monotonically with increasing pressure until the maxi-mum volume configuration is reached. Moreover, since the free stroke is associated to folding, its magnitude is dictated by the torsional stiffness of creases. We investigate the free stoke of fluidic origami assuming rigid facets and ideal end caps. Similar approaches have been used extensively for other rigid-foldable origami research. We also conduct a parametric study to reveal the correlation between free stoke and underlying Miura-ori designs.

2.3.1 Simplified approach for free stroke analysis

In this approach, we assume ideal end caps that can seal the internal pressure without hindering the shape changes of the end opening. In this way, the pressure-induced deformations of fluidic origami follow the rigid-folding kinematics defined in Equation 2.1, and the corresponding total energy can be approximated as the summation of crease strain energies and the work done by pressure:

$$U_t = \Pi_c + W_p = \sum_i \frac{1}{2} k_i^c (\phi_i - \phi_i^0)^2 - P(V - V^0) \tag{2.2}$$

where ϕ_i and k_i^c are the dihedral opening angles and torsional stiffness of the crease lines, respectively (Figure 2(a)). $\phi_1 = \pi - 2\theta, \phi_2 = 2\theta$, and $\phi_3 = 2\arcsin[\cos\theta(1 - \sin^2\theta\sin^2\gamma)^{(1/2)}]$. ϕ_i^0 and V^0 are initial crease angles and enclosed volume corresponding to the resting folding angle θ^0. The folding angle at a given internal pressure can be calculated by solving the following equation

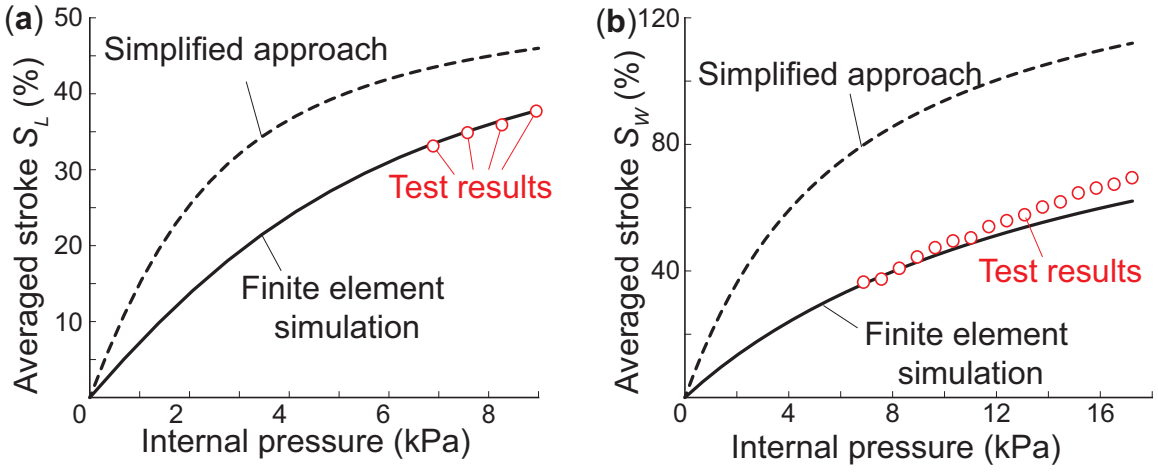

Figure 2.4: Free stroke analysis of the fluidic origami prototype. (a) Performance of extension type of fluidic origami prototype along the length direction (b) Performance of the contraction type of fluidic origami prototype along its width direction. Free stroke performance based on the simplified approach (shown in dashed line), FEA (shown in solid line), and experimental measurements (shown with red circles)

based on virtual work principle,

$$-P\frac{dV}{d\theta} + \sum_i k_i^c(\phi_i - \phi_i^0)\frac{d\phi_i}{d\theta} = 0 \tag{2.3}$$

The solution of θ can be inserted into Equation 2.1 to calculate the free stroke. It is evident from equations 2.2 and 2.3 that the free stroke performance is directly related to the crease torsional stiffness k_i^c in that stiffer crease lines will reduce the free stroke at a given pressure level. The crease torsional stiffness estimation for the 3D printed extension and contraction fluidic origami prototypes are detailed in Appendix A. The calculated free stroke magnitude with respect to pressure are shown as the dashed curves in Figure 2.4(a) and Figure 2.4(b).

2.3.2 Correlation between free stroke and underlying Miura-ori design

A unique advantage regarding fluidic origami is the possibility to program its actuation performance by tailoring the underling Miura-ori design variables. According to the rigid-folding kinematics, the normalized free stroke is only related to the sector angle γ and folding angle θ^0, therefore, we conducted parametric analyses to examine the correlations between the free stroke in length direction and these two crucial design parameters using TPU material properties. We

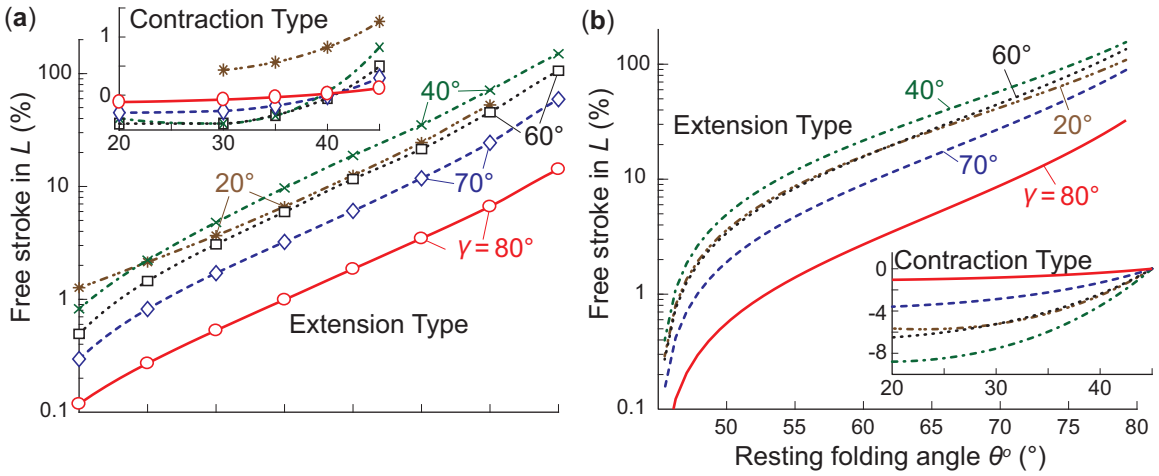

Figure 2.5: Parametric analyses illustrating the correlation between free stroke performance, the sector angle γ, and resting folding angle θ^0 of the underlying Miura-ori. (a) Results from the finite element analysis, where each maker is a simulation result. (b) Results based on the rigid-folding kinematics model. In both (a) and (b), the main figures show the free stroke of the extension type of fluidic origami with $\theta^0 > 45°$, and the small insert figures show those of contraction type. Note that the Y axes are in logarithmic scale in the main figures and linear scale in the inserted small figures.

compare the results of theoretical analysis with the results of FEA simulations (Figure 2.5). In this parametric analysis, other design parameters such as the crease line lengths and thinning parameters are chosen from a set of baseline designs listed in Table 2.2, and the internal pressure P is set at 34.5kPa. By carefully examining the results from this parametric study and the experiment results discussed in previous two subsections, we can come to the following conclusions regarding free stroke performance. 1) The extension type of fluidic origami actuators generate significantly more free stoke along the length direction than the contraction type. The simplified model and finite element simulations predict close to 100% of free stroke magnitude from the extension type, but less than 10% from contraction type. 2) Among the extension types of actuators based on different Miura-ori designs, those with resting folding angle close to 90° and sector angle close to 40° provide the largest free stroke. Especially, the free stroke increases monotonically as the resting folding angle θ^0 increases. 3) The overall trends between free stroke and Miura-ori designs are the same between the results from simplified model and finite element simulation. Therefore, the end caps do not qualitatively change the relationship between free stroke and Miura-ori designs; instead, they only reduce the magnitude of the achievable free stroke.

2.4 Block Force Performance of the Fluidic Origami

Block force is another important performance metric because it directly indicates the load carrying capacity of fluidic origami. In this study, block force is defined as the reaction force of the fluidic origami when its stroke is held at zero, that is, the two ends of the fluidic origami are fixed. To examine this performance metric without unnecessary complexities, we only investigate the block force of the extension type fluidic origami in the length direction L using the baseline designs in Table 3. We introduce a normalized block force B defined as follows:

$$B = \frac{F}{PA^0} \tag{2.4}$$

where F is the magnitude of the block force according to an internal pressurization P, and $A = 2a^2 \sin\theta^0 \sin\gamma\sqrt{1-\sin^2\theta^0\sin^2\gamma}$ is the cross-section area of the fluidic origami at the initial resting configuration. Unlike the free stroke, the block force is more closely related to the facet material stiffness rather than crease torsional stiffness. Therefore in this section, we examine the block force by using an equivalent truss-frame model that has been used extensively by many other studies on origami-inspired structures and materials [32, 139, 80, 30]. A parametric study is also conducted and discussed.

2.4.1 Block force analysis based on the truss frame model

The equivalent pin-jointed truss frame model used in this approach essentially converts the continuous origami into a discrete system with a finite degrees of freedom. In this model, the creases are represented by stretchable truss elements, and the facets are triangulated along the short diagonals with additional truss elements to provide a first order estimation of their bending. Torsional spring stiffness are assigned to the dihedral angles defined by the truss elements along the creases and across facets to represent the crease torsional and facet bending stiffness, respectively (Figure 2.6(a)). In this way, the truss frame model can establish the connection between block force and facet compliance while assuming ideal end caps.

Three geometric transformation matrices and vectors are necessary for analyzing the pressure-induced block force based on this truss frame model. They are 1) a compatibility matrix $\mathbf{C}$ correlating the vector of pin-joint displacements $\mathrm{d}\mathbf{x}$ to the vector of truss member stretches $\mathbf{e}$ so

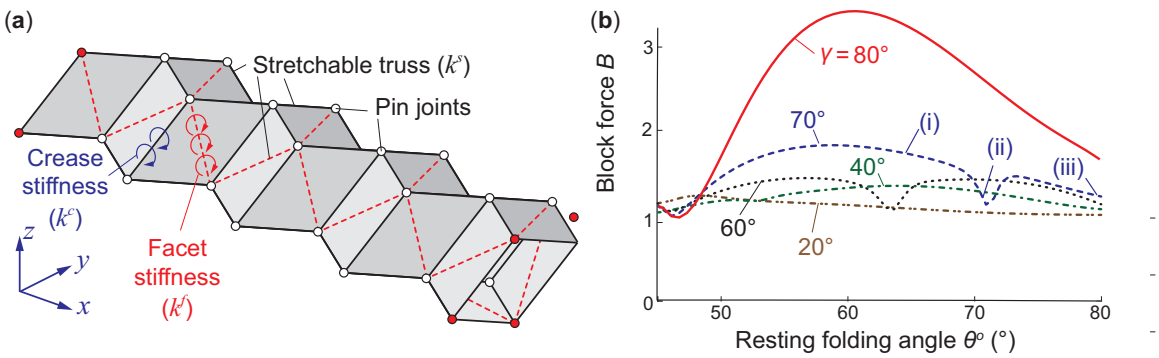

Figure 2.6: Block force analysis of the fluidic origami. (a) Illustration of the equivalent truss-frame model. The solid lines are truss elements representing the crease lines, and dashed lines are truss elements that triangulate the facets. Pin joints at the ends are highlighted as solid circles, and their displacements along the x-axis are set to zero. (b) Parametric analyses illustrating the correlations between block force performance, the sector angle γ, and the resting folding angle θ^0 of the underlying Miura-ori. Besides γ and θ^0 , other fluidic origami designs are chosen from Table 3, and the pressure is set at 34.5kPa. Some curves based on the truss-frame model show notable dips regarding the block force magnitude, one of them is labeled by (ii) as an example. These dips are related to the occurrence of non-uniform deformation. Here, (i),(ii), and (iii) are three fluidic origami designs with the same sector angle but different resting folding angles, and their deformation patterns under pressure are shown in detail in Figure 2.7

that $\mathbf{e} = \mathbf{C}d\mathbf{x}$; 2) a transformation matrix $\mathbf{J}$ correlating d$\mathbf{x}$ to the vector of crease angle changes $d\phi$ so that $d\phi = \mathbf{J}d\mathbf{x}$; and 3) a row vector $\mathbf{D}$ correlating d$\mathbf{x}$ to the internal volume change dV so that $dV = \mathbf{D}d\mathbf{x}$ The total stiffness matrix $\mathbf{K}$ of the fluidic origami truss frame model is the summation of the facet stretching/shear stiffness K^s and the bending stiffness K^b. K^s equals to $\mathbf{C^T\Lambda C}$ where $\Lambda^s = diag(k_1^s, k_2^s, ..., k_n^s)$ is a diagonal matrix containing the equivalent stretch stiffness k_i^s of the truss elements; $\mathbf{K}^b$ equals to $\mathbf{J}^T\mathbf{\Lambda}^b\mathbf{J}$ where $\mathbf{\Lambda}^b = diag(k_1^c, k_2^c, ..., k_m^c, k_1^f, k_2^f, ..., k_p^f)$ is a diagonal matrix containing the equivalent crease torsional stiffness k_i^c and facet bending stiffness k_i^f. The derivation of $\mathbf{C}$, $\mathbf{J}$, and $\mathbf{D}$ matrices have been discussed extensively in previous publications [80, 30], and the necessary details for calculating $k_i^s, k_i^c$and, k_i^f are provided in Appendix A for clarity. To analyze the block force while assuming ideal end caps, the longitudinal displacements of the end nodes are assumed zero, however, their transverse displacements are not constrained (Figure 9(a)). The vector of vertices displacement dx can be calculated as a function of internal pressure:

$$d\mathbf{x} = P\mathbf{K}^{-1}\mathbf{D}^T \tag{2.5}$$

and the reaction block force can be calculated as the summation of the internal forces from the truss

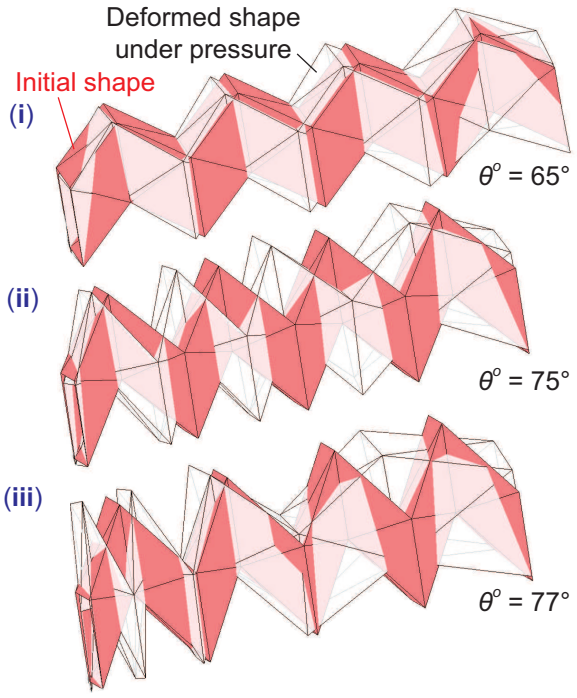

Figure 2.7: The non-uniform deformation pattern of the fluidic origami based on truss-frame model at 34.5kPa of internal pressure. All three fluidic origami modules shown here share the same sector angle $\gamma = 70°$, and their corresponding block force performance are labeled in Figure 2.6. It is evident that fluidic origami module with a higher resting folding angle θ^0 shows a stronger non-uniform deformation, by which some unit cells contract in their length direction and other cells extend significantly

stretches onto the end nodes and the pressure acting on cross-section area at the ends.

2.4.2 Block force v/s Miura-Ori designs

Similar to the free stroke, the block force performance of the fluidic origami module is directly related to the underlying Miura-ori designs, such relationship can be illustrated by the parametric study results in Figure 10. The magnitudes of the normalized block force vary significantly as the sector angle and resting folding angle change. By examining the parametric analysis results from both truss frame model and finite elements simulations, we can come to the following conclusions regarding block force performance. 1) Fluidic origami modules with $\gamma = 80°$ generates higher block force than other tested sector angles. In particular, those with $\gamma = 40°$ perform relatively poorly regarding block force, even though they generate a large free stroke as shown in Figure 2.5. 2) The block force performance peaks when the resting folding angle θ^0 is designed in the range between 60°

and 70°. This is also different from the free stroke analysis results, which recommends a close to 90° resting folding angle for large free stroke. 3) End caps do not qualitatively change the relationships between the block force and Miura-ori designs. However, they significantly reduce the magnitude of achievable block force.

Besides the block force magnitude, another significant difference between the truss-frame model results and FEA simulation is the deformation pattern of fluidic origami. The truss-frame model predicts that the fluidic origami with higher resting folding angle θ^0 deforms non-uniformly, that is, unit cells at one end of the fluidic origami module contract in the length L direction, while the cells at the opposite end elongate significantly (Figure 2.7). This is because at higher folding angles, the fluidic origami is close to be fully folded; its creases and facets are oriented in a way that can easily accommodate the non-uniform deformations. That is, non-uniform deformations would primarily invoke crease folding without inducing much facet deformations. A similar phenomenon was also studied based on eigenvalue analysis in previous literature [32, 80], where interested readers can learn more about the underlying physical principles. The finite element results, on the other hand, do not show such non-uniform deformations. In the truss-frame model, the end caps of fluidic origami module are assumed ideal, that is, they can seal the internal pressure and accommodate the shape changes from folding. Moreover, the facets are assumed to have zero thickness. However in the finite element model, the end caps are no longer ideal and the facets have finite thickness. The non-ideal end caps and thick facets impose additional constraints to the deformation of fluidic origami, preventing the non-uniform deformation from occurring.

2.5 Summary and Conclusion

Via analytical investigation, finite element simulation, experiment validation, and design optimization, this study holistically examines the actuation performance of a plant-inspired fluidic origami cellular structure. In particular, we aim to understand the influences of thick facet material stiffness and pressure sealing end caps in order to obtain practical guidelines for implementing the fluidic origami. To this end, we construct CAD models of the fluidic origami modules based on realistic design parameters to ensure that they can be fabricated via commercially accessible 3D printers while remaining pressure proof. Two fluidic origami prototypes based on different designs, 3D printing methods, and materials are fabricated for experimental validation.

We then use two different approaches to examine the free stroke and block force performance of fluidic origami. The first approach is to use simplified analytical models that assume zero facet thickness and ideal end caps. In particular, for free-stroke analysis we use a model based on rigid-folding kinematics; and for block force analysis we use an equivalent truss-frame model. These analytical models have been used extensively for the previous studies of origami applications. The second approach is to use the more comprehensive nonlinear finite element simulation. Comparing the results from these different approaches can reveal the influence of thick facet material stiffness and realistic end caps. It is found that the thick facets and end caps reduce the magnitude of free stroke and block force. They also alter the deformation pattern of fluidic origami under pressure. That is, in the free-stroke analysis, the end caps induce localized bending deformation at each end of the fluidic origami module; in the block force analysis, the thick facet and end caps prevent the non-uniform elongation and contraction predicted by the truss-frame model. However, thick facet and end caps do not qualitatively change the relationships between the actuation performance and the underlying Miura-ori design.

Based on these insights, we developed a customized generic algorithm, based on the finite element model, to identify the optimal fluidic origami designs for actuation. We find an optimal resting folding angle to maximize the actuation capability, while the sector angle in Miura-ori can be tailored to effectively program the actuation performance. Therefore, this study provides the practical guidelines for implementing fluidic origami for many applications that require embedded actuation such as morphing air-frame and soft robots.

Chapter 3

Peristaltic locomotion without digital controllers: Exploiting multi-stability in origami to coordinate robotic motion

abstract

This study examines a novel approach to generate peristaltic-like locomotion in a segmented origami robot. Specifically, we demonstrate the use of multi-stability embedded in the origami skeleton to eliminate the need for multiple actuators or digital controllers to coordinate the complex robotic movements in peristaltic crawling. The crawling robot in this study consists of two serially connected bistable origami segments, each featuring a generalized Kresling design and a foldable anchoring mechanism. Mechanics analysis and experimental testing reveal that the nonlinear elastic behaviors of this dual-segment module, especially its rapid deformation due to the non-monotonic energy landscape and force-displacement relationship, can create a deterministic deformation sequence or actuation cycle. This cycle can then be used to generate the different phases in a peristaltic-like locomotion gait. Instead of individually controlling the segment deformation like in earthworm and

other crawling robots, we only control the total length of this robot. Therefore, this approach can significantly reduce the total number of actuators needed for locomotion and simplify the control requirements. Moreover, the richness in Kresling origami design offers us substantial freedom to tailor the locomotion performance. The results of this study will contribute to a paradigm shift in how we can use the mechanics of multi-stability for robotic actuation and control.

3.1 Introduction

Limbless and metameric invertebrates like the earthworm use peristalsis to crawl over uneven surfaces, burrow through soil, and navigate in confined spaces with ease. The body of an earthworm consists of many segments that are grouped into several "driving modules". Each module includes three types of segments according to their states of deformation: "contracting", "anchoring", and "extending" [127] (Figure 3.1(a)). In a peristaltic locomotion cycle, the contracting segment expands in diameter and contracts in length by engaging its longitudinal muscles (Figure 3.1(b)). The extending segment deforms oppositely by engaging its circular muscles. When a contracting segment reaches the fully-contracted shape, it becomes an anchoring segment, which can firmly attach itself to its surrounding by further deploying hair-like bristles (aka. *setae*) on its surface. By carefully *coordinating* the deformation of its segments, the earthworm can generate a retrograde peristaltic wave that propagates towards the tail end of its body, thus driving itself forward (Figure 3.1(a)).

The locomotion performance of a peristaltic gait is easily tunable by changing the number of these three types of segments in a driving module [24, 25]. The absence of complex external appendages like legs or wings makes the driving module design compact and light. As a result, peristaltic locomotion has been implemented in many worm-inspired crawling robots for field exploration and in-pipe inspection. However, these robots typically require many actuators—such as pneumatic chambers [8, 65, 173], shape memory alloy (SMA) springs [26], electric motors [27], or permanent magnets [137]—to activate their segments *individually*. Moreover, a complicated control architecture is also necessary to coordinate the individual segment deformation to achieve peristaltic locomotion (Figure 3.1(c)). This can lead to a cumbersome mechatronic setup that can significantly constrain the overall application potential, especially when these robots need to be completely soft and un-tethered [132].

To address this issue, we examine the use of non-monotonic energy landscape and force-

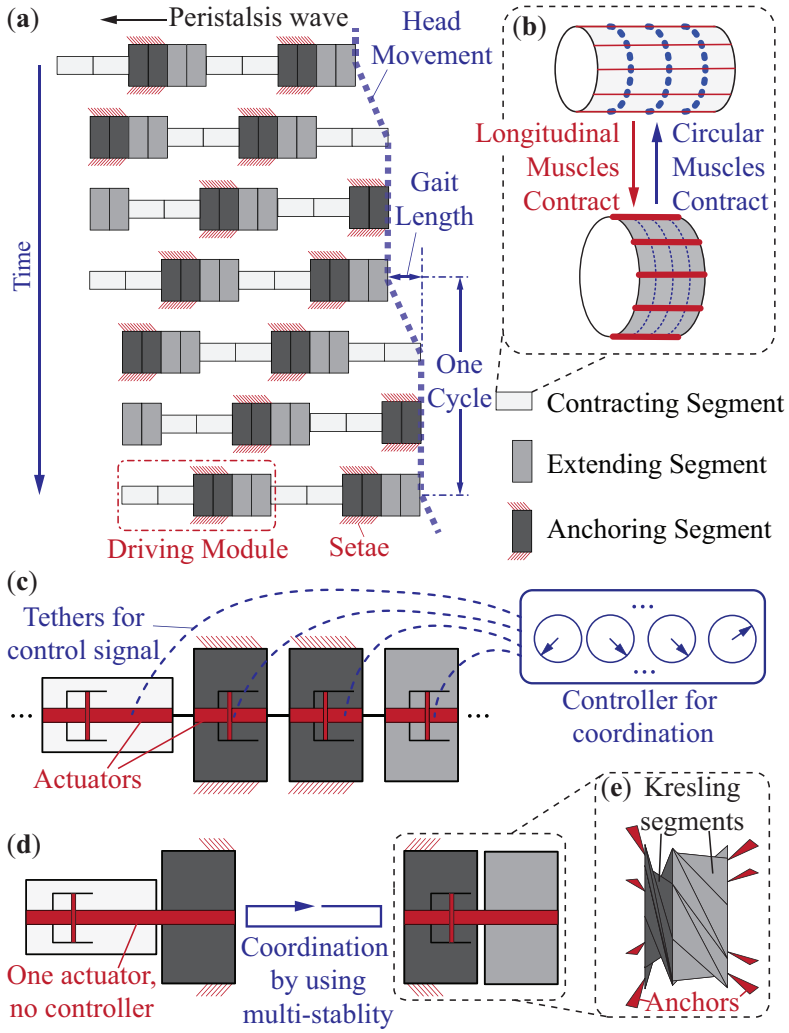

Figure 3.1: The vision of using multi-stability to drastically simplify the mechatronic setup for generating peristaltic locomotion. (a) Peristaltic locomotion cycle in an earthworm. The earthworm body moves forward while the peristaltic wave propagates backwards. For clarity, the earthworm body consists of six identical segments and two driving modules. (b) The muscular actuation scheme of an earthworm segment. The alternate contraction of longitudinal and circular muscles drives the segment deformation and anchoring actions. (c) The mechatronic setup of a traditional earthworm-inspired robot that requires many actuators and a complicated controller. (d) The proposed peristalsis locomotion mechanism that uses multi-stability to eliminate the need of multiple actuators and controllers. (e) A to-scale schematic diagram of the dual-Kresling driving module and its foldable anchors.

displacement relationship in *multi-stable* origami to generate peristaltic-like locomotion without relying on multiple actuators or digital controllers. A material or structure is multi-stable when it possesses more than one stable equilibria (or states). It can remain at one of its stable states without any external aid, and switch between these states by external or internal actuation. The potential energy landscape of a multi-stable system has multiple peaks and valleys by definition, which creates non-monotonic force-displacement relationships. Under certain loading conditions, this non-monotonic behavior can induce large deformation through a rapid release of elastic energy (also referred to as "snap-through" in some scenarios). This rapid deformation in multi-stable system is the driving mechanism underpinning many nastic plant movements [37], and it has found various engineering applications like energy harvesting [48, 16, 171], vibration isolation [61, 62, 53], as well as actuation and morphing [119, 146, 4].

Regarding the applications in robotics, multi-stability also shows promise in amplifying the authority and speed of robotic actuation [26, 69], actuating an untethered soft swimming robot [12] or increasing the precision and repeatability of a micro-robotic end effector [11]. More importantly, recent studies reveal that multi-stability can be harnessed to drastically reduce or even eliminate the need for using digital controllers to generate soft robot locomotion [128], mechanical logic gates [152], non-peristaltic crawling [147], and coordinated oscillation [125]. Logical programming for robotic gripping is also proven feasible by using soft bistable valves [126]. It is worth emphasizing that in some of these studies, the necessary condition to achieve robotic functions is the non-monotonic energy landscape or force-displacement relationship, while multi-stability serves as a mechanism to achieve the desired non-monotonic behavior.

In this study, we show that by exploiting the non-monotonic energy landscape in the multi-stable Kresling origami, we can create peristaltic-like crawling locomotion with only one actuator and without any digital controllers (Figure 3.1(d)). Origami is an ancient art of paper folding wherein folding a 2D sheet along prescribed crease lines results in the creation of complex 3D shapes. Over the past few decades, it has become a framework for constructing deployable structures [174], mechanical metamaterials [84], and reconfigurable robots [133]. Origami mechanisms are inherently lightweight, compact, and compliant. More importantly, they can exhibit unique mechanical properties—such as auxetics, programmable nonlinear stiffness, and multi-stability [138, 169, 144, 145, 160, 33, 81, 23, 66]—due to the nonlinear kinematics of folding.

The crawling robot, in this study, consists of a driving module consisting of two serially

connected *Kresling segments* and foldable anchors (Figure 3.1(e)). We designed the Kresling pattern according to the desired kinematics and bistability so that these segments can exhibit both longitudinal and radial deformation via folding. When its total length is increased and decreased by a linear actuator, the dual-Kresling driving module can display a deterministic deformation sequence (or "actuation cycle") that includes two rapid "jumps". We then designate different parts of this actuation cycle as the phases of a peristaltic-like locomotion gait. By doing so, we can eliminate the need for using individual actuators for each segment or using digital controllers to coordinate these actuators. That is, the peristaltic locomotion is essentially "coordinated" by the nonlinear mechanics of Kresling origami. Therefore, this study will contribute to a paradigm shift in how we can use multi-stability for robotic actuation and control.

We first proposed this concept of peristaltic locomotion using Kresling origami mechanics in a single case study without any experimental validation or in-depth investigation [6]. Therefore, the propose of this letter is to examine the correlations between origami design, folding mechanics, and locomotion performance comprehensively through both analytical and experimental efforts. The following sections of this letter will (2) detail the design, analysis, and characterization of the elementary Kresling origami segments; (3) elucidate the creation of a deformation sequence (or "actuation cycle") using the rapid deformations caused by multi-stability; (4) discuss the experimental validation of the peristaltic-like locomotion using this actuation cycle and a comprehensive parametric study of gait length; and (5) conclude this study with summary and discussion.

3.2 Generalized Kresling Origami Segment

The centerpiece of the peristaltic crawling robot in this study is a driving module consisting of two serially connected Kresling origami segments. The Kresling pattern consists of a linear array of mountain and valley folds defined by triangular facets (Figure 3.2(a)). By attaching the two ends of this array (marked by *), we obtain a twisted polygonal prism with a regular polygon at its top and bottom. These two end polygons remain rigid throughout the folding motion. Kresling origami was initially studied as a buckling mode in thin cylindrical shells subjected to torsion [55, 72]. Since then, it has been used extensively as a template for deployable structures or robotic skeletons [60, 110, 118]. Kresling origami suits this study well because it has the desired tubular cross-section, and more importantly, it is inherently bistable, thus exhibits the desired non-monotonic

energy landscape and force-displacement curve. A Kresling segment can settle in a fully-extended or a fully-contracted stable state, and it shows a large deformation between these two states. This bistability originates from its non-rigid-foldable nature. The triangular facets remain undeformed at the two stable states, but must deform while folding between these two states. Indeed, if these triangular facets were strictly rigid, the Kresling segment would not fold. For clarity, we refer the fully-contracted stable state as the state (0) and the fully-extended stable state as the state (1) hereafter.

3.2.1 Design of the generalized Kresling origami

The design of a *traditional* Kresling segment is fully defined by three independent parameters: the number of sides of the base and top polygon N, the side length of the polygon P, and an angle ratio λ, which is related to the angle between polygon side and valley crease in the triangular facets (Figure 3.2(a)). The length of the valley and mountain creases are:

$$D_i = 2R\cos(\gamma - \lambda\gamma), \tag{3.1}$$

$$B_i = \sqrt{P^2 + D_i^2 - 2PD_i\cos(\lambda\gamma)}, \tag{3.2}$$

where $\gamma\,(=\pi/2-\phi)$ is the angle between the diagonal and side of the end polygon, $R\,(=0.5P/\sin\phi)$ is its circumscribed radius, and $\phi = \pi/N$. The traditional Kresling design, however, has a shortcoming: Its length at the fully-contracted stable state (0) is always zero. This is impossible in practice due to the finite material thickness, more importantly, it significantly constrains the design space available for tailoring the kinematics of peristalsis crawling. To address this issue, we created a *generalized* Kresling pattern by adding the fourth independent design variable: a non-zero segment length at stable state (0) (aka. $L_{(0)}$ in Figure 3.2(a)) [6]. The triangular facets are "stretched" as a result and their geometry is adjusted accordingly:

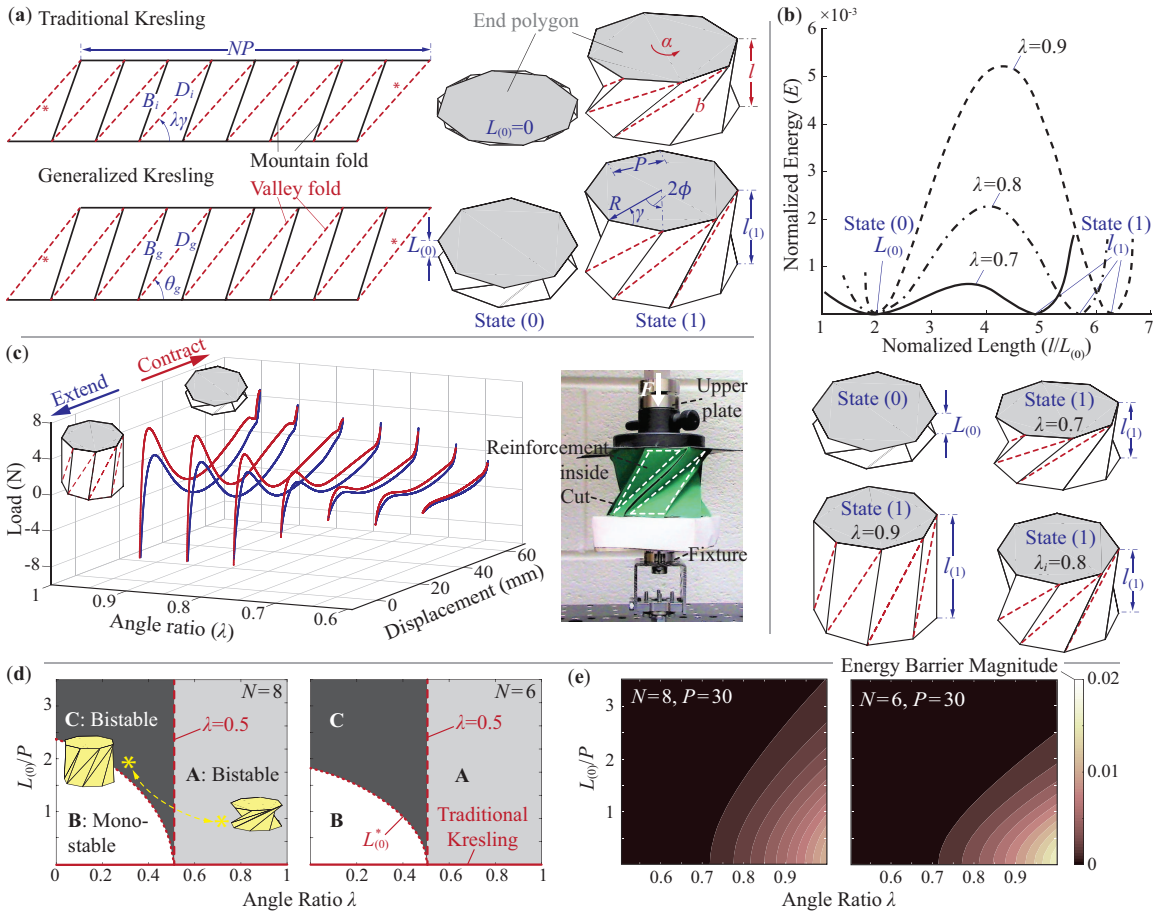

Figure 3.2: Design, analysis, and experimental characterization of the generalized Kresling origami. a) Crease pattern and the folded segment of both traditional and generalized Kresling origami showing the important design parameters and variables related to folding. The traditional Kresling always has a zero-length at the fully-contracted state (0), while the generalized Kresling has a "user-defined" $L_{(0)}$. b) The normalized strain energy versus length of three Kresling segment designs of different angle ratios but the same $L_{(0)}(= 20\text{mm})$, $N(= 8)$, and $P(= 30\text{mm})$. Increasing the angle ratio can increase the bistability strength and length of the segment at the fully-extended state (1). c) Experimentally measured force-displacement curves of Kresling segments with different angle ratios but the same $L_{(0)}(= 10\text{mm})$, $N(= 8)$, and $P(= 30\text{mm})$. One can clearly see the correlation between angle ratio and bistability strength in terms of the maximum reaction force between stable states. The inserted picture on the right shows the experimental setup. d) Results of parametric study depicting influence of the normalized $L_{(0)}$ versus λ landscape showing bistability range for different N. The designs in Region **A** are always bistable. The Kresling segments in inset are identical with configuration in Region **C** as state (1) and the one in Region **A** as state (0). e) The normalized $L_{(0)}$ versus λ (for $\lambda \geq 0.5$) landscape showing bistability strength for different N. In these two plots, the color map (labeled on the right) represents the height of the normalized energy barrier between two stable states. Therefore, a darker color means weaker bistability and vice versa.

$$D_g = \sqrt{D_i^2 + L_{(0)}^2}, \tag{3.3}$$

$$B_g = \sqrt{B_i^2 + L_{(0)}^2}. \tag{3.4}$$

$$\theta_g = \cos^{-1}\left(\frac{P^2 + D_g^2 - B_g^2}{2PD_g}\right) \tag{3.5}$$

Here, θ_g is the angle between polygon side and valley crease, the subscript "i" denotes the traditional Kresling and "g" denotes the generalized Kresling. By using this generalized design, we can freely assign *non-zero* lengths to the Kresling segment at both fully-contracted state (0) and fully-extended stable state (1).

To characterize the bistability of generalized Kresling segments, we adopt the equivalent truss frame approach [60]. This approach uses pin-jointed truss elements to represent the mountain and valley creases and assumes that the valley creases do not change their length during folding [110, 118]. In this way, the triangular facet deformations induced by folding between the two stable states can be approximated as the stretching and compression of the truss elements along mountain creases. More specifically, the mountain crease trusses are un-deformed at the two stable states, but they are compressed as we fold the Kresling segment between its two states. To describe the Kresling folding deformation, we use three variables: the relative rotation angle between the top and bottom end polygon during folding (α), the overall length of the Kresling segment (l), and the length of the truss element along mountain creases (b). These three variables apply to both traditional and generalized Kresling, and they are inter-dependent. Notably, the values of α are the same between the traditional and generalized Kresling, so we can use it as the independent variable and obtain a closed-form solution describing the folding kinematics:

$$l(\alpha) = \sqrt{L_{(0)}^2 + 2R^2\left[\cos(\alpha + 2\phi) - \cos(\alpha_{(0)} + 2\phi)\right]}, \tag{3.6}$$

$$b(\alpha) = \sqrt{2R^2(1 - \cos(\alpha)) + l^2}. \tag{3.7}$$

Here, $\alpha_{(0)}(= 2\lambda\gamma)$ is the angle between top and bottom polygon at the fully-contracted stable state (0). Angle $\alpha_{(1)}$ corresponding to the fully-extended stable state (1) can be found by setting the

mountain crease length b equal to its undeformed length B_g:

$$\alpha_{(1)} = \{\min(\alpha)|b(\alpha) = B_g\}. \tag{3.8}$$

Alternatively, $\alpha_{(1)}$ can be computed as:

$$\alpha_{(1)} = 2(1-\lambda)\gamma \ \ \forall \ \lambda > 0.5 \tag{3.9}$$

The equivalent strain (ϵ) and strain energy (U) due to folding are

$$\epsilon = \frac{b}{B_g} - 1 \text{ and } U = \frac{1}{2}K\epsilon^2, \tag{3.10}$$

where K represents the constituent sheet material stiffness. For the purpose of this analysis, we normalize the strain energy U by K, and define the non-dimensional strain energy as

$$E = \frac{1}{2}\epsilon^2. \tag{3.11}$$

Figure 3.2(b) illustrates the normalized strain energy of three Kresling designs with the same $L_{(0)}$ but different angle ratios λ. The two potential energy wells are evident in these analytical results. Moreover, as the angle ratio increases, the effective strain ϵ increases, consequently increasing the bistability strength in terms of the height of energy barrier between stable states. For a given $L_{(0)}$, the transition from mono-stability to bistability takes place at $\lambda = 0.5$ and bistability is strongest when $\lambda = 1$.

3.2.2 Experimental characterization

To confirm the correlation mentioned above between bi-stability strength and angle ratio; we fabricated and tested prototypes of the generalized Kresling segments using paper (Daler - Rowney Canford 150 gsm). We first prepared the 2D drawing of Kresling pattern in SOLID-WORKS™and cut them out of paper with perforated creases on a plotter cutter (Cricut Maker®). We then manually folded the cut pattern into the Kresling segment and attached its top and bottom polygons to the universal testing machine (ADMET eXpert 5601). To accommodate the relative rotation of these end polygons, we designed a custom rotation fixture consisting of a dual ball-bearing hub (Figure

3.2(c)). Certain adjustments to the Kresling segment fabrication were necessary to facilitate smooth folding and obtain consistent results for higher angle ratio designs. First, we cut the mountain creases to alleviate any excessive stresses that can lead to tearing after a few loading cycles. A similar approach is used in the "Flexigami" [110]. Secondly, we added triangular reinforcements to the facets to increase their stiffness relative to the creases, strengthening overall bistability (Figure 3.2(c)).

Figure 3.2(c) also illustrates the measured force-displacement curves of several Kresling segment prototypes. The correlation between angle ratio and bistability strength is evident in that a segment with a higher angle ratio demands a more significant actuation force to be switched between stable states. Moreover, we observe a hysteresis loop between the extension and contraction cycles. The force-displacement curves remain almost identical under repeated measurement cycles; unless we excessively extend the segment beyond its stable state causing tears in the creases. Thus we think this hysteresis behavior is intrinsic to the system, and it probably originates due to the contact between triangular facets and the plasticity of the paper. Nonetheless, we can minimize this hysteresis by the cutting and reinforcement techniques so that it will not significantly affect the generation of the actuation cycle.

3.2.3 Parametric design study on the Kresling bi-stability

We performed further parametric analyses to fully understand the correlation between design parameters and stability properties of the generalized Kresling segment. In this study, we varied the number of polygon sides N, the polygon side length P, the fully-contracted segment length ($L_{(0)}$), and the angle ratio (λ). To ensure consistency, we normalized the fully-contracted segment length ($L_{(0)}$) based on the base polygon side length (P). Results of the parametric study show that changes in P or N do not fundamentally alter the segment stability. The generalized Kresling segments are always bistable regardless of the $L_{(0)}$ value if $\lambda > 0.5$ (Region A in Figure 3.2(d)). The segments are always mono-stable if λ is precisely 0.5. If $\lambda < 0.5$, the generalized Kresling segments can transit from being mono-stable to bi-stable as $L_{(0)}$ increases (Region B to C in Figure 3.2(d), respectively). Decreasing N lowers the transition curve between Region B and C; thus decreasing the design space available in mono-stable region. The magnitude of the critical $L^*_{(0)}$ at the boundary between Region B and C is

$$L^*_{(0)} = \sqrt{2R^2\left(\cos(2\lambda\gamma) + \cos(2\lambda\gamma + 2\phi)\right)} \tag{3.12}$$

However, upon closer inspection, we find that the bistable segment designs in Region C are redundant. That is, for any bistable Kresling design in Region C with an angle ratio of $\lambda < 0.5$, we can find an identical one in Region A with an angle ratio of $1 - \lambda$. Moreover, the "$L_{(0)}$" in Region C indeed represents the segment length at its fully-extended stable state. Therefore, we neglect the bistable Kresling designs with $\lambda < 0.5$ hereafter.

Figure 3.2(e) shows the effects of adjusting segment N and $L_{(0)}$ on its bistability strength, which is characterized by the normalized strain energy barrier between the two stable states. Higher strain energy barrier corresponds to stronger bistability strength and vice-versa. Generally speaking, increasing the $L_{(0)}$ while keeping other design parameters unchanged would decrease the bistability strength. Therefore, Kresling segments with a smaller $L_{(0)}$ and a larger angle ratio λ exhibit stronger bistability. Moreover, the polygon side length P is unrelated to bistability, while a reduction in N can increase the bistability strength. These parametric design studies can help us tailor the crawling locomotion gait performance in the following sections.

3.3 Actuation Cycle from the Multi-stable Driving Module

In this section, we use a case study to illustrate how to harness the multi-stability in the Kresling origami to generate a deterministic deformation sequence (or "actuation cycle") with only one actuator. In this case study, the driving module consists of two generalized Kresling segments of different angle ratios and bistability strengths (Figure 3.3(a)). Without any loss of generality, we assume $\lambda_{\mathrm{I}} \geqslant \lambda_{\mathrm{II}}$, where the subscript "I" and "II" represents the two constituent Kresling segments, respectively. The Kresling design parameters used for this dual-segment driving module are listed in Table 3.1. To generate the actuation cycle, we stretch and compress this driving module at its two ends without manipulating its two segments individually. That is, we only increase and decrease the *total length* (l_t) of the driving module without directly controlling the individual segment lengths.

Table 3.1: Design parameters of the two Kresling segments in the driving module.

Parameter	**Segment I**	**Segment II**
N	8	8
P (mm)	30	30
λ	0.8	0.6
$L_{(0)}$ (mm)	15	5

To identify the actuation cycle, we first need to find how the driving module strain energy

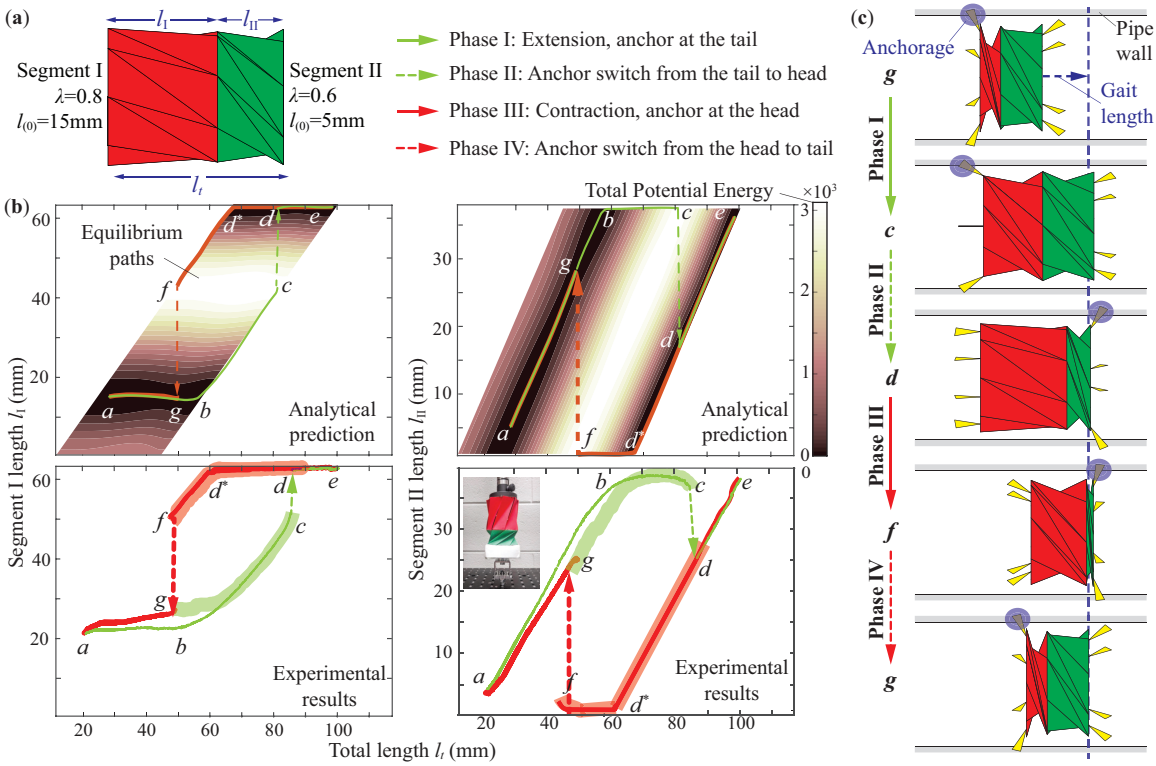

Figure 3.3: Formation of the actuation cycle in the multi-stable Kresling driving module and the corresponding peristaltic-like locomotion gait. (a) The design of the driving module and the nomenclature denoting the different phases in the actuation cycle. (b) Analytical prediction (up) and experimental results (below) of the Segment I and II deformations versus the prescribed change in the total length of the driving module. In the two plots of analytical prediction, the color map (labeled on the right) represents the total potential energy landscape. Darker color represents lower energy and vice versa. The transparent thick curve superimposed on the full equilibrium path (below) depicts the experimental measurements of the actuation cycle only. (c) To-scale schematic diagram of the peristaltic-like crawling gait that is generated using the four phases in the actuation cycle and foldable anchors. Design parameters of the driving module are listed in Table 3.1.

changes when the total length (l_t) of the driving module is changed from its minimum to maximum and vice versa. This will enable us to get the individual segment deformations and identify the path the system follows as total length (l_t) is changed. We applied a customized optimization algorithm to the landscape of total strain energy (Figure 3.3(b)) [113]. In this optimization, the objective function is the total strain energy $E_t = E_{\mathrm{I}} + E_{\mathrm{II}}$ according to Equation (3.11). The independent variable is the segment length l_{I} or l_{II}, and they must satisfy the equality constraint $l_{\mathrm{I}} + l_{\mathrm{II}} = l_t$, and be within the bounds $l_{\mathrm{I\ min}} \leq l_{\mathrm{I}} \leq l_{\mathrm{I\ max}}$, and $l_{\mathrm{II\ min}} \leq l_{\mathrm{II}} \leq l_{\mathrm{II\ max}}$. In this way, the optimization problem becomes: Find the value for l_{I} (or l_{II}) which locally minimizes the scalar objective function E_t for a given a prescribed total length l_t, and satisfies the given equality constraint. Results of this optimization are shown as the "equilibrium paths" in Figure 3.3(b), and B details a more comprehensive optimization procedure involving multiple Kresling segments.

We start by stretching the driving module when its two segments are both at its fully-contracted stable state (0) (point a in Figure 3.3(b)). During the stretching, the Kresling segments deform by following the equilibrium path $a \to g \to b \to c \to d \to e$ until both of them reach the fully-extended stable state (1) (point e in Figure 3.3(b)). Then, we compress the driving module and observe that the segments follow a different equilibrium path $e \to d \to d^* \to f \to g \to a$ until they come back to the state (0) (Supplemental Video A).

In these equilibrium paths, We observe two distinct "jumps" caused by the non-monotonic energy landscape of the multi-stable origami. One occurs during the stretching from $c \to d$, and the other during the compression from $f \to g$ (Figure 3.3(b)). When these jumps occur, a branch of local energy minima reaches its end so that the driving module is forced to deform to a distant branch of energy minima quickly. During these jumps, the two Kresling segments change their length significantly, while their total length (l_t) remains almost the same. By combining parts of these equilibrium paths and the two jumps, we can construct an "actuation cycle": $g \to b \to c \to d \to d^* \to f \to g$. This actuation cycle consists of four consecutive "phases": In Phase I ($g \to b \to c$), Segment I increases in length significantly while Segment II remains almost fully extended. Phase II ($c \to d$) is the first jump, by which Segment I quickly reaches the fully-extended state, but Segment II contracts significantly in length. In Phase III ($d \to d^* \to f$), Segment II continues to contract in length until reaching its fully-contracted state, Segment I also contracts but to a lesser degree. The final Phase IV ($f \to g$) is the second jump, by which Segment I quickly deforms to its fully-contracted state, but Segment II extends in length significantly.

Here, it is worth emphasizing that in this driving module, multi-stability of the two segments is a sufficient but not necessary condition to cause the deterministic deformation sequence in the actuation cycle. Indeed, similar deformation sequence could be achieved if the force-displacement curve of at least one segment has a non-monotonic force-displacement relationship [117, 97].

We experimentally verified the formation of this actuation cycle in a paper-based prototype of the driving module (Figure 3.3(b)) (Supplemental Video A). The fabrication procedure and experimental set up are the same as the single Kresling segment tests. The universal testing machine was used to prescribe the change in the total length of the driving module (l_t). To accurately measure the segment deformation, we obtained high-resolution video footage of the driving module and used the MATLAB® Image Processing Toolbox™ to measure the length of Kresling segments (l_I and l_{II}).

The measured actuation cycle, including the two jumps, agrees well with the analytical predictions. The experiment is repeatable over multiple extension and contraction cycles. However, there are slight discrepancies between the analytical prediction and experiment measurements. More specifically; the measured total lengths at which the jumps occur are slightly different from the predictions and the jump magnitudes are lower. The deviation from the ideal actuation cycle is mainly attributed to the hysteresis observed in individual segment testing. Errors are also introduced during the fabrication and measurement stages. The experimental results show that the equilibrium path from g to b in Phase I is not fully closed as depicted in the analytical prediction. A further experiment with the Phases of actuation cycle shows that the "real" Phase I slightly differs from the observed equilibrium path during extension, but it doesn't change the location of jumps in Phase II and Phase IV (Figure 3.3(b)). Nonetheless, these discrepancies do not hinder the creation of peristaltic-like locomotion as we will discuss in the following section.

3.4 Locomotion Gait Generation

In this section, we show how the actuation cycle, combined with foldable anchors, can create peristaltic-like crawling locomotion. Segments in the earthworm body increase in diameter while contracting in length and vice versa (Figure 3.1(b)). This is an important component for achieving peristaltic locomotion because it provides a mechanism to anchor the fully-contracted segment to its surroundings by the setae. The diameter of Kresling segment, on the other hand, does not change

Table 3.2: Anchor design parameters, units in mm

	Parameter	Seg. I	Seg. II
Foam cube	Length	18	15
	Width	15	15
	Thickness	10	10
Connector sheet	Length	15	15
	Width	15	15
	Thickness	0.5	0.5

when its length changes. This necessitates the design of anchors which mimic the radial deformation of earthworm segments.

3.4.1 Anchor Design

We designed the anchors by taking advantage of the folding kinematics of Kresling segments. These anchors are attached to the triangular facets, so they can deploy and increase the effective diameter when the segments are contracting (Figure 3.4(a,b)). They have plastic foam cubes at their tips to create sufficient friction and thus a strong anchorage to their surroundings (a pipe of 47.5mm radius in this case). Moreover, we define a "cut-off" length for each segment to ensure proper anchor deployment. When the Kresling segment contracts longitudinally below its cut-off length, its anchors should be deployed far enough to create an anchorage with its surroundings. For Segment I, its cut-off length is the length at the point c on its equilibrium path as shown in Figure 3.3; for Segment II, its cut-off length corresponds to the point d. We then determined the dimensions of these anchors according to these cut-off lengths, folding kinematics of the Kresling, and the pipe inner diameter (Table 3.2). The anchors are designated as tail anchor and head anchor according to their position on the robot. The tail anchor is attached to Segment I and the head anchor is attached to the Segment II. In this way, the effective diameter of the Segment I is larger than the pipe diameter during Phase I of the actuation cycle, while the diameter of Segment II is larger than the pipe during Phase III (Figure 3.4(c)). Moreover, the anchoring location switches from Segment I to II in the Phase II jump, and switches back from Segment II to I in the Phase IV jump.

3.4.2 Peristaltic-like Locomotion Gait

By combining the dual-segment multi-stable driving module and the properly designed anchors, we now complete the design of crawling robot and harness the actuation cycle to generate a

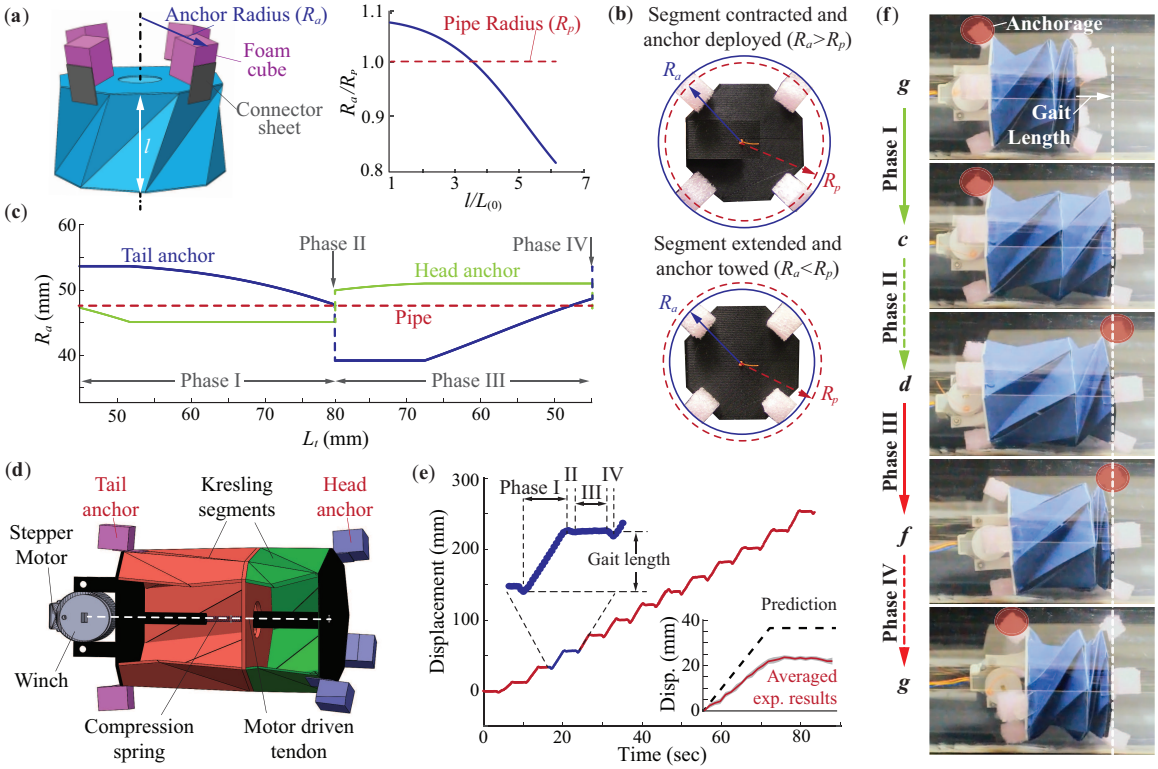

Figure 3.4: Fabrication and testing of the multi-stable Origami crawler prototype. (a) Left: a 3D CAD rendering of the Kresling segment with the anchors attached. Right: The relationship between the effective anchor radius (R_a), pipe radius (R_p), and segment length (l). (b) The fabricated segment showing maximum and minimum anchor radius with respect to the pipe radius. (c) The change in anchor radius during one actuation cycle. The anchor design parameters are in Table 3.2. (d) 3D CAD rendering of the crawler with a cutaway view showing the motor-winch actuation mechanism. (e) The measured movement of the robotic head over many actuation cycles. The insert figure at the upper left corner illustrates the four different phases in one actuation cycle. Insert at the lower right compares the averaged gait in the experiment and the analytical prediction. The shaded band is the standard deviation. (f) The observed four phases of the actuation cycle in the origami crawling robot. The anchor switch locations and gait length are highlighted.

peristaltic-like locomotion gait. More specifically, the four consecutive phases in the actuation cycle can be used to alternate the anchoring locations between the head and tail of the driving module, resulting in a net forward displacement as detailed below (Figure 3.3(c)):

In Phase I ($g \rightarrow b \rightarrow c$), the crawling robot is anchored at its tail because its Segment I is below its cut-off length. Meanwhile, the robot body is increasing in its total length by the actuator input, giving a net forward displacement.

In Phase II ($c \rightarrow d$), the jump between the equilibrium paths switches the anchor location from the tail to the head. No head or tail displacement occurs during this jump.

In Phase III ($d \rightarrow d^* \rightarrow f$), the crawling robot is anchored at its head because its Segment II is now below its cut-off length. Meanwhile, the robot body is contracting in its total length, moving the tail forward.

In the final Phase IV ($f \rightarrow g$), the second jump occurs and the anchor location switches back from head to the tail. At the end of this phase, the crawling robot returns to its original configuration of the actuation cycle, i.e. at the start of Phase I. The "gait length" is the total forward movement of the crawling robot head after one actuation cycle. It is equal to the change in driving module length (l_t) between two jumps; i.e. Gait Length = $l_t(c \rightarrow d) - l_t(g \rightarrow f)$. The actuation cycle from Phase I to Phase IV can be repeated to drive the robot forward continuously.

To experimentally validate the peristaltic-like locomotion induced by multi-stability, we fabricated and tested a proof-of-concept prototype of the crawling robot. This prototype features the same Kresling origami and anchor designs as in the analytical case study (Table 3.1 and 3.2). A compression spring-winch mechanism attached to the two end plates of this robot is used to control its total length (Figure 3.4(d)). A 5V stepper motor drives this spring-winch mechanism, and the motor rotation is pre-programmed using Ardruino METRO 328 and motor-shield v2.3. To decrease driving module length, the robot's stepper motor turns the winch, pulling in the attached tendon. To increase the total length, the motor turns the winch in the opposite direction to release the tendon. The compression spring provides the internal force to keep the tendon taut. To measure the locomotion performance, we took high-quality video footage of the crawling robot in action and used the Computer Vision Toolbox in MATLAB® (Supplemental Video B). We developed a computer program using the Kalman filter based motion tracking algorithm to track the movement of the head of the robot.

The experimental results summarized in Figure 3.4(e,f) agree quite well with the analyt-

Table 3.3: Features and performance of the final origami crawler prototype

Feature	**Value**
Mass	70 g
Maximum length	90 mm
Minimum length	55 mm
Average speed	3.3 mm/sec
Average gait length	22 mm
Average cycle duration	6.7 sec

ical predictions in Figure 3.3(c) regarding the segment deformation sequence and anchor location switches. Moreover, the robot locomotion cycle is uniform and repetitive (Figure 3.4(e)). There is a discrepancy regarding the magnitude of gait length between the experiment and analysis, and two factors can contribute to this. One is that the analytical prediction uses an idealistic model to characterize the Kresling bi-stability so it does not fully capture the behaviors of the physical prototypes with anchors and actuators integrated (also evident in Figure 3.3(b)). The other factor is the slippage between the pipe and robot anchors, which results from the temporary loss of contact during the anchor switching in Phase II and IV. Regardless, this experiment firmly validates the feasibility of using multi-stability in the Kresling origami to create the peristaltic-like locomotion with only one actuator and without a complex control architecture to coordinate the segments. That is, the deformation of the segments and anchorage locations are "coordinated" directly by the mechanics of elastic multi-stability.

Table 3.3 summarizes the features and locomotion performances of the dual-segment multi-stable origami crawler. It is important to highlight that the actuation cycle induced by multi-stability is independent of the *rate* of stretching/compression in total length. Therefore, by changing the rotational speed of the motor one can adjust the *frequency* of the locomotion cycle and thus the averaged crawling speed, however, the motor speed does not affect the gait length in one locomotion cycle. The gait length is only related to the Kresling origami design and the corresponding multi-stability. We detail this further in the following parametric study.

3.4.3 Parametric Study: Gait Length

It is clear from the actuation cycle study that the locomotion gait length depends on the driving module deformation between the two jumps and the magnitude of these jumps, and the underlying Kresling design also plays an important role. To uncover the correlations between peri-

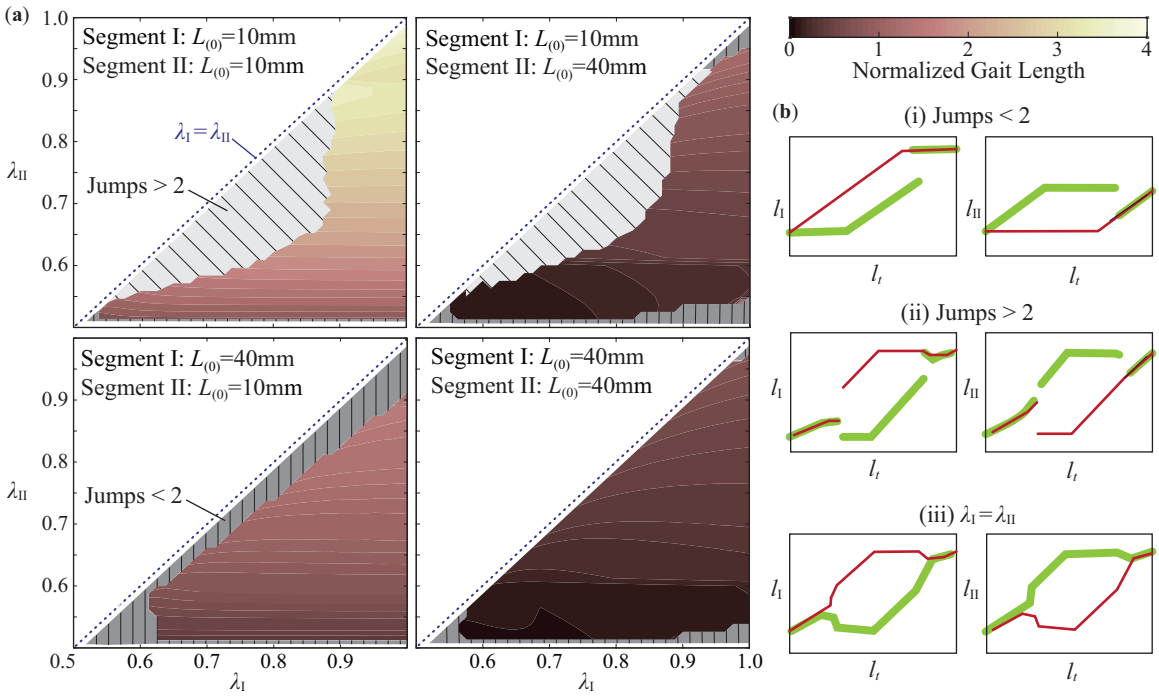

Figure 3.5: Parametric Study depicting the influence of segment angle ratio (λ) and fully-contracted length ($L_{(0)}$) on the gait length of crawling robot. (a) Results of the parametric study depicting the influence of segment bistability strengths on locomotion gait length for different fully-contracted lengths ($L_{(0)}$). Here, the color map represents the normalized locomotion gait length, and the color bar on the right applies to all four plots. (b) Examples of equilibrium paths that do not exhibit any properly defined, four-phased actuation cycles with certain combinations of segment angle ratios. $N = 8$, $P = 30$mm.

stalsis gait length and Kresling origami design, we performed a parametric study by combining two segments of different bistability strength (aka. different angle ratios and $\lambda_{\mathrm{I}} \geqslant \lambda_{\mathrm{II}}$). To ensure consistency, we normalize the gait length based on the fully-contracted length of the driving module $l_{t\ min}$. Results of the parametric study show that, for a given λ_{I}, normalized gait length increases as λ_{II} increases. On the other hand, for a given λ_{II}, normalized gait length is insensitive to changes in λ_{I} (Figure 3.5(a)). Moreover, the fully-contracted lengths of the segments ($L_{(0)}$) have a significant influence on the normalized gait length and permissible combinations of segment angle ratios. Generally speaking, the normalized gait length decreases as $L_{(0)}$ increases. However, smaller $L_{(0)}$ can make more combinations of angle ratios unfeasible for peristalsis locomotion as we detail below.

There are three possible scenarios by which peristaltic-like locomotion is unachievable. In the first scenario, there are less than two jumps in the actuation cycle. If the jump during the contraction phase does not occur (case (i) in Figure 3.5(b)), both segments will contract monotonically and anchor to the pipe, thus preventing any further locomotion. If the jump during the extension phase does not occur, both segments will elongate monotonically, thus losing proper anchorage at both ends of the robot. In the second scenario, there are more than two jumps in the actuation cycle (case (ii) in Figure 3.5(b)). It is difficult to generate anchorage switches from these jumps consistently, and the resulting actuation cycle becomes unnecessarily complicated. Moreover, the presence of multiple jumps during the extension or contraction phase may reduce the jump magnitude, making peristaltic motion unachievable. Therefore, we choose not to perform any detailed study of this multiple-jump scenario. The third scenario occurs when $\lambda_{\mathrm{I}} = \lambda_{\mathrm{II}}$ (case (iii) in Figure 3.5(b)). In this case, there are no discernible jumps that can create any actuation cycles.

3.5 Summary and Conclusion

In this study, we demonstrated the use of multi-stability embedded in a robotic origami skeleton to create peristaltic-like locomotion without the need for multiple actuators or complicated controllers. By combining two bistable Kresling origami segments into a driving module and increasing/decreasing its total length, one can generate a deterministic deformation sequence (or actuation cycle). This actuation cycle has two discrete "jumps" that can significantly change the length of two constituent Kresling segments without affecting their total length. These jumps are the result of the complicated non-monotonic energy landscape caused by the nonlinear mechanics of

folding, and they naturally divide the actuation cycle into four distinct phases. We then designed and experimentally validated a peristaltic-like robotic crawling by using two phases for moving the robot forward and the other two for switching the anchoring locations. To ensure proper anchorage to the surroundings, we designed and implemented foldable anchors according to the kinematics of Kresling folding. The results of this work show that the nonlinear mechanics of multi-stability can be used to directly coordinate the robotic motion and drastically simplify the mechatronic setup and control of compliant robots.

While we have used a compression spring-winch based linear actuator to control the length of the driving module, any other mechanism that can work in the required deformation range may be used to actuate the robot. The scale independence of the origami mechanism ensures that the same robot design principles can be used to create nano/micro-scale as well as large-scale robots. Moreover, it is worth highlighting that although Kresling origami is used in this study for its simplicity and versatility, the principle of using elastic multi-stability to generate peristaltic-like locomotion is applicable to any other segmented robot systems, as long as the segment can (i) exhibit a coupled longitudinal and radial deformations (aka. expanding radially while shrinking longitudinally, and vice versa) and (ii) exhibit a strong non-monotonic force-displacement relationship.

In future work, the fabrication and modeling precision of the generalized Kresling origami will be improved to provide more accurate analysis of the locomotion performance. Origami possesses many other unique properties, such as programmable stiffness and auxetics, which could also be exploited for soft robotic applications. Finally, the results of this study can be used to create an efficient and hybrid approach for soft robot control. In this approach, the lower-level control tasks (such as locomotion gait generation) are taken up by the embedded mechanics of multi-stability in the mechanical domain, while the high-level control tasks (such as adapting locomotion direction and speed according to the working environment) are achieved by sensors and controllers in the digital domain. In essence, folding induced multi-stability can impart a "mechanical intelligence" to the robotic body as a foundation for this vision of hybrid soft robot control.

Chapter 4

Bi-directional locomotion using multi-stable compliant mechanisms

4.1 Introduction

Previously, we have shown that we can harness the multi-stability in compliant robotic skeleton to generate peristaltic-like locomotion gait and simplify the control architecture significantly. Specifically, we designed a compliant robot by serially connecting two bistable generalized Kresling origami segments. We require only a single linear actuator to control the length of robot and generate a coordinated actuation cycle for uni-directional peristaltic-like crawling gait. In this study, we further explore the design space of multi-stable mechanisms to investigate whether bi-directional locomotion is possible. We perform multi-stability analysis on the compliant robotic skeleton composed of two serially connected 'generic' tri-stable and bi-stable mechanisms to study the viability of bi-directional crawling locomotion gait. After multiple numerical studies we see that such bidirectional locomotion is indeed possible, but the design space of useful mechanisms is very small, making viable mechanisms very rare indeed. This means it becomes important to accurately fabricate such mechanisms with appropriate hinge stiffness.

4.2 Synthesis of Compliant Mechanisms

We used Pseudo Rigid Body Modeling (PRBM) to synthesize the compliant multi-stable mechanisms. Ideally, a segment of peristaltic crawling robot should satisfy two criteria: (a) exhibit a large deformation range between the fully-extended and fully-contracted states; (b) contract radially when extended longitudinally and vice-a-versa. To satisfy the first requirement we chose a 'generic' compliant slider-crank bi-stable mechanism with a large deformation range between two stable states as basis for synthesizing multi-stable mechanisms. We can attach foldable anchors to the rigid links of the bi-stable mechanism to fulfill the second requirement. The planar bi-stable mechanism is composed of two compliant slider-crank mechanisms that are mirror images of each other (Figure 4.1(a)). The resulting planar mechanism still has a rotational degree of freedom about axis perpendicular to the plane of the motion. Thus, one way to design a spatial stable mechanism is to combine two such planar mechanisms orthogonally to create 3-D bistable compliant mechanism that has a single translation DoF along the longitudinal axis (Figure 4.1(b)). We study one tri-stable mechanism that is created by nesting two serially connected bi-stable mechanisms inside a larger bi-stable mechanism (Figure 4.2(a)). We can then proceed to tailor the geometry of these mechanisms to form appropriate multi-stable mechanism that can potentially generate bi-directional crawling locomotion gait.

The bistable compliant mechanism design is parameterized using its length in fully-extended state (L_1), length in fully-contracted state (L_0), link length l_1 and link length l_4. These lengths also correspond to the two bistable states of the compliant mechanism. The link l_1 is the base link, link l_2 is rigid link that can freely rotate about the hinge N_1. Finally, we have the compliant link which is composed of links l_3 and l_4 joined together at hinge N_3 via a compliant hinge. The links l_1 and l_4 are constrained to be parallel to each other throughout the motion of the mechanism. that is We need one more parameter to fully-define the link lengths of the compliant mechanism. We choose the angle between links l_1 and l_2 in fully-contracted state (θ_{02}) as the third design parameter. Now we use loop closure equations to find the lengths of links l_2 and l_3 and then angle θ_{03}.

$$l_2 = \frac{(L_0 - L_1)}{2\sin\theta_{02}} \tag{4.1}$$

$$l_3 = \sqrt{(l_1 - l_4 - l_2\cos\theta_{02})^2 + (\frac{L_0 + L_1}{2})^2} \tag{4.2}$$

$$\theta_{03} = \arcsin\frac{(L_0 + L_1)}{2l_3} \tag{4.3}$$

During the motion the energy is stored and released in the compliant hinge, which is denoted by the change in the angle of hinge N_3 given by μ_3. Additionally, there is energy difference between two stable states, damping, and friction that has to be accounted for. We assume that it is proportional to the change in hinge angle N_2 given by μ_2. We use MATLAB fmincon function to find the intermediate values of angles θ_2 and θ_3 for different lengths L ($L_0 \leq L \leq L_1$), since we can't solve the equation directly. We agin invoke the loop closure equation to find the values of θ_2 and θ_3 that satisfy the following equations.

$$L = l_2\sin\theta_2 + l_3\sin\theta_3 \tag{4.4}$$

$$l_1 - l_4 = l_2\cos\theta_2 + l_3\cos\theta_3 \tag{4.5}$$

$$\tag{4.6}$$

Let the compliant hinge stiffness be K_3 and K_2 refers to the stiffness introduced to account for the friction between components, damping and the energy difference between two stable states. The total energy of the bistable mechanism during the motion is then given as,

$$\mu_2 = (\theta_2 - \theta_{02}) \tag{4.7}$$

$$\mu_3 = -(\theta_3 - \theta_{03}) \tag{4.8}$$

$$E = \frac{1}{2}K_2\mu_2^2 + \frac{1}{2}K_3\mu_3^2 \tag{4.9}$$

Now that we have expression for the energy of a compliant mechanism we combine multiple such mechanisms to create the compliant robotic skeleton as described in following section.

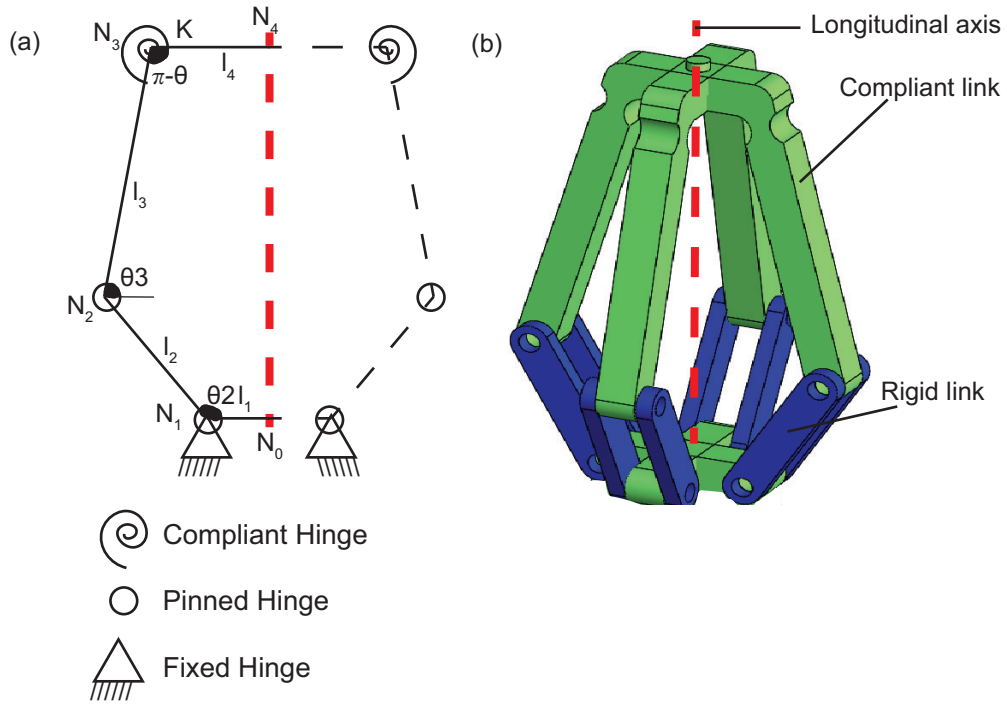

Figure 4.1: Geometry of the bi-stable compliant mechanism. (a) The geometric and design parameters of the compliant slider-crank mechanism. (b) 3-D rendering of the compliant bi-stable mechanism depicting longitudinal axis.

Table 4.1: Geometric Parameters for bidirectionally crawling robot driving module

Segment I Tri-stable mechanism		Segment II Bi-stable mechanism	
Parameter	Size	Parameter	Size
l_t (mm)	[20 32 70 20]	l_b	[20 20 76 20]
l_{tb} (mm)	[20 20 40 20]		
θ_{it}	[4.19 1.34]	θ_{ib}	[4.19 1.44]
θ_{itb}	[4.19 1.32]		
K_t	200	K_b	1100
K_{tb}	100		

4.2.1 Compliant Robotic Skeleton

We start with the 'generic' bistable mechanism inspired from slider crank type mechanism. One potential design for the driving module of the bidirectionally crawling robot is composed of spatially arranged serially connected tri-stable and bi-stable mechanism as shown in figure 4.3(b). The geometric parameters of these segments are detailed in Table 4.1. Next, we use the Equilibrium paths search algorithm described in Appendix B to find the equilibrium path followed by the system under extension and contraction phases. The energy plots of the individual segments in the driving module are shown in Figure 4.2(a). We need only a single linear actuator to control the total length (l_t) of the robot and to generate bi-directional actuation cycles.

4.3 Actuation Cycle Formation

The numerical simulation for the compliant multi-stable mechanism described in table 4.1 leads to equilibrium path that can lead to the bidirectional locomotion. We start by stretching the driving module when its two segments are both at its fully-contracted stable state (0) (point a in Figure 4.2(b)). During the stretching, the compliant segments deform by following the equilibrium path $a \rightarrow j \rightarrow b \rightarrow c \rightarrow d \rightarrow e \rightarrow f$ until both of them reach the fully-extended stable state (1) (point f). Then, we compress the driving module and observe that the segments follow a different equilibrium path $f \rightarrow e \rightarrow g \rightarrow h \rightarrow i \rightarrow j \rightarrow a$ until they come back to the state (0).

We observe four distinct jumps in these equilibrium paths. Two jumps occur during the stretching: first one from $b \rightarrow c$, and the second one from $d \rightarrow e$. The next two jumps occur during the compression: first one from $g \rightarrow h$, and the second one from $i \rightarrow j$ (Figure 3.3(b)). When

these jumps occur, a branch of local energy minima reaches its end so that the driving module is forced to quickly "jump" to a distant branch of energy minima. During these jumps, the two compliant segments change their length significantly, while their total length (l_t) remains the same. By combining parts of these equilibrium paths and the four jumps, we can construct two "actuation cycles":(Cycle-1) $j \to b \to c \to h \to i \to j$; (Cycle-2) $h \to c \to d \to e \to g \to h$.

Each of these actuation cycles consist of four consecutive "phases": For Cycle-1, in Phase I ($j \to b$), both segments increase in length significantly until Segment II fully-extends. Phase II ($b \to c$) is the first jump, by which Segment I quickly extends to a large length, but Segment II contracts to fully-contracted state. In Phase III ($c \to h \to i$), Segment I continues to contract in length, and Segment I remains in a fully-contracted state. The final Phase IV ($i \to j$) is the second jump, by which Segment I quickly deforms to its fully-contracted state, but Segment II extends in length.

For Cycle-2, in Phase I ($h \to c \to d$), both segments increase in length significantly until Segment I fully-extends. Phase II ($d \to e$) is the first jump, by which Segment I quickly contracts to a small length, but Segment II extends to almost fully-extended state. In Phase III ($e \to g$), Segment II continues to contract in length, and Segment I remains at a semi-contracted state. The final Phase IV ($g \to h$) is the second jump, by which Segment I quickly extends to its almost fully-extended state, but Segment II contracts to fully-contracted state.

4.4 Bidirectional Locomotion Gait Generation

We combine the dual-segment multi-stable driving module with the properly designed anchors to harness the two actuation cycles and generate a peristaltic-like bi-directional crawling locomotion gait. We designate Cycle-1 as forward locomotion cycle and Cycle-2 as backward locomotion cycle. The anchors for Segment I are designed such that, in Cycle-1 the *"active"* anchors are located on part B-1 and in Cycle-2 they are located on part B-2 (Figure 4.3(a)). This clever alternate activation of the anchors on Segment I makes bi-directional locomotion possible. Now, the four consecutive phases in the actuation cycle can be used to alternate the anchoring locations between the head and tail of the driving module, resulting in a net forward/backward displacement as detailed below:

In the **forward locomotion Cycle-1** we have four phases as shown in Figure 4.4(a): In

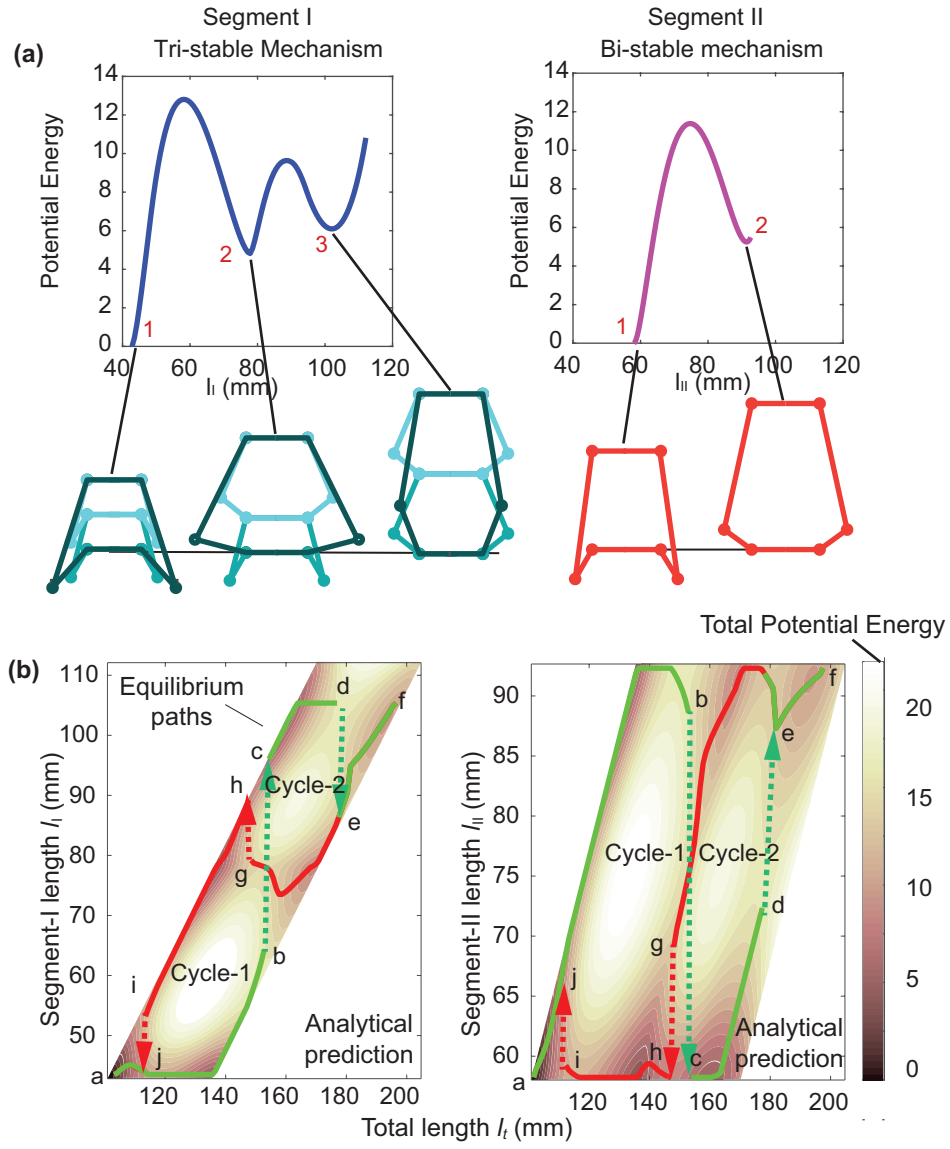

Figure 4.2: Actuation cycle formation for Bi-directional crawling locomotion gait. (a) The energy plots for individual segments in the driving module. The stable configurations of tri-stable and bi-stable compliant mechanisms are depicted underneath. (b) The equilibrium paths for the driving module of bi-directionally crawling robot clearly show the formation two cycles with Cycle-1 depicting forward locomotion actuation cycle and Cycle-2 depicting backward locomotion actuation cycle. Green curve represents extension of the driving module and the red curve represents contraction. The jumps are depicted by dashed lines with arrows showing direction of the jump. The energy landscape for the driving module is superimposed for added clarity. The lighter color shows region of high energy and vice-a-versa.

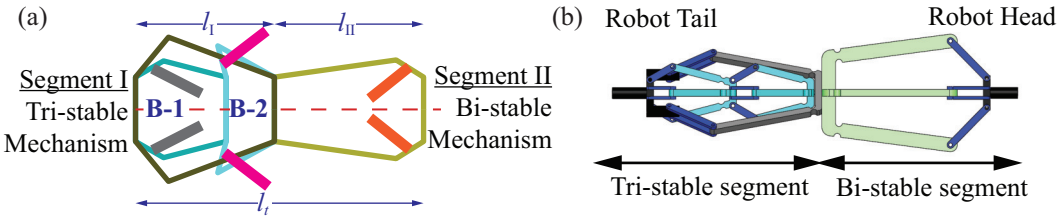

Figure 4.3: The driving module of compliant robot for bidirectional locomotion (a) Tri-stable and bi-stable segment schematic with anchors attached to the bi-stable elements. The *"active"* anchor locations on Segment I are also shown. (b) Bi-directional crawling robot 3-D rendering with compliant and rigid links.

Phase I ($j \rightarrow b$), the crawling robot is anchored at its tail because its Segment I is below its cut-off length. Meanwhile, the robot body is increasing in its total length by the actuator input, giving a net forward displacement.

In Phase II ($b \rightarrow c$), the jump between the equilibrium paths switches the anchor location from the tail to the head. No head or tail displacement occurs during this jump.

In Phase III ($c \rightarrow h \rightarrow i$), the crawling robot is anchored at its head because its Segment II is now below its cut-off length. Meanwhile, the robot body is contracting in its total length, moving the tail forward.

In the final Phase IV ($i \rightarrow j$), the second jump occurs and the anchor location switches back from head to the tail. At the end of this phase, the crawling robot returns to its original configuration of the actuation cycle, i.e. at the start of Phase I. The "gait length" is the total forward movement of the crawling robot head after one actuation cycle. And the actuation cycle from Phase I to Phase IV can be repeated to drive the robot forward continuously.

In the **backward locomotion Cycle-2** we have four phases as shown in Figure 4.4(b): In Phase I ($h \rightarrow c \rightarrow d$), the crawling robot is anchored at its head because its Segment II is below its cut-off length. Meanwhile, the robot body is increasing in its total length by the actuator input, giving a net backward displacement.

In Phase II ($d \rightarrow e$), the jump between the equilibrium paths switches the anchor location from the head to the tail. No head or tail displacement occurs during this jump.

In Phase III ($e \rightarrow g$), the crawling robot is anchored at its tail because its Segment II is now below its cut-off length. Meanwhile, the robot body is contracting in its total length, moving the head backward.

In the final Phase IV ($g \rightarrow h$), the second jump occurs and the anchor location switches

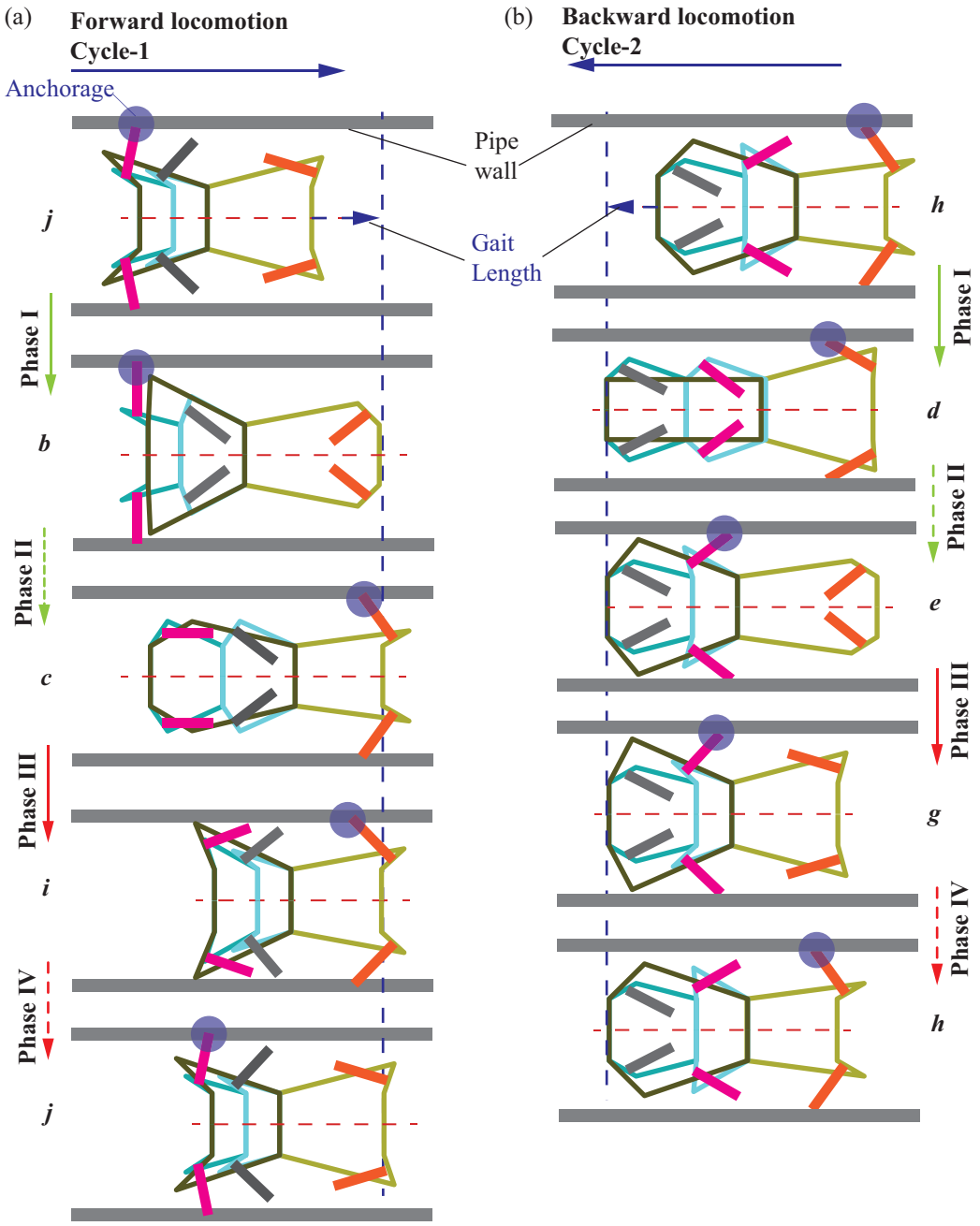

Figure 4.4: Generation of bi-directional peristaltic-like crawling locomotion gait. (a) Forward locomotion gait generation i.e. Cycle-1. The corresponding actuation cycle phases are depicted for clarity. The anchors on part B-2 of Segment I are *"inactive"* for this cycle. The *"active"* anchors on part B-1 of Segment I are depicted in pink and the same ones on Segment II are depicted in orange. (b) Backward locomotion gait generation i.e. Cycle-2. The corresponding actuation cycle phases are depicted for clarity. The anchors on part B-1 of Segment I are *"inactive"* for this cycle. The *"active"* anchors on part B-2 of Segment I are depicted in pink and the same ones on Segment II are depicted in orange.

back from tail to the head. At the end of this phase, the crawling robot returns to its original configuration of the actuation cycle, i.e. at the start of Phase I. The "gait length" is the total backward movement of the crawling robot tail after one actuation cycle. And the actuation cycle from Phase I to Phase IV can be repeated to drive the robot backward continuously.

To switch the locomotion gait from forward to backward and vice-a-versa we only need to change the total length (l_t) of the driving module between h and c. If we continue to increase length of the driving module from h we will switch to backward locomotion Cycle-2. If we continue to decrease length of the driving module from c we will switch to forward locomotion Cycle-1.

4.5 Compliant mechanism fabrication and experimental results

We 3-D printed the compliant links of bi-stable and tri-stable mechanisms out of polypropylene material in Ultimaker-S5. We used circular contour type notch flexure hinge for compliant link joint. The resultant mechanisms are depicted in Figure 4.1(b) and Figure 4.3(b). We experimented with different hinge parameters and link thicknesses to find how the hinge stiffness and ultimately the energy of the mechanism changes for different link lengths. As shown in Figure 4.5 we see that higher hinge width and link thickness leads to larger compliant hinge stiffness K_3. But the same cannot be said true for K_2. The variation in K_2 are much harder to characterize, and it looks very difficult that we would be able to tailor the energy of second stable state accurately with current fabrication techniques. The 3-D printing errors and local material property variations, and fabrication errors also make it very difficult to replicate the experimental results for multiple mechanisms with same design parameters.

4.6 Conclusion and summary

In our simulations we found that the design space for bidirectional locomotion generation using multi-stable mechanisms designed using current technique is quite narrow. The fabrication methods like 3-D printing are really good for prototyping and lead to successfully bistable compliant mechanisms every time. But it is difficult to replicate the experimental results for energy-displacement curve and consequently the hinge stiffness values. Our initial investigations with Kresling origami

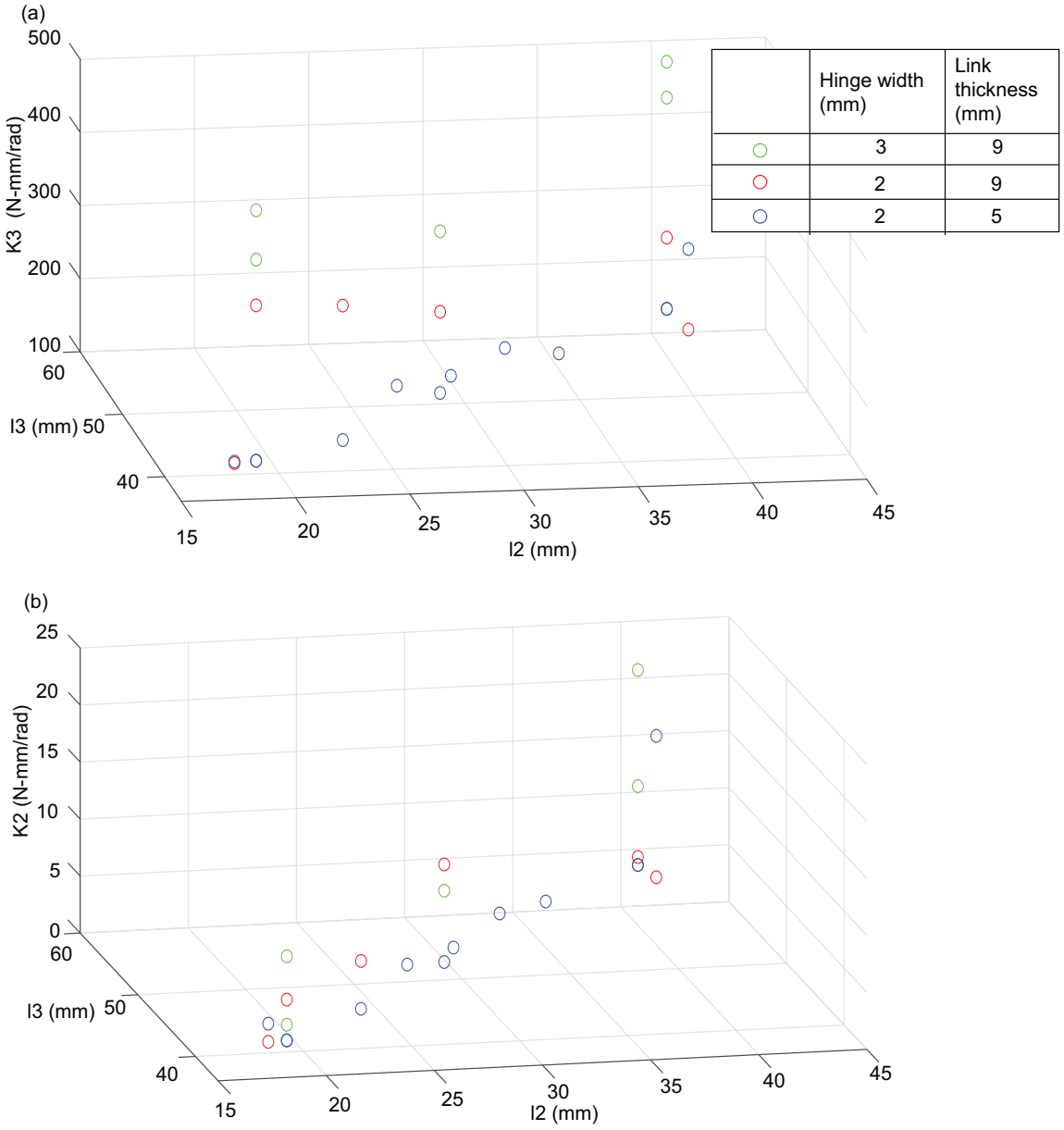

Figure 4.5: Results of the experiment to characterize hinge stiffness w.r.t. the design parameters. (a) Hinge stiffness of compliant link K_3 w.r.t. link lengths l_2, l_3, and different combinations of compliant hinge width and link thicknesses. (b) Hinge stiffness of compliant link K_2 w.r.t. link lengths l_2, l_3, and different combinations of compliant hinge width and link thicknesses.

bistable mechanism resulted in repeatable results and the bistability strength and characteristics were function of mainly one parameter the angle ratio in the design space when other design parameters were held constant. In our parametric studies we found that the relative bistability strengths of the two segments of the driving module in Kresling origami were sufficient to provide required actuation cycle and there was certain flexibility in terms of choosing different designs, which guaranteed the motion-sequencing for a large design space. In contrast to this behavior, the design space of the driving module for bidirectional locomotion is quite narrow. It is also difficult to accurately predict and fabricate the exact stiffness distribution that we want in the 3-D printed compliant mechanism, so that led to failure to generate bidirectional actuation cycle in our experiment.

Theoretically if we could find bistable and tristable mechanisms with the energy-displacement curves as shown in Figure 4.2 then we would definitely have bidirectional locomotion. Our initial hypothesis was that with the advanced and accurate 3-D printer at our disposal we will be able to have better control over the stiffness properties of the fabricated mechanisms, but the performance of the 3-D printer for soft polymers cannot be guaranteed in the same way as the stiff polymers. The material properties of the 3-D printer material also cannot be guaranteed to be uniform throughout. That and the fact that the design space of the driving module capable of bidirectional locomotion is too 'narrow' leads us to think that maybe we need to look for alternatives for the bistable mechanisms. So we keep this topic as part of future research.

Chapter 5

Physical Reservoir Computing with Origami

Abstract

The field of soft robotics is extensively influenced by biological systems, which are composed of a highly interconnected network of sensors and actuators spread throughout the body, that interact with the environment and enable them to perform various tasks efficiently. Similarly, to successfully implement the soft robots in an unstructured dynamic environment, the robot body and *the brain* a.k.a. central controller have to continuously and efficiently exchange the information about the current state of the robot and the surrounding environment and take appropriate actions. Recently, a new paradigm has emerged in artificial recurrent neural network training called *physical reservoir computing*, in which physical dynamical systems are used as computational reservoirs to perform complex computing tasks – such as, approximating non-linear dynamical systems, pattern generation, and even machine learning. We propose that an origami meta-material can also be used as a dynamic reservoir. Furthermore, we show how physical reservoir computing can be implemented in soft origami robots through the example of an earthworm-inspired robot. The results of this work will pave the way for intelligently designed origami-based robots with *embodied intelligence*. These next generation of soft robots will be able to coordinate and modulate their activities autonomously such as switching locomotion gait and resisting external disturbances while

navigating through unstructured environments.

5.1 Introduction

The animal kingdom is an endless source of inspiration for soft robotics [98, 76]. Researchers have constructed compliant robots that can mimic all kinds of animal motions, like octopus locomotion [14], elephant trunk grasping [47], insect flying [90], jellyfish and fish swimming [64, 131, 67], as well as snake and insects crawling [130, 164, 143]. These robots share many similarities with animals regarding their shape and motion kinematics; however, their underlying sensing, actuation, and control architectures could be fundamentally different. Our engineered soft robots typically rely on a centralized controller (aka. an "electronic brain") that takes up all computing work to process sensor information, generate control commands, and make decisions. This approach often struggles to achieve high actuation speed and control effectiveness as soft robots exhibit virtually infinite degrees of freedom and complicated dynamic characteristics. On the other hand, animals have highly interconnected networks of nerves and muscles that can share the workload with the brain [52, 74, 153]. The animal body's morphology is an integral part of its actuation, control, and ultimately its "brain's" decision-making process, leading to far superior efficiency than our engineered soft robots.

Motivated by this disparity, an increasing number of researchers have embraced soft bodies' nonlinear dynamics as a computational resource to create an embodied intelligence and control [121, 122, 9, 49, 108, 105, 150]. As a result, a new computational paradigm called *morphological computation* has emerged in which the physical body of the robot itself takes part in performing low-level control tasks, such as locomotion coordination and modulation, to simplify the overall control architecture significantly [121, 122, 49, 9, 39]. The contributions of body morphology to cognition and control involve three major categories [105]: (1) Morphology facilitating control: wherein the physical design enables certain behaviors such as motion sequencing (e.g., passive dynamic walker [15]). (2) Morphology facilitating perception: wherein the physical design enables sensing (e.g., the nonuniform distribution of cells in the compound eyes of fly [35]). (3) Morphological computation, such as the *physical reservoir computing* (PRC), wherein a physical body performs genuine computations. Among these, physical reservoir computing shows promising potentials because of its balanced simplicity and versatility to perform applicable computation with encoding and decoding

[105].

Reservoir computing is a computational framework based on artificial recurrent neural networks (RNNs), which have been used extensively for problems involving time-series prediction like the stock market and weather forecasting, robotic motion planning and control, text and speech recognition [56, 93, 92, 91, 89, 142, 150, 106]. In RNNs, the output of the current time step depends on the results from the previous time step in addition to the current input. Since RNNs involve both forward and back-propagation of input data, training them became a challenging task. To address this difficulty, Jaeger introduced the concept of a *fixed* recurrent neural network as Echo State Networks (ESNs) [56], and Maass introduced Liquid State Machines (LSMs) [93]. Later, these two concepts merged under the umbrella of reservoir computing (RC). In RC, the neural network (aka. the "reservoir") has fixed interconnections and input weights, and only the linear output readout weights are trained by simple techniques such as, linear or ridge regression. The reservoir's dynamics transforms the input data stream into a high-dimensional state space, capturing its nonlinearities and time-dependent information for computation tasks.

More importantly, the reservoir's fixed nature opens up the possibility of using physical bodies — such as a random network of nonlinear spring and mass oscillators [49, 50, 102], tensegrity structures [121, 122, 10, 9], and soft robotic arms [108, 85, 109] — to conduct computation, hence the paradigm of *Physical* Reservoir Computing. These physical systems have been shown to possess sufficient computational power to achieve complex computing tasks e.g. emulating other nonlinear dynamic systems, pattern generation [10, 9, 49, 50, 108, 150], speech recognition [29], and machine learning [109, 150, 106, 102]. More importantly, robotic bodies with sufficient nonlinear dynamics can also perform like a physical reservoir and directly generate locomotion gait without using the traditional controllers [9, 150, 17, 1, 155]. Despite the recent progress, physical reservoir computing is still a nascent field, and it is worthwhile to examine the computing power of a wide variety of different compliant mechanical systems, especially those with broad application potential in soft robotic locomotion and intelligent structures. These dynamic physical reservoirs with embeded feedback can function as soft robotic skeletons, and simultaneously generate and maintain the periodic trajectories essential for animal-inspired locomotion gait generation.

One such compliant mechanism that has garnered much attention over recent years is Origami – a traditional play of folding paper into sophisticated and three-dimensional shapes. Origami mechanisms are compact, easy to fabricate, and scale-independent (aka. Origami robots can

be fabricated at different scales but still follow similar folding principles [124, 133, 111, 104]). Origami mechanisms with complex crease folding patterns and integrated actuator-sensor network exhibit many desirable soft body properties. Similar to the soft and compliant biological systems, origami features non-linear kinematics and dynamics, non-linearly varying stiffness, large deformation range, compliance, and shape-morphing capability. The origami mechanisms are stiff enough to be used as structural skeleton for robots, and at the same time are compliant enough to provide large deformation range required for soft robotic locomotion. Over the past decades, origami has evolved into an engineering framework for constructing multi-transformable deployable structures [31, 103, 104, 96], advanced materials and shape morphing structures [141, 144, 170, 84, 166, 66, 124, 111]. It is already a popular engineering platform for constructing soft robotic skeletons that mimic wide range of animal motions, e.g. worm-like crawling, insect-like walking, and octopus arm-inspired manipulation [133, 101, 100, 3, 114, 115, 165, 112, 26, 58, 130, 68]. Moreover, the nonlinear mechanics and dynamics induced by folding could also enhance robotic performance [5, 172, 135]. Thus, we investigate the use of origami as a physical reservoir and show that origami based robotic skeleton can indeed generate the periodic patterns for autonomous locomotion gait generation.

We show that origami's nonlinear folding dynamics possesses significant computing power, which could add a valuable dimension to the field of origami-based engineering. A mechanical system must exhibit several essential properties to perform as a reservoir [150]. The first one is high-dimensionality, which allows the reservoir to gather as much information possible from the input data stream, separating its spatio-temporal dependencies and projecting it onto a high-dimensional state-space. The second one is nonlinearity so that the reservoir acts as a nonlinear filter to map the information from the input stream. All the computation complexity is associated with this nonlinear mapping, thus training the linear static readout becomes a straightforward task. The third one is fading memory (or short-term memory), ensuring that only the recent input history influences the current output. The fourth one is separation property to classify and segregate different response signals correctly, even with small disturbances or fluctuations. Moreover, if two input time series differed in the past, the reservoir should produce different states at subsequent time points [78]. Our physics-informed numerical simulations prove that origami inherently satisfies these four requirements and can complete computation tasks like emulation, pattern generation, and output modulation.

Moreover, we conduct extensive numerical simulations to uncover the linkage between

origami design and its computing power, providing the guideline to optimize computing performance. Finally, we demonstrate how to directly embed reservoir computing in an origami robotic body to generate earthworm-like peristalsis crawling without using any traditional controllers. This study's results could foster a new family of origami-based soft robots that operate with simple mechatronics, interact with the environment through distributed sensor and actuator networks, and respond to external disturbances by modulating their activities.

In what follows: Section (5.2) details the construction of an origami reservoir, including the lattice framework used to simulate its nonlinear dynamics. Section 5.3 elucidates the origami reservoir's computing power through various numerical experiments. Section 5.5 discusses the parametric analysis that uncovers the linkages between computing performance and physical design. Section 5.6 applies the reservoir computing to an origami robot's crawling problem. Finally, Section 5.7 concludes this paper with a summary and discussion.

5.2 Constructing The Origami Reservoir

In this study, we construct a physical reservoir using the classical Miura-ori sheets. We can easily modify basic Miura-ori geometry to create complex structures such as curved Miura-ori surfaces, stacked Miura-ori, or origami-tubes with various cross sections. We show that origami can indeed act as a reservoir and even simplest of origami pattern can be turned into peristaltic crawling robot powered by reservoir computing. Miura-ori is essentially a periodic tessellation of unit cells, each consisting of four identical quadrilateral *facets* with *crease* lengths a b and an internal sector angle γ (Figure 5.1(a)) [138, 141]. The folded geometry of Miura-ori can be fully defined with a dihedral *folding angle* θ ($\in [0, \pi/2]$) between the x-y reference plane and its facets (Figure 5.1(b)). The reservoir size is defined as $n \times m$, where n and m are the number of origami *nodes* (aka. vertices where crease lines meet) in x and y-directions, respectively. N is the total number of creases in the origami reservoir.

5.2.1 Dynamics Modeling of the Origami

To investigate this origami reservoir's computing capacity, one must first obtain its time responses under dynamic excitation. To this end, we adopt and expand the lattice framework approach to simulate its nonlinear dynamics [138, 86, 41]. In this approach, origami creases are

represented by pin-jointed stretchable truss elements with prescribed spring coefficient K_s. Folding (or bending) along the crease line is simulated by assigning torsional spring coefficient K_b. We further triangulate the quadrilateral facets with additional 'virtual' truss elements to estimate the facet bending with additional torsional stiffness (typically, K_b across the facets is larger than those along the creases). The rationale behind adding these 'virtual' facet bending creases can be explained with the example of Miura-ori structure. The ideal Miura-ori structure is rigid foldable when the in-plane folding motion is the primary deformation mode. Meanwhile, Miura-ori also exhibits out-of-plane twisting and saddle-shaped deformations that can be prominent in dynamic responses [138, 141]. More importantly, these non-rigid folding deformations are desirable for reservoir computing. The presence of two out-of-plane deformation modes depends mainly on the ratio between facet bending stiffness ($k_{b,f}$) and crease folding stiffness (k_b). When $k_{b,f}/k_b >> 1$, we can assume Miura-ori to be rigid-foldable. When $k_{b,f}$ and k_b are comparable, out-of-plane deformation modes can be significant. The non-rigid foldable behavior of many origami mechanisms and metastructures has been studied with this approach, for example, square twist pattern [145], Kresling pattern [5], and Miura-ori pop-through defect [86].

In essence, lattice framework approach discretizes the continuous origami sheet into a network of pin-jointed truss elements connected at the nodes (Figure 5.1(c)). A typical reservoir consists of an interconnected network of units governed by nonlinear dynamics. The origami reservoir, in this case, consists of a network of nodes with their interconnections defined by the underlying crease pattern. It's important to note that our approach does not include the non-linear/ hyperelastic material constitutive model, the effects due to finite material thickness, viscosity, and non-linear stiffness changes, etc. The corresponding governing equations of motion, in terms of node #p's displacement ($\mathbf{x}_p$) as an example, are:

$$m_p \ddot{\mathbf{x}}_p^{(j)} = \mathbf{F}_p^{(j)}, \tag{5.1}$$

where the superscript "(j)" represents the j^{th} time step in numerical simulation, and m_p is the equivalent nodal mass. Unless noted otherwise, the mass of the origami sheet is assumed to be equally distributed to all its nodes. $\mathbf{F}_p^{(j)}$ is the summation of internal and external forces acting on this node in that

$$\mathbf{F}_p^{(j)} = \sum \mathbf{F}_{s,p}^{(j)} + \sum \mathbf{F}_{b,p}^{(j)} + \mathbf{F}_{d,p}^{j} + \mathbf{F}_{a,p}^{(j)} + m_p \mathbf{g}, \tag{5.2}$$

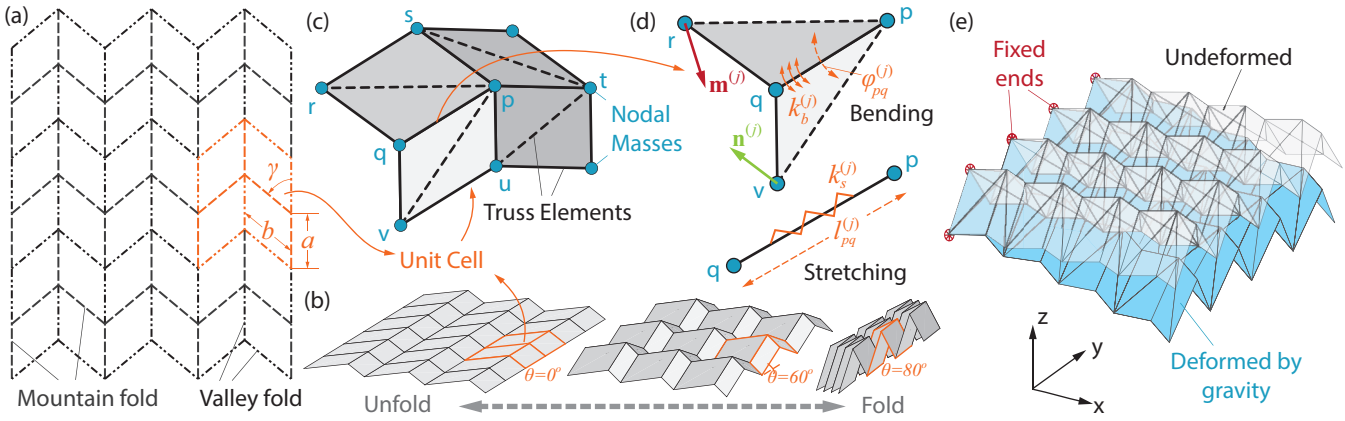

Figure 5.1: The nonlinear Truss-frame approach for simulating the origami dynamics. (a) The crease pattern of the classical Miura-ori, with a unit cell highlighted. (b) The rigid-folding kinematics of the Miura-ori. (c) The truss-frame approach discretizes the Miura-ori unit cell, showing the distribution of truss elements along the creases and across the facets, as well as the nodal masses. (d) Detailed kinematics and mechanics set up to analyze the bending and stretching along the truss $\#pq$. Notice that $\mathbf{m}^{(j)}$ and $\mathbf{n}^{(j)}$ are the current surface normal vectors defined by triangles $\#pqr$ and $\#pqv$, respectively. (e) The bending of the Miura-ori sheet under its weight. This simulation serves to validate appropriate material property assignments.

where the five terms on the right hand side are the forces from truss stretching, crease/facet bending, equivalent damping, external actuation, and gravity, respectively. The formulation of these forces are detailed below.

Truss stretching forces: The truss elements are essentially elastic springs with axial stretching stiffness ($K_s^{(j)} = EA/l^{(j)}$). Here, EA is the material constant, and $l^{(j)}$ is the truss element's length at the current j^{th} time step. Thus, the axial stiffness is updated at each time-step, accommodating the truss element's increase in stiffness as it is compressed and vice-a-versa. The stretching forces from a truss connecting node #p and one of its neighbouring nodes $\#q$ is,

$$\mathbf{F}_{s,p}^{(j)} = -K_s^{(j)} \left(l_{pq}^{(j)} - l_{pq}^{(0)} \right) \frac{\mathbf{r}_p^{(j)} - \mathbf{r}_q^{(j)}}{|\mathbf{r}_p^{(j)} - \mathbf{r}_q^{(j)}|} \tag{5.3}$$

where $l_{pq}^{(0)}$ is the truss length at its initial resting state. $\mathbf{r}_p^{(j)}$ and $\mathbf{r}_q^{(j)}$ are the current position vectors of these two nodes, respectively. To calculate the total truss stretching forces acting on node $\#p$, similar equations apply to all of its neighbour nodes through trusses (e.g., node q, r, s, t, u, and v in Figure 5.1(c)).

Crease/facet bending forces: The crease folding and facet bending are simulated with torsional spring coefficient ($K_b^{(j)} = k_b l^{(j)}$), where k_b is torsional stiffness *per unit length*. Here, we adopt the formulation developed by Liu and Paulino [86]. For example, if the stored potential energy due to the crease folding along the truss between $\#p$ and $\#q$ is: $\mathbf{u}_{pq}^{(j)} = \frac{1}{2}K_b^{(j)}(\varphi_{pq}^{(j)} - \varphi_{pq}^{(0)})^2$, then the force acting on nodes $\#p$ is:

$$\mathbf{F}_{b,p}^{(j)} = -\frac{\partial \mathbf{u}_{pq}^{(j)}}{\partial \varphi_{pq}^{(j)}} \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}} = -K_b^{(j)}(\varphi_{pq}^{(j)} - \varphi_{pq}^{(0)}) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}} \tag{5.4}$$

where $\varphi_{pq}^{(j)}$ is the current dihedral angle along truss pq (aka. the dihedral angle between the triangles $\#pqr$ and $\#pqv$ in 5.1(d)), and $\varphi_{pq}^{(0)}$ is the corresponding initial value. $\varphi_{pq}^{(j)}$ can be calculated as

$$\varphi_{pq}^{(j)} = \eta \arccos\left(\frac{\mathbf{m}^{(j)} \cdot \mathbf{n}^{(j)}}{|\mathbf{m}^{(j)}||\mathbf{n}^{(j)}|}\right) \text{ modulo } 2\pi \tag{5.5}$$

$$\eta = \begin{cases} \text{sign}\left(\mathbf{m}^{(j)} \cdot \mathbf{r}_{pv}^{(j)}\right), & \mathbf{m}^{(j)} \cdot \mathbf{r}_{pv}^{(j)} \neq 0 \\ 1. & \mathbf{m}^{(j)} \cdot \mathbf{r}_{pv}^{(j)} = 0 \end{cases} \tag{5.6}$$

Here, $\mathbf{m}^{(j)}$ and $\mathbf{n}^{(j)}$ are current surface normal vector of the triangles $\#pqr$ and $\#pqv$, respectively, in that $\mathbf{m}^{(j)} = \mathbf{r}_{rq}^{(j)} \times \mathbf{r}_{pq}^{(j)}$ and $\mathbf{n}^{(j)} = \mathbf{r}_{pq}^{(j)} \times \mathbf{r}_{pv}^{(j)}$. In addition, $\mathbf{r}_{pq}^{(j)} = \mathbf{r}_p^{(j)} - \mathbf{r}_q^{(j)}$, $\mathbf{r}_{rq}^{(j)} = \mathbf{r}_r^{(j)} - \mathbf{r}_q^{(j)}$, and $\mathbf{r}_{pv}^{(j)} = \mathbf{r}_p^{(j)} - \mathbf{r}_v^{(j)}$. This definition of $\varphi_{pq}^{(j)}$ ensures that the folding angle for valley crease lies in $(0, \pi]$ and the folding angle for mountain crease lies in $(\pi, 2\pi]$. The derivative between folding angle $\varphi_{pq}^{(j)}$ and the nodal $\#p$'s current position vector is

$$\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}} = \left(\frac{\mathbf{r}_{pv}^{(j)} \cdot \mathbf{r}_{pq}^{(j)}}{|\mathbf{r}_{pq}^{(j)}|^2} - 1\right) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_v^{(j)}} - \left(\frac{\mathbf{r}_{rq}^{(j)} \cdot \mathbf{r}_{pq}^{(j)}}{|\mathbf{r}_{pq}^{(j)}|^2}\right) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_r^{(j)}} \tag{5.7}$$

where

$$\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_r^{(j)}} = \frac{|\mathbf{r}_{pq}^{(j)}|}{|\mathbf{m}^{(j)}|^2} \mathbf{m}^{(j)}, \tag{5.8}$$

$$\frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_v^{(j)}} = -\frac{|\mathbf{r}_{pq}^{(j)}|}{|\mathbf{n}^{(j)}|^2} \mathbf{n}^{(j)}. \tag{5.9}$$

Again, to calculate the total crease folding and facet bending forces acting on node $\#q$,

similar equations apply to trusses connected to this node (e.g., truss pq, pr, ps, pt, pu, and pv in Figure 5.1(d)).

Damping forces: Estimating damping ratio and damping force is essential to achieve realistic dynamic responses and reduce numerical simulation error accumulation. In this study, we follow the formulation developed in [51, 41]. This formulation first calculates the average velocity of a node with respect to its neighboring nodes ($\mathbf{v}_{\text{avg}}^{(j)}$) to effectively remove the rigid body motion components from the relative velocities and ensure that these components are not damped. Then damping force $\mathbf{F}_{d,p}^{(j)}$ applied on node $\#p$ is given by

$$\mathbf{F}_{d,p}^{(j)} = -c_d^{(j)}(\mathbf{v}_p^{(j)} - \mathbf{v}_{\text{avg}}^{(j)}) \tag{5.10}$$

$$c_d^{(j)} = 2\zeta\sqrt{K_s^{(j)} m_p} \tag{5.11}$$

where $c_d^{(j)}$ is the equivalent damping coefficient, and ζ is the damping ratio.

Actuation force: In the origami reservoir, two types of creases receive actuation. The first type is "input creases," and they receive input signal $u(t)$ required for emulation and output modulation tasks. The second type is "feedback creases," and they receive reference or current output signal $z(t)$ required by all computing tasks in this study except for the emulation task. In the case of multiple outputs, different groups of feedback creases are present. Here, the selection of input and feedback creases are random. There are many methods to implement actuation to deliver input $u(t)$ and reference/feedback signal $z(t)$ to the reservoir. For example, the actuation can take the form of nodal forces on a mass-spring-damper network [49, 50], motor generated base rotation on octopus-inspired soft arm [108], or spring resting length changes in a tensegrity structure [10]. In origami, the actuation can take the form of moments that can fold or unfold the selected creases. We assume that the resting angle $\varphi^{(0)}$ of the input and feedback creases will change — in response to the actuation at every time step — to a new equilibrium $\varphi_{a,0}^{(j)}$ in that [123, 10]

$$\varphi_{a,0}^{(j)} = W_{\text{in}} \tanh(u^{(j)}) + \varphi^{(0)} \quad \text{for input creases;} \tag{5.12}$$

$$\varphi_{a,0}^{(j)} = W_{\text{fb}} \tanh(z^{(j)}) + \varphi^{(0)} \quad \text{for feedback creases.} \tag{5.13}$$

where W_{in} and W_{fb} are the input and feedback weight associated with these actuated creases. They are assigned before the training and remain unchanged after that. $u^{(j)}$ and $z^{(j)}$ are the

input and feedback signal at the j^{th} time step. The magnitude of W_{in} and W_{fb} are selected such that $\varphi_{a,0}^{(j)} \in [0, 2\pi)$ and consistent with the folding angle assignment. This approach of assigning new equilibrium folding angles is similar to traditional neural network studies that use tanh as a nonlinear activation function to transform function $z(t)$ into a new one with magnitudes between $[-1, 1]$. Additionally, it prevents actuator saturation due to spurious extreme values of $z(t)$. Denote the torsional stiffness of actuated creases by $K_{b,a}^{(j)}$, and we can update Equation (5.4) for the actuated creases (using node $\#p$ as an example)

$$\mathbf{F}_{a,p}^{(j)} = -K_{b,a}^{(j)} \left(\varphi_{pq}^{(j)} - \varphi_{a,0,pq}^{(j)} \right) \frac{\partial \varphi_{pq}^{(j)}}{\partial \mathbf{r}_p^{(j)}}, \tag{5.14}$$

The calculation of other terms in this equation are the same as those in the force from crease folding and facet bending. In this work, we focus on compliant sheet materials that can easily fold into origami shapes and provide sufficient compliance for soft robotic motions. We also want to ensure that our numerical simulation results are applicable to different material selections. The range of material parameters in aforementioned equations is set according to such considerations. For example, the k_s and k_b values in the baseline designs come from (1) averaged results from our prior experiments using folded thick paper and PET polymer sheets [68, 135], and (2) a careful survey of relevant literature [138, 141, 31, 144, 145, 86, 87]. It is worth noting that obtaining the equivalent k_s and k_b values is not trivial, and it depends on many material and geometric factors, including the origami facets' size and sheet material thickness, etc. Later in the parametric studies, we choose a relatively wide range of material properties to accommodate for such complexity. More importantly, we can also ensure the reservoir computing result can apply to origami reservoirs with different polymer/plastic-like material selections.

Once the governing equations of motion are formulated with above considerations, they are solved using MATLAB's `ode45` solver with 10^{-3} second time-steps. As an example, we show a simulation of the Miura-ori sheet deformation under its own weight (Figure 5.1 (e)). Although the governing equation of motions use nodal displacement $\mathbf{x}^{(j)}$ as the independent variables, we use the dihedral crease angles $\varphi^{(j)}$ as the *reservoir state* variables to characterize the origami's time responses. This is because measuring crease angles is easier to implement by embedded sensors, and $\varphi^{(j)}$ can be directly calculated from $\mathbf{x}^{(j)}$ via the Equations 5.5 and 5.6.

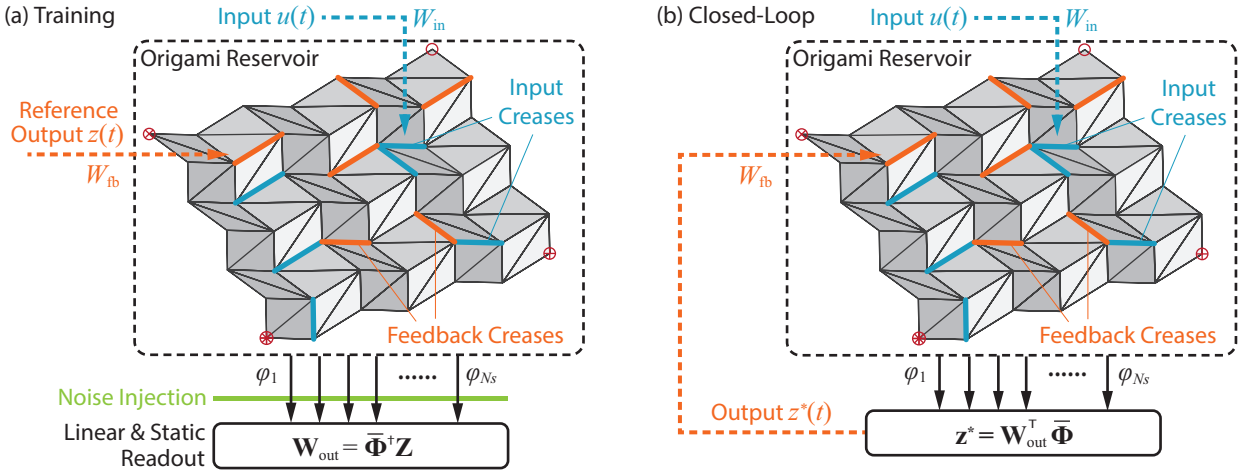

Figure 5.2: The setup of physical reservoir computing with origami. (a) The training phase. The feedback creases receive the reference (or targeted) output $z(t)$; while white noise is added to the reservoir state matrix $\bar{\mathbf{\Phi}}$ before calculating output weights $\mathbf{W}_{\text{out}}$; (b) The closed-loop phase. The output weights obtained in the training phase are used to calculate the current output, which is fed back to the feedback creases.

5.2.2 Setting Up Reservoir Computing

Similar to the actuated creases (aka. input creases and feedback creases), we designate "sensor creases" for measuring the reservoir states. We denote N_a as the number of actuated creases, and N_s for sensor creases. It is worth noting that, the actuated creases are typically small subset of all origami creases (i.e., $N_a < N$). The sensor creases, on the other hand, can be all of the origami creases ($N_s = N$) or a small subset as well ($N_s < N$).

Once the selections of input, feedback, and sensor creases are completed, one can proceed to the computing. Physical reservoir computing for tasks that require feedback, and output modulation consists of two phases: The "training phase" and "closed-loop phase." While the emulation tasks require the training phase only.

Training phase: In this phase, we use the teacher forcing to obtain the readout weights $\mathbf{W}_{\text{out}}$ corresponding to every reservoir state (aka. the dihedral angles of the sensor creases). Suppose one wants to train the reservoir to generate a nonlinear time series $z(t)$ (aka. the reference output). The feedback creases receive the reference output and it dynamically excites the origami reservoir under an open-loop condition without feedback (Figure 5.2(a)). The reservoir states $\varphi^{(j)}$ at every time step are measured and then compiled into a matrix $\mathbf{\Phi}$.

Table 5.1: Design of a baseline origami reservoir in this study

Reservoir size and material properties	
Parameter	Value
Size	9×9
Nodal Mass	7 g
EA	100 N
$k_{b,a}$	1 N/(m-rad)
k_b	0.2525 N/(m-rad)
Facets k_b	10 N/(m-rad)
ζ	0.2
Geometric design of Miura-ori	
Parameter	Value
a	16 mm
b	10 mm
γ	50°
θ	60°
Actuator and sensor creases	
Parameter	Value
No. of sensors (N_s)	N
No. of actuators (N_a)	$0.45N$
No. of Feedback creases	$0.3N$
No. of Input creases	$0.15N$

Once the numerical simulation is over, we segregate the reservoir state matrix $\mathbf{\Phi}$ into the washout step, training step, and testing step. The washout step data is discarded to eliminate the initial transient responses. We then calculate the output readout weights $\mathbf{W}_{\text{out}}$ using the training step data via simple linear regression:

$$\mathbf{W}_{\text{out}} = [\mathbf{1}\ \mathbf{\Phi}]^{+}\mathbf{Z} = \bar{\mathbf{\Phi}}^{+}\mathbf{Z} \tag{5.15}$$

where, $[.]^{+}$ refers to the Moore-Penrose pseudo-inverse to accommodate non-square matrix. $\mathbf{1}$ is a column of ones for calculating the bias term $W_{\text{out},0}$ to shift the fitted function when necessary. $\mathbf{Z}$ contains the reference signals at each time step, and it is a matrix if more than one references are present. Lastly, we use testing step data to verify reservoir performance. It is worth noting that white noise of amplitude 10^{-3} is superimposed on the reservoir state matrix during training to ensure the robustness of the readout result against numerical imperfections, external perturbations [50], and instrument noise in "real-world" applications.

Closed-loop phase: Once the training phase is over and readout weights are obtained, we run the reservoir in the closed-loop condition. That is, instead of using the reference output $z(t)$, the current output $z^*(t)$ is sent to the feedback creases (Figure 5.2(b)), and

$$z^*(t) = W_{\text{out},0} + \sum_{i=1}^{N_s} W_{\text{out},i}\varphi_i(t) = \mathbf{W}_{\text{out}}^T \bar{\mathbf{\Phi}} \tag{5.16}$$

where, N_s is the number of sensor creases, and $\bar{\mathbf{\Phi}} = [\mathbf{1}\ \mathbf{\Phi}]$. Thus, the reservoir runs autonomously in the closed-loop phase without any external interventions.

We study the closed loop performance of reservoir by calculating the Mean Squared Error (MSE) using M time-steps as follows:

$$\text{MSE} = \frac{1}{M}\sum_{j=1}^{M}\left(z(j) - z^*(j)\right)^2 \tag{5.17}$$

To estimate performance when multiple reference outputs are present, we combine the MSEs by taking a norm over the individual MSEs.

5.3 Computation Tasks By the Origami Reservoir

In this section, we use the origami reservoir to emulate multiple nonlinear filters simultaneously, perform pattern generation, and modulate outputs. The baseline variables for the origami geometric design, material properties, and reservoir parameters are given in Table 5.1.

5.3.1 Emulation Task

This sub-section shows that the origami reservoir can emulate multiple nonlinear filters simultaneously using a single input. Such emulation is a benchmark task for evaluating the performance in RNN training [2] and prove the multi-tasking capability of physical reservoirs [49, 108]. Note that the emulation task involves only the training phase, so there are no feedback creases in this case. Consequently, we excite the reservoir by sending the input function $u(t)$ to the input creases and train it to find three sets of readout weights in parallel via linear regression. Here, $u(t)$ is a product of three sinusoidal functions with different frequencies, and the three target nonlinear filters are 2^{nd}-order nonlinear dynamic system $z_1(t)$, a 10^{th}-order nonlinear dynamic system $z_2(t)$,

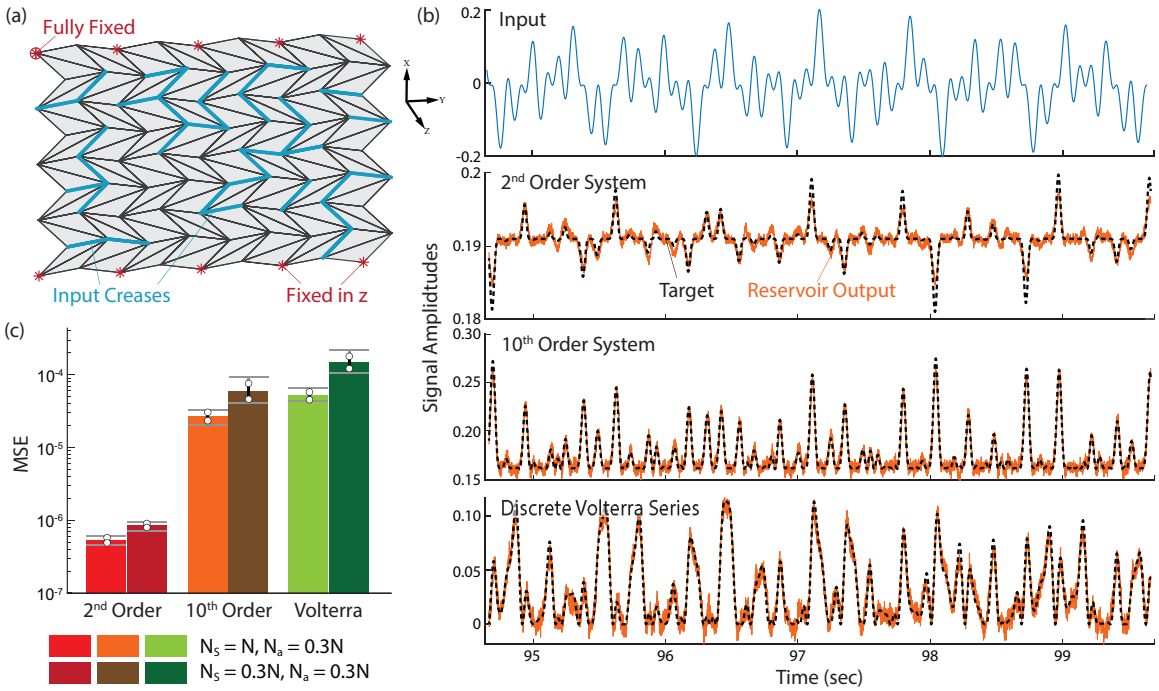

Figure 5.3: Emulation tasks with the origami reservoir. (a) The Miura-ori reservoir used for this task with input creases highlighted. Appropriate boundary conditions are also necessary. (b) Examples of trajectories generated in the emulation task including (from top to bottom) input signal $u(t)$, 2^{nd} order, 10^{th} order system, and Volterra series. Dashed curves are the targeted trajectories, and solid curves are the result of the reservoir. (c) Error analysis of the emulation tasks. Circles are the standard deviation of MSE, and horizontal bars are the corresponding extreme values.

Table 5.2: Emulation task functions

Type	Functions in discretized form (at j^{th} time step)
Input	$u(j) = 0.2\sin(2\pi f_1 j\Delta t)\sin(2\pi f_2 j\Delta t)\sin(2\pi f_3 j\Delta t)$ $f_1 = 2.11$ Hz, $f_2 = 3.73$ Hz, $f_3 = 4.33$ Hz, $\Delta t = 10^{-3}$
2^{nd} order system	$z_1(j+1) = 0.4z_1(j) + 0.4z_1(j)z_1(j-1) + 0.6(u(j\Delta t))^3 + 0.1$
10^{th}-order system	$z_2(j+1) = 0.3z_2(j-1) + 0.05z_2(j-1)\sum_{i=1}^{10} z_2(j-i)$ $+1.5u((j-10)\Delta t)u((j-1)\Delta t) + 0.1$
Discrete Volterra series	$z_3(j+1) = 100\sum_{\tau_1=0}^{T}\sum_{\tau_2=0}^{T} h(\tau_1,\tau_2)u(j-\tau_1)u(nj-\tau_2)$ $h(\tau_1,\tau_2) = \exp\left(\frac{(\tau_1\Delta t-\mu_1)^2}{2\sigma_1^2} + \frac{(\tau_2\Delta t-\mu_2)^2}{2\sigma_2^2}\right)$ $\mu_1 = \mu_2 = 0.1, \sigma_1 = \sigma_2 = 0.05$

and discrete Volterra series $z_3(n)$ (detailed in Table 5.2).

We use a 9×9 Miura-ori reservoir shown in Figure 5.3(a) in this task, exciting the reservoir from complete rest and training it for 100 seconds. We discard the first 50 seconds of data as the washout step, use the data from the next 45 seconds to calculate the optimum static readout weights, and then use the last 5 seconds of data to calculate the MSE for performance assessments. Results in Figure 5.3(b) show that the origami reservoir can emulate these three nonlinear filters. As the nonlinearity and complexity of the nonlinear filter increases, MSE also increases (Figure 5.3(c)).

Moreover, we compare the emulation performance when all N creases are used as sensor creases versus when only actuated creases are used as sensors ($N_s = N_a = pN$). The increase in MSE is marginal in the latter case. Therefore, the origami satisfies the previously mentioned nonlinearity and fading memory requirements to be a physical reservoir, and one only needs to use the input creases angles as the reservoir states to simplify the reservoir setup.

5.3.2 Pattern Generation Task

Pattern generation tasks are essential for achieving periodic activities such as robotic locomotion gait generation and manipulator control where persistent memory is required. That is, by embedding these patterns (or limit cycles) in the origami reservoir, one can generate periodic trajectories in the closed-loop. We again use a 9×9 Miura-ori reservoir and randomly select 30% of its creases as the feedback creases (Figure 5.4(a)). This task does not require input creases. These feedback creases are divided into two groups for the two components of 2D trajectories. We run the training phase for 100 seconds for each pattern, discard the initial 15 seconds of data as the washout step and use the next 51 seconds' data to calculate the optimum output readout weights.

Generating nonlinear Limit cycles: In the following results, the origami reservoir demonstrates its computation capability via generating quadratic limit cycle, Van der Pol oscillator, and the Lissajous curve in closed-loop. The quadratic limit cycle is defined by two differential equations:

$$\begin{aligned} \dot{x}_1 &= x_1 + x_2 - \epsilon(t) x_1 \left(x_1^2 + x_2^2\right), \\ \dot{x}_2 &= -2x_1 + x_2 - x_2 \left(x_1^2 + x_2^2\right) \end{aligned} \tag{5.18}$$

where the parameter $\epsilon(t)$ determines the shape of the limit cycle ($\epsilon(t) = 1$ in this case). The Van

der Pol oscillator is defined by:

$$\begin{aligned} \dot{x}_1 &= x_2, \\ \dot{x}_2 &= -x_1 + \left(1 - x_1^2\right) x_2 \end{aligned} \tag{5.19}$$

The Lissajous curve is a graph of two sinusoidal signals parameterized by their frequency ratio ($f_1/f_2 = 0.5$) and phase difference ($\delta = \pi/2$):

$$\begin{aligned} x_1 &= \sin\left(f_1 t + \delta\right) \\ x_2 &= \sin\left(f_2 t\right) \end{aligned} \tag{5.20}$$

The Figure 5.4(b), Figure 5.4(c), and Figure 5.4(d) show the results of Quadratic limit cycle, Van der Pol oscillator and Lissajous curve generation task, respectively. The origami reservoir can generate all three periodic trajectories just by changing the output readout weights. The MSE for Quadratic limit cycle, Van der Pol oscillator, and Lissajous curves, calculated using the data for first 10 seconds' closed-loop run (M = 10000), are 3.28×10^{-7}, 2.03×10^{-5}, and 5.5×10^{-4}, respectively. As expected, MSE increases as the complexity of the curve increases.

Stability and robustness of the pattern generation: After finding the readout weights, we test the stability of these three limit cycles by starting the origami reservoir from total rest in the close-loop and running it for more than 1000 seconds. The limit cycle is stable if and only it can recover the pattern from zero initial conditions and stays on target for at least 1000 seconds of simulation [50, 108]. The results in Figure 5.4 indicate that the torsional moments generated from the feedback signals on the feedback creases are sufficient to recover and maintain the three limit cycles from total rest. Small phase differences occur between generated trajectories and the targets because the reservoir takes a slightly different path than the target, and the Lissajous curve takes more than 15 seconds to recover fully. Nonetheless, the origami reservoir successfully passes this test.

To further analyze the robustness of reservoir-generated limit cycles, we simulate actuator and sensor failures. As the origami reservoir generates the Van der Pol limit cycles in these tests, all feedback and sensor creases stop working (aka. their signals set to zero) for 10 seconds. We conduct these tests when all creases are used as sensor creases ($N_s = N$) and when only feedback creases are sensor creases ($N_s = N_a = 0.3N$). The simulation results in Figure 5.4(e) show that, although the

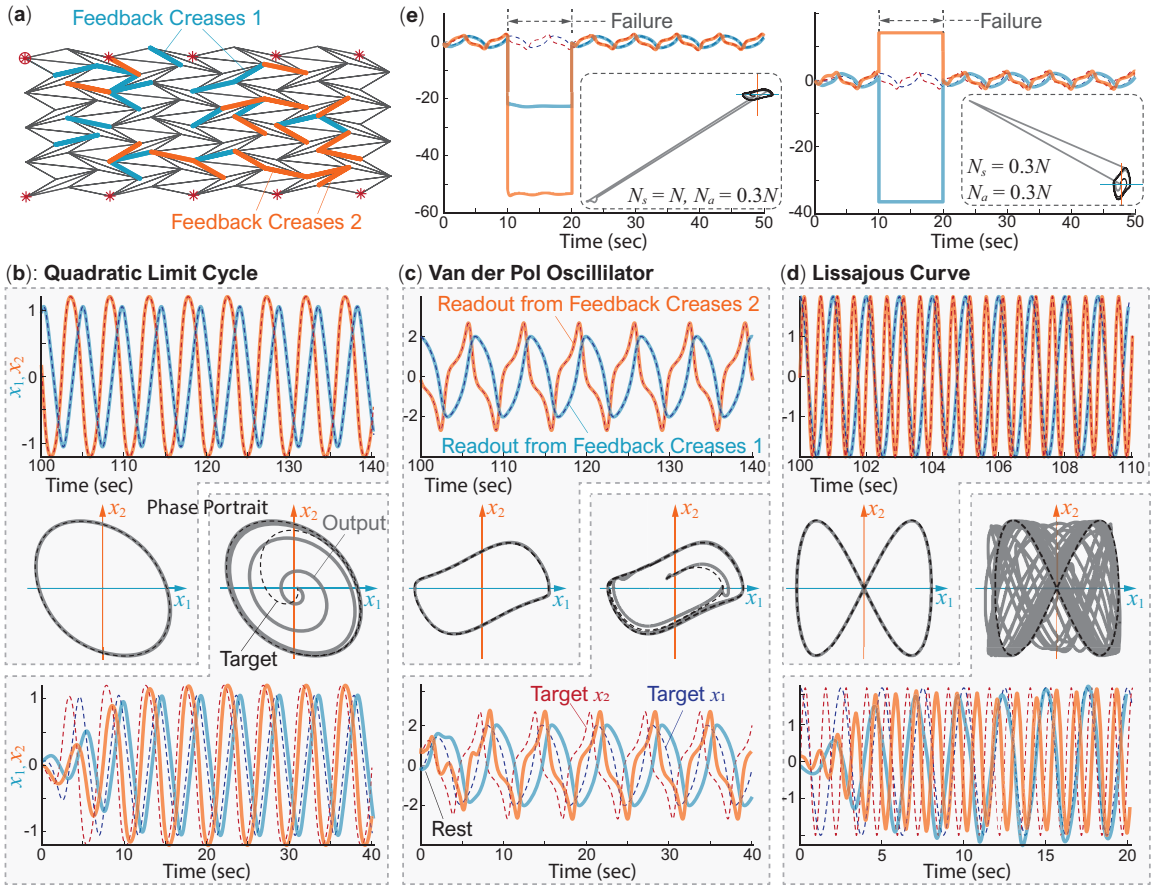

Figure 5.4: Stable pattern generation under closed-loop using the Miura-ori reservoir. (a) This task's origami reservoir includes two groups of feedback creases required to generate 2D limit cycles. (b-d) The closed-loop trajectories of quadratic limit cycle, Van der Pol oscillator, and the Lissajous curve, respectively. In these plots, the first row of time responses shows the closed-loop output after 100s of training. The third row of time responses shows how the trained reservoir can recover the targeted limit cycles from an initial resting condition. The corresponding phase portraits are as shown in the second row. Here, the dashed curves are targeted trajectories, and the solid curves are the reservoir's outputs. (e) Van der Pol limit cycle recovery after the temporary failure of sensor and actuator creases. The two simulations are the same except for the number of sensor creases ($N_s = N$ for the first test, $N_s = 0.3N$ for the second). The insert figures show the corresponding phase-portraits.

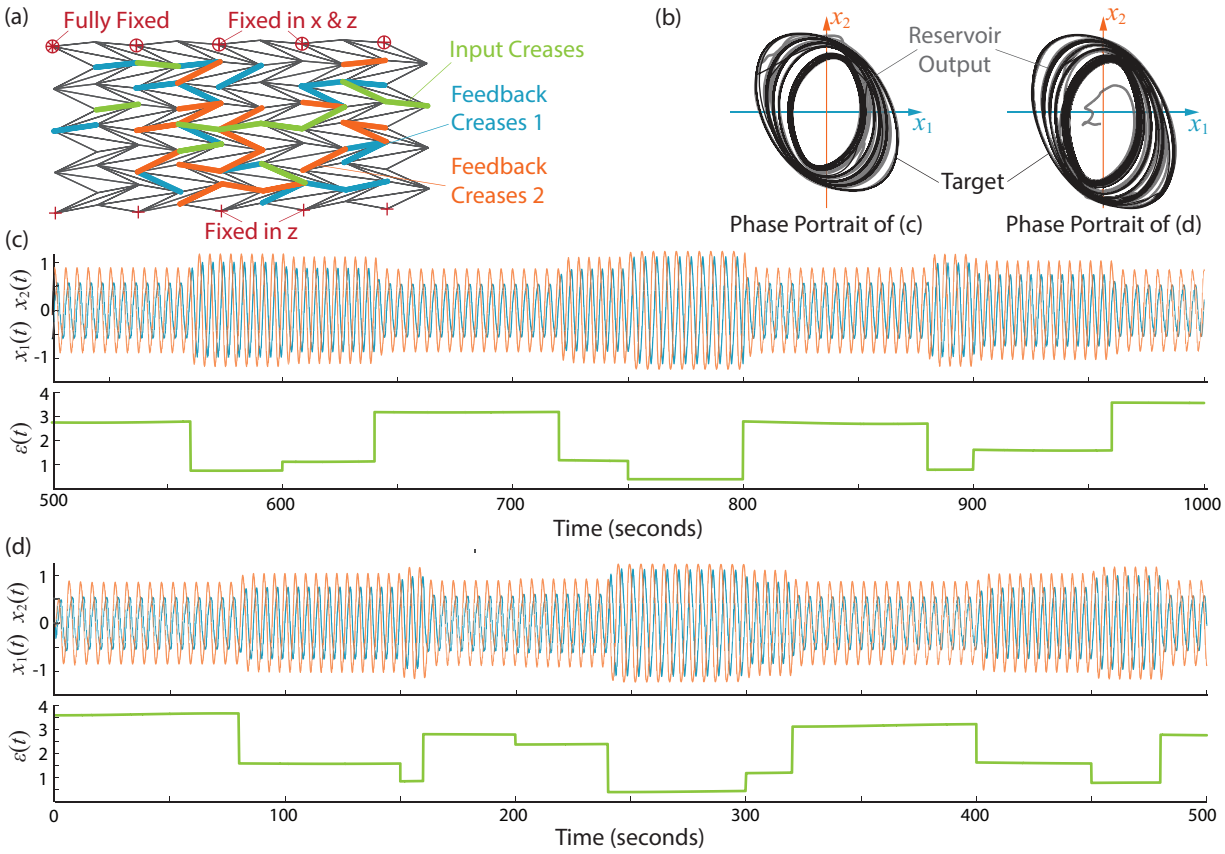

Figure 5.5: Results of modulation task under closed-loop using The Miura-ori reservoir. (a) This task's origami reservoir includes two groups of feedback creases and input creases. (b) The phase-portraits, where the targeted trajectories are overlaid on top of the reservoir output. (c) Quadratic limit cycle trajectories under closed-loop and the corresponding input signal $\epsilon(t)$. The results are obtained after 500 seconds of training. (d) Closed-loop trajectory recovery from the initial resting conditions.

reservoir diverges to a trajectory far away from the target during the actuator and sensor failure, it can immediately recover the Van der Pol limit cycles after the end of these failures.

5.3.3 Output Modulation Task

Output modulation capability allows the reservoir to adjust its output according to a randomly varying input signal without changing the readout weights. This ability is also essential for soft robotic control applications because it allows the robot to switch behaviors according to external stimuli or environmental changes. In this task, we randomly select input creases, which

account for 15% of the total creases, in addition to the feedback creases (Figure 5.5(a)). Moreover, all creases are used as sensor creases ($N_s = N$). The simulation results in Figure 5.5(b) shows the generated quadratic limit cycles with modulated input (Equation (5.18)). We are also able to recover the closed-loop trajectory from the initial resting conditions, as shown in Figure 5.5(c). The phase portraits for the two tasks are depicted in Figure 5.5(d). The origami reservoir can react to the input and modulate the magnitude of the quadratic limit cycles. The MSE is 3.8×10^{-4}, which is remarkably small, considering this task's complexity.

5.3.4 Output Modulation plus Emulation Task

We further show that it is also possible to combine the tasks described earlier to enrich the multi-tasking capabilities of origami reservoir. To distinguish this task from the multi-tasking capability demonstrated in emulation task, we call this 'hybrid' task.We can tap additional readouts from the reservoir to sense the unknown external input that is being applied to the reservoir. These inputs can be present due to environmental interactions, such as ground condition, or external forces acting on the reservoir that are difficult to measure using traditional sensors. Additionally, the reservoir can be trained to respond to the external input via change/ modulation in the output. For example, we can combine quadratic limit cycle output modulation with emulation task as follows:

Pattern generation and modulation: We train the reservoir to generate a quadratic limit cycle described in equation 5.18 autonomously and respond to changing external input ($\epsilon(t)$) by changing the generated limit cycle. This process takes place under closed loop as shown in section 5.3.3, i.e. we feed the current output back to the reservoir feedback creases.

Emulation: We can sense the unknown input applied at input creases through an additional readout. This process takes place under open loop as shown in section 5.3.1 , i.e. we only train the readout.

5.4 Emulation Task Experiment

We demonstrate the origami reservoir's computational power through a simple experiment of the emulation task described earlier. The experimental setup consists of an APS Dynamics vibration exciter (shaker), power amplifier, National Instruments DAQ, a laptop to generate input command, and a camera to capture the vertical vertices displacements. Detailed origami reservoir

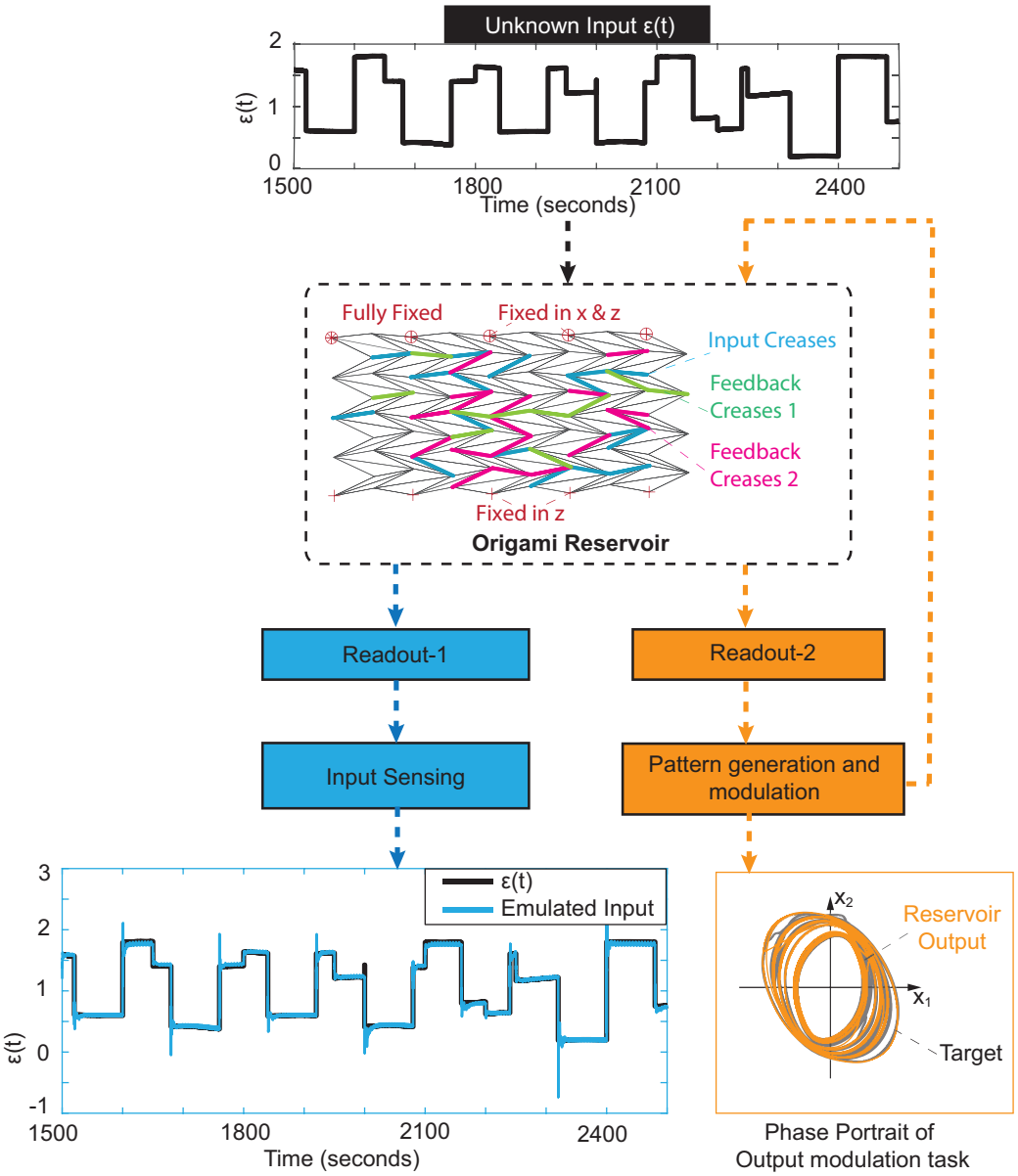

Figure 5.6: Output modulation plus emulation task with origami reservoir. (a) The setup/flow for 'hybrid' tasks with origami reservoir. (b) Emulated input signal for quadratic limit cycle output modulation task shown in figure 5.5.

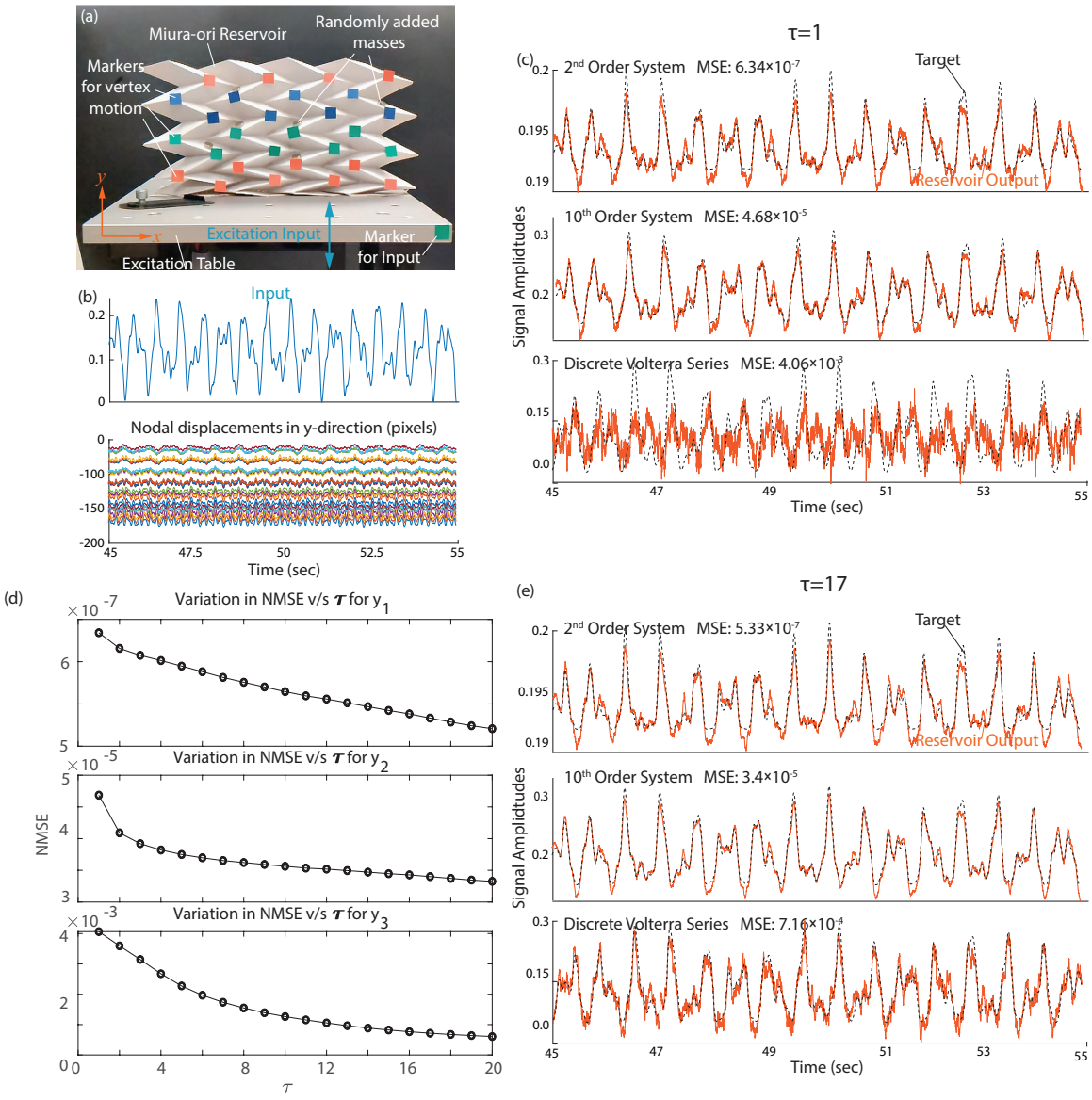

Figure 5.7: Proof-of-concept experiment showing the emulation task with the origami reservoir. (a) The experimental setup depicting the excitation table and paper-based Miura-ori reservoir with sensor node markers attached. The base of the reservoir is fixed to the base plate of the exciter. Notice that additional mass are added at random locations to break the origami's symmetry. (b) The base-plate displacement/input signal (u(t)) and recorded nodal displacements in y-direction. (c) Examples of trajectories generated in the emulation task including (from top to bottom) input signal u(t), 2^{nd}-order, 10^{th}-order system, and Volterra series. Dashed curves are the targeted trajectories, and solid curves are the result of the reservoir. (d) The effect of multiplexing parameter τ on reservoir performance. (e) Examples of trajectories generated in the emulation task for $\tau = 17$ including (from top to bottom) input signal u(t), 2^{nd}-order, 10^{th}-order system, and Volterra series. Dashed curves are the targeted trajectories, and solid curves are the result of the reservoir.

Table 5.3: Design of the origami reservoir in the emulation task expriment

Reservoir size and material properties	
Parameter	Value
Size	9×9
Mass of the reservoir	20 g
Geometric design of Miura-ori	
Parameter	Value
a	31 mm
b	39.2 mm
γ	50.73°
θ	60°
Actuator and sensor creases	
Parameter	Value
No. of total nodes (N)	81
No. of sensor nodes (N_s)	27

design parameters are in Table 5.3. The reservoir is placed vertically on the vibration exciter and fixed rigidly to its base plate, as shown in Figure 5.7(a). The external input ($u(t)$) is provided to the base of origami. To experimentally demonstrate the origami reservoir's computing power without unnecessary hardware complexities, we use the vertical (y-direction) displacement of origami nodes as the reservoir state vector. We capture the deformation of origami through a slow-motion video camera at 480 fps with 720p resolution. The nodal displacement in the y-direction is measured through post-processing of the video data in MATLAB. The training procedures for the emulation task remain the same. The results in Figure 5.7 clearly show that the physical origami reservoir can indeed emulate the three non-linear systems described in Table 5.2.

It is worth emphasizing that using a single base excitation as external input and the vertical nodal displacements as the reservoir states only taps into a part of the origami's high dimensional dynamics. Therefore, the mean square error in the experiment is more significant than those with embedded sensors and actuators (Figure 5.3). Distributing sensors and actuators on the origami creases will provide better control over the shape of origami, as shown in the example of soft robotic crawling. Nevertheless, this experiment serves as physical proof that the Miura-ori based reservoir can perform complex computational tasks.

5.4.1 Improving reservoir performance through Multiplexing

We can further improve the results obtained from the experiment via a technique called 'multiplexing'. This step occurs in the post-processing stage. In multiplexing we increase the number of total nodes available for computation via introduction of virtual nodes. These virtual nodes are composed of state information from τ previous time-steps and it is called the 'multiplexing parameter'. If during the experiment we receive state information from N_s nodes, then using multiplexing we increase the total number of nodes available for computation to $(N_s\tau + 1)$ with one nodes added as a bias node for optimum weight calculation. As a result, we introduce different timescale between reservoir dynamics and the input to output computation, which results in higher computation power [109].

To study the effect of multiplexing parameter τ we change it from $\tau_{min} = 0$ to $\tau_{max} = 20$, effectively introducing delayed state information from up to 20 previous time-steps. Figure 5.7(d) shows introduction of τ indeed improves the emulation task performance as the NMSE for all 3 tasks goes on decreasing monotonically as τ increases. Comparing the plots in Figure 5.7(c) and (e)we see that the performance for emulating 10^{th}-order system, and Discrete Volterra series can be dramatically improved by adding time-step delay.

5.5 Correlating Physical Design and Computing Performance

In this section, we use the mean squared error (MSE) as the metric to examine the connections between the origami reservoir's design and computing performance. In particular, This analysis aims to investigate the sensitivity of MSE to different parameter changes and identify the optimal origami designs. To this end, in-depth parametric analyses are conducted to examine the effect of (1) reservoir size and material properties, (2) crease pattern geometry, and (3) feedback and sensor crease distribution. We use both Van der Pol and quadratic limit cycle generation tasks to ensure the broad applicability of parametric study results.

5.5.1 Reservoir Size, Material Properties, and Vertices Perturbation

Figure 5.8(a) and Figure 5.8(b) show the results of parametric analyses for Quadratic limit cycle and Van der Pol oscillator limit cycle generation, respectively. We observe that feedback crease distribution affects reservoir computing performance quite significantly. In particular, poorly

Table 5.4: Variables for reservoir size and material properties parametric study

Parameter	Base value	Distribution
Nodal mass (g)	7	[1,50]
Geometric imperfections	Standard Miura-ori	$\sigma = \chi \exp(\frac{-\|\|(N_i - N_j)\|\|}{l})$ $\mu = 0$, $\chi = 0.4a$, $l = 4a$
Truss torsional stiffness N/(m-rad)	$k_{b,a} = 1$, $k_b = 0.2525$	$k_{b,a} = 1$, $k_b \in [0.005, 0.5]$

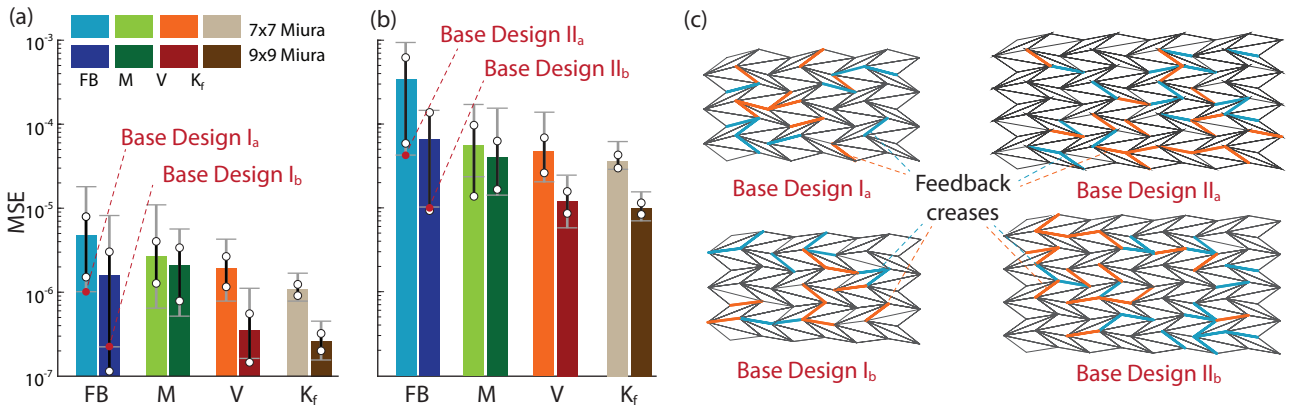

Figure 5.8: Effect of reservoir size and material properties on the reservoir computing performance. (a) The distribution of MSE from the Quadratic limit cycle simulations using random feedback crease distributions and different design parameter distributions. Here "FB" stands for feedback crease distribution, "M" stands for nodal mass distribution, "V" stands for origami vertices geometry perturbation, and "K_f" stands for crease torsional stiffness distribution. It is worth emphasizing that the "FB" results come from one parametric study of 72 unique designs, and the "M," "V," and "K_f" are results of the subsequent simulation. The bar charts depict the average value, standard deviation (circles), and extreme values (horizontal bars) of MSE. (b) A similar result from the Van der Pol limit cycle generation task. (c) The feedback crease distributions of the four different baseline designs used in this parametric study.

distributed feedback creases might result in failed pattern generating tasks. Therefore, we first conduct numerical simulations by randomly changing the feedback crease distributions (72 unique designs in total) and identifying the best performing one (with the least MSE). We refer to this best performing feedback crease distribution as the *base design* (Figure 5.8(c)) for the following parametric studies. Then, we conduct another parametric study regarding the nodal mass, crease stiffness, and vertices perturbation. We vary these three parameters, one at a time, for 72 randomly selected designs (six batches of jobs in parallel on a computer with 12 cores). The baseline values and range of the parameters are in Table 5.4.

The origami reservoir performance turns out to be highly sensitive to the nodal mass variation. As opposed to the uniform nodal mass in base design, a randomly distributed nodal mass can significantly increase or decrease the MSE for both pattern generation tasks. However, randomly distributing mass in an origami sheet is quite challenging in practical applications. So the use of varying mass distribution should be judicially done based on the particular application at hand. On the other hand, the origami performance is much less sensitive to the crease torsional stiffness. By randomly changing the stiffness, one can achieve performance at par with the base design.

Moreover, we investigate the effects of random geometric imperfection in the base designs of origami reservoir. To this end, we adopt the formulation introduced by Liu et al. [87], which introduce small perturbations to the nodal positions in folded origami. Such imperfections are inevitable in practice due to various manufacturing defects. It is found that these small imperfections do not worsen the MSE significantly and in fact could reduce the MSE by a moderate degree.

It is also worth noting that the larger 9×9 Miura origami reservoir performs better than the smaller one because larger origami contains more folding angles to constitute the reservoir state matrix. Therefore, the high-dimensionality of a reservoir is desirable to produce smaller MSE.

5.5.2 Origami Design

A unique advantage of origami based structures and materials is their considerable freedom to tailor the geometric design. To this end, we start from the Base Design IIa and IIb of 9×9 Miura-ori reservoir, vary its crease length ratio (a/b) and internal sector angle (γ), and then run the quadratic limit cycle and Van der Pol limit cycle tasks with 100 crease length and sector angle combinations at three folding angles. The results of the parametric analysis for $\theta = 50°$, $\theta = 60°$,and $\theta = 70°$ are shown in Figure 5.9(a), Figure 5.9(b), and Figure 5.9(c), respectively. We observe that,

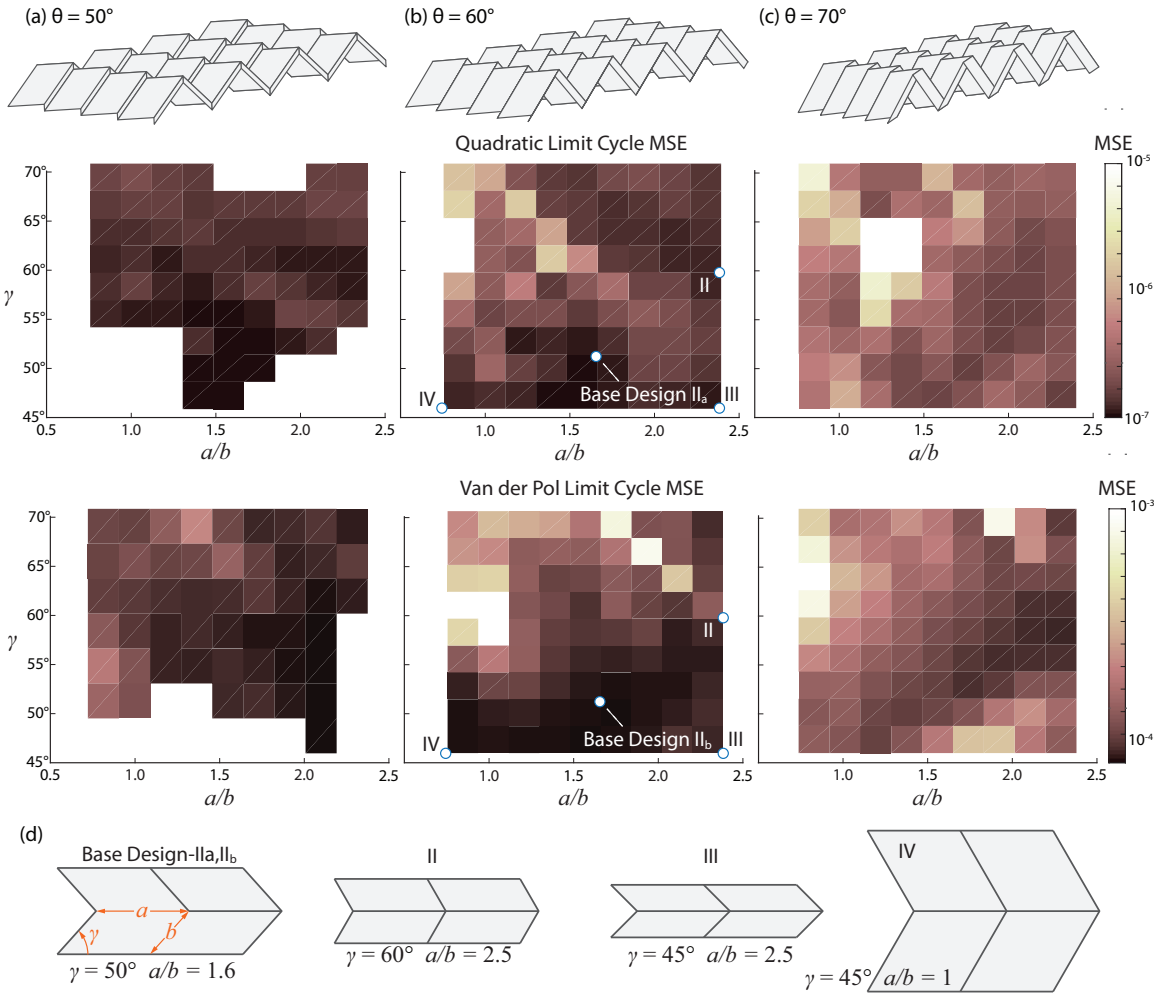

Figure 5.9: Effect of Miura-ori geometric design on the reservoir performance. (a-c) The Miura-ori geometry and the corresponding landscape of MSE distribution when $\theta = 50°$, $60°$, and $70°$, respectively for Quadratic and Van der Pol limit cycle generation. The lighter and darker regions correspond to larger and smaller errors, respectively, while the white regions depict origami designs that failed the computing task. (d) The unit cell geometry of four representative designs with the same crease a length but different sector angles γ and crease length ratios a/b.

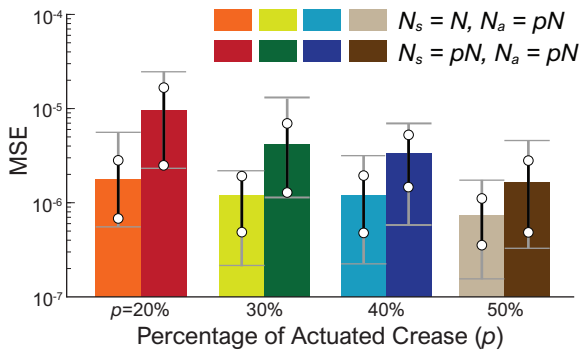

Figure 5.10: Effect of varying the number of actuator and sensor creases.

at lower folding angles (flatter origami), the origami reservoir has a higher possibility to fail the pattern generation tasks. The computing performance improves significantly with a reduced MSE as the origami folds more (or as θ increases). This trend is probably because highly folded origami offers an increased range of folding motion.

Moreover, there are two design sets with the lowest MSE: $a/b \approx 1.5$, $\gamma \approx 45°$, and $a/b \approx 2.5$, $\gamma \approx 60°$. Figure 5.9(d) shows unit cell geometry of four representative designs in this range. Generally speaking, a moderate to high crease-length ratio and small sector angles can create "skewed" origami patterns that appear to give better computing performance across all values folding angles. The best designs here have MSEs at the order of 10^{-7}, which is of the same magnitude as we found previously by tailoring the nodal mass and crease stiffness.

5.5.3 Actuator and Sensors Distribution

Finally, it is important, for practical applications, to find the minimum amount of input/feedback and sensor creases required for achieving acceptable computing performance. To this end, we start with the 9×9 Miura-ori reservoir and conduct two tests. In the first test, we vary the percentage of feedback creases ($N_a = 0.2N, 0.3N, 0.4N, 0.5N$, each with 24 randomly generated crease distributions) while using all crease dihedral angles to constitute the reservoir state matrix (i.e., $N_s = N$). In the second test, we use the same feedback crease design and only use these feedback creases' dihedral angles to formulate the reservoir state matrix (i.e., $N_s = N_a$).

We find that if only 20% of crease are used for feedback, the origami reservoir might fail the quadratic limit cycle task. On the other hand, the MSE reduces only marginally as we increase the

percentage of feedback creases beyond 30% (Figure 5.10). Therefore, we can conclude that using only $30\% - 40\%$ of total creases as the feedback and sensors crease will provide us an adequate computing performance. This result is significant because it shows that, even though a large size (high-dimensionality) of the reservoir is essential for computing performance, one does not need to measure (readout) every reservoir state. In this way, the practical implementation of the origami reservoir can be significantly simplified.

In conclusion, the parametric analyses lay out the strategy to optimize the origami reservoir performance by tailoring the underlying physical and computational design. A larger origami with a higher-dimension can ensure low computational error, but one only needs to use 30% 40% of its creases as the feedback and sensor creases to tap into the origami's computing capacity. Meanwhile, the distribution of these feedback and sensor creases must be carefully chosen with extensive simulations. To further improve computing performance, one can tailor the origami's mass distribution, crease stiffness, and geometric design. Among these options, optimizing the folding geometry should be the most effective because it is easy to implement in practical applications.

5.6 Application to soft robotics

This section demonstrates the application of origami reservoir computing to design soft robots with embedded sensor-actuator network. We apply PRC to design a peristaltic crawling soft robot and a quadruped soft robot. Finally, we apply PRC to design origami based manipulator to generate desired end-effector trajectory.

5.6.1 Soft robotic crawling

This section demonstrates the application of origami reservoir computing to generate an earthworm-inspired peristaltic crawling gait in a robotic system. The earthworm uses peristalsis to navigate uneven terrain, burrow through soil, and move in confined spaces. The lack of complex external appendages (aka., legs or wings) makes earthworm-inspired robots ideal for field exploration, disaster relief, or tunnel drilling [8, 65, 26]. The body of an earthworm consists of segments (metamerism) grouped into multiple "driving modules" [127, 5]. Each driving module includes contracting, anchoring, and extending segments actuated by antagonistic muscles (Figure 5.11(a)). During peristaltic locomotion, these segments alternately contract, anchor (to the environment with

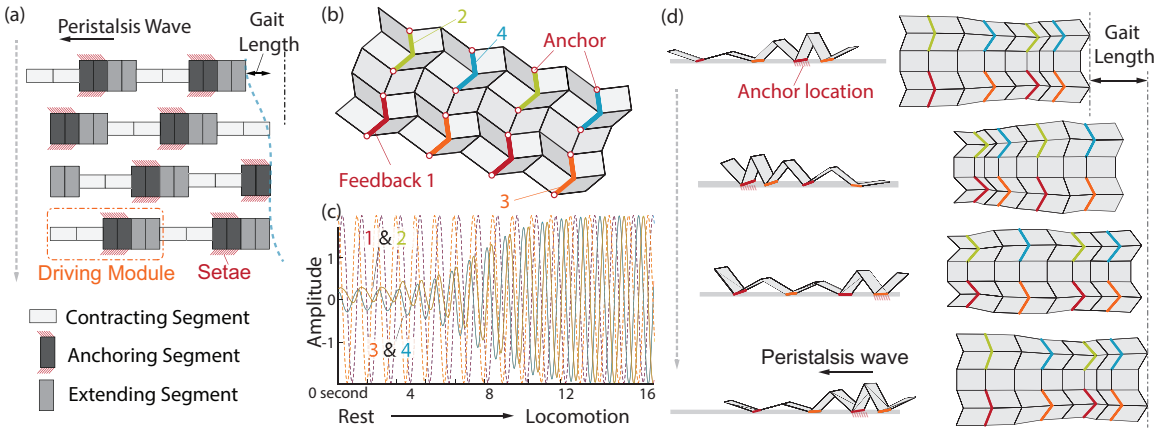

Figure 5.11: Reservoir computing powered crawling origami robot. (a) The kinematics of a peristaltic locomotion cycle in an earthworm. For clarity, the earthworm body is simplified and consists of six identical segments organized into two driving modules. The earthworm body moves forward while the peristaltic wave of anchoring segments (or driving modules) propagates backward. (b) The design of an earthworm inspired origami crawling robot that features two stripes of Miura-ori connected by a zig-zag shaped "ridge." This robot has four groups of feedback creases. (c) The closed-loop trajectory generated by the feedback creases after training (shown in solid lines) superimposed with the targetted trajectory (shown in dashed lines). (d) Peristaltic locomotion cycle in the origami robot as a result of the generated trajectory.

the help of *setae*), and extend to create a propagating peristalsis wave, thus moving the body forward.

We design an earthworm-inspired origami robot consisting of two 3×9 Miura-ori reservoir connected via a stiff central bridge (5.11(b)). The left and right half of the robots are symmetric in design, and the central bridge design allows differential motion between the two halves to facilitate turning in response to the external input. In each origami reservoir, we embed two groups of feedback creases (Figure 5.11(b)) with feedback weights assigned such that their values for the front and back-half are equal but opposite to each other. This arrangement reduces the number of reference outputs needed to generate a crawling gait. To create a peristalsis locomotion gait, we train the origami reservoirs to generate multiple harmonic signals with a phase difference of $\pi/2$ among them (aka. a pattern generation task). We train the robot for 100 seconds and discard the first 15 seconds of data as the washout step.

The ground reactions are modeled by setting the vertical velocity of masses (i.e. point contact with node) touching the ground to zero and the horizontal friction coefficient to be infinite.

Inspired from the setae in earthworm, we assume presence of passive foldable anchors that lose/grip anchoring to the ground periodically. We apply ideal anchors to the bottom origami creases that are in contact with the surface below. These anchors are assumed to be kinematically attached to the ground when the associated origami crease folds and relaxed as the crease unfolds (or flattens). Such anchor design is feasible by leveraging the origami facets' folding motion, as shown in the author's previous study [5]. This is a hard constraint that can impact the nature and the performance of locomotion. However, it simplifies the study of the body influence by assuming perfect friction conditions in every simulation.

Figure 5.11(c) depicts the closed-loop response and the limit cycle recovery from total rest (MSE is 3.9×10^{-04}). Figure 5.11(d) (and Supplemental Video A) illustrates the robotic locomotion generated by reservoir computing. As the origami reservoir generates the multiple harmonic signals with a phase difference, its folding motion naturally "synchronizes" to these signals, generating a peristaltic wave of folding and unfolding. As a result, the robot crawls forward like an earthworm, without using any traditional controllers.

5.6.2 Quadruped Robot

In this section we design a quadruped robot embedded with PRC. We combine four mirror symmetric 4×7 arc-Miura patterns to create a four-legged soft origami robot structure (Figure 5.12). We add a 11×3 Miura-ori bridge between left and right side to provide additional freedom for movement between two sides, and a potential location for electronic hardware placement. The quadruped robot is designed such that each leg receives different feedback signal and can move independently of another. We assume ideal ground contact conditions, similar to the ones used for crawling robot studies. The resulting robot can exhibit different quadruped gaits e.g. walk, trot and bound. We use simplified sinusoidal equations to simulate walking, trotting and bounding gait as shown below. We have fixed the actuation frequency f to 1Hz for this study. The robot can successfully generate the gaits under closed loop and rhythmically move its legs corresponding to the required gait (Figure 5.12).

The equations for walking gait are given by,

$$\begin{aligned} x_1 &= \sin(2\pi ft) \\ x_2 &= \sin(2\pi ft + \frac{3\pi}{2}) \\ x_3 &= \sin(2\pi ft + \pi) \\ x_4 &= \sin(2\pi ft + \frac{\pi}{2}) \end{aligned} \tag{5.21}$$

The equations for trotting gait are given by,

$$\begin{aligned} x_1 &= \sin(2\pi ft) \\ x_2 &= \sin(2\pi ft + \pi) \\ x_3 &= \sin(2\pi ft + \pi) \\ x_4 &= \sin(2\pi ft) \end{aligned} \tag{5.22}$$

The equations for bounding gait are given by,

$$\begin{aligned} x_1 &= \sin(2\pi ft) \\ x_2 &= \sin(2\pi ft + \pi) \\ x_3 &= \sin(2\pi ft) \\ x_4 &= \sin(2\pi ft + \pi) \end{aligned} \tag{5.23}$$

This example shows the versatility of origami mechanisms as an excellent platform for designing soft robotic structures and skeletons. Such origami robot when embedded with the PRC framework can autonomously generate the desired locomotion gait trajectories, and physically moving the appendages as suitable for the movement.

5.7 Summary and Conclusion

We demonstrate the physical reservoir computing capability of origami via extensive benchmark simulations and parametric studies. First, we develop a simulation environment to study the nonlinear origami dynamics and detail the origami reservoir setup. This reservoir successfully achieves many computing tasks such as emulation, pattern generation, and modulation, all of which

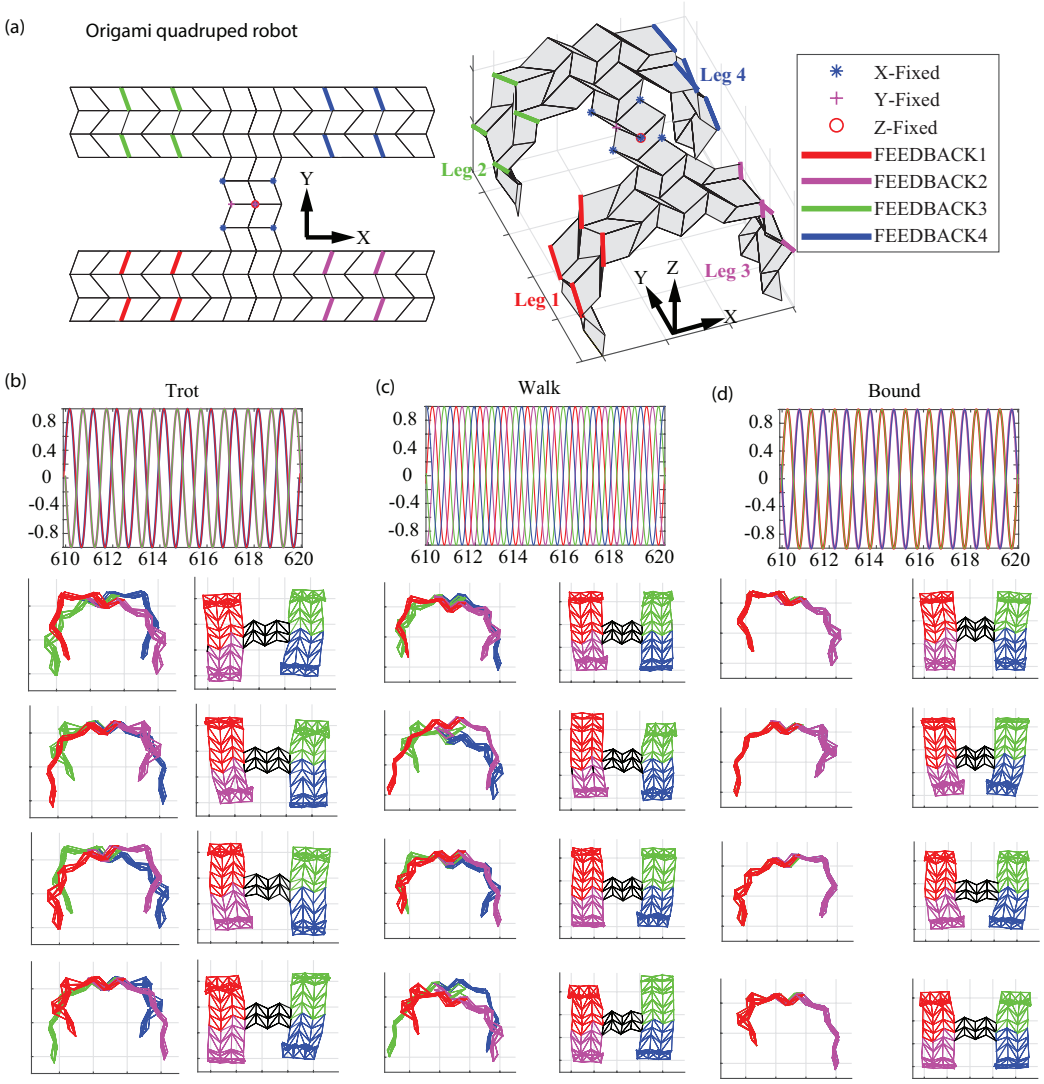

Figure 5.12: PRC embedded Origami quadruped robot: (a) The crease pattern (left) and folded robot (right) depicting feedback creases and boundary conditions applied. (b) Trotting gait: (top) Sample trajectories generated for over 600s when system was switched to closed loop after 120s of training. (below) The gait generation in the robot corresponding to the actuation trajectories. Leg-1 and 4 are moving in phase and Leg-2 and 3 are moving in phase. (c) Walking gait: (top) Sample trajectories generated for over 600s when system was switched to closed loop after 120s of training. (below) The gait generation in the robot corresponding to the actuation trajectories. All the legs are moving out of phase from each other by 90 degrees. (d) Bounding gait: (top) Sample trajectories generated for over 600s when system was switched to closed loop after 120s of training. (below) The gait generation in the robot corresponding to the actuation trajectories. Leg-1 and 2 are moving in phase and Leg-3 and 4 are moving in phase.

are relevant to robotic applications. We also conduct comprehensive parametric analysis to uncover the linkage between origami reservoir design and its computing performance. This new knowledge base offers us a guideline to optimize computing performance. To the authors' best knowledge, this is the first study to rigorously examine the performance of physical reservoir computer from the lens of the physical design. Finally, we demonstrate how to embed reservoir computing into an origami robot for control without traditional controllers through the example of peristaltic crawling and quadruped robot.

We list four requirements for a mechanical system to be a reservoir in the introduction, and origami satisfies all these requirements. The tessellated origami structures are inherently high-dimensional. For example, a 7×7 Miura-ori with 49 nodes contains $N = 60$ crease dihedral angles, all or a small portion of them can serve as the reservoir states. The nonlinearity of origami partly originates from the nonlinear kinematic relationships between these crease angles and external geometry. Also, since origami patterns are highly structured (ordered), small perturbations in the material properties, imperfections of crease geometry, and the introduction of local actuation are sufficient to destroy the regularity and create disorder. These properties make origami highly nonlinear dynamic reservoirs. The origami reservoir's performance in the emulation task proves that it can act as a nonlinear filter and satisfies fading memory property. Nonlinear patterns can be embedded into the origami reservoir, and the resulting pattern generation is robust against external disturbances and recoverable under different initial conditions, proving separation property. Finally, adding the feedback can create persistent memory, which is conducive to learning new tasks.

For future robots to work autonomously in unstructured and dynamic environments, the robot body and brain have to work together by continuously exchanging information about the current condition, processing this information, and taking appropriate actions. The physical reservoir computing embodied robots shown in this study presents a step toward this vision. The reservoir embedded in the robot body directly gathers information from the distributed sensor-actuator network to perform low-level control tasks like locomotion generation. The resulting soft robot can generate the global target behavior autonomously without controlling every element individually. Moreover, the generated trajectories could be robust against external disturbances and modulated according to changing working conditions.

A challenge in implementing physical reservoir computing is the many sensors and actuators required, even though these sensors and actuators can be simple individually. Our results contribute

in this regard by showing that only a small portion of origami creases are required to be equipped with sensors and actuators to tap into the reservoir computing power. The essential advantage of an origami reservoir compared to a typical neural reservoir is that it can serve as the physical body of the robot and at the same time perform nonlinear computation for control tasks. Meanwhile, it will be imprudent to directly compare origami reservoirs' computing performance with other physical reservoirs. Only a handful of studies examined the use of soft physical bodies for reservoir computing (e.g., soft silicone arms, tensegrity structures, compliant spine robot). And these systems are often constrained by other non-computing-related requirements. Thus, a comparative study between different physical reservoirs is a worthy topic for future research.

In summary, origami reservoir computing provides an attractive pathway for facilitating synergistic collaboration between the soft robot's body and the brain. The reservoir computing, coupled with unique mechanical properties that origami can offer — multi-stability [84, 66, 159], nonlinear stiffness [84, 141, 144, 66], and negative Poisson's ratio [84, 141, 66] — opens up new avenues to the next generation of soft robots with embedded mechanical intelligence.

Chapter 6

Major Contributions and Future Work

We repeat the research objective stated in Chapter 1. In this section we summarize the major contributions of this work and expound upon the future directions for extending this research. As a researcher I have understood that when we find an answer to some question; more often than not it leads to more questions and introduces us to new avenues and directions which are yet unexplored. The summary given here will hopefully do the same and lead us to exploring more exciting avenues in the field of soft robotic control and design.

Research Objective

Developing design and analysis framework for hybrid mechanical-digital control of soft robots: from mechanics-based motion sequencing to physical reservoir computing

6.1 Multi-stability and motion sequencing

Harness multi-stability of origami and compliant mechanisms for mechanics-based motion sequencing in soft robots

In our research, the motion-sequencing is hard-coded or embedded in the origami robot skeleton. We wanted to emulate earthworm inspired peristaltic locomotion gait in an origami robot. We started with two Kresling origami mechanisms with different bistability strengths and serially combined them to create a driving module for robot. The multi-stability analysis of such a driving module produces a deterministic and periodic cycle during extension and contraction phase. We combine part of the extension phase, part of contraction phase and two snap-through actions generated during the displacement controlled actuation of the driving module to form an actuation cycle that is repeated for periodic actuation of the robot. As a result, we only need a single linear actuator to control the total length of the robot. Additional actuators and digital controllers are not required to generate a robust locomotion gait, as the individual segment deformations are controlled by the embedded multi-stability. Thus, this approach reduces control requirement drastically. Additionally, the locomotion speed can be increased or decreased by increasing the operating frequency of actuator. This is an instance of 'morphological control' wherein the robotic skeleton itself takes part in performing low-level control tasks.

We also experimentally demonstrated the motion-sequencing in the multi-stable Kresling origami robot skeleton. After extensive experiments we proved the existence of periodic actuation cycle in the driving module composed of Kresling origami similar to peristalsis in earthworm. We also successfully fabricated a proof-of-concept prototype of a Kresling origami robot that could uniformly generate peristaltic locomotion actuation cycle. The Kresling robot is actuated via a linear actuator composed of a stepper motor-compression spring assembly. The central controller for the robot is arduino based controller that moves the stepper motor back and forth to change the length of the driving module of the robot. This resulting actuation mechanism is much simpler than the complex logic required to individually control the segment lengths in multi-segmented peristaltic gait robot. The future extension of this work can be to design a compliant mechanism that will make the robot truly soft and compliant. This will also enable addition of bending capability to the robot. One more avenue to explore would be to find a way to switch the bistability strength of driving module segments, because that is a pretty straightforward way to reverse the direction of locomotion of the

robot. There are many bistable mechanisms that could be even more suitable for motion-sequencing bases peristaltic crawling robot. It will be interesting to study performance of different mechanisms and exploration of the rich design space available to these mechanisms.

The compliant robot described above is only capable of unidirectional locomotion. To expand the functionality of such robot, we also studied a compliant mechanism based driving module, that is composed of a tri-stable and a bistable mechanism. Our theoretical investigations suggested that it's indeed possible to generate two actuation cycles with this driving module. The complex energy landscape of such multi-stable mechanism leads to four snap-through actions, which when aligned properly make bidirectional locomotion possible. One actuation cycle is for forward-moving locomotion and the other one is for backward-moving locomotion. Again, we only need to change the total length of driving module to generate peristaltic locomotion actuation cycle and to switch from forward-moving to backward-moving locomotion cycle. Our experimental efforts to design a viable driving module for bidirectional locomotion were unfortunately not successful. We started with compliant crank-slider inspired mechanism, which gave really good theoretical results. Our initial attempts at fabricating a 3-D printed compliant mechanism however failed to generate repetitive behavior. We could not fabricate identical mechanisms and the 3-D printed mechanisms failed to have graded bistability strengths for various hinge parameters leading to a failure to generate reproducible bistability strength for different fabricated mechanisms. The further attempt use steel shims as compliant hinge also failed to generate reproducible results required for estimation of stiffness variations in tri-stable and bistable mechanism. This just goes to show that we still have to learn a lot about precision fabrication of compliant mechanisms. Nevertheless, this topic is worthy for future research and spans many interesting areas from material science, compliant mechanism design and 3D printing technology.

The aim of this study was to develop a detailed framework for harnessing multi-stability of origami and compliant mechanisms. Such mechanisms are very useful for motion sequencing applications, where a sudden jump or snap-through to a distant equilibrium path can lead to a completely new configuration. Additionally, such behavior is generally repetitive in compliant mechanisms, as opposed to catastrophic failure that occurs in stiff materials. Hard-coding such periodic and/ or repetitive behavior has led to design of mechanical logic gates, memory units, motion-sequencing for robotic movement, etc. Multi-stability is just one of many attractive properties that the origami and compliant mechanisms possess – for example, negative and variable stiffness, auxeticity, modularity

and multi-transformability, etc. Till now these properties were mainly curious but powerful ideas that were out of reach due to the gap between theoretical analysis and the available state-of-the-art technology. However, in recent years the field of soft robotics has advanced remarkably. Many researchers are developing stretchable, elastic and bendable sensors and actuators, flexible electronics, novel smart materials, 4D printing techniques; that can actually be implemented as the components of compliant robotic skeleton. Availability of such technology will greatly enhance the efforts to harness unique mechanical properties of origami and compliant mechanisms.

In summary, the results from our research suggest that a rich multi-stable mechanism energy landscape with multiple peak and valleys can potentially provide more functionality to the design of soft robot and a possibility to embed multiple behaviors in the same robotic skeleton, with simplest of controllers.

6.2 Physical Reservoir Computing with origami

Develop framework for the design of Physical Reservoir Computing embedded soft origami robots with distributed sensor and actuator network

The physical reservoir computing is a very nascent field of research, with only a handful of publications in the field of soft robotics. PRC takes inspiration from related research in Recurrent Neural networks (RNNs) called Reservoirs with the neural network considered as a black box non-linear dynamical system that includes only training the output readout which is mostly linear and static. The implications of this lead to a powerful result proved by Hauser and group, that a network of non-linear spring-mass-damper dynamic system can be used as a Reservoir. This gave way to what we now know as physical reservoir computing where a highly non-linear dynamic reservoir is used to perform complex computations. Since soft and compliant mechanisms satisfy this criteria it naturally led to connections with morphological computation with applications in autonomous soft robotic control.

This research intends to provide fundamental framework to design and analyze next-generation of intelligent, autonomous soft origami robots. We hope that PRC is a major stepping stone towards fully autonomous soft robots with adaptive functionality. The central idea of PRC is to use soft, compliant bodies with non-linear dynamic behavior and distributed actuation-sensor networks as physical reservoirs, so origami is a natural choice to design dynamic reservoirs for soft robotic appli-

cations. Origami also offers versatile approach for creating complex 3-D mechanisms with periodic tessellations and multiple curvatures, from flat-foldability to transformative property. Thus complex and sophisticated transformable robots can be designed via using principles of origami.

We showed through extensive numerical simulation that origami reservoir can indeed function as a dynamic reservoir to perform task such as emulation, pattern generation, output modulation, multi-tasking, etc. We also performed multiple parametric studies to uncover the effect of different geometric and material parameters on the reservoir performance. The studies suggest that the feedback crease distribution has the highest impact on the reservoir computing power. Still, more work needs to be done to find optimum design for given computational tasks that reservoir needs to perform. The big question still remains if there exists a truly optimum configuration that gives best performance for all the tasks imaginable. We still have to figure out what the overall limitations of the origami reservoir are in terms of computing power and the kind of computations that it can perform. This work being the first one in this area.

The next major area of research is the practical implementation of reservoir computing framework to origami reservoir. We experimentally demonstrated the reservoir computing power of Miura-ori based origami reservoir for emulation tasks. This task is performed in the open loop so we can observe the reservoir state remotely. So it does not require embedded sensors and actuators. But that is not the case for closed loop task with origami reservoir. Until we find ways to design origami reservoirs with low-profile, embedded sensing and actuation technology their widespread use for soft robotics will remain out of reach.

In this research we also designed multiple robots with earthworm inspired crawling, and quadruped robots with multiple locomotion gaits. We also showed that it's possible to control end-effector trajectory of a soft manipulator with PRC. We successfully embedded locomotion patterns in the robots that are capable of producing prescribed locomotion gaits theoretically. These locomotion gaits were initially simulated under idealized conditions where the effect of ground reactions and friction conditions was not taken into account. This is an important constraint in practical implementation and real-life deployment. We enhanced the capability of our dynamic simulation with introduction of impulse-based model to simulate ground collisions and friction. There are still many limitations to such simulation. First, the origami mechanism is simulated as a discrete network of extendable, massless trusses joined together at the origami vertices which are point masses. The origami folding is simulated with rotational hinges connecting the facets of the origami. This formu-

lation is powerful enough to simulate effects of external forces and the resulting folding motion, but it has limited capability to simulate ground contact, since we only have point contact with ground instead of surface as in real-life. Thus, improving the model to create realistic simulations that can help in designing the robot virtually is an active research topic.

With these things in mind, the immediate extension of our work would be to understand the intricacies in soft robotic implementation of PRC framework. For example, the possible designs of origami pattern for earthworm-inspired crawling robot and the quadruped robot with optimum actuator-sensor network distribution. Finding ways to add more functionality to the soft robot by implementing gait modulation, gait switching (addition of turning gait for crawling robot and smooth switching of walking, trotting, bounding gait for quadruped robot). Performing parametric analysis to uncover the correlations between the robot performance and robot design parameters (underlying origami design, material parameters, and actuator-sensor distribution) is also an important topic.

In future, the PRC coupled with unique mechanical properties that origami and compliant mechanisms can offer – multi-stability, nonlinear stiffness and negative Poisson's ratio – will open up new avenue of research in designing next generation of soft origami robots with embedded mechanical intelligence. Thus, we can take advantage of versatile mechanical properties of origami to create truly powerful, adaptable, multi-functional soft robots. For example, it might be possible to design a robot with embedded multi-stability and in combination with PRC can be used to modulate the behavior of soft robot, by switching from one stable state to another.

In summary, the scope of research in the highly multi-disciplinary field of soft robotics is no longer limited to just robotics. The application of reservoir computing for soft robot design and control just proves how deeply intertwined and exciting the research in this area is to various branches of engineering and fundamental science. Our research has paved the way for a new type of soft robots with ability to perform online computations using their complex dynamics. The results of our work will hopefully lead towards new instances of *Embodied Intelligence* for truly autonomous robots.

6.3 List of Publications

Following is the full list of publications that directly resulted from the work I undertook during the course of my research.

[1] Architected origami materials: How folding creates sophisticated mechanical properties S Li, H Fang, S Sadeghi, P Bhovad, KW Wang, Advanced materials 31 (5), 1805282 (2019)
https://doi.org/10.1002/adma.201805282

[2] Peristaltic locomotion without digital controllers: Exploiting multi-stability in origami to coordinate robotic motion P Bhovad, J Kaufmann, S Li, Extreme Mechanics Letters 32, 100552 (2019)
https://doi.org/10.1016/j.eml.2019.100552

[3] Actuation performance of fluidic origami cellular structure: a holistic investigation H Sane, P Bhovad, S Li, Smart Materials and Structures 27 (11), 115014 (2018)
https://doi.org/10.1088/1361-665X/aadfac

[4] Using multi-stable origami mechanism for peristaltic gait generation: A case study P Bhovad, S Li, ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, V05BT07A061–V05BT07A061
https://doi.org/10.1115/DETC2018-85932

[5] Locomotion of an origami inspired nonholonomic system V Fedonyuk, P Bhovad, S Li, P Tallapragada, Dynamic Systems and Control Conference 59148, V001T03A005 (2019)
https://doi.org/10.1115/DSCC2019-9016

[6] Physical Reservoir Computing with Origami: A novel framework for autonomous soft robotic locomotion P Bhovad, S Li, Scientific Reports 11, 13002 (2021)
https://doi.org/10.1038/s41598-021-92257-1

Appendices

Appendix A Equivalent stiffness parameter formulation

This section details the formulation used for defining the stiffness parameters used in this study: They are the equivalent crease torsional stiffness k_i^c facet bending stiffness k_i^f and facet stretching stiffness k_i^s where the sub-index i identifies the different creases or facets in the fluidic origami module. The simplified free-stroke analysis Section 2.3 uses only the crease torsional stiffness, while the truss-frame model in Section 2.4 uses all three stiffness parameters. Much of the published work on such stiffness calculation is applicable to uniform thin sheets made of linearly elastic materials, but the fluidic origami modules are fabricated from 3-D printed hyperelastic materials. Moreover, the fabricated modules feature v-grooves along the crease lines to facilitate proper folding. Currently, there is no universal framework to describe mechanical properties of such structures, so we resort to careful approximation techniques to describe these stiffness properties. In the subsections below, we present the stiffness parameter formulation which approximates the folding characteristics of the 3D printed fluidic origami as seen in the experiments and finite element analyses.

A.1 Crease torsional stiffness

The crease torsional stiffness characterizes the elastic folding and unfolding behaviors along the crease lines. To derive this parameter, we approximate the crease as a non-uniform cantilever beam subjected to an external moment M (Figure 1). The magnitude of this moment is chosen such that it creates crease deformations of the same order as seen in finite element simulations. The applied moment is higher in the free-stroke analysis than in the block force analysis; because the crease folding is more significant in the former case. To simplify the analysis, we neglect the constraining effects due to surrounding facets so that the crease stiffness parameter is calculated based on the following equations:

$$k_i^c = \frac{M}{\phi(w_i)} \tag{1}$$

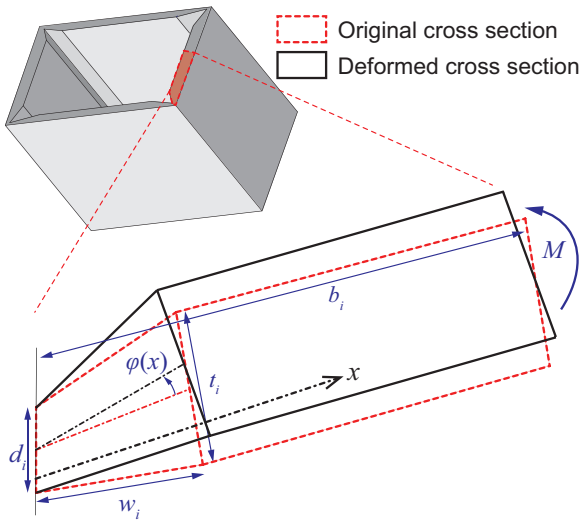

Figure 1: An example of the cross-section area of half of a crease, which is modeled as a non-uniform cantilever beam for crease torsional stiffness characterization.

where the crease folding angle $\phi(w_i)$ is related to the applied moment M and resting crease dihedral angle ϕ^0 is defined at the stress-free state:

$$\phi(w_i) = \int_0^{b_i} \frac{M}{EI(x)} dx + \phi^0 \tag{2}$$

and

$$I(x) = \begin{cases} \frac{1}{12} l_i (d_i + (t_i - d_i)\frac{x}{w_i})^3 & 0 \le x \le w_i \\ \frac{1}{12} l_i t_i^3 & w_i \le x \le b_i \end{cases} \tag{3}$$

The crease geometry parameters l_i, d_i, w_i, t_i and b_i are chosen based on the 3D CAD designs shown in Chapter 2. The ideal end caps are simulated by assigning zero crease stiffness at the boundary. Since the crease stiffness is a function of resting dihedral opening angle ϕ^0, it is dependent on the Miura-ori resting folding angle θ^0.

A.2 Facet bending stiffness

The facet bending stiffness is added to the truss elements that triangulates the facets in the equivalent truss-frame model. The facet bending is not considered in the rigid-folding kinematics, but it's important to be taken into account for the fluidic modules fabricated from realistic materials. The 3D printed modules show notable facet bulging when the internal pressure is very high. To derive

this stiffness parameter, we use the formulation introduced by Lobkovskey et al. for small bending angle de-formations ($\leq 0.1 radian$) [88]. This assumption is consistent with the facet deformations seen in experiments and finite element simulation. We define a scaling parameter SF to ensure that the equivalent facet bending stiffness k_i^f is at least an order higher than the crease stiffness k_i^c so that

$$k_i^f = SF \frac{E t_{fi}^3}{12(1-\nu^2)} (\frac{l_{di}}{t_{fi}})^(1/3) \tag{4}$$

where the elastic modulus E is estimated based on the dog bone tests shown in Figure 4,$\nu = 0.45$ is the Poisson's ratio, and t_{fi} is the facet thickness. Note that the facet bending stiffness is function of diagonal truss element length l_{di} which in turn is related to the crease lengths a, b and facet sector angle γ.

A.3 Facet stretching stiffness

The facet stretching stiffness is assigned to all of the truss elements in the truss-frame model. Filipov et al. introduced a stretching stiffness formulation to model thin sheets in origami [32]. They assumed isotropic material properties and uniform sheet thickness to describe in-plane stretching and shearing behavior of initially unbent facets. We adopt the same formulation, which provides satisfactory results to qualitatively describe the fluidic origami actuator performance. The equivalent stretching stiffness k_i^s for the truss element is calculated by,

$$k_{ki}^s = E \frac{A_{ki}}{k_{ki}} \tag{5}$$

where the sub-index k = a, b, or d represents the different truss members in the facet $\#i$ (Figure 2(a)), so that

$$A_{bi} = \frac{t_f(a^2 - \nu b^2)}{2a(1-\nu^2)} \ , \ A_{ai} = \frac{t_f(b^2 - \nu a^2)}{2b(1-\nu^2)} \ , \ A_{di} = \frac{\nu t_f (a^2+b^2)^{1.5}}{2ab(1-\nu^2)} \tag{6}$$

and

$$l_{bi} = b \ , \ l_{ai} = a \ , \ l_{di} = \sqrt{a^2 + b^2 - 2ab\cos\gamma} \tag{7}$$

Table 1: The normalized stiffness parameters of four examples of fluidic origami design shown in Figure 2

Stiffness	$\gamma = 20°, \theta^0 = 77.9°$	$\gamma = 20°, \theta^0 = 88°$	$\gamma = 60°, \theta^0 = 1°$	$\gamma = 60°, \theta^0 = 77.9°$
k^c/a	3.9	3.8	4.5	3.9
k^f/a	103.3	103.3	139.8	139.8
$k^s a$	$k_a^s = k_b^s = 1.4e^{-3}$ $k_d^s = 8.5e^{-3}$	$k_a^s = k_b^s = 1.4e^{-3}$ $k_d^s = 8.5e^{-3}$	$k_a^s = k_b^s = 1.4e^{-3}$ $k_d^s = 3.4e^{-3}$	$k_a^s = k_b^s = 1.4e^{-3}$ $k_d^s = 3.4e^{-3}$

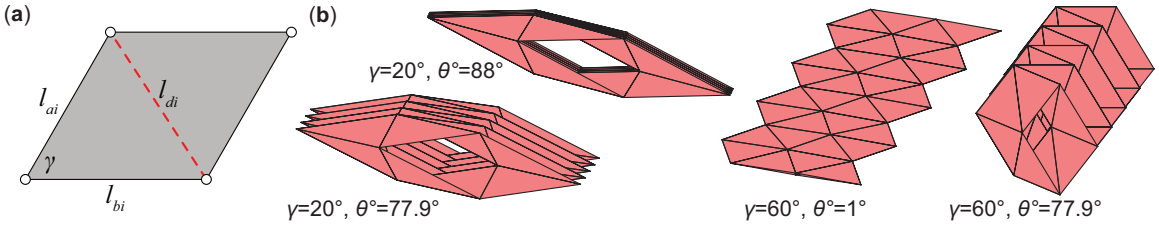

Figure 2: Estimating the facet bending and stretching stiffness.(a) The schematic diagram of a facet showing the truss-frame elements and geometric variables. (b) The four different examples of Miura-ori designs, and their corre-sponding stiffness parameters are summarized in Table A.3.

Thus, the stretching stiffness is inversely proportional to the length of truss elements. Based on the formulations above, we can calculate the stiffness parameters for every facet and crease, and use them for the parametric analysis of free-stroke and block force. Table A.3 shows the calculated stiffness parameters corresponding to four different examples of Miura-ori designs (Figure 2(b)). Results in this table clearly show the connection between the stiffness parameters and the Miura-ori designs. The crease folding stiffness k_i^c decreases when θ^0 increases. while change in γ doesn't have any effect on its value. The facet bending stiffness k_i^f increases when γ increases, while change in θ^0 doesn't have any effect on its value. The crease stretching stiffness k_a^s and k_b^s is independent of the changes in θ^0 and γ; while the facet stretching stiffness k_d^s decreases with an increasing γ.

Appendix B Equilibrium Paths Search

In this section, we elucidate how to calculate the equilibrium paths of a driving module consisting of any number of serially connected bistable segments. We calculate the "equilibrium path" followed by the driving module via searching for its *local potential energy minima* at a prescribed total length. The total potential energy (E_t) and total length (l_t) of a driving module of n bistable segments are defined as,

$$E_t = \sum_{i=1}^{n} E_i(l_i), \text{ and } l_t = \sum_{i=1}^{n} l_i, \tag{8}$$

where l_i is current length of the i^{th} segment and E_i is the corresponding strain potential energy.

The search for equilibrium paths can be defined as an optimization problem. The objective function of this optimization is the total strain energy E_t (a scalar function), and it is to be minimized over the $\mathcal{R}^{n-1}$ vector space of individual segment lengths $\mathbf{L} = [l_1 \dots l_{n-1}]$. An equality constraint regarding the prescribed total length must be satisfied so that $l_n = l_t - \sum_{i=1}^{n-1} l_i$. Therefore, the optimization problem can be described as follows:

$$\begin{aligned}
&\text{Minimize } E_t = \sum_{i=1}^{n-1} E_i(l_i) + E_n(l_t - \sum_{i=1}^{n-1} l_i) \\
&\text{satisfying the bounds } l_{i\ \min} \le l_i \le l_{i\ \max} \text{ and } l_{n\ \min} \le l_n \le l_{n\ \max}, \\
&\text{where } i = 1, 2 \dots n-1
\end{aligned} \tag{9}$$

The optimization problem described in equation 9 is solved for every prescribed total length of the driving module l_t, which is increased from its minimum to the maximum value with an incremental step Δl_t,

$$\begin{aligned}
l_t^{\min} &= \sum_{i=1}^{n} l_{i\ (0)}, \\
l_t^{\max} &= \sum_{i=1}^{n} l_{i\ (1)}, \\
\text{and } m &= \frac{(l_t^{\max} - l_t^{\min})}{\Delta l_t},
\end{aligned} \tag{10}$$

where m is the total number of increments. $l_{i\ (0)}$ and $l_{i\ (1)}$ are the i^{th} segment's length at its fully-

contracted stable state (0) and fully-extended stable state (1), respectively. Notice that $l_{i\ (0)}$ is different from $l_{i\ \min}$ by definition, and typically $l_{i\ \min} \leq l_{i\ (0)}$. Similarly $l_{i\ \max} \geq l_{i\ (1)}$.

For the j^{th} increment in l_t^j $(j = 2...m)$, the solutions of the optimization problem are the vectors of individual segment lengths $\mathbf{L}^j = [l_i^j ... l_{n-1}^j]$ corresponding to a local minima of E_t. The following pseudo-code describes the optimization algorithm used to search for the equilibrium path of driving module *when it is stretched from* $l_t^{\min}$.

Step 1: Initialize the optimization problem using the segment lengths at their fully-contracted stable states (aka. $l_{i\ (0)}$). We do not have to perform optimization for this first increment because it already corresponds to an energy minima. For segments $i = 1, 2...n-1$ define,

$$\begin{aligned} & l_i^1 = l_{i\ (0)} \text{ and } l_t^1 = l_t^{\min} \\ & \mathbf{L}^1 = [l_1^1 \dots l_{n-1}^1]. \end{aligned} \tag{11}$$

Step 2: Initiate the next increment step $j = 2$ so that,

$$\begin{aligned} & l_t^2 = l_t^1 + \Delta l_t \\ & \mathbf{L}_0^2 = [l_1^1 \dots l_{n-1}^1] = \mathbf{L}^1. \end{aligned} \tag{12}$$

Here, l_t^2 is the total length of driving module at this increment step, and $\mathbf{L}_0^2$ is the initial input needed to solve the optimization problem.

Step 3: Solve the optimization problem described in Equation 9. Here, $\mathbf{L}_0^2$ is the first guess for finding the optimized lengths of individual segments and the optimized result is recorded as $\mathbf{L}^2$. The length of the last segment is calculated as,

$$l_n^2 = l_t^2 - \sum_{i=1}^{n-1} l_i^2. \tag{13}$$

The corresponding total energy of the module is then written as,

$$E_t^{2\ \min} = \sum_{i=1}^{n} E_i^2(l_i^2) \tag{14}$$

Step 4: Prepare the next increment step by setting $\mathbf{L}_0^{j+1} = \mathbf{L}^j$ and $l_t^j = l_t^{\min} + (j-1)\Delta l_t$, $(j = 3...m)$. Then solve the optimization problem again using the procedures in Step 3. Notice that

the optimization output $\mathbf{L}^j$ in the j^{th} increment step is always used as the initial guess $\mathbf{L}_0^{j+1}$ of the next $(j+1)^{\text{th}}$ increment step.

Step 5: Repeat Steps 2 to 4 until $j = m$.

Equations 13 and 14 together provide the segment lengths and potential energy along the equilibrium path of this driving module when it is stretched from the minimun length $l_t^{\min}$. The equilibrium path for compressing the driving module need not be same as the path for stretching it. Thus, the similar procedure can be used to search for the other equilibrium path of the driving module when it is compressed from the maximum length $l_t^{\max}$.

Appendix C Origami Dynamics and PRC framework Algorithms

This section presents the basic algorithms for emulation and closed loop pattern generation tasks, so that the results described in this work can be reproduced. We also provide algorithms for Quadruped robot origami pattern design, since they involve modifications in the basic Miura-ori pattern, that is used in earlier demonstration studies. The algorithms are all developed in MATLAB® and use ode solver and regression functions in-built in the software.

C.1 Code for Emulation task of PRC with Miura-ori metasheet

```
clc
clear
thetai = 0;   % starting with flat state
xN = 3; % no. of cells in x-direction
yN = 3; % no. of cells in y-direction
zeta = 0.2; % damping coefficient
theta0 = [60]*pi/180;  % target folded state angle

b = 10; % miura unit-cell-length
a = 1.1*b; % miura unit-cell-length
angle = 50*pi/180;% miura unit-cell-angle
[NODESi,SPRINGS,CREASEi,Kcr,N_S,LE,RE,BE,TE,S_N] = Miura_geometry ...
(a,b,angle,thetai,xN,yN);
% defining miura-geometry with nodes and truss elements
[NODES0]=miura_geom(a,b,angle,theta0,xN,yN); % folded state of miura-ori
Ncr = [a b]; %
[INPUT,op_N,Ns,Ss,Srow,Scol,Fold_Sp,CREASE] = input_para(NODES0,SPRINGS,CREASEi,Kcr, a,Ncr);
% defining feedback and input creases
%% setting spring stiffnesses; Material constants; fixed nodes
EA = 100;
Kaxial = EA;
Kfacet =10;
Kmin = 0.005;
Kmax = 0.5;
m = 0.01*ones(1,Ns); % constant nodal mass
% Kcrease = 1*ones(Ss,1); % constant crease stiffness
Kcrease = Kmin+(Kmax-Kmin)*rand(Ss,1); % variable crease stiffness
Kcrease([INPUT]) = 1;
% I set higher stiffness at input creases, you can try playing with this parameter.
for i = 1:Ss
if Kcr(i)==-1
    Kcrease(i) = Kfacet; % facet stiffness
end
end
% boundary conditions: fixed nodes
Fix_x = [LE(1:2:length(LE))];
Fix_y = [LE(1)];
Fix_z = [LE(1:2:length(LE)) RE(1:2:length(RE))];
FIXED = [Fix_x Fix_y Fix_z];
```

```
% plotting 3-D geometry
AA = (1:Ss).*any(CREASE');
k = 1;
for j = find(AA)
facets(k,:) = CREASEi(j,[1 3 4]);
facets(k+1,:) = CREASEi(j,[2 3 4]);
k = k+2;
end

for i = 1:length(facets)
fill3(NODES0(facets(i,:),1),NODES0(facets(i,:),2),NODES0(facets(i,:),3),[0.9 0.9 0.9])
hold on
end
p7 = plot3(NODES0(Fix_x,1),NODES0(Fix_x,2),NODES0(Fix_x,3),'*b');
p8 = plot3(NODES0(Fix_y,1),NODES0(Fix_y,2),NODES0(Fix_y,3),'+m');
p9 = plot3(NODES0(Fix_z,1),NODES0(Fix_z,2),NODES0(Fix_z,3),'or');
for j = 1:length(INPUT)
p10 = fill3([NODES0(Srow(INPUT(j)),1) NODES0(Scol(INPUT(j)),1)],[NODES0(Srow(INPUT(j)),2) ...
NODES0(Scol(INPUT(j)),2)],[NODES0(Srow(INPUT(j)),3) NODES0(Scol(INPUT(j)),3)], ...
[0.5 0],'EdgeColor','r');
end
legend([p7,p8,p9,p10],{'X','Y','Z','INPUT'})
hold off
axis 'equal'
view([70 60])

li = zeros(Ss,1);
for k = 1:Ss
li(k) = norm(NODES0(SPRINGS(k,1),:)-NODES0(SPRINGS(k,2),:));  % nominal length
end
phi0 = zeros(Ss,1);
for k = find(AA)
   [phi0(k),~,~,~,~] = find_phi_dphi(NODES0,CREASE(k,:));  % nominal fold angle
end
%% Adding effect due to gravity
ic = zeros(1,6*Ns); % initial conditions
tspan = 0:10^-3:10; % simulation time in seconds
res = 10^-6;
tic
options = odeset('RelTol',res,'AbsTol',res);
[~,Yg] = ode45(@(t,Y) myode_gravity(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0, Kaxial,Kcrease,CREASE, ...
NODES0,zeta, m,Fix_x,Fix_y,Fix_z), tspan, ic,options); % solving the ODE
toc
Pxg = Yg(:,1:Ns);
Pyg = Yg(:,Ns+1:Ns+Ns);
Pzg = Yg(:,2*Ns+1:2*Ns+Ns);
NODES0fg = NODES0 +[Pxg(end,:);Pyg(end,:);Pzg(end,:)]'; % equilibrium nodes

phi0 = zeros(Ss,1);
for k = find(AA)
   [phi0(k),~,~,~,~] = find_phi_dphi(NODES0fg,CREASE(k,:));  % equilibrium fold angle
```

```
end
li = zeros(Ss,1);
for k = 1:Ss
li(k) = norm(NODES0fg(SPRINGS(k,1),:)-NODES0fg(SPRINGS(k,2),:));  % equilibrium length
end
% plotting equilibrium shape of miura
figure
for i = 1:(Ss)
fill3([NODES0fg(Srow(i),1) NODES0fg(Scol(i),1)],[NODES0fg(Srow(i),2) NODES0fg(Scol(i),2)], ...
[NODES0fg(Srow(i),3) NODES0fg(Scol(i),3)],[0.5 0],'EdgeColor','black')
hold on
end
p7 = plot3(NODES0fg(Fix_x,1),NODES0fg(Fix_x,2),NODES0fg(Fix_x,3),'*b');
p8 = plot3(NODES0fg(Fix_y,1),NODES0fg(Fix_y,2),NODES0fg(Fix_y,3),'+m');
p9 = plot3(NODES0fg(Fix_z,1),NODES0fg(Fix_z,2),NODES0fg(Fix_z,3),'or');

for j = 1:length(INPUT)
p10 = fill3([NODES0fg(Srow(INPUT(j)),1) NODES0fg(Scol(INPUT(j)),1)],[NODES0fg(Srow(INPUT(j)),2) ...
NODES0fg(Scol(INPUT(j)),2)],[NODES0fg(Srow(INPUT(j)),3) NODES0fg(Scol(INPUT(j)),3)], ...
[0.5 0],'EdgeColor','r');
end
legend([p7,p8,p9,p10],{'X','Y','Z','INPUT'})
hold off
grid on
axis 'equal'
view([70 60])
xlabel('x')
ylabel('y')
zlabel('z')
%% TRAINING the OUTPUTS
delt = 1*1E-3; % time-step
time = 0:delt:1.0; % training time
endT = length(time);
% setting up input
f1 = 2.11*0.001;
f2 = 3.73*0.001;
f3 = 4.33*0.001;
eps =@(t) 0.5*(0.1*sin(2*pi*f1*t).*sin(2*pi*f2*t)+2).*square(t*pi/80)+ ...
0.2*(0.2*sin(2*pi*f1*t).*sin(2*pi*f3*t)+2).*square(t*pi/150)+ ...
0.1*(0.1*sin(2*pi*f2*t).*sin(2*pi*f3*t)+2).*square(t*pi/200)+2;
xin =@(t) 0.5*(eps(t));
% TRAINING the OUTPUTS
win = zeros(Ss,1);
win(INPUT) = 1;

tic
ic = zeros(1,6*Ns);
tspan = time;
options = odeset('RelTol',res,'AbsTol',res);
[~,Y] = ode45(@(t,Y) myode(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0,Kaxial,Kcrease,CREASE,NODES0fg, ...
zeta,m,win,INPUT,xin,Fix_x,Fix_y,Fix_z,time), tspan, ic,options);
```

```
Px = Y(:,1:Ns);
Py = Y(:,Ns+1:Ns+Ns);
Pz = Y(:,2*Ns+1:2*Ns+Ns);
toc

% find current spring folding angle matrix
gamma = 1*1E-3; % max. noise amplitude
L = zeros(endT,length(Fold_Sp));
for count = 1:endT
NODES = NODES0fg+[Px(count,:);Py(count,:);Pz(count,:)]';
noise1=-gamma+(2*gamma)*rand(length(Fold_Sp),1);  % adding noise
phi = zeros(Ss,1);
for k = Fold_Sp
    [phi(k),~,~,~,~] = find_phi_dphi(NODES,CREASE(k,:));
end
L(count,:) = phi(Fold_Sp) + noise1;
end

figure
plot(time,L)
%
%% Training readout using Linear Least Squares
% Setup the function you want to emulate here as emu_fun.
n_wash = 30000;
t_train = n_wash+1:floor(2*endT/3);
t_test = floor(2*endT/3)+1:endT;
%
[Wout1,bint1,r1,rint1,stats1] = regress(emu_fun(t_train),[ones(length(t_train),1) L(t_train,:)]);
emu_result=sum(Wout1(2:end).*L(t_test,:)')+Wout1(1);   % emulation task output

stats1(1)

%%
% save data.mat

%% all the related functions

function [NODES,SPRINGS,CREASE,Kcr,N_S,Left_Edge,Right_Edge,Bottom_Edge,Top_Edge,S_N] = Miura_geomet
[NODES,L,W,V,fa]=miura_geom(a,b,gamma,thetai,xN,yN);
x_max = max(NODES(:,1));
y_max = max(NODES(:,2));
Ncr = [a b fa];
Fcr = [fa];
Ns = length(NODES);
eps =  1E-4;
% Node Connections
N_N = zeros(Ns,Ns);
for k = 1:length(Ncr)
for i = 1:Ns
   for j = setdiff(1:Ns,1:i)
        if abs(norm(NODES(i,:)-NODES(j,:))-Ncr(k))<=eps
```

```
           N_N(i,j)=1;
         end
   end
end
end

% Spring definition
[Srow,Scol]=find(N_N);
SPRINGS = [Srow,Scol];
Ss = length(SPRINGS);

% Facet definition
Facet_Spring = zeros(Ss,1);
for k = 1:length(Fcr)
for i = 1:Ss
li(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));  % nominal length in flat state
if abs(li(i)-Fcr(k))<eps
    Facet_Spring(i)=1;
end
end
end
%Creases definition
    Left_Edge=[];
    Right_Edge=[];
    Bottom_Edge=[];
    Top_Edge=[];
for i = 1:Ns
    if abs(NODES(i,1)-0)<eps
    Left_Edge = [Left_Edge i];
    elseif abs(NODES(i,1)-x_max) <eps
    Right_Edge = [Right_Edge i];
    end
    if abs(NODES(i,2)-0)<eps
    Bottom_Edge = [Bottom_Edge i];
    elseif abs(NODES(i,2)-(y_max)) <eps
    Top_Edge = [Top_Edge i];
    end
end

[CREASE,Kcr,N_S,S_N] = get_creases(SPRINGS,Ss,Ns,Facet_Spring);

end

%-----
function [CREASE,Kcr,N_S,S_N] = get_creases(SPRINGS,Ss,Ns,Facet_Spring)
for j = 1:Ss
    if Facet_Spring(j)==1
         Kcr(j)= -1 ;
    else
         Kcr(j)= +1 ;
    end
```

```
        kka = ones(Ss,1);
        kkb = ones(Ss,1);
        for i = 1:Ss-1
            k = setdiff(1:Ss,j);
            [~,ia] = intersect(SPRINGS(k(i),:),SPRINGS(j,1));
            [~,ib] = intersect(SPRINGS(k(i),:),SPRINGS(j,2));
            kka(k(i)) = isempty(ia);
            kkb(k(i)) = isempty(ib);
        end
        arr = sort([SPRINGS(kka==0,1); SPRINGS(kka==0,2); SPRINGS(kkb==0,1); SPRINGS(kkb==0,2)]);
        [C,~,ic] = unique(arr);
        a_counts = accumarray(ic,1);
        value_counts = [C, a_counts];
        mp = setdiff(value_counts(:,1).*(a_counts==2),[SPRINGS(j,:) 0]);
        if length(mp)>=2
        CREASE(j,1:4) = [mp(1:2); SPRINGS(j,:)'];
        else
        CREASE(j,1:4) = zeros(1,4);
        Kcr(j) = 0;
        end

end

% Creating Node to Spring and Nodes Matrix
for i = 1:Ns
    for j = 1:Ss
    if ismember(i,SPRINGS(j,:))==1
N_S(i,j) = setdiff(SPRINGS(j,:),i);
    else
N_S(i,j) = 0;
    end
    end
end
% Kcr = 0 for boundary crease, +1 for M/V , -1 for facet
S_N = zeros(Ns,10);
for i = 1:Ns
 [~,Sco] = setdiff(N_S(i,:),0);
 S_N(i,1:length(Sco)) = Sco';
end
end

%-----
function [phi,dphidpx,dphidpy,dphidpz,cosphi] = find_phi_dphi(NODES,CREASE)
Pl = NODES(CREASE(1),:);
Pi = NODES(CREASE(2),:);
Pj = NODES(CREASE(3),:);
Pk = NODES(CREASE(4),:);

m = cross((Pi-Pj),(Pk-Pj));
n = cross((Pk-Pj),(Pk-Pl));
```

```
cosphi = dot(m,n)/norm(n)/norm(m);
if dot(m,(Pk-Pl))==0
    phi = mod(1*real(acos(cosphi)),2*pi);
else
    phi = mod(sign(dot(m,(Pk-Pl)))*real(acos(cosphi)),2*pi);
end
dphidpl = -norm(Pk-Pj)/(norm(n))^2*n;
dphidpi = norm(Pk-Pj)/(norm(m))^2*m;
dphidpj = (dot((Pi-Pj),(Pk-Pj))/(norm(Pk-Pj))^2-1)*dphidpi - ...
dot((Pk-Pl),(Pk-Pj))/(norm(Pk-Pj))^2*dphidpl;
dphidpk = (dot((Pk-Pl),(Pk-Pj))/(norm(Pk-Pj))^2-1)*dphidpl - ...
dot((Pi-Pj),(Pk-Pj))/(norm(Pk-Pj))^2*dphidpi;

dphidpx = [dphidpl(1) dphidpi(1)  dphidpj(1)  dphidpk(1) ];
dphidpy = [dphidpl(2) dphidpi(2)  dphidpj(2)  dphidpk(2) ];
dphidpz = [dphidpl(3) dphidpi(3)  dphidpj(3)  dphidpk(3) ];
end

%-----
function [NODES,L,W,V,fa]=miura_geom(a,b,gamma,theta,xN,yN)
Ht = a*sin(gamma)*sin(theta);
L = b*tan(gamma)*cos(theta)./(sqrt(1+cos(theta).^2*tan(gamma)^2));
W = a*sqrt(1-sin(theta).^2*sin(gamma)^2);
V = b./sqrt(1+cos(theta).^2*tan(gamma)^2);

A = [0 0 0];
B = [L V 0];
C = [2*L 0 0];
D = [0 W Ht];
E = [L W+V Ht];
F = [2*L W Ht];
G = [0 2*W 0];
H = [L 2*W+V 0];
I = [2*L 2*W 0];
NODESb = [A; B; C; D; E; F; G; H; I];
% Add periodic copies of miura in x and y direction
xNodes = [repmat(B,xN,1); repmat(C,xN,1); repmat(E,xN,1); repmat(F,xN,1); repmat(H,xN,1); ...
repmat(I,xN,1)+repmat([[1:xN]'*2*L zeros(xN,2)],6,1);
yNodes = [];
for i = 1:yN
yNodesm = [D; G; E; H; F; I; xNodes((2*xN+1):end,:)]+repmat([0 i*2*W 0],6+length(xNodes)-(2*xN),1);
yNodes = [yNodes; yNodesm];
end
NODES = [NODESb;xNodes;yNodes];
fa = sqrt(a^2+b^2-2*a*b*cos(pi-gamma));
end

%-----
function [INPUT,op_N,Ns,Ss,Srow,Scol,Fold_Sp,CREASE] = input_para(NODES0,SPRINGS,CREASEi,Kcr,a,Ncr)
Ns = length(NODES0);
```

```
Ss = length(SPRINGS);
Srow = SPRINGS(:,1);
Scol = SPRINGS(:,2);
li = zeros(Ss,1);
for i = 1:Ss
li(i) = norm(NODES0(SPRINGS(i,1),:)-NODES0(SPRINGS(i,2),:));  % nominal length in flat state
end

AA = (1:Ss).*any(CREASEi');
%correcting mountain-valley assignment: dist is negative then mountain ;
% dist is positive then valley
% phi>pi for mountain;  phi<pi for valley
CREASE = CREASEi;
for j = find(AA)
% CREASE(j,:)
mid1 = (NODES0(CREASEi(j,1),:)+NODES0(CREASEi(j,2),:))/2; % facet
mid2 = (NODES0(CREASEi(j,3),:)+NODES0(CREASEi(j,4),:))/2; % crease
[phi,~,~,~,~] = find_phi_dphi(NODES0,CREASEi(j,:));
if abs(li(j)-a)<10^-6
dist = mid1(3)-mid2(3);
if dist<0 && phi<pi || dist>0 && phi>pi
    CREASE(j,1:4) = CREASEi(j,1:4);
elseif dist<0 && phi>pi || dist>0 && phi<pi
    CREASE(j,1:2) = fliplr(CREASEi(j,1:2));
end
else
dist = mid1(3)-mid2(3);
if dist>0 && phi<pi || dist<0 && phi>pi
    CREASE(j,1:4) = CREASEi(j,1:4);
elseif dist>0 && phi>pi || dist<0 && phi<pi
    CREASE(j,1:2) = fliplr(CREASEi(j,1:2));
end
end
end

% figure
% Gr = graph(SPRINGS(:,1), SPRINGS(:,2),phi0);
% plot(Gr,'EdgeLabel',Gr.Edges.Weight)

Fold_Sp = intersect(find(Kcr==1),AA);
y_cr = [];
for k = 1:length(Ncr)
for i = Fold_Sp
    if abs(li(i)-Ncr(k))<10^-5
        long_crk = i;
    else
        long_crk = [];
    end
    y_cr = [y_cr long_crk];
end
end
```

```
y_c = unique(y_cr);
op_N = length(Fold_Sp);
INPUT = y_c(randperm(length(y_c),ceil(op_N*25/100))); % selecting 25% creases as input creases

end

%-----
function [dYdt] = myode_gravity(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0i,Kaxial, ...
Kcrease,CREASE,NODES0,zeta,m,Fix_x,Fix_y,Fix_z)
Px = Y(1:Ns);
Py = Y(Ns+1:Ns+Ns);
Pz = Y(2*Ns+1:2*Ns+Ns);
vx = Y(3*Ns+1:3*Ns+Ns);
vy = Y(4*Ns+1:4*Ns+Ns);
vz = Y(5*Ns+1:5*Ns+Ns);

NODES = NODES0+[Px Py Pz];
l = zeros(Ss,1);
for i = 1:Ss
l(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));
end
Faxialx=zeros(Ss,Ns);
Faxialy=zeros(Ss,Ns);
Faxialz=zeros(Ss,Ns);
Fcreasex=zeros(Ss,Ns);
Fcreasey=zeros(Ss,Ns);
Fcreasez=zeros(Ss,Ns);

for j = 1:Ss
[NSco] = setdiff(N_S(:,j),0);
% Axial Force Calculations
dldpi = (NODES(NSco(2),:)-NODES(NSco(1),:))/norm(NODES(NSco(2),:)-NODES(NSco(1),:));
Faxialx(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(1) dldpi(1)];
Faxialy(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(2) dldpi(2)];
Faxialz(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(3) dldpi(3)];
phi0 = phi0i;
phi = phi0i;
end
AA = (1:Ss).*any(CREASE');
for j = find(AA)
% Crease folding force Calculations
    [phi(j),dphidpx,dphidpy,dphidpz,~] = find_phi_dphi(NODES,CREASE(j,:));
    Fcreasex(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpx;
    Fcreasey(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpy;
    Fcreasez(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpz;
end

% Damping Force Calculations
Fdampingx = zeros(1,Ns);
Fdampingy = zeros(1,Ns);
Fdampingz = zeros(1,Ns);
```

```
for i = 1:Ns
[Nco,Sco] = setdiff(N_S(i,:),0);
dVdampx(1:length(Nco)) = (vx(Nco)-vx(i))/2;
dVdampy(1:length(Nco)) = (vy(Nco)-vy(i))/2;
dVdampz(1:length(Nco)) = (vz(Nco)-vz(i))/2;
cdamp = 2*zeta*sqrt(Kaxial./l(Sco)*m(i));

Fdampingx(i) = (sum(cdamp.*dVdampx'));
Fdampingy(i) = (sum(cdamp.*dVdampy'));
Fdampingz(i) = (sum(cdamp.*dVdampz'));
clear dVdampx dVdampy dVdampz
end

% Add gravity
Fg = -9.81*m;
% Calculating New Velocity and Position
Fx = sum(Faxialx)+sum(Fcreasex)+(Fdampingx);
Fy = sum(Faxialy)+sum(Fcreasey)+(Fdampingy);
Fz = sum(Faxialz)+sum(Fcreasez)+(Fdampingz)+Fg;

ax = (Fx)./m;
ay = (Fy)./m;
az = (Fz)./m;

ax(Fix_x) = 0;
ay(Fix_y) = 0;
az(Fix_z) = 0;

dYdt = [Y(3*Ns+1:3*Ns+3*Ns); [ax ay az]'];

end

%-----
function [dYdt] = myode(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0i,Kaxial,Kcrease,CREASE,NODES0, ...
zeta,m,win,INPUT,xin,Fix_x,Fix_y,Fix_z,time)
Px = Y(1:Ns);
Py = Y(Ns+1:Ns+Ns);
Pz = Y(2*Ns+1:2*Ns+Ns);
vx = Y(3*Ns+1:3*Ns+Ns);
vy = Y(4*Ns+1:4*Ns+Ns);
vz = Y(5*Ns+1:5*Ns+Ns);

NODES = NODES0+[Px Py Pz];
l = zeros(Ss,1);
for i = 1:Ss
l(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));
end
Faxialx=zeros(Ss,Ns);
Faxialy=zeros(Ss,Ns);
Faxialz=zeros(Ss,Ns);
for j = 1:Ss
```

```
[NSco] = setdiff(N_S(:,j),0);
% Axial Force Calculations
dldpi = (NODES(NSco(2),:)-NODES(NSco(1),:))/norm(NODES(NSco(2),:)-NODES(NSco(1),:));
Faxialx(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(1) dldpi(1)];
Faxialy(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(2) dldpi(2)];
Faxialz(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(3) dldpi(3)];
end
Fcreasex=zeros(Ss,Ns);
Fcreasey=zeros(Ss,Ns);
Fcreasez=zeros(Ss,Ns);
phi = zeros(Ss,1);
phi0 = phi0i;
phi0(INPUT) = win(INPUT)*tanh(xin(t))+phi0i(INPUT);
AA = (1:Ss).*any(CREASE');
for j = find(AA)
% Crease folding force Calculations
    [phi(j),dphidpx,dphidpy,dphidpz,~] = find_phi_dphi(NODES,CREASE(j,:));
    Fcreasex(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpx;
    Fcreasey(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpy;
    Fcreasez(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpz;
end

% Damping Force Calculations
Fdampingx = zeros(1,Ns);
Fdampingy = zeros(1,Ns);
Fdampingz = zeros(1,Ns);
for i = 1:Ns
[Nco,Sco] = setdiff(N_S(i,:),0);

dVdampx(1:length(Nco)) = (vx(Nco)-vx(i))/2;
dVdampy(1:length(Nco)) = (vy(Nco)-vy(i))/2;
dVdampz(1:length(Nco)) = (vz(Nco)-vz(i))/2;
cdamp = 2*zeta*sqrt(Kaxial./l(Sco)*m(i));

Fdampingx(i) = (sum(cdamp.*dVdampx'));
Fdampingy(i) = (sum(cdamp.*dVdampy'));
Fdampingz(i) = (sum(cdamp.*dVdampz'));
clear dVdampx dVdampy dVdampz
end

% Calculating New Velocity and Position
Fx = sum(Faxialx)+sum(Fcreasex)+(Fdampingx);
Fy = sum(Faxialy)+sum(Fcreasey)+(Fdampingy);
Fz = sum(Faxialz)+sum(Fcreasez)+(Fdampingz);
ax = (Fx)./m;
ay = (Fy)./m;
az = (Fz)./m;
ax(Fix_x) = 0;
ay(Fix_y) = 0;
az(Fix_z) = 0;
dYdt = [Y(3*Ns+1:3*Ns+3*Ns); [ax ay az]'];
```

```
% size(dYdt)

end
```

C.2 Code for Closed loop task of PRC with Miura-ori metasheet

```
clc
clear
thetai = 0;   % starting with flat state
xN = 3; % no. of cells in x-direction
yN = 3; % no. of cells in y-direction
zeta = 0.2; % damping coefficient
theta0 = [60]*pi/180;  % target folded state angle

b = 10; % miura unit-cell-length
a = 1.1*b; % miura unit-cell-length
angle = 50*pi/180;% miura unit-cell-angle
[NODESi,SPRINGS,CREASEi,Kcr,N_S,LE,RE,BE,TE,S_N] = Miura_geometry(a,b,angle,thetai,xN,yN);
% defining miura-geometry with nodes and truss elements
[NODES0]=miura_geom(a,b,angle,theta0,xN,yN); % folded state of miura-ori
Ncr = [a b]; %
[FEEDBACK1,FEEDBACK2,INPUT,op_N,Ns,Ss,Srow,Scol,Fold_Sp,CREASE] = ...
input_para(NODES0,SPRINGS,CREASEi,Kcr,a,Ncr); % defining feedback and input creases
FB = [FEEDBACK1 FEEDBACK2];
%% setting spring stiffnesses; Material constants; fixed nodes
EA = 100;
Kaxial = EA;
Kfacet =10;
Kmin = 0.005;
Kmax = 0.5;
m = 0.01*ones(1,Ns); % constant nodal mass
Kcrease = 1*ones(Ss,1); % constant crease stiffness
% Kcrease = Kmin+(Kmax-Kmin)*rand(Ss,1); % variable crease stiffness
Kcrease([FB INPUT]) = 1;
for i = 1:Ss
if Kcr(i)==-1
    Kcrease(i) = Kfacet; % facet stiffness
end
end
% boundary conditions: fixed nodes
Fix_x = [LE(1:2:length(LE))];
Fix_y = [LE(1)];
Fix_z = [LE(1:2:length(LE)) RE(1:2:length(RE))];
FIXED = [Fix_x Fix_y Fix_z];
% plotting 3-D geometry
AA = (1:Ss).*any(CREASE');
k = 1;
for j = find(AA)
facets(k,:) = CREASEi(j,[1 3 4]);
facets(k+1,:) = CREASEi(j,[2 3 4]);
```

```
k = k+2;
end

for i = 1:length(facets)
fill3(NODES0(facets(i,:),1),NODES0(facets(i,:),2),NODES0(facets(i,:),3),[0.9 0.9 0.9])
hold on
end
p7 = plot3(NODES0(Fix_x,1),NODES0(Fix_x,2),NODES0(Fix_x,3),'*b');
p8 = plot3(NODES0(Fix_y,1),NODES0(Fix_y,2),NODES0(Fix_y,3),'+m');
p9 = plot3(NODES0(Fix_z,1),NODES0(Fix_z,2),NODES0(Fix_z,3),'or');
for j = 1:length(FEEDBACK1)
p10 = fill3([NODES0(Srow(FEEDBACK1(j)),1) NODES0(Scol(FEEDBACK1(j)),1)], ...
[NODES0(Srow(FEEDBACK1(j)),2) NODES0(Scol(FEEDBACK1(j)),2)],[NODES0(Srow(FEEDBACK1(j)),3) ... NODES0
end
for j = 1:length(FEEDBACK2)
p11 = fill3([NODES0(Srow(FEEDBACK2(j)),1) NODES0(Scol(FEEDBACK2(j)),1)], ...
[NODES0(Srow(FEEDBACK2(j)),2) NODES0(Scol(FEEDBACK2(j)),2)],[NODES0(Srow(FEEDBACK2(j)),3) ... NODES0
end
legend([p7,p8,p9,p10,p11],{'X','Y','Z','FEEDBACK1','FEEDBACK2'})
hold off
axis 'equal'
view([70 60])

li = zeros(Ss,1);
for k = 1:Ss
li(k) = norm(NODES0(SPRINGS(k,1),:)-NODES0(SPRINGS(k,2),:));  % nominal length
end
phi0 = zeros(Ss,1);
for k = find(AA)
   [phi0(k),~,~,~,~] = find_phi_dphi(NODES0,CREASE(k,:));  % nominal fold angle
end
%% Adding effect due to gravity
ic = zeros(1,6*Ns); % initial conditions
tspan = 0:10^-3:10; % simulation time
res = 10^-6;
tic
options = odeset('RelTol',res,'AbsTol',res);
[~,Yg] = ode45(@(t,Y) myode_gravity(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0,Kaxial, ...
Kcrease,CREASE,NODES0,zeta,m,Fix_x,Fix_y,Fix_z), tspan, ic,options); % solving the ODE
toc
Pxg = Yg(:,1:Ns);
Pyg = Yg(:,Ns+1:Ns+Ns);
Pzg = Yg(:,2*Ns+1:2*Ns+Ns);
NODES0fg = NODES0 +[Pxg(end,:);Pyg(end,:);Pzg(end,:)]'; % equilibrium nodes

phi0 = zeros(Ss,1);
for k = find(AA)
   [phi0(k),~,~,~,~] = find_phi_dphi(NODES0fg,CREASE(k,:));  % equilibrium fold angle
end
li = zeros(Ss,1);
for k = 1:Ss
```

```
li(k) = norm(NODES0fg(SPRINGS(k,1),:)-NODES0fg(SPRINGS(k,2),:));  % equilibrium length
end
% plotting equilibrium shape of miura
figure
for i = 1:(Ss)
fill3([NODES0fg(Srow(i),1) NODES0fg(Scol(i),1)],[NODES0fg(Srow(i),2) NODES0fg(Scol(i),2)], ...
[NODES0fg(Srow(i),3) NODES0fg(Scol(i),3)],[0.5 0],'EdgeColor','black')
hold on
end
p7 = plot3(NODES0fg(Fix_x,1),NODES0fg(Fix_x,2),NODES0fg(Fix_x,3),'*b');
p8 = plot3(NODES0fg(Fix_y,1),NODES0fg(Fix_y,2),NODES0fg(Fix_y,3),'+m');
p9 = plot3(NODES0fg(Fix_z,1),NODES0fg(Fix_z,2),NODES0fg(Fix_z,3),'or');

for j = 1:length(FEEDBACK1)
p10 = fill3([NODES0fg(Srow(FEEDBACK1(j)),1) NODES0fg(Scol(FEEDBACK1(j)),1)], ...[NODES0fg(Srow(FEEDB
end
for j = 1:length(FEEDBACK2)
p11 = fill3([NODES0fg(Srow(FEEDBACK2(j)),1) NODES0fg(Scol(FEEDBACK2(j)),1)], ...
[NODES0fg(Srow(FEEDBACK2(j)),2) NODES0fg(Scol(FEEDBACK2(j)),2)],[NODES0fg(Srow(FEEDBACK2(j)),3) ...
NODES0fg(Scol(FEEDBACK2(j)),3)],[0.5 0],'EdgeColor','g');
end

for j = 1:length(INPUT)
p12 = fill3([NODES0fg(Srow(INPUT(j)),1) NODES0fg(Scol(INPUT(j)),1)],[NODES0fg(Srow(INPUT(j)),2) ...
end
legend([p7,p8,p9,p10,p11,p12],{'X','Y','Z','FEEDBACK1','FEEDBACK2','INPUT'})
hold off
grid on
axis 'equal'
view([70 60])
xlabel('x')
ylabel('y')
zlabel('z')
%% TRAINING the OUTPUTS
delt = 1*1E-3; % time-step
time = 0:delt:500.0; % training time
endT = length(time);
% setting up input
f1 = 2.11*0.001;
f2 = 3.73*0.001;
f3 = 4.33*0.001;
eps =@(t) 0.5*(0.1*sin(2*pi*f1*t).*sin(2*pi*f2*t)+2).*square(t*pi/80)+ ...
0.2*(0.2*sin(2*pi*f1*t).*sin(2*pi*f3*t)+2).*square(t*pi/150)+ ...
0.1*(0.1*sin(2*pi*f2*t).*sin(2*pi*f3*t)+2).*square(t*pi/200)+2;
xin =@(t) 0.5*(eps(t));
%
ic = [0.5 0.5];
[x1s,x2s] = reference(time,ic,eps); % Reference output
% setting up feedback weights
wfbv = zeros(1,length(FB));
for k = 1:length(FB)
```

```
    kk = FB(k);
    if phi0(kk)>pi
wfbv(k) = (2*pi-0.2-phi0(kk))/tanh(max(x2s));
    elseif phi0(kk)<pi
wfbv(k) = (0.2-phi0(kk))/tanh(min(x2s));
    end
end
wfbval = mean(wfbv);
phi0A = phi0;
% TRAINING the OUTPUTS
wfb = zeros(Ss,1);
wfb(FB) = wfbval;
win = zeros(Ss,1);
win(INPUT) = 1;

tic
ic = zeros(1,6*Ns);
tspan = time;
options = odeset('RelTol',res,'AbsTol',res);
[~,Y] = ode45(@(t,Y) myode(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0,Kaxial,Kcrease,CREASE,NODES0fg,zeta,m,win,
INPUT,xin,Fix_x,Fix_y,Fix_z,wfb,x1s,x2s,FEEDBACK1,FEEDBACK2,time), tspan, ic,options);
Px = Y(:,1:Ns);
Py = Y(:,Ns+1:Ns+Ns);
Pz = Y(:,2*Ns+1:2*Ns+Ns);
toc

% find current spring folding angle matrix
gamma = 1*1E-3; % max. noise amplitude
L = zeros(endT,length(Fold_Sp));
for count = 1:endT
NODES = NODES0fg+[Px(count,:);Py(count,:);Pz(count,:)]';
noise1=-gamma+(2*gamma)*rand(length(Fold_Sp),1);  % adding noise
phi = zeros(Ss,1);
for k = Fold_Sp
    [phi(k),~,~,~,~] = find_phi_dphi(NODES,CREASE(k,:));
end
L(count,:) = phi(Fold_Sp) + noise1;
end

figure
plot(time,L)
%
% Training readout using Linear Least Squares
n_wash = 30000;
t_train = n_wash+1:floor(2*endT/3);
t_test = floor(2*endT/3)+1:endT;
%
[Wout1,bint1,r1,rint1,stats1] = regress(x1s(t_train),[ones(length(t_train),1) L(t_train,:)]);
[Wout2,bint2,r2,rint2,stats2] = regress(x2s(t_train),[ones(length(t_train),1) L(t_train,:)]);
y1tL=sum(Wout1(2:end).*L(t_test,:)')+Wout1(1);   % current output-1
y2tL=sum(Wout2(2:end).*L(t_test,:)')+Wout2(1);   % current output-2
```

```
stats1(1)
stats2(1)

%% Closed loop simulation
time1=time(endT):delt:500.0+time(endT); % simulation time
endT1=length(time1);

tic
ic = Y(endT-1,:);
tspan = time1;
options = odeset('RelTol',res,'AbsTol',res);
[~,YcL] = ode45(@(t,Y) myodeC(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0,Kaxial,Kcrease,CREASE,NODES0fg,zeta,m, .
Fix_x,Fix_y,Fix_z,wfb,win,INPUT,xin,Wout1,Wout2,FEEDBACK1,FEEDBACK2,Fold_Sp), tspan, ic,options);
PxcL = YcL(:,1:Ns);
PycL = YcL(:,Ns+1:Ns+Ns);
PzcL = YcL(:,2*Ns+1:2*Ns+Ns);
toc
% find current spring folding angles, outputs
LcL = zeros(endT1,length(Fold_Sp));
for count = 1:endT1
NODES = NODES0fg+[PxcL(count,:);PycL(count,:);PzcL(count,:)]';
phi = zeros(Ss,1);
for k = Fold_Sp
    [phi(k),~,~,~,~] = find_phi_dphi(NODES,CREASE(k,:));
end
LcL(count,:) = phi(Fold_Sp);
end
Y1cL=sum(Wout1(2:end).*LcL')+Wout1(1);   % current output-1
Y2cL=sum(Wout2(2:end).*LcL')+Wout2(1);   % current output-2

%% plotting closed loop
icRL = [Y1cL(1) Y2cL(1)];
[x1L,x2L] = reference(time1,icRL,eps);

MSE1 = sum((Y1cL(1:10000)-x1L(1:10000)').^2)/10000
MSE2 = sum((Y2cL(1:10000)-x2L(1:10000)').^2)/10000
MSE = sqrt(MSE1^2+MSE2^2)
figure
plot(time1,x1L,time1,x2L,time1,Y1cL,time1,Y2cL)
figure
plot(x1L,x2L,Y1cL,Y2cL)
%% closed loop starting from rest
time2=0:delt:500.0; % simulation time
endT2=length(time2);

tic
ic = zeros(1,6*Ns);
tspan = time2;
options = odeset('RelTol',res,'AbsTol',res);
tic
```

```
[~,YcL0] = ode45(@(t,Y) myodeC(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0,Kaxial,Kcrease,CREASE,NODES0fg,zeta,m,
Fix_x,Fix_y,Fix_z,wfb,win,INPUT,xin,Wout1,Wout2,FEEDBACK1,FEEDBACK2,Fold_Sp), tspan, ic,options);
toc
PxcL0 = YcL0(:,1:Ns);
PycL0 = YcL0(:,Ns+1:Ns+Ns);
PzcL0 = YcL0(:,2*Ns+1:2*Ns+Ns);

% find current spring folding angles, outputs
LcL0 = zeros(endT2,length(Fold_Sp));
for count = 1:endT2
NODES = NODES0fg+[PxcL0(count,:);PycL0(count,:);PzcL0(count,:)]';
phi = zeros(Ss,1);
for k = Fold_Sp
    [phi(k),~,~,~,~] = find_phi_dphi(NODES,CREASE(k,:));
end
LcL0(count,:) = phi(Fold_Sp);
end
%
Y1cL0=sum(Wout1(2:end).*LcL0')+Wout1(1);   % current output-1
Y2cL0=sum(Wout2(2:end).*LcL0')+Wout2(1);   % current output-2
x1L0 = x1s(1:endT2);
x2L0 = x2s(1:endT2);
figure
plot(time2,x1L0,time2,x2L0,time2,Y1cL0,time2,Y2cL0)
figure
plot(x1L0,x2L0,Y1cL0,Y2cL0)

%%
% save Quad_modulation_9b9_G_m0p01_3new.mat

%% all the related functions

function [NODES,SPRINGS,CREASE,Kcr,N_S,Left_Edge,Right_Edge,Bottom_Edge,Top_Edge,S_N] = ...
Miura_geometry(a,b,gamma,thetai,xN,yN)
[NODES,L,W,V,fa]=miura_geom(a,b,gamma,thetai,xN,yN);
x_max = max(NODES(:,1));
y_max = max(NODES(:,2));
Ncr = [a b fa];
Fcr = [fa];
Ns = length(NODES);
eps =  1E-4;
% Node Connections
N_N = zeros(Ns,Ns);
for k = 1:length(Ncr)
for i = 1:Ns
   for j = setdiff(1:Ns,1:i)
        if abs(norm(NODES(i,:)-NODES(j,:))-Ncr(k))<=eps
           N_N(i,j)=1;
        end
   end
end
```

```
end

% Spring definition
[Srow,Scol]=find(N_N);
SPRINGS = [Srow,Scol];
Ss = length(SPRINGS);

% Facet definition
Facet_Spring = zeros(Ss,1);
for k = 1:length(Fcr)
for i = 1:Ss
li(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));  % nominal length in flat state
if abs(li(i)-Fcr(k))<eps
    Facet_Spring(i)=1;
end
end
end
%Creases definition
    Left_Edge=[];
    Right_Edge=[];
    Bottom_Edge=[];
    Top_Edge=[];
for i = 1:Ns
    if abs(NODES(i,1)-0)<eps
    Left_Edge = [Left_Edge i];
    elseif abs(NODES(i,1)-x_max) <eps
    Right_Edge = [Right_Edge i];
    end
    if abs(NODES(i,2)-0)<eps
    Bottom_Edge = [Bottom_Edge i];
    elseif abs(NODES(i,2)-(y_max)) <eps
    Top_Edge = [Top_Edge i];
    end
end

[CREASE,Kcr,N_S,S_N] = get_creases(SPRINGS,Ss,Ns,Facet_Spring);

end

function [CREASE,Kcr,N_S,S_N] = get_creases(SPRINGS,Ss,Ns,Facet_Spring)
for j = 1:Ss
    if Facet_Spring(j)==1
         Kcr(j)= -1 ;
    else
         Kcr(j)= +1 ;
    end

        kka = ones(Ss,1);
        kkb = ones(Ss,1);
        for i = 1:Ss-1
            k = setdiff(1:Ss,j);
```

```
            [~,ia] = intersect(SPRINGS(k(i),:),SPRINGS(j,1));
            [~,ib] = intersect(SPRINGS(k(i),:),SPRINGS(j,2));
            kka(k(i)) = isempty(ia);
            kkb(k(i)) = isempty(ib);
        end
        arr = sort([SPRINGS(kka==0,1); SPRINGS(kka==0,2); SPRINGS(kkb==0,1); SPRINGS(kkb==0,2)]);
        [C,~,ic] = unique(arr);
        a_counts = accumarray(ic,1);
        value_counts = [C, a_counts];
        mp = setdiff(value_counts(:,1).*(a_counts==2),[SPRINGS(j,:) 0]);
        if length(mp)>=2
        CREASE(j,1:4) = [mp(1:2); SPRINGS(j,:)'];
        else
        CREASE(j,1:4) = zeros(1,4);
        Kcr(j) = 0;
        end

end

% Creating Node to Spring and Nodes Matrix
for i = 1:Ns
    for j = 1:Ss
    if ismember(i,SPRINGS(j,:))==1
N_S(i,j) = setdiff(SPRINGS(j,:),i);
    else
N_S(i,j) = 0;
    end
    end
end
% Kcr = 0 for boundary crease, +1 for M/V , -1 for facet
S_N = zeros(Ns,10);
for i = 1:Ns
 [~,Sco] = setdiff(N_S(i,:),0);
 S_N(i,1:length(Sco)) = Sco';
end
end
%----
function [phi,dphidpx,dphidpy,dphidpz,cosphi] = find_phi_dphi(NODES,CREASE)
Pl = NODES(CREASE(1),:);
Pi = NODES(CREASE(2),:);
Pj = NODES(CREASE(3),:);
Pk = NODES(CREASE(4),:);

m = cross((Pi-Pj),(Pk-Pj));
n = cross((Pk-Pj),(Pk-Pl));

cosphi = dot(m,n)/norm(n)/norm(m);
if dot(m,(Pk-Pl))==0
    phi = mod(1*real(acos(cosphi)),2*pi);
else
    phi = mod(sign(dot(m,(Pk-Pl)))*real(acos(cosphi)),2*pi);
```

```
end
dphidpl = -norm(Pk-Pj)/(norm(n))^2*n;
dphidpi = norm(Pk-Pj)/(norm(m))^2*m;
dphidpj = (dot((Pi-Pj),(Pk-Pj))/(norm(Pk-Pj))^2-1)*dphidpi - ...
dot((Pk-Pl),(Pk-Pj))/(norm(Pk-Pj))^2*dphidpl;
dphidpk = (dot((Pk-Pl),(Pk-Pj))/(norm(Pk-Pj))^2-1)*dphidpl - ...
dot((Pi-Pj),(Pk-Pj))/(norm(Pk-Pj))^2*dphidpi;

dphidpx = [dphidpl(1) dphidpi(1)  dphidpj(1)  dphidpk(1) ];
dphidpy = [dphidpl(2) dphidpi(2)  dphidpj(2)  dphidpk(2) ];
dphidpz = [dphidpl(3) dphidpi(3)  dphidpj(3)  dphidpk(3) ];
end

%-----
function [NODES,L,W,V,fa]=miura_geom(a,b,gamma,theta,xN,yN)
Ht = a*sin(gamma)*sin(theta);
L = b*tan(gamma)*cos(theta)./(sqrt(1+cos(theta).^2*tan(gamma)^2));
W = a*sqrt(1-sin(theta).^2*sin(gamma)^2);
V = b./sqrt(1+cos(theta).^2*tan(gamma)^2);

A = [0 0 0];
B = [L V 0];
C = [2*L 0 0];
D = [0 W Ht];
E = [L W+V Ht];
F = [2*L W Ht];
G = [0 2*W 0];
H = [L 2*W+V 0];
I = [2*L 2*W 0];
NODESb = [A; B; C; D; E; F; G; H; I];
% Add periodic copies of miura in x and y direction
xNodes = [repmat(B,xN,1); repmat(C,xN,1); repmat(E,xN,1); repmat(F,xN,1); ...
repmat(H,xN,1); repmat(I,xN,1)]+repmat([[1:xN]'*2*L zeros(xN,2)],6,1);
yNodes = [];
for i = 1:yN
yNodesm = [D; G; E; H; F; I; xNodes((2*xN+1):end,:)]+repmat([0 i*2*W 0],6+length(xNodes)-(2*xN),1);
yNodes = [yNodes; yNodesm];
end
NODES = [NODESb;xNodes;yNodes];
fa = sqrt(a^2+b^2-2*a*b*cos(pi-gamma));
end

%-----
function [FEEDBACK1,FEEDBACK2,INPUT,op_N,Ns,Ss,Srow,Scol,Fold_Sp,CREASE] = ...
input_para(NODES0,SPRINGS,CREASEi,Kcr,a,Ncr)
Ns = length(NODES0);
Ss = length(SPRINGS);
Srow = SPRINGS(:,1);
Scol = SPRINGS(:,2);
li = zeros(Ss,1);
for i = 1:Ss
```

```
li(i) = norm(NODES0(SPRINGS(i,1),:)-NODES0(SPRINGS(i,2),:));  % nominal length in flat state
end

AA = (1:Ss).*any(CREASEi');
%correcting mountain-valley assignment: dist is negative then mountain ;
%dist is positive then valley
% phi>pi for mountain;  phi<pi for valley
CREASE = CREASEi;
for j = find(AA)
% CREASE(j,:)
mid1 = (NODES0(CREASEi(j,1),:)+NODES0(CREASEi(j,2),:))/2; % facet
mid2 = (NODES0(CREASEi(j,3),:)+NODES0(CREASEi(j,4),:))/2; % crease
[phi,~,~,~,~] = find_phi_dphi(NODES0,CREASEi(j,:));
if abs(li(j)-a)<10^-6
dist = mid1(3)-mid2(3);
if dist<0 && phi<pi || dist>0 && phi>pi
    CREASE(j,1:4) = CREASEi(j,1:4);
elseif dist<0 && phi>pi || dist>0 && phi<pi
    CREASE(j,1:2) = fliplr(CREASEi(j,1:2));
end
else
dist = mid1(3)-mid2(3);
if dist>0 && phi<pi || dist<0 && phi>pi
    CREASE(j,1:4) = CREASEi(j,1:4);
elseif dist>0 && phi>pi || dist<0 && phi<pi
    CREASE(j,1:2) = fliplr(CREASEi(j,1:2));
end
end
end

% figure
% Gr = graph(SPRINGS(:,1), SPRINGS(:,2),phi0);
% plot(Gr,'EdgeLabel',Gr.Edges.Weight)

Fold_Sp = intersect(find(Kcr==1),AA);
y_cr = [];
for k = 1:length(Ncr)
for i = Fold_Sp
    if abs(li(i)-Ncr(k))<10^-5
        long_crk = i;
    else
        long_crk = [];
    end
    y_cr = [y_cr long_crk];
end
end
y_c = unique(y_cr);
op_N = length(Fold_Sp);
fbs = y_c(randperm(length(y_c),ceil(op_N*45/100)));

FEEDBACK1 = fbs(1:ceil(1*length(fbs)/3));
```

```
FEEDBACK2 = fbs(1*ceil(length(fbs)/3)+1:ceil(2*length(fbs)/3));
INP = fbs(ceil(2*length(fbs)/3)+1:end);
INPUT = INP(1:10);
% INT = setdiff(fbs,FB);
% INPUT = INT(randperm(length(INT),17));
end

%-----
function [dYdt] = myode_gravity(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0i,Kaxial,Kcrease,CREASE, ...
NODES0,zeta,m,Fix_x,Fix_y,Fix_z)
Px = Y(1:Ns);
Py = Y(Ns+1:Ns+Ns);
Pz = Y(2*Ns+1:2*Ns+Ns);
vx = Y(3*Ns+1:3*Ns+Ns);
vy = Y(4*Ns+1:4*Ns+Ns);
vz = Y(5*Ns+1:5*Ns+Ns);

NODES = NODES0+[Px Py Pz];
l = zeros(Ss,1);
for i = 1:Ss
l(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));
end
Faxialx=zeros(Ss,Ns);
Faxialy=zeros(Ss,Ns);
Faxialz=zeros(Ss,Ns);
Fcreasex=zeros(Ss,Ns);
Fcreasey=zeros(Ss,Ns);
Fcreasez=zeros(Ss,Ns);

for j = 1:Ss
[NSco] = setdiff(N_S(:,j),0);
% Axial Force Calculations
dldpi = (NODES(NSco(2),:)-NODES(NSco(1),:))/norm(NODES(NSco(2),:)-NODES(NSco(1),:));
Faxialx(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(1) dldpi(1)];
Faxialy(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(2) dldpi(2)];
Faxialz(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(3) dldpi(3)];
phi0 = phi0i;
phi = phi0i;
end
AA = (1:Ss).*any(CREASE');
for j = find(AA)
% Crease folding force Calculations
    [phi(j),dphidpx,dphidpy,dphidpz,~] = find_phi_dphi(NODES,CREASE(j,:));
    Fcreasex(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpx;
    Fcreasey(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpy;
    Fcreasez(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpz;
end

% Damping Force Calculations
Fdampingx = zeros(1,Ns);
Fdampingy = zeros(1,Ns);
```

```
Fdampingz = zeros(1,Ns);
for i = 1:Ns
[Nco,Sco] = setdiff(N_S(i,:),0);
dVdampx(1:length(Nco)) = (vx(Nco)-vx(i))/2;
dVdampy(1:length(Nco)) = (vy(Nco)-vy(i))/2;
dVdampz(1:length(Nco)) = (vz(Nco)-vz(i))/2;
cdamp = 2*zeta*sqrt(Kaxial./l(Sco)*m(i));

Fdampingx(i) = (sum(cdamp.*dVdampx'));
Fdampingy(i) = (sum(cdamp.*dVdampy'));
Fdampingz(i) = (sum(cdamp.*dVdampz'));
clear dVdampx dVdampy dVdampz
end

% Add gravity
Fg = -9.81*m;
% Calculating New Velocity and Position
Fx = sum(Faxialx)+sum(Fcreasex)+(Fdampingx);
Fy = sum(Faxialy)+sum(Fcreasey)+(Fdampingy);
Fz = sum(Faxialz)+sum(Fcreasez)+(Fdampingz)+Fg;

ax = (Fx)./m;
ay = (Fy)./m;
az = (Fz)./m;

ax(Fix_x) = 0;
ay(Fix_y) = 0;
az(Fix_z) = 0;

dYdt = [Y(3*Ns+1:3*Ns+3*Ns); [ax ay az]'];

end

%----
function [x1,x2] = reference(time,ic,eps)
options = odeset('RelTol',10^-12,'AbsTol',10^-12);
% tspan = time;
% [t,y]=ode45(@(t,y) VanderPol(t,y), tspan, ic,options);
% x1 = y(:,1);
% x2 = y(:,2);
%
% function dydt = VanderPol(t,y)
%     dydt(1,1) = y(2);
%     dydt(2,1) = -y(1)+(1-y(1)^2)*y(2)*1;
% end
%------
tspan = time;
[~,y]=ode45(@(t,y) QuadraticLC(t,y), tspan, ic,options);
x1 = y(:,1);
x2 = y(:,2);
```

```
    function dydt = QuadraticLC(t,y)
        dydt(1,1) = y(1)+y(2)-eps(t)*y(1)*(y(1)^2+y(2)^2);
        dydt(2,1) = -2*y(1)+y(2)-y(2)*(y(1)^2+y(2)^2);
    end

% %------
% f1 = 1;
% f2 = 2;
% delta = pi/2;
% x1 = f2*sin(2*pi*f1*time+delta)';
% x2 = f2*sin(2*pi*f2*time)';

end

%-----
function [dYdt] = myode(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0i,Kaxial,Kcrease,CREASE,NODES0,zeta,m, ...
win,INPUT,xin,Fix_x,Fix_y,Fix_z,wfb,x1,x2,FEEDBACK1,FEEDBACK2,time)
Px = Y(1:Ns);
Py = Y(Ns+1:Ns+Ns);
Pz = Y(2*Ns+1:2*Ns+Ns);
vx = Y(3*Ns+1:3*Ns+Ns);
vy = Y(4*Ns+1:4*Ns+Ns);
vz = Y(5*Ns+1:5*Ns+Ns);

NODES = NODES0+[Px Py Pz];
l = zeros(Ss,1);
for i = 1:Ss
l(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));
end
Faxialx=zeros(Ss,Ns);
Faxialy=zeros(Ss,Ns);
Faxialz=zeros(Ss,Ns);
for j = 1:Ss
[NSco] = setdiff(N_S(:,j),0);
% Axial Force Calculations
dldpi = (NODES(NSco(2),:)-NODES(NSco(1),:))/norm(NODES(NSco(2),:)-NODES(NSco(1),:));
Faxialx(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(1) dldpi(1)];
Faxialy(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(2) dldpi(2)];
Faxialz(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(3) dldpi(3)];
end
Fcreasex=zeros(Ss,Ns);
Fcreasey=zeros(Ss,Ns);
Fcreasez=zeros(Ss,Ns);
phi = zeros(Ss,1);
phi0 = phi0i;
fb1 = interp1(time,x1',t);
fb2 = interp1(time,x2',t);
phi0(FEEDBACK1) = wfb(FEEDBACK1)*tanh(fb1)+phi0i(FEEDBACK1);
phi0(FEEDBACK2) = wfb(FEEDBACK2)*tanh(fb2)+phi0i(FEEDBACK2);
phi0(INPUT) = win(INPUT)*tanh(xin(t))+phi0i(INPUT);
AA = (1:Ss).*any(CREASE');
```

```
for j = find(AA)
% Crease folding force Calculations
    [phi(j),dphidpx,dphidpy,dphidpz,~] = find_phi_dphi(NODES,CREASE(j,:));
    Fcreasex(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpx;
    Fcreasey(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpy;
    Fcreasez(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpz;
end

% Damping Force Calculations
Fdampingx = zeros(1,Ns);
Fdampingy = zeros(1,Ns);
Fdampingz = zeros(1,Ns);
for i = 1:Ns
[Nco,Sco] = setdiff(N_S(i,:),0);

dVdampx(1:length(Nco)) = (vx(Nco)-vx(i))/2;
dVdampy(1:length(Nco)) = (vy(Nco)-vy(i))/2;
dVdampz(1:length(Nco)) = (vz(Nco)-vz(i))/2;
cdamp = 2*zeta*sqrt(Kaxial./l(Sco)*m(i));

Fdampingx(i) = (sum(cdamp.*dVdampx'));
Fdampingy(i) = (sum(cdamp.*dVdampy'));
Fdampingz(i) = (sum(cdamp.*dVdampz'));
clear dVdampx dVdampy dVdampz
end

% Calculating New Velocity and Position
Fx = sum(Faxialx)+sum(Fcreasex)+(Fdampingx);
Fy = sum(Faxialy)+sum(Fcreasey)+(Fdampingy);
Fz = sum(Faxialz)+sum(Fcreasez)+(Fdampingz);
ax = (Fx)./m;
ay = (Fy)./m;
az = (Fz)./m;
ax(Fix_x) = 0;
ay(Fix_y) = 0;
az(Fix_z) = 0;
dYdt = [Y(3*Ns+1:3*Ns+3*Ns); [ax ay az]'];
% size(dYdt)

end

%-----
function [dYdt] = myodeC(t,Y,Ss,Ns,SPRINGS,N_S,li,phi0i,Kaxial,Kcrease,CREASE,NODES0,zeta,m, ...
Fix_x,Fix_y,Fix_z,wfb,win,INPUT,xin,wout1,wout2,FEEDBACK1,FEEDBACK2,Fold_Sp)
Px = Y(1:Ns);
Py = Y(Ns+1:Ns+Ns);
Pz = Y(2*Ns+1:2*Ns+Ns);
vx = Y(3*Ns+1:3*Ns+Ns);
vy = Y(4*Ns+1:4*Ns+Ns);
vz = Y(5*Ns+1:5*Ns+Ns);
```

```
NODES = NODES0+[Px Py Pz];
l = zeros(Ss,1);
for i = 1:Ss
l(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));
end
Faxialx=zeros(Ss,Ns);
Faxialy=zeros(Ss,Ns);
Faxialz=zeros(Ss,Ns);
Fcreasex=zeros(Ss,Ns);
Fcreasey=zeros(Ss,Ns);
Fcreasez=zeros(Ss,Ns);

phi = zeros(Ss,1);
for j = Fold_Sp
    [phi(j),~,~,~,~] = find_phi_dphi(NODES,CREASE(j,:));
end
L = phi(Fold_Sp) ;
Y1=sum(wout1(2:end).*L)+wout1(1);  % current output-1
Y2=sum(wout2(2:end).*L)+wout2(1);  % current output-2

for j = 1:Ss
[NSco] = setdiff(N_S(:,j),0);
% Axial Force Calculations
dldpi = (NODES(NSco(2),:)-NODES(NSco(1),:))/norm(NODES(NSco(2),:)-NODES(NSco(1),:));
Faxialx(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(1) dldpi(1)];
Faxialy(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(2) dldpi(2)];
Faxialz(j,NSco) = -Kaxial/l(j)*(l(j)-li(j))*[-dldpi(3) dldpi(3)];
end
phi0 = phi0i;
phi0(FEEDBACK1) = wfb(FEEDBACK1)*tanh(Y1)+phi0i(FEEDBACK1);
phi0(FEEDBACK2) = wfb(FEEDBACK2)*tanh(Y2)+phi0i(FEEDBACK2);
phi0(INPUT) = win(INPUT)*tanh(xin(t))+phi0i(INPUT);
AA = (1:Ss).*any(CREASE');
for j = find(AA)
% Crease folding force Calculations
    [phi(j),dphidpx,dphidpy,dphidpz,~] = find_phi_dphi(NODES,CREASE(j,:));
    Fcreasex(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpx;
    Fcreasey(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpy;
    Fcreasez(j,CREASE(j,:)) = -Kcrease(j)*l(j)*(phi(j)-phi0(j))*dphidpz;
end

% Damping Force Calculations
Fdampingx = zeros(1,Ns);
Fdampingy = zeros(1,Ns);
Fdampingz = zeros(1,Ns);
for i = 1:Ns
[Nco,Sco] = setdiff(N_S(i,:),0);

dVdampx(1:length(Nco)) = (vx(Nco)-vx(i))/2;
dVdampy(1:length(Nco)) = (vy(Nco)-vy(i))/2;
dVdampz(1:length(Nco)) = (vz(Nco)-vz(i))/2;
```

```
cdamp = 2*zeta*sqrt(Kaxial./l(Sco)*m(i));

Fdampingx(i) = (sum(cdamp.*dVdampx'));
Fdampingy(i) = (sum(cdamp.*dVdampy'));
Fdampingz(i) = (sum(cdamp.*dVdampz'));
clear dVdampx dVdampy dVdampz
end

% Calculating New Velocity and Position
Fx = sum(Faxialx)+sum(Fcreasex)+(Fdampingx);
Fy = sum(Faxialy)+sum(Fcreasey)+(Fdampingy);
Fz = sum(Faxialz)+sum(Fcreasez)+(Fdampingz);

ax = (Fx)./m;
ay = (Fy)./m;
az = (Fz)./m;
ax(Fix_x) = 0;
ay(Fix_y) = 0;
az(Fix_z) = 0;

dYdt = [Y(3*Ns+1:3*Ns+3*Ns); [ax ay az]'];

end
```

C.3 Code for crease pattern design generation for Quadruped Robot

```
clc
clear
% Input parameters for Miura-ori based design (in mm)
b = 10; % quadrilateral facet crease length-1
a = 1.1*b; % quadrilateral facet crease length-2
gamma = 70*pi/180; % angle between the Miura-ori quadrilateral facets
xN1 = 3; % nodes in x direction should be odd number for kinematic compatibility with both legs
yN1 = 4; % nodes in y direction
% Arc-Miura design for quadruped leg
l1 = 8; % quadrilateral facet crease length-1
b1 = b; % quadrilateral facet crease length-2
gamma1 = gamma; % angle-1 between the Miura-ori quadrilateral facets
gamma2 = 50*pi/180; % angle-2 between the Miura-ori quadrilateral facets
xN2 = 5; % should be odd for matching left and right leg
yN2 = yN1;
th_Ai = pi;
xN3 = xN1; % miura nodes in the bridge
yN3 = 3; % minimum 3 nodes required for kinematic compatibility

Ns11 = (xN2-1)*yN2;
Ns1 = xN1*yN1;
Ns21 = (xN2-1)*yN2;
Nsf1 = Ns11+Ns1+Ns21;
Nsf2 = xN3*yN3;
```

```
Nsf3 = Ns11+Ns1+Ns21;
% initial geometry
[NODESi1,x_max1,y_max1,vi,fa,vzi] = miura_original(a,b,gamma,xN1,yN1,th_Ai);
[NODESi2,l2,b1,b2,f_l,v1,v2,x_max2,y_max2] = Arc_Miura(l1,b1,gamma1,gamma2,xN2,yN2);
[NODESi3,x_max3,y_max3,vi3,fa3,vzi3] = miura_original(a,b,gamma,xN3,yN3,th_Ai); % middle bridge
% Assembling arc-miura, miura and arc-miura to form 2 legs
x_movei = x_max1-vi;
NODESi21 = NODESi2(yN2+1:end,:);
NODESi21(:,1) = NODESi2(yN2+1:end,1)+x_movei;
NODESi11 = NODESi2(1:(xN2-1)*yN2,:);
if mod(yN1,2)==0
    NODESi11(:,1) = NODESi2(1:(xN2-1)*yN2,1)-NODESi2(end,1)+v1;
else
    NODESi11(:,1) = NODESi2(1:(xN2-1)*yN2,1)-NODESi2(end,1);
end
NODESi = [NODESi11;NODESi1;NODESi21];
NODESif1 = NODESi;
NODESif1(:,1) = NODESi(:,1)-NODESi11(1,1); % left side
% Assembling other side
NODESif2 = NODESi3;
NODESif2(:,1) = NODESi3(:,1)+x_max2-v1;
NODESif2(:,2) = (NODESi3(:,2))+y_max1+y_max3/2; % middle bridge

NODESif3 = NODESif1;
NODESif3(:,2) = flip(NODESif1(:,2))+y_max1+2*y_max3; % right side

NODESif = [NODESif1; NODESif2; NODESif3;];
x_max = max(NODESif(:,1));
y_max = max(NODESif(:,2));

Ncr = [a l1 l2 b b1 b2 fa f_l]; % truss element lengths to be selecetd for crease folding
Fcr = [fa f_l]; % truss element lengths to be selecetd for facet bending
Ecrx = [0 vi  x_max x_max-v2 x_max-v1]; % left and right edge x-dirn
Ecry = [0 y_max]; % bottom and top edge y-dirn
Mcrx = [x_max2-vi x_max2 x_max2-vi+x_max3 x_max2-vi+x_max3-vi3]; % middle edges x-dirn
Mcry = [y_max2 y_max2+2*y_max3]; % middle edges y-dirn
center = 0;
% generating crease to node matrix, dihedral angle matrix, truss element
% assignment matrix, facet and crease assignment matrix ,and assigning left, right, top, and bottom
[SPRINGS,CREASEi,Kcr,N_S,LE,RE,BE,TE,S_N,Ss,Ns,Srow,Scol] = ...
Reservoir_geometry(NODESif,Ncr,Fcr,Ecrx,Ecry,x_max,y_max,Mcrx,Mcry,Nsf1,Nsf2,xN3,yN1,center);
AA = (1:Ss).*any(CREASEi');
k = 1;
for j = find(AA)
facets(k,:) = CREASEi(j,[1 3 4]);
facets(k+1,:) = CREASEi(j,[2 3 4]);
k = k+2;
end
% folded configuration
Ss1 = (3*yN2-2)*(xN2-1)+yN2-1;
Ss2 = (3*yN1-2)*(xN1-1);
```

```
Ss3 = (3*yN2-2)*(xN2-1);
Ss4 = (3*yN3-1)*(xN3-1)+yN3+2;
% CREASE(Ss1+Ss2+Ss3+Ss4:end,1:2) = fliplr(CREASE(Ss1+Ss2+Ss3+Ss4:end,1:2));
th_A = [0.99 0.9 0.8 0.6 0.4]*pi;
figure
for k = 3
[NODES01,x_max01,y_max01,v01,fa,vz0] = miura_original_fold(a,b,gamma,xN1,yN1,th_A(k));
[NODES021,~,~] = arc_miura_pattern_fold(l2,b1,l1,gamma1,gamma2,xN2,yN2,th_A(k),1);
[NODES011,~,~] = arc_miura_pattern_fold(l2,b1,l1,gamma1,gamma2,xN2,yN2,th_A(k),2);
[NODES03,x_max03,y_max03,v03,fa3,vz03] = miura_original_fold(a,b,gamma,xN3,yN3,th_A(k));
% Assembling arc-miura, miura and arc-miura to form 2 legs
x_movef = x_max01-v01;
z_movef = vz0;
NODES01f =  NODES01(yN1+1:(xN1-1)*yN1,:);
NODES01f(:,1) = NODES01(yN1+1:(xN1-1)*yN1,1)-NODES011(1,1);
NODES021f = NODES021;
NODES021f(:,1) = NODES021(:,1)+x_movef-NODES011(1,1);
NODES021f(:,3) = NODES021(:,3)+z_movef;
NODES011f = NODES011;
if mod(yN1,2)==0
    NODES011f(:,1) = NODES011(:,1)-NODES011(1,1)+v01;
else
    NODES011f(:,1) = NODES011(:,1)-NODES011(1,1);
end
NODES011f(:,3) = NODES011(:,3)-NODES011(end,3)+z_movef;
NODES0 = [NODES011f;NODES01f;NODES021f];
NODES0f1 = NODES0;
NODES0f1(:,1) = NODES0(:,1)-NODES0(1,1);
NODES0f1(:,3) = NODES0(:,3)-NODES0(1,3);

% Assembling other side
NODES0f2 = NODES03;
NODES0f2(:,1) = NODES03(:,1)-NODES011(1,1)-v01;
NODES0f2(:,2) = (NODES03(:,2))+2*y_max03;
NODES0f2(:,3) = NODES03(:,3)-NODES0(1,3); % middle bridge

NODES0f3 = NODES0f1;
NODES0f3(:,2) = flip(NODES0f1(:,2))+y_max01+2*y_max03; % right side
NODES0f = [NODES0f1; NODES0f2; NODES0f3;];
% plots
for i = 1:Ss
fill3([NODES0f(Srow(i),1) NODES0f(Scol(i),1)],[NODES0f(Srow(i),2) ...
NODES0f(Scol(i),2)],[NODES0f(Srow(i),3) NODES0f(Scol(i),3)],[0.5 0],'EdgeColor','black')
hold on
end
title(['Folding Angle = ',num2str(th_A(k)*180/pi)])
end
axis 'equal'
grid on
xlabel('x')
ylabel('y')
```

```
zlabel('z')

NODES0f(:,3) = NODES0f(:,3)+1;  % Raising the origami above ground

function [NODESi,x_max,y_max,v,fa,vz] = miura_original(a,b,gamma,xN,yN,th_A)
et_Z = acos(sin(gamma)^2*cos(th_A)+cos(gamma)^2);
et_A = acos(1-(4*cos(gamma)^2/(1+cos(et_Z))));
k = 1;
for i = 1:xN
    for j = 1:yN
      if mod(j,2)==0
        x = (i-1)*a*sin(et_A/2)+b*cos(et_Z/2);
      else
        x = (i-1)*a*sin(et_A/2);
      end
y = (j-1)*b*sin(et_Z/2);
      if mod(i,2)==0
        z = a*cos(et_A/2);
      else
        z = 0;
      end
      NODESi(k,:) = [x y z];
      k = k+1;
    end
end
x_max = max(NODESi(:,1));
y_max = max(NODESi(:,2));
v = b*cos(et_Z/2);
vz = a*cos(et_A/2);
fa = sqrt(a^2+b^2-2*a*b*cos(pi-gamma));
end

function [NODES0,x_max,y_max,v,fa,vz] = miura_original_fold(a,b,gamma,xN,yN,th_A)
et_Z = acos(sin(gamma)^2*cos(th_A)+cos(gamma)^2);
et_A = acos(1-(4*cos(gamma)^2/(1+cos(et_Z))));
k = 1;
for i = 1:xN
    for j = 1:yN
      if mod(j,2)==0
        x = (i-1)*a*sin(et_A/2)+b*cos(et_Z/2);
      else
        x = (i-1)*a*sin(et_A/2);
      end
y = (j-1)*b*sin(et_Z/2);
      if mod(i,2)==~0
        z = a*cos(et_A/2);
      else
        z = 0;
      end
      NODES0(k,:) = [x y z];
      k = k+1;
```

```
    end
end
x_max = max(NODES0(:,1));
y_max = max(NODES0(:,2));
v = b*cos(et_Z/2);
vz = NODES0((xN)*yN,3);
fa = sqrt(a^2+b^2-2*a*b*cos(pi-gamma));
end

function [NODES,l2,b1,b2,f_l,v1,v2,x_max,y_max] = Arc_Miura(l1,b1,gamma1,gamma2,xN,yN)
b2 = b1*sin(gamma1)/sin(gamma2);
w = b2*sin(gamma2);
v2 = b2*cos(gamma2);
v1 = b1*cos(gamma1);

l2 = l1+b2*cos(gamma2)-b1*cos(gamma1);
f_l = sqrt(l2^2+b1^2-2*l2*b1*cos(pi-gamma1));
f_min = sqrt(l2^2+b2^2-2*l2*b2*cos(gamma2));
a1 = l1+l2;

k = 1;
for i = 1:xN
    for j = 1:yN
        if mod(i,2)==~0 && mod(j,2)==~0
        x = (i-1)/2*a1;
        elseif  mod(i,2)==0 && mod(j,2)==~0
        x = i/2*a1-l2;
        elseif mod(i,2)==0 && mod(j,2)==0
        x = v1 + i/2*a1-l1;
        elseif mod(i,2)==~0 && mod(j,2)==0
        x = v1 + (i-1)/2*a1;
        end
        y = (j-1)*w;
        z = 0;
        NODES(k,:) = [x y z];
        k = k+1;
    end
end
%
x_max = max(NODES(:,1));
y_max = (yN-1)*w;

% figure
% plot(NODES(:,1),NODES(:,2),'o-')
% grid on
end

function [NODES0n,x_max,y_max] = arc_miura_pattern_fold(l2,b1,l1,gamma1,gamma2,xN,yN,th_A,ii)
cos_et_VZ = sin(gamma2)^2*cos(th_A)+cos(gamma2)^2;
et_VA = acos(1-4*cos(gamma2)^2/(1+cos_et_VZ));
cos_et_MZ = sin(gamma1)^2*cos(th_A)+cos(gamma1)^2;
```

```
et_MZ = acos(cos_et_MZ);
et_MA = acos(1-4*cos(gamma1)^2/(1+cos_et_MZ));

zeta = et_VA-et_MA;
R1 = sqrt((l1^2+l2^2-2*l1*l2*cos(et_VA))/(2*(1-cos(zeta))));
R2 = sqrt((l1^2+l2^2-2*l1*l2*cos(et_MA))/(2*(1-cos(zeta))));
zetaa2 = acos((R1^2+R2^2-l2^2)/(2*R1*R2));
zetab1 = acos((2*R1^2-b1^2*cos(et_MZ/2)^2)/(2*R1^2));
k = 1;

for i = xN
    for j = yN
        if mod(j,2)==~0 && mod(i,2)==~0
            thmax = (i-1)*(zeta)/2;
        elseif mod(j,2)==0 && mod(i,2)==~0
            thmax = (i-1)*(zeta)/2+zetab1;
        elseif mod(j,2)==~0 && mod(i,2)==0
            thmax = (i-2)*(zeta)/2+zeta-zetaa2;
        elseif mod(j,2)==0 && mod(i,2)==0
            thmax = (i-2)*(zeta)/2+zetab1+zetaa2;
        end
    end
end
for i = 1:xN
    for j = 1:yN
        if mod(i,2)==~0
            r = R1;
        else
            r = R2;
        end

        if mod(j,2)==~0 && mod(i,2)==~0
            th = (i-1)*(zeta)/2;
        elseif mod(j,2)==0 && mod(i,2)==~0
            th = (i-1)*(zeta)/2+zetab1;
        elseif mod(j,2)==~0 && mod(i,2)==0
            th = (i-2)*(zeta)/2+zeta-zetaa2;
        elseif mod(j,2)==0 && mod(i,2)==0
            th = (i-2)*(zeta)/2+zetab1+zetaa2;
        end
y = (j-1)*b1*sin(et_MZ/2);
if ii==1
z = r*cos(th);
x = r*sin(th);
elseif ii==2
z = r*cos(th-thmax);
x = r*sin(th-thmax);
end
NODES0(k,:) = [x y z];
k = k+1;
    end
```

```
end
NODES0(:,3) = NODES0(:,3)-NODES0(1,3);
NODES0n = NODES0;
x_max = max(NODES0n(:,1));
y_max = max(NODES0n(:,2));

end

function [SPRINGS,CREASE,Kcr,N_S,Left_Edge,Right_Edge,Bottom_Edge,Top_Edge,S_N,Ss,Ns,Srow,Scol] = ..
Reservoir_geometry(NODES,Ncr,Fcr,Ecrx,Ecry,x_max,y_max,Mcrx,Mcry,Nsf1,Nsf2,xN3,yN1,center)
Ns = length(NODES);
eps =  1E-4;
% Node Connections
N_N = zeros(Ns,Ns);
for k = 1:length(Ncr)
for i = 1:Ns
   for j = setdiff(1:Ns,1:i)
        if abs(norm(NODES(i,:)-NODES(j,:))-Ncr(k))<=eps
           N_N(i,j)=1;
        end
        if i == 40 && j == 67 || i == 69 && j == 109 %for 7 y nodes in leg
%         if i == 32 && j == 51 || i == 53 && j == 85 %for 5 y nodes in leg
            N_N(i,j)=0;
        end
   end
end
end

% Spring definition
[Srow,Scol]=find(N_N);
SPRINGS = [Srow,Scol];
Ss = length(SPRINGS);

% Facet definition
Facet_Spring = zeros(Ss,1);
for k = 1:length(Fcr)
for i = 1:Ss
li(i) = norm(NODES(SPRINGS(i,1),:)-NODES(SPRINGS(i,2),:));  % nominal length in flat state
if abs(li(i)-Fcr(k))<eps
    Facet_Spring(i)=1;
end
end
end
%Creases definition
    Left_Edge=[];
    Right_Edge=[];
    Bottom_Edge=[];
    Top_Edge=[];
for i = 1:Ns
    if abs(NODES(i,1)-0)<eps
    Left_Edge = [Left_Edge i];
```

```
    elseif abs(NODES(i,1)-x_max) <eps
    Right_Edge = [Right_Edge i];
    end
    if abs(NODES(i,2)-0)<eps
    Bottom_Edge = [Bottom_Edge i];
    elseif abs(NODES(i,2)-(y_max)) <eps
    Top_Edge = [Top_Edge i];
    end
end

[CREASE,Kcr,N_S,S_N] = get_creases(SPRINGS,Ss,Ns,Facet_Spring,NODES,center);
end

function [CREASE,Kcr,N_S,S_N] = get_creases(SPRINGS,Ss,Ns,Facet_Spring,NODES,center)
for j = 1:Ss
        kka = ones(Ss,1);
        kkb = ones(Ss,1);
        for i = 1:Ss-1
            k = setdiff(1:Ss,j);
            [~,ia] = intersect(SPRINGS(k(i),:),SPRINGS(j,1));
            [~,ib] = intersect(SPRINGS(k(i),:),SPRINGS(j,2));
            kka(k(i)) = isempty(ia);
            kkb(k(i)) = isempty(ib);
        end
        arr = ([SPRINGS(kka==0,1); SPRINGS(kka==0,2); SPRINGS(kkb==0,1); SPRINGS(kkb==0,2)]);
        [C,~,ic] = unique(arr);
        a_counts = accumarray(ic,1);
        value_counts = [C, a_counts];
        mp = setdiff(value_counts(:,1).*(a_counts==2),[SPRINGS(j,:) 0]);
        if length(mp)>=2
        CREASE(j,1:4) = [mp(1:2); SPRINGS(j,:)'];
        else
        CREASE(j,1:4) = zeros(1,4);
        Kcr(j) = 0;
        end
    if Facet_Spring(j)==1
         Kcr(j)= -1 ;
    else
         Kcr(j)= +1 ;
    end

end

% Creating Node to Spring and Nodes Matrix
for i = 1:Ns
    for j = 1:Ss
    if ismember(i,SPRINGS(j,:))==1
N_S(i,j) = setdiff(SPRINGS(j,:),i);
    else
N_S(i,j) = 0;
    end
```

```
    end
end
% Kcr = 0 for boundary / undriven crease, +1 for M/V , -1 for facet
S_N = zeros(Ns,10);
for i = 1:Ns
 [~,Sco] = setdiff(N_S(i,:),0);
 S_N(i,1:length(Sco)) = Sco';
end
end

function [FEEDBACK1,FEEDBACK2,FEEDBACK3,FEEDBACK4,IN,Kaxial,Kcrease,op_N, ...
phi0,dt,Ns,Ss,m,Fold_Sp,li,CREASE] = ...
input_para(NODES0,SPRINGS,LE,BE,RE,TE,CREASEi,Kcr,Ncr,Ss1,Ss2,Ss3,Ss4,center)
Ns = length(NODES0);
Ss = length(SPRINGS);

for i = 1:Ss
li(i) = norm(NODES0(SPRINGS(i,1),:)-NODES0(SPRINGS(i,2),:));  % nominal length in flat state
end
AA = (1:Ss).*any(CREASEi');
Fold_Sp = intersect(find(Kcr==1),AA);
y_cr = [];
for k = 1:length(Ncr)
for i = Fold_Sp
    if abs(li(i)-Ncr(k))<10^-5
        long_crk = i;
    else
        long_crk = [];
    end
    y_cr = [y_cr long_crk];
end
end
y_c = unique(y_cr);
op_N = length(Fold_Sp)

fbs1 = intersect(y_c,1:Ss1);
fbs2 = intersect(y_c,Ss1+Ss2+Ss3+Ss4+1:Ss1+Ss2+Ss3+Ss4+Ss1+2);
fbs3 = intersect(y_c,Ss1+floor(2*Ss2/3)+1:Ss1+Ss2+Ss3);
fbs4 = intersect(y_c,Ss1+Ss2+Ss3+Ss4+Ss1+floor(2*Ss2/3)+1:Ss)
FEEDBACK1 = [fbs1(1:2:4) fbs1(7:2:9)]
FEEDBACK2 = [fbs2(1:2:4) fbs2(7:2:9)]
FEEDBACK3 = [fbs3(1+3:2:4+3) fbs3(7+3:2:9+3)]
FEEDBACK4 = [fbs4(6:2:9) fbs4(7+5:2:9+5)]
% FEEDBACK1 = fbs1(randperm(length(fbs1),ceil(op_N*5/100)))
% FEEDBACK2 = fbs2(randperm(length(fbs2),ceil(op_N*5/100)))
% FEEDBACK3 = fbs3(randperm(length(fbs3),ceil(op_N*5/100)))
% FEEDBACK4 = fbs4(randperm(length(fbs4),ceil(op_N*5/100)))
ins = [48    50    53   108   114   116   174   176   178  181]
% IN = ins(randperm(length(ins),ceil(op_N*5/100)));
IN = ins;
FB = [FEEDBACK1 FEEDBACK2 FEEDBACK3 FEEDBACK4];
```

```
% setting spring stiffnesses; Material constants
EA = 1000;
m = 0.01*ones(Ns,1);
Kaxial = EA*ones(Ss,1);
Kmin = 0.005;
Kmax = 1;
Kfold = 2;
Kfacet =100;
Kcrease(FB) = 10;
% wmax = max(sqrt(Kaxial./m));
dt = 1;
%correcting mountain-valley assignment: dist is negative then mountain ;
%dist is positive then valley
% phi>pi for mountain;  phi<pi for valley
CREASE = CREASEi;
for j = find(AA)
% CREASE(j,:)
mid1 = (NODES0(CREASEi(j,1),:)+NODES0(CREASEi(j,2),:))/2; % facet
mid2 = (NODES0(CREASEi(j,3),:)+NODES0(CREASEi(j,4),:))/2; % crease
rad1 = norm(mid1-[center(1) center(2)+mid1(2) center(3)]);
rad2 = norm(mid2-[center(1) center(2)+mid2(2) center(3)]);
dist = rad1 - rad2;
[phi,~,~,~,~] = find_phi_dphi(NODES0,CREASEi(j,:));
if dist>0 && phi<pi || dist<0 && phi>pi
    CREASE(j,1:4) = CREASEi(j,1:4);
elseif dist>0 && phi>pi || dist<0 && phi<pi
    CREASE(j,1:2) = fliplr(CREASEi(j,1:2));
end
end

phi0 = zeros(Ss,1);
Kcrease = zeros(Ss,1);
for i = find(AA)
   [phi0(i),~,~,~,~] = find_phi_dphi(NODES0,CREASE(i,:),phi0(i));  % target fold angle
%---
if Kcr(i)==1
    Kcrease(i) = Kfold;
elseif Kcr(i)==-1
    Kcrease(i) = Kfacet;
end
end
% Kcrease(FEEDBACK1) = 5*10^-3*(Kaxial);
end

function [phi,dphidpx,dphidpy,dphidpz,cosphi] = find_phi_dphi(NODES,CREASE,phi0)
Pl = NODES(CREASE(1),:);
Pi = NODES(CREASE(2),:);
Pj = NODES(CREASE(3),:);
Pk = NODES(CREASE(4),:);

m = cross((Pi-Pj),(Pk-Pj));
```

```
n = cross((Pk-Pj),(Pk-Pl));

cosphi = dot(m,n)/norm(n)/norm(m);
    if dot(m,(Pk-Pl))==0
        phi = mod(1*real(acos(cosphi)),2*pi);
    else
        phi = mod(sign(dot(m,(Pk-Pl)))*real(acos(cosphi)),2*pi);
    end

dphidpl = -norm(Pk-Pj)/(norm(n))^2*n;
dphidpi = norm(Pk-Pj)/(norm(m))^2*m;
dphidpj = (dot((Pi-Pj),(Pk-Pj))/(norm(Pk-Pj))^2-1)*dphidpi - ...
dot((Pk-Pl),(Pk-Pj))/(norm(Pk-Pj))^2*dphidpl;
dphidpk = (dot((Pk-Pl),(Pk-Pj))/(norm(Pk-Pj))^2-1)*dphidpl - ...
dot((Pi-Pj),(Pk-Pj))/(norm(Pk-Pj))^2*dphidpi;

dphidpx = [dphidpl(1) dphidpi(1)  dphidpj(1)  dphidpk(1) ];
dphidpy = [dphidpl(2) dphidpi(2)  dphidpj(2)  dphidpk(2) ];
dphidpz = [dphidpl(3) dphidpi(3)  dphidpj(3)  dphidpk(3) ];
end
```

Bibliography

[1] Adrian K Agogino, Vytas SunSpiral, and David Atkinson. Super ball bot-structures for planetary landing and exploration. *NASA Technical Report*, 2018.

[2] Amir F. Atiya and Alexander G. Parlos. New results on recurrent network training: unifying the algorithms and accelerating convergence. *IEEE Transactions on Neural Networks*, 11(3):697–709, 2000.

[3] Christoph H. Belke and Jamie Paik. Mori: A Modular Origami Robot. *IEEE/ASME Transactions on Mechatronics*, 22(5):2153–2164, 2017.

[4] Eran Ben-haim, Lior Salem, Yizhar Or, and Amir D. Gat. Single-Input Control of Multiple Fluid-Driven Elastic Actuators Via Interaction Between Bi-Stability and Viscosity. *ArXiv*, pages 1–7, 2019.

[5] Priyanka Bhovad, Joshua Kaufmann, and Suyi Li. Peristaltic locomotion without digital controllers: Exploiting the origami multi-stability to coordinate robotic motions. *Extreme Mechanics Letters*, 32:100552, 2019.

[6] Priyanka Bhovad and Suyi Li. Using multi-stable origami mechanism for peristaltic gait generation: A case study. In *ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V05BT07A061–V05BT07A061. American Society of Mechanical Engineers, 2018.

[7] Mustafa Boyvat, Je Sung Koh, and Robert J. Wood. Addressable wireless actuation for multijoint folding robots and devices. *Science Robotics*, 2(8):1–10, 2017.

[8] Ariel Calderon, Joakin C. Ugalde, Longlong Chang, Juan Cristobal Zagal, and Nestor O Perez-Arancibia. An earthworm-inspired soft robot with perceptive artificial skin. *Bioinspiration & Biomimetics*, pages 0–12, mar 2019.

[9] K. Caluwaerts, M. D'Haene, D. Verstraeten, and B. Schrauwen. Locomotion without a brain: Physical reservoir computing in Tensegrity structures. *Artificial Life*, 19(1):35–66, 2013.

[10] Ken Caluwaerts and Benjamin Schrauwen. The body as a reservoir: locomotion and sensing with linear feedback. In *Conference proceedings : 2nd international conference on morphological computation*, page 3, 2011.

[11] V Chalvet, Y Haddab, and P Lutz. A microfabricated planar digital microrobot for precise positioning based on bistable modules. *IEEE Transactions on Robotics*, 29(3):641–649, jun 2013.

[12] T Chen, O R Bilal, K Shea, and C Daraio. Harnessing bistability for directional propulsion of soft, untethered robots. *Proceedings of the National Academy of Sciences*, 115(22):5698–5702, may 2018.

[13] Y Chen, R Peng, and Z You. Origami of thick panels. *Science (80-.)*, 349:396–400, 2015.

[14] M. Cianchetti, M. Calisti, L. Margheri, M. Kuba, and C. Laschi. Bioinspired locomotion and grasping in water: The soft eight-arm OCTOPUS robot. *Bioinspiration and Biomimetics*, 10(3):035003, 2015.

[15] Steven H Collins, Martijn Wisse, and Andy Ruina. A Three-Dimensional Walking Robot with Two Legs and Knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

[16] M F Daqaq, R Masana, A Erturk, and D Dane Quinn. On the role of nonlinearities in vibratory energy harvesting: A critical review and discussion. *Applied Mechanics Reviews*, 66(4):040801, may 2014.

[17] Jonas Degrave, Ken Caluwaerts, Joni Dambre, and Francis Wyffels. Developing an embodied gait on a compliant quadrupedal robot. *IEEE International Conference on Intelligent Robots and Systems*, 2015-Decem:4486–4491, 2015.

[18] E D Demaine and J O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, 2007.

[19] L H Dudte, E Vouga, T Tachi, and L Mahadevan. Programming curvature using origami tessellations. *Nat. Mater.*, 15:583–588, 2016.

[20] J Dumais and Y Forterre. 'vegetable dynamicks': the role of water in plant movements. *Annu. Rev. Fluid Mech.*, 44:453–478, 2012.

[21] M Eidini and G H Paulino. Unraveling metamaterial properties in zigzag-base folded sheets. *Sci. Adv.*, 1:e1500224–e1500224, 2015.

[22] H Fang, S Li, H Ji, and K W Wang. Uncovering the deformation mechanisms of origami metamaterials by introducing generic degree-four vertices. *Phys. Rev. E*, 94, 2016.

[23] H Fang, S Li, H Ji, and K W Wang. Uncovering the deformation mechanisms of origami metamaterials by introducing generic degree-four vertices. *Physical Review E*, 94(4):043002, oct 2016.

[24] H Fang, S Li, K W Wang, and J Xu. Phase coordination and phase–velocity relationship in metameric robot locomotion. *Bioinspiration & Biomimetics*, 10(6):066006, oct 2015.

[25] Hongbin Fang, Suyi Li, K. W. Wang, and Jian Xu. A comprehensive study on the locomotion characteristics of a metameric earthworm-like robot. *Multibody System Dynamics*, 34(4):391–413, aug 2015.

[26] Hongbin Fang, Yetong Zhang, and K. W. Wang. Origami-based earthworm-like locomotion robots. *Bioinspiration and Biomimetics*, 12(6):065003, 2017.

[27] Hadi Fekrmandi and Phillip Hillard. A pipe-crawling robot using bio-inspired peristaltic locomotion and modular actuated non-destructive evaluation mechanism. *Bioinspiration, Biomimetics, and Bioreplication IX*, 1096508(March):7, 2019.

[28] S Felton, M Tolley, E D Demaine, D Rus, and R Wood. A method for building self-folding machines. *Science (80-.)*, 345:644–646, 2014.

[29] Chrisantha Fernando and Sampsa Sojakka. Pattern recognition in a bucket. In Wolfgang Banzhaf, Jens Ziegler, Thomas Christaller, Peter Dittrich, and Jan T. Kim, editors, *Advances in Artificial Life*, pages 588–597, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[30] E T Filipov, K Liu, T Tachi, M Schenk, and G H Paulino. Bar and hinge models for scalable analysis of origami. *Int. J. Solids Struct.*, 124:26–45, 2017.

[31] E. T. Filipov, G. H. Paulino, and T. Tachi. Origami tubes with reconfigurable polygonal cross-sections. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 472(2185):20150607, 2016.

[32] E T Filipov, T Tachi, and G H Paulino. Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials. *Proc. Natl. Acad. Sci.*, 112:12321–12326, 2015.

[33] E T Filipov, T Tachi, and G H Paulino. Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials. *Proceedings of the National Academy of Sciences*, 112(40):12321–12326, oct 2015.

[34] E.T. Filipov and M. Redoutey. Mechanical characteristics of the bistable origami hypar. *Extreme Mechanics Letters*, 25:16–26, 2018.

[35] Dario Floreano, Ramon Pericet-Camara, Stéphane Viollet, Franck Ruffier, Andreas Brückner, Robert Leitel, Wolfgang Buss, Mohsine Menouni, Fabien Expert, Raphaël Juston, Michal Karol Dobrzynski, Geraud L'Eplattenier, Fabian Recktenwald, Hanspeter A. Mallot, and Nicolas Franceschini. Miniature curved artificial compound eyes. *Proceedings of the National Academy of Sciences of the United States of America*, 110(23):9267–9272, 2013.

[36] Y Forterre. Slow, fast and furious: understanding the physics of plant movements. *J. Exp. Bot.*, 64:4745–4760, 2013.

[37] Y Forterre. Slow, fast and furious: understanding the physics of plant movements. *Journal of Experimental Botany*, 64(15):4745–4760, 2013.

[38] Y Forterre, J M Skotheim, J Dumais, and L Mahadevan. How the venus flytrap snaps. *Nature*, 433:421–425, 2005.

[39] Rudolf M. Füchslin, Andrej Dzyakanchuk, Dandolo Flumini, Helmut Hauser, Kenneth J. Hunt, Rolf H. Luchsinger, Benedikt Reller, Stephan Scheidegger, and Richard Walker. Morphological computation and morphological control: Steps toward a formal theory and applications. *Artificial Life*, 19(1):9–34, 2013.

[40] M. Garrad, G. Soter, A. T. Conn, H. Hauser, and J. Rossiter. A soft matter computer for soft robots. *Science Robotics*, 4(33), 2019.

[41] Amanda Ghassaei, Erik D Demaine, and Neil Gershenfeld. Fast, interactive origami simulation using gpu computation. *Origami*, 7:1151–1166, 2018.

[42] Benjamin Gorissen, Edoardo Milana, Arne Baeyens, Eva Broeders, Jeroen Christiaens, Klaas Collin, Dominiek Reynaerts, and Michael Volder. Hardware Sequencing of Inflatable Nonlinear Actuators for Autonomous Soft Robots. *Advanced Materials*, 31(3):1804598, jan 2019.

[43] B Gramüller, H Köke, and C Hühne. Holistic design and implementation of pressure actuated cellular structures. *Smart Mater. Struct.*, 24, 2015.

[44] Michael D. Grissom, Vilas Chitrakaran, Dustin Dienno, Matthew Csencits, Michael Pritts, Bryan Jones, William McMahan, Darren Dawson, Chris Rahn, and Ian Walker. Design and experimental testing of the OctArm soft robot manipulator. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage, editors, *Unmanned Systems Technology VIII*, page 62301F, may 2006.

[45] Simon D Guest and Sergio Pellegrino. The folding of triangulated cylinders, part i: geometric considerations. *Journal of applied mechanics*, 61(4):773–777, 1994.

[46] Brandon H Hanna, Spencer P Magleby, Robert J Lang, and Larry L Howell. Force–Deflection Modeling for Generalized Origami Waterbomb-Base Mechanisms. *Journal of Applied Mechanics*, 82(8):81001, 2015.

[47] Michael W. Hannan and Ian D. Walker. Kinematics and the Implementation of an Elephant's Trunk Manipulator and Other Continuum Style Robots. *Journal of Robotic Systems*, 20(2):45–63, 2003.

[48] R L Harne and K W Wang. A review of the recent research on vibration energy harvesting via bistable systems. *Smart Materials and Structures*, 22(2):023001, jan 2013.

[49] Helmut Hauser, Auke J. Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer, and Wolfgang Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105(5-6):355–370, 2011.

[50] Helmut Hauser, Auke J. Ijspeert, Rudolf M. Füchslin, Rolf Pfeifer, and Wolfgang Maass. The role of feedback in morphological computation with compliant bodies. *Biological Cybernetics*, 106(10):595–613, 2012.

[51] Jonathan Hiller and Hod Lipson. Dynamic Simulation of Soft Multimaterial 3D-Printed Objects. *Soft Robotics*, 1(1):88–101, 2014.

[52] Matej Hoffmann and Vincent C. Müller. Simple or complex bodies? Trade-offs in exploiting body morphology for control. *Studies in Applied Philosophy, Epistemology and Rational Ethics*, 28(January 2018):335–345, 2017.

[53] Nan Hu and Rigoberto Burgueño. Buckling-induced smart applications: recent advances and trends. *Smart Materials and Structures*, 24(6):063001, 2015.

[54] Giles W. Hunt and Ichiro Ario. Twist buckling and the foldable cylinder: An exercise in origami. *International Journal of Non-Linear Mechanics*, 40(6):833–843, jul 2005.

[55] Giles W Hunt and Ichiro Ario. Twist buckling and the foldable cylinder: an exercise in origami. *International Journal of Non-Linear Mechanics*, 40(6):833–843, 2005.

[56] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

[57] Herbert Jaeger. ESNTutorialRev.pdf. 2002:1–46, 2008.

[58] Donghwa Jeong and Kiju Lee. Design and analysis of an origami-based three-finger manipulator. *Robotica*, pages 1–14, 2017.

[59] Yijie Jiang, Lucia M. Korpas, and Jordan R. Raney. Bifurcation-based embodied logic and autonomous actuation. *Nature Communications*, 10(1):128, 2019.

[60] Cai Jianguo, Deng Xiaowei, Zhou Ya, Feng Jian, and Tu Yongming. Bistable behavior of the cylindrical origami structure with kresling pattern. *Journal of Mechanical Design*, 137(6):061406, 2015.

[61] D R Johnson, R L Harne, and K W Wang. A disturbance cancellation perspective on vibration control using a bistable snap-through attachment. *Journal of Vibration and Acoustics*, 136(3):031006, mar 2014.

[62] D R Johnson, M Thota, F Semperlotti, and K W Wang. On achieving high and adaptable damping via a bistable oscillator. *Smart Materials and Structures*, 22(11):115027, oct 2013.

[63] M Johnson, Y Chen, S Hovet, S Xu, B Wood, H Ren, J Tokuda, and Z T H Tse. Fabricating biomedical origami: a state-of-the-art review. *Int. J. Comput. Assist. Radiol. Surg.*, 12:2023–2032, 2017.

[64] Aniket Joshi, Adwait Kulkarni, and Yonas Tadesse. Fludojelly: experimental study on jellyfish-like soft robot enabled by soft pneumatic composite (spc). *Robotics*, 8(3):56, 2019.

[65] M. Kamata, S. Yamazaki, Y. Tanise, Y. Yamada, and T. Nakamura. Morphological change in peristaltic crawling motion of a narrow pipe inspection robot inspired by earthworm's locomotion. *Advanced Robotics*, 32(7):386–397, 2018.

[66] Soroush Kamrava, Davood Mousanezhad, Hamid Ebrahimi, Ranajay Ghosh, and Ashkan Vaziri. Origami-based cellular metamaterial with auxetic, bistable, and self-locking properties. *Scientific Reports*, 7:46046, 2017.

[67] Robert K. Katzschmann, Andrew D. Marchese, and Daniela Rus. Hydraulic autonomous soft robotic fish for 3D swimming. *Springer Tracts in Advanced Robotics*, 109:405–420, 2016.

[68] Joshua Kaufmann, Priyanka Bhovad, and Suyi Li. Harnessing the Multistability of Kresling Origami for Reconfigurable Articulation in Soft Robotic Arms. *Soft Robotics*, 00(00):soro.2020.0075, feb 2021.

[69] S W Kim, J S Koh, J G Lee, J Ryu, M Cho, and K J Cho. Flytrap-inspired robot using structurally integrated actuation based on bistability and a developable surface. *Bioinspiration & Biomimetics*, 9(3):036004, mar 2014.

[70] Yoonho Kim, Hyunwoo Yuk, Ruike Zhao, Shawn A. Chester, and Xuanhe Zhao. Printing ferromagnetic domains for untethered fast-transforming soft materials. *Nature*, 558(7709):274–279, 2018.

[71] K. G. Kirby and N. Day. The neurodynamics of context reverberation learning. *[1990] Proceedings of the Twelfth Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1781–1782, 1990.

[72] Biruta Kresling. Natural twist buckling in shells: From the hawkmoth's bellows to the deployable kresling-pattern and cylindrical miuraori. In *Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures*, pages 1–4, 2008.

[73] R J Lang. A computational algorithm for origami design. pages 98–105, 1996.

[74] C. Laschi and B. Mazzolai. Lessons from animals and plants: The symbiosis of morphological computation and soft robotics. *IEEE Robotics and Automation Magazine*, 23(3):107–114, 2016.

[75] C Laschi, B Mazzolai, and M Cianchetti. Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Sci. Robot.*, 1, 2016.

[76] Cecilia Laschi, Barbara Mazzolai, and Matteo Cianchetti. Soft robotics: Technologies and systems pushing the boundaries of robot abilities. *Science Robotics*, 1(1):eaah3690, 2016.

[77] F. Lechenault and M. Adda-Bedia. Generic bistability in creased conical surfaces. *Phys. Rev. Lett.*, 115:235501, Dec 2015.

[78] Robert Albin Legenstein and Wolfgang Maass. *What makes a dynamical system computationally powerful?*, pages 127–154. MIT Press, 1 edition, 2007.

[79] S Li, H Fang, and K W Wang. Recoverable and programmable collapse from folding pressurized origami cellular solids. *Phys. Rev. Lett.*, 117, 2016.

[80] S Li and K W Wang. Fluidic origami: a plant-inspired adaptive structure with shape morphing and stiffness tuning. *Smart Mater. Struct.*, 24, 2015.

[81] S Li and K W Wang. Fluidic origami: a plant-inspired adaptive structure with shape morphing and stiffness tuning. *Smart Materials and Structures*, 24(10):105031, 2015.

[82] S Li and K W Wang. Fluidic origami with embedded pressure dependent multi-stability : a plant inspired innovation. *J. R. Soc. Interface*, 12, 2015.

[83] S Li and K W Wang. Plant-inspired adaptive structures and materials for morphing and actuation: a review. *Bioinspir. Biomim.*, 12, 2017.

[84] Suyi Li, Hongbin Fang, Sahand Sadeghi, Priyanka Bhovad, and Kon Well Wang. Architected Origami Materials: How Folding Creates Sophisticated Mechanical Properties. *Advanced Materials*, 31(5):1–18, 2019.

[85] Tao Li, Kohei Nakajima, Matteo Cianchetti, Cecilia Laschi, and Rolf Pfeifer. Behavior switching using reservoir computing for a soft robotic arm. *Proceedings - IEEE International Conference on Robotics and Automation*, 1:4918–4924, 2012.

[86] K Liu and G H Paulino. Nonlinear mechanics of non-rigid origami: an efficient computational approach. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 473(2206):20170348, 2017.

[87] Ke Liu, Larissa S Novelino, Paolo Gardoni, and Glaucio H Paulino. Big influence of small random imperfections in origami-based metamaterials. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 476(2241):20200236, 2020.

[88] A Lobkovsky, S Gentges, H Li, D Morse, and T A Witten. Scaling properties of stretching ridges in a crumpled elastic sheet. *Science (80-.)*, 270:1482–1485, 1995.

[89] Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[90] K. Y. Ma, P. Chirarattananon, S. B. Fuller, and R. J. Wood. Controlled Flight of a Biologically Inspired, Insect-Scale Robot. *Science*, 340(6132):603–607, may 2013.

[91] Wolfgang Maass. Liquid state machines: motivation, theory, and applications. In *Computability in context: computation and logic in the real world*, pages 275–296. World Scientific, 2011.

[92] Wolfgang Maass, Prashant Joshi, and Eduardo D. Sontag. Computational aspects of feedback in neural circuits. *PLoS Computational Biology*, 3(1):0015–0034, 2007.

[93] Wolfgang Wolfang Maass, Henry Markram, and Thomas Natschläger. The " liquid computer": A novel strategy for real-time computing on time series. *Special Issue on Foundations of Information Processing of TELEMATIK*, 8(1):39–43, 2002.

[94] Stephen T. Mahon, Anthony Buchoux, Mohammed E. Sayed, Lijun Teng, and Adam A. Stokes. Soft robots for extreme environments: Removing electronic control. *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, pages 782–787, 2019.

[95] M McArthur and R J Lang. *Folding paper: The Infinite Possibilities of Origami*. Tuttle Publishing, 2013.

[96] David Melancon, Benjamin Gorissen, Carlos J. Garcia-Mora, Chuck Hoberman, and Katia Bertoldi. Multistable inflatable origami structures at the meter-scale. *Nature*, (accepted)(August 2020), 2021.

[97] Edoardo Milana, Benjamin Gorissen, Michael De Voider, and Dominiek Reynaerts. Design of a bi-segmented soft actuator with hardware encoded quasi-static inflation sequence. *2018 IEEE International Conference on Soft Robotics, RoboSoft 2018*, pages 108–113, 2018.

[98] Aslan Miriyev and Mirko Kovač. Skills for physical artificial intelligence. *Nature Machine Intelligence*, 2(11):658–660, 2020.

[99] S Miyashita, S Guitron, M Ludersdorfer, C R Sung, and D Rus. An untethered miniature origami robot that self-folds, walks, swims, and degrades. In *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume vol 2015, pages 1490–1496. IEEE, 2015. –June.

[100] Shuhei Miyashita, Steven Guitron, Shuguang Li, and Daniela Rus. Robotic metamorphosis by origami exoskeletons. *Science Robotics*, 2(10):eaao4369, 2017.

[101] Shuhei Miyashita, Steven Guitron, Kazuhiro Yoshida, Shuguang Li, Dana D. Damian, and Daniela Rus. Ingestible, controllable, and degradable origami robot for patching stomach wounds. *Proceedings - IEEE International Conference on Robotics and Automation*, 2016-June(5):909–916, 2016.

[102] Giorgio Morales, Samuel G Huam, Joel Telles, Yuki Yamanaka, Kohei Nakajima, Takaharu Yaguchi, and Helmut Hauser. Mass-Spring Damper Array as a Mechanical Medium for Computation. *International Conference on Artificial Neural Networks*, 1:208–217, 2018.

[103] Jessica Morgan, Spencer P. Magleby, and Larry L. Howell. An Approach to Designing Origami-Adapted Aerospace Mechanisms. *Journal of Mechanical Design*, 138(5):052301, 2016.

[104] Elissa Morris, Daniel A McAdams, and Richard Malak. The state of the art of origami-inspired products: A review. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 50169, page V05BT07A014. American Society of Mechanical Engineers, 2016.

[105] Vincent C. Müller and Matej Hoffmann. What Is Morphological Computation? On How the Body Contributes to Cognition and Control. *Artificial Life*, 23(1):1–24, feb 2017.

[106] Kohei Nakajima. Physical reservoir computing-an introductory perspective. *Japanese Journal of Applied Physics*, 59(6), 2020.

[107] Kohei Nakajima and Ingo Fischer. Reservoir computing, 2021.

[108] Kohei Nakajima, Helmut Hauser, Rongjie Kang, Emanuele Guglielmino, Darwin G. Caldwell, and Rolf Pfeifer. A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. *Frontiers in Computational Neuroscience*, 7(July):1–19, 2013.

[109] Kohei Nakajima, Helmut Hauser, Tao Li, and Rolf Pfeifer. Exploiting the dynamics of soft materials for machine learning. *Soft Robotics*, 5(3):339–347, 2018.

[110] Nigamaa Nayakanti, Sameh H. Tawfick, and A. John Hart. Twist Coupled Kirigami Cellular Metamaterials and Mechanisms. *Extreme Mechanics Letters*, 2017.

[111] Xin Ning, Xueju Wang, Yi Zhang, Xinge Yu, Dongwhi Choi, Ning Zheng, Dong Sung Kim, Yonggang Huang, Yihui Zhang, and John A Rogers. Assembly of advanced materials into 3d functional structures by methods inspired by origami and kirigami: a review. *Advanced Materials Interfaces*, 5(13):1800284, 2018.

[112] Larissa S. Novelino, Qiji Ze, Shuai Wu, Glaucio H. Paulino, and Ruike Zhao. Untethered control of functional origami microrobots with distributed actuation. *Proceedings of the National Academy of Sciences*, 117(39):24096–24101, 2020.

[113] Young Seok Oh and Sridhar Kota. Synthesis of Multistable Equilibrium Compliant Mechanisms Using Combinations of Bistable Mechanisms. *Journal of Mechanical Design*, 131(2):021002, 2009.

[114] Cagdas D. Onal, Michael T. Tolley, Robert J. Wood, and Daniela Rus. Origami-Inspired Printed Robots. *IEEE/ASME Transactions on Mechatronics*, 20(5):2214–2221, 2015.

[115] Cagdas D. Onal, Robert J. Wood, and Daniela Rus. An origami-inspired approach to worm robots. *IEEE/ASME Transactions on Mechatronics*, 18(2):430–438, 2013.

[116] J T B Overvelde, J C Weaver, C Hoberman, and K Bertoldi. Rational design of reconfigurable prismatic architected materials. *Nature*, 541:347–352, 2017.

[117] Johannes T. B. Overvelde, Tamara Kloek, Jonas J. A. D'haen, and Katia Bertoldi. Amplifying the response of soft actuators by harnessing snap-through instabilities. *Proceedings of the National Academy of Sciences*, 112(35):10863–10868, 2015.

[118] Alexander Pagano, Tongxi Yan, Brian Chien, A Wissa, and S Tawfick. A crawling robot driven by multi-stable origami. *Smart Materials and Structures*, 26(9):094007, 2017.

[119] Ajit S. Panesar and Paul M. Weaver. Optimisation of blended bistable laminates for a morphing flap. *Composite Structures*, 94(10):3092–3105, oct 2012.

[120] Sung Jin Park, Mattia Gazzola, Kyung Soo Park, Shirley Park, Valentina Di Santo, Erin L. Blevins, Johan U. Lind, Patrick H. Campbell, Stephanie Dauth, Andrew K. Capulli, Francesco S. Pasqualini, Seungkuk Ahn, Alexander Cho, Hongyan Yuan, Ben M. Maoz, Ragu Vijaykumar, Jeong Woo Choi, Karl Deisseroth, George V. Lauder, L. Mahadevan, and Kevin Kit Parker. Phototactic guidance of a tissue-engineered soft-robotic ray. *Science*, 353(6295):158–162, 2016.

[121] Chandana Paul. *Investigation of Morphology and Control in Biped Locomotion*. PhD thesis, University of Zurich, Switzerland, 2004.

[122] Chandana Paul. Morphological computation. A basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54(8):619–630, 2006.

[123] Chandana Paul, Francisco J. Valero-Cuevas, and Hod Lipson. Design and control of tensegrity robots for locomotion. *IEEE Transactions on Robotics*, 22(5):944–957, 2006.

[124] Edwin A Peraza-Hernandez, Darren J. Hartl, Richard J. Malak Jr, and Dimitris C. Lagoudas. Origami-inspired active structures: a synthesis and review. *Smart Materials and Structures*, 23(9):094001, 2014.

[125] D J Preston, H J Jiang, V Sanchez, P Rothemund, J Rawson, M P Nemitz, W-K Lee, Z Suo, C J Walsh, and G M Whitesides. A soft ring oscillator. *Science Robotics*, 4(31):eaaw5496, 2019.

[126] Daniel J. Preston, Philipp Rothemund, Haihui Joy Jiang, Markus P. Nemitz, Jeff Rawson, Zhigang Suo, and George M. Whitesides. Digital logic for soft devices. *Proceedings of the National Academy of Sciences*, page 201820672, 2019.

[127] Kj Quillin and Quillin. Kinematic scaling of locomotion by hydrostatic animals: ontogeny of peristaltic crawling by the earthworm lumbricus terrestris. *The Journal of experimental biology*, 202 (Pt 6)(1999):661–74, mar 1999.

[128] Ahmad Rafsanjani, Katia Bertoldi, and André R. Studart. Programming soft robots with flexible mechanical metamaterials. *Science Robotics*, 4(29):eaav7874, 2019.

[129] Ahmad Rafsanjani, Lishuai Jin, Bolei Deng, and Katia Bertoldi. Propagation of pop ups in kirigami shells. *Proceedings of the National Academy of Sciences of the United States of America*, 116(17):8200–8205, 2019.

[130] Ahmad Rafsanjani, Yuerou Zhang, Bangyuan Liu, Shmuel M. Rubinstein, and Katia Bertoldi. Kirigami skins make a simple soft actuator crawl. *Science Robotics*, 3(15):1–8, 2018.

[131] Ziyu Ren, Wenqi Hu, Xiaoguang Dong, and Metin Sitti. Multi-functional soft-bodied jellyfish-like swimming. *Nature Communications*, 10(1), 2019.

[132] S I Rich, R J Wood, and C Majidi. Untethered soft robotics. *Nature Electronics*, 1(2):102–112, feb 2018.

[133] D Rus and M T Tolley. Design, fabrication and control of origami robots. *Nature Reviews Materials*, 3(6):101–112, jun 2018.

[134] S Sadeghi and S Li. Fluidic origami cellular structure with asymmetric quasi-zero stiffness for low-frequency vibration isolation. *Smart Materials and Structures*, pages 11–14, 2019.

[135] Sahand Sadeghi, Samuel Allison, Blake Betsill, and Suyi Li. TMP Origami Jumping Mechanism with Nonlinear Stiffness. *Smart Materials and Structures*, 2021.

[136] Sahand Sadeghi and Suyi Li. Dynamic folding of origami by exploiting asymmetric bi-stability. *Extreme Mechanics Letters*, 40:100958, 2020.

[137] Norihiko Saga and Taro Nakamura. Development of a peristaltic crawling robot using magnetic fluid on the basis of the locomotion mechanism of the earthworm. *Smart Materials and Structures*, 13(3):566–569, 2004.

[138] M. Schenk and S. D. Guest. Origami folding: A structural engineering approach. In *Origami 5 Fifth International Meeting of Origami Science Mathematics and Education*, pages 1–16, 2011.

[139] M Schenk and S D Guest. Geometry of miura-folded metamaterials. *Proc. Natl. Acad. Sci.*, 110:3276–3281, 2013.

[140] M Schenk, A D Viquerat, K A Seffen, and S D Guest. Review of inflatable booms for deployable space structures: packing and rigidization. *J. Spacecr. Rockets*, 51:762–778, 2014.

[141] Mark Schenk and Simon D. Guest. Geometry of Miura-folded metamaterials. *Proceedings of the National Academy of Sciences*, 110(9):3276–3281, feb 2013.

[142] Benjamin Schrauwen, David Verstraeten, and Jan Van Campenhout. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks. p. 471-482 2007*, pages 471–482, 2007.

[143] Sangok Seok, Cagdas Denizel Onal, Kyu Jin Cho, Robert J. Wood, Daniela Rus, and Sangbae Kim. Meshworm: A peristaltic soft robot with antagonistic nickel titanium coil actuators. *IEEE/ASME Transactions on Mechatronics*, 18(5):1485–1497, 2013.

[144] J L Silverberg, A A Evans, L McLeod, R C Hayward, T C Hull, C D Santangelo, and I Cohen. Using origami design principles to fold reprogrammable mechanical metamaterials. *Science*, 345(6197):647–650, aug 2014.

[145] J L Silverberg, J H Na, A A Evans, B Liu, T C Hull, C D Santangelo, R J Lang, R C Hayward, and I Cohen. Origami structures with a critical transition to bistability arising from hidden degrees of freedom. *Nature Materials*, 14(4):389–393, mar 2015.

[146] J Sun, Q Guan, Y Liu, and J Leng. Morphing aircraft based on smart materials and structures: A state-of-the-art review. *Journal of Intelligent Material Systems and Structures*, 27(17):2289–2312, feb 2016.

[147] Z Suo, D J Preston, L Belding, P Rothemund, S Kurihara, A Ainla, and G M Whitesides. A soft, bistable valve for autonomous control of soft actuators. *Science Robotics*, 3(16):eaar7986, 2018.

[148] T Tachi. Freeform variations of origami. *J. Geom. Graph.*, 14:203–215, 2010.

[149] Tomohiro Tachi. Designing Freeform Origami Tessellations by Generalizing Resch's Patterns. *Journal of Mechanical Design*, 135(11):111006, oct 2013.

[150] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[151] Michael T. Tolley, Robert F. Shepherd, Bobak Mosadegh, Kevin C. Galloway, Michael Wehner, Michael Karpelson, Robert J. Wood, and George M. Whitesides. A Resilient, Untethered Soft Robot. *Soft Robotics*, 1(3):213–223, 2014.

[152] Benjamin Treml, Andrew Gillman, Philip Buskohl, and Richard Vaia. Origami mechanologic. *Proceedings of the National Academy of Sciences*, page 201805122, jun 2018.

[153] Deepak Trivedi, Christopher D. Rahn, William M. Kier, and Ian D. Walker. Soft robotics: Biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics*, 5(3):99–117, 2008.

[154] T Tsuchiya, K Kameyama, Y Kishi, M Yoshimura, N Kanzawa, and M Sameshima. Tyrosine phosphorylation in plant bending. *Nature*, 407:37–37, 2000.

[155] Gabriel Urbain, Jonas Degrave, Benonie Carette, Joni Dambre, and Francis Wyffels. Morphological properties of mass-spring networks for optimal locomotion learning. *Frontiers in Neurorobotics*, 11(MAR):1–13, 2017.

[156] S Vasista and L Tong. Topology-optimized design and testing of a pressure-driven morphing-aerofoil trailing-edge structure. *AIAA J.*, 51:1898–1907, 2013.

[157] S Vasista, L Tong, and K C Wong. Realization of morphing wings: a multidisciplinary challenge. *J. Aircr.*, 49:11–28, 2012.

[158] R Vos and R Barrett. Mechanics of pressure-adaptive honeycomb and its application to wing morphing. *Smart Mater. Struct.*, 20, 2011.

[159] S Waitukaitis, R Menaut, B G Chen, and M van Hecke. Origami multistability: from single vertices to metasheets. *Physical Review Letters*, 114(5):055503, feb 2015.

[160] Scott Waitukaitis, Rémi Menaut, Bryan Gin-ge Chen, and Martin van Hecke. Origami multistability: From single vertices to metasheets. *Phys. Rev. Lett.*, 114:055503, Feb 2015.

[161] Ian D. Walker, Darren M. Dawson, Tamar Flash, Frank W. Grasso, Roger T. Hanlon, Binyamin Hochner, William M. Kier, Christopher C. Pagano, Christopher D. Rahn, and Qiming M. Zhang. Continuum robot arms inspired by cephalopods. In Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage, editors, *Unmanned Ground Vehicle Technology VII*, page 303, may 2005.

[162] P Wang, T A Meyer, V Pan, P K Dutta, and Y Ke. The beauty and utility of dna origami. *Chem.*, 2:359–382, 2017.

[163] Michael Wehner, Ryan L. Truby, Daniel J. Fitzgerald, Bobak Mosadegh, George M. Whitesides, Jennifer A. Lewis, and Robert J. Wood. An integrated design and fabrication strategy for entirely soft, autonomous robots. *Nature*, 536(7617):451–455, 2016.

[164] Yichuan Wu, Justin K Yim, Jiaming Liang, Zhichun Shao, Mingjing Qi, Junwen Zhong, Zihao Luo, Xiaojun Yan, Min Zhang, Xiaohao Wang, et al. Insect-scale fast moving and ultrarobust soft robot. *Science Robotics*, 4(32):eaax1594, 2019.

[165] Ruibo Yan, Ming Luo, Zhenyu Wan, Yun Qin, Junius Santoso, Erik Skorina, and Cagdas Onal. OriSnake: Design, Fabrication and Experimental Analysis of a 3-D Origami Snake Robot. *IEEE Robotics and Automation Letters*, 3(3):1–1, 2018.

[166] Zheng Yan, Fan Zhang, Jiechen Wang, Fei Liu, Xuelin Guo, Kewang Nan, Qing Lin, Mingye Gao, Dongqing Xiao, Yan Shi, Yitao Qiu, Haiwen Luan, Jung Hwan Kim, Yiqi Wang, Hongying Luo, Mengdi Han, Yonggang Huang, Yihui Zhang, and John A Rogers. Controlled Mechanical Buckling for Origami-Inspired Construction of 3D Microstructures in Advanced Materials. *Advanced Functional Materials*, 26(16):2629–2639, apr 2016.

[167] Dian Yang, Bobak Mosadegh, Alar Ainla, Benjamin Lee, Fatemeh Khashai, Zhigang Suo, Katia Bertoldi, and George M. Whitesides. Buckling of Elastomeric Beams Enables Actuation of Soft Machines. *Advanced Materials*, 27(41):6323–6327, 2015.

[168] Dian Yang, Mohit S. Verma, Ju Hee So, Bobak Mosadegh, Christoph Keplinger, Benjamin Lee, Fatemeh Khashai, Elton Lossner, Zhigang Suo, and George M. Whitesides. Buckling Pneumatic Linear Actuators Inspired by Muscle. *Advanced Materials Technologies*, 1(3):31–33, 2016.

[169] H. Yasuda and J. Yang. Reentrant origami-based metamaterials with negative Poisson's ratio and bistability. *Physical Review Letters*, 114(18):1–5, 2015.

[170] Hiromi Yasuda, Balakumaran Gopalarethinam, Takahiro Kunimine, Tomohiro Tachi, and Jinkyu Yang. Origami-Based Cellular Structures with In Situ Transition between Collapsible and Load-Bearing Configurations. *Advanced Engineering Materials*, 1900562:1900562, 2019.

[171] Davood Younesian and Mohammad Reza Alam. Multi-stable mechanisms for high-efficiency and broadband ocean wave energy harvesting. *Applied Energy*, 197:292–302, 2017.

[172] Z. Zhakypov, M. Falahi, M. Shah, and J. Paik. The design and control of the multi-modal locomotion origami robot, tribot. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4349–4355, 2015.

[173] X. Zhang, T. Pan, H. L. Heung, P. W. Y. Chiu, and Z. Li. A biomimetic soft robot for inspecting pipeline with significant diameter variation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7486–7491, Oct 2018.

[174] Shannon A. Zirbel, Robert J. Lang, Mark W. Thomson, Deborah A. Sigel, Phillip E. Walkemeyer, Brian P. Trease, Spencer P. Magleby, and Larry L. Howell. Accommodating Thickness in Origami-Based Deployable Arrays. *Journal of Mechanical Design*, 135(11):111005, 2013.