8-26-2022

Mechanologic: Designing Mechanical Devices that Compute

Michelle Berry
Syracuse University, msberry02@gmail.com

Abstract

Despite their initial success and impact on the development of the modern computer, mechanical computers were quickly replaced once electronic computers became viable. Recently, there has been increased interest in designing devices that compute using modern and unconventional materials. In this dissertation, we investigate multiple ways to realize a mechanical device that can compute, with a main focus on designing mechanical equivalents for wires and transistors. For our first approach at designing mechanical wires, we present results on the propagation of signals in a soft mechanical wire composed of bistable elements. When we send a signal along bistable wires that do not support infinite signal propagation, we find that signals can propagate for a finite distance controlled by a penetration length for perturbations. We map out various parameters for this to occur, and present results from experiments on wires made of soft elastomers. Our second approach for designing mechanical devices that compute focuses on designing the topology of the configuration space of a linkage. By programming the configuration space through small perturbations of the bar lengths in the linkage, we are able to design a linkage that gates the propagation of a soliton in a Kane-Lubensky chain. This dissertation also includes other results related to the study of small length changes in linkages and an analysis of a version of a mechanical transistor compatible with the soft bistable wires.

MECHANOLOGIC: DESIGNING MECHANICAL DEVICES THAT COMPUTE

by

Michelle Senick Berry

B.A., Goucher College, 2017

M.S., University of Massachusetts, Amherst, 2020

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Physics

Syracuse University

August 2022

Acknowledgements

There are so many people from both UMass Amherst and Syracuse that have encouraged and supported me along the way, and I wouldn't have gotten to this point without help from every single one of them. However, the person I owe the biggest thanks to is my advisor Chris Santangelo.

I can't thank Chris enough for supporting me over the past five years. Right after I passed my qualifying exams and was about to start working on research full time, I learned that he was moving from UMass to Syracuse. Throughout that whole process, and with the additional problems that 2020 introduced, he was always in my corner supporting me and making sure I didn't fall through the cracks. He has taught me so much over the years, and I'm glad I chose to stick with him.

To JP, Papa, and Dad

Contents

List of Figures

List of Tables

List of Publications

Chapters 2, 3, and 5 of the dissertation are comprised of the work carried out in the following papers:

- M Berry, ME Lee-Trimble, and CD Santangelo. Topological transitions in the configuration space of non-Euclidean origami. *Physical Review E*, 101(4):043003, 2020.
- M Berry, YJ Kim, D Limberg, RC Hayward, and CD Santangelo. Mechanical signaling cascades. *Under Review*, 2022
- M Berry, ME Lee-Trimble, D Limberg, RC Hayward, and CD Santangelo. Configuration space engineering for the topological gating of mechanical devices. *Under Review*, 2022

Chapter 1

Introduction and Background

With the current state of high-power computing and the ever increasing viability of quantum computing, it's reasonable to question why anyone would want to build a computer out of unconventional materials knowing that the processing power would be nothing compared to modern electronics. Mechanical devices that compute played an integral part in the development of the modern computer, but as technology advanced, they fell to the wayside in favor of new systems that were more compact, powerful, and reliable.

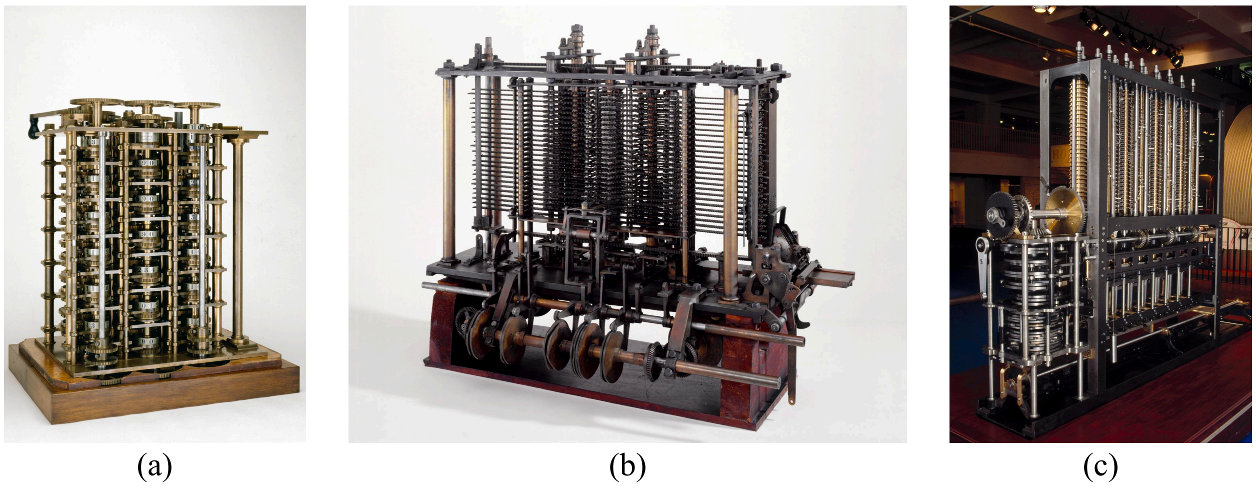


Figure 1.1: (a) The Difference Engine No. 1 built by Babbage [37]. (b) The Analytical Engine built by Babbage [36]. (c) The Analytical Engine No. 2 built by the London Science Museum based on Babbage's designs [38].

The argument in favor of looking into designing mechanical devices that compute focuses not on the computational power of the device but in the specifics of the system doing the computation. One could imagine certain situations where you might not want to use traditional semiconductor electronics to perform a calculation.

What if you're trying to put a computer in an environment that is so extreme that your electronics don't work well? NASA looked into this with the Automaton Rover for Extreme Environments (AREE) project which focused on designing a rover that can survive the environment on Venus [48]. If we can construct computers out of materials that are better suited to an extreme environment, more opportunities open up to explore and take data in places previously unaccessible.

A completely mechanical computer would require no external power supply. Again, this would be advantageous in extreme environments. It would also be useful for large scale sensing applications. If we could construct a tiny mechanical computer that requires no power source to sense a change in environmental conditions, we could freely distribute these over a large area, and depending on the material they are constructed with, not worry about the impact they would have if left there indefinitely.

When we say mechanical, we don't necessarily mean hard, rigid components like previous versions of mechanical computers. In the most general sense, mechanical means a physical component moving to perform a calculation. That movement could be a continuous motion of a rigid system, or it could be stretching/compression of a soft system. With these soft systems, we have the opportunity to also utilize swelling/shrinking materials that respond to environmental changes.

As shown in Fig. 1.1, previously constructed mechanical computers are very obviously not the right device for these motivating applications, so we need to re-design them from the ground up. We specifically focused on studying mechanical computers made out of linkages. The novelty in a computer built with linkages lies not in what it can do, but in the fact that it is built with linkages. Using a linkage as the base for a computational system opens up new opportunities for practical application of mechanical computers. Because of their simplicity, it is conceivable for them to be constructed at the molecular scale. They could also be created with flexures instead of rotary joints, which opens up a wide variety of options for how to manufacture them and what material to use.

In this first chapter, we dive deeper into what it means for a device to compute. If we want to make the statement that our system or device can compute, we need a well-defined, working definition of what exactly that means. We start off in Sec. 1.1 with a mathematical description of linkages which will allow us to analyze the behavior and motion of devices in future sections. We then transition to presenting definitions from computation theory related to a systems ability to perform a computation (Sec. 1.2) and outline possible ways to demonstrate that a device can compute. Finally, we connect the two previous sections and explore possible avenues for designing linkages that compute.

1.1 Linkages

A linkage is a system of fixed-length bars connected at their endpoints by freely rotating joints [24]. Linkages can be **flexible** and perform some non-trivial motion or be **rigid** and unable to move. These terms will be well-defined further in this section. The bars and joints can also be referred to as rods or links and vertices respectively. Any linkage can be represented as a graph with bars (rods, links) corresponding to edges and joints (vertices) corresponding to nodes.

We define the position of vertex i as $\mathbf{p}_i \in \mathbb{R}^d$ and the length of the bar joining vertices i and j as $L_{(i,j)}$. We can write a constraint for each bar that keeps its length constant.

$$|\mathbf{p}_i - \mathbf{p}_j|^2 = L^2_{(i,j)} \tag{1.1}$$

Differentiating the length constraint equation gives

$$(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}'_i - \mathbf{p}'_j) = 0 \tag{1.2}$$

where $\mathbf{p}'_i \in \mathbb{R}^d$ is the velocity of vertex i. Any set of vectors $\mathbf{p}'$ that satisfies Eq. 1.1 for all bars in the linkage is called an **infinitesimal flex**. In simple terms, $\mathbf{p}'$ tells us the non-trivial ways a linkage with fixed-length bars can move. Translation and rotation are considered trivial motions.

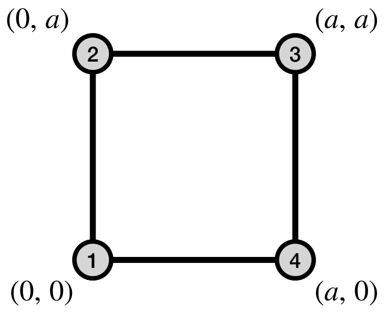


Figure 1.2: A square linkage with vertex positions $(0,0)$, $(0,a)$, (a,a), and $(a,0)$.

Throughout this section, we will look at a simple linkage sometimes called the four-bar linkage. We first consider a version of it with four rigid bars of equal length with vertex positions shown in Fig. 1.2. The length constraints for this linkage are

$$\begin{aligned}
|\mathbf{p}_1 - \mathbf{p}_2|^2 &= a^2 \\
|\mathbf{p}_2 - \mathbf{p}_3|^2 &= a^2 \\
|\mathbf{p}_3 - \mathbf{p}_4|^2 &= a^2 \\
|\mathbf{p}_4 - \mathbf{p}_1|^2 &= a^2,
\end{aligned} \tag{1.3}$$

and differentiating them gives

$$\begin{aligned}(\mathbf{p}_1-\mathbf{p}_2)\cdot(\mathbf{p}_1'-\mathbf{p}_2') &= [(0,0)-(0,a)]\cdot[(x_1',y_1')-(x_2',y_2')] = -ay_1'+ay_2' = 0\\ (\mathbf{p}_2-\mathbf{p}_3)\cdot(\mathbf{p}_2'-\mathbf{p}_3') &= [(0,a)-(a,a)]\cdot[(x_2',y_2')-(x_3',y_3')] = -ax_2'+ax_3' = 0\\ (\mathbf{p}_3-\mathbf{p}_4)\cdot(\mathbf{p}_3'-\mathbf{p}_4') &= [(a,a)-(a,0)]\cdot[(x_3',y_3')-(x_4',y_4')] = ay_3'-ay_4' = 0\\ (\mathbf{p}_4-\mathbf{p}_1)\cdot(\mathbf{p}_4'-\mathbf{p}_1') &= [(a,0)-(0,0)]\cdot[(x_4',y_4')-(x_1',y_1')] = ax_4'-ax_1' = 0.\end{aligned} \tag{1.4}$$

1.1.1 Rigidity Matrix

To solve for the infinitesimal flexes, we rewrite Eq. 1.2 as a matrix equation as

$$R\mathbf{p}' = 0 \tag{1.5}$$

to construct the **rigidity matrix** $R \in \mathbb{R}^{m\times dn}$ where n is the number of vertices and m is the number of bars. The infinitesimal flexes live in the null-space of R, which is referred to as the **configuration space** of the linkage. For the four-bar linkage, the rigidity matrix equation is

$$\begin{bmatrix} 0 & -a & 0 & a & 0 & 0 & 0 & 0\\ 0 & 0 & -a & 0 & a & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & a & 0 & -a\\ -a & 0 & 0 & 0 & 0 & 0 & a & 0\end{bmatrix}\begin{bmatrix} x_1'\\ y_1'\\ x_2'\\ y_2'\\ x_3'\\ y_3'\\ x_4'\\ y_4'\end{bmatrix} = 0. \tag{1.6}$$

This initially has a solution of

$$
\begin{aligned}
y_2' &= y_1' \\
x_3' &= x_2' \\
y_3' &= y_4' \\
x_4' &= x_1'.
\end{aligned}
\tag{1.7}
$$

If we pin $\mathbf{p}_1$ and $\mathbf{p}_4$ by setting $x_1' = y_1' = x_4' = y_4' = 0$, we remove the translation and rotation motions of the linkage, and this solution reduces down to

$$y_2' = y_3' = 0 \ \ , \ \ x_2' = x_3' \tag{1.8}$$

which corresponds to the top half of the linkage shifting to the side as shown in Fig. 1.3.

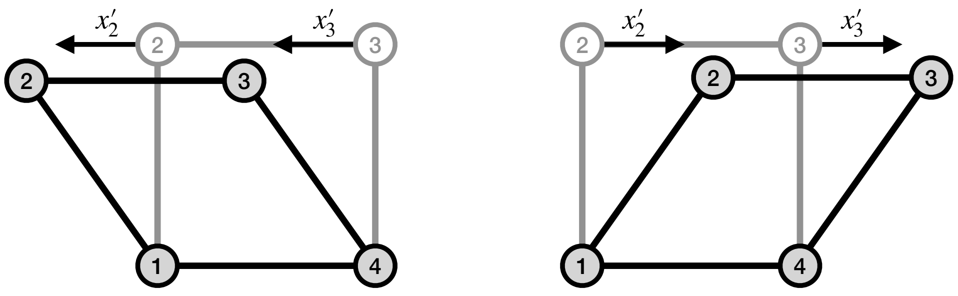


Figure 1.3: Vertices 2 and 3 can both move horizontally in the same direction. This results in the top half of the linkage moving to the left or right.

1.1.2 Types of Rigidity

Although we think have good intuition for identifying if a linkage is rigid or not, that intuition only applies to a specific kind of rigidity/flexibility. There are multiple definitions of rigid, each with a corresponding definition of a possible flex or motion.

The infinitesimal flexes described above are also called **first order flexes** due to them being a first derivative of the length constraints. A linkage is **first order rigid** if every possible first order flex is trivial [3, 4]. If we take a second derivative of the length constraints, we get

$$(\mathbf{p}_i' - \mathbf{p}_j') \cdot (\mathbf{p}_i' - \mathbf{p}_j') + (\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}_i'' - \mathbf{p}_j'') = 0 \tag{1.9}$$

where $\mathbf{p}_i'' \in \mathbb{R}^d$. A **second order flex** is a set of vectors $(\mathbf{p}', \mathbf{p}'')$, where $\mathbf{p}'$ is a first order flex, that satisfies Eq. 1.1 for all bars in the linkage. A second order flex $(\mathbf{p}', \mathbf{p}'')$ is **trivial** if $\mathbf{p}'$ is a trivial first order flex. A linkage is **second order rigid** if every possible second order flex is trivial [109].

Continuously differentiating Eq. 1.1 k times give the equation

$$\sum_{a=0}^{k} \binom{k}{a} (\mathbf{p}_i^{(a)} - \mathbf{p}_j^{(a)}) \cdot (\mathbf{p}_i^{(k-a)} - \mathbf{p}_j^{(k-a)}) = 0. \tag{1.10}$$

An n^{th} **order flex** is a set of vectors $(\mathbf{p}', \mathbf{p}'', \ldots, \mathbf{p}^{(n)})$, where $\mathbf{p}', \mathbf{p}'', \ldots, \mathbf{p}^{(n-1)}$ are flexes of their respective order, that satisfy Eq. 1.10 for all $k = 1, 2, \ldots, n$ and for all bars in the linkage. An n^{th} order flex $(\mathbf{p}', \mathbf{p}'', \ldots, \mathbf{p}^{(n)})$ is **trivial** if $\mathbf{p}'$ is a trivial first order flex. A linkage is n^{th} **order rigid** if every possible n^{th} order flex is trivial [19, 22, 20].

Within the context of this dissertation, we won't discuss the definitions and implications of higher order rigidity. Going forward, the definition of an infinitesimal/first order flex will be the most important and most used.

1.1.3 Self Stresses

A **self stress** is a row redundancy in the rigidity/compatibility matrix. Physically, it's the amount of stress that each bar can hold that keeps each vertex at equilibrium. A positive stress can be thought of as the bar "pulling" on the vertices it is attached to, and a negative stress can be thought of as "pushing" on its vertices. If we apply a stress $\sigma_{(i,j)}$ to the bar connecting vertices i and j, we can write the constraint that vertex i is at equilibrium as

$$\sum_j \sigma_{i,j}(\mathbf{p}_i - \mathbf{p}_j) = 0. \tag{1.11}$$

We represent self stresses with the stress vector σ which has one component per bar in the linkage. It follows that

$$\sigma^T \mathbf{R} = 0 \tag{1.12}$$

where R is the rigidity matrix for the linkage [109]. When determining if a linkage has a self stress, the position of the vertices in the linkage is important. The rigidity matrix for a linkage will be different depending on the starting positions of the linkage. For example, in the position shown in Fig. 1.2, there are no row redundancies in our rigidity matrix from Eq. 1.6, so that configuration of the linkage does not support a self stress.

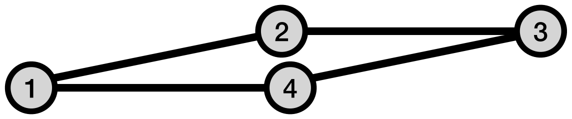


Figure 1.4: An almost flattened four bar linkage. In the fully flattened position, the bars would all be colinear. The vertex positions in the flattened position are: $\mathbf{p}_1 = (0,0)$, $\mathbf{p}_2 = (a,0)$, $\mathbf{p}_3 = (2a,0)$, $\mathbf{p}_4 = (a,0)$.

If we instead look at the linkage when it is in the configuration shown in Fig 1.4, we calculate the following rigidity matrix:

$$\begin{aligned}
(\mathbf{p}_1 - \mathbf{p}_2) \cdot (\mathbf{p}_1' - \mathbf{p}_2') &= [(0,0) - (a,0)] \cdot [(x_1', y_1') - (x_2', y_2')] = -ax_1' + ax_2' = 0 \\
(\mathbf{p}_2 - \mathbf{p}_3) \cdot (\mathbf{p}_2' - \mathbf{p}_3') &= [(a,0) - (2a,0)] \cdot [(x_2', y_2') - (x_3', y_3')] = -ax_2' + ax_3' = 0 \\
(\mathbf{p}_3 - \mathbf{p}_4) \cdot (\mathbf{p}_3' - \mathbf{p}_4') &= [(2a,0) - (a,0)] \cdot [(x_3', y_3') - (x_4', y_4')] = ax_3' - ax_4' = 0 \\
(\mathbf{p}_4 - \mathbf{p}_1) \cdot (\mathbf{p}_4' - \mathbf{p}_1') &= [(a,0) - (0,0)] \cdot [(x_4', y_4') - (x_1', y_1')] = ax_4' - ax_1' = 0.
\end{aligned} \tag{1.13}$$

$$\begin{bmatrix} -a & 0 & a & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -a & 0 & a & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & 0 & -a & 0 \\ -a & 0 & 0 & 0 & 0 & 0 & a & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.14}$$

The rows of this rigidity matrix are not independent, therefore the linkage has a self stress in this configuration. Now let's calculate the self stresses of the four bar linkage. Expanding out Eq. 1.11 for each vertex gives

$$\begin{aligned} \sigma_{(1,2)}(\mathbf{p}_1 - \mathbf{p}_2) + \sigma_{(1,4)}(\mathbf{p}_1 - \mathbf{p}_4) = 0 \\ \sigma_{(2,1)}(\mathbf{p}_2 - \mathbf{p}_1) + \sigma_{(2,3)}(\mathbf{p}_2 - \mathbf{p}_3) = 0 \\ \sigma_{(3,2)}(\mathbf{p}_3 - \mathbf{p}_2) + \sigma_{(3,4)}(\mathbf{p}_3 - \mathbf{p}_4) = 0 \\ \sigma_{(4,3)}(\mathbf{p}_4 - \mathbf{p}_3) + \sigma_{(4,1)}(\mathbf{p}_4 - \mathbf{p}_1) = 0 \end{aligned} . \tag{1.15}$$

By plugging in the vertex positions and simplifying, we see that the self stress is defined by

$$\sigma_{(1,2)} = -\sigma_{(2,3)} = -\sigma_{(3,4)} = \sigma_{(4,1)} \tag{1.16}$$

where all stress values are the same magnitude, $\sigma_{(1,2)}$ and $\sigma_{(4,1)}$ have opposite signs as $\sigma_{(2,3)}$ and $\sigma_{(3,4)}$.

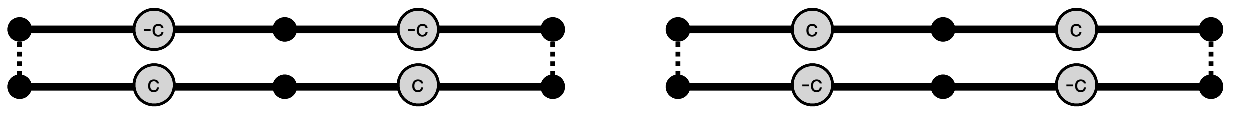


Figure 1.5: The flattened four bar linkage with stress values labeled on each bar. Dotted lines indicate vertices that are connected. Bars have been drawn on two levels for visualization purposes.

1.1.4 Configuration Space

As mentioned earlier, the configuration space of a linkage is the space that contains the set of vectors $\mathbf{p}'$ that satisfies Eq. 1.1 for each bar in the linkage. Instead of tracking the positions of each vertex in a linkage to determine how its moving, we can usually parameterize the motion using a much smaller number of variables.

The dimension of the configuration space is the number of degrees of freedom of the linkage. The configuration space lives in a space with dimension equal to the number of variables that parameterize its motion. For a one degree of freedom linkage, the configuration space will be a line/curve in space. It is possible that this line intersects with itself and has locations where you see multiple paths extending out from a single point. This point is called a **critical point** or **branch point**. Each path extending out from the branch point represents a unique infinitesimal motion. It turns out that these branch points correspond to configurations of the linkage that support self stresses.

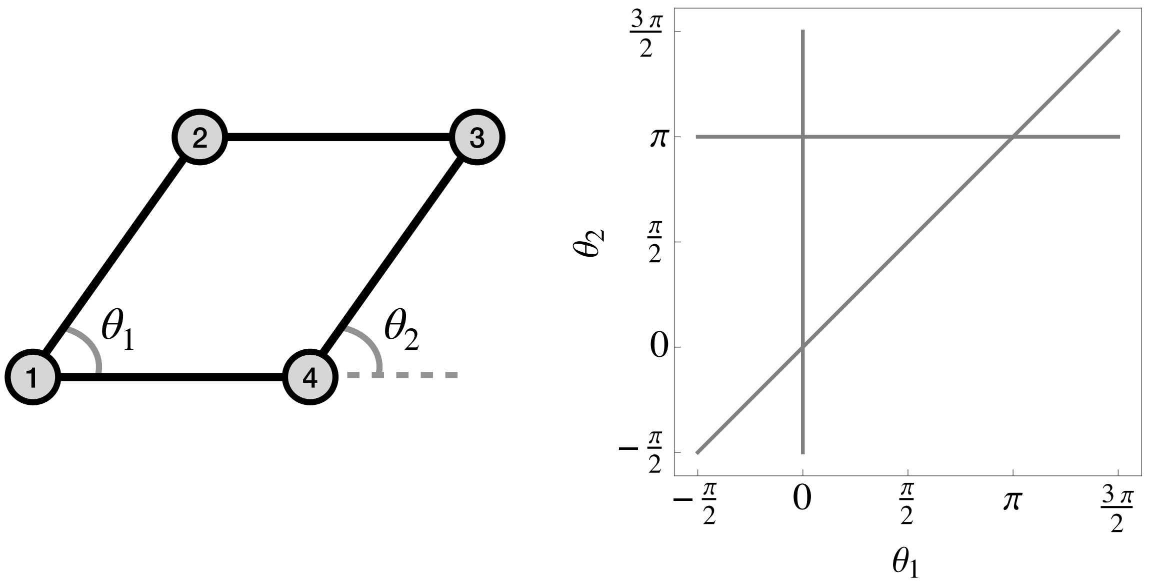


Figure 1.6: The motion of a four bar linkage can be parameterized by two angles θ_1 and θ_2. The configuration space consists of straight branches and has three branch points, each with two intersecting branches.

The four bar linkage is parameterized by two angles θ_1 and θ_2. Without loss of generality, we can pin vertices 1 and 4 so that the bar connecting them doesn't move. This will remove any trivial flexes. Using the variable change $(x_2, y_2) = (a\cos\theta_1, a\sin\theta_1)$, $(x_3, y_3) = (a\cos\theta_2 + a, a\sin\theta_2)$, we can rewrite the length constraint equations as follows.

$$\begin{aligned}
|\mathbf{p}_1 - \mathbf{p}_2|^2 &= |(0,0) - (a\cos\theta_1, a\sin\theta_1)|^2 = a^2(\cos\theta_1{}^2 + \sin\theta_1{}^2) = a^2(1) = a^2 \\
|\mathbf{p}_2 - \mathbf{p}_3|^2 &= |(a\cos\theta_1, a\sin\theta_1) - (a\cos\theta_2 + a, a\sin\theta_2)|^2 \\
&= a^2(\cos\theta_1 - \cos\theta_2 - 1)^2 + a^2(\sin\theta_2 - \sin\theta_2)^2 \qquad (1.17) \\
|\mathbf{p}_3 - \mathbf{p}_4|^2 &= |(a\cos\theta_2, a\sin\theta_2) - (a,0)|^2 = a^2(\cos\theta_1{}^2 + \sin\theta_1{}^2) = a^2(1) = a^2 \\
|\mathbf{p}_4 - \mathbf{p}_1|^2 &= |(a,0) - (0,0)|^2 = a^2
\end{aligned}$$

All length constraints quickly simplify to equal a^2 except for the one on the length of the bar connecting vertices 2 and 3. This equation can be plotted parametrically to generate the configuration space shown in Fig. 1.6.

This was a very simple example of generating a configuration space, but as the linkage gets more complex, it gets more unwieldy to do this by hand. The *mechanisms* package developed for Mathematica has many built in functions that define linkage objects using vertex positions and information about how they are connected and calculate things like the length constraint equations automatically. For one degree of freedom linkages with a configuration space that lives in three dimensions, things get even more difficult. Methods for plotting those configuration spaces are discussed in Chapter 5 and Appendix A.

1.1.5 Bar Length Perturbations

A general approach to investigating the behavior of linkages was to allow the bar lengths to vary slightly. To describe how small length changes in the rigid bars of the linkage affect the possible infinitesimal motions of the linkage, we apply a small displacement to each vertex that does *not* preserve the length of the bars.

$$\mathbf{p}_n = \mathbf{p}_n^{(0)} + \mathbf{p}_n^{(1)} + \mathbf{p}_n^{(2)} \quad , \quad L_{(i,j)}^2 = L_{(i,j)}^{2(0)} - \delta\ell_{(i,j)}^{(1)} + \delta\ell_{(i,j)}^{(2)} \tag{1.18}$$

1.1.5.1 Length Constraints

With these new definitions for the vertex position and bar length, we can rewrite the length constraint equations.

$$\begin{aligned} |\mathbf{p}_i - \mathbf{p}_j|^2 =& |(\mathbf{p}_i^{(0)} + \mathbf{p}_i^{(1)} + \mathbf{p}_i^{(2)}) - (\mathbf{p}_j^{(0)} + \mathbf{p}_j^{(1)} + \mathbf{p}_j^{(2)})|^2 \\ =& |(\mathbf{p}_i^{(0)} - \mathbf{p}_j^{(0)}) + (\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}) + (\mathbf{p}_i^{(2)} - \mathbf{p}_j^{(2)})|^2 = L_{(i,j)}^{2(0)} + \delta\ell_{(i,j)}^{(1)} + \delta\ell_{(i,j)}^{(2)} \end{aligned} \tag{1.19}$$

If we expand out the above equation and keep only first and second order terms, the new length constraints simplify down to

$$|\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}|^2 + 2(\mathbf{p}_i^{(0)} - \mathbf{p}_j^{(0)}) \cdot (\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}) + 2(\mathbf{p}_i^{(0)} - \mathbf{p}_j^{(0)}) \cdot (\mathbf{p}_i^{(2)} - \mathbf{p}_j^{(2)}) = \delta\ell_{(i,j)}^{(1)} + \delta\ell_{(i,j)}^{(2)}. \tag{1.20}$$

From the definition of the rigidity matrix $(\mathbf{p}_i - \mathbf{p}_j) \cdot (\mathbf{p}'_i - \mathbf{p}'_j) = \mathbf{R}\mathbf{p}'$, we can make the substitution

$$\begin{aligned} 2(\mathbf{p}_i^{(0)} - \mathbf{p}_j^{(0)}) \cdot (\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}) = 2[\mathbf{R}\mathbf{p}^{(1)}]_{(i,j)} \\ 2(\mathbf{p}_i^{(0)} - \mathbf{p}_j^{(0)}) \cdot (\mathbf{p}_i^{(2)} - \mathbf{p}_j^{(2)}) = 2[\mathbf{R}\mathbf{p}^{(2)}]_{(i,j)} \end{aligned} \tag{1.21}$$

so that the length constraint equation becomes

$$2[\mathbf{R}\mathbf{p}^{(1)}]_{(i,j)} + 2[\mathbf{R}\mathbf{p}^{(2)}]_{(i,j)} + |\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}|^2 = \delta\ell_{(i,j)}^{(1)} + \delta\ell_{(i,j)}^{(2)}. \tag{1.22}$$

where $\mathbf{R}$ is the rigidity matrix.

1.1.5.2 Self-Stresses

Next, we use our perturbed length constraints with the self-stress equations. Multiplying Eq. 1.18 by a self stress vector σ^T gives

$$\sigma^T |\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}|^2 = \sigma^T \delta\ell_{(i,j)}^{(1)} + \sigma^T \delta\ell_{(i,j)}^{(2)} \tag{1.23}$$

since $\sigma^T \mathbf{R} = \mathbf{0}$. This separates into the first and second order equations

$$\sigma^T \delta\ell_{(i,j)}^{(1)} = 0 \;\;\rightarrow\;\; \sum_{(i,j)} \sigma_{(i,j)} \delta\ell_{(i,j)}^{(1)} = 0 \tag{1.24}$$

$$\sigma^T |\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}|^2 = \sigma^T \delta\ell_{(i,j)}^{(2)} \;\;\rightarrow\;\; \sum_{(i,j)} \sigma_{(i,j)} |\mathbf{p}_i^{(1)} - \mathbf{p}_j^{(1)}|^2 = \sum_{(i,j)} \sigma_{(i,j)} \delta\ell_{(i,j)}^{(2)} \tag{1.25}$$

This simplification relies on the assumption that there is a self-stress for the rigidity matrix and that it works at both first and second order.

1.1.5.3 Example: Three-Bar Colinear Linkage

We will apply these deformations to a simple example linkage to demonstrate the information we can learn about the motion of the linkage. Start with a linkage of three vertices on a line, labeled 1, 2, and 3, with vertex positions $\mathbf{p}_1 = (0,0)$, $\mathbf{p}_2 = (1,0)$, and $\mathbf{p}_3 = (2,0)$. Each vertex is connected to the other two, forming a flattened triangle.

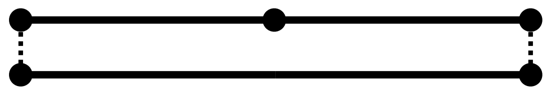


Figure 1.7: A linkage with three connected bars that can be thought of as a flattened triangle. Dotted lines indicate vertices that are connected. Bars have been drawn on two levels for visualization purposes.

Calculating Stresses Using Eq. 1.11, we can write out the system of equations needed to calculate the self stresses for this linkage.

$$\begin{aligned}
\sigma_{(1,2)}(\mathbf{p}_1-\mathbf{p}_2)+\sigma_{(1,3)}(\mathbf{p}_1-\mathbf{p}_3) &= \sigma_{(1,2)}(-1,0)+\sigma_{(1,3)}(-2,0)=0\\
\sigma_{(2,1)}(\mathbf{p}_2-\mathbf{p}_1)+\sigma_{(2,3)}(\mathbf{p}_2-\mathbf{p}_3) &= \sigma_{(2,1)}(1,0)+\sigma_{(2,3)}(-1,0)=0\\
\sigma_{(3,1)}(\mathbf{p}_3-\mathbf{p}_1)+\sigma_{(3,2)}(\mathbf{p}_3-\mathbf{p}_2) &= \sigma_{(3,1)}(2,0)+\sigma_{(3,2)}(1,0)=0
\end{aligned} \tag{1.26}$$

This system of equations simplifies to $\sigma_{(1,2)} = \sigma_{(2,3)} = -2\sigma_{(1,3)}$. We now have two solutions, one with $\sigma_{(1,2)} = \sigma_{(2,3)} > 0$ and one with $\sigma_{(1,2)} = \sigma_{(2,3)} < 0$. For simplicity, we will pick $\sigma_{(1,2)} = 2$, $\sigma_{(2,3)} = 2$, and $\sigma_{(1,3)} = -1$.

Applying Stresses Expanding Eq. 1.24 and substituting in our self stress values gives

$$\begin{aligned}
\sigma_{(1,2)}\delta\ell^{(1)}_{(1,2)}+\sigma_{(2,3)}\delta\ell^{(1)}_{(2,3)}+\sigma_{(1,3)}\delta\ell^{(1)}_{(1,3)} &= 2\delta\ell^{(1)}_{(1,2)}+2\delta\ell^{(1)}_{(2,3)}-\delta\ell^{(1)}_{(1,3)}\\
&= 2\left(\delta\ell^{(1)}_{(1,2)}+\delta\ell^{(1)}_{(2,3)}\right)-\delta\ell^{(1)}_{(1,3)}=0.
\end{aligned} \tag{1.27}$$

This equation tells us that if we want to preserve the self stresses, the net change in square of the length of the 'top half' of the linkage (the two shorter bars) will be twice as large as the change of the 'bottom half' (the one longer bar), and that both changes can be either increases or decreases in length. Including just the first order length changes, the length squared of each bar is written as

$$L^2_{(1,2)} = 1+\delta\ell^{(1)}_{(1,2)},\;\; L^2_{(2,3)} = 1+\delta\ell^{(1)}_{(2,3)},\;\; L^2_{(1,3)} = 4+\delta\ell^{(1)}_{(1,3)}. \tag{1.28}$$

Taking the square root and expanding up to first order gives us new lengths of each bar.

$$L_{(1,2)} \approx 1+\frac{\delta\ell^{(1)}_{(1,2)}}{2},\;\; L_{(2,3)} \approx 1+\frac{\delta\ell^{(1)}_{(2,3)}}{2},\;\; L_{(1,3)} \approx 2+\frac{\delta\ell^{(1)}_{(1,2)}+\delta\ell^{(1)}_{(2,3)}}{2} \tag{1.29}$$

Each new length must still satisfy $L_{(1,2)} + L_{(2,3)} = L_{(1,3)}$. This tells us that for first order changes in the length of the bars, the vertices will stay on the same line, but the middle vertex may shift away from the midpoint.

For the second order equation, we need to think of how the linkage could move/change as bars stretch or shrink. Any changes that keep the vertices on the same line are covered by the first order equation. Because of that, we can focus on $\mathbf{p}_n^{(1)}$'s that only add a vertical change in position.

Any combination of moving one or two of the three vertices vertically will turn the linkage into a very flat triangle, with the possibility of a rotation away from the horizontal axis. The most general vertical shift we can consider is $\mathbf{p}_1^{(1)} = 0$, $\mathbf{p}_2^{(1)} = y\hat{\mathbf{y}}$, and $\mathbf{p}_3^{(1)} = 0$. Any other change in the location of the vertices can be absorbed into a translational/rotational motion. Using these values, the second order equation simplifies to

$$4y^2 = 2\left(\delta\ell^{(2)}_{(1,2)} + \delta\ell^{(2)}_{(2,3)}\right) - \delta\ell^{(2)}_{(1,3)} \tag{1.30}$$

The second order changes in length will cause the middle vertex to pop up or down in addition to any translational/rotational motions or shifts in the location of the middle vertex. If $y = 0$, then this equation looks exactly like the first order equation and will reproduce the first order results.

1.1.6 Summary

This section provided a brief summary of linkages, focusing on the topics related to the work presented in this dissertation. The goal was to provide the background necessary to follow the analysis presented in Chapters 4 and 5. We also presented a new method for analyzing the behavior of linkages when their bar lengths change slightly that is utilized in Chapter 2.

1.2 Computation

We will now transition to a discussion on computation. If we want to design mechanical devices that compute, we need to fully define what we mean by the term "compute". First, we outline multiple possible definitions. Once we have a list of options, we discuss which ones we intend to use based on their compatibility with mechanical systems. The section starts off by using the vocabulary of computation theory [95] before transitioning to talking more generally about how we can integrate those definitions into the language of linkages.

1.2.1 Definitions of "Compute"

1.2.1.1 Finite State Machines

A **finite state machine** or **finite automaton** is the simplest model of a system that can compute. Some examples of systems that are finite state machines are vending machines, elevators, traffic lights, combination locks, and coin-operated turnstiles. The finite state machine is defined by a set of states the system can be in, a list of possible transitions between states, and the rules for how each transition is triggered. We also should define an initial state that the system will start in.

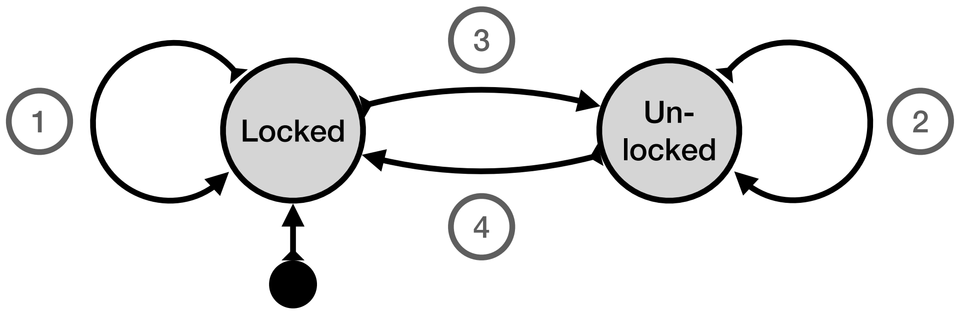


Figure 1.8: The state chart of a coin operated turnstile. Each transition is given a label, and the details of each transition are shown in Table 1.2.1.1. The black circle represents the "entrance" to the system. For this specific diagram, it represents a person walking up to a locked turnstile.

Label	Current State	Input	Next State
1	Locked	push turnstile	Locked
2	Un-locked	insert coin	Un-locked
3	Locked	insert coin	Unlocked
4	Un-locked	push turnstile	Locked

Table 1.1: Coin-operated turnstile transitions.

The coin-operated turnstile example shown in Fig. 1.8 is a very simple example of a finite state machine. As the device or system gets more complex, the state diagram and corresponding transition rules get more complicated and detailed. For example, if we wanted to describe a preheating oven as a finite state machine, that would involve using state diagrams nested within states of a larger diagram, defining transitions based on multiple variables, and utilizing parallel state diagrams that send information to each other.

1.2.1.2 Turing Machines

In the most general sense, a **Turing Machine** is a system that can simulate any computer algorithm. Anything that a 'computer' can do a Turing machine can do. Therefore, relating any system constructed to a Turing machine gives us information about how much computing ability that system has.

The most basic example of a Turing machine is a machine that reads symbols on a strip of tape and edits them, changes state, and move along the tape according to a set of instructions. Mathematically, a Turing machine is defined by some function $\delta(q, a)$ where q is the state the machine is in and a is what the tape says at the machine's current location.

$$\delta(q, a) = (r, b, L) \tag{1.31}$$

When δ takes the input q and a, it tells the machine to change the tape to b at the current location, switch to state r, and then move to the left (right if there is a R). All of the information about how the Turing machine works is contained in the transition function δ. The set of the current state, tape contents, and machine location corresponds to a configuration of the machine.

The input is a string of 1's and 0's. The machine has some set of instruction for what to do if it is in a specific state and sees a specific number at its current location on the string. The machine will continue to 'follow instructions' and do whatever the algorithm tell it to do with that string of numbers until it stops for whatever reason and accepts or rejects the output. After the machine has followed all instructions and stopped doing things, the string will be a new set of 1's and 0's. This new string is considered the output. The unique properties of a Turing machine are as follows:

- A Turing machine can read and write on the tape.
- The machine can move in both directions along the tape.
- The tape is infinite.
- The condition for accepting/rejecting the tape happen instantaneously.

A basic example of a Turing machine is presented below.

Current State	Symbol Read	Write Instruction	Move Instruction	Next State
0	blank	write blank	L	1
	b	write a	R	1
	a	write b	R	0
1	blank	write blank	R	stop
	b	write a	L	1
	a	write b	L	1

Table 1.2: Turing Machine Example.

The machine will start at a spot, move around and change symbols based on the table above, and eventually enter the 'stop' state when it is done. This is considered to be the machine completing a calculation.

A system is **Turing complete** if it can simulate any Turing machine. A common method for proving a system is turing complete is to map a basic Turing machine example to the target system. With this method, it has been shown that a wide variety of system are considered Turing complete including unconventional ones such as Magic: The Gathering and Minesweeper [18, 56].

1.2.1.3 Boolean Logic

Boolean Logic is built around two Boolean values TRUE and FALSE. These values are commonly represented as 1 and 0, high and low, or yes and no. **Boolean operation** are operations that act on and manipulate the two Boolean values. Boolean operations take in some number of input values, and output a single value. NOT is defined as replacing the current value with the other value. AND will output TRUE when both input values are TRUE. OR will output TRUE when at least one input is TRUE.

NOT	AND	OR
$\neg 0 = 1$	$0 \wedge 0 = 0$	$0 \vee 0 = 0$
$\neg 1 = 0$	$0 \wedge 1 = 0$	$0 \vee 1 = 1$
	$1 \wedge 0 = 0$	$1 \vee 0 = 1$
	$1 \wedge 1 = 1$	$1 \vee 1 = 1$

Table 1.3: The basic Boolean logic operations acting on all possible input combinations. The three symbols $\neg$, $\wedge$, and $\vee$ represent the three operations.

The set of **secondary Boolean operations** can be constructed as combinations of the basic operations. XOR, called "exclusive or" will output TRUE when only one of the inputs are TRUE. The NAND, NOR, and XNOR operations are the direct negation of the AND, OR, and XOR operations.

A system is **Functionally complete** if it can create all possible truth tables for all six logic operations AND, OR, NAND, NOR, XOR, XNOR. It turns out that we can generate the 6 necessary truth tables using only {AND, NOT}, {NAND}, or {NOR}. Creating any

NAND	NOR	XOR	XNOR
$\neg(0 \wedge 0) = 1$	$\neg(0 \vee 0) = 1$	$0 \oplus 0 = 0$	$\neg(0 \oplus 0) = 1$
$\neg(0 \wedge 1) = 1$	$\neg(0 \vee 1) = 0$	$0 \oplus 1 = 1$	$\neg(0 \oplus 1) = 0$
$\neg(1 \wedge 0) = 1$	$\neg(1 \vee 0) = 0$	$1 \oplus 0 = 1$	$\neg(1 \oplus 0) = 0$
$\neg(1 \wedge 1) = 0$	$\neg(1 \vee 1) = 0$	$1 \oplus 1 = 0$	$\neg(1 \oplus 1) = 1$

Table 1.4: The secondary Boolean logic operation acting on all possible input combinations. The $\oplus$ symbol represents the XOR operation.

one of those sets with a given system is enough to show functional completeness of that system.

1.2.1.4 Turing Complete vs. Functionally Complete

It turns out that a Turing machine can be created using the full set of logic gates, so we can say that the set of all logic gates is Turing complete. Because all logic gates can be created with just a NAND gate, and the set of all logic gates is Turing complete, the ability for a system to create a NAND gates shows that it is Turing complete. Showing that your system can create a NAND gate is a common method for supporting the statement that your system can compute because it implies that the system is Turing complete.

1.2.2 Computation in Mechanical Systems

1.2.2.1 Transistors and Logic Gate Circuits

In electronic devices, logic gates are constructed as a circuit of wires and transistors. Instead of TRUE and FALSE, we measure a high or low voltage at a location in the circuit. Transistors used in logic gate circuits are essentially electronic switches with an ON or OFF state. Without any voltage applied to the input wire, the voltage measured at the output is low and the transistor is in the OFF state. When a voltage is applied to the input wire, the output voltage is measured as high and the transistor is in the ON state.

We can demonstrate that a system can compute by using it to create logic gates. One method is to directly design the logic gates necessary to prove functional completeness. This could involve extensive testing and revision of the design to get the behavior just right. However, we could also create logic gates by directly replicating the transistor circuits used for electronic logic gates. To do so, we would need a mechanical analog for a wire and transistor.

A mechanical transistor is a device that has two states: one where a mechanical signal can pass over it,and one where the mechanical signal is blocked. Ideally, we would be able to switch between those two states by sending a mechanical signal to some section of the transistor that initiates the transition. The mechanical wires used would need to allow some sort of signal to travel along then, and they would need to behave like electronics wires at intersections.

1.2.2.2 Finite State Machines

A more abstract way to show that a system computes is to show that it can be mapped to a finite state machine. For the chosen system, we need to define a list of distinct states and ways to transition between those states. We then need to define a map between the states and transition of our system to the states and transitions of the target finite state machine.

For a linkage whose configuration space has many interconnected critical points, the states of our finite state machine would be the configurations at each critical point. Our list of which states we can transition to or from is determined by which critical points have branches connecting them. The input that triggers a transition could be a force or displacement that moves the linkage along a branch.

1.3 Dissertation Outline

In Chapter 2, we extend the work shown in Sec. 1.1 related to small length perturbations of linkages to origami systems and use those results to analyze non-Euclidean origami systems. This chapter does not relate directly to the goal of designing mechanical devices that compute, but is included in this dissertation because of its use of the linkage analysis techniques introduced in Sec. 1.1 which are then used in Chapter 4 as well.

In Chapter 3, we discuss the viability of a specific mechanical wire design and present results on situation where these wires will function reliably in devices. By fully characterizing the regimes where these wires can be used in devices, we open up the opportunity to use them to design devices that can compute. Then in Chapter 4, we presents our work analyzing a specific linkage called the flip-flop and investigating how to use it as a transistor that is compatible with the mechanical wires discussed in Chapter 3. While simulations showed that the flip-flop was a promising candidate for a transistor, experiments with soft materials showed unwanted behavior in certain states. The combination of Chapters 3 and 4 represent our approach to designing mechanical devices that compute by creating mechanical wires, transistors, and logic gates.

In Chapter 5, we focus in on how to control the behavior of linkages by exploring methods for engineering the configuration space of a general linkage. Using these methods, we design a linkage that acts as a gate (transistor) for the Kane-Lubensky chain. This work contributes to both the wire and transistor path to computation as well as the more ambiguous finite state machine definition.

Chapter 2

Topological transitions in the configuration space of non-Euclidean origami

Abstract

Origami structures have been proposed as a means of creating three-dimensional structures from the micro- to the macroscale, and as a means of fabricating mechanical metamaterials. The design of such structures requires a deep understanding of the kinematics of origami fold patterns. Here, we study the configurations of non-Euclidean origami, folding structures with Gaussian curvature concentrated on the vertices, for arbitrary origami fold patterns. The kinematics of such structures depends crucially on the sign of the Gaussian curvature. As an application of our general results, we show that the configuration space of non-intersecting, oriented vertices with positive Gaussian curvature decomposes into disconnected subspaces; there is no pathway between them without tearing the origami. In contrast, the configuration space of negative Gaussian curvature vertices remain connected. This provides a new,

and only partially explored, mechanism by which the mechanics and folding of an origami structure could be controlled.

2.1 Introduction

Origami and kirigami have been proposed as a framework to engineer new materials with complex mechanical responses [33, 90, 108, 93, 12]. To this end, new fabrication methods have been developed to enable the folding of three dimensional structures from thin films [82, 11, 87]. Though most examples of origami structures are foldable from an initially flat sheet, two threads of research suggest a need to understand the motions of a broader class of "curved" origami. First, kirigami structures, initially flat structures with holes which can be glued together along their free edges to create intrinsically buckled structures [12]. Second, newer origami fabrication methods have enabled vertices with Gaussian curvature and curved faces [87, 35, 2, 6, 29].

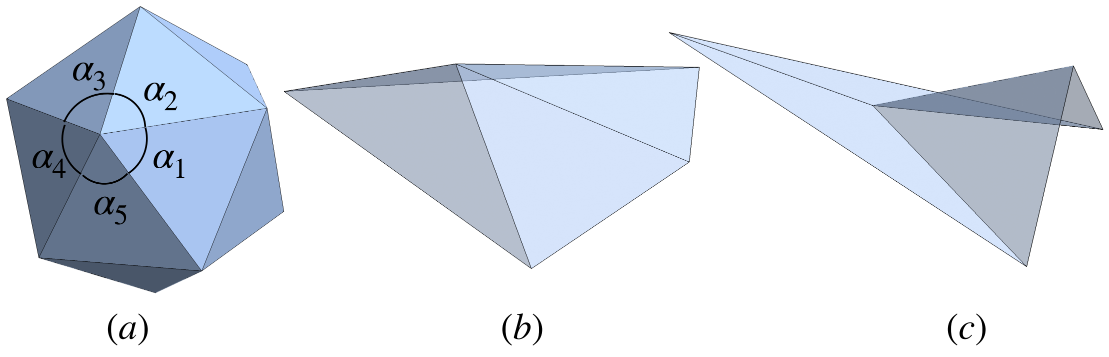


Figure 2.1: (a) A generic non-Euclidean origami structure. The vertex Gaussian curvature is defined by $K = 2\pi - \sum_i \alpha_i$ (b,c) Degree four vertices with positive and negative Gaussian curvatures respectively necessarily buckle out of the plane.

This paper analyzes the kinematics of non-Euclidean origami in the limit that it is almost flat. By "non-Euclidean origami," we mean that faces are flat, but that the vertices have Gaussian curvature (Fig. 2.1 a–c). This Gaussian curvature manifests as either a deficit or excess angle when summing the sector angles around the internal vertices (Fig. 2.1a). By "almost flat," we mean that both the sum of sector angles around internal vertices is near 2π and that the dihedral angles of the folds are nearly π. In this limit, we will develop a general framework for studying origami motions, and make contact with both the kinematics of flat origami structures [14] and continuum equations governing the small deformations of elastic sheets [92].

Understanding whether an origami fold pattern can be folded without tearing is NP-hard [1]. More generally, when mapping out the space of possible configurations of a given origami fold pattern, the configuration space can be geometrically complex. Additionally, these spaces can undergo topological changes as the fold pattern changes that lead to changes in the mechanical properties of origami [65].

Here, we show that vertex Gaussian curvature can induce a topological change in the configuration space of general origami structures. We will show that origami with positive Gaussian curvature vertices have configuration spaces that become disconnected, and that such disconnection need not (and likely does not) occur for negative Gaussian curvature. We apply our general approach to elaborate on the kinematics and energetics of single vertex origami.

2.2 Mathematical Formulation

We model origami by a collection of polygonal faces meeting at point-like vertices and joined along line-like, rigid edges, as shown in Fig. 2.1 for triangular faces. We find it useful to distinguish internal vertices, whose number we will denote V_i, from boundary vertices, whose number is V_b. Note that in traditional origami nomenclature a "vertex" denotes only the internal vertices. Similarly, we denote the internal and boundary edges by E_i and E_b, respectively. The internal edges are the folds in the origami literature.

We are primarily interested in determining the isometries of a given origami fold pattern, *i.e.* the motions that preserve the length of all edges and the angles between any two adjacent edges on the same face. In the case of triangular faces, the angle constraint is redundant – once the length of all the edges are known, the angles between edges are already uniquely determined. Thus, we will focus mainly on origami with triangular faces. This is not very restrictive; we will see that the configuration space of an origami structure with polygonal faces can be obtained by taking a lower dimensional slice through the configuration space of a suitable triangulated origami fold pattern. To define the discrete Gaussian curvature of an internal vertex, we measure the sector angles, α_i, between adjacent folds with one end on a given vertex (Fig. 2.1a). The Gaussian curvature of that vertex is then $K_n = 2\pi - \sum_i \alpha_i$ [77].

One of the primary features of triangulated origami is that the number of infinitesimal isometries is almost precisely balanced by the number of constraints. This is true for any Gaussian curvature though it manifests in different ways when $K_n = 0$ on each internal vertex. Understanding this distinction turns out to be important to developing a fuller picture of the origami configuration space so we review it here. If $\mathbf{X}_n$ denotes the three dimensional position of the n^{th} vertex, then any pair of vertices joined by an edge induces a geometrical constraint,

$$(\mathbf{X}_n - \mathbf{X}_m)^2 = L_{nm}^2, \tag{2.1}$$

where L_{nm} is the length of the edge between n and m. We then write $\mathbf{u}_n$ (Fig. 2.2b) as the displacement of the n^{th} vertex, and find that, to first order, motions are governed by the linear equations

$$(\mathbf{X}_n - \mathbf{X}_m) \cdot (\mathbf{u}_n - \mathbf{u}_m) = 0. \tag{2.2}$$

There is one equation of this type for each edge (n, m) joining vertex n to m.

To understand the generic behavior of Eq. (2.2), we note that there are $E_i + E_b$ con-

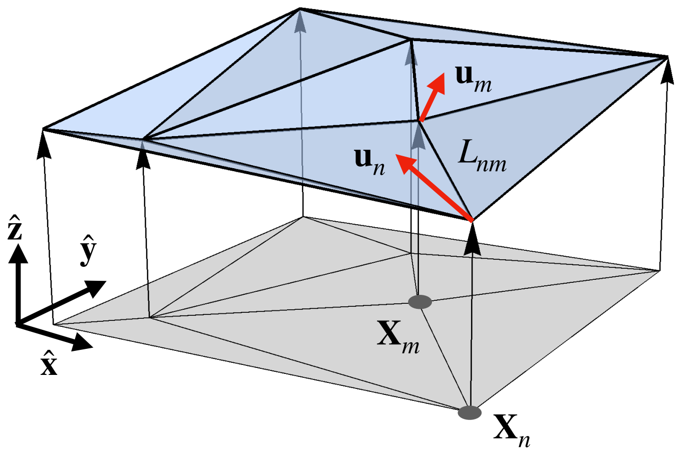


Figure 2.2: A nearly flat origami structure can be projected to a fold pattern in the $xy-$plane. In-plane and out-of-plane displacements are unambiguously decomposable.

straints, one for each edge and $3V_i + 3V_b$ naive degrees of freedom associated with the three-dimensional displacements of the vertices. A triangulated origami fold pattern also satisfies both Euler's theorem, $F - E_i - E_b + V_i + V_b = 1$, where F is the number of faces, and satisfies the $2E_i + E_b = 3F$ to account for the fact that each face is associated to three edges but each internal edge joins two faces. Similarly, we have $E_b = V_b$ because the boundary of the fold pattern is a polygon. Taken together, these equations imply $E_i = V_b + 3V_i - 3$ and so naive counting suggests that the dimension of the configuration space of origami is $D = V_b + 3$. Six of these degrees of freedom are Euclidean motions.

Though this generic counting should be valid for most configurations, it fails when the origami is flat because the constraints at first order are not all independent. In that case, only the in-plane deformations are fixed by the length constraints: any vertex can be displaced vertically without causing a first-order change in the edge lengths. Though this suggests that $D = V_i + V_b + 3$, it turns out that there are additional constraints at quadratic order in the lengths, If we define $\mathbf{h} = (h_1, h_2, \cdots)$ as a vector specifying the vertical displacement of each of the vertices above the $xy-$plane, then a necessary and sufficient condition for a motion to be an isometry to second order is

$$\mathbf{h}^T \mathbf{Q}_n \mathbf{h} = 0, \tag{2.3}$$

for each internal vertex, n, where the matrix $\mathbf{Q}_n$ depends on the sector angles of internal vertex n [14]. The left-hand side of Eq. (2.3) is the Gaussian curvature of internal vertex n induced by the height changes [14] so Eq. (2.3) is simply the statement that no infinitesimal deformation can change the Gaussian curvature of the internal vertices. There are precisely enough quadratic constraints, one for each internal vertex, to recover the generic result, $D = V_b + 3$.

We now wish to modify Eq. (2.3) to allow for internal vertices to have a small but nonzero Gaussian curvature. In this regime, the vertices continue to remain almost planar and, in Appendix A (Sec. 2.6), we show that the geometrical constraints at each vertex should be modified to

$$\mathbf{h}^T \mathbf{Q}_n \mathbf{h} = K_n, \tag{2.4}$$

where K_n is the Gaussian curvature of vertex n and $\mathbf{Q}_n$ is the same matrix that appears in Eq. (2.3) for flat origami. This is our main result and is accurate to quadratic order in the displacements. Despite the plausible form of Eq. (2.4), the proof that Eq. (2.4) correctly determines the isometries to quadratic order is somewhat involved.

Eq. (2.4) can be contrasted to the equations governing the small isometries of a continuum elastic sheet, which are governed by the approximate equations [92]

$$K = -\frac{1}{2} \sum_{ijkl=1}^{2} \epsilon_{ik}\epsilon_{jl} \partial_i \partial_j h \partial_k \partial_l h, \tag{2.5}$$

where ϵ_{ij} is the antisymmetric Levi-Civita symbol with $\epsilon_{12} = 1$, $h(x, y)$ is the vertical height of the elastic sheet above the $xy-$plane, and $K(x, y)$ is the Gaussian curvature. Eq. (2.5) is accurate in the limit of small slopes $|\partial_i h| \ll 1$, which is precisely the same limit of our discrete formulation. In that sense, Eq. (2.4) is a discrete analogue to the better known continuum result of Eq. (2.5).

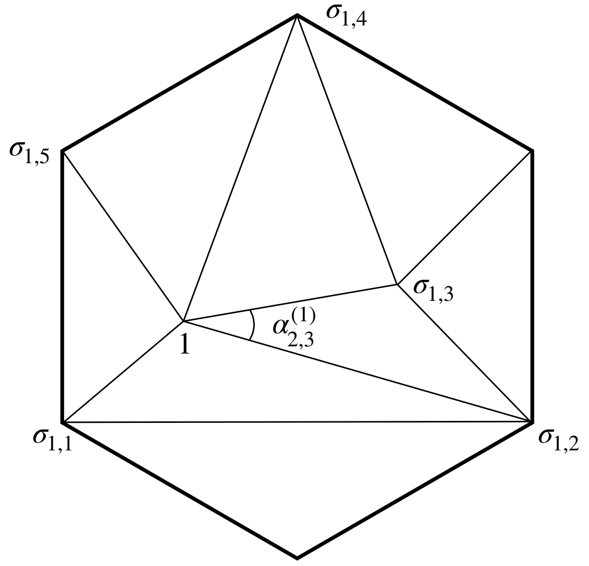


Figure 2.3: Notation for the vicinity of a single vertex.

In Appendix B (Sec. 2.7), we also show that we can rewrite the Gaussian curvature around any internal vertex n in closed form. To do this consistently requires some additional notation (Fig. 2.3). Let $\sigma(n, 1)$ through $\sigma(n, N(n))$ be the vertices connected to an internal vertex n in counterclockwise order, where $N(n)$ is the number of edges with n at one end. We also denote L_{nm} as the length of the edge joining vertex n to m. Then,

$$K_n = -\frac{1}{2}\sum_{i}^{N(n)}\sum_{j}^{N(n)} \left(\frac{h_{\sigma(n,i)} - h_n}{L_{\sigma(n,i)n}}\right) M^{(n)}_{ij} \left(\frac{h_{\sigma(n,j)} - h_n}{L_{\sigma(n,j)n}}\right). \tag{2.6}$$

The matrix $M^{(n)}_{ij}$ is an $N(n) \times N(n)$ square matrix depending on the sector angles around each internal vertex. To define $M^{(n)}_{ij}$, let $\alpha^{(n)}_{i,i+1}$ be the sector angle between vertex $\sigma(n, i)$

and $\sigma(n, i+1)$ around the vertex n. Then

$$\begin{aligned} M_{ij}^{(n)} &= -\csc\alpha_{i,i+1}^{(n)}\delta_{i,j+1} - \csc\alpha_{i-1,i}^{(n)}\delta_{i,j-1} \\ &\quad +(\cot\alpha_{i,i+1}^{(n)} + \cot\alpha_{i-1,i})^{(n)}\delta_{ij}. \end{aligned} \tag{2.7}$$

For sector angles smaller than π, the matrices $\mathbf{M}^{(n)}$ have two zero eigenvalues, one negative eigenvalue, and the remaining eigenvalues are positive (see Ref. [52] or Appendix C of Ref. [14] for a detailed proof).

2.3 Single vertices

To better understand Eq. (2.6), consider an origami structure with one internal vertex from which N folds emerge (Fig. 2.4a). This case has been analyzed in some depth due to the correspondence between origami vertices of degree N and spherical linkages with N segments [52, 100]. We denote the height of the central vertex h_0 and the heights of the surrounding vertices h_1 through h_N, and we explicitly eliminate rigid body motions by fixing the heights of vertices $(h_0, h_1, h_2) = (0, 0, 0)$. One quadratic constraint remains on the remaining heights, h_3 to h_N, leaving $N-3$ distinct degrees of freedom.

So what does the configuration space of a single non-Euclidean vertex look like? We suppose $\mathbf{e}_{-,i}$ are the components of the normalized eigenvector corresponding to the negative eigenvalue, $-\lambda_-$, of $\mathbf{M}^{(0)}$. We then suppose $\mathbf{e}_{n,i}$ are the components of the n^{th} normalized eigenvector with positive eigenvalues, λ_n. We can then attempt to solve Eq. (2.6) with the ansatz

$$\frac{h_i - h_0}{L_{i0}} = \frac{1}{\sqrt{2\lambda_-}}c_-\mathbf{e}_{-,i} + \sum_n \frac{1}{\sqrt{2\lambda_n}}c_n\mathbf{e}_{n,i}. \tag{2.8}$$

We find that $c_-^2 - \sum_n c_n^2 = K_0$, where K_0 is the Gaussian curvature of vertex 0.

When $K_0 = 0$, we recover the results of Ref. [14]: the solution forms a cone described

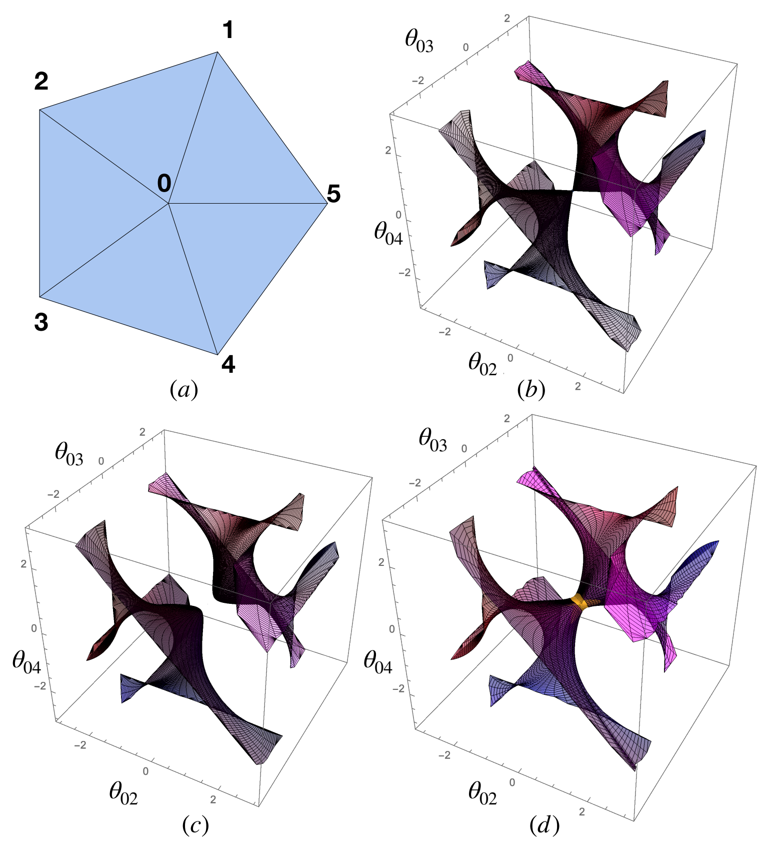


Figure 2.4: The configuration space of a symmetric five-fold vertex (a) near the flat state with zero (b), positive (c), and negative (d) Gaussian curvature projected onto the fold angles $(\theta_{02}, \theta_{03}, \theta_{04})$. The fraction of red and blue, $[r, b]$, in the coloring is determined by $[(\theta_{01} - \pi)/(2\pi), (\theta_{05} + \pi)/(2\pi)]$.

by the equation $c_- = \pm\sqrt{c_n^2}$ with a singularity at $c_- = c_n = 0$. Each nappe of the solution space is characterized by the sign of c_- (called branch signs in [14]). When $K_0 > 0$, we must instead solve,

$$c_-^2 = K_0 + \sum_n c_n^2, \tag{2.9}$$

showing that $|c_-| \geq K_0$. This would seem to imply that the two nappes have split into two disconnected components characterized by the sign of c_-. Finally, we turn to $K_0 < 0$, for which

$$c_-^2 + |K| = \sum_n c_n^2. \tag{2.10}$$

Here, it is clear that there is no obstruction to $c_- = 0$. Instead, $\sum_n c_n^2 \geq |K|$. We conclude that the conical configuration space is one in which both nappes remain connected near the flat state but are connected by a neck (Fig. 2.4d). This is quite different than what happens for degree four vertices [107].

In Fig. 2.4, we numerically plot the configuration space of a symmetric degree-5 vertex. To do this, we compute radial trajectories from a known configuration of the origami vertex. Each point of the radial trajectory is found in a sequence of steps. For each step, we solve Eq. (2.2) to identify the infinitesimal isometries from any configuration that is not flat and project the previous tangent direction onto the new tangent space. After finding a new configuration using the linear isometry, we numerically minimize the energy functional,

$$E = \frac{1}{2}\sum_{nm}\left[(\mathbf{X}_n - \mathbf{X}_m)^2 - L_{nm}^2\right]^2, \tag{2.11}$$

where the sum is over edges joining vertex n to m, using the BFGS ("QuasiNewton") algorithm in Mathematica 11. This prevents numerical errors in the linear isometries from building up as the integration proceeds. This process proceeds until one of the fold angles exceeds π or $-\pi$, indicating that a face has come into contact with an adjacent face. Finally, the trajectories are assembled into a mesh to produce a surface.

Generically, we find that the configuration space near the flat state follows the analytical results we obtained. Specifically, it appears that the configuration space decomposes into two nappes with the topology of a disk which are either touching at one point ($K = 0$), disconnected ($K > 0$), or connected by a narrow neck ($K < 0$). At first glance, this appears to contradict Streinu and Whitely [100], who showed that the configuration space of single vertices with $K_0 > 0$ is always connected. However, in their analysis, faces can pass through each other; whereas in Fig. 2.4, fold angles must remain strictly between $-\pi$ and π. It does appear that when faces are allowed to pass through each other, isometric trajectories can pass from one nappe to the other for any K. In this case, however, the surfaces become difficult to plot, even more difficult to understand, and, in any case, are unphysical.

Though our results, so far, apply only to origami with triangular faces, they can be adapted to understand the mechanics of non-Euclidean structures more generally. To start, consider a single degree-four vertex, one with only four folds emerging from a central vertex, a special case that has been recently explored in Ref. [106]. The configuration space of a degree-four vertex can be obtained from Fig. 2.4 by considering a particular planar slice. For example, if we create a degree-four vertex by removing fold θ_{02} from Fig. 2.4a, the configuration space of the degree-four vertex is the intersection of the surfaces in Fig. 2.4 with the plane $\theta_{02} = 0$. This configuration space is, therefore, one dimensional and the two nappes become disconnected for both positive and negative Gaussian curvature.

This kind of reasoning can also be used to explore the configuration spaces of non-triangulated origami. If we are given an arbitrary origami fold pattern, any non-triangular faces can be triangulated, introducing new fold angles, $(\phi_1, \cdots, \phi_M)$. The proper isometries of the non-triangulated origami are then the intersection of the triangulated configuration space with the hyperplane defined by $(\phi_1, \cdots, \phi_M) = 0$. Therefore, the dimension of the configuration space becomes $D = V_b - 3 - M$, where M is the number of diagonals added to triangulate the fold pattern. Because these hyperplanes pass through the origin (where the origami is unfolded), they do not change the connectedness of the configuration spaces of triangulated origami shown in Fig. 2.4.

The configuration spaces in Fig. 2.4b – d give us a first picture of the interplay between origami energetics and kinematics. If we imagine that a torsional spring of stiffness κ has been placed on each fold of Fig. 2.4a, the energy functional would be $E = (\kappa/2)\sum_{i=1}^{N} \theta_{nN}^2$. The equi-energy surfaces are given by spheres centered on the state with $\theta_{0i} = 0$ and so the ground state is the configuration (or configurations) that are closest to the flat state. Appendix B (Sec. 2.7) provides some mathematical machinery to expand this discussion to general origami fold patterns. In addition to determining the kinematics of an origami structure near the flat state, the matrix $M_{ij}^{(n)}$ also determines the fold angles as a function of the vertex heights through

$$\theta_{\sigma(n,i)n} = \sum_{j}^{N(n)} M_{ij}^{(n)} \left(\frac{h_{\sigma(n,j)} - h_n}{L_{\sigma(n,j)n}} \right), \tag{2.12}$$

where $\theta_{\sigma(n,i)n}$ is the fold angle connecting vertex $\sigma(n,i)$ to vertex n. Note that the quadratic terms in Eq. (2.12) actually vanish so this equation is accurate to quadratic order as well. Using Eq. (2.12) we can write an energy functional for a nearly-flat origami structure as

$$\begin{aligned} E &= \frac{1}{2} \sum_{n} \sum_{ijk}^{N(n)} \kappa_{n\sigma(n,i)} M_{ij}^{(n)} M_{ik}^{(n)} \\ &\quad \times \left(\frac{h_{\sigma(n,j)} - h_n}{L_{\sigma(n,j)n}} \right) \left(\frac{h_{\sigma(n,k)} - h_n}{L_{\sigma(n,k)n}} \right) \end{aligned} \tag{2.13}$$

where the sum over n is over internal vertices only. Any fold that joins an internal vertex n to a boundary vertex k has torsional stiffness κ_{nk} whereas a fold connecting internal vertex n to internal vertex m has stiffness $2\kappa_{nm}$ because such folds are double counted in Eq. (2.13). Thus, for a single vertex with equal fold stiffness κ and zero equilibrium fold angles, the decomposition of deformations in terms of collective variables c_- and c_n yields an energy

$$E = \frac{1}{2} \kappa \left[\lambda_- (c_-)^2 + \sum_n \lambda_n (c_n)^2 \right]. \tag{2.14}$$

In order to minimize Eq. (2.14), we found it convenient to introduce hyperbolic coordinates, ξ and a unit vector $\hat{\mathbf{N}}$ in the space of deformations, represented by the coordinates $(c_-, c_1, \cdots)$. How we do this will depend on the sign of the Gaussian curvature. For $K_0 > 0$, we introduce collective variable ξ such that $c_- = \pm K_0 \cosh \xi$ and $c_n = K_0 N_n \sinh \xi$, where N_n are the components of $\hat{\mathbf{N}}$. When $K_0 < 0$, we instead use $c_- = |K| \sinh \xi$ and $c_n = \pm |K| N_n \cosh \xi$.

Therefore,

$$E = \frac{\kappa K^2}{2} \begin{cases} \lambda_- \cosh^2\xi + \sinh^2\xi \sum_n \lambda_n (N_n)^2, & K_0 > 0, \\ \lambda_- \sinh^2\xi + \cosh^2\xi \sum_n \lambda_n (N_n)^2, & K_0 < 0. \end{cases} \tag{2.15}$$

There is an obvious generalization of Eq. (2.15) to the case when the fold stiffnesses are not all equal.

For both signs of K_0, Eq. (2.15) has a minimum at $\xi = 0$. When $K_0 > 0$, this implies $E = \kappa K^2 \lambda_- c_-^2/2$ and is independent of the choice of N_n or the values of λ_n. There are two energy minima corresponding to the two points closest to the flat state in Fig. 2.4c, independent of any other details of the shape. This reflects the fact that, at least when all folds have the same torsional stiffness, a single vertex will buckle up or down symmetrically. When $K_0 < 0$, on the other hand, the component of N_n corresponding to the smallest eigenvalue λ_n will be 1 and the remaining components will be 0. Hence, $E = \kappa K^2 \lambda_{n_{min}} c_{n_{min}}^2/2$, where n_{min} is the index of the smallest eigenvalue.

2.4 Conclusions

To conclude, we have derived the form of the configuration space of non-Euclidean origami for small amounts of Gaussian curvature near the flat state. For single positive Gaussian curvature vertices, the configuration is characterized by nappes that are separated near the flat state, whereas for negative Gaussian curvature, the configuration space remains connected. Though we have analyzed the case of a single degree-N vertex in detail, the procedure we have used can be applied to explore the kinematics and energetics of more complex, nearly flat origami structures with or without Gaussian curvature. We first consider the case of multiple vertices with $K_n > 0$. Around each vertex, Eq. (2.6) establishes a single equation for h_n as a function of the heights of the vertices surrounding it. We further assume that this equation has two distinct real solutions for h_n. Then the analysis in the previous section establishes that no matter how we deform the boundary vertices, there is no way for the configuration of this vertex to pass from one configuration space nappe to the other. The conclusion is that distinct branches of the configuration space of a complex, origami fold pattern that are distinguished by a $K > 0$ vertex being on different nappes are topologically disconnected – if they were not, there would be also be a way of passing from one nappe to the other on a single vertex. Unfortunately, it is difficult to determine whether or not every combination of nappes can be realized when $K > 0$. The case for $K < 0$ is murkier because, while a single vertex remains connected, there is no reason that global constraints might not lead to disconnected components of the configuration space. Indeed, this must be possible in principle, as triangulated fold patterns with disconnected configuration spaces, albeit rare, have been found [94].

Finally, we note that this work provides a new mechanism by which the mechanical response of an origami metamaterial sheet can be molded. In principle, an initially flat structure could be stiffened by imposing a small amount of positive Gaussian curvature. Moreover, Gaussian curvature provides a new means of controlling how a responsive origami structure self-folds by separating the individual nappes so that misfolding is significantly less likely. This suggestion will be followed up in a future work.

2.5 Acknowledgements

We acknowledge funding from the National Science Foundation under grant NSF DMR-1822638 and useful conversations with D.W Atkinson, Z. Rocklin and B. Chen. This work was done in part at the Aspen Center for Physics under grant NSF PHY-1607611.

2.6 Appendix A. Generalizing the formalism to nonzero Gaussian curvature

The problem we seek to solve in this paper lies in reconciling the linear and quadratic length-preserving motions. When vertices have Gaussian curvature, the vertices will not typically lie flat. Hence, we would expect them to be well-described by the linear equation Eq. (2.2). As the Gaussian curvature goes to zero, however, quadratic constraints must somehow emerge.

As before, we will approach the analysis of the possible motions by expanding around the flat state. We expect this expansion to be valid so long as the Gaussian curvature of the vertices is sufficiently small. Denoting the planar angles around any vertex with α_n, the discrete Gaussian curvature is $K = 2\pi - \sum_n \alpha_n$. We imagine that the deformation of a structure is governed by an expansion of the form

$$\mathbf{X}_n = \mathbf{X}_n^{(0)} + \mathbf{u}_n^{(1)} + \mathbf{u}_n^{(2)} \tag{2.16}$$

where $\mathbf{X}_n^{(0)}$ is the position of a flattened origami structure and the superscript of $\mathbf{u}$ represents the order in a formal expansion of the displacement.

Because we are expanding the deformations around an otherwise flat structure, the equilibrium lengths of the edge connecting vertex n and m will not be represented by the distances between the planar vertex positions, $\mathbf{X}_n^{(0)}$. Instead, we let $\Delta_{nm} = L_{nm}^2 - (\mathbf{X}_n^{(0)} - \mathbf{X}_m^{(0)})^2$ measure the deviation of the equilibrium edge lengths from the lengths of the edges when projected to the $xy-$plane. We denote $\mathbf{\Delta}$ the vector formed by concatenating the components Δ_{nm} for each edge. We similarly write $\mathbf{u}^{(1)}$ and $\mathbf{u}^{(2)}$ as the concatenation of the vertex displacements at first and second order. Finally, introduce a quadratic function, $\mathbf{f}(\mathbf{u})$ with components $(\mathbf{u}_n - \mathbf{u}_m)^2$ for each edge, (n, m). Then we have

$$\mathbf{\Delta} = \mathbf{R}\mathbf{u}^{(1)} + \mathbf{R}\mathbf{u}^{(2)} + \mathbf{f}(\mathbf{u}^{(1)}), \tag{2.17}$$

where $\mathbf{R}$ is the compatibility matrix mapping vertex displacements to linear changes in the edge lengths [19, 69].

To linear order, one should solve $\mathbf{\Delta} = \mathbf{R}\mathbf{u}^{(1)}$. However, this linear equation can only have a solution if the left-hand side of the equation lies in the image of $\mathbf{R}$. We denote the projection of a vector into the image of $\mathbf{R}$ with a subscript $||$, and a projection into the orthogonal complement $\perp$. Therefore, Eq. (2.17) decomposes into the pair

$$\mathbf{\Delta}_{||} = \mathbf{R}\mathbf{u}^{(1)} + \mathbf{R}\mathbf{u}^{(2)} + \mathbf{f}_{||}(\mathbf{u}^{(1)}) \tag{2.18}$$

$$\mathbf{\Delta}_{\perp} = \mathbf{f}_{\perp}(\mathbf{u}^{(1)}) \tag{2.19}$$

Eq. (2.18) can now be solved order by order. To first order, $\mathbf{u}^{(1)} = \mathbf{u}_{||} + \mathbf{h}$, where $\mathbf{u}_{||}$ is any solution of $\mathbf{R}\mathbf{u}_{||} = \mathbf{\Delta}_{||}$, and $\mathbf{h}$ is in the right null space of $\mathbf{R}$. At the next order, we obtain a correction $\mathbf{R}\mathbf{u}^{(2)} = -\mathbf{f}_{||}(\mathbf{u}_{||} + \mathbf{h})$.

Since we are expanding around a flat origami structure, we can further restrict the structure of $\mathbf{u}_{||}$ and $\mathbf{h}$. Particularly, it must be that $\mathbf{h}$ can only involve the three in-plane Euclidean motions and the vertical displacements of all of the vertices. Consequently, $\mathbf{u}_{||}$ can be chosen so that the vertex displacements lie in the $xy-$plane and $\mathbf{h}$ can then contain only vertex displacements along the $\hat{\mathbf{z}}$.

Eq. (2.19) is not dispensed with so easily. It remains a quadratic constraint on $\mathbf{h}$ of the form

$$\mathbf{\Delta}_{\perp} = \mathbf{f}_{\perp}(\mathbf{u}_{||} + \mathbf{h}) = \mathbf{f}_{\perp}(\mathbf{u}_{||}) + \mathbf{f}_{\perp}(\mathbf{h}). \tag{2.20}$$

The last equality follows from the fact that $\mathbf{u}_{||}$ is perpendicular to $\mathbf{h}$ and $\mathbf{f}$ is quadratic.

Finally, we neglect $\mathbf{f}_\perp(\mathbf{u}_{||})$ since it is quadratic in $|\boldsymbol{\Delta}_{||}|$. This is valid when $|\boldsymbol{\Delta}_\perp| \sim |\boldsymbol{\Delta}_{||}|$.

To interpret Eq. (2.20), we let $\{\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \cdots\}$ be the basis of wheel stresses of ker $\mathbf{R}^T$ described in Ref. [14]. In this basis,

$$\boldsymbol{\sigma}_n \cdot \mathbf{f}(\mathbf{h}) \approx \boldsymbol{\sigma}_n \cdot \boldsymbol{\Delta}, \tag{2.21}$$

where the left-hand side can be interpreted as the discrete Gaussian curvature at vertex n, or alternatively as a quadratic form, $\mathbf{h}^T \mathbf{Q}_n \mathbf{h}$ [14]. Finally,

$$\mathbf{h}^T \mathbf{Q}_n \mathbf{h} = \boldsymbol{\sigma}_n \cdot \boldsymbol{\Delta} \equiv K_n. \tag{2.22}$$

We note that, when $K_n = 0$, Eq. (2.22) reproduces the results of Chen *et al.* [14] for flat origami. Notice that the right-hand side of Eq. (2.22) involves only lengths of the bonds, encoded through $\boldsymbol{\Delta}$. This is then a discrete version of Gauss' *theorema egregium*, which relates the Ricci curvature on a surface – a completely intrinsic quantity – to the Gaussian curvature – an extrinsic quantity.

2.6.1 Relation to linear analysis

Rather than expanding the deformations around a nearly flat state. We could have solved Eq. (2.2) directly from a slightly deformed state. Here, we demonstrate that our approach yields the same results to linear order. Let $\mathbf{X}_n = \mathbf{X}_n^{(0)} + h_n \hat{\mathbf{z}}$, where $\mathbf{X}_n^{(0)}$ has no $\hat{\mathbf{z}}$ component. Similarly, write $\mathbf{u}_n = \mathbf{w}_n + h_n^{(1)} \hat{\mathbf{z}}$, where $\mathbf{w}_n$ has no $\hat{\mathbf{z}}$ component. Eq. (2.2) then reads

$$2\left(\mathbf{X}_n^{(0} - \mathbf{X}_m^{(0)}\right) \cdot (\mathbf{w}_n - \mathbf{w}_m) + 2(h_n^{(0)} - h_m^{(0)})(h_n^{(1)} - h_m^{(1)}) = 0. \tag{2.23}$$

Let σ_i^{nm} be a wheel stress around vertex i. Then we have

$$\sum_{nm} \sigma_i^{nm} 2(h_n^{(0)} - h_m^{(0)})(h_n^{(1)} - h_m^{(1)}) = 0, \tag{2.24}$$

where the sum is over all edges. Rewriting this in terms of the concatenated vectors $\mathbf{h}$, we obtain

$$\left(\mathbf{h}^{(0)}\right)^T \mathbf{Q}_i \mathbf{h}^{(1)} = 0. \tag{2.25}$$

Alternatively, if we expand Eq. (2.22) around $\mathbf{h}^{(0)}$ which satisfies $\mathbf{h}^{(0)T}\mathbf{Q}_i\mathbf{h}^{(0)} = K_i$, we also obtain Eq. (2.25).

2.7 Appendix B. Single vertices

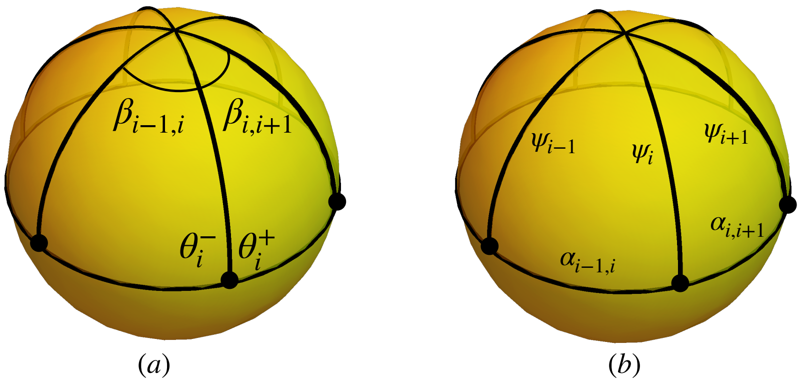


Figure 2.5: The intersection of a sphere with a vertex at its center is a spherical polygon, which we decompose into triangular slices as shown. (a) The dihedral angle of the i^{th} fold is $\theta_i^+ + \theta_i^-$. (b) The side lengths are the planar angles $\alpha_{i,i+1}$ and the angle the folds make with respect to the $xy-$plane, ψ_i.

We denote the central vertex with 0 and number the boundary vertices from $n = 1$ to N. Denote $\alpha_{n,n+1}$ as the angle between fold n and $n+1$, interpreted assuming $\alpha_{N,N+1} = \alpha_{N,1}$, and assume that $\alpha_{n,n+1}$ is always between 0 and π. Since we are interested in single vertices near the flat state, it is useful to change variables from the vertex heights to the angles made by the folds with respect to the $\hat{\mathbf{z}}$ axis, oriented with respect to the reference $z-$axis: $\psi_n = \pi/2 + (h_0 - h_n)/L_{n0}$ where L_{n0} is the length of fold n.

It is well known that a single vertex can be interpreted as a spherical polygon in which the side lengths are given by the planar angles $\alpha_{n,n+1}$ and the dihedral angles by the interior angles of the polygon [100] (Fig. 2.5); this connection has been used to explore the full configuration space of single origami vertices in general [52, 100]. Fig. 2.5 shows that such a polygon can be decomposed into triangular slices. Spherical trigonometry then allows one to write the dihedral angles entirely in terms of the ψ_n. For small deformations, these N angles $\psi_n = \pi/2 + \delta\psi_n$ where $\delta\psi_n = (h_n - h_0)/L_{n0}$. Finally, we define θ_n as the dihedral angle made by the n^{th} fold; the diagram in Fig. 2.2 shows that $\theta_n = \theta_n^+ + \theta_n^-$. Finally, we let $\theta_n = \pi - \delta\theta_n$ and assume $\delta\theta_n$ is small.

Expanding to quadratic order, we obtain the linear relationship

$$\delta\theta_n = \sum_m M_{nm}\delta\psi_m \tag{2.26}$$

where $M_{nm} = -\csc\alpha_{n,n+1}\delta_{n,m+1} - \csc\alpha_{n-1,n}\delta_{n,m-1} + (\cot\alpha_{n,n+1} + \cot\alpha_{n-1,n})\delta_{nm}$. Expanding the angles $\beta_{n,n+1}$ around $\alpha_{n,n+1}$ and using $\sum_n \beta_{n,n+1} = 2\pi$, we also find an expression for the Gaussian curvature of the vertex, $K = 2\pi - \sum_n \alpha_{n,n+1}$,

$$K = -\frac{1}{2}\sum_{nm} \delta\psi_n \delta\psi_m M_{nm}. \tag{2.27}$$

Comparing Eq. (2.27) to Eq. (2.4) provides a connection between the matrix $\mathbf{Q}$ governing the configuration space in terms of the vertex heights to the matrix $\mathbf{M}$, having components M_{nm}, governing the configuration space in terms of angles $\delta\psi_n$. In particular, while $\mathbf{Q}$ should have an additional zero eigenvalue from global translations of the vertex in the $\hat{\mathbf{z}}$ direction, it shares the same number of positive and negative eigenvalues as $\mathbf{M}$ [14].

Chapter 3

Mechanical signaling cascades

Abstract

Mechanical computing has seen resurgent interest recently owing to the potential to embed sensing and computation into new classes of programmable metamaterials. To realize this, however, one must push signals from one part of a device to another, and do so in a way that can be reset robustly. We investigate the propagation of signals in a bistable mechanical cascade uphill in energy. By identifying a penetration length for perturbations, we show that signals can propagate uphill for finite distances and map out parameters for this to occur. Experiments on soft elastomers corroborate our results.

3.1 Introduction

Mechanical devices that compute have apparently existed for thousands of years [13], yet have fallen out of favor since the advent of the modern electronic computer. However, the recent interest in mechanical metamaterials has led to a resurgence of interest in new forms of mechanical computing using modern – and predominantly soft – materials [8, 112], resulting in demonstrations of stable memory [17], boolean logic [45, 47, 74, 75, 99, 105, 76], and pattern recognition [30] among other computational and logical tasks. The dream is to create materials that change their behavior based on simple logic in response to external stimuli. To achieve that, in addition to logical elements, one requires convenient methods to transmit signals from one part of a material to another and a way to reset the state of a device or signal.

Recent work has demonstrated devices that can propagate mechanical signals by taking advantage of a series of interacting bistable elements of varying design [25, 111, 42, 44, 54, 55, 85, 84]. Previous work has shown that for asymmetric bistable units, transitioning from a higher to lower energy state allows for propagation over arbitrarily long distances [27, 26, 83, 88]. When utilizing symmetric bistable units, a decreasing grading of the interaction energy between bistable elements [43] or the energy barrier of the element itself [63] allows for stable propagation. Specifically designed bistable elements [63] or active components [9] can produce reversible signal propagation.

In this paper, we focus on a horizontal chain of bistable units (Fig. 3.1(b)) which act as relays that can each be in either a "left" (no signal) or "right" (signal) state [88, 85]. In order to perform multiple computations using chains with these bistable elements, we will need the ability to reset our system by sending a signal in both directions along the chain, indicating that it is important to also understand whether signals can propagate along uniform chains with no bias or a bias infavor of the "left" state.

To further explore this signaling cascade and gain insight into how many mechanical computations can really be performed using coupled, bistable elements, we consider this problem for finite chains of bistable elements as a function of both the difference in minimum energies, Δ, and barrier height E (Fig. 3.1(b)). Specifically, we look at the cases where the potential is symmetric and favors neither state, and cases where the potential favors the "left" state. Since we imagine working primarily with soft materials, we focus on the overdamped

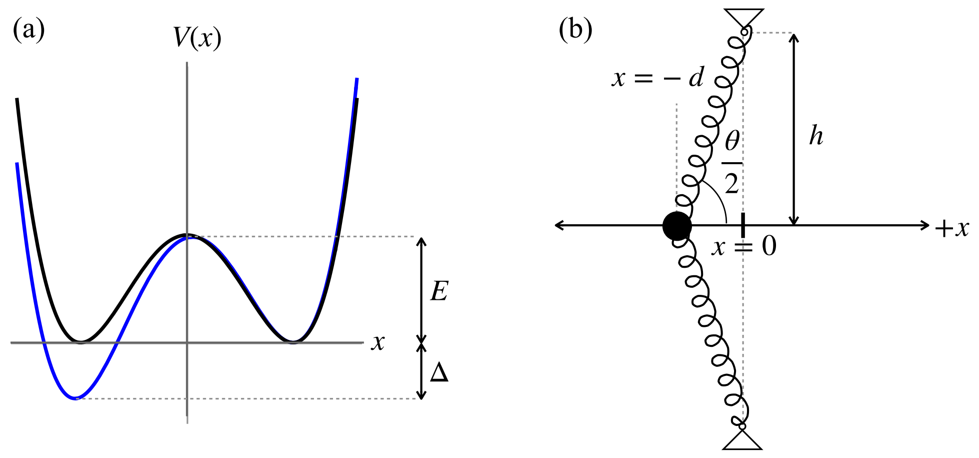


Figure 3.1: (a) The general shape of a bistable potential. The black line represents a symmetric potential with barrier height E. The blue line represents an asymmetric potential biased to the left where Δ is the potential difference between the two states. We measure the barrier height E with respect to the higher minimum. (b) An elastic bistable unit at equilibrium. The linear springs each have stiffness k. The torsional spring located at the point mass m (filled black circle) has torsional modulus s and equilibrium angle θ_0. The potential minima for the bistable unit are located at $x = \pm d$. The potential barrier is located at $x = 0$.

limit. This provides us with a set of simple scaling laws governing when signals propagate in terms of the number of elements and the energy of the bistable elements and provides limits on the number of computations that could be conceivably performed and reset.

3.2 Mechanical Signalling Cascades

3.2.1 Bistable elements

We consider a series of mechanical relays whose state is represented by a scalar variable x subject to a bistable potential, $V(x)$ (Fig. 3.1(a)). The potential is characterized by two minima having a difference in energy, Δ. The energy barrier between them is given by E and measures the barrier height with respect to the higher energy minimum. Though we will pursue a theoretical approach that is agnostic on the detailed form of the bistable elements, it is helpful to have a particular mechanical model in mind that can be implemented both experimentally and in simulations. We will model bistable elements as two linear springs (Fig. 3.1(b)) with ends attached to fixed points [85].

The potential energy of this two-spring bistable element is

$$V(x) = b\left(\sqrt{x^2+h^2} - \sqrt{h^2+d^2}\right)^2 \tag{3.1}$$

where x is the horizontal displacement of the point mass, $2h$ is the distance between fixed vertices, and $\sqrt{h^2+d^2}$ is the equilibrium length of the two springs. The two minima are at $x = \pm d$, and the energy barrier height is $E = b[\sqrt{h^2+d^2} - h]^2$. To bias the bistable element towards one of the two stable states, we can add a torsional spring at the mass. This adds an additional term of the form

$$V_{tor}(\theta) = \frac{1}{2}s\left(\theta - \theta_0\right)^2 \tag{3.2}$$

to the potential energy of the bistable element, where s is the torsional modulus, θ_0 is the equilibrium angle of that modulus, and $\tan(\theta/2) = h/x$.

Since we are interested primarily in signals driven up a potential barrier, we will choose the equilibrium angle to yield zero torsional energy when the bistable element is in the left-most state. This leads to an approximate barrier height,

$$E = b\left(h - \sqrt{h^2+d^2}\right)^2 + \frac{s}{2}\left(\pi - \theta_0\right)^2 + \mathcal{O}(s^2), \tag{3.3}$$

and asymmetry,

$$\Delta = \frac{s}{2}\left(2\pi - 4\tan^{-1}\frac{h}{d}\right)^2 + \mathcal{O}(s^2). \tag{3.4}$$

Thus, b predominantly controls the barrier height E while s controls the energy difference between minima, Δ (Fig. 3.1(a)).

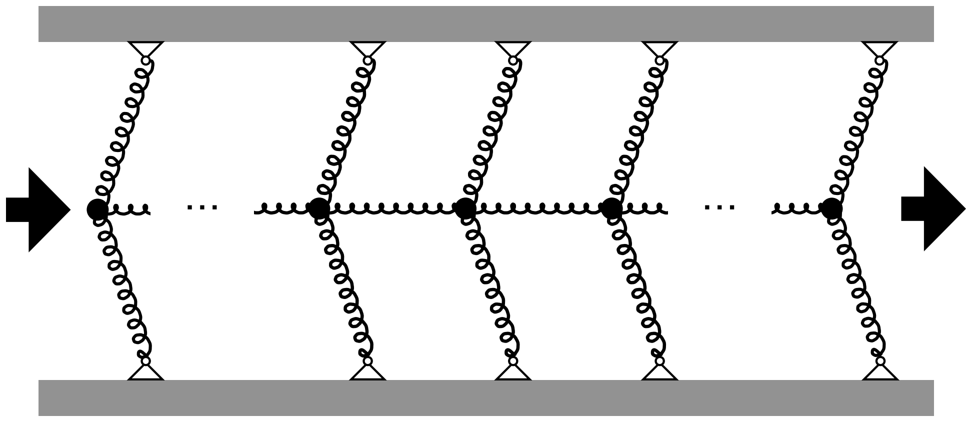


Figure 3.2: A finite length wire in the "left" (no signal) state. Each bistable element is at the $x = -d$ potential minimum. When the left-most point mass is pushed over the energy barrier and into the $x = +d$ energy minimum, the interaction spring connecting adjacent point masses allow that transition to propagate along the wire. If the right-most mass moves to the right, the entire wire is in the "right" (signal) state, and we say that the signal fully propagated along the wire. If only a portion of the bistable elements transition to the $x = +d$ energy minimum, we say that the signal only propagated a finite distance.

3.2.2 Equations of Motion

To form the one dimensional chain of bistable elements, we connect neighboring elements together at the point masses with linear springs of rest length a and spring constant k (Fig. 3.2) [85]. The position of the n-th point mass with respect to the fixed end points of the beam is given by x_n. Writing out Newton's second law for the ith mass gives

$$m\frac{d^2x_i}{dt^2} - k[x_{i+1} - 2x_i + x_{i-1}] + \gamma\frac{dx_i}{dt} + \frac{dV(x_i)}{dx_i} = 0 \tag{3.5}$$

where γ is a friction coefficient. To obtain the continuum limit, we make the variable change $x_{i\pm n} = u(y \pm na, t)$. The function $u(y,t)$ now represents the displacement of the beam at location y along the wire. Taking the limit where $a \to 0$, the equation of motion for the ith mass becomes

$$m\frac{\partial^2 u}{\partial t^2} - ka^2\frac{\partial^2 u}{\partial y^2} + \gamma\frac{\partial u}{\partial t} + \frac{dV}{du} = 0. \tag{3.6}$$

Before we analyze the equation of motion, we replace y, t, and u with dimensionless variables. First, we rescale $u \to a\tilde{u}$, $y \to a\tilde{y}$, and $t \to \tau\tilde{t}$ where a is the length of the interaction springs and τ is some characteristic time.

$$\frac{ma}{\tau^2}\frac{\partial^2\tilde{u}}{\partial\tilde{t}^2} - ka\frac{\partial^2\tilde{u}}{\partial\tilde{y}^2} + \frac{\gamma a}{\tau}\frac{\partial\tilde{u}}{\partial\tilde{t}} + \frac{1}{a}\frac{dV}{d\tilde{u}} = 0 \tag{3.7}$$

Rescaling $V \to EV'$ by the barrier height of the potential, E, as defined in Fig. (3.1), we obtain

$$\frac{ma}{\tau^2}\frac{\partial^2\tilde{u}}{\partial\tilde{t}^2} - ka\frac{\partial^2\tilde{u}}{\partial\tilde{y}^2} + \frac{\gamma a}{\tau}\frac{\partial\tilde{u}}{\partial\tilde{t}} + \frac{E}{a}\frac{dV'}{d\tilde{u}} = 0 \tag{3.8}$$

Finally, we multiply the entire equation by the ratio a/E.

$$\frac{ma^2}{E\tau^2}\frac{\partial^2\tilde{u}}{\partial\tilde{t}^2} - \frac{ka^2}{E}\frac{\partial^2\tilde{u}}{\partial\tilde{y}^2} + \frac{\gamma a^2}{E\tau}\frac{\partial\tilde{u}}{\partial\tilde{t}} + \frac{dV'}{d\tilde{u}} = 0 \tag{3.9}$$

All terms in the equation of motion for the nth mass are now dimensionless. When $ma^2/E\tau^2 \ll 1$, the system is overdamped and the first term of Eq. (3.9) can be neglected. To reduce the number of coefficients in Eq. (3.9), we rescale $\tilde{y}$ and $\tilde{t}$ again in the following way:

$$y' = \sqrt{\frac{E}{ka^2}}\tilde{y} \qquad t' = \frac{E\tau}{\gamma a^2}\tilde{t} \tag{3.10}$$

The equation of motion in the overdamped limit is now

$$-\frac{\partial^2\tilde{u}}{\partial y'^2} + \frac{\partial\tilde{u}}{\partial t'} + \frac{\partial V'}{\partial\tilde{u}} = 0 \tag{3.11}$$

with all dimensionless terms and variables. We have the following relationships between our physical variables and the new dimensionless variables:

$$\begin{aligned} y' &= \sqrt{\frac{E}{ka^2}}\tilde{y} = \sqrt{\frac{E}{ka^2}}\frac{1}{a}y = \frac{1}{a^2}\sqrt{\frac{E}{k}}y \\ t' &= \frac{E\tau}{\gamma a^2}\tilde{t} = \frac{E\tau}{\gamma a^2}\frac{1}{\tau}t = \frac{E}{\gamma a^2}t \end{aligned} \tag{3.12}$$

Thus, given a solution to Eq. (3.11), $\tilde{u}(y', t')$, we have

$$u(y,t) = a\tilde{u}\left(\sqrt{\frac{E}{k}}\frac{y}{a^2}, \frac{E}{\gamma a^2}t\right). \tag{3.13}$$

Finally, if the last element of the chain is free, this requires $\partial u/\partial y|_{y=L} = 0$ on the rightmost boundary at $y = L$. Therefore, $\partial\tilde{u}/\partial y' = 0$ when $y' = L' = \sqrt{E/k}L/a^2$. On the left-most boundary at $y = y' = 0$, we fix the initial displacement, $\tilde{u}(0)$.

3.2.3 Solutions

If we look for stationary solutions, we can find a first-integral for Eq. (3.11) resulting in

$$-\frac{1}{2}\left(\frac{\partial\tilde{u}}{\partial y'}\right)^2 + V'(\tilde{u}(y')) = C, \tag{3.14}$$

for some constant C. Applying the boundary condition on the right, we see that $C = V'(\tilde{u}(L'))$. Thus, we obtain a general solution

$$\frac{\partial\tilde{u}}{\partial y'} = \sqrt{2}\sqrt{V'[\tilde{u}(y')] - V'[\tilde{u}(L')]}, \tag{3.15}$$

where the choice of positive sign outside the square root is consistent with our boundary conditions.

When $\tilde{u}(0) = d/a$, one solution to Eq. (3.15) is $\tilde{u}(y') = d/a$: the signal will propagate completely from one side to the other. The difference in elastic energy from the ground state solution (with $\tilde{u}(y') = -d/a$) will scale with L and, thus, we expect it to be prohibitive for particularly long cascades. However, we also expect solutions with $\tilde{u}(0) = d/a$ but $\tilde{u}(L') < 0$, with a characteristic, dimensionless length η governing the penetration of a signal into the chain. Thus we expect solutions, $u(y)$, will transition over a length scale $\ell = \eta a^2\sqrt{k}/\sqrt{E}$. The elastic energy cost, in this case, will scale with ℓ rather than L. We thus expect two regimes of behavior: when $\ell < L$, an initial perturbation initiated on the "left" of the signaling cascade will penetrate only a finite distance ℓ; when $\ell > L$, we expect all elements to be on the "right" – the signal will propagate through the entire network. Therefore the equation

$$L = \eta a^2\sqrt{\frac{k}{E}} \tag{3.16}$$

represents the boundary between an initial perturbation propagating a finite distance and propagating through the entire network. This equation tells us the scaling relationship between various wire parameters and can be used to predict the general location and shape of the $\ell < L$ and $\ell > L$ regions for a given set of wire parameters.

It is instructive to consider a specific example, for which $V'(\tilde{u}) = (\tilde{u}^2 - d^2/a^2)^2/2$, and Eq. (3.11) takes the form of the generalized Fisher equation [32, 49]

$$\frac{\partial u'(y',t)}{\partial t} - D\frac{\partial^2 u'(y',t)}{\partial y'^2} + u'\left(u'^n - 1\right) = 0 \tag{3.17}$$

with $n = 2$. Based on solutions presented in [70], we construct the stationary solution

$$u'(y') = -1 + 2\frac{1}{1 + ae^{\sqrt{2}(y')}} \tag{3.18}$$

valid for a half infinite line starting at $y' = 0$, where $a = (1-u_0)/(1+u_0)$, $\tilde{u}(0) = u_0$. Putting this back into the elastic energy yields

$$\mathcal{E} = \frac{(u_0 - 2)(1 + u_0)^3}{3\sqrt{2}}, \tag{3.19}$$

for $-1 < u_0 \leq 1$. This expression is minimized when $u_0 = -1$, implying that a signal never propagates in a symmetric, infinitely long chain. Despite this, finite sized chains are likely to behave differently. Indeed, these results hint that chains with length $L < a^2\sqrt{k/E}$ – shorter than the intrinsic scale of Eq. (3.18) – may still propagate signals.

3.3 Results

3.3.1 Simulations

To corroborate our scaling analysis, we performed simulations of the dynamical system in Fig. 3.2 directly by integrating Eq. (3.5). To obtain the overdamped limit, we set $m = 0$. Consider a wire of length L consisting of bistable elements with equilibrium positions at $x + i = \pm d$. The wire is initially in the ground state with all beams in the $x_i = -d$ "left" position. To initiate a signaling cascade, we applied a displacement of $2d$ to the first beam to move it into the $x_1 = +d$ "right" position. After a sufficient length of time that the motion has stopped, we recorded the position x_L of the last beam in the wire. If $x_L < 0$, the signal only propagated a finite distance. If $x_L > 0$, the signal propagated through the entire network.

We show the relationship between the barrier height E and the length of the wire L in Fig. 3.3(a). To verify the scaling relationship between L and E in Eq.(3.16), we fit the boundary between the finite and full propagation regions to an expression of the form

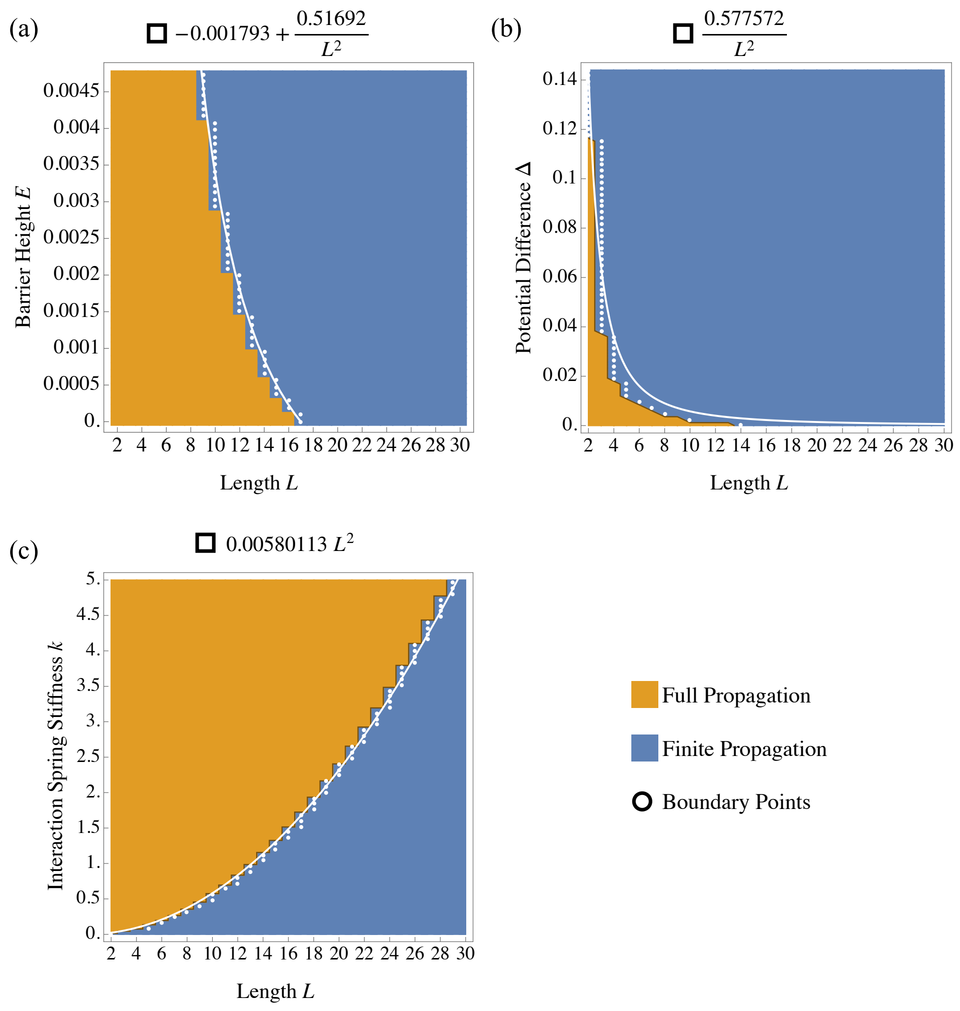


Figure 3.3: For all three plots, the wire length parameters are $h = 1$, $a = 1$, and $d = 1/4$.(a) Scaling relationship between the potential barrier height E and the length of the wire when varying the beam stiffness. The beam stiffness b ranges from 0 to 5. The interaction spring stiffness k is set to 1, and the torsional modulus s is set to 0. Each boundary point corresponds to a simulated wire with a specific choice of length L and characteristic energy E. (b) Scaling relationship between the potential difference between minima Δ and the length of the wire when varying the torsional modulus. The torsional spring stiffness s ranges from 0 to 0.3. The interaction spring stiffness k and beam stiffness b are both set to 1. Each boundary point corresponds to a simulated wire with a specific choice of length L and torsional modulus s. (c) Scaling relationship between the interaction spring stiffness and the length of the wire. The interaction spring stiffness k ranges from 0 to 5. The beam stiffness b is set to 1, and the torsional modulus $s$is set to 0. Each boundary point corresponds to a simulated wire with a specific choice of length L and interaction spring stiffness k.

$E = C_1 L^{-2} + C_2$, with $C_2 \approx -2 \times 10^{-3}$ and $C_1 \approx 0.51$. This suggests a scaling relationship

$$E - E_0 = \frac{\eta^2 a^4}{L^2} k \tag{3.20}$$

with $E_0 \approx -2 \times 10^{-3}$. Turning this around,

$$k = \frac{E - E_0}{\eta^2 a^4} L^2. \tag{3.21}$$

We also show the relationship between the potential difference between minima Δ and the length of the wire in Fig. 3.3(b). We find $\Delta \approx C_1 L^{-2} + C_2$, with $C_2 = 0$ and $C_1 \approx 0.58$.

The relationship between k and wire length is depicted in Fig. 3.3(c) for $E \approx 10^{-3}$. An expression of the form $k \approx C_1 L^2$ with $C_1 \approx 6 \times 10^{-3}$ fits the boundary between the finite and full propagation regions well. Using the previous results for $\eta^2 a^4$, E and E_0, we see that C_1 is consistent with $(E - E_0)/(\eta^2 a^4) \approx 6 \times 10^{-3}$.

3.3.2 Experiments

To validate our results experimentally, bistable elements were designed using Fusion 360 CAD software and fabricated using a Formlabs Form2 SLA 3D printer with the Formlabs Elastic 50A resin, which has flexible and stretchable properties after curing. The design for the wires is shown in Fig. 3.4. The bistable elements were elastic beams that buckle under compression, and the nearest neighbor interactions were facilitated by linear springs. The wire was printed in an unstressed state, and compressed using a rigid frame, printed using Formlabs Grey resin. (Fig. 3.4).

To bias the beams to buckle either to the left or right, pre-curvature was added to the elastic beams by changing the angle of the beam at the fixed end points and midpoint as shown in Fig. 3.5(e). For wires with biased beams, there was no need to compress the wire

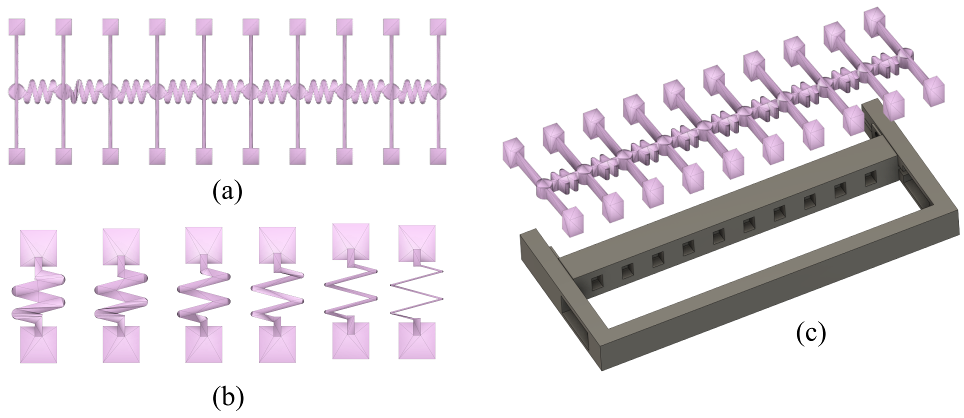


Figure 3.4: 3D models of wire components. (a) The initial state of a wire with symmetric bistable beam elements. (b) Wire slotted into a holding device (frame) that compresses all bistable beam elements to buckle them uniformly. (c) Interaction springs of varying thickness. Different spring stiffnesses are achieved through changing the thickness of the spring. Individual interaction springs are printed specifically for measuring their stiffness.

before sending a signal.

The wire model used for simulations had seven parameters, L, h, d, a, k, b, and s where values were selected for each. When verifying the scaling relationship found with Eq. (3.16) with simulations, we picked default values for each parameter that simplified our analysis as much as possible. We measured each of these parameters directly from the printed beams in order to run simulations that approximated the behavior of the printed wires. The height of the beam h and the distance between beams a were found through direct meaurement of the wire. We determined values for the location of the two potential minima d, the beam stiffness b, the interaction spring stiffness k, and the torsional modulus s using force-displacement measurements of the bistable beams and interaction springs.

Force-displacement measurements were performed using a custom setup composed of a linear displacement stage (Zaber Technologies Inc., T-LSM 100) and a load cell (Loadstar Sensors Inc., RPG-10). The wire was placed in the rigid frame and compressed to buckle the beams and set the wire into one of two stable states. The frame was mounted vertically and a signal passed from top to bottom during the test with a mechanical push. Using the linear displacement stage, we slowly pushed on the bistable wire with the load cell and recorded the force exerted on the wire as a function of displacement as the beams transitioned between stable configurations (Fig. 3.5(f-h)). To apply the force to the beam, we used a rigid component printed out of the same material as the frame and attached it directly to the midpoint of the beam. The rigid component was fixed to the wire so that the snap-through transition did not cause the device to lose contact with the load cell.

We conducted force-displacement measurements for the linear interaction springs using a TA.XTplus texture analyzer (Stable Micro Systems). Each end of the spring was mounted directly in the texture analyzer, as shown in Fig. 3.5(i), and the spring was slowly compressed and stretched.

To determine the stiffness k of the linear interaction springs, we fit a straight line to the force-displacement data and recorded the slope. To determine an approximate value of b for symmetric beams, we fit the derivative of Eq. 3.1 to the force-displacement data. For pre-curved beams, we added the torsional spring term Eq. 3.2 to Eq. 3.1 before taking the derivative and fitting it to the force-displacement data. Details of the methods used to determine b and s for both symmetric and biased beams are discussed in Appendix A (Sec. 3.6).

To gather data on what wire parameter combinations allow for finite or full signal propagation, we sent a signal down the wire by hand by pushing on the first beam until it reached the right-buckled position. The beam was held in that position using tweezers for a few seconds to allow the wire to settle into a final position. If the initial displacement caused all subsequent beams to snap through to the right-buckled position, then the signal was considered to have fully propagated along the wire. If the initial displacement moved some of the beams but not all of them, then the signal only propagated a finite distance. By changing the length of the wire and the stiffness of the beams and springs, we then plotted our experimental data the same way we plotted the simulation data in Fig. 3.3 and compared

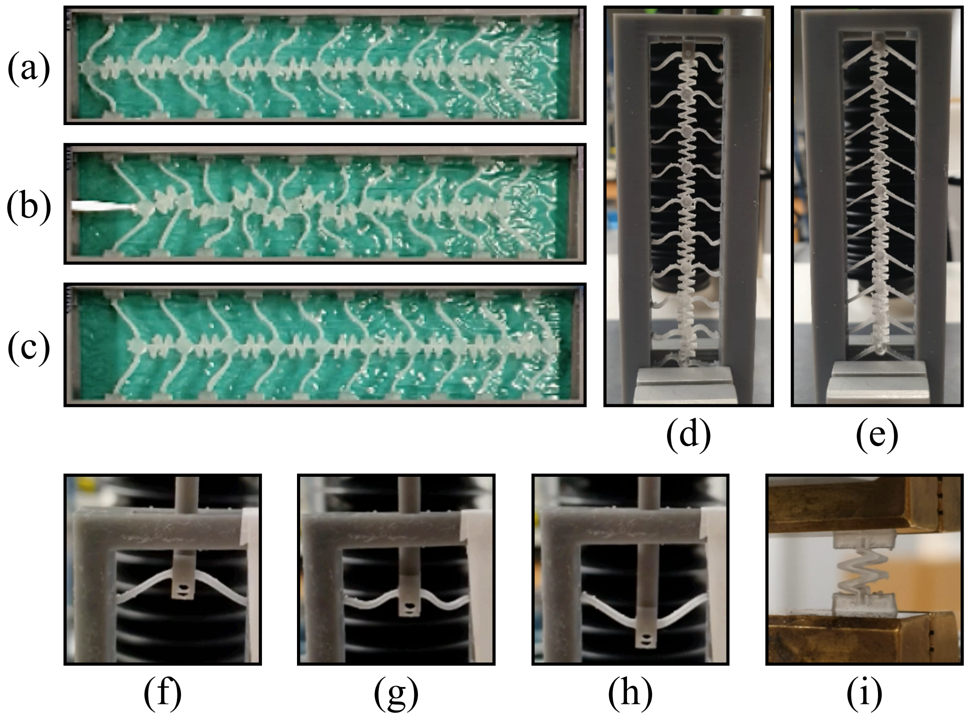


Figure 3.5: A wire before (a), during (b), and after (c) a signal is fully propagated through manual displacement of the first bistable beam in the wire. Comparison between a wire with symmetric (d) and asymmetric (e) bistable beam elements when mounted in the frame. (f)-(h) Recording the force-displacement data for a single bistable beam. (i) The interaction spring mounted and ready for force-displacement measurements. All beams shown have an end-to-end distance of 24mm. All interaction springs have a rest length of 11.2mm. The length of the full wires shown in (a-e) is approximately 10cm.

the results.

Fig. 3.6 compares the finite and full propagation regions from experiments and simulations for different values of the interaction spring stiffness and torsional modulus. Comparing the regions produced by varying the interaction spring stiffness shows only a qualitative match between simulations and experiments. When measuring the stiffness of the printed springs, they were only compressed a small amount and were not allowed to buckle side to side. When sending a signal along the printed wires, however, there was substantial buckling of the interaction springs, as shown in Fig. 3.5(b). Thus, the additional buckling lowered the effective barrier height from what was measured. Indeed, we found quantitative agreement

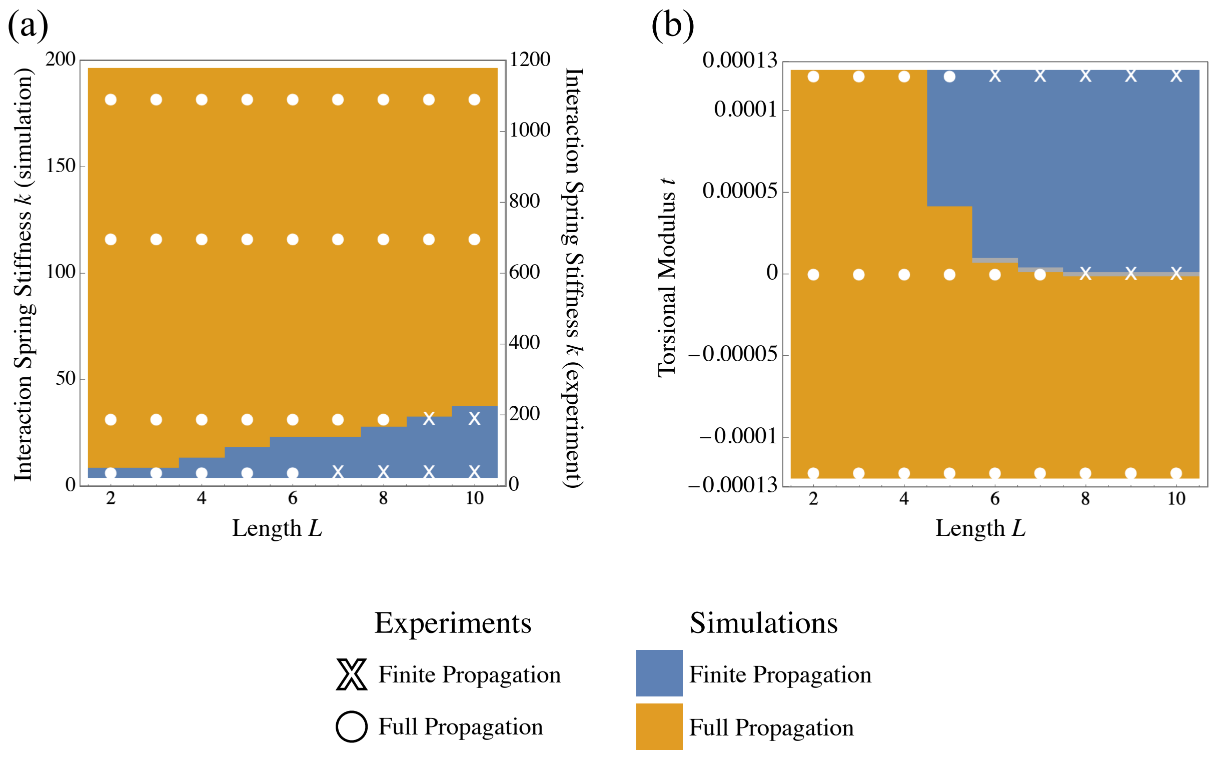


Figure 3.6: Comparison between signal propagation experiments and numerical simulations. (a) Qualitative comparison between simulations and experiments for varying interaction spring stiffness. The wire parameter values used for simulations are $h = 0.012m$, $d = 0.0055m$, $a = 0.0112m$, $b = 120N/m$, and $s = 0N/m$. (b) Direct comparison between simulations and experiments for varying minima difference Δ of the bistable beam potential. In simulations, this was done by varying the torsional stiffness on each bistable element. In experiments, this was done by changing the angle of the beam at its fixed endpoints (see Fig. 3.5(e) for example). The wire parameter values used for simulations are $h = 0.012m$, $d = 0.0043m$, $a = 0.0112m$, $b = 230N/m$, and $k = 30N/m$.

between simulation and experimental results if we reduced the simulation interaction spring stiffness by a factor of 6 from what was initially estimated from deformations that do not buckle.

Using this result, we used a rescaled value for the interaction spring stiffness for the simulations in Fig. 3.6(b). When the torsional modulus was negative, so that propagating a signal causes the beams to transition from a higher energy state to a lower energy state, we confirmed that signals always propagate fully for any number of elements. For positive values of the torional modulus, however, signal propagation depended on the number of elements and was consistent with predictions from simulations.

3.4 Conclusion/Discussion

In this paper we analyze signaling cascades of bi-stable mechanical elements coupled by springs triggered by the imposition of a fixed displacement on one of the elements. While signals can propagate at any distance when the bistable elements switch from a higher energy to lower energy state, here we find that signals can only propagate a finite distance when the elements transition to either a higher energy state or a state with the same energy. Numerical analysis of simulation data in the overdamped limit shows that signals can propagate when

$$L < \eta a^2 \sqrt{\frac{k}{E - E_0}}, \tag{3.22}$$

even for a completely symmetric signalling cascade.

Though our analysis was for a simple signalling cascade, one expects even more complex digital mechanical logic elements to also present energy barriers that must be overcome by a propagating signal. For true reversible logic, this finite propagation length might present a true limitation to the size and complexity of a device when realizing repeatable, complex logic in a soft mechanical system.

3.5 Acknowledgements

The authors gratefully acknowledge support for this work provided by the US Army Research Office through grant W911NF-21-1-0068.

3.6 Appendix A: Fit for Simulation Parameters

3.6.1 Interaction Springs

We fit a straight line that crosses the origin to the force-displacement data for springs of varying thickness. The slope of the line is the linear spring stiffness of the printed springs.

Thickness (mm)	Stiffness (N/m)
0.6	30.77
0.8	187.12
1.0	715.47
1.2	1123.95

Table 3.1: Interaction spring stiffness for various beam thicknesses.

3.6.2 Symmetric Bistable Beams

We recorded force-displacement data for a single bistable beam and integrated it numerically using a custom MATLAB program to get the potential-displacement data for our printed beams. We measured h directly from the wire. We calculated d by taking half the distance between the two minima in the potential-displacement data.

We then took the beam potential $V(x)$ from Eq. (3.1) and fit it to our force- and potential-displacement data in one of five ways: fit $-dV/dx$ to the maximum of the force data, fit $-dV/dx$ to the minimum of the force data, fit $V(x)$ to the energy barrier of the potential data, or by using the built-in Mathematica function FindFit with both the force and potential data. The first three values of b are averaged together to give an approximate beam stiffness, and the last two are left as-is.

End-to-End Distance (mm)	h (m)	d (m)
22	0.011	0.0066
24	0.012	0.0055
26	0.013	0.0043

Table 3.2: Dimensions for symmetric bistable beams

		FindFit Stiffness (N/m)	
End-to-End Distance (mm)	Average Stiffness (N/m)	Force Data	Potential Data
22	80	83	83
24	119	135	123
26	187	211	178

Table 3.3: Beam stiffness for symmetric bistable beams.

3.6.3 Asymmetric Bistable Beams

For the bistable beams printed with pre-curvature, we need to determine the torsional modulus s and the beam stiffness b. Just as with the symmetric beams, we measured h directly from the wire and calculated d by taking half the distance between the two minima in the potential-displacement data. We took the beam potential $V(x)$ that included the terms in both Eq. (3.1) and (3.2) and used the built-in Mathematica function FindFit to solve for both b and s at the same time.

	h (m)	d (m)	b (N/m)	s (J)
Uphill Push	0.012	0.0043	233	0.00013
Downhill Push	0.012	0.0047	162	0.00013

Table 3.4: Beam stiffness and torsional modulus for asymmetric bistable beams. First row: the beam transitions from a lower minimum to a higher minimum (uphill push). Second row: the beam transitions from a higher minimum to a lower minimum (downhill push).

Chapter 4

The Flip-Flop Linkage

In this chapter, we discuss the flip-flop, the first candidate for a mechanical transistor that we investigated, and outline the theoretical and experimental work done trying to integrate the flip-flop into the mechanical wires from Chapter 3. The devices discussed in this chapter didn't end up working as intended, but the time spent investigating the flip-flop's behavior provided useful information that guided future work.

4.1 Description

The flip-flop is a linkage designed specifically to be used as a mechanical bit that can be in a 0 or 1 state [76]. There are two distinct motions of the flip-flop: one where the top half (vertices 6, 7, and 8) moves side to side, and one where the bottom half (vertices 1, 2, and 3) moves side to side (Fig. 4.1). Once either the top or bottom half shifts to one side, the linkage is 'locked' into that motion and only that half is allowed to move.

The flip-flop can be thought of as a modified version of the four-bar linkage presented in Chapter 1. In this chapter, we will use a version of the four-bar linkage where the bar connecting vertices 1 and 4 in Fig. 1.2 is removed and two pinned vertices are added. The

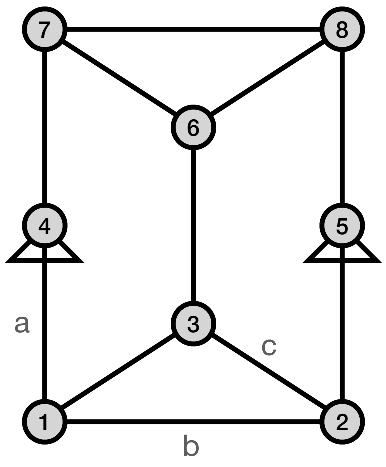


Figure 4.1: The flip-flop linkage. The triangle on vertices 4 and 5 represents a pinned joint. Vertex coordinates are as follows: $\mathbf{p}_1 = (0,0), \mathbf{p}_2 = (b,0), \mathbf{p}_3 = (\frac{b}{2}, \frac{a}{2}), \mathbf{p}_4 = (0,a), \mathbf{p}_5 = (b,a), \mathbf{p}_6 = (\frac{b}{2}, \frac{3a}{2}), \mathbf{p}_7 = (0,2a), \mathbf{p}_8 = (b,2a)$

bars connected at one end to the pinned vertices will be called rotors because they can rotate around that fixed point. To fully explain and develop some intuition for why the flip-flop has this locking behavior, we will slowly modify the four-bar linkage until it becomes the flip-flop linkage.

To start off, we assign two positions of the four-bar linkage to represent the 0 and 1 states (Fig. 4.2). In the 0 state, both rotors will be vertical. Any shift to either the left or right will be considered the 1 state. The four-bar linkage is allowed to transition between all three configurations shown in Fig. 4.2 in a smooth motion. All bars in Fig. 4.2 are shown as having the same length, but the linkage will behave in the same way as long as the two rotors are the same length and the middle beam and pinned vertices are the same length/distance apart.

If we add an extra rotor to the four bar linkage as shown in Fig. 4.3, we can create both rigid and flexible linkages. When the additional rotor is the same length as the other two and is vertical at the 0 state, then the linkage is still able to shift side to side. If one of those conditions is not met, then the linkage is rigid and locked in place.

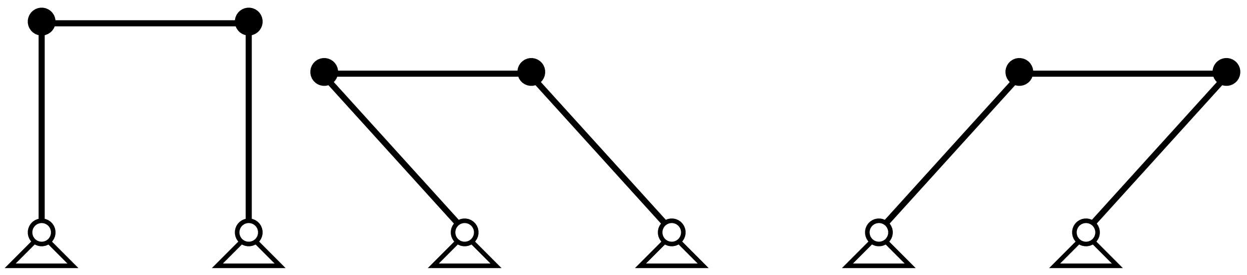


Figure 4.2: The 0 state of the four bar linkage will be the configuration where both rotors are vertical (left-most diagram). The 1 state of the rotor will be the configuration where the whole linkage is shifted to one side or the other (center and right-most diagram). The direction (left or right) that the linkage shifts is not important at this point, just that it has shifted away from the 0 state.

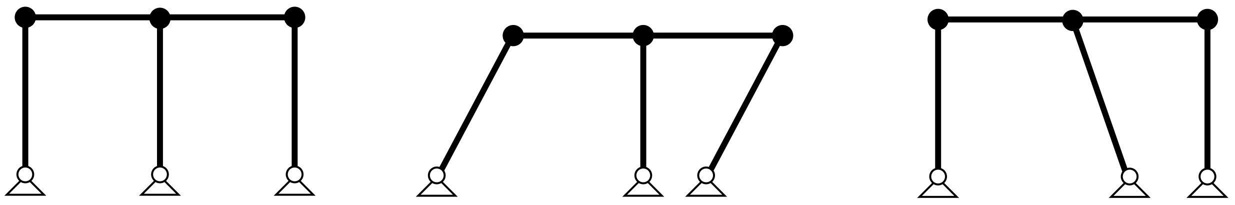


Figure 4.3: The linkage on the left is allowed to shift from side to side because each rotor is the same length and all bars are vertical in the 0 state. The linkages in the center and on the right are rigid because the middle rotor is not the same length as the outer two rotors. These are only two of the ways to create a linkage with this arrangement of bars and fixed vertices that is rigid.

The next modification to the linkage in Fig. 4.3 that we need to make is to replace the bar connecting the three rotors with a rigid triangle (Fig. 4.4). This linkage may *look* different from the one shown in Fig. 4.3, but when writing out the length constraint equations and calculating the allowed motions of the linkage, the two linkages move in the same way near the 0 state.

The flip-flop can be thought of as two overlapping copies of the linkage in Fig. 4.4. When both the top and bottom half are both in the 0 state, we can shift one half to the side without a problem. Each half can move just like the linkage in Fig. 4.4. Once one half is shifted, the other is locked because the three rotors for that half are not all vertical.

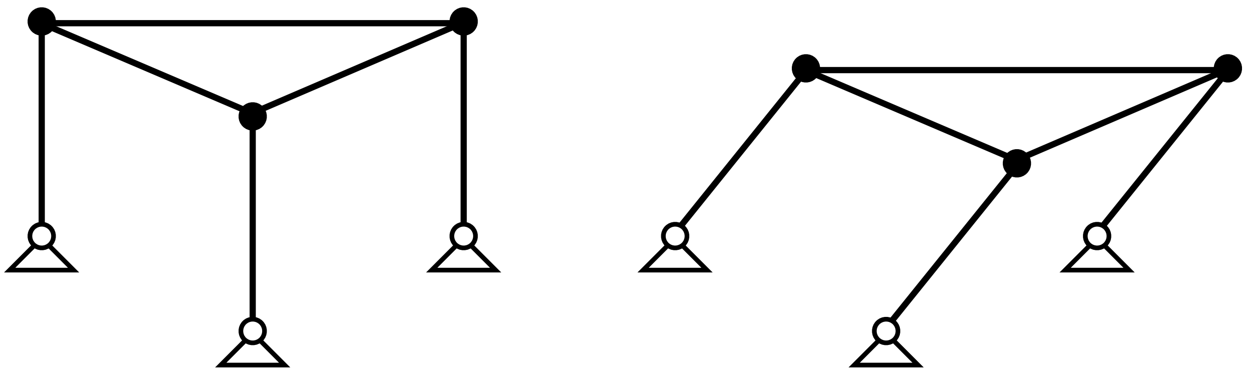


Figure 4.4: This linkage is still allowed to shift side to side.

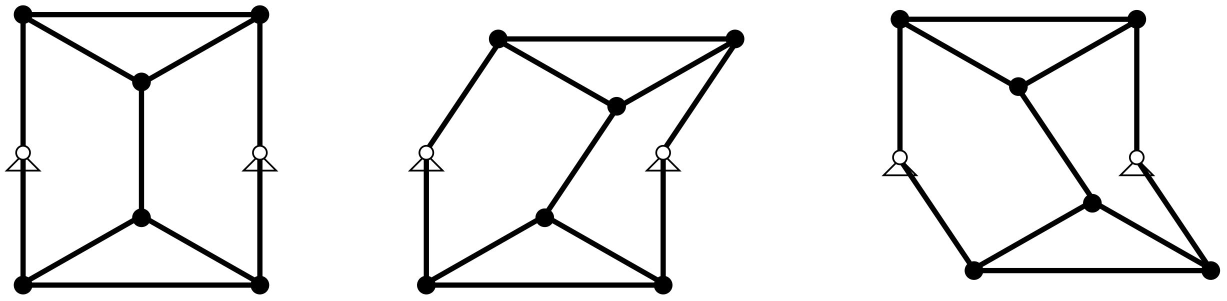


Figure 4.5: The top and bottom half of the flip flop can shift side to side. On the far left, both the top and bottom half are in the 0 state. Once one half is shifted to the 1 state, the remaining half is locked in the 0 state because the three vertical bars are not all at the same angle.

It turns out that multiple flip-flops can be combined with other base components to create multiple logic gates, including a NAND gate [76]. Therefore, the flip-flop can be used to create a functionally complete system, which is one of the many conditions for saying a system can compute. From here, we consider two questions:

1. What property of the flip-flop linkage makes it behave like a logical bit? If we can identify that property, can we design other linkages that are also have bit-like behavior?
2. Does the locking behavior of the flip-flop allow us to use it directly as a transistor? How could it be integrated into a mechanical wire to gate a signal?

4.2 Rigidity Theory Analysis

In this section, we will use the linkage analysis techniques from Chapter 1 to investigate properties of the flip-flop and search for information related to its locking behavior. The exact version of the flip-flop that we will look at is shown in Fig. 4.1.

4.2.1 Self Stresses

The first thing we will do is solve for any self stresses of the flip flop. By substituting our vertex positions for the 0 state (shown in Fig. 4.1) into Eq. 1.11, we can write out the system of equations needed to calculate the self stresses for the flip-flop. If we label bars (1,2), (1,4), (2,5), (4,7), (5,8), and (7,8) as the "outer" bars and (1,3), (2,3), (3,6), (6,7), and (6,8) as the "inner" bars, this block of equations simplifies down to the following self stress:

$$\sigma_{inner} = -2\sigma_{outer} \tag{4.1}$$

For the self stress we found for the flip-flop at the 0 state, we can have one of two situations:

- All of the outer bars have a positive stress and are under tension, and all of the inner bars have a negative stress and are under compression.
- All of the outer bars have a negative stress and are under compression, and all of the inner bars have a positive stress and are under tension.

4.2.2 Length Constraint Perturbations

Next, we will perform the length constraint perturbation analysis introduced in Chapter 1 and used in Chapter 2 on the flip-flop.

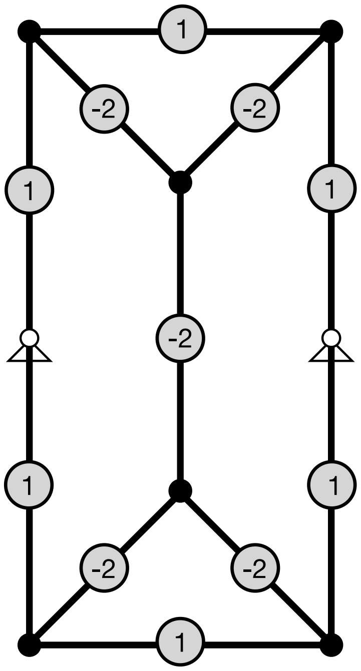


Figure 4.6: Self Stresses for the flip flop in the 0 state. Each value on the bars represents the magnitude of the self stress. The values shown represent one of the two possible self stresses.

4.2.2.1 First Order Equation

We start off by looking at the first order length perturbations from Eq. 1.24. Substituting in the self stress found in the previous section and rearranging the equation gives us

$$\begin{aligned}
&2\left(\delta\ell^{(1)}_{(1,3)} + \delta\ell^{(1)}_{(2,3)} + \delta\ell^{(1)}_{(3,6)} + \delta\ell^{(1)}_{(6,7)} + \delta\ell^{(1)}_{(6,8)}\right) \\
&\quad = \delta\ell^{(1)}_{(1,2)} + \delta\ell^{(1)}_{(1,4)} + \delta\ell^{(1)}_{(2,5)} + \delta\ell^{(1)}_{(4,7)} + \delta\ell^{(1)}_{(5,8)} + \delta\ell^{(1)}_{(7,8)}.
\end{aligned} \tag{4.2}$$

Now we look directly at the length of each bar. Including the first order length changes, the length squared of each bar is written as

$$\begin{aligned}
&L^2_{(1,2)} = b^2 + \delta\ell^{(1)}_{(1,2)}, \ \ L^2_{(1,3)} = c^2 + \delta\ell^{(1)}_{(1,3)}, \ \ L^2_{(1,4)} = a^2 + \delta\ell^{(1)}_{(1,4)} \\
&L^2_{(2,3)} = c^2 + \delta\ell^{(1)}_{(2,3)}, \ \ L^2_{(2,5)} = a^2 + \delta\ell^{(1)}_{(2,5)}, \ \ L^2_{(3,6)} = a^2 + \delta\ell^{(1)}_{(3,6)} \\
&L^2_{(4,5)} = b^2 + \delta\ell^{(1)}_{(4,5)}, \ \ L^2_{(4,7)} = a^2 + \delta\ell^{(1)}_{(4,7)}, \ \ L^2_{(5,8)} = a^2 + \delta\ell^{(1)}_{(5,8)} \\
&L^2_{(6,7)} = c^2 + \delta\ell^{(1)}_{(6,7)}, \ \ L^2_{(6,8)} = c^2 + \delta\ell^{(1)}_{(6,8)}, \ \ L^2_{(7,8)} = b^2 + \delta\ell^{(1)}_{(7,8)}.
\end{aligned} \tag{4.3}$$

Taking the square root of all the lengths and expanding up to first order gives

$$\begin{aligned}
&L_{(1,2)} \approx b + \frac{\delta\ell^{(1)}_{(1,2)}}{2b}, \ \ L_{(1,3)} \approx c + \frac{\delta\ell^{(1)}_{(1,3)}}{2c}, \ \ L_{(1,4)} \approx a + \frac{\delta\ell^{(1)}_{(1,4)}}{2a} \\
&L_{(2,3)} \approx c + \frac{\delta\ell^{(1)}_{(2,3)}}{2c}, \ \ L_{(2,5)} \approx a + \frac{\delta\ell^{(1)}_{(2,5)}}{2a}, \ \ L_{(3,6)} \approx a + \frac{\delta\ell^{(1)}_{(3,6)}}{2a} \\
&L_{(4,5)} \approx b + \frac{\delta\ell^{(1)}_{(4,5)}}{2b}, \ \ L_{(4,7)} \approx a + \frac{\delta\ell^{(1)}_{(4,7)}}{2a}, \ \ L_{(5,8)} \approx a + \frac{\delta\ell^{(1)}_{(5,8)}}{2a} \\
&L_{(6,7)} \approx c + \frac{\delta\ell^{(1)}_{(6,7)}}{2c}, \ \ L_{(6,8)} \approx c + \frac{\delta\ell^{(1)}_{(6,8)}}{2c}, \ \ L_{(7,8)} \approx b + \frac{\delta\ell^{(1)}_{(7,8)}}{2b}.
\end{aligned} \tag{4.4}$$

Because there are so many ways to deform the original shape, the same kind of result that was found for the example linkage in Fig. 1.7 doesn't show up here.

4.2.2.2 Second Order Equation

Next, we will go up to second order and expand and rearrange Eq. 1.25. If we substitute in the flex that moves the top half of the flip-flop (vertices 6, 7, and 8), the only terms on the left hand side of Eq. 1.25 that are nonzero correspond to bars $(4,7)$, $(3,6)$, and $(5,8)$.

$$\begin{aligned}
|\mathbf{p}_3^{(1)} - \mathbf{p}_6^{(1)}|^2 &= |0 - x\hat{\mathbf{p}}|^2 = |x\hat{\mathbf{p}}|^2 = x^2 \\
|\mathbf{p}_4^{(1)} - \mathbf{p}_7^{(1)}|^2 &= |0 - x\hat{\mathbf{p}}|^2 = |x\hat{\mathbf{p}}|^2 = x^2 \\
|\mathbf{p}_5^{(1)} - \mathbf{p}_8^{(1)}|^2 &= |0 - x\hat{\mathbf{p}}|^2 = |x\hat{\mathbf{p}}|^2 = x^2
\end{aligned} \tag{4.5}$$

Eq. 1.25 then becomes

$$\begin{aligned}
\sum_{(i,j)} \sigma_{(i,j)} \delta\ell_{(i,j)}^{(2)} &= \sigma_{(4,7)}|\mathbf{p}_4^{(1)} - \mathbf{p}_7^{(1)}|^2 + \sigma_{(3,6)}|\mathbf{p}_3^{(1)} - \mathbf{p}_6^{(1)}|^2 + \sigma_{(5,8)}|\mathbf{p}_5^{(1)} - \mathbf{p}_8^{(1)}|^2 \\
&= (1)x^2 + (-2)x^2 + (1)x^2 \\
&= 0
\end{aligned} \tag{4.6}$$

This tells us that for the motion of the flip-flop that we are interested in, any change to that motion caused by small length changes to the bars is either all at first order or at some higher order. Because translation in the x or y direction would also make the left hand sides simplify to zero, the only possible motion that could have second order effects would be rotation. However, because we pinned two of the vertices in the linkage, we have already removed any rotational motions. Therefore, any useful information is either in the first order equation, or happens at higher orders.

4.3 Flip-Flop as a Transistor

In this section, we explore ways to integrate the flip-flop into the mechanical wire studied in Chapter 3 so that it can act as a transistor. The side-to-side motion of the flip flop combined with its locking behavior make it a good candidate to use with our mechanical wires – the side-to-side motion is similar to the horizontal motion of the bistable beams when they transition from one minima to the other.

When a signal is sent down the wire, each beam shifts horizontally. If we connected a beam to each side of the top half of a flip-flop, it would also shift horizontally when the signal travels across it. By controlling the position of the bottom half of the flip-flop, we could control if a signal fully propagated along the wire.

The flip-flop would start off in a fully neutral position (both the top and bottom halves at state 0) with the beams in the wire all buckled to the left. If we sent a signal along the wire, the top half of the flip-flop would shift to the right, and the signal would propagate across the flip-flop. If we instead shifted the bottom half of the flip-flop to the side before sending the signal down the wire, the top half of the flip-flop won't be able to move, and the signal should be blocked.

First, we explore the behavior of the wire and flip-flop system through simulations in Mathematica. Using the *mechanisms* package, we construct the system out of fixed vertices, linear springs, and rigid bars. Then, just as we did in Chapter 3, we initiate a signal by first displacing a beam somewhere in the system. With this initial condition, we then let Mathematica numerically solve the equations of motion for the system for a time period long enough that the system has stopped moving. Because we consider a single example at one time, the length of that time period is manually adjusted for each new example.

4.3.1 Simulation Design and Testing

To connect the flip-flop to the wire, we first added a horizontal linear spring to vertices 7 and 8 the flip flop. These extra springs were then connected directly to the midpoint of the bistable beams in a wire. Fig. 4.7 shows a schematic of what that type of connection looks like. Because we need to avoid overlapping bars due to future experimental constraints, there is a limit to how short those connecting beams can be. However, there isn't a similar limit to how long we make them. To introduce some consistency in our design that makes combining multiple wires and flip-flops simpler, we chose to "remove" a single bistable beam and “replace” it with the flip-flop. The location we removed and replaced a bistable beam is outlined by a blue box in Fig. 4.7.

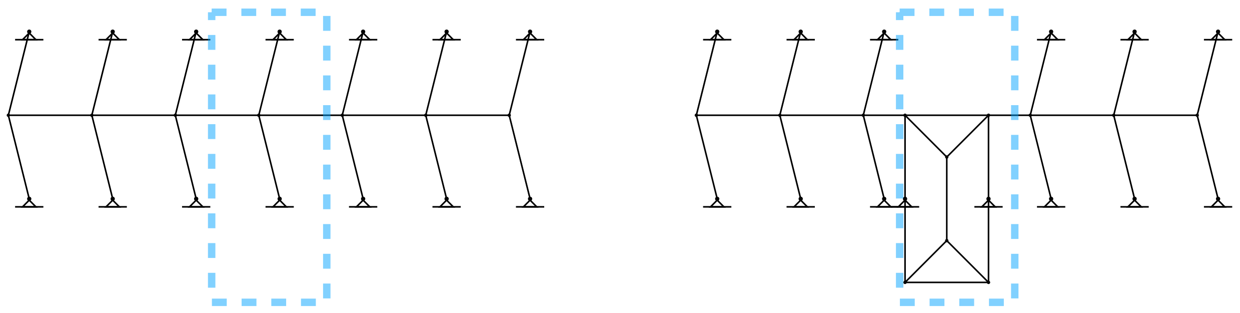


Figure 4.7: The wire consists of bistable beams with dimensions $h = 1$ and $d = 1/4$ (see Chapter 3 for details on the design of the wire). The flip-flop has bar lengths $a = 1$, $b = 1$, and $c = \frac{1}{\sqrt{2}}$. The two bars connecting the flip-flop to the wires both have length $\frac{1}{2}$.

For this system containing a wire and transistor, the transistor can be in two states. In the un-gated state where the bottom half of the flip-flop is neutral, a signal can fully propagate across the whole wire. In the gated state where the bottom half is shifted to the side, a signal will be blocked by the gate and only propagate a finite distance along the wire. Just like in Chapter 3, we initiate a signal by pushing on the left most beam of the wire to move it into the right-buckled state.

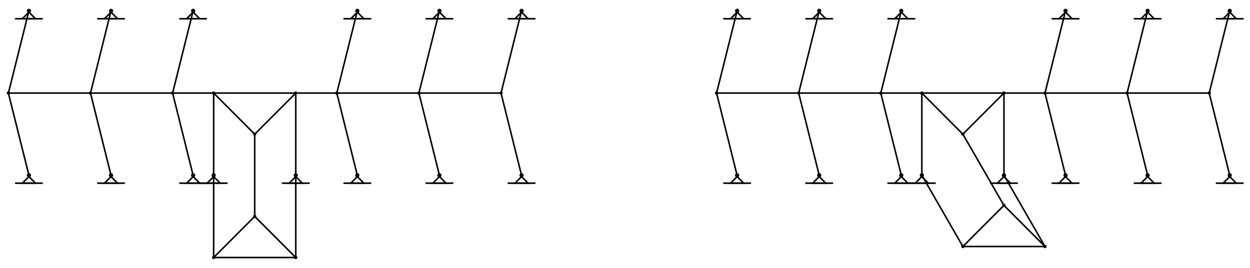


Figure 4.8: Left: the flip-flop in the un-gated state. Right: the flip-flop in the gated state.

Fig. 4.8 shows the flip-flop in both the un-gated and gated states. To actually switch between these two states, there needs to be some sort of input going to the bottom half of the flip-flop that causes this switch. It could be an external push from some other component, but because of how we chose to combine the flip-flop and wires, it's very easy to attach another wire to the bottom half. With this full design, all inputs to, and outputs from, the flip-flop are provided by mechanical wires (Fig. 4.9). To change the state of the flip-flop transistor from un-gated to gated, we send a signal along the bottom left wire.

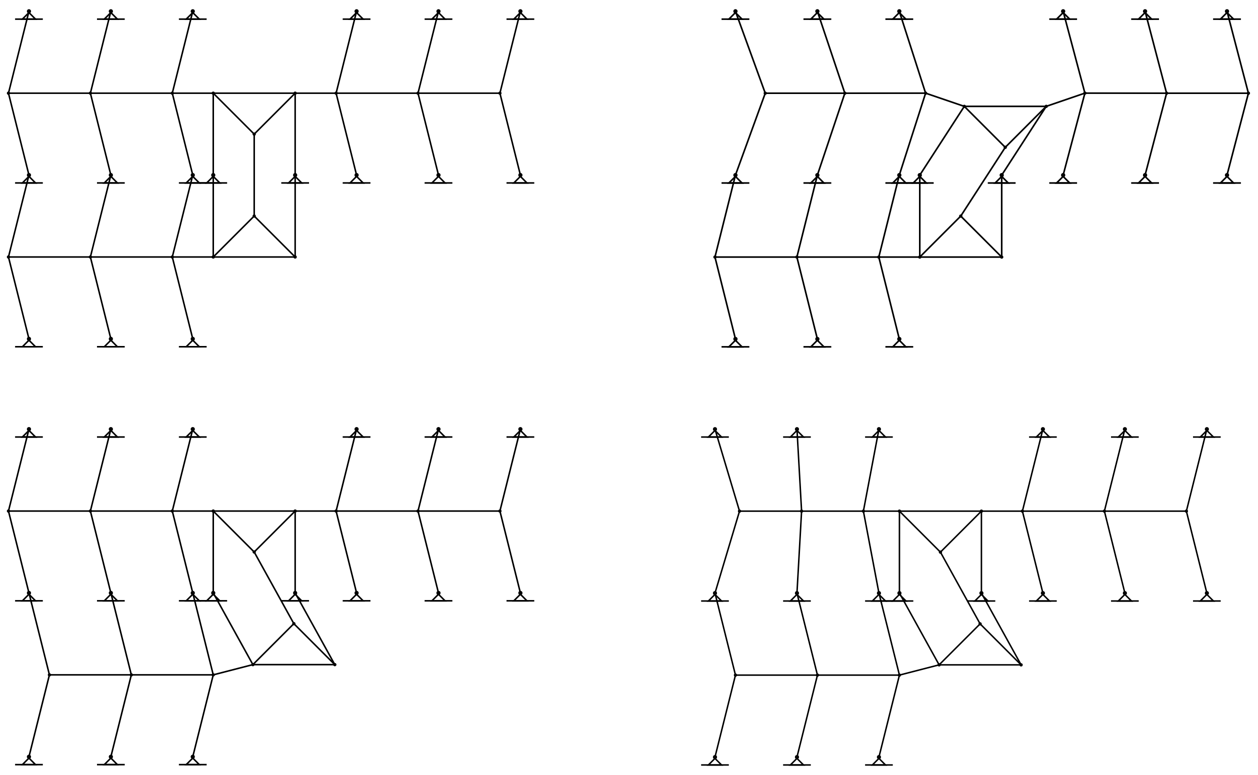


Figure 4.9: Top Row: the full device in the un-gated state before (left) and after (right) a signal is sent along the top wire. Bottom Row: the full device in the gated state before (left) and after (right) a signal is sent along the top wire.

Fig. 4.9 shows the results of sending a signal along the top wire when the flip-flop is gated and un-gated. In these simulations, we set all linear spring stiffnesses in the wires equal to 1 (including the springs connecting the wire to the flip-flop) and set the torsional modulus equal to 0, essentially removing the torsional spring at the midpoint of the bistable beams. For the first round of simulations, the flip-flop was made of rigid bars. With these combinations of stiffnesses, the flip-flop transistor behaved exactly as expected.

4.3.2 Flip-Flop Logic Gates

The device shown in Fig. 4.9 with three wires attached to it can be directly compared to the transistors used in electronic circuits. Fig. 4.10 shows both the flip-flop device and the circuit diagram symbol for a transistor with the corresponding inputs and outputs labeled. The input labeled base is where we send a signal to the transistor. When the base input reads a value of 1/high/+V, a value of 1/high/+V sent to the collector is transmitted across the transistor and read at the emitter location. The device shown in Fig. 4.10 has similar but opposite behavior. With a reading of 0/low at the base input, a value of 1/high/+V sent to the collector is transmitted across the transistor and read at the emitter location. While our device still has the gating property that a transistor has, the output behavior for a given input behavior is flipped around.

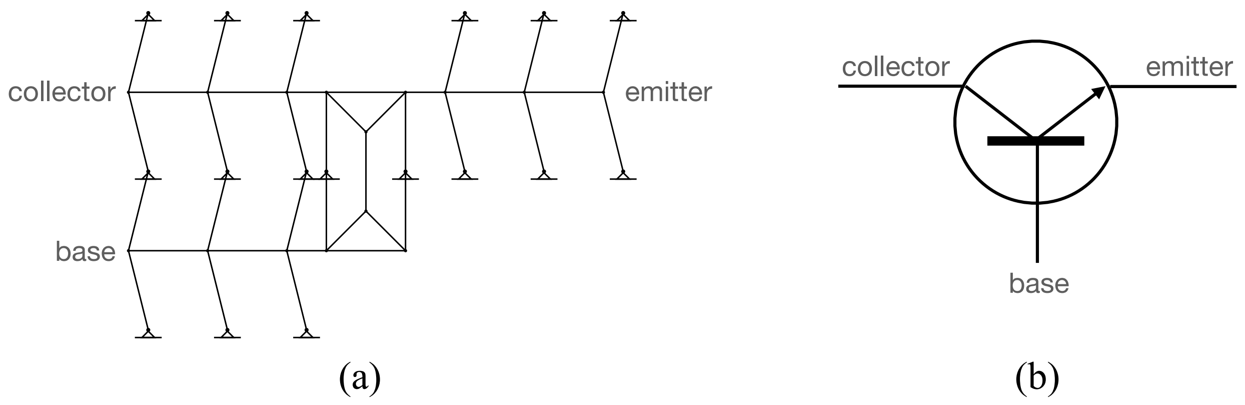


Figure 4.10: Comparison between (a) a flip-flop transistor and (b) an electronic transistor.

To remedy this, we can slightly modify the initial state of our wire and flip-flop transistor, as shown in Fig. 4.11. This new version of our device now behaves exactly like a transistor. When a signal is sent along the base wire (from right to left), it switches the flip-flop from the gated to un-gated state, allowing a signal to propagate from the collector endpoint to the emitter endpoint.

Unlike electronic transistors, our mechanical transistor has a bit of flexibility (figuratively

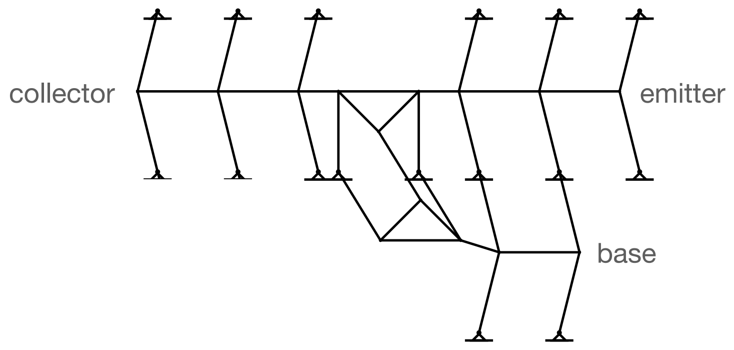


Figure 4.11: With this modification to the base wire and bottom half of the flip-flop, the device now behaves exactly like a transistor.

speaking) in how it behaves depending on the orientation of the base wire. That flexibility allows us to construct both a NOR and an AND gate. The truth tables for both gates are shown in Table 4.3.2

NOR

A	B	OUT
0	0	1
0	1	0
1	0	0
1	1	0

AND

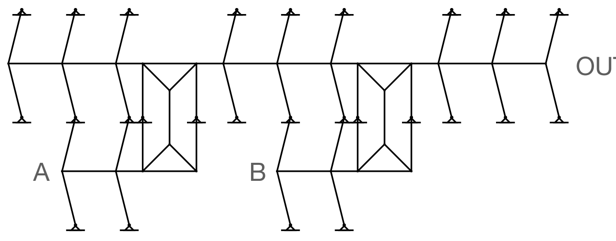
A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

Table 4.1: Truth tables for the NOR and AND gates.

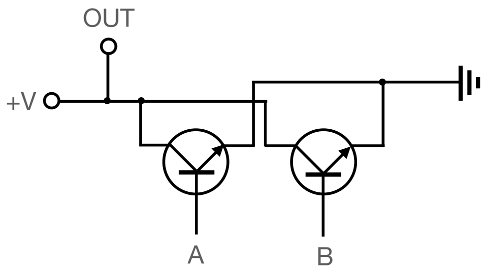
An input of 0 corresponds to us not sending a signal to that input. An input of 1 corresponds to us displacing the beam at that input and allowing a signal to propagate. Once we provide the proper signal to inputs A and B, we read the output by initiating a signal at the left end of the top wire.

Our version of a NOR gate uses the flip-flop transistor shown in Fig. 4.10. If a signal is initiated at either input A or at input B, the flip-flop is shifted into the gated state. When a signal is sent along the top wire, the flip-flop blocks that signal from propagating all the way to the OUT output.

Our version of an AND gate uses the flip-flop transistor shown in Fig. 4.11. If a signal

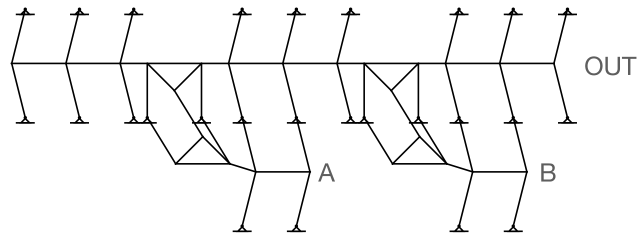


(a)

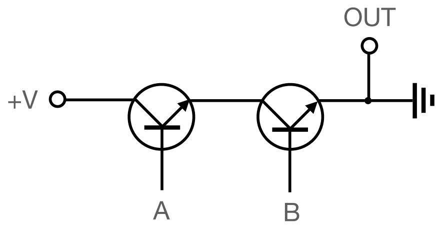


(b)

Figure 4.12: Comparison between (a) a NOR gate with flip-flops and (b) the circuit diagram for a NOR gate.



(a)



(b)

Figure 4.13: Comparison between (a) an AND gate with flip-flops and (b) the circuit diagram for an AND gate.

is initiated at either input A or at input B, the flip-flop is shifted from the gated state to the un-gated state. Both flip-flops need to be switched to the un-gated state in order for a signal on the top wire to propagate all the way to the OUT output.

4.3.3 Experimental Design and Testing

With a simulation demonstration of the flip-flop being used to create logic gates, we transitioned to designing an experimental version of the flip-flop that is compatible with our experimental mechanical wires shown in Chapter 3. We wanted to verify that our designs would work as 3D printed devices before investigating how to construct more complicated logic gates. The designing, printing, and testing of these devices was done by YongJae Kim and David Limberg. YongJae was responsible the design and testing of the mechanical wires themselves, and David focused on creating the flip-flop component and integrating it into YongJae's wires.

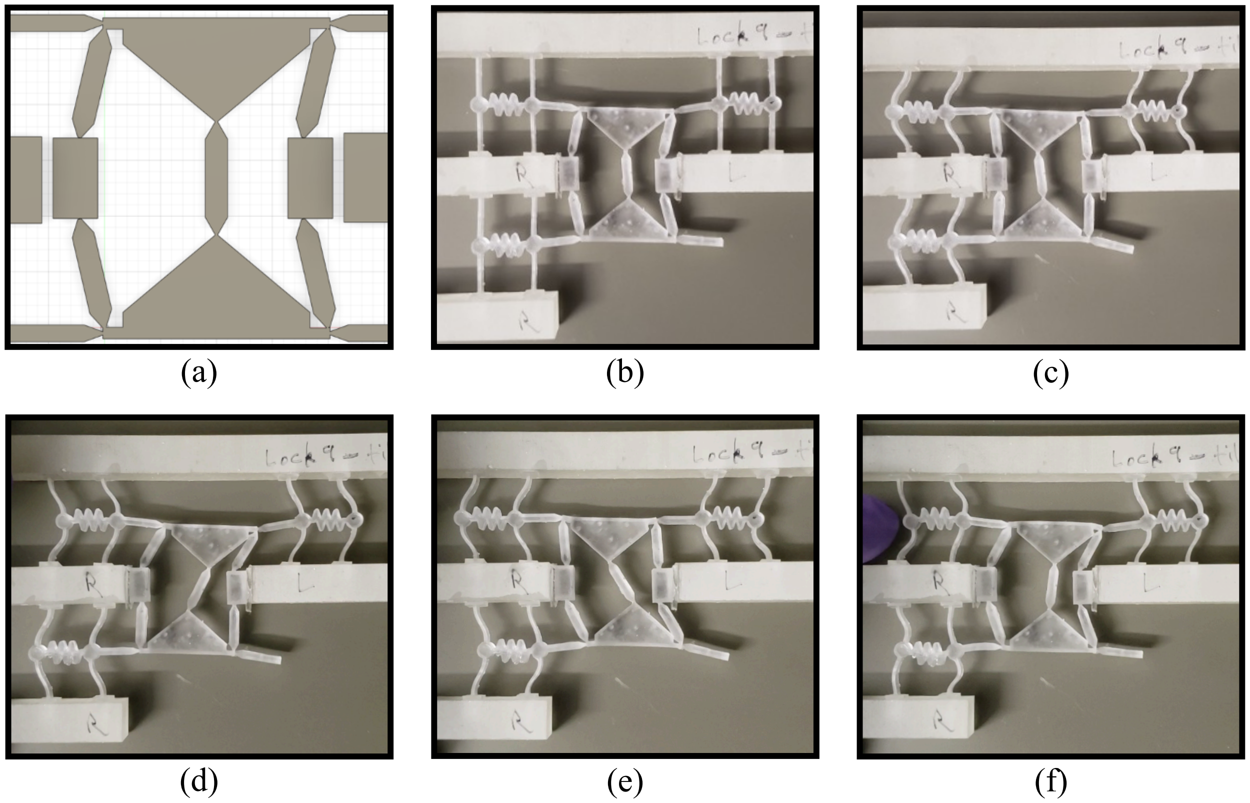


Figure 4.14: Experimental design of the flip-flop as a transistor in our mechanical wires. (a) Fusion 360 CAD design for a flip-flop that can be 3D printed with elastic resin. (b) A printed device assembled in the rigid frame with beams in the uncompressed state. (c) The device in the compressed state. (d) The device in the un-gated state after a signal has been sent along the top wire. (e) The device in the gated state. (f) The device failing to properly block a signal.

After multiple rounds of designing, printing, testing, and editing, we settled on a final flip-flop design shown in Fig. 4.14(a). The biggest difference between the simulation design and this design is that there is a large gap in-between the beams on each side of the flip-flop (in Fig. 4.1, these are the beams connecting vertices 7 and 4, 4 and 1, 8 and 5, and 5 and 2). The gap is there to accommodate the frame needed to hold and compress the mechanical wires. This gap was present in the original design for the flip-flop in [76], and we initially removed it to simplify the linkage analysis of the flip-flop. The gap does not affect the motions of the flip-flop because we can still keep all five vertical beams the same length.

The first design choice we made was to print the triangles in the flip-flop as solid pieces to keep them as rigid as possible. Because the flip-flop needs to be printed as one continuous piece of material to be compatible with our specific wire design, the rotary joints at each vertex needed to be replaced by flexures. Each beam has a uniform thickness everywhere except for the very ends where they taper to a point that is as small as allowed given the resolution limits on the 3D printer used.

Because we need the flip-flop to be in the neutral position when the beams are buckled, we needed to assemble the device in the frame with the flip-flop in a state where both halves are buckled to the right as shown in Fig. 4.14(b) (a position that is not allowed for a totally rigid device). Because our flip-flop is made of an elastic material, we can assemble the device this way without issue, but the flip-flop will be in a stressed state, with possible out of plane buckling, when the beams are un-compressed. Once we compress the beams, they will all shift to the left, which in turn will move the flip-flop to its neutral position as shown in Fig. 4.14(c).

Fig. 4.14(c) shows the flip-flop in the un-gated state. When a signal is initiated on the top wire from the left, the top half of the flip-flop shifts to the side and allows the signal to fully propagate (Fig. 4.14(d)). In this un-gated state, our device behaves as expected even with the multiple differences between the ideal, rigid flip-flop and our soft, 3D printed version.

When testing the gated state, we notice a lot of undesirable behavior. Fig. 4.14(e) shows the flip-flop in the gated state. When we initiate a signal on the top wire from the left, the flip-flop does not block the signal at all. The top half of the flip-flop shifts to the right, the signal fully propagates, and then the bottom half of the flip-flop shifts back to the left and returns to the un-gated position. A frame from a video of this motion is shown in Fig.

4.14(f). Just as the signal on the top wire finishes propagating, we see the bottom half of the flip-flop begin to return to the neutral position as shown by the position of the beams in the bottom wire. We refer to this behavior as a “backflow” problem. We intended to only send a signal from the left to the right along the top wire, but because of the breakdown in the flip-flop’s behavior, we saw signals going in the backwards direction (from right to left) along the bottom wire.

Our first attempt at investigating the backflow problem was to test the gated state while holding the bottom half of the flip flop in place. Unfortunately, this did not improve the behavior of the device. The elastic material that the flip-flop is made of is too compliant – the locking property of the flip-flop we wanted to use to get transistor-like behavior essentially disappeared completely. Instead of seeing the backflow problem, we saw out of plane buckling of the flip-flop.

4.3.3.1 Troubleshooting: Energy Landscape

In the previous simulations of the combined flip-flop and wire devices, we made the flip-flop completely rigid. Based on experiments, our assumption that the fip-flop would be rigid (or at least much stiffer than the wires) when constructed with a soft material was not correct. To investigate why the flip-flop did not behave correctly in the gated state and if there was a design change we could make to fix this behavior, we analyzed the energy landscape for a flip-flop made out of linear springs.

Fig. 4.15 shows the potential energy of the flip-flop when all 11 rigid bars are replaced with linear springs with a stiffness of 1 and rest length equal to the length of the original rigid bars. To collect our data, we displaced vertices 1 and 7 (Fig. 4.1) by some amount D_{top} and D_{bottom}, pinned vertices 1 and 7 in place, then let the system relax into the minimum energy position for that amount of displacement. We repeated this process for all combinations of displacements and visualized the results using ContourPlot[]. Because all of our units have

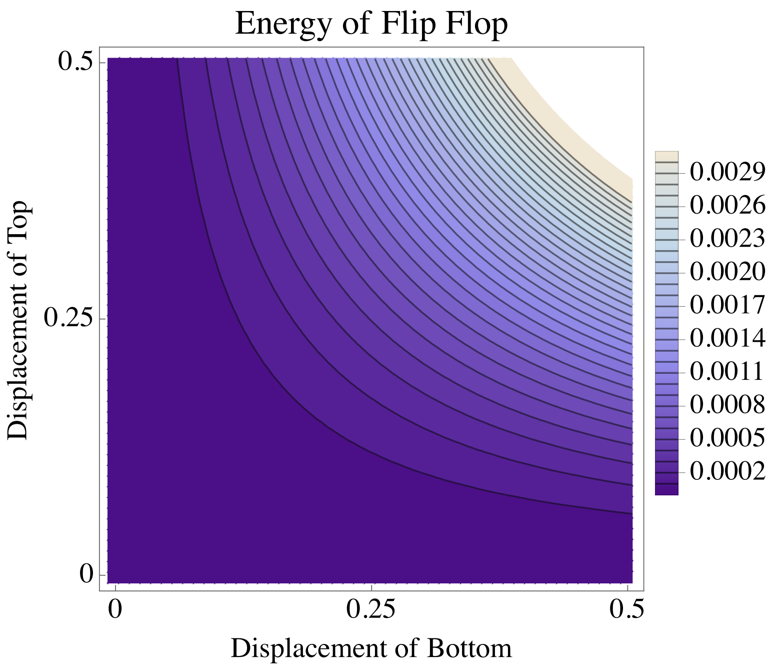


Figure 4.15: The total potential energy of the flip-flop when the top and bottom halves are shifted to the right.

been arbitrary, the values on the legend are just meant to indicate which contours correspond to higher and lower energies.

Because all of the bars in the linkage can now stretch and compress slightly, the flip flop can jump between the branches of its rigid configuration space more easily. For a completely rigid flip-flop, the configuration space lies along the x- and y-axis in Fig. 4.15. Ideally, we would keep this behavior with experimental versions and only the top OR the bottom of the flip flop could move. However, the energy cost of jumping directly from the bottom right corner to top left corner is low enough that the flip flop can perform this motion outside of the ideal configuration space. This transition is what we saw when the flip-flop behavior broke down in our transistor examples.

4.4 Summary

There were multiple roadblocks to creating functional flip-flop transistors for our mechanical wire design. While the method for integrating the flip-flop with our mechanical wires seemed promising in simulations, the system did not work well enough to use in devices made of soft, elastic materials. If the flip-flop is to be used as a transistor in a mechanical computation device, we need to either 1. change how we realize the flip-flop so that it works with the soft 3D printed wires, 2. change how we realize the mechanical wires so that it works with a rigid flip-flop that has rotary joints, or 3. find a new way to realize both the wires and the flip-flop.

Chapter 5

Configuration space engineering for the topological gating of mechanical devices

Abstract

Linkages are mechanical devices constructed from rigid bars and freely rotating joints studied both for their utility in engineering and as mathematical idealizations in a number of physical systems. Recently, there has been a resurgence of interest in designing linkages to perform certain tasks from the physics community. We describe a method to design the topology of the configuration space of a linkage by first identifying the manifold of critical points, then perturbing around such critical configurations. We then demonstrate our procedure by designing a mechanism to gate the propagation of a soliton in a Kane-Lubensky chain of interconnected rotors.

5.1 Introduction

Linkages serve as prototypical mechanical models for many different physical systems, including animal limbs and joints [10, 89, 66], polymer physics [72], protein allostery [97, 98, 40, 46, 104], DNA rigidity [71, 61] origami [65, 101, 67] and jamming [96, 21, 23]. At a basic level, a linkage is a graph whose edges have a fixed length but whose vertices are otherwise freely rotating joints. Yet this superficial simplicity belies behavior that can be surprisingly complex. One of the first important mathematical results was Kempe's universality theorem, which showed that a linkage can be designed such that a given vertex traces out a portion of any rational algebraic curve. [57, 62].

The results of following the proof's design procedures can be unwieldy for even simple curves, yet there are many applications where the precise motion of the vertices of a linkage is less important than the motion's qualitative features. An example is the celebrated Kane-Lubensky (KL) chain [50], a series of rotors joined by springs, which supports the propagation of a soliton called a "spinner" in which each rotor, in turn, rotates a full $360°$ degrees [16]. The existence and behavior of this soliton is robust under length changes to the rotors, depending on the topology of the configuration space rather than its particular shape [68]. Additionally, many linkages have branched configuration spaces, meaning that many different qualitative motions are accessible. For example, generic origami and kiragami mechanisms have highly branched configuration spaces, leading to pluripotency [39, 15]. Similar branched configuration spaces have been used to design mechanical logic devices [76] and kinematotropic mechanisms that can change how many degrees of freedom they can access [110, 34]. For flexible or imperfectly fabricated mechanisms, in which the fine structure of the motion cannot be controlled anyway, understanding how the topology of their configuration spaces relate to qualitative motions is crucial.

In this paper, we introduce an approach to linkage design that explores this relationship by focusing on the critical points of a configuration space [60, 31]. At a critical point, a mechanism has an anomalously large class of potential linear motions available to it, but higher order corrections from the mechanical constraints restrict the motion to a subset of these motions [80, 20, 22, 78]. Critical points are delicate; even small perturbations of the mechanism geometry will destroy them. However, by carefully controlling those perturbations, we show that we can construct a mechanism that allows some control over the configuration space's topology. Our design approach can be summarized in two steps: (1) design a mechanism with a branched configuration space, then (2) perturb the mechanism geometry away from the branched configuration space to a smooth one with controlled topology. To illustrate this approach, we will apply our design methodology to the KL chain. By replacing one of the unit cells with a designed mechanism, we show that the propagation of the spinner soliton can be controllably gated.

In Sec. 5.2, we review relevant parts of rigidity theory and mechanisms. In Sec. 5.3, we describe mathematical tools that provide a geometrical interpretation to critical points. This interpretation will provide the basis of our design methodology, which we will illustrate with an example containing five bars. Finally, in Sec. 5.4, we will use our formalism to explicitly design a mechanism to gate the KL chain. Importantly, the operation of the resulting gate is robust with respect to small perturbations. Finally, we conclude with a brief discussion highlighting new directions enabled by this work.

5.2 Critical points in mechanisms

5.2.1 Mathematical rigidity

In this section, we review the basic mathematical description of mechanisms. Though we focus on linkages, which are constructed entirely from free-rotating joints and inextensible bars, the formalism can be generalized to mechanisms with other holonomic constraints. We define a linkage as a collection of V vertices in d dimensions joined by E rigid bars. The configuration of a linkage can then always be represented by a point, $\mathbf{u}$, in the space of vertex positions, which we will denote as $\mathcal{M}$, and has dimension $M = Vd$. We assume there are E bars in the linkage and denote the length of the α^{th} bar, $\ell_\alpha(\mathbf{u})$. The configuration space of the linkage can then be represented by the family of equations,

$$\ell_\alpha^2(\mathbf{u}) = L_\alpha^2 \tag{5.1}$$

where L_α is the target length of the α^{th} bar. Note that Eq. (5.1) is written using the square of $\ell_\alpha(\mathbf{u})$ so that it is can be an analytic function everywhere. By replacing $\ell_\alpha(\mathbf{u})$ with a more general class of functions in Eq. (5.1), we can also describe mechanisms with more complex components beyond rigid bars.

Rather than analyzing the configuration space for specific values of L_α, we will instead analyze the entire family of configuration spaces that can occur with a fixed network topology by changing the L_α. Between Kempe's universality theorem and the potential arbitrariness of $\ell_\alpha(\mathbf{u})$, however, it is indeed difficult to say a great deal more about the configuration space with any kind of generality. Therefore, we assume that $\ell^2(\mathbf{u})$ is an analytic function of $\mathbf{u}$ and that $E \leq Vd$. With these assumptions, the Jacobian matrix, whose components are

$$J_{\alpha i}(\mathbf{u}) = \frac{\partial \ell_\alpha^2(\mathbf{u})}{\partial u_i}, \tag{5.2}$$

provides critical information about the mechanism. Naively, one would expect the configuration space of the mechanism to be $D = M - E$ (for $M > E$). Indeed, the inverse function theorem implies that the configuration space is a smooth D dimensional manifold in any open set of $\mathcal{M}$ in which the Jacobian matrix is full rank. At such a configuration $\mathbf{u}$, the tangent space coincides with the right null space of $J_{\alpha i}(\mathbf{u})$,

$$\sum_i J_{\alpha i}(\mathbf{u})\delta u_i = 0. \tag{5.3}$$

The solutions δu_i of Eq. (5.3) are called zero modes.

Any point $\mathbf{u}_C$ at which the Jacobian fails to be full rank, on the other hand, we call a critical point, and the corresponding edge lengths $\ell_\alpha^2(\mathbf{u}_C)$ we call a critical value. Critical points are characterized by self stresses, σ_α, which are elements of the left null space of $J_{\alpha i}(\mathbf{u}_C)$,

$$\sum_\alpha \sigma_\alpha J_{\alpha i}(\mathbf{u}_C) = 0. \tag{5.4}$$

Because of their relation to critical points, we will see that self stresses play an important role in the topology of the configuration space.

Sard's theorem ensures that critical values (but not necessarily critical points) are a set of measure zero. In that sense, most choices of edge lengths lead to a configuration space that is a smooth D dimensional manifold. Consequently, any change in the configuration space's topology that occurs as the L_α change must happen at a critical point. Thus, these critical points also govern the overall topology of the configuration space of a mechanism.

In the next section, we proceed to analyze the geometry of the configuration space at and near such critical points.

5.2.2 Shape of the configuration space at critical points

To understand the shape of the configuration space, we expand $\ell_\alpha^2(\mathbf{u}+\delta\mathbf{u})$ for small deformations, $\delta\mathbf{u}$ having components δu_i, around the critical point, obtaining

$$0 = \sum_i J_{\alpha i}\delta u_i + \frac{1}{2}\sum_{ij}\frac{\partial^2 \ell_\alpha^2(\mathbf{u}_C)}{\partial u_i \partial u_j}\delta u_i \delta u_j + \mathcal{O}(\delta u^3). \tag{5.5}$$

It is common at this stage to write a formal series expansion, $\delta\mathbf{u} = \delta\mathbf{u}^{(1)} + \delta\mathbf{u}^{(2)} + \cdots$, and substitute it into Eq. (5.5). One finds $\delta u^{(1)}$ is a zero mode of the Jacobian satisfying [22]

$$\frac{1}{2}\sum_{\alpha}\sum_{ij}\sigma_{\alpha}^{(n)}\frac{\partial^2\ell_{\alpha}^2(\mathbf{u}_C)}{\partial u_i \partial u_j}\delta u_i^{(1)}\delta u_j^{(1)} = 0, \tag{5.6}$$

where $\{\sigma_{\alpha}^{(1)}, \sigma_{\alpha}^{(2)}, \cdots\}$ is a basis for the space of self stresses at $\mathbf{u}_C$.

To proceed, we make further assumptions. The most important of these is that *Eq. (5.6) completely characterizes the local geometry of the critical point.* It is well-known that if no solution to Eq. (5.6) exists then the linkage is rigid, but the converse does not necessarily hold. There are mechanisms whose rigidity is only visible at higher order, as well as mechanisms that are rigid at order larger than two but, nevertheless, are mobile [20]. Experience suggests that these examples are rarer than the better behaved examples we consider here, but we are unaware of any results quantifying their rarity or even a simple means to determine when Eq. (5.6) is sufficient to describe the geometry of the critical point accurately. For the scope of this paper, it will prove sufficient to assume we can safely truncate our expansion of $\delta\mathbf{u}$ at second order and check, *post hoc*, that the results produced by our design procedure satisfy our assumptions.

We will make three other assumptions as well:

1 *All critical points, $\mathbf{u}_C$, lying on a configuration space of constant L_α are isolated.* There are linkages for which this fails and for which the entire configuration space lies along a sequence of critical points (see, for example, [91]). Note, however, that there are also mechanisms with large D which do satisfy this assumption [15, 51, 53]. In this paper, we will ultimately focus on example mechanisms with only a single degree of freedom, so this will not prove a particularly strong assumption, but in this section we allow D to be general and only specialize to $D = 1$ subsequently.

2 *All critical points have exactly one self stress.* This assumption is certainly not always true. It fails, for example, in flat origami mechanisms [15]. Generally, however, we will see that, qualitatively, critical points with several self stresses appear to require more fine-tuning. This assumption implies that there will be $D+1$ zero modes at each critical point by the rank-nullity theorem applied to the Jacobian matrix at $\mathbf{u}_C$.

3 *The matrix*

$$\sum_\alpha \sigma_\alpha \frac{\partial^2 \ell_\alpha^2(\mathbf{u}_C)}{\partial u_i \partial u_j}$$

has nonzero eigenvalues when restricted to the zero modes at $\mathbf{u}_C$. This assumption allows us to simplify the characterization of the critical points. Notice that without assumption 2, this characterization would be more difficult because the Eq. (5.6) would yield a system of quadratic equations rather than a single equation.

While all of these assumptions will play a role in our analysis, one could relax some of them at the expense of complicating the design procedure. Our examples will satisfy them, however, and we leave it for future work to understand which are truly required and which are conveniences.

Suppose we choose a basis for the zero modes at $\mathbf{u}_C$, $\{\boldsymbol{\zeta}_1, \cdots, \boldsymbol{\zeta}_{D+1}\}$, writing $\delta u_i^{(1)} = \sum_n c_n \zeta_{n,i}$. Then Eq. (5.6) becomes

$$\sum_{nm} \mathcal{Q}_{nm} c_n c_m = 0 \tag{5.7}$$

where $\mathcal{Q}_{nm}$ is a symmetric matrix given by

$$\mathcal{Q}_{nm} = \sum_{ij} \sum_\alpha \zeta_{n,i} \zeta_{m,j} \sigma_\alpha^{(1)} \partial^2 \ell_\alpha^2(\mathbf{u}_C) / \partial u_i \partial u_j. \tag{5.8}$$

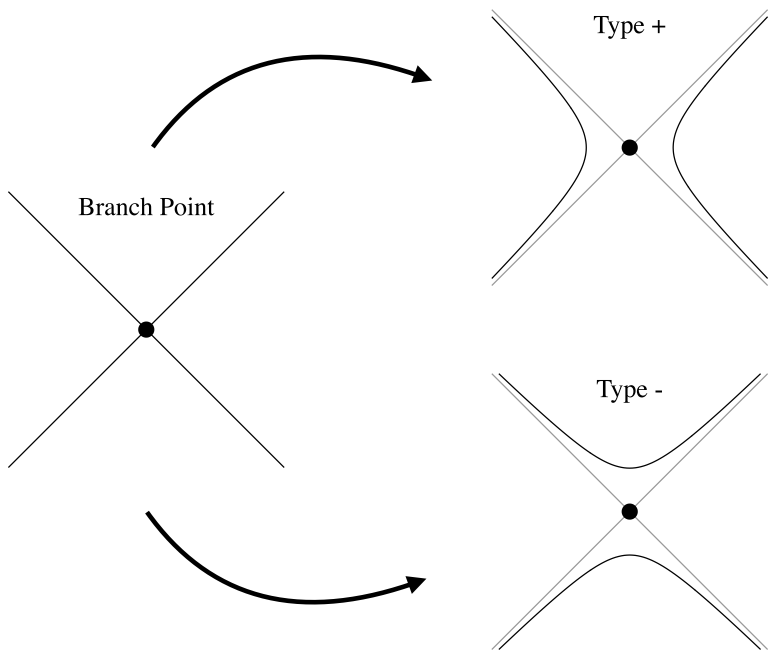


Figure 5.1: Schematic of how a configuration space with a branch point split into one of two types of smooth, disconnected configuration spaces. The choice of sign is arbitrary.

Under our assumptions, there are just two possibilities. If $\mathcal{Q}_{nm}$ is either positive- or negative-definite, the linkage is rigid: there is no solution to Eq. (5.7) other than $c_n = 0$. If $\mathcal{Q}_{nm}$ has a combination of positive and negative eigenvalues, however, the geometry of the configuration space at $\mathbf{u}_C$ is that of a cone. This is precisely what happens in single-vertex flat origami [15, 7] (Fig. 5.1). We call such a point $\mathbf{u}_C$ a branch point, though this space of possible zero modes is sometimes called a kinematic tangent cone [79].

5.2.3 Shape of the configuration space near critical points

We next ask what happens to the configuration space of a mechanism when the lengths are deformed from their critical values, $L_\alpha = L_\alpha^{(c)} + \delta L_\alpha$. A lengthy calculation shows (see Appendix 5.6)

$$\sum_{nm} \mathcal{Q}_{nm}(c_n - \delta c_n)(c_m - \delta c_m) = \Delta \tag{5.9}$$

where the deformation is along the zero modes at $\mathbf{u}_C$, $\sum_n c_n \zeta_{n,i}$ as before, and δc_n and Δ are quantities whose value depends linearly on the length changes, δL_α, to lowest order.

We first consider what happens when $\Delta = 0$. In that case, when $\delta c_n = 0$, Eq. (5.9) recovers the results from the previous section: there is either a rigid point or a branch point at $c_n = 0$ corresponding to the critical point $\mathbf{u}_C$. When $\delta c_n \neq 0$, however, the critical point itself moves by $\approx \sum_n \delta c_n \zeta_{n,i}$.

When $\Delta \neq 0$ and $\mathcal{Q}$ has only positive eigenvalues (the critical point is second order rigid), we have two possibilities: (1) $\Delta > 0$ implies the solution to Eq. (5.9) is an ellipsoid in $D+1$ dimensions (it is almost rigid [41]), and (2) $\Delta < 0$ implies there is no solution to Eq. (5.9). The opposite occurs if $\mathcal{Q}_{nm}$ has only negative eigenvalues.

Finally, we consider the case of a branch point, for which $\mathcal{Q}_{nm}$ has eigenvalues of opposite sign. To develop intuition, it is useful to consider the special case of a branch point when $D = 1$. Then $\mathcal{Q}_{nm}$ is a 2×2 matrix with two eigenvalues of opposite sign. The solutions to Eq. (5.9) take the form of two hyperbolas in the plane spanned by the zero modes at $\mathbf{u}_C$ whose precise configuration depends on the sign of Δ (Fig. 5.1 for characteristic examples for both signs of Δ). For $D > 1$, branch points also break up into smooth surfaces but do so, presumably, in a more complex way that depends on the signature of $\mathcal{Q}_{nm}$ (see Ref. [7] for an example in origami).

As an illustrative example, we turn to the well-studied four-bar linkage shown in Fig. 5.2a. The four-bar linkage is constructed from two rotors of length L_1 and L_3 pinned at one end and joined at the other by a bar of length L_2. The system configuration can be parameterized as a point in four dimensions with coordinates (x_1, y_1, x_2, y_2), and the configuration space is one dimensional. When $L_1 = L_2 = L_3 = a$, there are three branch points each having a single self stress and two zero modes. The configuration space is shown in Fig. 5.2b in terms of the two rotor angles θ_1 and θ_2. By slightly increasing the length of $L_2 > a$, the branch points all split into a pair of hyperbolas oriented opposite each other in the quadrants

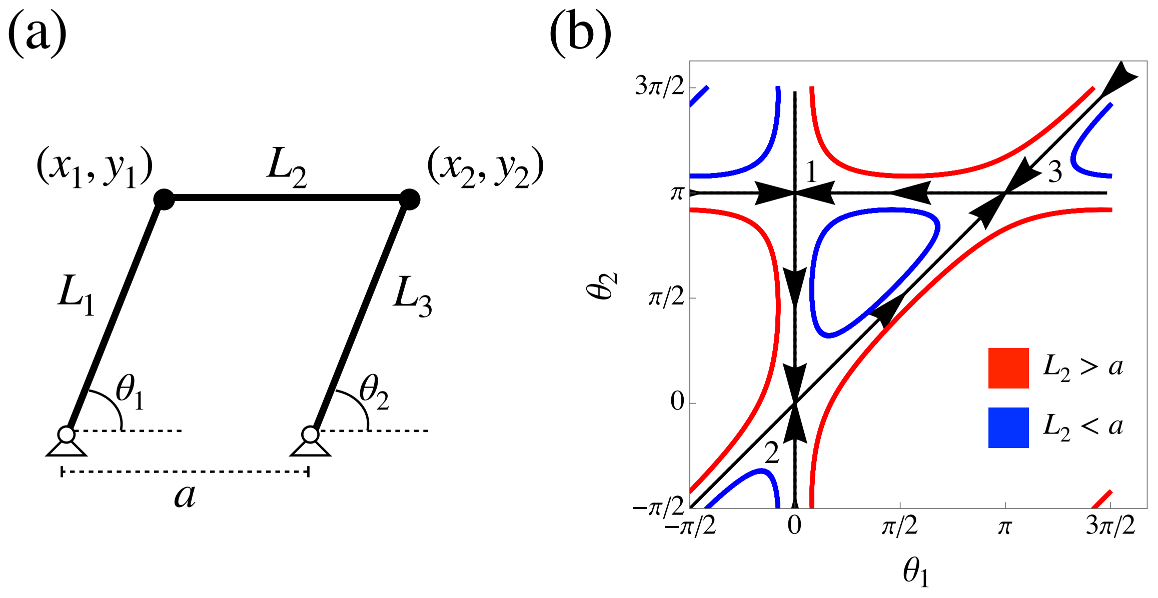


Figure 5.2: (a) Schematic of the planar, four-bar linkage with variables defined. (b) Projection of the configuration space of the two rotor mechanism with $L_1 = L_2 = L_3 = a$ projected into (θ_1, θ_2) plane (black). This choice of lengths has three branch-like critical points. Deforming the length of L_2 results in a smooth configuration space with either one (red) or two (blue) components. The arrows indicate the direction of the tangent form $t_i(\mathbf{u})$ from Eq. (5.10).

spanned by the configuration space when $L_2 = a$. On the other hand, $L_2 < a$ results in the branch point splitting into a pair of hyperbolas in the other pair of quadrants. As a result of switching the orientation of the hyperbolas, the configuration space goes from having a single component for $L_2 > a$ to two disconnected components when $L_2 < a$.

5.3 Controlling configuration space topology

We noted earlier that the topology of the configuration space cannot change without passing through an intermediate critical point. If it could, this would contradict the notion that the configuration space is smooth when the Jacobian $J_{\alpha i}$ is full rank. This fact and the analysis of Sec. 5.2 suggests a method for controlling the topology of the configuration space: (1) find a set of lengths L_α for which the configuration space has many branch points, and (2) perturb the lengths, $L_\alpha \to L_\alpha + \delta L_\alpha$, such that the branch points split into smooth hyperbolas in the desired configuration. For the four bar linkage in Fig. 5.2b, for example, if we could control how each of the three branch points split independently, we would have complete control over how the configuration space winds around the torus defined by the angles (θ_1, θ_2) as well as the number of components in the configuration space.

5.3.1 The geometry of the critical configuration set

Since we are interested in understanding how to choose edge lengths, L_α, to control the topology of the configuration space of a linkage, we will consider all possible mechanisms that have the same connectivity but arbitrary values of L_α. To do so, we define an antisymmetric tensor

$$t_{i_1 \cdots i_D}(\mathbf{u}) = \sum_{j_1 \cdots j_E} \epsilon_{i_1 \cdots i_D j_1 \cdots j_E} \frac{\partial \ell_1^2(\mathbf{u})}{\partial u_{j_1}} \cdots \frac{\partial \ell_E^2(\mathbf{u})}{\partial u_{j_E}} \tag{5.10}$$

where $\epsilon_{i_1 \cdots i_D j_1 \cdots j_E}$ is the antisymmetric Levi-Civita tensor. Importantly, $t_{i_1 \cdots i_D}(\mathbf{u}) = 0$ if and only if $\mathbf{u}$ is a critical point. This is because the components of $t^{i_1 \cdots i_D}$ are the $E \times E$ minors of the Jacobian matrix. When these all vanish the Jacobian matrix has lower rank (see Appendix 5.7 for a more detailed discussion). Thus, Eq. (5.10) identifies all possible critical points in mechanisms sharing the same connectivity. Versions of Eq. (5.10) have been studied to identify singularities in robot manipulators [102, 103, 64, 113, 28].

The tangent form allows us to define the *critical configuration set* as the locus of points for which

$$t_{i_1 \cdots i_D}(\mathbf{u}) = 0. \tag{5.11}$$

In many practical cases, and all of the cases we consider in this paper, it is possible to solve Eq. (5.11) analytically. Note however, that the solutions to Eq. (5.11) only provide the configurations where the Jacobian of the mechanism is not full rank. Therefore, some of the solutions may not satisfy all of our assumptions from Sec. 5.2.2. We conjecture that our assumptions are valid on all but a set of measure zero of the critical configuration set but are not aware of or able to produce a proof of this.

To help understand the geometry of the critical configuration set, we return to our previous example, the planar, four-bar linkage from Fig. 5.2a. In this example, $D = 1$ but $M = 4$ since the mechanism configurations are specified by points (x_1, y_1, x_2, y_2). If the two pinned vertices are located at $(0, 0)$ and $(a, 0)$ and we restrict $L_\alpha > 0$ (so no bars have zero length), this critical set is described by the two-dimensional manifold of configurations in which all vertices are co-linear, $y_2 = y_3 = 0$.

For one degree of freedom mechanisms ($D = 1$), Eq. (5.10) provides another way of understanding how the configuration space topology changes with changing lengths near a critical point $\mathbf{u}_C$. In that case, $t_i(\mathbf{u})$ is a vector field everywhere tangent to the zero modes of the mechanism, which follows from the simple fact that it is always orthogonal to the constraints (Appendix 5.7). Thus $t_i(\mathbf{u})$ can be thought of as a local vector field whose integral curves trace out curves of constant L_α. That is, when $t_i(\mathbf{u}) \neq 0$, curves of constant L_α can be parameterized by the solutions

$$\frac{du_i(s)}{ds} = t_i[\mathbf{u}(s)]. \tag{5.12}$$

We show this in Fig. 5.2b using arrows pointing along t_i projected onto the rotor angles. Because $t_i(\mathbf{u})$ is divergence-free (Appendix 5.7), each branch point has two arrows pointing in and two arrows pointing out. Note that $t_i(\mathbf{u})$ provides a way to think about the mechanism configuration space as a dynamical system. This dynamical system should not be confused with the motions of the physical mechanism, however, which can move either parallel or antiparallel to $t_i(\mathbf{u})$ equally well. This is also distinct from the dynamical system approach obtained for a periodic (or nearly periodic) mechanism as an iterated map [58, 59].

Eq. (5.12) also provides an intuitive way to understand the hyperbolas formed by the configuration space near branch points that arise from Eq (5.9). We project $t_i(\mathbf{u})$ near $\mathbf{u}_C$ onto the plane spanned by the two zero modes, $\boldsymbol{\zeta_1}$ and $\boldsymbol{\zeta_2}$. Since $t_i(\mathbf{u})$ is tangent to the configuration spaces, we expect the trajectories approach this plane as they approach $\mathbf{u}_C$. After projection, we obtain a 2D vector field whose components are,

$$T_n(c_1, c_2) = \sum_i \zeta_{n,i} t_i(\mathbf{u}_C + c_1\boldsymbol{\zeta_1} + c_2\boldsymbol{\zeta_2}). \tag{5.13}$$

The integral curves of T_n then trace the projection of the configuration space onto the plane spanned by the zero modes near the branch point.

In this projection, the constant L_α trajectories are quite limited in how they can appear. We know that $T_n(0,0) = 0$, but because we assume branch points are isolated, the projected tangent vector $T_n(c_1, c_2) \neq 0$ elsewhere. Now suppose that the critical point is a branch point. The projection of the configuration space on the plane of zero modes will have the form of a hyperbolic fixed point, with a stable and unstable manifold associated with the configuration space branches that solve Eq. (5.7) (Fig. 5.1). Thus, we would generically expect the trajectories near the branch point to appear hyperbolic when projected onto the plane of zero modes. Though we do not work with second order rigid points here, these considerations also limit what the trajectories do near such rigid points [41].

5.3.2 The geometry of the critical value set

For any point $\mathbf{u}_C$ in the critical configuration set, $\ell_\alpha^2(\mathbf{u}_C)$ gives its corresponding critical value: the set of squared bar lengths that would be required for the system to be in configuration $\mathbf{u}_C$. We will call the image of the critical configuration set in the space of squared lengths the *critical value set.* We again illustrate with the four-bar linkage: the set of critical values is a self-intersecting surface $(L_1^2, L_2^2, L_3^2) = (x_1^2, (x_2 - x_1)^2, (a - x_2)^2)$. In Fig. 5.3, we show the critical value set in terms of (L_1, L_2, L_3) rather than the squared lengths to make the

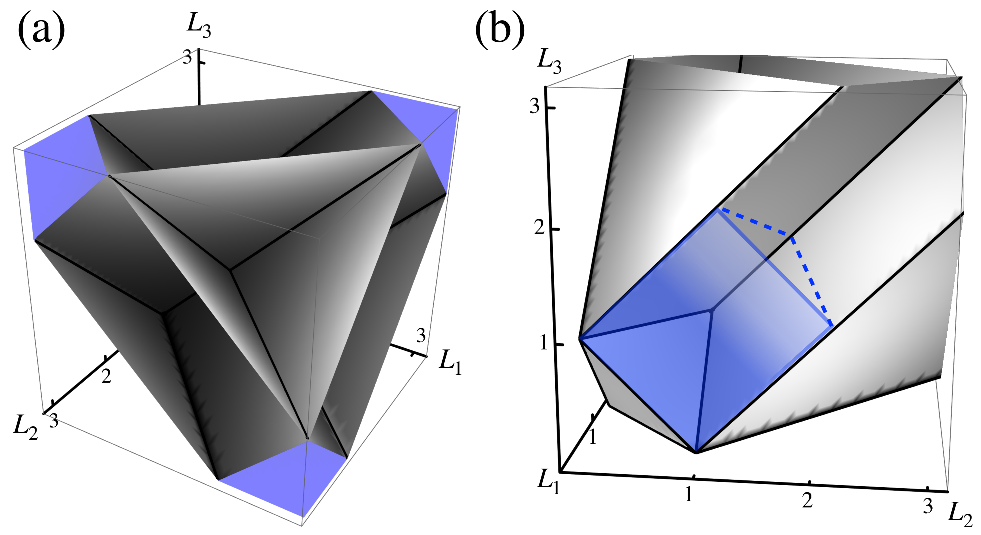


Figure 5.3: The critical value set for the four bar linkage, plotted in terms of (L_1, L_2, L_3) in units of a. There is one critical point in the configuration space along any smooth portion of the set. Self intersecting lines indicate choices with two critical points and the triply self intersection point at $(L_1, L_2, L_3) = (a, a, a)$ is the unique choice with three critical points. (b) shows a different view of the surface with a cutout on the $L_1 = 0$ plane showing the shape of one of the enclosed volumes.

surface slightly more compact and easier to understand visually. Note that we have included additional leaves on either the $L_1 = 0$, $L_2 = 0$, or $L_3 = 0$ plane which happen to contain only rigid critical points; this is a natural consequence of the fact that any mechanism with two pinned vertices and one edge having zero length must always be rigid.

Were we to choose the L_α to lie anywhere along the portion of the critical value set in Fig. 5.3, the resulting linkage would have one or more critical points. It is also apparent that Fig. 5.3 self intersects. At such a self-intersection, there will be multiple critical configurations, $\mathbf{u}_C$, corresponding to the same choices of edge lengths, L_α. Thus, if we choose the L_α along a line of self-intersection, there are two branch points. If we choose L_α along a smooth portion of the critical value set, there is only one critical point. Interestingly, Fig. 5.3 shows that at $(L_1, L_2, L_3) = (a, a, a)$ three individual sheets self intersect. Therefore we expect that that choice of L_α is the unique place where three branch points coincide (as seen in Fig. 5.2b).

The critical value set contains more information than just the location of critical values. If the critical value set is locally a smooth manifold, the self-stresses at such a critical point are always normal to the critical values (see Sec. 5.7). Though the converse is not generally true – normals need not also be self stresses – if the critical value set is a manifold of codimension one it must necessarily coincide with the single self stress at that point and there can be no other self stresses. We also see that splitting a branch point amounts to choosing δL_α transverse to the critical value set. On one side of the surface in Fig. 5.3, a branch point splits into one pair of smooth branches; on the other side it splits into the opposite pair. This endows the calculation of how branch points split under small perturbations of the lengths with a concise geometrical meaning.

With this understanding of the critical value set, we can classify the distinct configuration spaces of the four-bar linkage in terms of the $2^3 = 8$ individual volumes enclosed by the surface in Fig. 5.3. For completeness, we note that these volumes correspond to standard results for the four-bar linkage found in the engineering literature, which can be classified by the sign of three functions [73]

$$\begin{aligned} \tau_1 &= a - L_1 + L_2 - L_3 \\ \tau_2 &= a - L_1 - L_2 + L_3 \\ \tau_3 &= L_2 + L_3 - a - L_1 \end{aligned} \tag{5.14}$$

derived from limits on the angles θ_1 and θ_2. When one of the τ_i are equal to zero, the configuration space contains the corresponding critical point from Fig. 5.2b. Thus,the critical value set in Fig. 5.3 agrees with the surfaces computed in Ref. [5, 81].

5.3.3 Three rotor system

Finally, in this section we will use these considerations to describe a design procedure for configuration space topology. To be concrete, it is helpful to consider a specific example, the three-rotor linkage in Fig. 5.4a. The three rotor linkage has three pinned joints attached to three bars of length r_1, r_2, and r_3 (the rotors) and whose opposite ends are joined by bars of length L_1 and L_2. Therefore, $\mathbf{u}$ is a six component vector and the five bars provide constraints, $\ell_\alpha^2(\mathbf{u}) = L_\alpha^2$, that limit the configuration space to a single degree of freedom generically.

Since the mechanism has five bars, it is difficult to visualize the critical set and critical value set. Nevertheless, we can still gain insight by restricting ourselves to the cross-section of $\mathcal{M}$ for which $r_1 = r_2 = r_3 = a$. We plot the cross section of the critical value set with the (L_1, L_2) plane in Fig. 5.4b. While this is a cross-section, the open regions in Fig. 5.4b still correspond to structures with different configuration space topologies, with the transitions from one distinct region to another through the critical value set occurring through a branch point. However, it is still a cross section of a higher dimensional space and care must be taken when interpreting the intersections of the critical value set. Choosing $L_1 = L_2 = a$ leads to a configuration space with twelve interconnected branch points, though it appears that only two lines meet at $L_1 = L_2 = a$ in Fig. 5.4b. The proliferation of branch points in this example can be understood from the fact that this linkage contains two pairs of four-bar linkages. Choosing all bars to have length a, therefore, maximizes the branch points of each individual sub-mechanism.

To identify these branch points, we solve $t_i(\mathbf{u}) = 0$ subject to the length constraints $\ell_\alpha^2(\mathbf{u}) = L_\alpha^2$ using Mathematica (Wolfram). At each critical point, we then solve Eq. (5.7) to obtain the tangents to the configuration space. The trajectories in Fig. 5.4c are obtained by first stepping along one of the obtained tangent vectors, then stepping along the configuration space in the direction indicated by $t_i(\mathbf{u})$ with a step size proportional to its magnitude. The step size is adjusted to maintain the edge lengths to less than one percent strain. Finally, the integration for each segment is terminated when the magnitude of $t_i(\mathbf{u})$ falls below a critical threshold, indicating that the integration has reached a point close to the next critical point.

Once terminated, we minimize $\sum_i t_i^2$ with respect to the configuration to verify that the integration has found the next branch point. The directions of integration inherited from $t_i(\mathbf{u})$ are indicated by the arrows in Fig. 5.4c.

Note, however, that the branch points shown in Fig. 5.4c are not all independent. Projecting the configuration space onto the θ_1-θ_2 plane must give the configuration space of the equivalent four-bar linkage found by ignoring the third rotor. In contrast, removing the first rotor is equivalent to the projection onto the θ_3-θ_2 plane. Consequently, any branch points that overlap in one of these two projections must, after a deformation, still be identical in projection and such overlapping branches appear or disappear together. From Fig. 5.4, this implies that branch points are paired $\{(1,2),(3,5),(4,6),(7,8),(9,11),(10,12)\}$.

We finally consider how to "program" the configuration space by adjusting the lengths of r_1, L_1 and L_2 away from their critical values. For each critical point, we plot the domain over which $\Delta > 0$ in Fig 5.5 as a table for each branch point. We next choose lengths according to the red dot in Fig. 5.5a, which increases r_1 at constant L_1 and L_2. The resulting configuration space is shown as the red curve in Fig. 5.5b. Note that the red point was chosen so that the configuration space is smooth but passes near the branch points. If the red curve has the topology we want already, we can stop now. If we instead wanted to switch the sign of the branch point pair $(4,6)$ to obtain a particular configuration space topology. From Fig. 5.5a we see that the three lengths (r_1, L_1, L_2) distinguish this pair of branch points from the rest. Inspection of Fig. 5.5a suggests an additional change in L_1 would switch the way only those two branch points split. The result of this perturbation is the blue curve in Fig. 5.5b. Note that Fig. 5.5b shows that, since each branch point has one self stress, the hyperbolas approach the plane spanned by the two zero modes as expected.

If we limit ourselves to perturbing only the bar lengths (r_1, L_1, L_2), Fig. 5.5a shows even more redundancy in how the branch points split than expected from our previous analysis that branch points split in pairs. That is, just three control lengths are not sufficient to obtain

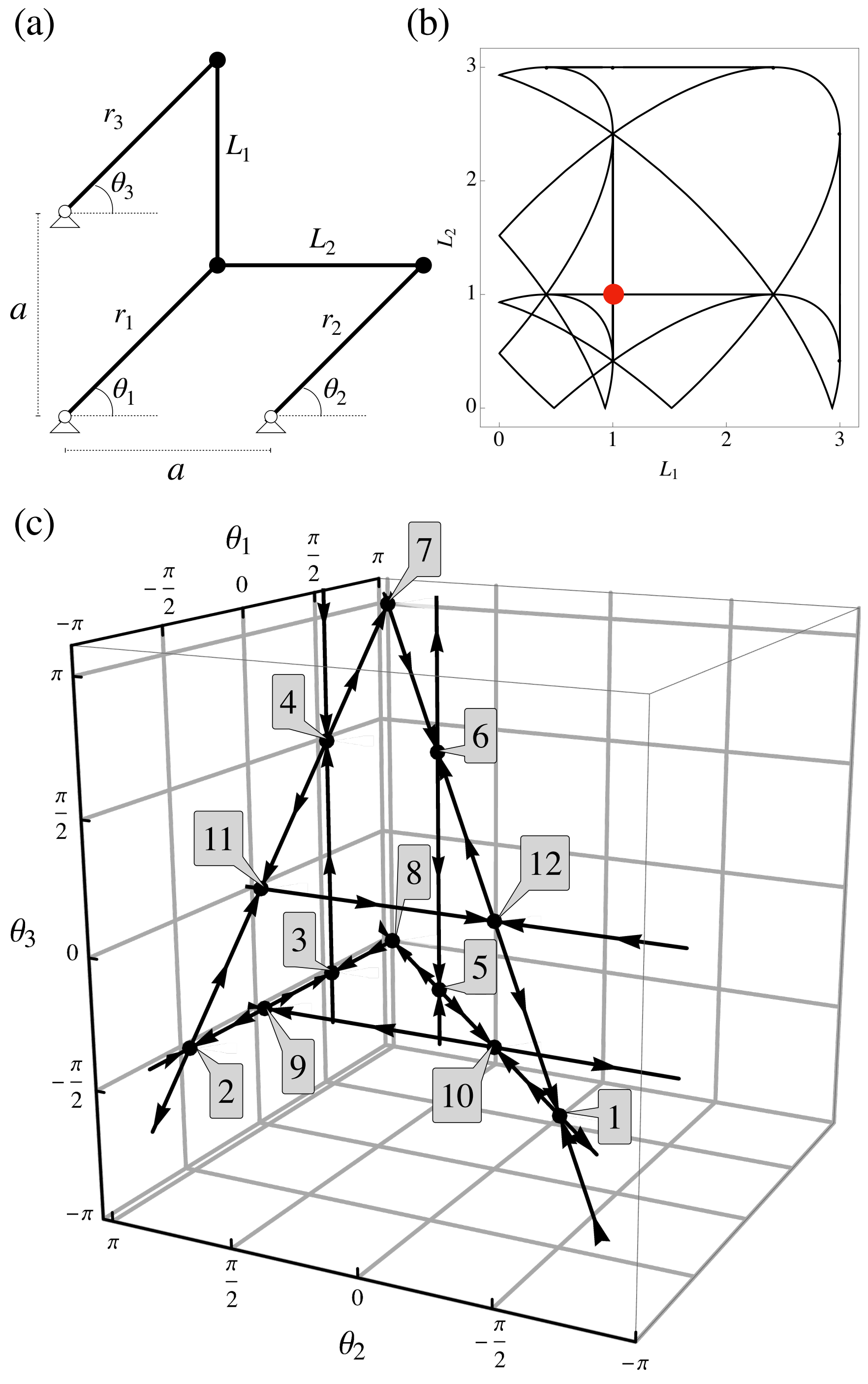


Figure 5.4: (a) Schematic of the planar, three rotor linkage with variables defined. (b) A cross-section of the critical value set for $r_1 = r_2 = r_3 = a$. (c) The 3D configuration space of the three rotor linkage with $r_1 = r_2 = r_3 = L_{13} = L_{23} = a$, corresponding to the red point in (b), contains twelve individual critical points. Arrows indicate the orientation of each configuration space segment.

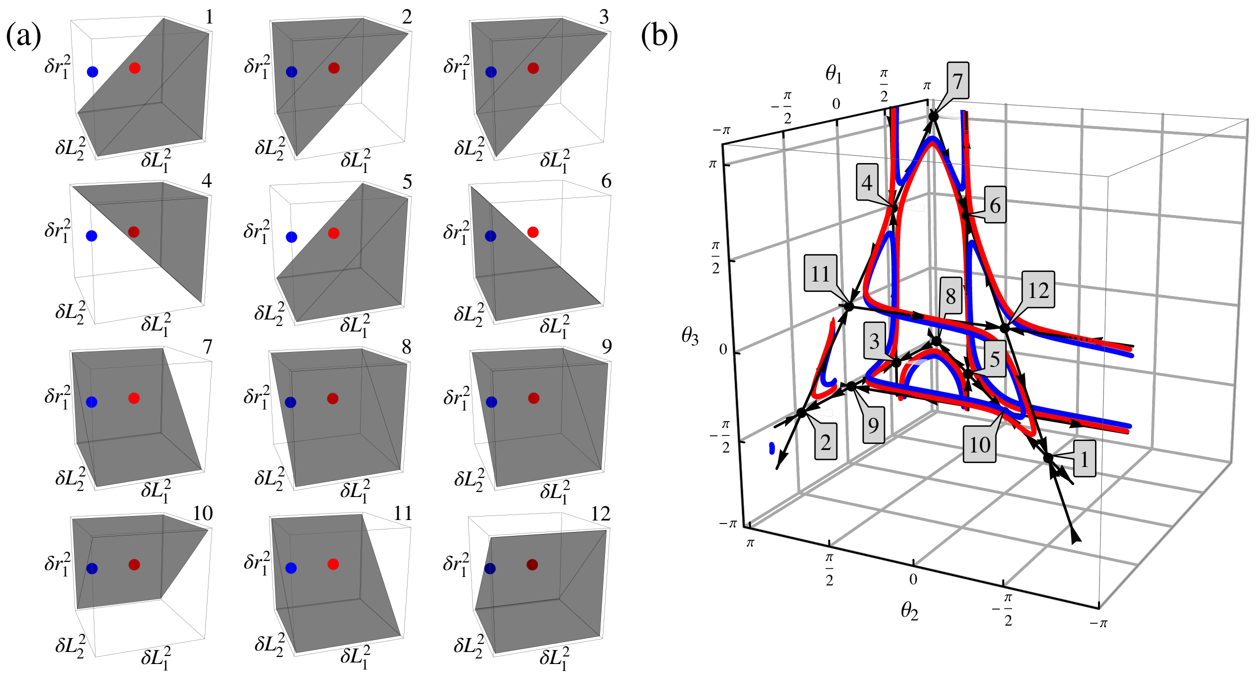


Figure 5.5: A schematic of programming the three rotor system. (a) A map showing how changing the length of r_1, L_1, and L_2 leads to different ways to split the branch points from Fig. 5.4. The red dot, corresponding to a change in length $r_1 = 1.05a$, leads to the red curve in (b). In order to change the topology of the configuration space by changing how branch points 4 and 6 split, an addition change to $L_1 = 0.9a$ can be effected (blue dot). The new configuration space is shown in (b) as a blue curve.

full control over the way the configuration space splits at the branch points. While it would be difficult to plot Fig. 5.5 using all five bar lengths, there seems to be no mathematical obstacle to generalizing the analysis to distinguish all six pairs of branch points independently.

5.4 The gated Kane-Lubensky chain

We finally apply our design methodology to design a mechanism that gates the propagation of a soliton in the Kane-Lubensky (KL) chain [50]. The KL chain is a topologically polarized lattice of rotors that has a zero mode on either the left edge or the right edge, depending on the choice of bar lengths. It was later discovered that the KL chain actually supported two distinct families of propagating solitons, the “flipper” and the “spinner” [16], that allowed a continuous pathway between the left and right edge modes. The spinner soliton, however, is topologically protected by the shape of the configuration space [16, 68]. In this section, we modify a single unit cell of a spinner-supporting KL chain with rotor length $r = 3a/2$ and $\ell = 3a/2$ by adding an additional two bars and one pinned vertex (Fig. 5.6a).

In the spinner phase of the KL chain, a full cycle consists of the soliton traveling back and forth across the chain once, and the KL chain returning to its initial configuration. After one full cycle, each rotor in the KL chain has rotated by 2π, with each rotor rotating by π each time the soliton passes. Here, we will show that these additional components can act as a gate by opening a gap in the full 2π rotation of the KL chain rotors, thereby obstructing the passage of the soliton.

In addition to the length of the two additional bars, L_1 and L_2, we also allow the location of the pinned vertex to be set an arbitrary distance D from the KL chain. In order to allow for different positions of the third pinned vertex, we augment $\mathbf{u}$ to include the y coordinate of the 3rd vertex but also augment the constraint functions to pin that vertex's y position. Thus, we use a constraint map

$$f_\alpha(\mathbf{u}) = \begin{pmatrix} \ell_1^2(\mathbf{u}) \\ \vdots \\ \ell_E^2(\mathbf{u}) \\ D^2(\mathbf{u}) \end{pmatrix}, \tag{5.15}$$

where $D(\mathbf{u})$ is the function that determines the distance between pinned vertex 3 and 2. Thus, the generalized constraints $f_\alpha(\mathbf{u})$ is a smooth function whose solution allows us to pin vertex 3 by setting the length D in Fig. 5.6a to an arbitrary value.

Using this generalized formulation, we can compute a cross section of the critical value set with $r = \ell = 3a/2$ and $L_1 = 2a$ (Fig. 5.6b). Fig. 5.6b shows that there are six distinct regions separated by critical points. The labels on each region correspond to the sign of

τ_1, τ_2, and τ_3 from Eq. 5.14 with respect to the four-bar linkage between vertices two and three. For concreteness, we choose $L_2 = 2a$ and $D = 5a/2$, on the boundary between the blue and red regions, as the initial lengths for our gate, resulting in a configuration space with two critical points (Fig. 5.6d). When $D < 5a/2$, the system is in the "red" regime and when $D > 5a/2$ it is in the "blue" regime. This choice determines whether the KL chain rotors wind around fully or not. Note that the projection of the configuration space in the θ_1-θ_2 plane never changes shape, but that the change in how the branch points split into hyperbolas determines whether the full range of angles is accessible to the system or not. To verify that the red and blue regimes correspond to ungated and gated behavior of the KL chain device, we use the *mechanisms* package [1] in Mathematica (Wolfram) to calculate the infinitesimal motions of the linkage and animate the those motions. As shown in Fig. 5.7, changing D controls whether or not the soliton can complete a full cycle along the KL chain.

To test our design, we constructed the gated KL chain in Fig. 5.6a numerically and constructed a single unit cell and gate from LEGO™ pieces. The design of the LEGO gate was chosen to be compatible with the LEGO realization of a KL chain shown in [16]. When testing different examples, we pushed on the various bars and rotors in the device to move it through all possible configurations. We tracked how the rotors 1 and 2 moved to determine if the gate was preventing a soliton from propagating. Fig. 5.7 shows a comparison between the simulated and LEGO chains with both D larger and smaller than $5a/2$. Movies of both chains in the gated and ungated states are provided in the Supplementary Material. In the case of an ungated chain, the soliton propagates from one end of the chain to the other (and back); for a gated chain the soliton propagates up to the location of the gate but is reflected.

Interestingly, the size of the gap in Fig. 5.6d is important for determining how the soliton is reflected from the gate. For very small gaps, which occurs when D is close to its critical value, the soliton can, temporarily, pass the gate but is, ultimately, prevented from

[1] Availabe on https://github.com/cdsantan/mechanisms

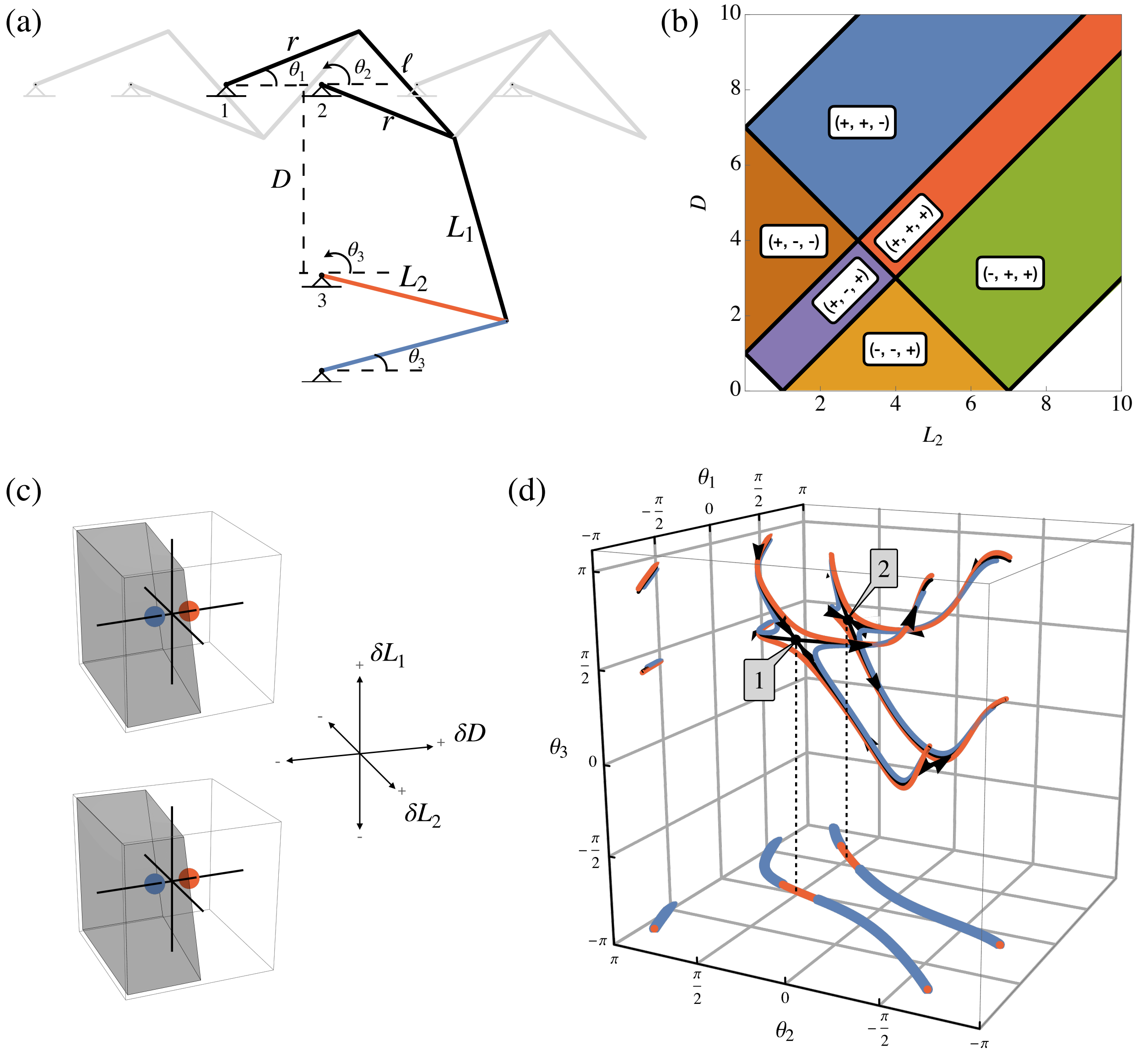


Figure 5.6: (a) A gated and ungated Kane-Lubensky chain controlled by the length D. (b) A cross-section of the critical set with $r = \ell = L_1 = 3a$ and $L_2 = 4a$. There is a critical point at $L_2 = 4a$ and $L_4 = 5a$. (c) Changing the position of the third rotor or the lengths of two beams can control whether the chain is gated or ungated. (d) The configuration space at and near the critical point as a function of the three rotor angles, and the projection of that configuration space onto the θ_1-θ_2 plane.

completing an entire cycle. For larger gaps, when D is farther from its critical value, the soliton appears to reflect from the gate. From Fig. 5.6c, the same effect can be achieved by changing the size of L_2 instead of D, since the plane divided the gray region from the transparent region is slightly angled in that direction. Movies of both simulated and LEGO chains that switch between the gated and ungated states by changing L_2 are also provided in the Supplementary Material.

Our analysis shows that the presence of a gap in the (θ_1, θ_2) plane blocks soliton propagation. In the example of Fig. 5.6a, changing the length D moves the device from from the gated (blue) region to the ungated (red) region of Fig. 5.6b. However, this is not the only pair of regions that produces a functioning gate. Indeed, the regions indicated in Fig. 5.6b as $(+,+,+)$ and $(-,-,+)$ are ungated with respect to propagation of the soliton, whereas the remaining regions are gated. Numerical experiments further show that if we had chosen L_1 to change length as well, we would have found even more regions of both gated and ungated behavior as we extended Fig. 5.6b. It becomes clear that there is a great deal of flexibility when choosing D, L_1, and L_2 to produce the desired dynamics of the final KL chain and gate system.

5.5 Conclusions

In this paper, we have described a procedure to design the topology of the configuration space of mechanical linkage. The idea rests on the ability to identify critical points and, especially, branch points – singular configurations of a linkage in which several pathways meet. By analyzing the shape of the configuration space near these branch points, we are able to design perturbations to the lengths and positions of a fixed set of vertices that change the shape of the topology of the configuration space in well-defined ways. As a demonstration, we used our techniques to design a gate for the propagation of the spinner

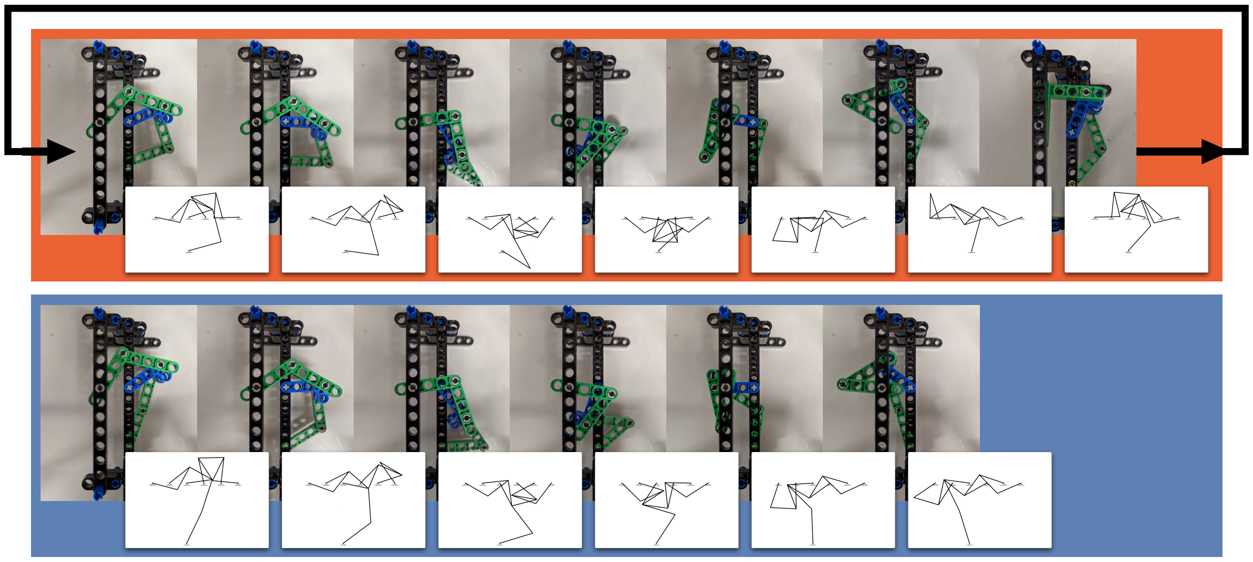


Figure 5.7: Top row (red): Ungated device made from LEGOs with the corresponding simulation. This device can continue rotating and return back to its initial position, as indicated by the arrow. Bottom row (blue): Gated device made from LEGOs with the corresponding simulation. This device gets stuck in the configuration shown in the last frame and is forced to reverse direction in order to continue moving.

soliton in a Kane-Lubensky chain. While we applied our approach to linkages with fixed edge length, there is no reason they would not also apply more generally to other systems with holonomic constraints.

Because the design procedure works by controlling configuration space topology, the resulting mechanisms should be quite robust to fabrication errors and the tolerance of the joints, so long as one chooses lengths L_α sufficiently far from the critical value set.

It would be interesting to extend this work in a few further directions. First, when bars are no longer rigid but elastic, there arises the possibility of a snap through transition between the different hyperbolas on either side of a branch point. Indeed, tuning various branches close to or farther from a branch point could be used to tune the ease of initiating a snap through transition. This could potentially lead to mechanical structures and mechanical metamaterials whose mechanical response can be reprogrammed *in situ*.

A second interesting extension would be to consider mechanisms built from responsive materials that are sensitive to external stimuli. In that case, the dynamic increase or decrease in the lengths of bars could be used to drive the pathway of a mechanism in an environmentally dependent manner. This could also be affected if the positions of certain pinned vertices could be made to depend on the external environment or the state of a second input mechanism. This would enable the realization of simple mechanical logic that is robust to some damage because it relies only on the topology of a configuration space [99, 75].

Finally, we note that our design principle exploits the fact that the configuration space topology can only change at critical points – configurations where the Jacobian of the constraints fails to be full rank. Our approach is somewhat reminiscent of Morse theory, in which the extrema of a scalar function can be related to the topology of the space on which that function is defined [86]. Morse theory has been used to study the configuration spaces of spherical (and other) linkages [51, 53], but we leave it to future work to make this connection more precise.

5.6 Appendix A: Quadratic critical point decompositions

In this appendix, we will show that Eq. (5.9) does, indeed, describe the configuration space near a critical point when the lengths of a linkage are perturbed from their critical values. We assume we have a mechanism with E edges and V vertices in d dimensions with $dV > E$. We further suppose that the configuration of the mechanism is at a critical point, $\mathbf{u}_C$, with corresponding critical values $(L_\alpha^{(c)})^2$. Let $\mathbf{u} = \mathbf{u}_C + \delta\mathbf{u}$ and correspondingly $L_\alpha = L_\alpha^{(c)} + \delta L_\alpha$, and expand the squared lengths to quadratic order, using Eq. (5.1,

$$2L_\alpha^{(c)}\delta L_\alpha + \delta L_\alpha^2 = \sum_i \frac{\partial \ell_\alpha^2(\mathbf{u}_C)}{\partial u_i}\delta u_i + \sum_{ij} \frac{1}{2}\frac{\partial^2 \ell_\alpha^2(\mathbf{u}_C)}{\partial u_i \partial u_j}\delta u_i \delta u_j \tag{5.16}$$

Finally, as in the main text, we assume that Eq. (5.16) completely characterizes the critical point, and that there is one self stress at $\mathbf{u}_C$, with components σ_α, and two zero modes, with components $\zeta_{1,i}$ and $\zeta_{2,i}$.

It will prove convenient to express Eq. (5.16) using an orthonormal basis in the space of square lengths, $\{\sigma_\alpha, e_\alpha^{(1)}, \cdots e_\alpha^{(E-1)}\}$. We similarly write δu_i in an orthonormal basis $\{\zeta_{1,i}, \zeta_{2,i}, \eta_{1,i}, \cdots, \eta_{E-1,i}\}$,

$$\delta u_i = c_1 \zeta_{1,i} + c_2 \zeta_{2,i} + \sum_{I=1}^{E-1} a_I \eta_{I,i}. \tag{5.17}$$

We first contract Eq. (5.16) with σ_α, we obtain an equation that can be expressed as

$$\begin{pmatrix} \mathbf{c}^T & \mathbf{a}^T \end{pmatrix} \begin{pmatrix} \mathcal{Q} & \mathcal{B} \\ \mathcal{B}^T & \mathcal{M} \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \mathbf{a} \end{pmatrix} = \tilde{\Delta}, \tag{5.18}$$

where the components of the matrices are given by

$$\tilde{\Delta} = \sum_\alpha \sigma_\alpha \left(2L_\alpha^{(c)} \delta L_\alpha + \delta L_\alpha^2 \right), \tag{5.19}$$

$$\mathcal{Q}_{nm} = \frac{1}{2} \sum_{\alpha i j} \zeta_{n,i} \zeta_{m,j} \sigma_\alpha \frac{\partial^2 \ell_\alpha^2}{\partial u_i \partial u_j}, \tag{5.20}$$

$$\mathcal{M}_{nm} = \frac{1}{2} \sum_{\alpha i j} \eta_{n,i} \eta_{m,j} \sigma_\alpha \frac{\partial^2 \ell_\alpha^2}{\partial u_i \partial u_j}, \tag{5.21}$$

and

$$\mathcal{B}_{nm} = \frac{1}{2} \sum_{\alpha i j} \zeta_{n,i} \eta_{m,j} \sigma_\alpha \frac{\partial^2 \ell_\alpha^2}{\partial u_i \partial u_j}. \tag{5.22}$$

We also assume that a_I are the components of the vector $\mathbf{a}$ and that c_1 and c_2 are the components of a two-dimensional vector $\mathbf{c}$. Finally, we complete the square in Eq. (5.18) to obtain

$$\left(\mathbf{c} + \mathcal{Q}^{-1}\mathcal{B}\mathbf{a}\right)^T \mathcal{Q} \left(\mathbf{c} + \mathcal{Q}^{-1}\mathcal{B}\mathbf{a}\right) = \tilde{\Delta} - \mathbf{a}^T \mathcal{B}^T \mathcal{Q}^{-1} \mathcal{B}\mathbf{a}. \tag{5.23}$$

Note that $\mathcal{Q}^{-1}$ exists because all of the eigenvalues of $\mathcal{Q}$ are nonzero by assumption.

Already, Eq. (5.18) is in the form of a conic section whose form depends on the eigenvalues of $\mathcal{Q}$. What remains is to show that $\mathbf{a}$ depends only on the length changes (and not $\mathbf{c}$) to lowest order and, ultimately, to find an expression to determine it.

To do this, we project Eq. (5.16) onto the remaining basis vectors, $e^{(n)}_\alpha$, in the space of square lengths. We obtain

$$\begin{aligned}\sum_m \sum_i \sum_\alpha e^{(n)}_\alpha \frac{\partial \ell^2_\alpha(\mathbf{u}_C)}{\partial u_i} \eta_{m,i} a_m + \frac{1}{2} \sum_{ij\alpha} e^{(n)}_\alpha \frac{\partial^2 \ell^2_\alpha(\mathbf{u}_C)}{\partial u_i \partial u_j} \delta u_i \delta u_j \\ = \sum_\alpha e^{(n)}_\alpha \left(2L^{(c)}_\alpha \delta L_\alpha + \delta L^2_\alpha\right)\end{aligned} \tag{5.24}$$

There are $E-1$ equations in Eq. (5.24) and $dV - E + 1$ zero modes at the critical point, the space spanned by $\delta u_i^\perp$ is $dV - (dV - E + 1) = E - 1$ dimensional. The matrix appearing in Eq. (5.24) is, consequently, square. Since we have already removed zero modes and self stresses, it is also invertible. We define a new matrix $\mathbf{M}$ such that its inverse $\mathbf{M}^{-1}$ is given by the components,

$$\mathbf{M}^{-1}_{nm} = \sum_i \sum_\alpha e^{(n)}_\alpha \frac{\partial \ell^2_\alpha(\mathbf{u}_C)}{\partial u_i} \eta_{m,i}. \tag{5.25}$$

This then allows us to solve Eq. (5.24) in powers of both δL_α and $\mathbf{c}$. To first order in both, we obtain

$$a_n \approx \sum_m M_{nm} \sum_\alpha 2 e^{(m)}_\alpha L^{(c)}_\alpha \delta L_\alpha + \mathcal{O}(c\delta L, c^2, \delta L^2). \tag{5.26}$$

We can now put together the results by defining

$$\delta \mathbf{c} = -\mathcal{Q}^{-1} \mathcal{B} \mathbf{a} \tag{5.27}$$

and

$$\Delta = \tilde{\Delta} - \delta \mathbf{c} \mathcal{Q} \delta \mathbf{c} \tag{5.28}$$

to obtain

$$\left(\delta\mathbf{c}-\delta\mathbf{c}\right)^T \mathcal{Q}\left(\delta\mathbf{c}-\delta\mathbf{c}\right)=\Delta \tag{5.29}$$

where Δ and $\delta\mathbf{c}$ depend linearly on the changes in lengths to lowest order. Therefore, small perturbations of the length are seen to produce trajectories that lie on a 2D conic section with a perturbed center.

While this is a rather intricate derivation, we could have obtained the correct answer up to order $\delta\mathbf{u}\sim\delta L^{1/2}$ more simply by assuming $\mathcal{O}(\mathbf{a})\sim\mathcal{O}(\mathbf{c})$. We have found the full form of Eq. (5.29) to be more useful in perturbing larger linkages, however, as it better captures the case that changes in the bar lengths perturb but do not completely eliminate critical points in the configuration space of a linkage.

5.7 Appendix B: Properties of the tangent form

The tangent form is defined as

$$t^{i_1\cdots i_D}(\mathbf{u})=\sum_{j_1\cdots j_N}\epsilon^{i_1\cdots i_D j_1\cdots j_N}\frac{\partial f_1(\mathbf{u})}{\partial u_{j_1}}\cdots\frac{\partial f_N(\mathbf{u})}{\partial u_{j_N}}, \tag{5.30}$$

where $\epsilon^{i_1\cdots i_D j_1\cdots j_N}$ is the antisymmetric Levi-Civita tensor. Next we compute some simple properties of the tangent form.

The tangent form is divergence free. This can be seen from the following calculation,

$$\begin{aligned}\frac{\partial t^{i_1\cdots i_D}(\mathbf{u})}{\partial u_{i_1}} &= \sum_{j_1\cdots j_N}\epsilon^{i_1\cdots i_D j_1\cdots j_N}\frac{\partial^2 f_1(\mathbf{u})}{\partial u_{i_1}\partial u_{j_1}}\cdots\frac{\partial f_N(\mathbf{u})}{\partial u_{j_N}} \\ \cdots &+ \sum_{j_1\cdots j_N}\epsilon^{i_1\cdots i_D j_1\cdots j_N}\frac{\partial f_1(\mathbf{u})}{\partial u_{j_1}}\cdots\frac{\partial}{\partial u_{i_1}}\frac{\partial f_N(\mathbf{u})}{\partial u_{j_N}} \\ &= 0\end{aligned} \tag{5.31}$$

where each term is zero due to the antisymmetry of the Levi-Civita tensor and the symmetry of partial derivatives.

For one degree of freedom mechanisms, the tangent form is a vector tangent to the configuration space away from critical points. First, we note that

$$\sum_{i_1} \frac{\partial f_\alpha}{\partial u_{i_1}} t^{i_1 \cdots i_D}(\mathbf{u}(s)) = 0 \tag{5.32}$$

which implies that $\frac{\partial f_\alpha}{\partial u_i} t^i(\mathbf{u}(s)) = 0$. Now suppose that $\mathbf{u}(s)$ traces the configuration space in a region where $t^{i_1 \cdots i_D}(\mathbf{u}(s))$ is nonzero. Then

$$\sum_i \frac{\partial f_\alpha(\mathbf{u}(s))}{\partial u_i} \frac{\partial u_i(s)}{\partial s} = 0. \tag{5.33}$$

Hence the configuration space is perpendicular to all of the $\partial f_\alpha(\mathbf{u})/\partial u_i$ but $t_i(\mathbf{u})$ is also perpendicular to all of them. Hence, they must be parallel. The more general case for mechanisms with more than one degree of freedom is more subtle but can also be computed.

The tangent form is zero at u if and only if u is a critical point. Ultimately, this is a consequence of the fact that the components of $t^{i_1 \cdots i_D}(\mathbf{u})$ are the $E \times E$ minors of the Jacobian of $\ell^2(\mathbf{u})$. Nevertheless, we demonstrate it here for completeness. There are E functions

$$\left\{ \frac{\partial f_1(\mathbf{u})}{\partial u_i}, \cdots, \frac{\partial f_E(\mathbf{u})}{\partial u_i} \right\}. \tag{5.34}$$

Since the zero modes are defined by the nonzero solutions, δu_i of

$$\sum_i \frac{\partial f_\alpha(\mathbf{u})}{\partial u_i} \delta u_i = 0 \tag{5.35}$$

the zero modes are in the orthogonal complement of the span of the vectors $\partial f_\alpha(\mathbf{u})/\partial u_i$. At a critical point, there must be additional zero modes and so the $\partial f_\alpha(\mathbf{u})/\partial u_i$ span a lower dimensional space and can no longer be linearly independent. Without loss of generality, we can take it to be $\alpha = 1$ so

$$\frac{\partial f_1(\mathbf{u})}{\partial u_i} = \sum_{\beta>1} c_\alpha \frac{\partial f_\beta(\mathbf{u})}{\partial u_i}. \tag{5.36}$$

Substituting this into the definition of $t_{i_1\cdots i_D}(\mathbf{u})$ and using Eq. (5.32), we immediately obtain $t_{i_1\cdots i_D}(\mathbf{u}) = 0$.

Similarly, if $t_{i_1\cdots i_D}(\mathbf{u}) = 0$ then the $\partial f_\alpha(\mathbf{u})/\partial u_i$ cannot all be linearly independent. One way to do see this is to choose D vectors $\mathbf{v}_n$ orthogonal to the $\partial f_\alpha(\mathbf{u})/\partial u_i$ for all α as well as to each other. Then

$$v_{1,i_1} \cdots v_{D,i_D} t_{i_1\cdots i_D}(\mathbf{u}) = \det \begin{pmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_D^T \\ \nabla f_1(\mathbf{u})^T \\ \vdots \\ \nabla f_E(\mathbf{u})^T \end{pmatrix} = 0, \tag{5.37}$$

where ∇ is the gradient in $\mathbf{u}$ and ^{T} denotes the transpose. Since the $\mathbf{v}_n$ are orthogonal to the other vectors one of the $\nabla f_\alpha(\mathbf{u})$ must be linearly dependent on the rest of them. We immediately obtain that there is at least one additional linear independent zero mode.

Self stresses are orthogonal to the critical value set. The critical set is defined as the set of points $\mathbf{u}_C$ such that $t_{i_1\cdots i_D}(\mathbf{u}_C) = 0$. The critical value set is the image of the critical set under the map $f_\alpha(\mathbf{u}_C)$. Suppose that $\mathbf{u}_C(s)$ is a one-parameter path of points in a smooth portion of the critical set. Then consider its image $F_\alpha(s)$,

$$f_\alpha(\mathbf{u}_C(s)) = F_\alpha(s). \tag{5.38}$$

If the derivative $\partial F_\alpha(s)/\partial s$ is nonzero then it is tangent to the critical value set. Therefore,

$$\frac{\partial F_\alpha(s)}{\partial s} = \sum_i \frac{\partial f_\alpha(\mathbf{u}_C(s))}{\partial u_i} \frac{\partial u_{C,i}(s)}{\partial s}. \tag{5.39}$$

If σ_α is a self stress then $\sum_\alpha \sigma_\alpha \partial f_\alpha / \partial u_I = 0$. Therefore we obtain

$$\sum_\alpha \sigma_\alpha \frac{\partial F_\alpha(s)}{\partial s} = 0. \tag{5.40}$$

Since this is true for any path in the critical value set, it follows that all self stresses are orthogonal to the critical value set.

Though the converse of this is not true – some vectors normal to the critical value set may not be self stresses – if the critical value set has codimension one then there can be only one self stress and the normal vector of the critical value set necessarily corresponds to that self stress.

Orientation The tangent form $t_{i_1 \cdots i_D}(\mathbf{u})$ carries additional useful geometrical information about the mechanism at regular (non-critical) configurations. When $D = 0$, $t(\mathbf{u})$ is a scalar whose sign was used to compute a topological index in periodic mechanisms [68]. Beyond this, it endows the configuration space with a natural orientation in any dimension. At a regular point on the configuration space of a mechanism, $\mathbf{x}$, $t_{i_1 \cdots i_D}(\mathbf{x}) dx^{i_1} \wedge \cdots \wedge dx^{i_D}$ is a differential form which provides a local orientation: for any basis of tangent vectors

$\{\zeta_{1,j}, \cdots, \zeta_{D,j}\}$,

$$\operatorname{sgn} \sum_{i_1 \cdots i_D} \zeta_{1,i_1} \cdots \zeta_{D,i_D} t_{i_1 \cdots i_D}(\mathbf{u}) = \pm 1 \tag{5.41}$$

However, note that this local orientation is only defined up to an overall sign, since we can always take one of the constraint functions to have the opposite sign.

Though we do not make a great deal of use of it in this paper, it is worth noting that if one is able to find two regions in which $t^{i_1 \cdots i_D}(\mathbf{u})$ has opposite signs, there must be a boundary between those regions for which $t^{i_1 \cdots i_D}(\mathbf{u})$ vanishes. That is, in principle we can use the tangent form to verify the existence of critical configurations.

Chapter 6

Conclusion

In this dissertation, we investigated how to design mechanical devices that compute. After discussing multiple ways to define a system that computes, we focused in on designing mechanical versions of wires and transistors. Using these components, we could in the future combine them to create logic gates, which can then be combined to create devices that compute through boolean logic operations. Designing mechanical wires and transistors with robust, predictable behavior is a first step towards eventually making mechanical devices that compute.

In Chapter 3, we investigated one design for a mechanical wire. This system consisted of a 1D chain of connected bistable elements made out of a soft material. The signal being sent along this wire was a transition wave where each element snaps from one minima in its bistable potential to the other. Previous work on these systems focused on wires that supported stable propagation over arbitrarily long distances, which was possible because each bistable element transitioned from a high to a low energy state as the signal propagated. Our work investigated the behavior of wires that do not support stable propagation – each element either transitions between two states of equal energy or from a low to a high energy state. We found that in these wires, signals can propagate a finite distance controlled by a penetration depth that depends on the wire parameters.

In Chapters 4 and 5, we focused on how to design mechanical transistors. After analyzing the general behavior of a linkage called the flip-flop, we mapped its behavior to that of a transistor and identified how to integrate it into the wire design discussed in Chapter 3. In simulations, we were able to use these designs for wires and transistors to construct functional AND and NOR gates. However, the transistor-like behavior of the flip-flop disappeared when it was constructed out of the same soft material used to make our wires in Chapter 3.

The flip-flop got its transistor-like behavior from critical points in its configuration space. This observation lead us to switch from the question of how do we design mechanical transistors to a more general question: how can we design the configuration space of a linkage? If we can control all aspects of a linkage's configuration space, including the presence and location of critical points, we can design other linkages that also behave like transistors. Chapter 5 introduces a method for programming the topology of a configuration space through small perturbations of the lengths of each bar in the linkage. Using this method, we designed a linkage that gates the propagation of a soliton in a Kane-Lubensky chain, where the Kane-Lubensky chain is a mechanical wire and the gate linkage is a transistor.

Our work presented two candidates each for a mechanical wire and mechanical transistor. To construct a device that computes, we need to combine these wires and transistor into logic gates. Using the soft wires from Chapter 3 along with the flip-flop presented in Chapter 4, we were able to create designs for logic gates, but these devices did not behave as predicted in experiments. The gated Kane-Lubensky chain presented in Chapter 5 behaves as expected in both simulations and experiments, but more work is needed to design logic gates using that version of a wire and transistor. In order to design fully-functional devices that can perform basic logic operations, we need to investigate more candidates for wires and transistors and identify designs that 1. have the same behavior in both simulations and experiments, and 2. can be combined to form logic gates. Because we have the ability to control the topology of the configuration space of a linkage, the ideal system to construct these new wires and transistors with would be linkages of rigid bars.

A final idea that was not covered in-depth was using the finite state machine definition of computation to create our mechanical devices that compute. As discussed in Chapter 1, a finite state machine can be represented by a directed graph called a state diagram. In the state diagram, the nodes correspond to the states of the system and the edges correspond to the transitions between states. It turns out that we can also represent the configuration space of a linkage as a directed graph, where the critical points are the nodes or vertices and the branches are the edges. This representation of a configuration space has multiple benefits. First, it allows us to visualize the configuration space of a 1 degree of freedom linkage that lives in more than 3 dimensions, a problem that will occur when investigating more complicated linkages. Second, it allows us to view the linkage as a finite state machine directly. With this graph representation of a linkage, we can then investigate the connections between computation ability and graph structure. After identifying a specific computation we want our finite state machine to perform and the corresponding graph, we can use our method for programming the configuration space of linkages to design a linkage with that target behavior.

Appendices

Appendix A

Documentation of Chapter 5 Code

Much of the visualization shown in Chapter 5 was done by solving for the relevant equations analytically using Mathematica's built-in capabilities and then plotting them using built-in functions as well. The surface shown in Fig. 5.3, the plot shown in Fig 5.4b, and the region plots in Fig. 5.5a and Fig. 5.6c were all generated in this way.

Plotting configuration spaces that live in 3D (such as the one shown in Fig. 5.5a) proved to be a non-trivial task. While we had many functions from the *mechanisms* package that made working with linkages easier, we were unable to find a way to solve for these 3D configuration spaces analytically and use Mathematica's built-in plotting functions to visualize them. In this appendix we will detail how we solved this problem numerically by tracing out the configuration space using the tangent field from Eq. 5.10.

To differentiate functions from Mathematica, functions from the *mechanisms* package, and functions we wrote specifically for Chapter 5, we will use the following notation:

- Functions in plain text are built-in Mathematica functions.
- Functions in *italics* are from the *mechanisms* package.
- Functions in **bold** are ones written specifically for the calculations and figures in Chap-

ter 5.

A.1 Creating and Working With Linkages

The *mechanisms* package gives us a wide variety of functions that let us define, manipulate, and analyze linkages made of rigid bars, springs, and a variety of other components. There are a handful of functions from that package that we will briefly describe here because of their use in our analysis of the configuration space of linkages.

A.1.1 Defining a Linkage

We can define a linkage using *Linkage*[] by specifying the location of vertices in xy-coordinates, any bars/springs in the linkage and which vertices they're connected to, and if any of the vertices are pinned in place. This creates a mechanism type object that we can manipulate using the functions in the package.

If specifying the exact vertex locations is difficult due to a non-symmetric configurations because of multiple different bar lengths, we can define a linkage that has the same connectivity using "simple" vertex locations, then use the *ChangeCellData*[] function to re-define the bars to have the correct length. Both methods can produce identical objects.

A.1.2 Motions of a Linkage

As discussed in Chapter 1, we know how to identify the possible motions of a linkage from a given position analytically using the rigidity matrix. The function *MechanismInfinitesimalMotions*[Mechanism(,positions)] does this for us (and it does it analytically using both package functions and built-in Mathematica functions). We give the function a mechanism with specific vertex positions, and it returns a list of two elements: an infinitesimal linear motion and, if necessary, a list of quadratic constraints that motion must satisfy. If the mechanism happens to be at a critical point and there is more than one infinitesimal motion, the function returns a two element list for each unique motion.

Now that we can identify the possible motions of a mechanism, we want the ability to visualize the mechanism moving with that motion. A trajectory of a mechanism will be a list of positions of the vertices that trace out how the mechanism moves. This trajectory corresponds to a portion of the configuration space of the linkage.

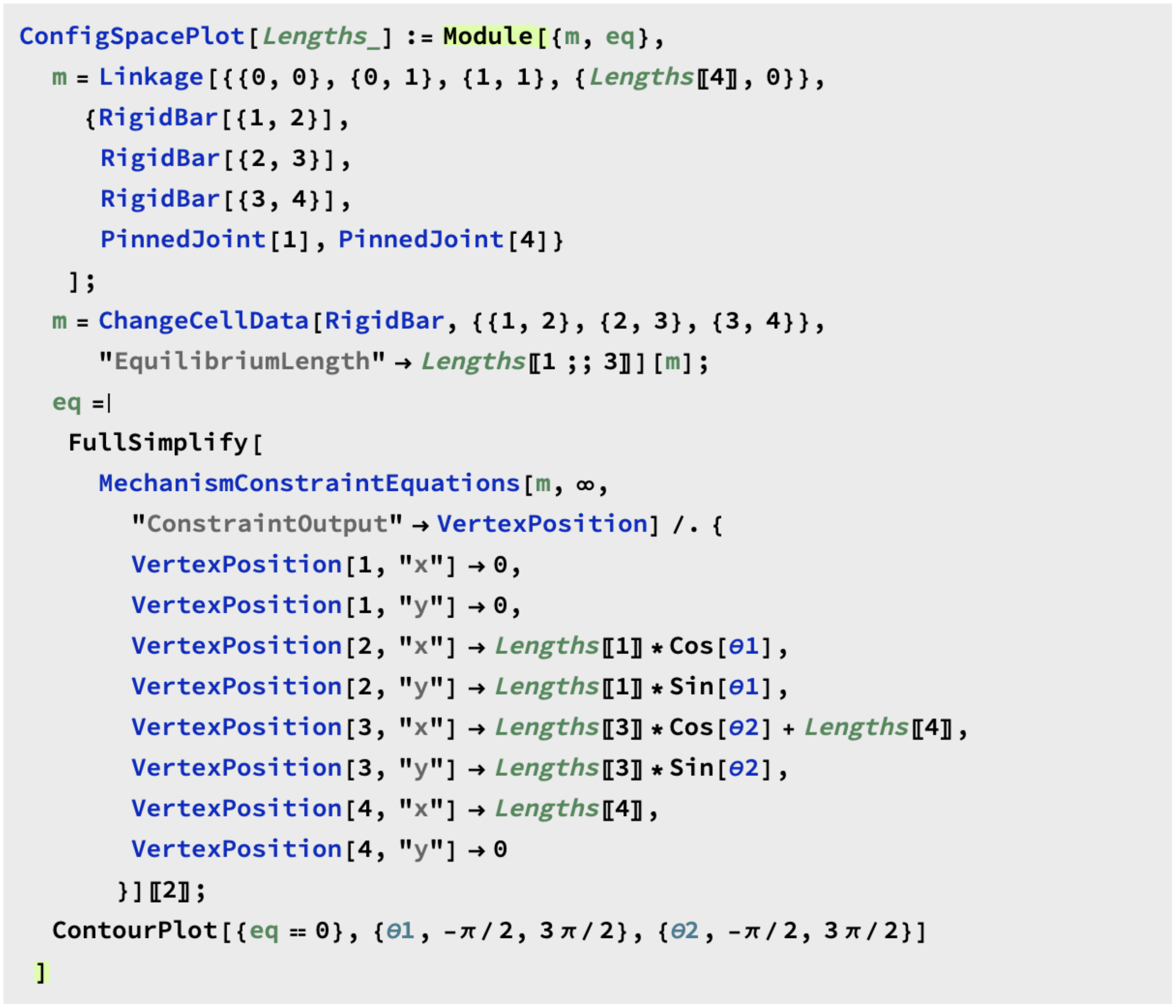
The function *MechanismIsometricTrajectory*[Mechanism, $\{\{\Delta x_1, \Delta y_1, \ldots\}, \ldots\}$, n] creates a trajectory using n steps through the configuration space of a mechanism starting in the displacement direction $\{\{\Delta x_1, \Delta y_1, \ldots\}, \ldots\}$. We give the function a mechanism and an initial direction, and it returns a list of n mechanism positions (x and y coordinates for all vertices in the mechanism) that can, for example, be used to plot a configuration space or create an animation of the mechanisms moving along the trajectory.

The displacement direction can be the output of *MechanismInfinitesimalMotions*[], or it can be a guess at the way a mechanism should move. Because *MechanismIsometricTrajectory* minimizes the energy of the mechanism as it takes each step, regardless of what direction we try to move our mechanism in, it will step to a position on the closest branch of the configuration space. Therefore, using a best guess for the direction of a branch can work for preliminary examples.

A.2 Visualization of 2D Configuration Spaces

To plot the configuration space of a 1 degree-of-freedom linkage with a configuration space that lives in 2D, we can utilize built-in Mathematica functions and directly plot the length constraint equations defined in Chapter 1.

To obtain these equations, we can use the *MechanismConstraintEquations[]* function. This set of equations uses the x and y positions of each vertex as variables, and it can be



```
ConfigSpacePlot[Lengths_] := Module[{m, eq},
  m = Linkage[{{0, 0}, {0, 1}, {1, 1}, {Lengths⟦4⟧, 0}},
    {RigidBar[{1, 2}],
     RigidBar[{2, 3}],
     RigidBar[{3, 4}],
     PinnedJoint[1], PinnedJoint[4]}
   ];
  m = ChangeCellData[RigidBar, {{1, 2}, {2, 3}, {3, 4}},
     "EquilibriumLength" → Lengths⟦1 ;; 3⟧][m];
  eq =
   FullSimplify[
     MechanismConstraintEquations[m, ∞,
       "ConstraintOutput" → VertexPosition] /. {
       VertexPosition[1, "x"] → 0,
       VertexPosition[1, "y"] → 0,
       VertexPosition[2, "x"] → Lengths⟦1⟧ * Cos[θ1],
       VertexPosition[2, "y"] → Lengths⟦1⟧ * Sin[θ1],
       VertexPosition[3, "x"] → Lengths⟦3⟧ * Cos[θ2] + Lengths⟦4⟧,
       VertexPosition[3, "y"] → Lengths⟦3⟧ * Sin[θ2],
       VertexPosition[4, "x"] → Lengths⟦4⟧,
       VertexPosition[4, "y"] → 0
      }]⟦2⟧;
  ContourPlot[{eq == 0}, {θ1, -π / 2, 3 π / 2}, {θ2, -π / 2, 3 π / 2}]
 ]
```

Figure A.1: Mathematica code for the ConfigSpacePlot2D[] function. This function is written specifically for a four bar linkage.

simplified in three ways:

1. Directly plug in the locations of all pinned vertices.

2. For any vertex connected directly to a pinned vertex by a rigid bar, convert its position to polar coordinates using $x = r\cos\theta$, $y = r\sin\theta$ where r is the length of the rigid bar.

3. Use the Mathematica function FullSimplify[] to simplify the entire system.

If you have substituted in all known constraints, all expressions except for one should simply to 0. The remaining equation should be only a function of your θ values.

Plotting this 2D function directly with Plot3D[] will produce a 2D surface in 3D. The configuration space of the linkage is where our expression equals 0, we can plot that equation using ContourPlot3D[] to visualize the configuration space.

A.3 Visualization of 3D Configuration Spaces

We were unable to find a way to plot the configuration space for a 1 degree-of-freedom linkage that lives in 3D space using Mathematica's built-in functions. If we try to plot the configuration space using the same method detailed above, the simplified length constraint equation has three unique variables, which makes it incompatible with the various 3D plot functions in Mathematica.

There are a couple things we can calculate related to a linkage and its configuration space.

- We can identify the vertex positions for all critical points of a linkage with specified lengths and 1 degree-of-freedom using **criticalPoints**[]
- Given a specific configuration of a linkage, we can identify the infinitesimal motions for the linkage (the ways a linkage can move from that position) using *MechanismInfinitesimalMotions*[]
- We can "follow" the motion of the linkage along its configuration space without knowing ahead of time what that motion will look like using *MechanismIsometricTrajectory*[].
- We can calculate the magnitude of the tangent field from Eq. 5.10 for a specific configuration of a linkage.

- We know that along a section of the configuration space between two critical points, the magnitude of the tangent field starts at zero at the critical point, increases as you follow the branch away from the critical point, then decreases back to zero as you reach the other critical point.

We were able to combine all of these smaller pieces to plot a 1 degree-of-freedom configuration space with arrows indicating the direction of the tangent field in both 2D and 3D. This method was used to generate the configuration space plots in Fig. 5.2b, Fig. 5.4c, and Fig. 5.6d.

Step 1: Identify the critical points

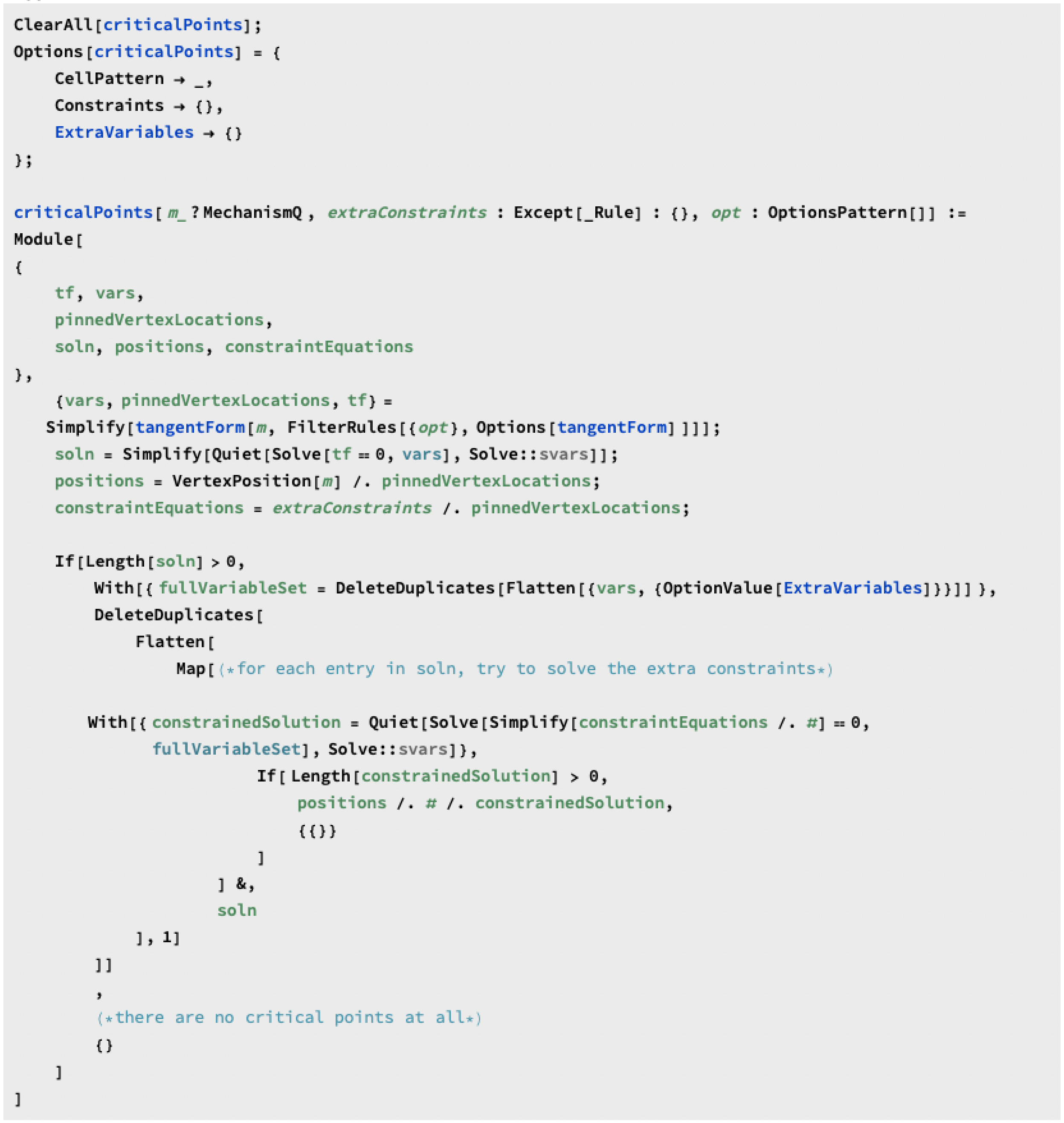
To identify the critical/branch points of a mechanism, we wrote the **criticalPoints**[] function. This function solves $t_i(\mathbf{u}) = 0$ subject to the length constraints $\ell_\alpha^2(\mathbf{u}) = L_\alpha^2$. A critical point is stored as a set of positions for each vertex in the linkage. The function returns a list of possible critical points.

The critical points returned by **criticalPoints**[] will be a function of some subset of vertex position variables, not an exact location for each vertex. To get exact locations, we need to add in the exact lengths for each bar.

The final list of critical points has an exact location for each vertex, which allows us to visualize the mechanism at the critical point and plot the exact location of the critical point in the 2D or 3D space that the configuration space lives in.

Step 2: Identify the segments that connect critical points

The configuration space is made of segments/branches that start and end at critical points. These segments represent the infinitesimal motions that the linkage has. At a given critical point, there are some number of segments branching off from that point. We use the

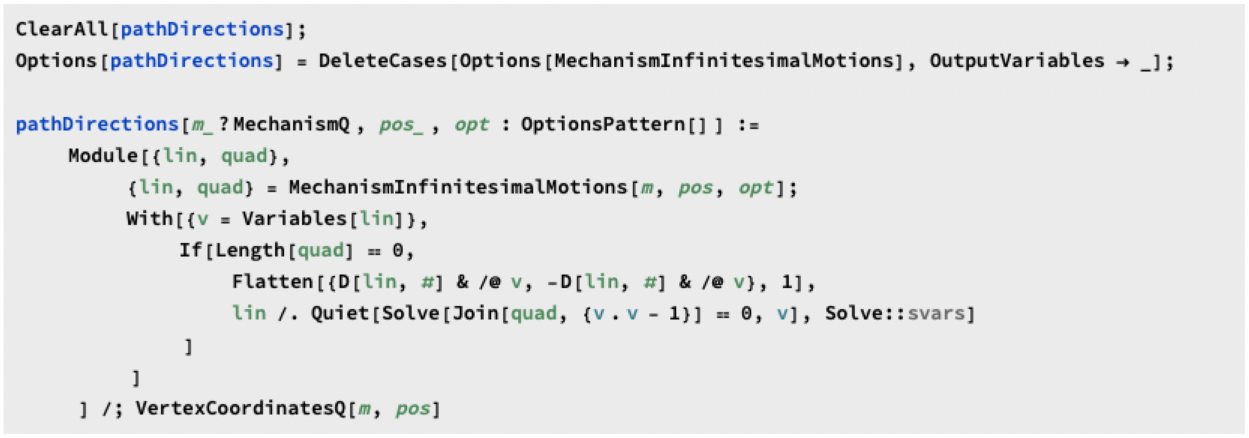


```
ClearAll[criticalPoints];
Options[criticalPoints] = {
    CellPattern → _,
    Constraints → {},
    ExtraVariables → {}
};

criticalPoints[ m_ ? MechanismQ , extraConstraints : Except[_Rule] : {}, opt : OptionsPattern[]] :=
Module[
{
    tf, vars,
    pinnedVertexLocations,
    soln, positions, constraintEquations
},
    {vars, pinnedVertexLocations, tf} =
   Simplify[tangentForm[m, FilterRules[{opt}, Options[tangentForm] ]]];
    soln = Simplify[Quiet[Solve[tf == 0, vars], Solve::svars]];
    positions = VertexPosition[m] /. pinnedVertexLocations;
    constraintEquations = extraConstraints /. pinnedVertexLocations;

    If[Length[soln] > 0,
        With[{ fullVariableSet = DeleteDuplicates[Flatten[{vars, {OptionValue[ExtraVariables]}}]] },
        DeleteDuplicates[
            Flatten[
                Map[(*for each entry in soln, try to solve the extra constraints*)

       With[{ constrainedSolution = Quiet[Solve[Simplify[constraintEquations /. #] == 0,
              fullVariableSet], Solve::svars]},
                         If[ Length[constrainedSolution] > 0,
                             positions /. # /. constrainedSolution,
                             {{}}
                         ]
                     ] &,
                     soln
            ], 1]
        ]]
        ,
        (*there are no critical points at all*)
        {}
    ]
]
```

Figure A.2: Mathematica code for the criticalPoints[] function.

pathDirections[] function to identify the direction that each segment points in by solving Eq. (5.7) to obtain the tangents to the configuration space.

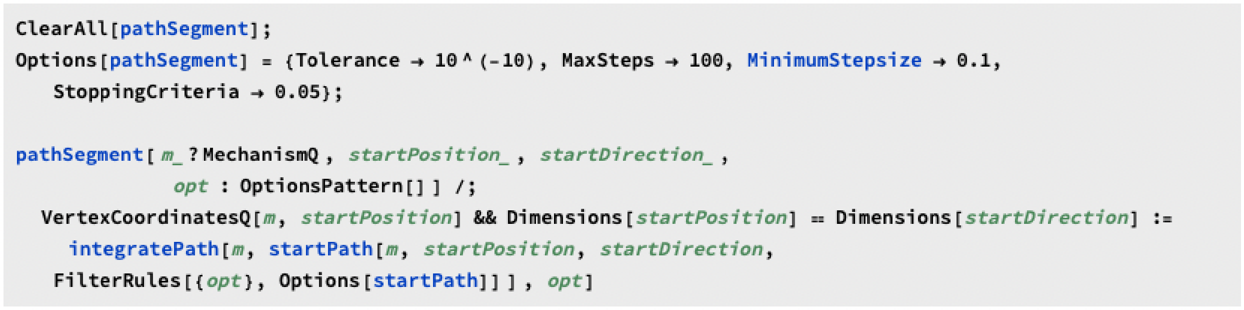
The directions returned by **pathDirections**[] are a list of initial velocities for each vertex



```
ClearAll[pathDirections];
Options[pathDirections] = DeleteCases[Options[MechanismInfinitesimalMotions], OutputVariables → _];

pathDirections[m_?MechanismQ , pos_ , opt : OptionsPattern[] ] :=
    Module[{lin, quad},
        {lin, quad} = MechanismInfinitesimalMotions[m, pos, opt];
        With[{v = Variables[lin]},
            If[Length[quad] == 0,
                Flatten[{D[lin, #] & /@ v, -D[lin, #] & /@ v}, 1],
                lin /. Quiet[Solve[Join[quad, {v . v - 1}] == 0, v], Solve::svars]
            ]
        ]
    ] /; VertexCoordinatesQ[m, pos]
```

Figure A.3: Mathematica code for the pathDirections[] function.

in the mechanism. We get one unique set of velocities per infinitesimal motion. For all the critical points of the mechanism, we get a new set of directions.

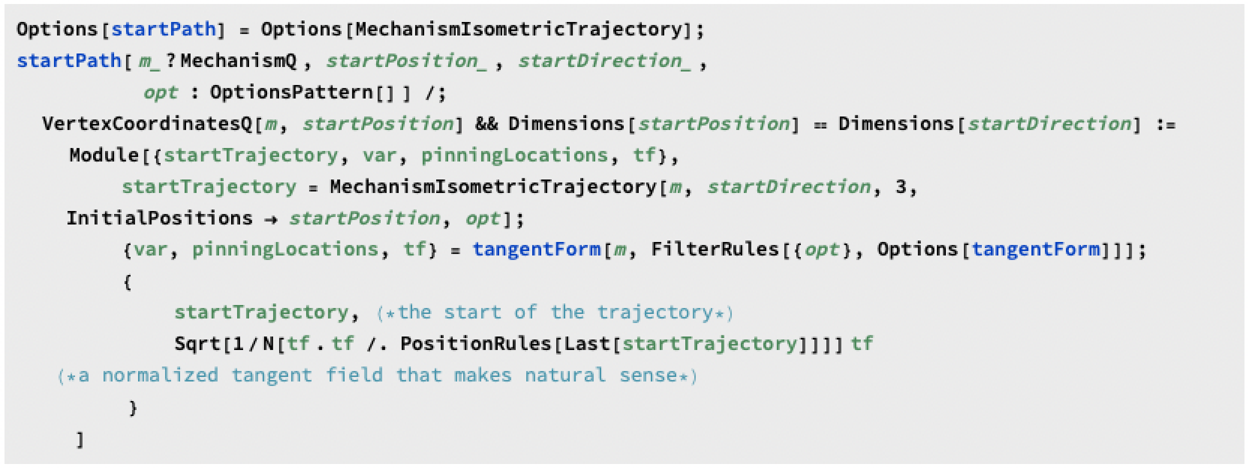


```
ClearAll[pathSegment];
Options[pathSegment] = {Tolerance → 10^(-10), MaxSteps → 100, MinimumStepsize → 0.1,
   StoppingCriteria → 0.05};

pathSegment[ m_?MechanismQ , startPosition_ , startDirection_ ,
            opt : OptionsPattern[] ] /;
  VertexCoordinatesQ[m, startPosition] && Dimensions[startPosition] == Dimensions[startDirection] :=
    integratePath[m, startPath[m, startPosition, startDirection,
   FilterRules[{opt}, Options[startPath]] ] , opt]
```

Figure A.4: Mathematica code for the pathSegment[] function.

Next, we need to follow each of these directions and trace out the segment they point along. To do this, we use the **pathSegment**[] function which utilizes the *MechanismIsometricTrajectory*[] function. We give **pathSegment**[] a mechanism, a critical point to start at, and a list of directions to travel away from that critical point. First, we step along one of the tangent vectors from our list of directions using **startPath**[].

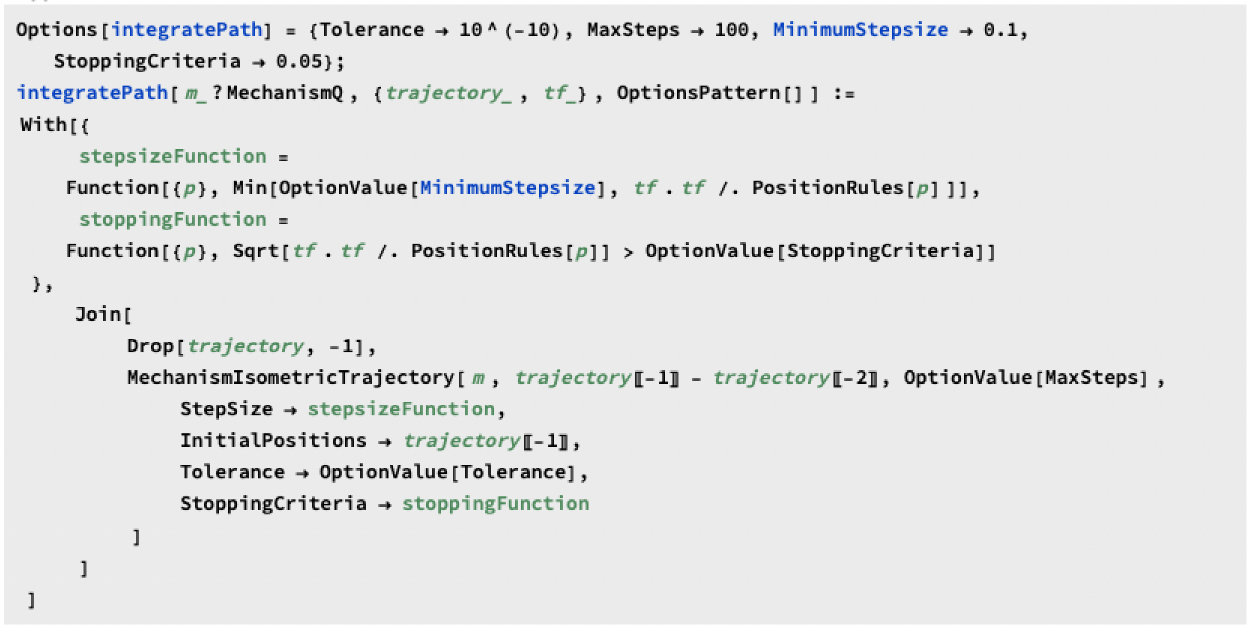
The function **integratePath**[] then uses *MechanismIsometricTrajectory*[] to step along the configuration space in the direction indicated by $t_i(\mathbf{u})$ with a step size proportional to its magnitude. The step size is adjusted to maintain the edge lengths to less than one percent



```
Options[startPath] = Options[MechanismIsometricTrajectory];
startPath[ m_ ? MechanismQ , startPosition_ , startDirection_ ,
          opt : OptionsPattern[] ] /;
  VertexCoordinatesQ[m, startPosition] && Dimensions[startPosition] == Dimensions[startDirection] :=
    Module[{startTrajectory, var, pinningLocations, tf},
        startTrajectory = MechanismIsometricTrajectory[m, startDirection, 3,
    InitialPositions → startPosition, opt];
        {var, pinningLocations, tf} = tangentForm[m, FilterRules[{opt}, Options[tangentForm]]];
        {
            startTrajectory, (*the start of the trajectory*)
            Sqrt[1 / N[tf . tf /. PositionRules[Last[startTrajectory]]]] tf
   (*a normalized tangent field that makes natural sense*)
        }
    ]
```

Figure A.5: Mathematica code for the startPath[] function.

strain. Finally, the integration for each segment is terminated by a stopping function that checks if the magnitude of $t_i(\mathbf{u})$ is below a critical threshold, indicating that the integration has reached the neighborhood of the next critical point.



```
Options[integratePath] = {Tolerance → 10 ^ (-10), MaxSteps → 100, MinimumStepsize → 0.1,
   StoppingCriteria → 0.05};
integratePath[ m_ ? MechanismQ , {trajectory_ , tf_} , OptionsPattern[] ] :=
With[{
      stepsizeFunction =
    Function[{p}, Min[OptionValue[MinimumStepsize], tf . tf /. PositionRules[p] ]],
      stoppingFunction =
    Function[{p}, Sqrt[tf . tf /. PositionRules[p]] > OptionValue[StoppingCriteria]]
 },
     Join[
         Drop[trajectory, -1],
         MechanismIsometricTrajectory[ m , trajectory⟦-1⟧ - trajectory⟦-2⟧, OptionValue[MaxSteps] ,
              StepSize → stepsizeFunction,
              InitialPositions → trajectory⟦-1⟧,
              Tolerance → OptionValue[Tolerance],
              StoppingCriteria → stoppingFunction
         ]
     ]
 ]
```

Figure A.6: Mathematica code for the integratePath[] function.

The final thing we can do is add a "direction" to the segments using the **pathOrientation[]** function. Depending on if we traveled along the path with or against the direction of

$t_i(\mathbf{u})$, this function will output a positive or negative value that we can use when plotting the final configuration space.

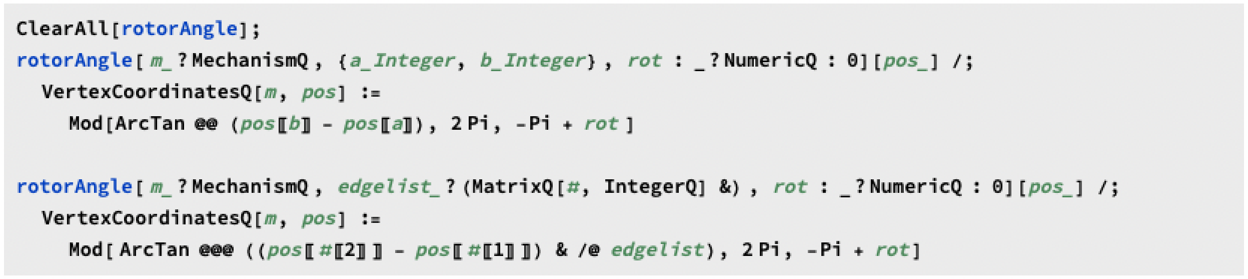
```
segmentOrientation[ vars_ , tf_ , {path1_, path2_} ] /;
  VertexCoordinatesQ[path1] && VertexCoordinatesQ[path2] :=
    Sign[ (vars /. PositionRules[path2 - path1]) . (tf /. Dispatch[PositionRules[path1]])]

ClearAll[pathOrientation];

Options[pathOrientation] = Options[tangentForm];

pathOrientation[ m_?MechanismQ , path_ , method_String : "Fast", opt : OptionsPattern[] ] :=
 With[{tfout = tangentForm[m, opt]}, With[{vars = tfout⟦1⟧ /. tfout⟦2⟧, tf = tfout⟦3⟧ /. tfout⟦2⟧},
      Switch[ method,
           "Slow",
      Mean[segmentOrientation[vars, tf, #] & /@ Drop[Rest[Partition[path, 2, 1]], -1]],
           "Fast", With[{midpoint = Floor[Length[path] / 2]},
       segmentOrientation[vars, tf, path⟦{midpoint, midpoint + 1}⟧]],
           _, $Failed
       ]
  ]] /; VertexCoordinateListQ[m, path] && (Length[path] ≥ 3)
(*
(*give it a list of paths and it orients all of them*)
pathOrientation[ m_?MechanismQ , pathList : {__?VertexCoordinateListQ} , method_String : "Fast",
  opt : OptionsPattern[] ] :=
With[{tfout=tangentForm[m,opt]},With[{vars = tfout⟦1⟧ /. tfout⟦2⟧,tf= tfout⟦3⟧ /. tfout⟦2⟧},
    Switch[ method,
        "Slow", Mean[segmentOrientation[vars,tf,#]& /@ Drop[Rest[Partition[#,2,1]],-1]]&/@pathList,
        "Fast",
    With[{midpoint = Floor[Length[#]/2]},segmentOrientation[vars,tf, #⟦{midpoint,midpoint+1}⟧]]& /@
     pathList,
        _, $Failed
    ]
]]
*)
```

Figure A.7: Mathematica code for the pathOrientation[] function.

This process of picking a critical point, finding all possible directions, tracing out the segment for each direction, and identifying the start and end critical points is repeated for all critical points in our list. This initially will give us a list with twice as many segments as we need. We use **pathOrientation**[] to filter and select only the paths that we integrated in the same direction as the tangent vector. After all of these steps, we have a list of critical points and all segments that connect them. Using these, we can construct the full configuration space.

Step 3: Plot all critical points and segments

At this point, all of our segments and critical points are stored as specific positions for each vertex in the linkage. For the linkages we looked at in Chapter 5, our configuration space lives in the space of rotor angles. We use the **rotorAngle**[] function to convert our list of x and y vertex positions (for both the critical points and the segments) to a set of two or three angles. The final configuration space will be plotted in this angle space.



```
ClearAll[rotorAngle];
rotorAngle[ m_ ? MechanismQ , {a_Integer, b_Integer} , rot : _ ? NumericQ : 0][pos_] /;
  VertexCoordinatesQ[m, pos] :=
    Mod[ArcTan @@ (pos⟦b⟧ - pos⟦a⟧), 2 Pi, -Pi + rot ]

rotorAngle[ m_ ? MechanismQ , edgelist_ ? (MatrixQ[#, IntegerQ] &) , rot : _ ? NumericQ : 0][pos_] /;
  VertexCoordinatesQ[m, pos] :=
    Mod[ ArcTan @@@ ((pos⟦ #⟦2⟧ ⟧ - pos⟦ #⟦1⟧ ⟧) & /@ edgelist), 2 Pi, -Pi + rot]
```

Figure A.8: Mathematica code for the rotorAngle[] function.

We use the **plotConfigurationSpace**[] and **plotConfigurationSpace3D**[] functions to plot each segment in 2D or 3D. Both functions take in the list of critical points and the list of segments after they have been transformed with **rotorAngle**[]. The critical points are plotted with ListPlot[] or ListPointPlot3d[] and labeled with numbers based on their order in the list.

The segments could be plotted with ListPlot[] or ListPointPlot3d[] as well, but to add the arrows to our plots, we needed to write the **orientedCurve**[] function. We use the points along the segment to draw very small line segments that will, eventually, form a smooth curve tracing out the segment. We then take a user defined number of these small segments and use the Arrow graphics object instead of a Line object. The result is a smooth curve

```
Options[plotConfigurationSpace] = Join[Options[ListPlot], {CurveStyle → {}, ArrowNumber → 5}];

plotConfigurationSpace[branchpoints_, paths_, opt : OptionsPattern[]] :=
    Show[
        ListPlot[branchpoints → Range[Length[branchpoints]],
   Flatten[{FilterRules[{opt}, Options[ListPlot]], PlotRange → All, PlotStyle → Red}]],
        If[OptionValue[ArrowNumber] > 0,

   Graphics[Flatten[{Arrowheads[0.025], OptionValue[CurveStyle],
     orientedCurve[#, OptionValue[ArrowNumber]] & /@ paths}]],
            Graphics[Flatten[{OptionValue[CurveStyle], curve /@ paths}]]
        ]
    ]
```

Figure A.9: Mathematica code for the plotConfigurationSpace[] function.

```
Options[plotConfigurationSpace] = Join[Options[ListPlot], {CurveStyle → {}, ArrowNumber → 5}];

plotConfigurationSpace[branchpoints_, paths_, opt : OptionsPattern[]] :=
    Show[
        ListPlot[branchpoints → Range[Length[branchpoints]],
   Flatten[{FilterRules[{opt}, Options[ListPlot]], PlotRange → All, PlotStyle → Red}]],
        If[OptionValue[ArrowNumber] > 0,

   Graphics[Flatten[{Arrowheads[0.025], OptionValue[CurveStyle],
     orientedCurve[#, OptionValue[ArrowNumber]] & /@ paths}]],
            Graphics[Flatten[{OptionValue[CurveStyle], curve /@ paths}]]
        ]
    ]
```

Figure A.10: Mathematica code for the plotConfigurationSpace3D[] function.

through 2D or 3D space that has a couple arrowheads along it. These arrowheads point in the direction of the tangent vector for that branch.

```
orientedCurve[pointList_, arrowNumber_ : 5, tol_ : 0.25] := With[
 {
     partitionedLine = Select[Partition[pointList, 2, 1, 1], Norm[#⟦2⟧ - #⟦1⟧] < tol &]
 },
     {Line /@ partitionedLine,
   Arrow /@ Partition[partitionedLine, Floor[Length[partitionedLine] / arrowNumber]]⟦All, -1⟧}
 ]
```

Figure A.11: Mathematica code for the orientedCurve[] function.

Bibliography

[1] Hugo Akitaya, Erik D Demaine, Takashi Horiyama, Thomas C Hull, Jason S Ku, and Tomohiro Tachi. Rigid foldability is np-hard. *arXiv preprint arXiv:1812.01160*, 2018.

[2] Roger C Alperin, Barry Hayes, and Robert J Lang. Folding the hyperbolic crane. *The Mathematical Intelligencer*, 34(2):38–49, 2012.

[3] L. Asimow and B. Roth. The rigidity of graphs. *Transactions of the American Mathematical Society*, pages 279–289, 1978.

[4] L. Asimow and B. Roth. The rigidity of graphs, ii. *Journal of Mathematical Analysis and Applications*, 1979.

[5] Clark R Barker. A complete classification of planar four-bar linkages. *Mechanism and Machine Theory*, 20(6):535–554, 1985.

[6] Nakul Prabhakar Bende, Arthur A Evans, Sarah Innes-Gold, Luis A Marin, Itai Cohen, Ryan C Hayward, and Christian D Santangelo. Geometrically controlled snapping transitions in shells with curved creases. *Proceedings of the National Academy of Sciences*, 112(36):11175–11180, 2015.

[7] M Berry, ME Lee-Trimble, and CD Santangelo. Topological transitions in the configuration space of non-euclidean origami. *Physical Review E*, 101(4):043003, 2020.

[8] Katia Bertoldi, Vincenzo Vitelli, Johan Christensen, and Martin Van Hecke. Flexible mechanical metamaterials. *Nature Reviews Materials*, 2, 2017.

[9] Alexander P. Browning, Francis G. Woodhouse, and Matthew J. Simpson. Reversible signal transmission in an active mechanical metamaterial. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 475(2227):20190146, 2019.

[10] Stuart C Burgess. A review of linkage mechanisms in animal joints and related bio-inspired designs. *Bioinspiration & Biomimetics*, 2021.

[11] Sebastien JP Callens and Amir A Zadpoor. From flat sheets to curved geometries: Origami and kirigami approaches. *Materials Today*, 21(3):241–264, 2018.

[12] Toen Castle, Yigil Cho, Xingting Gong, Euiyeon Jung, Daniel M Sussman, Shu Yang, and Randall D Kamien. Making the cut: Lattice kirigami rules. *Physical review letters*, 113(24):245502, 2014.

[13] George C Chase. History of mechanical computing machinery. *IEEE Annals of the History of Computing*, 2(03):198–226, 1980.

[14] Bryan Gin-ge Chen and Christian D Santangelo. Branches of triangulated origami near the unfolded state. *Physical Review X*, 8(1):011034, 2018.

[15] Bryan Gin-ge Chen and Christian D Santangelo. Branches of triangulated origami near the unfolded state. *Physical Review X*, 8(1):011034, 2018.

[16] Bryan Gin-ge Chen, Nitin Upadhyaya, and Vincenzo Vitelli. Nonlinear conduction via solitons in a topological mechanical insulator. *Proceedings of the National Academy of Sciences*, 111(36):13004–13009, 2014.

[17] Tian Chen, Mark Pauly, and Pedro M. Reis. A reprogrammable mechanical metamaterial with stable memory. *Nature*, 589(7842):386–390, 2021.

[18] Alex Churchill, Stella Biderman, and Austin Herrick. Magic: The gathering is turing complete. `https://arxiv.org/abs/1904.09828`, 2019.

[19] Robert Connelly. The rigidity of certain cabled frameworks and the second-order rigidity of arbitrarily triangulated convex surfaces. *Advances in Mathematics*, 37:272–299, 1980.

[20] Robert Connelly and Herman Servatius. Higher-order rigidity—what is the proper definition? *Discrete & Computational Geometry*, 11(2):193–200, 1994.

[21] Robert Connelly, Jeffrey D Shen, and Alexander D Smith. Ball packings with periodic constraints. *Discrete & Computational Geometry*, 52(4):754–779, 2014.

[22] Robert Connelly and Walter Whiteley. Second-order rigidity and prestress stability for tensegrity frameworks. *SIAM Journal on Discrete Mathematics*, 9(3):453–491, 1996.

[23] Ojan Khatib Damavandi, Varda F Hagh, Christian D Santangelo, and M Lisa Manning. Energetic rigidity. i. a unifying theory of mechanical stability. *Physical Review E*, 105(2):025003, 2022.

[24] Erik D. Demaine and Joseph ORourke. *Geometric Folding Algorithms.* Cambridge University Press, 2007.

[25] B. Deng, J. R. Raney, K. Bertoldi, and V. Tournat. Nonlinear waves in flexible mechanical metamaterials. *Journal of Applied Physics*, 130(4):040901, 2021.

[26] Bolei Deng, Pai Wang, Vincent Tournat, and Katia Bertoldi. Nonlinear transition waves in free-standing bistable chains. *Journal of the Mechanics and Physics of Solids*, 136:103661, 2020.

[27] Bolei Deng, Yuning Zhang, Qi He, Vincent Tournat, Pai Wang, and Katia Bertoldi. Propagation of elastic solitons in chains of pre-deformed beams. *New Journal of Physics*, 21(7):073008, 2019.

[28] Peter S Donelan. Singularity-theoretic methods in robot kinematics. *Robotica*, 25(6):641–659, 2007.

[29] Jakob A. Faber, Andres F. Arrieta, and André R. Studart. Bioinspired spring origami. *Science*, 359(6382):1386–1391, 2018.

[30] Yan Fang, Victor V. Yashin, Steven P. Levitan, and Anna C. Balazs. Pattern recognition with "materials that compute". *Science Advances*, 2(9):1–11, 2016.

[31] Igor Fernández de Bustos, Josu Aguirrebeitia, Rafael Avilés, and Rubén Ansola. Second order mobility analysis of mechanisms using closure equations. *Meccanica*, 47(7):1695–1704, 2012.

[32] R. A. Fisher. The wave of advance of advantageous genes. *Annals of Eugenics*, 7(4):355–369, 1937.

[33] Kazuko Fuchi, Alejandro R Diaz, Edward J Rothwell, Raoul O Ouedraogo, and Junyan Tang. An origami tunable metamaterial. *Journal of Applied Physics*, 111(8):084905, 2012.

[34] Carlo Galletti and Pietro Fanghella. Single-loop kinematotropic mechanisms. *Mechanism and Machine Theory*, 36(6):743–761, 2001.

[35] Milton R Garza, Edwin A Peraza-Hernandez, and Darren J Hartl. Self-folding origami surfaces of non-zero gaussian curvature. In *Behavior and Mechanics of Multifunctional Materials XIII*, volume 10968, page 109680R. International Society for Optics and Photonics, 2019.

[36] Science Museum Group. Babbage's analytical engine. `https://collection.sciencemuseumgroup.org.uk/objects/co62243/difference-engine-no-1-difference-engine`. Last Accessed May 26, 2022.

[37] Science Museum Group. Difference engine no. 1. `https://collection.sciencemuseumgroup.org.uk/objects/co62243/difference-engine-no-1-difference-engine`. Last Accessed May 26, 2022.

[38] Science Museum Group. Difference engine no. 2, designed by charles babbage, built by science museum. `https://collection.sciencemuseumgroup.org.uk/objects/co526657/difference-engine-no-2-designed-by-charles-babbage-built-by-science-museum-difference-engine`. Last Accessed May 26, 2022.

[39] Elliot Hawkes, B An, Nadia M Benbernou, H Tanaka, Sangbae Kim, Erik D Demaine, D Rus, and Robert J Wood. Programmable matter by folding. *Proceedings of the National Academy of Sciences*, 107(28):12441–12445, 2010.

[40] BM Hespenheide, DJ Jacobs, and MF Thorpe. Structural rigidity in the capsid assembly of cowpea chlorotic mottle virus. *Journal of Physics: Condensed Matter*, 16(44):S5055, 2004.

[41] Miranda Holmes-Cerfon, Louis Theran, and Steven J Gortler. Almost-rigidity of frameworks. *Communications on Pure and Applied Mathematics*, 74(10):2185–2247, 2021.

[42] Myungwon Hwang and Andres F. Arrieta. Input-independent energy harvesting in bistable lattices from transition waves. *Scientific Reports*, 8(1):1–9, 2018.

[43] Myungwon Hwang and Andres F. Arrieta. Solitary waves in bistable lattices with stiffness grading: Augmenting propagation control. *Physical Review E*, 98:042205, 2018.

[44] Myungwon Hwang and Andres F Arrieta. Topological wave energy harvesting in bistable lattices. *Smart Materials and Structures*, 31(1):015021, 2021.

[45] Alexandra Ion, Ludwig Wall, Robert Kovacs, and Patrick Baudisch. Digital mechanical metamaterials. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 977–988, 2017.

[46] Donald J Jacobs, Andrew J Rader, Leslie A Kuhn, and Michael F Thorpe. Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Bioinformatics*, 44(2):150–165, 2001.

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

[48] NASA Jet Propulsion Laboratory Jonathan Sauder. Automaton rover for extreme environments (aree). `https://www.nasa.gov/directorates/spacetech/niac/2017_Phase_I_Phase_II/Automaton_Rover_Extreme_Environments/`. Last Accessed May 26, 2022.

[49] P. Kaliappan. An exact solution for travelling waves of ut = duxx + u - uk. *Physica D: Nonlinear Phenomena*, 11(3):368–374, 1984.

[50] CL Kane and TC Lubensky. Topological boundary modes in isostatic lattices. *Nature Physics*, 10(1):39–45, 2014.

[51] Michael Kapovich and John Millson. On the moduli space of polygons in the euclidean plane. *Journal of Differential Geometry*, 42(1):133–164, 1995.

[52] Michael Kapovich and John J Millson. Hodge theory and the art of paper folding. *Publications of the Research Institute for Mathematical Sciences*, 33(1):1–31, 1997.

[53] Michael Kapovich and John J Millson. On the moduli space of a spherical polygonal linkage. *Canadian Mathematical Bulletin*, 42(3):307–320, 1999.

[54] Shmuel Katz and Sefi Givli. Solitary waves in a bistable lattice. *Extreme Mechanics Letters*, 22:106–111, 2018.

[55] Shmuel Katz and Sefi Givli. Solitary waves in a nonintegrable chain with double-well potentials. *Physical Review E*, 100(3):32209, 2019.

[56] Richard Kaye. Infinite versions of minesweeper are turing complete. `https://web.mat.bham.ac.uk/R.W.Kaye/minesw/infmsw.pdf`, 2007.

[57] Alfred B Kempe. On a general method of describing plane curves of the nth degree by linkwork. *Proceedings of the London Mathematical Society*, 1(1):213–216, 1875.

[58] Jason Z Kim, Zhixin Lu, Ann S Blevins, and Dani S Bassett. Nonlinear dynamics and chaos in conformational changes of mechanical metamaterials. *Physical Review X*, 12(1):011042, 2022.

[59] Jason Z Kim, Zhixin Lu, Steven H Strogatz, and Danielle S Bassett. Conformational control of mechanical networks. *Nature Physics*, 15(7):714–720, 2019.

[60] P Kumar and S Pellegrino. Computation of kinematic paths and bifurcation points. *International Journal of Solids and Structures*, 37(46-47):7003–7027, 2000.

[61] Dongsheng Lei, Alexander E Marras, Jianfang Liu, Chao-Min Huang, Lifeng Zhou, Carlos E Castro, Hai-Jun Su, and Gang Ren. Three-dimensional structural dynamics of dna origami bennett linkages using individual-particle electron tomography. *Nature communications*, 9(1):1–8, 2018.

[62] Zijia Li, Josef Schicho, and Hans-Peter Schröcker. Kempe's universality theorem for rational space curves. *Foundations of Computational Mathematics*, 18(2):509–536, 2018.

[63] Gabriele Librandi, Eleonora Tubaldi, and Katia Bertoldi. Programming nonreciprocity and reversibility in multistable mechanical metamaterials. *Nature Communications*, 12(1):3454, 2021.

[64] H. Lipkin and E. Pohl. Enumeration of singular configurations for robotic manipulators. *Journal of Mechanical Design*, 113(3):272–279, 1991.

[65] Bin Liu, Jesse L Silverberg, Arthur A Evans, Christian D Santangelo, Robert J Lang, Thomas C Hull, and Itai Cohen. Topological kinematics of origami metamaterials. *Nature Physics*, 14(8):811, 2018.

[66] Bo Liu, Wenjie Ge, Dianbiao Dong, Lei Zheng, and Guoxiong Zhang. Kinematic analysis and optimization of a kangaroo geared five-bar knee joint mechanism. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1557–1563. IEEE, 2017.

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

[68] Po-Wei Lo, Christian D Santangelo, Bryan Gin-ge Chen, Chao-Ming Jian, Krishanu Roychowdhury, and Michael J Lawler. Topology in nonlinear mechanical systems. *Physical Review Letters*, 127(7):076802, 2021.

[69] TC Lubensky, CL Kane, Xiaoming Mao, Anton Souslov, and Kai Sun. Phonons and elasticity in critically coordinated lattices. *Reports on Progress in Physics*, 78(7):073901, 2015.

[70] W.X. Ma and B. Fuchssteiner. Explicit and exact solutions to a kolmogorov-petrovskii-piskunov equation. *International Journal of Non-Linear Mechanics*, 31(3):329–338, 1996.

[71] Alexander E Marras, Lifeng Zhou, Hai-Jun Su, and Carlos E Castro. Programmable motion of dna origami mechanisms. *Proceedings of the National Academy of Sciences*, 112(3):713–718, 2015.

[72] Martial Mazars. Statistical physics of the freely jointed chain. *Physical Review E*, 53(6):6297, 1996.

[73] J. Michael McCarthy and Gim Song Soh. *Geometric Design of Linkages*. Interdisciplinary Applied Mathematics. Springer New York, 2010.

[74] Tie Mei, Zhiqiang Meng, Kejie Zhao, and Chang Qing Chen. A mechanical metamaterial with reprogrammable logical functions. *Nature Communications*, 12(1):1–11, 2021.

[75] Zhiqiang Meng, Weitong Chen, Tie Mei, Yuchen Lai, Yixiao Li, and C. Q. Chen. Bistability-based foldable origami mechanical logic gates. *Extreme Mechanics Letters*, 43:101180, 2021.

[76] Ralph C. Merkle, Robert A. Freitas, Tad Hogg, Thomas E. Moore, Matthew S. Moses, and James Ryley. Mechanical computing systems using only links and rotary joints. *Journal of Mechanisms and Robotics*, 10(6), 2018.

[77] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[78] Andreas Müller. Higher-order analysis of kinematic singularities of lower pair linkages and serial manipulators. *Journal of Mechanisms and Robotics*, 10(1), 2018.

[79] Andreas Müller. Kinematic tangent cone–a useful concept for the local mobility and singularity analysis. In *IFToMM World Congress on Mechanism and Machine Science*, pages 337–346. Springer, 2019.

[80] Andreas Müller and Dimiter Zlatanov. *Singular Configurations of Mechanisms and Manipulators*. Springer, 2019.

[81] Mees Muller. A novel classification of planar four-bar linkages and its application to the mechanical analysis of animal systems. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 351 1340:689–720, 1996.

[82] Jun-Hee Na, Arthur A Evans, Jinhye Bae, Maria C Chiappelli, Christian D Santangelo, Robert J Lang, Thomas C Hull, and Ryan C Hayward. Programming reversibly self-folding origami with micropatterned photo-crosslinkable polymer trilayers. *Advanced Materials*, 27(1):79–85, 2015.

[83] Neel Nadkarni, Andres F. Arrieta, Christopher Chong, Dennis M. Kochmann, and Chiara Daraio. Unidirectional transition waves in bistable lattices. *Physical Review Letters*, 116:244501, 2016.

[84] Neel Nadkarni, Chiara Daraio, Rohan Abeyaratne, and Dennis M. Kochmann. Universal energy transport law for dissipative and diffusive phase transitions. *Physical Review B*, 93:104109, 2016.

[85] Neel Nadkarni, Chiara Daraio, and Dennis M. Kochmann. Dynamics of periodic mechanical structures containing bistable elastic elements: From elastic to solitary wave propagation. *Physical Review E*, 90(2):023204, 2014.

[86] Liviu I Nicolaescu et al. *An invitation to Morse theory.* Springer, 2007.

[87] Paul Plucinsky, Benjamin A Kowalski, Timothy J White, and Kaushik Bhattacharya. Patterning nonisometric origami in nematic elastomer sheets. *Soft matter*, 14(16):3127–3134, 2018.

[88] Jordan R Raney, Neel Nadkarni, Chiara Daraio, Dennis M Kochmann, Jennifer A Lewis, and Katia Bertoldi. Stable propagation of mechanical signals in soft media using stored elastic energy. *Proceedings of the National Academy of Sciences*, 113(35):9722–9727, 2016.

[89] Gert Roos, Heleen Leysen, Sam Van Wassenbergh, Anthony Herrel, Patric Jacobs, Manuel Dierick, Peter Aerts, and Dominique Adriaens. Linking morphology and motion: a test of a four-bar mechanism in seahorses. *Physiological and Biochemical Zoology*, 82(1):7–19, 2009.

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

[91] Josef Schicho. And yet it moves: Paradoxically moving linkages in kinematics. *Bulletin of the American Mathematical Society*, 59(1):59–95, 2022.

[92] HS Seung and David R Nelson. Defects in flexible membranes with crystalline order. *Physical Review A*, 38(2):1005, 1988.

[93] Jesse L Silverberg, Arthur A Evans, Lauren McLeod, Ryan C Hayward, Thomas Hull, Christian D Santangelo, and Itai Cohen. Using origami design principles to fold reprogrammable mechanical metamaterials. *Science*, 345(6197):647–650, 2014.

[94] Jesse L Silverberg, Jun-Hee Na, Arthur A Evans, Bin Liu, Thomas C Hull, Christian D Santangelo, Robert J Lang, Ryan C Hayward, and Itai Cohen. Origami structures

with a critical transition to bistability arising from hidden degrees of freedom. *Nature materials*, 14(4):389, 2015.

[95] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2006.

[96] S Sirono. Dilatancy of two-dimensional disk packing. *EPL (Europhysics Letters)*, 96(3):30003, 2011.

[97] Adnan Sljoka. *Algorithms in rigidity theory with applications to protein flexibility and mechanical linkages*. York University Toronto, Ontario, Canada, 2012.

[98] Adnan Sljoka and Alexandr Bezginov. Predicting protein hinge motions and allostery using rigidity theory. In *AIP Conference Proceedings*, volume 1368, pages 167–170. American Institute of Physics, 2011.

[99] Yuanping Song, Robert M. Panas, Samira Chizari, Lucas A. Shaw, Julie A. Jackson, Jonathan B. Hopkins, and Andrew J. Pascall. Additively manufacturable micro-mechanical logic gates. *Nature Communications*, 10(1):822, 2019.

[100] Ileana Streinu and Walter Whiteley. Single-vertex origami and spherical expansive motions. In *Japanese Conference on Discrete and Computational Geometry*, pages 161–173. Springer, 2004.

[101] Tomohiro Tachi. Simulation of rigid origami. *Origami*, 4(08):175–187, 2009.

[102] Krzysztof Tchoń. Singularity avoidance in robotic manipulators: A differential form approach. *Robotica*, 13(6):599–606, 1995.

[103] Krzysztof Tchoń. Singularity avoidance in robotic manipulators: A differential form approach. *Systems & control letters*, 30(4):165–176, 1997.

[104] Shawna Thomas, Xinyu Tang, Lydia Tapia, and Nancy M Amato. Simulating protein motions with rigidity analysis. *Journal of Computational Biology*, 14(6):839–855, 2007.

[105] U. Waheed, C. W. Myant, and S. N. Dobson. Boolean and/or mechanical logic using multi-plane mechanical metamaterials. *Extreme Mechanics Letters*, 40:100865, 2020.

[106] Scott Waitukaitis, Peter Dieleman, and Martin van Hecke. Non-Euclidean Origami. pages 1–8, 2019.

[107] Scott Waitukaitis and Martin van Hecke. Origami building blocks: Generic and special four-vertices. *Physical Review E*, 93(2):023003, 2016.

[108] Zhiyan Y Wei, Zengcai V Guo, Levi Dudte, Haiyi Y Liang, and L Mahadevan. Geometric mechanics of periodic pleated origami. *Physical review letters*, 110(21):215501, 2013.

[109] Walter Whiteley. Rigidity and scene analysis. *Handbook of Discrete and Computational Geometry*, pages 1327–1354, 2004.

[110] Karl Wohlhart. Kinematotropic linkages. In *Recent advances in robot kinematics*, pages 359–368. Springer, 1996.

[111] H. Yasuda, L. M. Korpas, and J. R. Raney. Transition waves and formation of domain walls in multistable mechanical metamaterials. *Phys. Rev. Applied*, 13:054067, 2020.

[112] Hiromi Yasuda, Philip R Buskohl, Andrew Gillman, Todd D Murphey, Susan Stepney, Richard A Vaia, and Jordan R Raney. Mechanical computing. *Nature*, 598(7879):39–48, 2021.

[113] D ZLATANOV, RG FENTON, and B BENHABIB. A unifying framework for classification and interpretation of mechanism singularities. *Journal of mechanical design*, 117(4):566–572, 1995.

VITA

AUTHOR: Michelle Berry

DEGREES AWARDED:

- M.S. (Physics), University of Massachusetts, Amherst, 2020
- B.A. (Mathematics, Physics), Goucher College, 2017

RESEARCH EXPERIENCE:

- Graduate Research Assistant, Department of Physics, Syracuse University (September 2020-May 2022).
- Graduate Research Assistant, Department of Physics, University of Massachusetts, Amherst (June 2019-August 2020).
- Summer Research Student, CERN (ATLAS Collaboration), (June 2016-August 2016)
- Summer Research Student, Department of Mathematics and Statistics, Sam Houston State University (June 2015-July 2015)

TEACHING EXPERIENCE:

- Teaching Assistant, Department of Physics, University of Massachusetts, Amherst (September 2017-May 2019)
- Supplemental Instruction Leader, Academic Center for Excellence, Goucher College (September 2014-May 2017)

ADDITIONAL EXPERIENCE:

- Prospective Student Coordinator, Physics Community Organization, University of Massachusetts, Amherst (January 2019-April 2020)

- Local Organizing Committee (member), UMass Conference for Undergraduate Women in Physics (May 2018-January 2019)

PUBLICATIONS:

- M Berry, ME Lee-Trimble, D Limberg, RC Hayward, and CD Santangelo. Configuration space engineering for the topological gating of mechanical devices. *Under Review*, 2022

- M Berry, YJ Kim, D Limberg, RC Hayward, and CD Santangelo. Mechanical signaling cascades. *Under Review*, 2022

- M Berry, ME Lee-Trimble, and CD Santangelo. Topological transitions in the configuration space of non-Euclidean origami. *Physical Review E*, 101(4):043003, 2020.

- M Berry, V Diaz, B Doleshal, T Martin, E Winn, and M Zhou. The component number of a twisted torus link. *Minnesota Journal of Undergraduate Mathematics*, 2(1), 2017

PRESENTATIONS:

Workshops:

- D Atkinson, M Berry, M. E. Lee-Trimble, "Bits, Knits, and Knots: Using Knitting as a Tool for Teaching STEM Concepts". Workshop at the STEM Education Institute, University of Massachusetts Amherst, May 2020

Contributed Talks:

- M Berry, D Limberg, M. E. Lee-Trimble, R Hayward, C Santangelo, "Configuration space engineering: gating mechanisms by controlling configuration space topology" (virtual talk). Presented at the APS March Meeting 2022

- M Berry, D Limberg, YJ Kim, R Hayward, C Santangelo, “Branching out into computation: using singularities to design mechanical logic” (virtual talk). Presented at the APS March Meeting 2021

- M Berry, R Hayward, C Santangelo, “Sending Signals Through Mechanical Wiring” (virtual talk). Presented at the APS March Meeting 2020

Posters:

- M Berry, R Hayward, C Santangelo, “Designing Mechanical Logic Systems” (poster). Presented at the Multifunctional Materials and Structures Gordon Research Conference, January 2020

- M Berry, R Hayward, C Santangelo, “Signal Propagation in Mechanical Logic Systems” (poster). Presented at the NEW.Mech Workshop, October 2019