TOWARDS DESIGNING A LOCKABLE SELF-FOLDING ORIGAMI

Lohit Malik

A MASTER'S THESIS

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF MASTER OF SCIENCE IN ENGINEERING

RECOMMENDED FOR ACCEPTANCE BY

THE DEPARTMENT OF

MECHANCAL AND AEROSPACE ENGINEERING

Advisors: Dr. Andrej Kosmrlj and Prof. Glaucio H. Paulino

January 2024

ABSTRACT

Origami, an art of paper folding, has displayed its importance in engineering by emerging as a tool for building three-dimensional (3D) structures out of patterned flat films. This has taken a step further through the introduction of self-folding origami. Recently, the term 'locking' has enrooted in the world of structures where origami stands out given its basic nature of folding. The possibility for an origami to self-fold and then lock itself in one of the metastable states can widen its scope in fields such as emergency shelters, robotics, and even the biomedical sector. For designing a self-folding lockable origami, special multi-stable designs have to be created. This work is a step towards it by offering a platform that can be used for quickly creating, testing, and identifying designs or design changes that can lead to multi-stable structures and how it can be exploited for possibly concluding on a lockable origami.

The thesis starts with approximating an origami based on a reduced-order bar and hinge model and quantifying the key elements contributing to the deformation process. This is followed by the design of the master computational strategy connecting a simple CAD user input to the MATLAB code developed. For this, an in-depth discussion on extracting useful information, identifying panels, and discretizing them is done. A thorough theoretical narrative about calculating total mechanical energy and systematically solving for the equilibrium along with deriving relevant analytical expressions is presented. A separate section draws on the aspect of self-folding and showcases strategies for physically connecting external stimuli to folding. Here, a mathematical result is presented that deals with the curvatures obtained as a result of keeping a bilayer plate in a thermal stimulus which is used to conclude upon mechanisms for controlled self-folding. Finally, two examples have been shown that showcase the folding of a well-known Miura origami unit cell under external forces and the self-folding of a simple fold under heat.

ACKNOWLEDGMENETS

I would like to first thank my advisor Dr. Andrej Kosmrlj for his continuous guidance throughout my master's degree. His command on his versatile knowledge is unparalleled and I feel privileged to be able to learn a lot from him by being a part of his group.

Another person who has been a part of my ups and downs is my batchmate and friend, Shaharyar. He has been the person who understood my academic life, motivated me during the tough times and has been there for me every time. I admire his mature nature, his passion for experiments and I have loved his company during my time at Princeton. Within Andrej's group, Tejas Dethe deserves a special mention. He has been the mentor everyone desires. I thank him for eliminating the intimidating feelings I experienced at Princeton through his positive acoustic cues. I felt very comfortable sharing my office with Roy Zhang. He has been the silent catalyst whose energetic and never getting tired nature helped me push myself to my limits. I hold a short but special relation with Jonathan Russ who was a post-doc in Prof. Paulino's group. He is the person whom I look up to for his grasp in mechanics and I thank him for his to the point teachings in both academics and life. Polina Zhilkina is the person whom I knew even before joining Princeton and her small recommendations have been the perfect footsteps for navigating my time at Princeton as an international student.

I cannot forget to mention, the most special person who has experienced every emotion of mine during my master's, Vidushi. I thank her for her patience, compatibility, endless care, blind trust, and utmost confidence in me that has added a new dimension to my life. Most importantly, thank you Mom and Dad for always proving how capable I am and encouraging me throughout my career. Your consistent motivation for the past 2 years has been instrumental in reaching this milestone.

TABLE OF CONTENTS

1. INTRODUCTION

A self-folding lockable origami contains a lot of keywords which carry equal importance in realizing the said goal. Understanding the physical meaning of having a structure that folds itself and locks in the folded state, is the first and the foremost question to answer. Fig. 1 shows a sequence of images that are snapshots of a movie displaying the folding of a flat pattern into a cube [1]. Note that no external loads are present, and no human intervention exists throughout the process. The change in shape takes place automatically, in this case, due to a rise in the ambient temperature. As the temperature is augmented, the panels rotate about one of their edges and this happens in a way that finally a cube is formed.

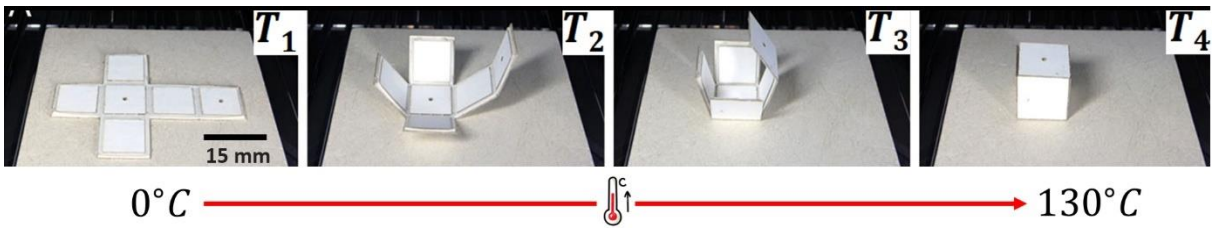


Fig. 1. Self-folding of a pattern into a cube due to rise in temperature from $0°C$ to $130°C$.

1.1. Origami – more than paper art

In the title self-folding lockable origami, the term ‘origami’ is related to the flat pattern lying on the plane in the left-most image in Fig. 1. Though, in general, through origami people recall paper folding art to create models such as dragons, however, origami is much more than that. With wide applications in space satellites [2], temporary foldable structures [3], biomedical devices such as drug-delivery robots [4], and even in packaging and storage [5], the simple property of folding in an origami has proven its power. The folding is ascribed to the creases laid down on the flat pattern which is usually the starting point of folding for most of the origami. These creases are simple set of lines that corroborate the folding due to a difference in thickness of the paper along the creases making the folding stiffness smaller. The side on which these creases are perforated define two types of creases, namely the mountain and the valley

creases. The mountain and the valley folds can be distinguished in a way that when origami is folded about the mountain crease, a mountain is formed, whereas when it is folded about a valley fold, an inverse of a mountain, i.e., a valley is generated. Moreover, the shortest loops formed by a set of creases define the panels in an origami which at the very least are triangles followed by polygons with higher number of edges. Fig. 2 explains the terms discussed here through the flat pattern shown in Fig. 1.

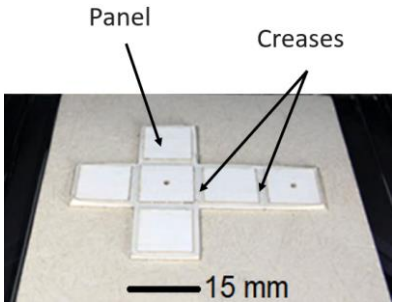


Fig. 2. Definition of creases and panels in an origami.

1.2. Self-folding origami – stimuli and mechanisms

With the definition of an origami and the usual related terms clear discussed, it's time to jump on to the other term in the title which is self-folding. It can be seen in Fig. 1., that the cube is obtained once the flat origami pattern folds itself as a result of the rise in temperature. Note that the same folding process can be realised with the use of some other stimuli too. For instance, electric field was used to bend an electroactive polymer which is a phenomenon that can be exploited along the creases of an origami to control folding [6]. Not only this but magnetic field can also be used to actuate assemblies containing magnetically responsive particles which can be tuned in a desired manner and effectively used to translate an origami structure as a result of local self-folding [7]. Stimuli as simple as light can be employed to be aimed at photo-responsive polymers in order to bend them in a controlled fashion to generate desirable geometries that can be altered with time to introduce movement [8]. Pneumatic systems are one of the widely used resources to guide the flow of pressurized air into or out of an origami

structure in return influencing its stiffness to actuate the whole pattern [9, 10]. Lastly, there are methods other than mechanical stimuli such as the change of pH or some chemical reaction that can lead to actuation [11] and hence can be effectively used in an origami that is to be used in a chemical environment.

Coming back to the use of heat or temperature in Fig. 1 to fold the pattern into a cube, different mechanisms can be used to transfer the effect of stimulus to the pattern. In an origami, these mechanisms are found at the creases as they are ones guiding local folding and hence, the global shape change of the pattern. A commonly used technique is that of a bilayer, which is basically a structure created by bonding two layers together that differ in their properties such as coefficient of thermal expansion if made of a thermally expandable material [12] or the water-absorbing quotient if a hydrogel, etc [13]. Due to a difference in expanded lengths and inability to freely expand, the bilayer bends resulting in a curvature which can be connected to the folding about a crease in an origami. An exhaustive discussion about this has been made in Section 5. However, it's not always necessary to use a bilayer, as, a single layer of a pre-stretched polymer also reacts to the change in temperature and additionally includes the ability to change its shape so as to reach the original state before stretching by exploiting the heat in the environment [14]. Therefore, there are plenty of ways in which an origami can be made to self-fold.

1.3. Locking – a clear definition

Now, in order to understand the meaning of the most important term in the title, i.e., locking, we again come back to the example shown in Fig. 1. Literally, locking means holding something in place and here too it means to have an origami structure that can hold itself in a mechanical configuration on its own. A scientific term to cover this would be to say that the origami is mechanically stable in that configuration. Being stable means that without the influence of any loads or external stimuli, the structure retains its shape and is in an energetically minimum state.

Combining all these terms together results in a self-folding lockable origami. However, let's understand what it would mean to have a self-folding lockable origami by the example of the self-folding cube discussed in Fig. 1. It is clear from Fig. 1 that once the temperature starts to drop back from $130°C$ to $0°C$, the cube will gradually unfold and ultimately, we will have the flat crease pattern at $0°C$. However, if the folded pattern would have locked itself at the stage when the cube was formed, the cube shape would have been retained even after the temperature dropped back to $0°C$. This difference is shown in Fig. 3, where in the top set of images the cube so folded does not lock itself in the folded state whereas the bottom set of images shows the scenario where the pattern would have showcased locking, and the cube shape would have been retained even after the temperature dropped back to its initial value.

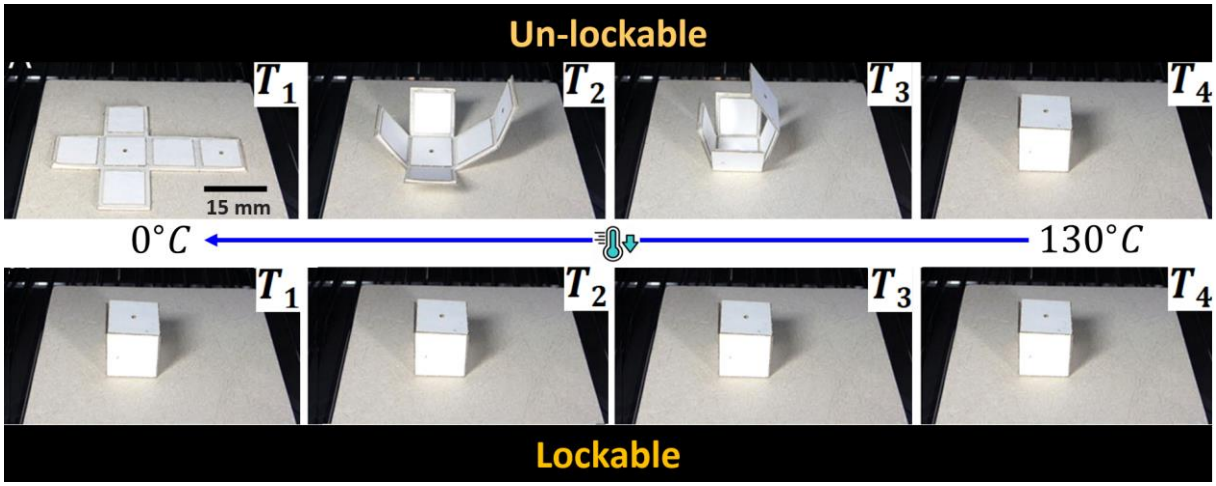


Fig. 3. Difference between the physical scenarios if the flat crease pattern folds to lock in the cube shape (bottom set of images) even after the removal of external stimulus and if it is unable to lock and unfolds to the flat pattern it started with (top set of images).

The self-folding cube presented above was a specific example where locking was not naturally possible. However, in general, for locking to be established in one of the folded states of an origami, it is clear that the structure should be mechanically stable in that state, i.e., a local minimum should exist corresponding to this state. This also implies that being bistable is a

necessary condition for a structure to be lockable. Once such an origami is crafted and a stable state is reached, it is crucial to understand if the structure remains there even after the external stimuli is removed. It is not straightforward to decide this as it is also dependent on the way in which the structure travels from one stable state to another and back. More importantly, how do the local minima corresponding to stable states of an origami change their nature with the external stimuli is a question that can guide us better on locking. This also concludes that bistability is a necessary but not sufficient condition for locking. Therefore, in order to provide a complete picture of locking and what are the conditions that make it possible, next section deals with the stored energy plots in an origami structure during the self-folding process.

1.4. Energy plots and their relationship with locking

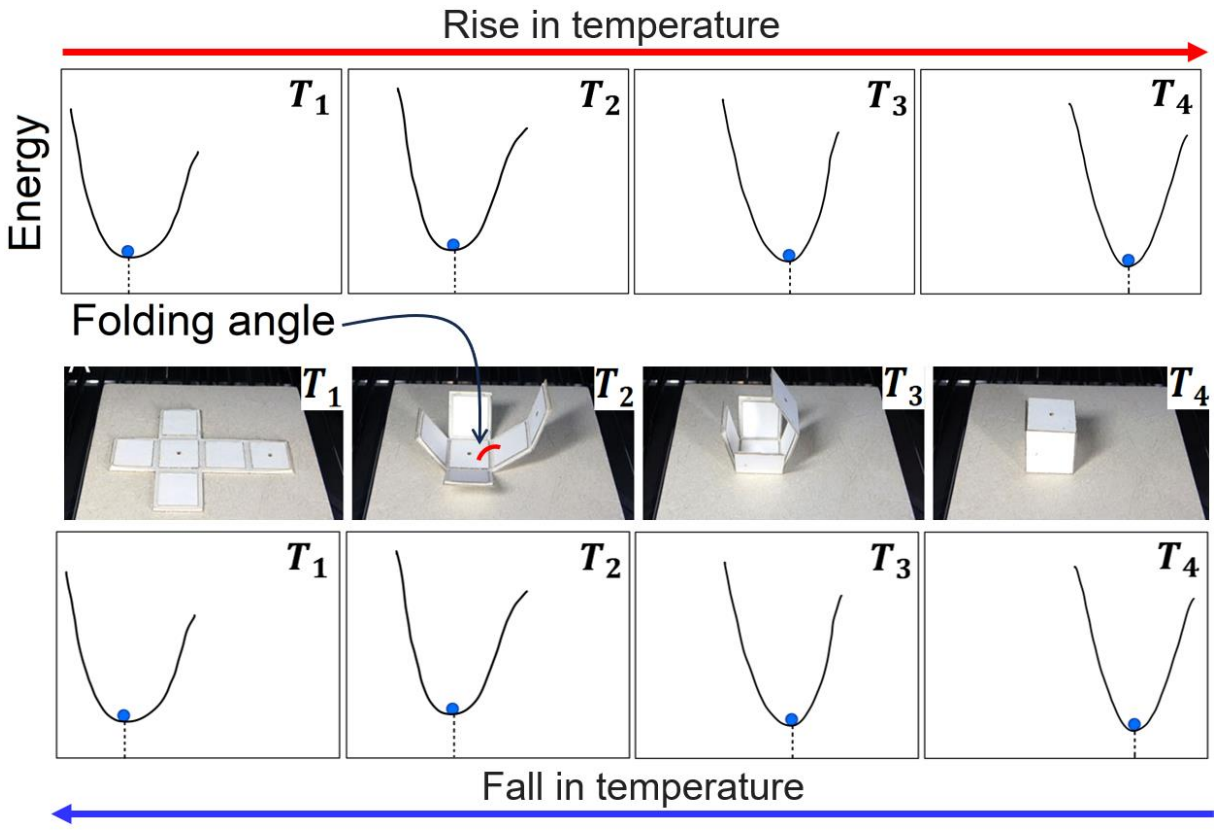


Fig. 4. Schematic of stored mechanical energy plots as a function of one of the folding angles (marked) for a self-folding cube and their variation with the change in the ambient temperature.

For any origami, a mechanically stable state is the one where it can be left freely without any external disturbance, and it can retain that shape. Therefore, it is logical to consider these states to be the ones where the origami can possibly lock itself. However, it is not straightforward to figure this out as the total energy stored in the system at any point of time during the folding process changes dynamically when self-folded. Another way to express this is to say that the energy plots change their nature with the intensity of the external stimuli, and it is hard to assume that if an origami reaches one of the stable states, it will retain that shape in order to lock there. To understand this concept, we start with the schematics of stored energy plots and its variation with the external stimulus for the example of self-folding cube discussed in Fig. 1.

Fig. 4 shows how the stored energy in the self-folding cube varies with the change in the temperature. Focusing on the top set of images, it can be seen how the local minima shifts to the right and also the nature of the plot changes a little with the rise in the temperature. The blue dot at the local minima denotes the local stable state or configuration of the structure at the corresponding value of temperature. Shifting to the plots at the bottom, the variation can be observed with the fall in the temperature and still a dynamic change in the energy plots is clear. However, in both the set of images, the blue dot remains in the only local minima and hence, the pattern first folds to a cube with temperature rise and then unfolds back to the flat pattern as temperature is dropped back.

It turns interesting and a clearer definition of locking is formed when we look at the energy plots for a bistable origami (Fig. 5). Similar to Fig. 4., in this case also, the energy plots vary dynamically with the intensity of the external stimulus, however, because we have a bistable origami under consideration, two local minima exist in the energy plots. From the top set of images in Fig. 5 (i.e., with the rise in external stimulus), it can be observed that the two local minima shift and also the values of the energy at those minima significantly change such that at the highest intensity of the stimulus, the first local minima vanishes, and the blue dot falls to

the second local minima. This means that the origami has attained a physically different stable state.

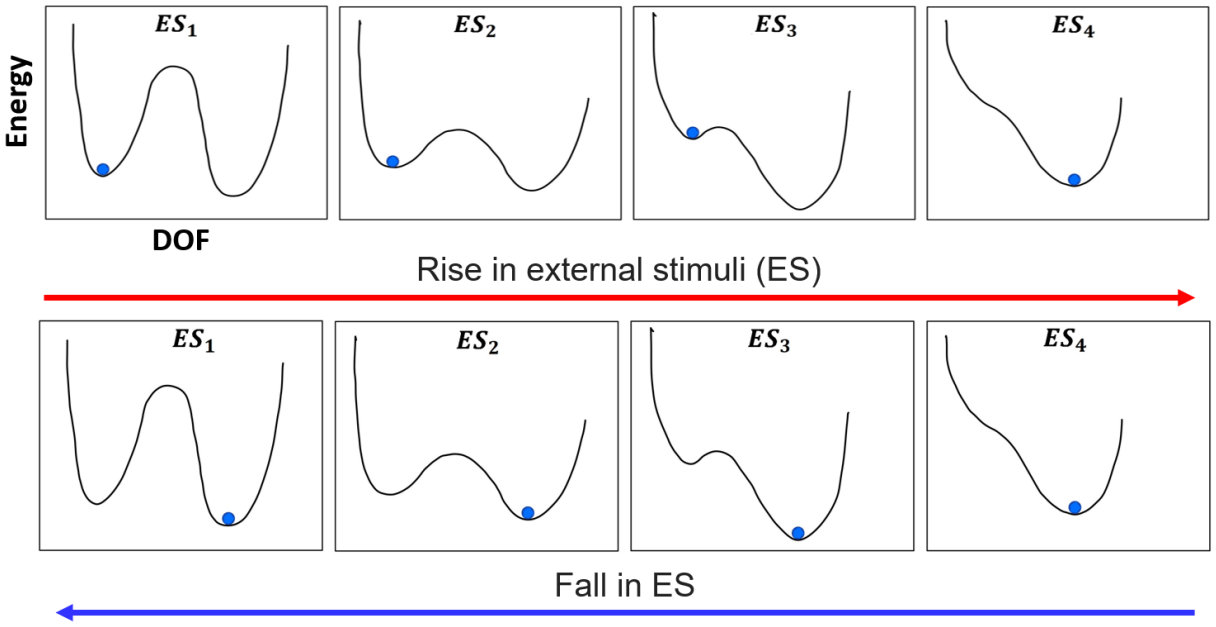


Fig. 5. Schematic of stored energy plots as a function of a degree of freedom (DOF) for a bistable origami self-folding due to a change in an external stimulus.

Focusing on the bottom set of images, a similar change in the energy plots is observed with the fall in the external stimulus. Moreover, the first local minima that had vanished reappears. However, a key difference here is that the blue dot retains itself in the second local minima which is what exactly locking means. It is important to note that the energy plot showcased in Fig. 5 is not true for every bistable origami as it was earlier concluded that being bistable is simply a necessary but not sufficient condition for locking. The energy plots in Fig. 5 present a possible bistable origami locking in the second stable state even after the removal of external stimuli and major reason for that is ascribed to the specific change in the nature of the plots with the stimuli. The manner in which the first minimum vanishes, the blue dot translates to the second minimum, and remains there afterwards implies locking and may not be true for every bistable origami.

Therefore, if one is able to design an origami that has similar energy plots and it retains itself in some other mechanical configuration as compared to what it started with even after the removal of the external stimuli, a self-folding lockable origami has been uncovered.

Once again, the example shown in Fig. 5 was a specific case of a lockable bistable origami. However, based on that, one can identify some key features about the how the energy plots should look like for a general self-locking origami. Firstly, an origami can be multi-stable which means there can be multiple local minima. Secondly, depending on the structure and boundary conditions including the intensity of the external stimuli, energetically, the origami may be lying in one of those minima. Now, in order to move to some other stable state, there has to be a relative change in the values of the stored energy at the current and the other stable states. This variation should be guided by the change in the nature of the energy plots in such a way that local minima in which the origami currently resides in, vanishes and the origami finds itself in another stable state. With the removal of the stimuli, the barrier between the new and the former stable states should rise while preventing possible transfer back to the former stable state, leaving the origami locked in a new stable state. Therefore, depending upon the specific design of the multi-stable origami and the manner in which the intensity of the external stimuli is altered, the energy landscape of the structure will change opening the possibility of self-locking.

1.5. Translating theory to derive a physical meaning of locking

It is equally important to understand how a self-folded structure locks itself based on the energy discussions in the earlier section. Melancon et al. came up with special combinations of triangular panels with formulated geometrical relationships governing their dimensions that resulted in a bistable global structure [9]. Different geometries and connection ideas were tested to figure out what specific constraints lead to a multi-stable structure. The complete structure so created was unfolded pneumatically and it was found that once the structure expands and the

pressure source is turned off (i.e., external stimulus is removed), the structure retains the expanded shape. These series of steps are shown in Fig. 6. This is what a self-actuated lockable structure would look like.

The basic unit that led to the bistability and hence, the locking of the whole structure is shown in Fig. 7a. Here, it can be seen that two angles on a triangular face have been marked, namely α_{IV} and β_{IV}. These angles abide with specific relationships that make sure that the resulting unit is bistable [9]. Also, it can be seen that a total of eight triangles have been connected with each other in a particular fashion which also contributes to the bistability. The authors tried other geometries and combinations too but not all of those turned out to be bistable. The proof of bistability of the unit is clear through Fig. 7b that shows the pressure-volume plot of the unit which is directly related to the energy plots discussed above. There are three points that cut the horizontal axis and the first and the third points from the left correspond to the local minima in the corresponding energy plot and hence denote the stable states of the unit.

These conclusions can be easily made after understanding the relationship between the energy plots and the given pressure-volume plot in Fig. 7b. Actually, the derivative of the stored energy (U) with respect to volume (V) is the pressure (P), i.e., $\frac{dU}{dV} = P$. From this it can be seen that wherever, $P = 0$, or the points at which the plot cuts the horizontal axis in Fig. 7b, we either have a local maxima or a local minima in the corresponding energy plot. Additionally, it can be observed from Fig. 7b, that between the first two points of intersection of the plot with the horizontal axis from the left, the magnitude of P is always positive which means that U grows with V in this region or $\frac{dU}{dV} > 0$ whereas in the section between the second and third intersection points, $\frac{dU}{dV} < 0$ exists. More importantly, based on the slope of the pressure-volume curve, one can comment upon the rate of change of the slope of energy with respect to volume, i.e., $\frac{d^2U}{dV^2}$.

Hence, the energy-volume plot will be concave up $\left(\frac{d^2U}{dV^2} > 0\right)$ between origin and the local maximum and afterwards the local minimum of the pressure-volume plot. Meanwhile, it will be concave down $\left(\frac{d^2U}{dV^2} < 0\right)$ between the local maximum and the local minimum points in the pressure-volume plot. Therefore, based on these relationships, the energy-volume plot can be constructed, and it can be concluded that the first and the third intersection points in the pressure-volume plot denote the local minima in the corresponding energy plot and hence, the stable states of the origami.

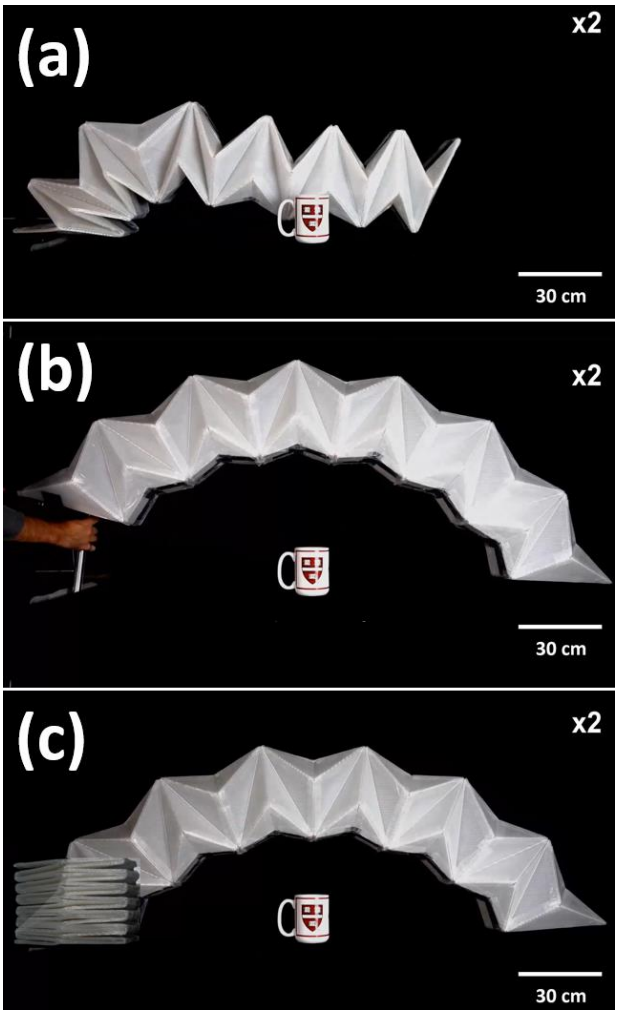


Fig. 6. Pneumatic deployment of a bistable origami structure: (a) Initiating the expansion of the structure using a pressure source on the bottom-left corner (b) Deployment completed and the pressure source is removed (c) Structure retains the expanded state. Retrieved from [9].

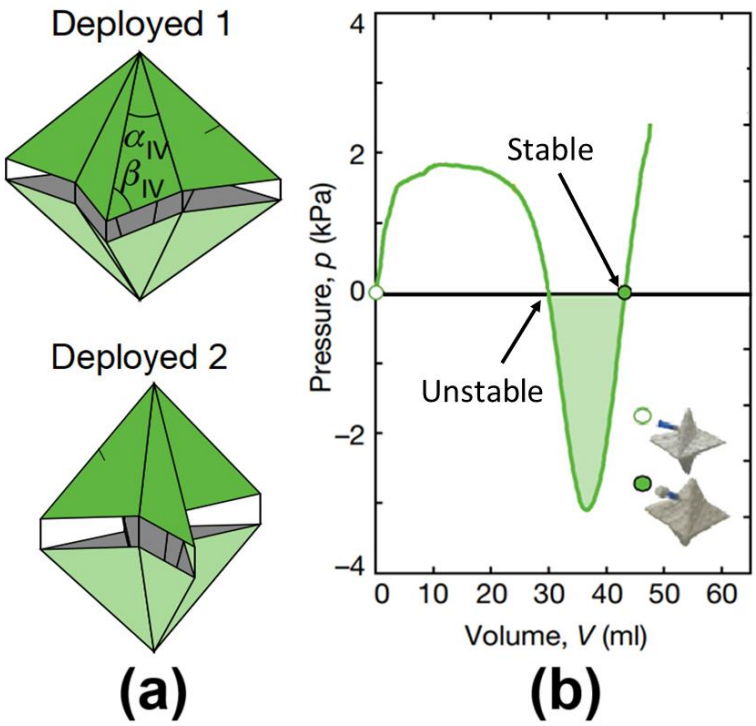


Fig. 7 (a) Basic bistable unit created by connecting triangles in a specific manner (b) Pressure vs volume enclosed plot showcasing the two stable states of the unit. Edited from [9].

2. NEED FOR A ROBUST MODEL

As it can be observed in the example discussed above [9], a special combination of specifically designed panels led to a bistable structure which is necessary condition for a structure to be possibly self-lockable. If the triangles were attached in a different way, bistability would not have been necessarily achieved. This leads to the generic question: how to figure out if a given design initially in a stable configuration will fold in a specific fashion to reach a geometrically different mechanically stable state? Intuitively drafting a crease pattern, printing it to build a physical model of the origami to ultimately find out its stable configurations experimentally is a discrete but tedious idea. Deriving an analytical solution ideally resulting in the set of degrees of freedom (DOF) corresponding to the stable states is only possible in particular simple cases [15, 16, 17]. This means that deriving a generalized analytical solution is close to impossible. Therefore, in the world of computers, its ideal to exploit the power of computation. Designing a computer program that can follow a strategic procedure to output the stable states of an origami design input is a much more efficient way out. A professional term for this process is called numerical modelling. However, to effectively use this numerical scheme to identify self-lockable structures, it is essential to come up with a set of rules that could guide us towards designing structures with locking nature. This will be surrounded by ideally establishing some design principles that will eliminate the testing of designs that might not lock and turn our focus towards more favourable structures. However, first we need to develop a numerical model that can model the folding process of an origami design considered.

2.1. Finite element method & why it is an overdo for origami

The most obvious way to model a physical phenomenon in general is to use finite element method (FEM) [18]. In this numerical method, a complex problem domain is divided into finite number of elements to create a mesh and then solved for each of the elements which is later on extrapolated to the whole geometry. For instance, consider a simple bar with some distributed

load applied on it. The bar is first divided into a number of smaller segments which can be of different shapes such as tetrahedral, cuboidal, etc. based on the analysis. After this, the deformations of each segment due to the load applied is obtained which is used to figure out the deformation of the bar as a whole.

FEM is a very handy technique, but the computation time rises with the index of complication in the geometry and boundary conditions. For an origami folding process, there are a number of deformations taking place and capturing all of them requires the use of a refined mesh which in turn negatively influences the computation time. Moreover, FEM is widely used to generate heat maps over the geometry, for example, for the stress distribution in case of a typical mechanics problem. The power of FEM is reflected in the attention to detail as using it one is able to capture the local deformations and stresses produced that might lead to, for instance, the mechanical failure of the system [18]. In our case, we are more interested to find out the stable configurations of a crease pattern or design considered which means we are concerned with the large deformations leading to a change in overall geometrical configuration of the origami. Therefore, FEM is not the most efficient technique here.

Instead, a model that captures the global shape changes without worrying about the deformations at every point in the geometry is a better candidate. A close approximation to an origami is a truss structure which is a combination of bars connected together in a particular fashion. However, a truss is usually developed to be a static structure used to distribute loads. In origami, the loads are indeed distributed but they influence the processes leading to folding. Therefore, a sense of rotation also needs to be incorporated into the approximation. Clubbing these ideas together creates the bar-and-hinge model in which the in-plane facet deformations are captured using stretchable bars while the bending of panels and folding about the creases is governed using rotational hinges [19, 20]. The working principle of the bar and hinge model is detailed in the next section.

2.2. Bar and hinge model

To understand how a bar and hinge model is an approximation to an origami structure, its crucial to know the types of deformations taking place during the folding of an origami and how the reduced order model is able to capture that. For that, we take the example of an origami known as the Miura origami whose unit cell is shown in Fig. 9.

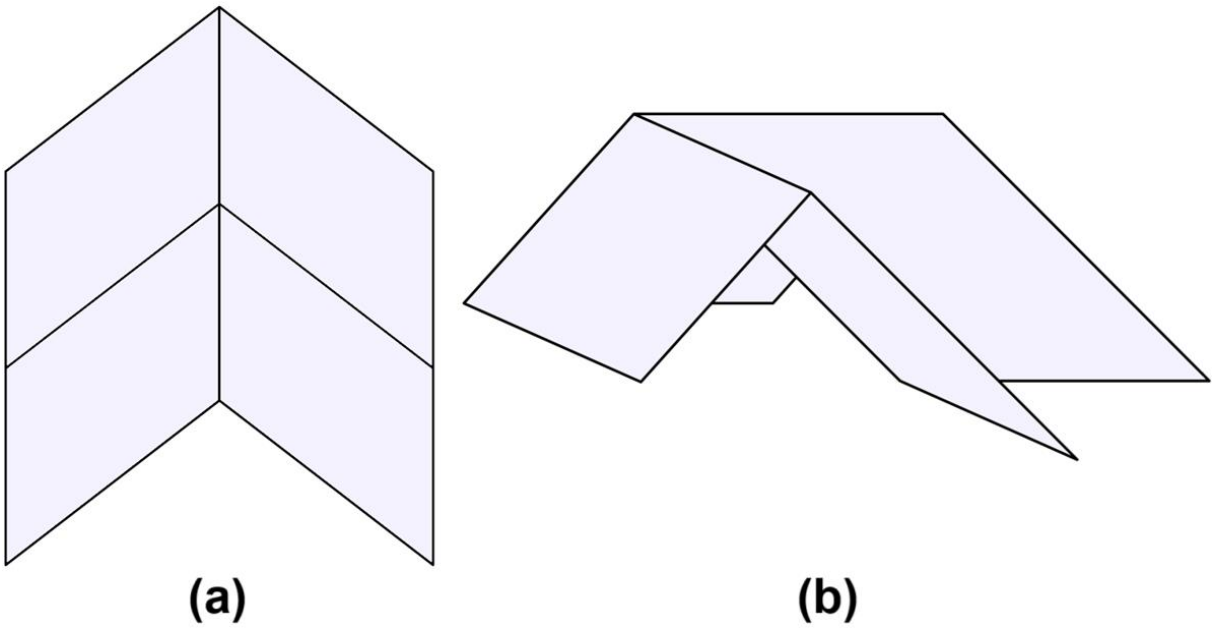


Fig. 9. Miura origami unit cell: (a) crease pattern (b) partially folded configuration.

2.2.1. Types of origami deformations

As the Miura origami unit cell goes from the flat crease pattern (Fig. 9a) to the folded configuration (Fig. 9b), the following deformations take place as also marked in Fig. 10. First of all, the creases (the black lines in Fig. 9) stretch locally (Fig. 10a) which also leads to the stretching of panels or the change in the orientation of the panels (Fig. 10b). Apart from this, the relative angle between two adjacent panels changes significantly during the folding process which can be termed as the folding about the creases (Fig. 10c). Lastly, if the material from which the origami is made is not very stiff, the panels might also bend during folding which can be captured by introducing the diagonals on a panel (Fig. 10d). In case the panel is a

quadrilateral, the bending of the panel is about the shorter diagonal because if we consider the bending energy per unit length, then choosing the shorter diagonal translates to smaller total bending energy. For panels that have edges greater than four, an algorithm dividing the panel in a specific way to ultimately quantify the smallest bending energy can be developed which has been carried out and explained in Section 3.3.

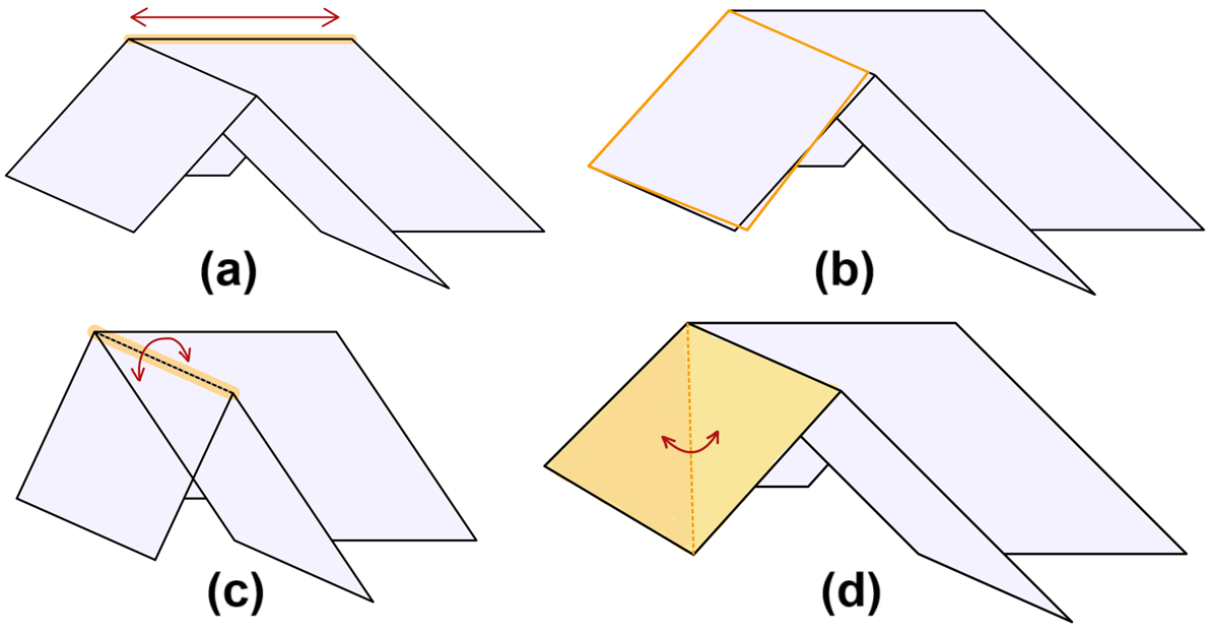


Fig. 10. Miura origami unit cell deformations: (a) stretching of a crease (b) stretching of panels (c) folding about a crease (d) bending of a panel.

2.2.2. Quantifying crease stretching

Now, with the major deformation types discussed, it will be useful to understand how a bar and hinge model captures all of these. For quantifying the stretching of the creases, they are considered as stretchable bars (brown-colored bars in Fig. 11), and to govern change in panel orientation, new stretchable bars are introduced along the shortest diagonals of the panel (blue-colored bars in Fig. 11) [21]. Note that these bars are connected by hinges (represented by black-colored dots in Fig. 11) and in the bar and hinge mode, the hinges are considered to not contribute to the folding process of origami. However, in reality, hinges may influence the

folding process given their possible resistance to folding but it is not captured in the model under consideration. This further clarifies why the bar and hinge model is an approximation of an origami.

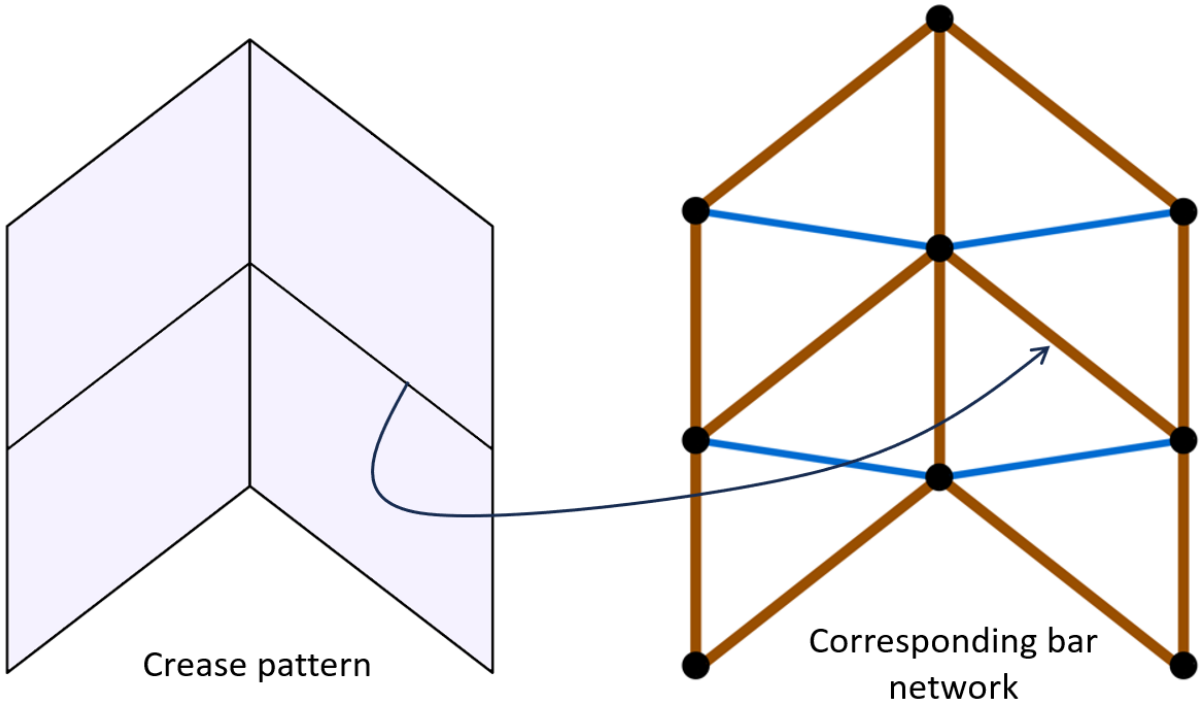


Fig. 11. Bar and hinge model capturing stretching of creases and stretching of panels by approximating defined creases as stretchable bars (brown) and introducing new creases along shortest diagonals (blue).

However, ignoring that, one can formulate the stretching energy due to the change in the length of a bar and then sum it over all the bars in the geometry. If we consider the linear regime, the stretching energy U_{str} can be expressed as:

$$U_{str} = \frac{1}{2} k_{str} (s)^2 \tag{1}$$

Here, k_{str} is the stretching stiffness of the bar and s is its change in length or stretch which can be written in terms of the nodal displacements of the nodes at the ends of the bar in consideration (see Section 4.2.1). k_{str} is related to the in-plane stiffness (Young's modulus E) of the material

from which the origami is made and can be formulated in terms of E, area of cross-section of the bar A and the reference length in the initial configuration L as:

$$k_{str} = \frac{EA}{L} \tag{2}$$

Once again, L can be written in terms of the nodal position of the end nodes making the stretching energy expression a function of the nodal positions. Note that it is straightforward to choose the value of L based on the actual initial length of the corresponding crease, however, quantifying A is not. Dynamically speaking, it is important to take into consideration the mass of the origami while approximating it using a bar and hinge model. This means that while focusing on a discretized panel (i.e., a panel with edge length equal to three), it is essential to distribute the mass of that panel through the three bars forming it. This can be done by first calculating the area enclosed by the triangular panel and then multiplying it by the thickness of the panel to obtain the volume which can be used along with the density to formulate the mass contained in a panel. This mass can be accordingly distributed along the bars based on their respective lengths such that the total mass turns out to be balanced. Also note that a bar separates two distinct panels, therefore, the mass of a bar or the cross-section area A of the bar should account for the masses of both these panels.

With the mass method portrayed, there is an established way to estimate the bar area A. Filipov et al., developed a scheme to formulate the bar areas such that the behaviour of the discretized model matched with the original panel under the same loading conditions [22]. As seen in Fig. 3 in [22], a different discretization technique is used to divide a rectangular panel into four triangles instead of two. Based on this, three types of bars are identified based on their relative positions and the area of each is formulated in terms of the dimensions (length, width, and thickness) of the corresponding panel and the Poisson's ratio (Equation 5 in [22]). Such expressions make sure that the tensile and shear loading response of the discretized model aligns

well with the continuum model one (Fig. 4 in [22]). Therefore, based on the discretization method discussed in this work (Section 3.3), similar expressions can be formulated for the bar area A. Overall, a connection between the actual origami and the discretized model can be made.

2.2.3. Formulating folding about creases and bending of panels

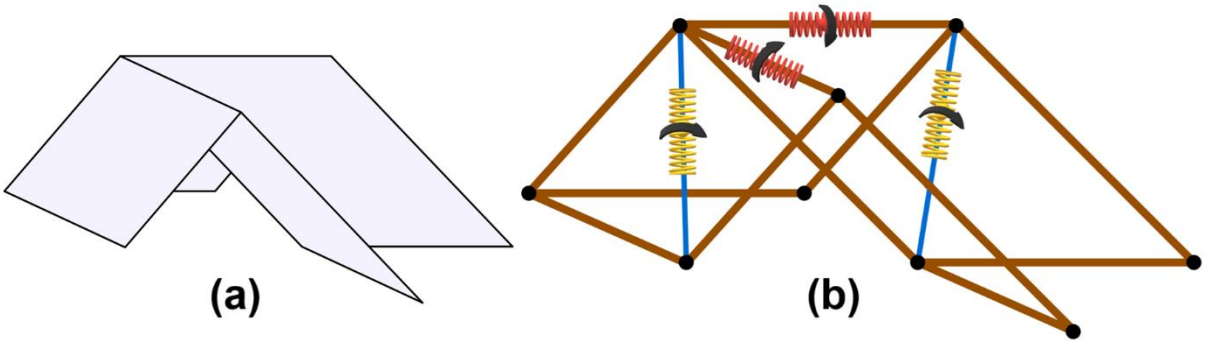


Fig. 12. Torsional springs added to the bar and hinge model to govern folding about creases (red-colored springs) and bending of panels (yellow-colored springs): (a) partially folded Miura unit cell (b) corresponding reduced-order model.

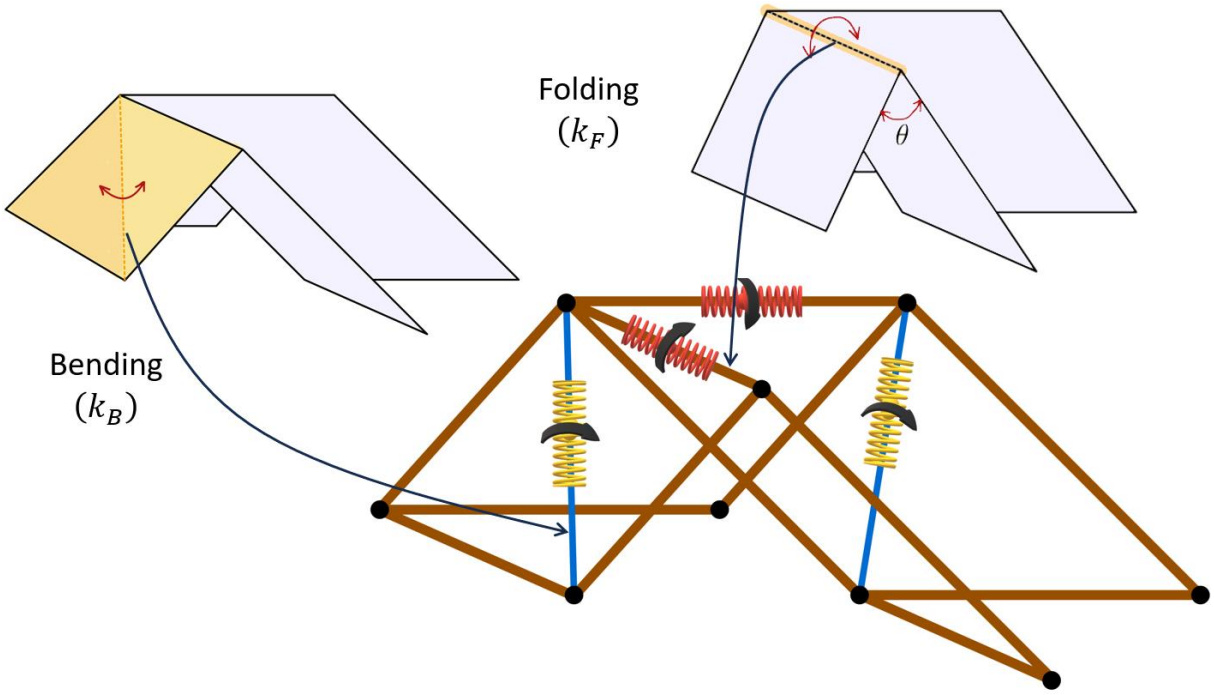


Fig. 13. Functions of two different torsional springs in the bar and hinge model.

In order to capture the folding about of the creases and bending of the panels, a simple bar and hinge model does not suffice. Therefore, torsional springs are considered about the defined creases to govern the folding (red-colored springs in Fig. 12) meanwhile new torsional springs are added along the diagonals marked to control the bending of panels (yellow-colored springs in Fig. 12) [23]. This creates two kinds of springs which are differentiated based on the magnitude of torsional stiffnesses. k_F and k_B denote the folding and bending torsional spring stiffnesses and their functions are shown in Fig. 13.

k_B and k_F are related to the out-of-plane stiffness of the origami material and the relationship can be drafted experimentally. A simple fold can be created with the same origami material which can be made to fold by applying a moment (M_{spr}) whose magnitude is in control. At the same time, the change in the angle between the adjacent planes about the folding creases $(\Delta\theta_{crease})$ can be measured. Finally, considering the linear regime, the ratio of M_{spr} and $\Delta\theta_{crease}$ will be related to the torsional stiffness. Depending on whether the fold under consideration is a defined crease or the one along the shortest diagonal, k_F and k_B can be quantified, respectively. In order to relate the panel or crease dimensions and the material properties to k_F and k_B, we also need a relation of these terms with the moment M_{spr}. From the theory of bending, it is clear that the moment M_{spr} is directly proportional to both Young's modulus E and moment of inertia I about the cross-section of the folding or bending crease while inversely proportional to the curvature ρ so formed. This can be expressed as $M_{spr} = \frac{EI}{\rho}$.

If the cross-section of the crease is rectangular with thickness t and dimension perpendicular to t in the plane of cross-section W, then $I = \frac{Wt^3}{12}$. Note that t takes different values depending upon the folding or the bending crease under consideration. The bending crease has its thickness equal to the thickness of the panel whereas the folding crease is usually thinner than that. Hence, combining the experimental and theoretical point of view, the torsional spring stiffness can be

written in terms of the Young's modulus and dimensions of the crease based on the relation $\frac{E}{\rho}\frac{Wt^3}{12} = k_{rot}\Delta\theta_{crease}$, where k_{rot} takes the value of k_B or k_F.

With the torsional stiffnesses defined, the rotational energy quantifying the folding and bending energies can be formulated as:

$$U_{rot} = \frac{1}{2}k_{rot}(\theta - \theta_0)^2, \qquad k_{rot} \epsilon \{k_B, k_F\} \tag{3}$$

Here, θ is the angle between the adjacent planes divided by the crease line or torsional spring under consideration, also known as the dihedral angle (Fig. 13). Meanwhile, θ_0 is the preferred dihedral angle or the initial dihedral angle of the crease. For example, in case of a Miura unit cell, a look at the flat crease pattern (Fig. 11) clarifies that θ_0 is 180° for all the interior creases. Obviously, no torsional springs are placed at the boundary creases as they only contribute to stretching. Now, θ too can be expressed in terms of the nodal positions of the relevant nodes as we will see in Section 4.2.2. This means that the torsional energies that include folding and bending energies can be formulated in terms of the nodal positions. Overall, the sum of all the mechanical energies during the folding process can be written in terms of the positions of all the nodes present. The mathematical formulation will be detailed in Section 4.

In conclusion, the bar and hinge model turns out to be a simple yet powerful modelling technique especially for an origami. However, despite the said advantages there are still some demerits of the method, and it may not be the best candidate in some cases. For instance, it considers the creases as straight bars but in reality, the crease might be curved which is clear from the presence of curved-crease origami in the literature [24]. For such a case, the curved crease itself can be approximated by discretizing it into a set of short straight bars combined together to form a curved crease. Moreover, due to the curved creases in a curved-crease origami, the panels are no longer planar which means the bar and hinge model that usually

discretizes a panel to figure out the bending creases will not work here. This is because, in such cases, the bending creases too will be curved and based on the discretization scheme developed in Section 3.3 while employing the bar and hinge model, incorrect bending creases will be defined leading to chaos in the reduced-order model.

Additionally, the bar and hinge model assumes that the panels do not buckle during the deformation process, however, there can be specific loading scenarios when the panel might buckle leading to a new bending crease which was not originally captured by the bar and hinge model. At the same time, it is safe to be not concerned of this mode of deformation as every panel is surrounded by stretchable bars which will resist the out-of-plane buckling of the panels.

The aspect of contact and penetration of two panels during the folding of an origami is not captured by the bar and hinge model. Based on the model, two distinct panels can penetrate into each other as no methodology has been drafted to avoid this. However, in reality, such a process is not physically possible. Liu and Paulino devised unique relations such that the torsional spring rotational stiffness rises to infinity as soon as the adjacent panels come closer to each other i.e., when the dihedral angle tends to zero [23], hence, preventing the contact of two adjacent panels. However, this still does not make sure that two non-adjacent panels will not penetrate into each other. In short, there can be scenarios where the bar and hinge model will not work but its simplicity and applicability to wide range of origami is something that makes it very useful for simulating origami.

Now, it is crucial to make an efficient framework that is able to figure out the key elements of a bar and hinge model acting as stretchable bars, folding, and bending torsional springs for any given origami. This takes us to the next section which deals with identifying and naming the nodes, bars, torsional springs, identifying panels, and discretizing them to obtain the bending creases.

3. MASTER COMPUTATIONAL DESIGN

With the reduced order model discussed, a computational strategy needs to be formulated to make the bar and hinge model work. Given that we are trying to develop an effortless framework for computationally modelling the folding process of any origami design considered, the first condition should span over the minimal user input. When a user thinks of an origami, usually, a crease pattern is a way to imprint it. A computationally intuitive way to draw that crease pattern is to use a computer-aided design (CAD) platform such as AutoCAD. An example of a Miura origami CAD developed in AutoCAD is shown in Fig. 14 that comprises of two-unit cells. Drawing such or a larger symmetric design is very easy as one can simply draw one unit cell and copy and paste it over and over again to build a tessellation. With that CAD file ready as the only simple user input, we exploit the data to extract useful information from it.

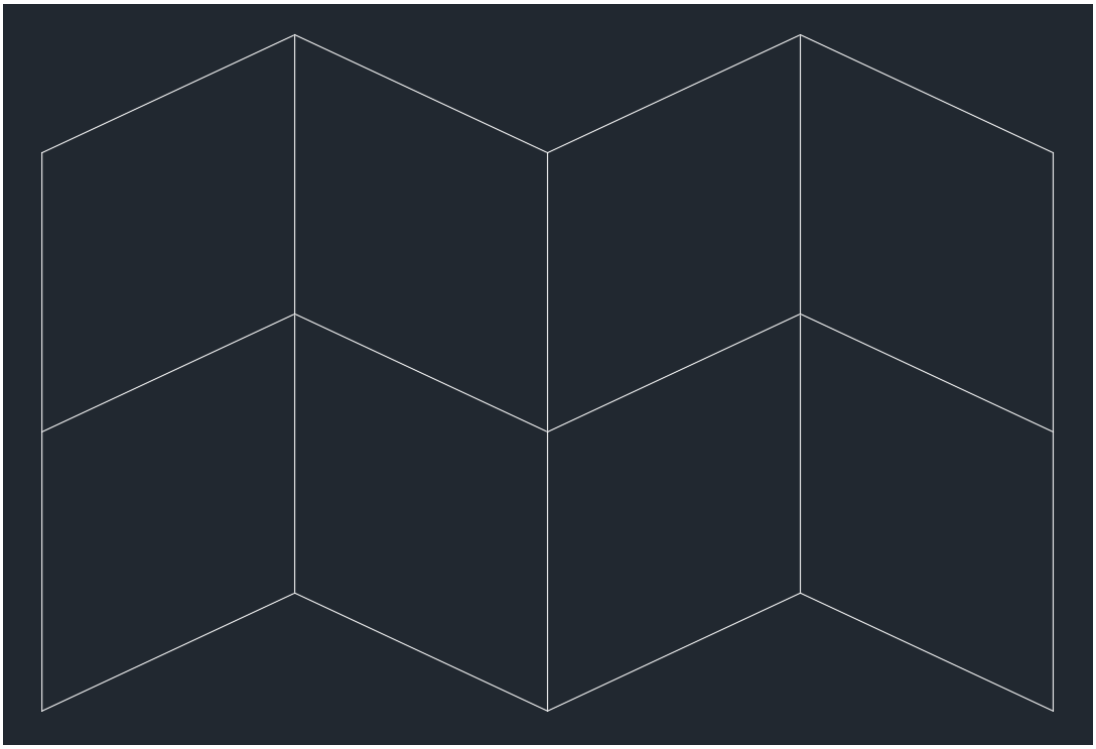


Fig. 14. A computer-aided design (CAD) of a Miura origami (2-unit cells) designed in AutoCAD.

3.1. A seamless pathway between CAD and MATLAB

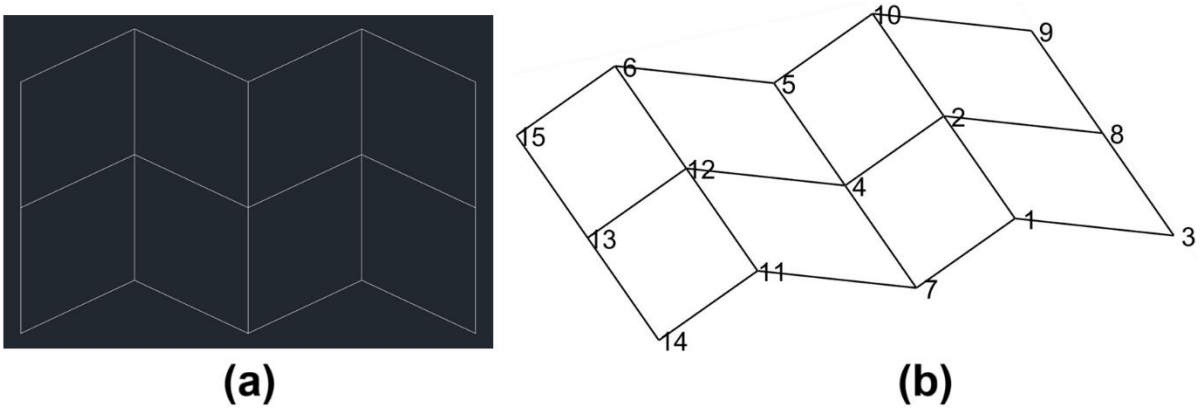


Fig. 15. Code successfully recreates the crease pattern from CAD file while presenting and storing useful information such as nodal names and positions: (a) Crease pattern drawn in AutoCAD (b) Crease pattern recreated in MATLAB.

Any CAD file contains the information related to the line segments and curves in the design and how they are connected with one another. In our case, i.e., the crease pattern of an origami as shown in Fig. 14., we have bunch of straight-line segments connected in a particular fashion with each other and this information is contained in the CAD file. The code developed in MATLAB contains an initial section that transports this information to MATLAB and processes it to recreate the crease pattern while numbering each vertex in a specific manner such that the information regarding the position of that vertex and other vertices it is connected to is stored in form of matrices. Specifically, two matrices are created. The first one deals with the nodes defined and their positions. This matrix consists of rows equal to the number of nodes identified and three columns which contain the X, Y, and the Z coordinates of that node, resulting in a matrix of the size equal to the number of nodes × 3.

The second matrix stores the connections of every node and in this case also, the number of rows in this matrix are equal to the number of nodes identified. Meanwhile the number of columns are equal to the highest number of connections a node has in the complete structure.

For instance, if in an origami, say, node 5 contains 4 connections which is the highest possible number of connections, then the number of columns in the matrix will be equal to 4. For nodes with smaller number of connections, say, 2, only the first two columns corresponding to that node or row number are filled whereas the remaining are left empty. This way one is easily able to access the connections of any node in an origami design considered. A working example of the same is shown in Fig. 15., when the code is able to recreate the crease pattern along with the useful information. The nodes are numbered in a specific order that does not necessarily match a logical order, but the code stores the information such as the current position of node 4 is $(2.5229,1.0794,0)$ and it is connected to nodes $\{2,7,12,5\}$.

3.2. Identifying panels

Once the nodes have been named and connections information has been established, next step is to identify the panels. This is an important step because this way we will be able to find out which panels have length greater than three and are required to be divided in a way to exhibit minimum bending energy. Visually as in Fig. 15, one can see that nodes $\{2,1,7,4\}$ build a quadrilateral shaped panel but how to use the data generated in the Section 3.1 to produce this information, is something of interest. Specifically, which nodes together make a closed loop needs to be figured out. However, at the same time one has to make sure to avoid scenarios like considering $1-3-8-9-10-2$ to be a loop as it consists of two smaller loops $1-3-8-2$ and $2-8-9-10$ (Fig. 15). Hence, the deal is to find out the nodes that come together to form the smallest possible closed loop. We first start with the case in which the origami or the crease pattern is two-dimensional (2D) or flat.

3.2.1 Identifying panels in a 2D origami crease pattern

In a 2D or flat crease pattern, one thing that can be advantageous is that the cross-product of any two vectors in the crease pattern will always be perpendicular to the plane in which the

crease pattern lies, and this is what has been exploited to build an algorithm for identifying panels in a 2D pattern. The procedure has been detailed below with a set of figures making it easier to visualize. We first choose any node in the pattern randomly and also one of its connections to start with. For example, as shown in Fig. 16a, we consider node 4 (say, parent node) and one of its connection i.e., node 2 in this case. Now, we can consider a vector joining nodes 4 and 2 (going from node 4 to node 2 i.e., $\vec{r}_{24}$ which is shown in Fig. 16b).

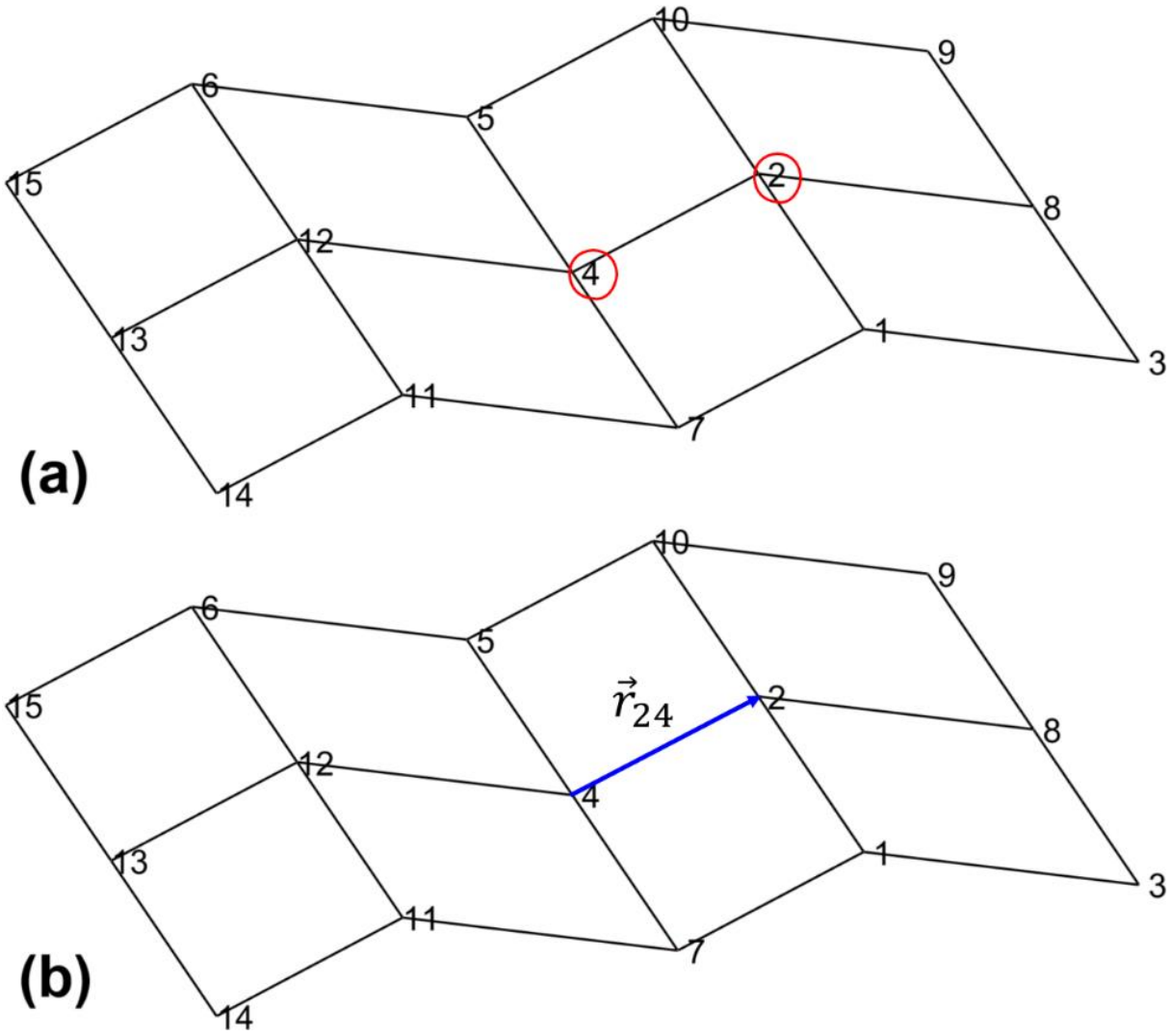


Fig. 16. Procedure of identifying panels in a 2D crease pattern: (a) Choosing a node and one of its connections (marked by red dots). (b) Connecting the chosen nodes to create a vector.

Now, we consider every connection of node 2 apart from the parent node (node 4) which in this case are nodes {1,8,10} and create the vectors pointing from node 2 to these connections (shown by red-colored vectors in Fig. 17 as $\vec{r}_{12}$, $\vec{r}_{82}$, and $\vec{r}_{102}$). With these vectors defined, we trace the rotation for going from $\vec{r}_{24}$ to each red-colored vectors, one by one. It can be observed that while going from $\vec{r}_{24}$ to $\vec{r}_{12}$ and $\vec{r}_{82}$, one has to make a clockwise (CW) turn whereas while going from $\vec{r}_{24}$ to $\vec{r}_{102}$, anti-clockwise (ACW) rotation works. Therefore, we choose node 10 as a part of the loop to be traced. In case we had more nodes that could have been traversed by rotating ACW, we would have chosen the node that lies at the largest ACW angle of rotation.

Visually, it is easy to observe how one can choose the correct node to be a part of the loop, however, it is equally crucial to understand how this can be achieved mathematically for implementation in the code. Once $\vec{r}_{24}$ has been defined in Fig. 16, it can be used to define a new vector rotated in a particular direction, say ACW. In this case, if the origami lies in the XY plane, we can create a rotation matrix for the rotation of a vector about the Z-axis in the ACW direction by a magnitude γ i.e., $R_{Z_{ACW}} = \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$. This can be multiplied by $\vec{r}_{24}$ to generate $\vec{r}'_{24}$ which is simply the ACW rotated version of $\vec{r}_{24}$. Now, taking the cross-product of $\vec{r}_{24}$ and $\vec{r}'_{24}$ and normalizing it gives a reference vector $\vec{q}_{ACW} = \frac{\vec{r}_{24} \times \vec{r}'_{24}}{|\vec{r}_{24} \times \vec{r}'_{24}|}$ that is perpendicular to the XY plane or the plane in which the origami lies. This will be used to identify the nodes creating a loop as detailed ahead.

For each of the vectors $\vec{r}_{12}$, $\vec{r}_{82}$, and $\vec{r}_{102}$, the normalised cross-products $\frac{\vec{r}_{24} \times \vec{r}_{12}}{|\vec{r}_{24} \times \vec{r}_{12}|}$, $\frac{\vec{r}_{24} \times \vec{r}_{82}}{|\vec{r}_{24} \times \vec{r}_{82}|}$, and $\frac{\vec{r}_{24} \times \vec{r}_{102}}{|\vec{r}_{24} \times \vec{r}_{102}|}$ are calculated, respectively. Now, if simply any of these vectors and $\vec{q}_{ACW}$ turn out to be the same, then that vector lies on the ACW turning side of $\vec{r}_{24}$, and hence, the node at which it ends might be the node that may be a part of the loop. In this example, as seen in Fig. 17,

only $\frac{\vec{r}_{24}\times\vec{r}_{102}}{|\vec{r}_{24}\times\vec{r}_{102}|} - \vec{q}_{ACW} = \vec{0}$, hence we say that node 10 maybe a part of the loop. To be sure, if node 10 is a part of the loop or not, we need to see if that node lies at the largest ACW turning angle or not. Although in this case, node 10 is the only node that lies on the ACW side of $\vec{r}_{24}$ but if there were more than one nodes on the ACW side, then we would calculate the turning angles for all those nodes and choose the largest one. This can be done through the dot product of $\vec{r}_{24}$ and the vector on the ACW side. In this case, the ACW turning angle is $\cos^{-1}\left(\frac{\vec{r}_{24}\cdot\vec{r}_{102}}{|\vec{r}_{24}||\vec{r}_{102}|}\right)$. With this formulation coded, it is simple to figure out nodes at the largest ACW turning angle and store them as a node of the loop to be identified.

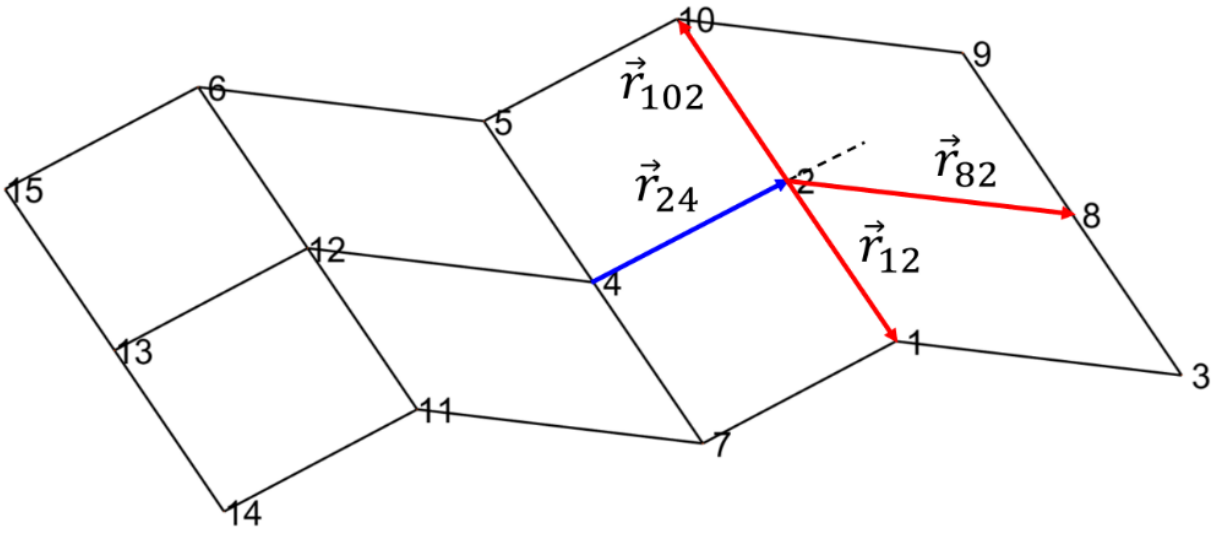


Fig. 17. Considering vectors from node 2 to all its connections to calculate angle of rotation on traversing from the blue-colored $\vec{r}_{24}$ to all the red-colored vectors, i.e., $\vec{r}_{12}$, $\vec{r}_{82}$, and $\vec{r}_{102}$.

If we continue to do the same for connections of node 10 and the forthcoming nodes until we find the parent node i.e., node 4 again, we have discovered a loop. In this case, after choosing node 10, if one follows the procedure discussed, node 5 turns out to be the node at the largest ACW turning angle. Once we reach node 5 and continue the process, we find node 4 abiding with the rules. As node 4 is the node we had started with, we have discovered a loop i.e., loop $4-2-10-5$ in this case (Fig. 18). Recall that after considering node 4 we chose node 2 as

one of its connections to trace out a loop $4-2-10-5$. The next step will be to consider some other connection of node 4, such as node 5, for instance. In this case, the loop traced will be $4-5-6-12$. Hence, these steps can be carried out for every node in the crease pattern until we find out all the loops.

However, the present technique can also lead to repetitions as one may discover the same loop by starting with different nodes. Therefore, in order to eliminate redundancy, the concept of half-edges is employed. This idea says that once an edge, say from node A to node B has been traversed (this does not include the half-edge in the opposite direction, i.e., from node B to node A) while discovering a loop, it can be flagged so that it is not considered again. For instance, in the example being discussed, once the loop $4-2-10-5$ has been traced, we can mark the edges $4-2$, $2-10$, $10-5$, and $5-4$ and avoid any scenario of loop discovery that includes the trace of any of these edges (as shown in Fig. 19, the traversed half-edges have been marked by blue-colored half arrows). Using this technique, it is also made sure that the loops discovered have the same orientation. For instance, the two loops found here showcase ACW orientation.

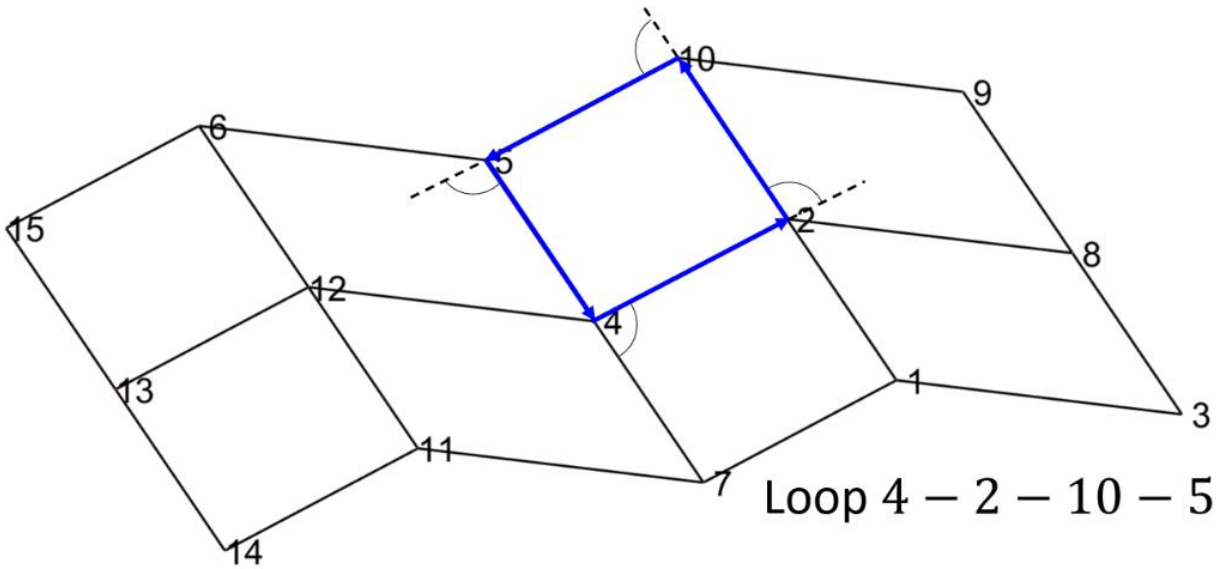


Fig. 18. Discovering loop $4-2-10-5$ based on the procedure discussed for identifying panels in a 2D crease pattern.

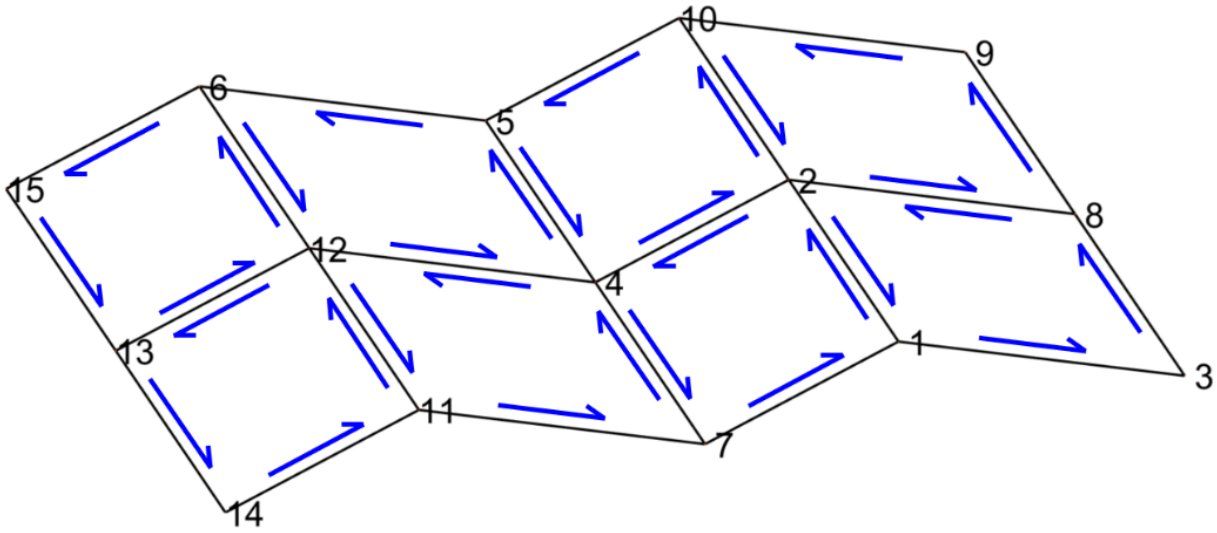


Fig. 19. Employing the concept of half-edges to eliminate redundancy in identifying all the panels in a 2D crease pattern. Blue-colored half-arrows depict traversed half-edges.

3.2.2. Generalized approach & pseudo code for 2D panel identification

The example discussed above is specific but here the generic algorithm is discussed which can be applied to any 2D origami crease pattern to identify the panels. First, with the data generated in Section 3.1, the code has stored the nodes of an origami, their positions, and their connections in form of matrices. We go over each of these nodes one by one and *for* every node (say, current node), all its connections are identified. *For* every of its connections (say, connection 1) a vector pointing from current node to connection 1 $(\vec{A})$ is created. The rotation matrix quantifying, say, the ACW rotation of a vector about an axis perpendicular to the origami plane (R_{ACW}) is formulated using which the ACW rotated $\vec{A}$, i.e., $\vec{A}_R = R_{ACW}\vec{A}$ is drafted. Based on this, their normalized cross-product is obtained and stored as $\vec{A}_{ACW} = \frac{\vec{A}\times\vec{A}_R}{|\vec{A}\times\vec{A}_R|}$.

Now, *for* every connection of connection 1 (say, connection 2), a vector from connection 1 to connection 2 $(\vec{B})$ is written followed by the normalised cross-product $\vec{C} = \frac{\vec{A}\times\vec{B}}{|\vec{A}\times\vec{B}|}$. Only if $\vec{C} - \vec{A}_{ACW} = \vec{0}$, it means connection 2 lies on the ACW side of the $\vec{A}$ and as all of this is running

over a *for* loop, we store all such nodes in a vector (say, ACW nodes). With this, *for* every node in the ACW nodes vector (say, ACW node), we define $\vec{D}$ going from connection 1 to the ACW node. Using this, the ACW turning angle for every ACW node is calculated as $\cos^{-1}\left(\frac{\vec{A}\cdot\vec{D}}{|\vec{A}||\vec{D}|}\right)$ and also stored in a vector. Finally, the node with the largest ACW turning angle is chosen as a part of the loop being identified. The second outermost *for* loop runs until the current node is met again (governed by a *while* loop) and finally, the loop identified is used to flag the edges that have been traversed. The above procedure is written in form of a pseudo code below.

---

for all the nodes identified (termed as current node)
  for every connection of current node (termed as connection 1)
    while connection 1 is not same as current node
      create vector $\vec{A}$ pointing from current node to connection 1
      multiply $\vec{A}$ by an ACW rotation matrix R_{ACW} about an axis perpendicular to the plane of origami to form a rotated vector $\vec{A}_R$
      take the cross-product $\vec{A}_{ACW} = \vec{A} \times \vec{A}_R$ and normalised it
      for all the connections of connection 1 (termed as connection 2)
        create vector $\vec{B}$ pointing from connection 1 to connection 2
        define $\vec{C} = \vec{A} \times \vec{B}$ and normalise it
        if $\vec{C} - \vec{A}_{ACW} = \vec{0}$
          store connection 2 in a column vector (ACW nodes vector)
          for every node in the ACW nodes vector (termed as ACW node)
            define $\vec{D}$ going from connection 1 to current ACW node
            calculate the ACW turning angle $= \cos^{-1}\left(\frac{\vec{A}\cdot\vec{D}}{|\vec{A}||\vec{D}|}\right)$
            store the angles in a vector (turning angles)
          end
        end
        find the ACW node with largest turning angle and add it to the loop being identified

```
            until current node is reached
        end
    end
end
    flag the edges of the loop identified
end
```

Ultimately, this algorithm leads to the development of an efficient section of code using which one can effortlessly identify all the panels in any 2D origami crease pattern. However, in a three-dimensional (3D) origami design which is usually the case with origami that do not start their folding process from a flat state, we cannot exploit the concept we used for the 2D case, i.e., the cross-product of any two vectors lying on the origami surface points in the same direction. A different technique is developed as shown in Section 3.2.2.

3.2.3. Identifying panels in a 3D origami using BFS algorithm

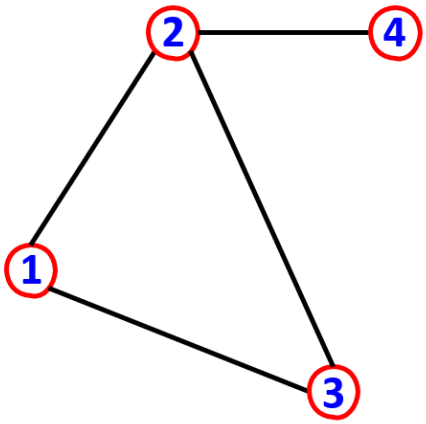


Fig. 20. Working of breadth-first search (BFS) algorithm using a graph.

For a 3D origami, a well-known algorithm from graph theory in the field of computer science is chosen. This is known as the breadth-first search (BFS) algorithm and is used to identify shortest cycles in a graph [25]. Our reduced-order model considers origami as a set of bars

connected together and the computational data currently treats that as a bunch of vertices connected together in a particular fashion. This is exactly what is the requisite for employing the BFS algorithm.

Let's take an example of a random graph (Fig. 20) to understand how this algorithm works. This graph consists of 4 nodes and a cycle $1 - 2 - 3$ which we can see visually. Here is how the BFS algorithm works to identify this cycle. A queue data structure is created, and the nodes are added to it as they are traversed. While traversing the nodes, a parent node and a distance is assigned to each. This distance is assigned as distance (child) = distance (parent) + 1. To start with, one can choose any node, say, we consider node 1 and we intentionally assign its parent as -1 and a distance of 0. Also, this is the first node in the queue. This arrangement is shown in the Table 1.

Node	1	2	3	4
Parent	-1			
Distance	0			

Queue	1			

Table 1. Working of BFS. Assigning parent nodes and distance to each traversed node along with updating the queue.

Now, we consider the connections of node 1, which, in this case are nodes 2 and 3. We add these nodes to the queue and also update the parent and distance column in the table (update shown in Table 2). Once again, the distance (node 2) = distance (node 1) + 1 = 0 + 1 = 1. With this, we strikethrough the first element in the queue and move on to the next element to follow the same procedure. The next element in the queue is node 2 which is connected to nodes 1, 3, and 4. We ignore node 1 as it was the node we started with and consider node 4 to update its

parent and distance as they are empty (Table 3). However, it's worth noting that the parent and distance of node 3 are already present.

Now, if parent (node 3) $\neq$ node 2 and parent (node 2) $\neq$ node 3, then we have discovered a cycle. In this case, parent (node 3) $= 1$ and parent (node 2) $= 1$, this means that nodes 2 and 3 are a part of the cycle which can be found by backtracking their respective parents. The parent of both of these is node 1, hence, nodes 1,2, and 3 make a cycle. The above procedure can be carried out by starting with any node to always find out the same cycle, $1 - 2 - 3$ in this case. Therefore, using BFS, one can identify the presence of such cycles in a graph.

Node	1	2	3	4
Parent	-1	1	1	
Distance	0	1	1	

Queue	~~1~~	2	3	

Table 2. Working of BFS. Assigning parent node and distance to nodes 2 and 3 apart from updating the queue.

Node	1	2	3	4
Parent	-1	1	1	2
Distance	0	1	1	2

Queue	~~1~~	2	3	4

Table 3. Working of BFS. Assigning parent node and distance to node 4 apart from updating the queue.

3.2.4. Generalized approach & pseudo code for BFS algorithm

The example shown in Fig. 20 to identify the shortest cycles in a graph was specific, but it can be generalized for any graph which can be used for identifying panels in a 3D origami. First, *for* every node in the origami identified (say, current node) and stored (see Section 3.1), a queue data structure is created and the first element in queue is taken as the current node. With this, distance and parent vectors are also formed. Every entry in the distance vector is set to be a huge number, say, 10^9. This is done because it will be unrealistic and rare to have an origami with these many number of nodes such that the distance at some node is of the order 10^9. However, the entry of distance at the index equal to the current node is changed to 0. Similarly, to the distance vector, every entry in the parent vector is -1. Now, *while* the size of the queue is greater than zero or *while* the queue is not empty, we list all the connections of the node placed in the first element of queue, i.e., connections of queue(1). *For* each of these connections (say, child), *if* the element in the distance vector at the position child or distance(child) $= 10^9$, a loop does not exist. Instead, we update this distance by replacing it by the sum of the distance at the first element in the queue and the numeral 1. Moreover, the parent at the index equal to child is replaced by queue(1). Finally, the child is added to the next available spot in the queue.

However, in case the *if* condition is not true, then there might be a possibility of identifying a loop which depends on further conditions. *If* the parent of queue(1) is not the child and the parent of the child is not queue(1), a cycle exists. More importantly, queue(1) and the child are a part of this loop. We backtrack the successive parents of the first element in the queue until you reach the current node and create a subloop called subloop 1 out of it such that it looks like [queue(1), parent(queue(1)), parent(parent(queue(1))),…, current node]. Similarly, we trace back the successive parents of the child to create a similar subloop. This subloop 2 takes the form [child, parent(child), parent(parent(child)), …]. Note that, the last element of the subloop 2 is not current node like in the case of sub loop 1. This is way to avoid repetition of nodes in a

loop to be identified and now, these two subloops can be smartly combined together to result in a loop which looks like [queue(1), parent(queue(1)), parent(parent(queue(1))),…, current node,…, parent(parent(child)), parent(child), child]. As soon as this cycle is stored, the related half-edges are marked as traversed making sure that no edge is re-travelled. The above generic algorithm has been written in form of a pseudo code as shown below:

---

for every node in the origami (termed current node)
  add current node to queue
  create a distance vector of size equal to the number of nodes in the origami with each entry equal to 10^9
  distance(current node) = 0
  create a parent vector of the same size as the distance vector and assign every element as -1
  while the queue is not empty
    list all the connections of the node in the first element in the queue, i.e., of queue(1)
    for each of these connections (termed child)
      if distance(child) $= 10^9$
        distance(child) = distance (queue(1)) + 1
        parent(child) = queue(1)
        add child to the next available spot in queue
      elseif parent(queue(1)) $\neq$ child ***and*** parent(child) $\neq$ queue(1)
        backtrack to parents of first element in the queue to make a subloop 1
        backtrack to parents of child to make subloop 2
        cycle identified = [subloop 1, flip(subloop 2)]
        store the cycle & flag edges of the cycle identified
      end
    end
    queue(1) = []
  end
end

Therefore, for the case of a 3D origami, BFS can be employed along with the concept of half-edges to avoid any sort of redundancy. The BFS used in the code finds out the shortest cycles for every node in the origami pattern while avoiding the edges already traversed. The accuracy of this algorithm is clearly seen when it is able to print the cycles in an origami comprising of panels with different lengths. For example, a Kresling origami pattern consists of an arrangement of connected triangular panels with two hexagons at the top and the bottom (schematic shown in Fig. 21a). In total, such a pattern has 12 triangular and 2 hexagonal panels. The code is able to first recreate the origami pattern based on the approach discussed in Section 3.1 (Fig. 21b) and then also able to print all the 14 panels present in the pattern (Fig. 21c).

One key observation in this technique is that the orientations of the panels printed are uniquely defined. If we use the right-hand thumb rule and turn the fingers in accordance with the orientation of the cycles printed (Fig. 21c), we notice that the thumb always points inside the 3D entity. For example, consider the printed loop $9-10-12$ in Fig. 21c. Using the right-hand thumb rule, it can be observed that the thumb points into the origami pattern. The same can be confirmed for any other loop. This observation is similar to the 2D panel identification case, as, there we had the thumb pointing out of the plane in which the crease pattern lies meanwhile here in the 3D case, the thumb either points inside or outside the geometry, creating a sense of symmetry.

Note that the unique orientations of the loops identified is just an observation, and it may not be true for some other origami structure. It is also unclear whether the BFS algorithm is the reason why such observations could be made. However, it was worth the attention, as, for the examples showcased, it captured the difference between the exterior and the interior space of the 3D entity by pointing the thumb in a particular direction which is essential for correctly considering the nodes for quantifying the dihedral angle in terms of the nodal positions (Section

4.2.2). In simple words, for any folding crease, there can be two different references for quantifying the dihedral angle, one can be from the inside of the geometry while the other can be from the outside. The fact that current BFS algorithm lists the panels with special orientations in the discussed cases, it gets easier to choose the same reference for formulating the dihedral angle for every folding crease in the origami which brings in a sense of consistency throughout the structure.

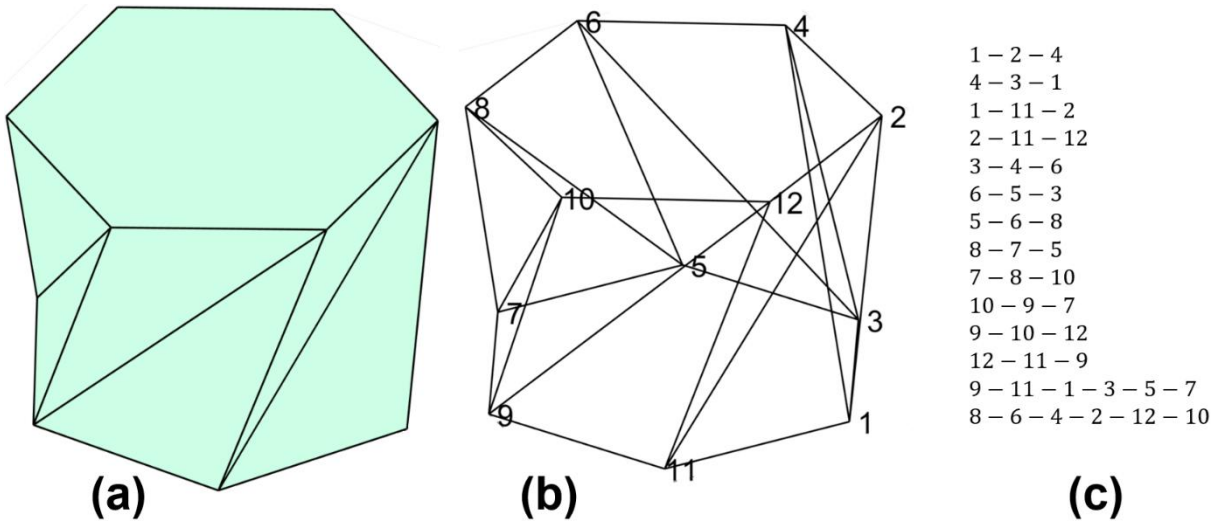


Fig. 21. Application of modified BFS algorithm in printing all the shortest cycles in a Kresling pattern: (a) A schematic of a Kresling origami (b) Recreation of the Kresling origami in MATLAB with node numbering (c) Identifying all 14 panels in the design including 12 triangles and 2 hexagons.

With the increment in the number of panels and hence, the growth of the tessellation, the number of edges and vertices naturally rise by a significant portion. Greater the number of nodes, higher is the time taken to scan them and list in the form of matrices (Section 3.1). At the same time, a rise in the number of edges automatically means an augmentation in the number of foldable creases too. Therefore, identifying the panels made out of these edges also demands larger computational cost. In simple words, computational complexity in this case increases with the geometrical complexity of the origami structure which may be attributed to the presence of

greater number of panels or even the number of connections a node has making the network complicated. This in turn increases the required time and resources for running the code which negatively influences the computational cost.

Therefore, a different method has been used for a 2D crease pattern (which is usually how an origami crease pattern exists) as the computational time is relatively very small in that case irrespective of the complexity in the crease pattern. However, this acts as a limitation as the code has to choose one of the two different techniques depending upon the origami design considered which is not as good as having a single equally efficient method to get the job done. Once the panels have been identified, more importantly, once it is known which panels have a length greater than three, next step is to discretize those panels into triangular pieces which in turn will generate bending creases for minimum bending energy and this is what the next section covers.

3.3. Discretizing the identified panels

As discussed in Section 2.2, to capture the bending of panels during the global folding process, we need to identify a set of bending creases dividing the panel into smaller ones so as to have smallest bending energy contribution. To do so, an algorithm has been developed that divides a geometry into two pieces separated by the shortest diagonal in the panel. This process is continued for the new panels generated until there is no panel left with a length greater than three. This procedure is detailed with figures as follows for a hexagonal panel to be discretized (Fig. 22).

For every node in the panel, the code first calculates the distance of every but the neighbouring nodes and stores the smallest one while avoiding any kind of repetition. For example, for node 1, distances from nodes 5, 4, and 3 are calculated and it turns out that among these, the distance from node 5 is the smallest, so that is stored in form of a matrix. The same is done for every

node while making sure that a distance is not calculated twice. For instance, when node 5 is under consideration, its distance from node 1 is already calculated, hence, it should not be obtained again. Doing so, a matrix is generated showcasing the smallest distances from each node to one another and it turns out that the distance between node 2 and 6 is the least. Therefore, this defines a new bending crease while dividing the hexagonal panel into a triangle and a pentagon. The matrix generated and the divided hexagon are shown in Fig. 23.

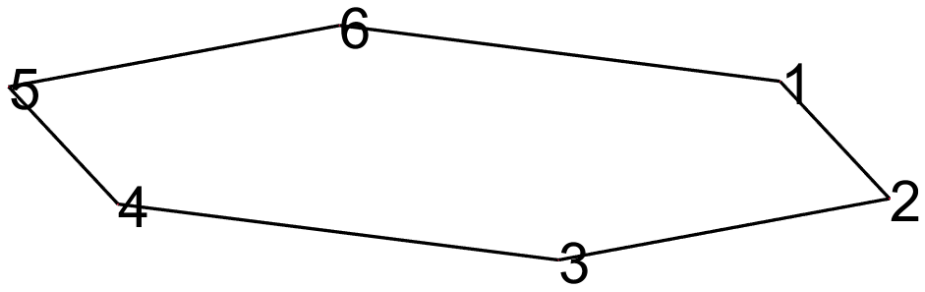
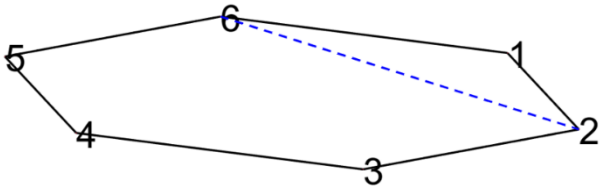


Fig. 22. A hexagonal panel to be discretized.



(a)

Node 1	Node 2	Distance
1	5	9.56e2
6	2	8.24e2
5	3	8.24e2
4	2	9.56e2

(b)

Fig. 23. (a) Hexagonal panel divided into a triangle and a pentagon separated by the shortest diagonal among the shortest diagonals per every node (b) A list of shortest distances from every node from which the smallest one is chosen.

The process is repeated for the pentagon which is divided into a quadrilateral and a triangle and then the quadrilateral is divided into two triangles as shown in Fig. 24. As a result, the hexagon gets discretized into four triangles and three bending creases which act as torsional bending springs are created. In this case, the bending creases are the diagonals $2-6, 6-3$, and $3-5$.

3.3.1. Generalized approach for discretization explained using pseudo code

As in earlier sections, the discretization scheme above was discussed based on a specific example which was a hexagonal panel. The approach can be generalized and the same has been discussed here based on the pseudo code developed. First, from the previous section, we have a matrix that contains all the loops that have been identified as panels in the origami. Note that the loops can be of different lengths and the matrix that stores them has the size: number of loops × number of nodes in largest loop in the origami. We first introduce a *while* loop that runs until we are out of loops to be discretized. At the same time, a pair matrix is defined which is a zero square matrix with size: number of nodes by number of nodes. This matrix is there to ensure that we do not repeat the calculation of distance calculation between two nodes. Now, we go over every loop that has a size greater than three one by one which is controlled by an *if* statement. *For* every such loop, we go over every node present in the loop (say, current node), and list all but the neighbouring nodes of that node which are stored in a vector termed as distant nodes.

Now, *for* every node in distant nodes (say, distant node), if the entry corresponding to (current node, distant node) in the pair matrix is zero, we proceed ahead and first make this and the value at (distant node, current node) equal to 1. This is followed by calculating the distance between these two nodes which is stored in a matrix (called distance matrix) along with these nodes. Once, these calculations for every distant node are completed, the row corresponding to the smallest distance in the distance matrix is identified and the two corresponding nodes are chosen to create a bending crease which divides the panel into two. As a result, out of the two new panels formed, the one with length greater than three is stacked back in the matrix containing the loops to be discretized and the cycle continues until all of the origami pattern has been divided into triangles.

```
loops to be discretized stored in form of a matrix (termed as loops to be discretized) from panel identification (Section 3.2)
while number of loops to be discretized > 0
   define a zero matrix of size: number of nodes × number of nodes (termed pair matrix)
   for every loop in loops to be discretized (termed as loop to discretize)
      if size(loop to discretize) > 3
         for every node in the loop to discretize (termed as current node)
            list all nodes apart from neighbouring nodes (termed as distant nodes)
               for every node in distant nodes (termed as distant node)
                  if pair matrix (current node, distant node) = 0
                     pair matrix (current node, distant node) = 1
                     pair matrix (distant node, current node) = 1
                     find the distance between current node and distant node
                     store the distance and these two nodes in a matrix (termed distance matrix)
                  end
               end
               find the row in distance matrix with smallest distance to identify discretizing crease
         end
         divide the panel under consideration into two panels based on the crease identified
         add the loops with length greater than three to loops to discretize while eliminating the
         original loop discretized
      end
   end
end
```

The algorithm so created is quite efficient and it is able to discretize complex geometries be it 2D or 3D. This is because once the panels have been identified, their orientation in the structure does not influence the discretization scheme as it is simply based on the positions of individual nodes in that panel. A working example of the same is shown in Fig. 25, where Fig. 25a is a

special kind of origami known as the Hypar origami and its discretized model is presented in Fig. 25b. Hypar origami's folding process physically includes distortion of panels as clear from the inset in Fig. 25b which includes a snapshot of a section of a partially folded physical model of the Hypar origami.

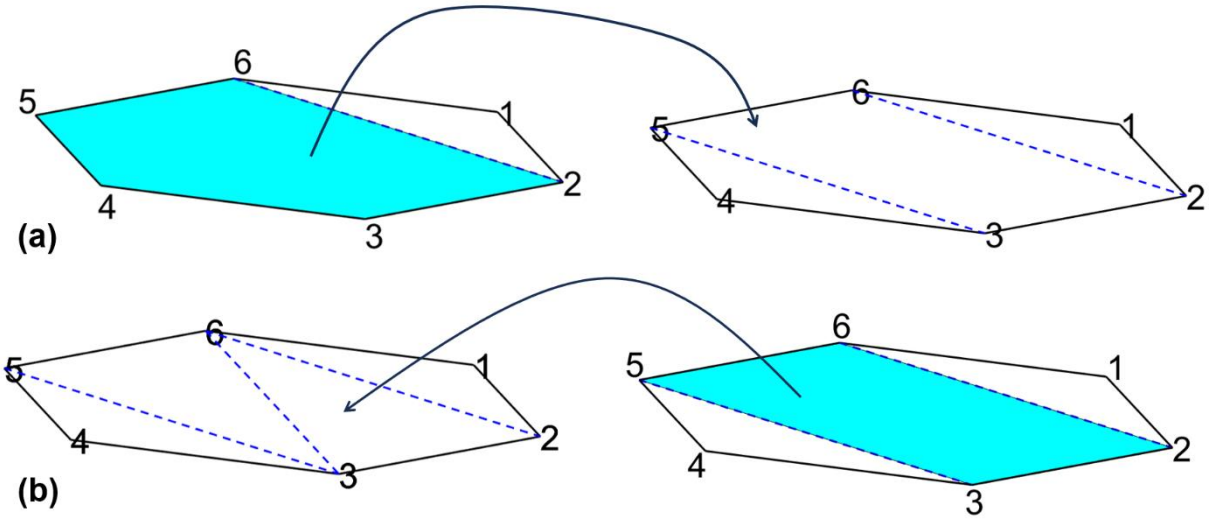


Fig. 24. (a) Pentagon that was a result of dividing the hexagon is further divided into a quadrilateral and a triangle (b) the quadrilateral is divided into two triangles ultimately dividing hexagon into four triangles.

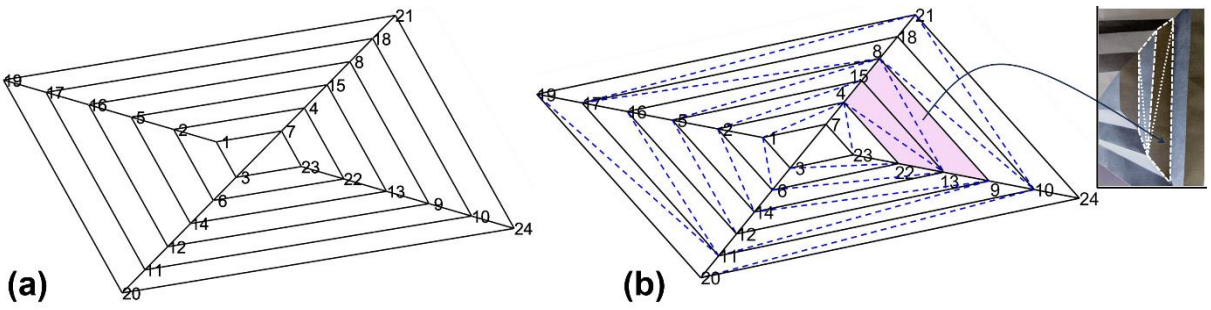


Fig. 25. (a) Hypar origami schematic (b) Discretized model of the Hypar origami (dashed blue-colored lines representing diagonals created) with triangular panels untouched while quadrilateral panels divided into triangles. Purple-colored panels correspond to the distorted panels shown in the inset presenting a physical model of Hypar origami.

A limitation of this technique is that it is not able to correctly discretize panels that are not planar. This might be the case when curved crease origami or simply a panel which is bent about an undefined diagonal is considered. In such cases, the nodes do not lie in the same plane and the algorithm might result in a diagonal that does not lie in any of the planes the panel lies in or even a worser scenario would result in a diagonal intersecting the panel itself. However, having a case in which a panel is bent about an undefined crease is rare and usually creases are defined to distinguish panels. As a matter of fact, 99% of the origami in the literature are simple, symmetric, and periodic with some even made up with the same kind of panels which are also regular.

3.4. Assembling elements for the bar and hinge model

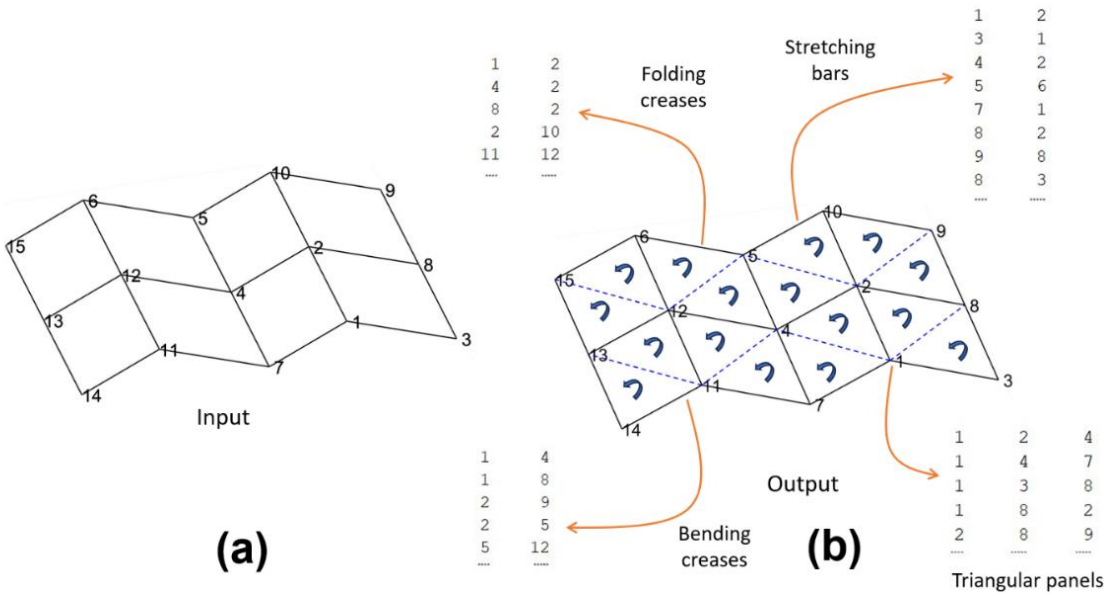


Fig. 26. Initial results of the code showcasing: (a) a simple user input i.e., a CAD file (b) the output as the key elements for the bar and hinge model.

As discussed in Section 2.2, we need to use the information imprinted so far to uncover the key elements to be employed in the bar and hinge model. These include the stretchable bars, torsional folding and bending springs. The stretchable creases can be easily extracted from the initial data generated during node numbering and the connections every node has with one

another. Recall that in the example discussed in Section 3.1, it was found that node 4 is connected to nodes {2,7,12,5}. This implies that the edges $4-2$, $4-7$, $4-12$, and $4-5$ act as stretchable bars. In this way while avoiding redundancy, the code is able to list all the edges acting as stretchable bars.

For figuring out the folding torsional springs, the data about stretchable bars is a starter. This is because every interior edge acts as a folding crease. Hence, we need to figure a way out through which the code is able to eliminate the boundary edges from the list of stretchable bars and finally list the folding springs. Recall from Section 3.2.1, to identify the panels, we started with a node and one of its connections. With that vector created, the next node was the one that lied at the largest ACW turn. This way the scheme was able to list all the shortest loops.

Surprisingly, the same technique can be used to identify the boundary loop. First step is to choose a node with the smallest number of connected neighbours as this node will lie on the boundary. In the example discussed in Section 3.2.1, in Fig. 17, such a node can be node 14, for instance. Now after choosing any of its connections to form a vector, instead of tracing to the node with the largest ACW rotation, chose the node that lies at the largest CW rotation. If there is no node on the CW turn side, then choose the node that lies at the smallest ACW rotation and continue the procedure to ultimately print the boundary loop.

In the case being discussed, if one chooses node 11 as the connection of node 14 to form the vector from node 14 to node 11, the next node based on the steps discussed above will be node 7 followed by node 1, node 3, node 8 and so on. In short, one will trace out the loop $14-11-7-1-3-8-9-10-5-6-15-13$ which is the boundary loop made up from the boundary edges. These edges can be erased from the list of stretchable bars to list the folding torsional springs.

Finally, for identifying the bending creases, the approach discussed in Section 3.3 suffices as bending creases are a byproduct of the procedure presented. With the panels discretized, bending creases are automatically created which are none other than the bending torsional springs. For example, in Fig. 24, the edges $2-6, 6-3$, and $3-5$ are the bending creases.

As a result, the code developed so far takes a very simple user input and is able to list all the elements for the practical use of the bar and hinge model as depicted in Fig. 26.

4. QUANTIFYING THE PHYSICAL PROCESS

4.1. Deriving the energy functional

After identifying the elements, to make the bar and hinge model work, we need a framework that translates the physical origami folding process as a result of load application or due to a change in the degree of some external stimulus. During the deformation process, apart from the work done by external forces, there is a resistance offered by the bars and the springs in the reduced-order model. This is reflected in the expression of total energy (U_T) (Equation 4) where the contribution of these elements is taken opposite to that of the external loadings.

$$U_T = U_{str} + U_{rot} - W \tag{4}$$

Here, U_{str} and U_{rot} are the total stretching and rotational energy, meanwhile W is the work done due to external forces. Substituting from Section 2.2, Equations 1 and 3, and knowing that external forces are applied on the nodes, Equation 4 can be rewritten as:

$$U_T = \sum_i \frac{1}{2} k_{str}^i s_i^2 + \sum_j \frac{1}{2} k_{rot}^j \left(\theta_j - \theta_{0_j}\right)^2 - \sum_n \overrightarrow{F_n} \cdot \overrightarrow{dx_n} \tag{5}$$

To restate, in Equation 5, $k_{str}, s, k_{rot}, \theta, \theta_0, \vec{F}$, and $\overrightarrow{dx}$ are the stretching stiffness, change in the length of the stretchable bar, torsional spring stiffness, instantaneous dihedral angle, preferred dihedral angle, applied external force on a node, and the displacement of that node, respectively. Note that the expressions for each of the energies and work done are summed over different elements (denoted by i, j, and n). For instance, the first expression on the right-hand side of Equation 5 is the total stretching energy which is summed over every stretchable bar identified in Section 3 and i denotes all such bars. Similarly, the second formulation depicts the total torsional spring energy which depending upon the value of k_{rot} i.e., either k_B representing the bending crease or k_F implying the folding crease, is summed up over all the bending creases and the folding creases, respectively. Together, these folding and bending creases are

represented by the index j. Finally, the work done W is also summed up over all the nodes on which the external load is being applied which is also a part of the user input and these nodes are identified using the index n.

Now, Equation 5 depicts the total energy of a reduced-order model of an origami structure at a point in its deformation process and in order to attain equilibrium which is a step towards finding out the stable states of that origami structure, the total energy functional needs to be minimized. However, the expression is a function of many variables which include the change in the length of the bars, deviation from preferred dihedral angle, and displacement of nodes. In order to be more systematic and to be able to solve for equilibrium, it is crucial to write down the total energy as a function of nodal positions which can be minimized to figure out the new nodal positions corresponding to mechanical equilibrium of the structure, and this is what the next section covers.

4.2. Total energy as a natural function of nodal positions

4.2.1. Stretching energy

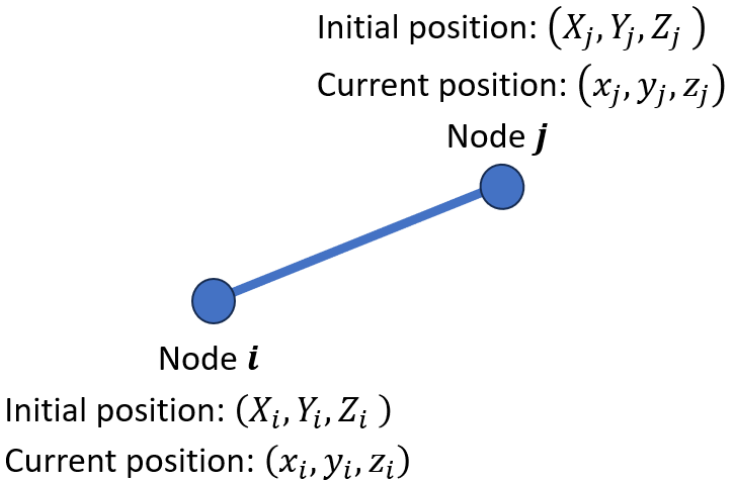


Fig. 27. A schematic of a stretchable bar with nodes i and j at its ends used to formulate change in its length as a function of nodal positions.

In the first expression of Equation 5, the term corresponding to change in a bar's length s can be written in terms of the positions of the nodes at the two ends of that bar. As shown in Fig. 27, the nodes at the two ends of the bar have an initial and an instantaneous position. Hence, initial, final, and the change in length of the bar can be written as:

$$l_{initial} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}$$

$$l_{final} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

$$s = l_{final} - l_{initial} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2} \quad (6)$$

4.2.2. Torsional spring energy

Therefore, for any given bar, depending on the nodes it is surrounded by, s can be easily formulated. Now, in the second expression of Equation 5, to express the dihedral angle θ as a function of nodal positions, a unique relation needs to be established. Fig. 28 shows a general crease about which two planes P1 and P2 lie, and they create a dihedral angle θ. Note that 4 nodes are required to define these two planes in relation to a common crease. Using these 4 nodes i.e., nodes a, b, c and d, one can define a vector $\vec{r}_{ac}$ along the common crease and two other vectors $\vec{r}_{dc}$ and $\vec{r}_{ab}$ along the edges of the planes P2 and P1, respectively. Taking the cross-product of $\vec{r}_{dc}$ and $\vec{r}_{ac}$ leads to a vector normal to the plane P2 such that $\vec{n}_{acd} = \vec{r}_{dc} \times \vec{r}_{ac}$ and similarly, $\vec{n}_{abc} = \vec{r}_{ac} \times \vec{r}_{ab}$ is the normal vector to plane P1. Note that instead of the two vectors chosen and shown on planes P1 ($\vec{r}_{ab}$) and P2 ($\vec{r}_{dc}$) in Fig. 28, one can choose any other vector in a plane. For instance, if instead of $\vec{r}_{dc}$, $\vec{r}_{da}$ is considered, taking its cross-product with $\vec{r}_{ac}$ would still result in a vector parallel to $\vec{n}_{acd}$. No matter what combinations are made, one would

still require the four nodes to describe the dihedral angle θ. This is because, interestingly, the dot product of the two normal vectors can be expressed in terms of the dihedral angle θ as:

$$\cos\theta = \frac{\vec{n}_{abc} \cdot \vec{n}_{acd}}{|\vec{n}_{abc}||\vec{n}_{acd}|}$$

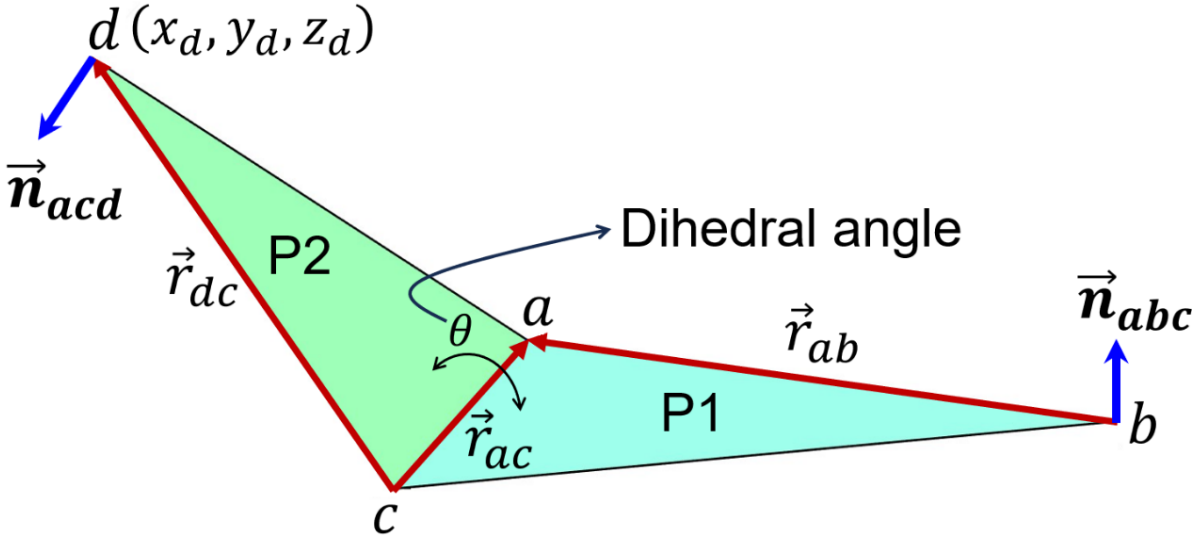


Fig. 28. A schematic of a crease separating two planes generating a dihedral angle calculated using the vectors in the two planes.

However, once we take the cos inverse to obtain θ, the angle always lies between 0 and π, given the range of the function $y = \cos^{-1} x$. There can be cases in which the expression above spits out the same dihedral angle, but their physical representation may uncover different relative orientations of the two planes. For instance, if there are two scenarios: in one case, the planes P1 and P2 differ by a dihedral angle of $120°$ whereas, in the other case, they behold a dihedral angle of $240°$. In both the cases, the expression derived will compute an angle of $120°$. In order to eliminate this issue, we can slightly modify the expression to have [23]:

$$\theta = \left[\zeta \cos^{-1}\left(\frac{\vec{n}_{abc} \cdot \vec{n}_{acd}}{|\vec{n}_{abc}||\vec{n}_{acd}|}\right)\right] \mathrm{mod} 2\pi \quad (7)$$

Here, ζ is a sign indicator such that $\zeta = sgn(\vec{n}_{acd} \cdot \vec{r}_{ab})$ when $\vec{n}_{acd} \cdot \vec{r}_{ab} \neq 0$ otherwise $\zeta = 1$. The operator 'mod' represents the remainder when a quantity is divided by another. Here, θ

turns out to be the remainder when ζ multiplied by the arccos expression is divided by 2π. At the end, as the edge and the crease vectors in the schematic i.e., Fig. 28 can be written in terms of the positions of the nodes, the dihedral angle θ has been expressed in terms of the same.

4.3. Solving for mechanical equilibrium – complete general theory at one place

With all the expressions in Equation 5 discussed, it is clear that the total energy U_T is a function of all positions of all the nodes or in a professional way, a function of all the DOF of the system. Now, in order to solve for equilibrium, we use the Newton-Raphson method which is also known as Newton's method and is widely used as a root-finding algorithm [26]. This section details on how to use the Newton's method specifically in order to find the equilibrium states of an origami. With U_T defined as a function of nodal positions, we can go ahead and minimize it. However, with so many DOF, how exactly does the procedure look like, is a question to be answered in this section. Firstly, U_T is differentiated with respect to every DOF one by one to generate a vector which is known as the residual force vector, say, $\vec{R}$ with a size equal to the number of DOF. As we are dealing in a cartesian coordinate system and if we consider the displacements of every node along the three directions X, Y, and Z as the DOF, the residual will take a form shown below:

$$\vec{R} = \left(\frac{\partial U_T}{\partial u_x^1}, \frac{\partial U_T}{\partial u_y^1}, \frac{\partial U_T}{\partial u_z^1}, \frac{\partial U_T}{\partial u_x^2}, \frac{\partial U_T}{\partial u_y^2}, \frac{\partial U_T}{\partial u_z^2}, \frac{\partial U_T}{\partial u_x^3}, \dots\dots\dots, \frac{\partial U_T}{\partial u_z^p}\right) \tag{8}$$

In Equation 8, the quantities $u_x^1, u_y^1, u_z^1, u_x^2, u_y^2, \dots$ are the individual DOF of the reduced-order system and also the variables on which $\vec{R}$ depends. Following this, as one usually does while minimizing, our goal is to equate $\vec{R}$ to zero, i.e., we want $\vec{R} = 0$. Doing so, we will obtain those special set of nodal positions that correspond to a zero residual force or equilibrium. However, $\vec{R}$ is a complicated function and solving directly for it is extremely difficult. Therefore, we Taylor expand the expression as follows:

$$\vec{R}|_{\vec{u}} = \vec{R}|_{\vec{u}_0} + \frac{\partial \vec{R}}{\partial \vec{u}}|_{\vec{u}_0} \cdot (\vec{u} - \vec{u}_0) \tag{9}$$

On the left-hand side in Equation 9, $\vec{u}$ represents the current values of the set of the DOF, $\vec{R}$ is a function of, meanwhile, the first expression on the right-hand side showcases the value of $\vec{R}$ at a particular set of values of these DOF (usually a guess value or zero) $\vec{u}_0$. The second expression on the right-hand side is the product of two quantities. The first quantity being the derivative of the residual force with respect to the DOF calculated at $\vec{u}_0$ and the second quantity is the difference between the current and the initial considered values of the DOF.

Taylor expansion is also known as linearly expanding a complex function and doing so, we are able to write down $\vec{R}$ as a linear function and convert it to a state when equating it to zero and solving the resulting equation is easier. Doing so, we obtain the following relationship:

$$(\vec{u} - \vec{u}_0) = -\left(\frac{\partial \vec{R}}{\partial \vec{u}}|_{\vec{u}_0}\right)^{-1} \cdot \left(\vec{R}|_{\vec{u}_0}\right) \tag{10}$$

From the above expression (Equation 10), we attain the new values of $\vec{u}$ that take the system that was initially at $\vec{u}_0$, to equilibrium. With this, the only quantity that needs to be obtained is the derivative of the residual force i.e., $\frac{\partial \vec{R}}{\partial \vec{u}}|_{\vec{u}_0}$. As this quantity is a derivative of a vector with respect to a vector, it results in a matrix which is known as the tangent stiffness matrix (K) that is of size $3p \times 3p$, where p is the number of nodes in the origami system. K can be represented as:

$$K = \begin{bmatrix} \frac{\partial^2 U_T}{\partial u_x^{1\,2}} & \frac{\partial^2 U_T}{\partial u_y^1 \partial u_x^1} & \frac{\partial^2 U_T}{\partial u_z^1 \partial u_x^1} & \cdots & \frac{\partial^2 U_T}{\partial u_z^p \partial u_x^1} \\ \frac{\partial^2 U_T}{\partial u_x^1 \partial u_y^1} & \frac{\partial^2 U_T}{\partial u_y^{1\,2}} & \frac{\partial^2 U_T}{\partial u_z^1 \partial u_y^1} & & \frac{\partial U_T}{\partial u_z^p \partial u_y^1} \\ & \vdots & & \ddots & \vdots \\ \frac{\partial^2 U_T}{\partial u_x^1 \partial u_z^p} & \frac{\partial^2 U_T}{\partial u_y^1 \partial u_z^p} & \frac{\partial^2 U_T}{\partial u_z^1 \partial u_z^p} & \cdots & \frac{\partial^2 U_T}{\partial u_z^{p\,2}} \end{bmatrix} \tag{11}$$

4.4. Efficiently translating equilibrium theory to the numerical model

Now, with the expressions developed for total energy U_T, it is straightforward to go ahead and code them to write a set of lines for calculating the single and double derivatives to obtain the residual force and tangent stiffness matrix which can be solved like a usual system of linear algebraic equations (Equation 10) to get the state of equilibrium. However, computationally this is a very expensive approach and therefore, we compute analytical expressions for the derivatives and input them in the code due to which the computational time falls significantly.

4.4.1. Analytical expressions for the components of the residual force vector

4.4.1.1. Derivative of the stretching energy

We are now aware that the stretching energy is given by the expression $U_{str} = \frac{1}{2} k_{str} s^2$ and $s = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}$ for a bar element formed by nodes i and j, from Equations 6 and 1, respectively. This means that the stretching energy corresponding to this bar is a function of the position of the nodes i and j which is a generalised formulation. Therefore, a section of the residual force vector due to the contribution of the stretching energy of this general bar can be obtained by taking the derivative of U_{str} with respect to the 6 DOF which are x_i, y_i, z_i, x_j, y_j, and z_j which is given by:

$$\left[\frac{\partial U_{str}}{\partial x_i}, \frac{\partial U_{str}}{\partial y_i}, \frac{\partial U_{str}}{\partial z_i}, \frac{\partial U_{str}}{\partial x_j}, \frac{\partial U_{str}}{\partial y_j}, \frac{\partial U_{str}}{\partial z_j}\right]$$

The expression for U_{str} can be substituted and the above vector can be restated as:

$$k_{str} s \left[\frac{\partial s}{\partial x_i}, \frac{\partial s}{\partial y_i}, \frac{\partial s}{\partial z_i}, \frac{\partial s}{\partial x_j}, \frac{\partial s}{\partial y_j}, \frac{\partial s}{\partial z_j}\right] \tag{12}$$

Calculating the derivatives of the change in length of the bar with respect to the mentioned DOF and substituting them in the Equation 12 above results in:

$$k_{str}s\left[\frac{\Delta x}{l_{final}}, \frac{\Delta y}{l_{final}}, \frac{\Delta z}{l_{final}}, \frac{-\Delta x}{l_{final}}, \frac{-\Delta y}{l_{final}}, \frac{-\Delta z}{l_{final}}\right] \tag{13}$$

Here, $l_{final} = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2}$ is the final length of the bar in which $\Delta x = x_i - x_j, \Delta y = y_i - y_j$, and $\Delta z = z_i - z_j$. The global residual vector is a large one and contains the derivative with respect to every DOF, therefore, the derivatives calculated above take their places in $\vec{R}$ based on the nodes i and j. For instance, the derivative $\frac{\partial U_{str}}{\partial x_1}$ will be placed in the first column of the row vector $\vec{R}$, meanwhile $\frac{\partial U_{str}}{\partial y_1}$ in the second, $\frac{\partial U_{str}}{\partial z_1}$ in the third, $\frac{\partial U_{str}}{\partial y_2}$ in the fifth and so on. Therefore, the contribution due to some node m i.e., the derivatives $\frac{\partial U_{str}}{\partial x_m}, \frac{\partial U_{str}}{\partial y_m}$, and $\frac{\partial U_{str}}{\partial z_m}$ will be placed from column number $3m - 2$ to $3m$ in $\vec{R}$. Hence, we have derived the analytical expressions for the residual force corresponding to the stretching energy U_{str}. Now, we focus on the formulations that can be developed for the contribution from the torsional spring energy.

4.4.1.2. Derivative of the rotational spring energy

Note that the rotational energy that includes both folding and bending torsional spring energy is dependent on the dihedral angle θ which is related to the nodal positions of four vertices that are required to form the relevant crease (see Section 4.2.2). Therefore, one can take the derivatives with respect to those nodal positions and depending upon the folding or the bending crease under consideration, the contributions can be added to the corresponding place in the residual vector, as detailed here.

The rotational spring energy is given by $U_{rot} = \frac{1}{2}k_{rot}(\theta - \theta_0)^2$, and $\theta = \left[\zeta \cos^{-1}\left(\frac{\vec{n}_{abc}\cdot\vec{n}_{acd}}{|\vec{n}_{abc}||\vec{n}_{acd}|}\right)\right] \mathrm{mod} 2\pi$, where the vectors $\vec{n}_{abc}$ and $\vec{n}_{acd}$ are the normal vectors shown in Fig. 28 which also showcases the nodes a, b, c and d. Therefore, we can obtain the derivative of U_{rot} with respect to all the DOF related to these 4 nodes which will result in a vector shown in Equation 14 and that can be further reduced to Equation 15:

$$\left[\frac{\partial U_{rot}}{\partial x_a}, \frac{\partial U_{rot}}{\partial y_a}, \frac{\partial U_{rot}}{\partial z_a}, \frac{\partial U_{rot}}{\partial x_b}, \frac{\partial U_{rot}}{\partial y_b}, \frac{\partial U_{rot}}{\partial z_b}, \frac{\partial U_{rot}}{\partial x_c}, \frac{\partial U_{rot}}{\partial y_c}, \frac{\partial U_{rot}}{\partial z_c}, \frac{\partial U_{rot}}{\partial x_d}, \frac{\partial U_{rot}}{\partial y_d}, \frac{\partial U_{rot}}{\partial z_d}\right] \tag{14}$$

$$k_{rot}\left[\frac{\partial \theta}{\partial x_a}, \frac{\partial \theta}{\partial y_a}, \frac{\partial \theta}{\partial z_a}, \frac{\partial \theta}{\partial x_b}, \frac{\partial \theta}{\partial y_b}, \frac{\partial \theta}{\partial z_b}, \frac{\partial \theta}{\partial x_c}, \frac{\partial \theta}{\partial y_c}, \frac{\partial \theta}{\partial z_c}, \frac{\partial \theta}{\partial x_d}, \frac{\partial \theta}{\partial y_d}, \frac{\partial \theta}{\partial z_d}\right] \tag{15}$$

These derivatives above can be expressed in terms of the nodal positions of nodes a, b, c and d as follows [23]:

$$\begin{bmatrix} \frac{\partial \theta}{\partial x_d} \\ \frac{\partial \theta}{\partial y_d} \\ \frac{\partial \theta}{\partial z_d} \end{bmatrix} = \frac{|\vec{r}_{ac}|}{|\vec{n}_{acd}|^2} \begin{bmatrix} \left((y_d - y_c)(z_a - z_c)\right) - \left((z_d - z_c)(y_a - y_c)\right) \\ \left((z_d - z_c)(x_a - x_c)\right) - \left((x_d - x_c)(z_a - z_c)\right) \\ \left((x_d - x_c)(y_a - y_c)\right) - \left((x_a - x_c)(y_d - y_c)\right) \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \theta}{\partial x_b} \\ \frac{\partial \theta}{\partial y_b} \\ \frac{\partial \theta}{\partial z_b} \end{bmatrix} = -\frac{|\vec{r}_{ac}|}{|\vec{n}_{abc}|^2} \begin{bmatrix} \left((y_a - y_c)(z_a - z_b)\right) - \left((z_a - z_c)(y_a - y_b)\right) \\ \left((z_a - z_c)(x_a - x_b)\right) - \left((x_a - x_c)(z_a - z_b)\right) \\ \left((x_a - x_c)(y_a - y_b)\right) - \left((x_a - x_b)(y_a - y_c)\right) \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \theta}{\partial x_c} \\ \frac{\partial \theta}{\partial y_c} \\ \frac{\partial \theta}{\partial z_c} \end{bmatrix} = \left(\frac{\vec{r}_{dc} \cdot \vec{r}_{ac}}{|\vec{r}_{ac}|^2} - 1\right) \begin{bmatrix} \frac{\partial \theta}{\partial x_d} \\ \frac{\partial \theta}{\partial y_d} \\ \frac{\partial \theta}{\partial z_d} \end{bmatrix} + \left(\frac{\vec{r}_{ab} \cdot \vec{r}_{ac}}{|\vec{r}_{ac}|^2}\right) \begin{bmatrix} \frac{\partial \theta}{\partial x_b} \\ \frac{\partial \theta}{\partial y_b} \\ \frac{\partial \theta}{\partial z_b} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \theta}{\partial x_a} \\ \frac{\partial \theta}{\partial y_a} \\ \frac{\partial \theta}{\partial a} \end{bmatrix} = \left(1 - \frac{\vec{r}_{ab} \cdot \vec{r}_{ac}}{|\vec{r}_{ac}|^2}\right) \begin{bmatrix} \frac{\partial \theta}{\partial x_b} \\ \frac{\partial \theta}{\partial y_b} \\ \frac{\partial \theta}{\partial z_b} \end{bmatrix} - \left(\frac{\vec{r}_{dc} \cdot \vec{r}_{ac}}{|\vec{r}_{ac}|^2}\right) \begin{bmatrix} \frac{\partial \theta}{\partial x_d} \\ \frac{\partial \theta}{\partial y_d} \\ \frac{\partial \theta}{\partial z_d} \end{bmatrix}$$

As in the earlier section, these derivatives can be placed in the residual force vector in the same way as the derivatives of the stretching energy were placed. Note that these expressions are

quite complex but as they are analytical in nature, they tend to decrease the computation time appreciably.

With the contribution of rotational energies discussed, we are left with the last term in the expression for total energy U_T (Equation 5) which is the work done due to the external forces W. It is straightforward to observe that effect of W on $\vec{R}$ is simply that the relevant elements in $\vec{R}$ are deducted by the force applied along those DOF as the derivative of W with respect to the DOF under consideration is simply the component of the force applied $\vec{F}$ in that direction. For example, say, a load $\vec{L} = (F_0, 0, -3F_0)$ is applied to node 5 in the reduced-order model of an origami, then as per the established convention, column numbers 13 and 15 are added by quantities $-F_0$ and $3F_0$, respectively.

4.4.2. Analytical expressions for elements in the tangent stiffness matrix

4.4.2.1. Contribution of stretching energy

From the tangent stiffness matrix shown in Equation 11, it can be seen that every element contains the double derivative of the energy with respect to the pairs of every DOF. For instance, the element in the first row second column is $\frac{\partial^2 U_T}{\partial u_y^1 \partial u_x^1}$ which is a derivative with respect to the DOF u_y^1 and u_x^1. Now, focusing on the stretching energy which is influenced by the stretch of a bar s that is a function of the nodal positions of the nodes at the two ends of the bar (nodes i and j in a generalised bar), the energy can be differentiated based on every possible pair of the DOF which are x_i, y_i, z_i, x_j, y_j, and z_j which will result in a total of 6×6 i.e., 36 pairs. Once calculated, these can then be accordingly placed in the global tangent stiffness matrix. Here is how one can obtain these derivatives:

Knowing that $U_{str} = \frac{1}{2} k_{str} s^2$, starting with the first one, i.e., $\frac{\partial^2 U_{str}}{\partial x_i \partial x_i}$ or $\frac{\partial^2 U_{str}}{\partial x_i^2}$. As $\frac{\partial U_{str}}{\partial x_i} = \frac{ks\Delta x}{l_{final}}$ (Equation 13):

$$\frac{\partial^2 U_{str}}{\partial x_i^2} = \frac{\partial}{\partial x_i}\left(\frac{\partial U_{str}}{\partial x_i}\right) = \frac{\partial}{\partial x_i}\left(\frac{k_{str} s \Delta x}{l_{final}}\right)$$

$$= \frac{\partial}{\partial x_i}\left(\frac{k_{str}\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}\right)(x_i - x_j)}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}\right)$$

This derivative can be solved using a combination of the product, quotient and chain rules as shown below:

$$\frac{\partial^2 U_{str}}{\partial x_i^2} = k_{str}\left[\frac{\partial}{\partial x_i}\left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right.\right.$$

$$\left. - \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}\right)\left(\frac{x_i - x_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}\right)$$

$$+ \left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right.$$

$$\left.\left. - \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}\right)\frac{\partial}{\partial x_i}\left(\frac{x_i - x_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}\right)\right]$$

$$\frac{1}{k_{str}}\frac{\partial^2 U_{str}}{\partial x_i^2}$$

$$= \left(\frac{x_i - x_j}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}\right)^2$$

$$+ \left(\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right.$$

$$\left. - \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2 + (Z_i - Z_j)^2}\right)\left[\frac{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} - \frac{(x_i - x_j)^2}{\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}}}{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\right]$$

The above expression can be simplified and written in terms of the stretch (s) and final length (l_{final}) of the bar:

$$\frac{\partial^2 U_{str}}{\partial x_i^2} = \frac{k_{str}}{l_{final}^3}\left[(l_{final} - s)(\Delta x)^2 + s l_{final}^2\right]$$

Similarly, other 35 double derivatives can be obtained and all of them are listed below:

$$\frac{\partial^2 U_{str}}{\partial y_i \partial x_i} = \frac{k_{str}}{l_{final}^3}\left[(l_{final} - s)\Delta x \Delta y\right], \quad \frac{\partial^2 U_{str}}{\partial z_i \partial x_i} = \frac{k_{str}}{l_{final}^3}\left[(l_{final} - s)\Delta z \Delta x\right], \quad \frac{\partial^2 U_{str}}{\partial x_j \partial x_i} = -\frac{\partial^2 U_{str}}{\partial x_i^2},$$

$$\frac{\partial^2 U_{str}}{\partial y_j \partial x_i} = -\frac{\partial^2 U_{str}}{\partial y_i \partial x_i}, \frac{\partial^2 U_{str}}{\partial z_j \partial x_i} = -\frac{\partial^2 U_{str}}{\partial z_i \partial x_i}, \frac{\partial^2 U_{str}}{\partial x_i \partial y_i} = \frac{\partial^2 U_{str}}{\partial y_i \partial x_i}, \frac{\partial^2 U_{str}}{\partial y_i^2} = \frac{k_{str}}{l_{final}^3}\left[(l_{final} - s)\Delta y^2 + s l_{final}^2\right],$$

$$\frac{\partial^2 U_{str}}{\partial z_i \partial y_i} = \frac{k_{str}}{l_{final}^3}\left[(l_{final} - s)\Delta z \Delta y\right], \quad \frac{\partial^2 U_{str}}{\partial x_j \partial y_i} = \frac{\partial^2 U_{str}}{\partial y_j \partial x_i}, \quad \frac{\partial^2 U_{str}}{\partial y_j \partial y_i} = -\frac{\partial^2 U_{str}}{\partial y_i^2}, \quad \frac{\partial^2 U_{str}}{\partial z_j \partial y_i} = -\frac{\partial^2 U_{str}}{\partial z_i \partial y_i},$$

$$\frac{\partial^2 U_{str}}{\partial x_i \partial z_i} = \frac{\partial^2 U_{str}}{\partial z_i \partial x_i}, \frac{\partial^2 U_{str}}{\partial y_i \partial z_i} = \frac{\partial^2 U_{str}}{\partial z_i \partial y_i}, \frac{\partial^2 U_{str}}{\partial z_i^2} = \frac{k_{str}}{l_{final}^3}\left[(l_{final} - s)\Delta z^2 + s l_{final}^2\right], \frac{\partial^2 U_{str}}{\partial x_j \partial z_i} = -\frac{\partial^2 U_{str}}{\partial z_i \partial x_i},$$

$$\frac{\partial^2 U_{str}}{\partial y_j \partial z_i} = -\frac{\partial^2 U_{str}}{\partial z_i \partial y_i}, \frac{\partial^2 U_{str}}{\partial z_j \partial z_i} = -\frac{\partial^2 U_{str}}{\partial z_i^2}, \frac{\partial^2 U_{str}}{\partial x_i \partial x_j} = \frac{\partial^2 U_{str}}{\partial x_j \partial x_i}, \frac{\partial^2 U_{str}}{\partial y_i \partial x_j} = \frac{\partial^2 U_{str}}{\partial x_j \partial y_i}, \frac{\partial^2 U_{str}}{\partial z_i \partial x_j} = \frac{\partial^2 U_{str}}{\partial x_j \partial z_i}, \frac{\partial^2 U_{str}}{\partial x_j^2} =$$

$$\frac{\partial^2 U_{str}}{\partial x_i^2}, \frac{\partial^2 U_{str}}{\partial y_j \partial x_j} = \frac{\partial^2 U_{str}}{\partial y_i \partial x_i}, \frac{\partial^2 U_{str}}{\partial z_j \partial x_j} = \frac{\partial^2 U_{str}}{\partial z_i \partial x_i}, \frac{\partial^2 U_{str}}{\partial x_i \partial y_j} = \frac{\partial^2 U_{str}}{\partial y_j \partial x_i}, \frac{\partial^2 U_{str}}{\partial y_i \partial y_j} = \frac{\partial^2 U_{str}}{\partial y_j \partial y_i}, \frac{\partial^2 U_{str}}{\partial z_i \partial y_j} = \frac{\partial^2 U_{str}}{\partial y_j \partial z_i},$$

$$\frac{\partial^2 U_{str}}{\partial x_j \partial y_j} = \frac{\partial^2 U_{str}}{\partial y_j \partial x_j}, \frac{\partial^2 U_{str}}{\partial y_j^2} = \frac{\partial^2 U_{str}}{\partial y_i^2}, \frac{\partial^2 U_{str}}{\partial z_j \partial y_j} = -\frac{\partial^2 U_{str}}{\partial z_j \partial y_i}, \frac{\partial^2 U_{str}}{\partial x_i \partial z_j} = \frac{\partial^2 U_{str}}{\partial z_j \partial x_i}, \frac{\partial^2 U_{str}}{\partial y_i \partial z_j} = \frac{\partial^2 U_{str}}{\partial z_j \partial y_i}, \frac{\partial^2 U_{str}}{\partial z_i \partial z_j} =$$

$$\frac{\partial^2 U_{str}}{\partial z_j \partial z_i}, \frac{\partial^2 U_{str}}{\partial x_j \partial z_j} = \frac{\partial^2 U_{str}}{\partial z_j \partial x_j}, \frac{\partial^2 U_{str}}{\partial y_j \partial z_j} = \frac{\partial^2 U_{str}}{\partial z_j \partial y_j}, \frac{\partial^2 U_{str}}{\partial z_j^2} = -\frac{\partial^2 U_{str}}{\partial z_j \partial z_i}$$

In order to figure out where these double derivatives are placed in the global tangent stiffness matrix, we assemble them in a unique way to uncover that the arrangements of specific set of these derivatives form four mini-matrices (of size 3×3 each covering all the 36 elements) that are directly related to particular regions in the global tangent stiffness matrix. These four matrices are shown here:

$$H_1 = \begin{bmatrix} \frac{\partial^2 U_{str}}{\partial x_i^2} & \frac{\partial^2 U_{str}}{\partial y_i \partial x_i} & \frac{\partial^2 U_{str}}{\partial z_i \partial x_i} \\ \frac{\partial^2 U_{str}}{\partial x_i \partial y_i} & \frac{\partial^2 U_{str}}{\partial y_i^2} & \frac{\partial^2 U_{str}}{\partial z_i \partial y_i} \\ \frac{\partial^2 U_{str}}{\partial x_i \partial z_i} & \frac{\partial^2 U_{str}}{\partial y_i \partial z_i} & \frac{\partial^2 U_{str}}{\partial z_i^2} \end{bmatrix}, H_2 = \begin{bmatrix} \frac{\partial^2 U_{str}}{\partial x_j \partial x_i} & \frac{\partial^2 U_{str}}{\partial y_j \partial x_i} & \frac{\partial^2 U_{str}}{\partial z_j \partial x_i} \\ \frac{\partial^2 U_{str}}{\partial x_j \partial y_i} & \frac{\partial^2 U_{str}}{\partial y_j \partial y_i} & \frac{\partial^2 U_{str}}{\partial z_j \partial y_i} \\ \frac{\partial^2 U_{str}}{\partial x_j \partial z_i} & \frac{\partial^2 U_{str}}{\partial y_j \partial z_i} & \frac{\partial^2 U_{str}}{\partial z_j \partial z_i} \end{bmatrix}$$

$$H_3 = \begin{bmatrix} \frac{\partial^2 U_{str}}{\partial x_i \partial x_j} & \frac{\partial^2 U_{str}}{\partial y_i \partial x_j} & \frac{\partial^2 U_{str}}{\partial z_i \partial x_j} \\ \frac{\partial^2 U_{str}}{\partial x_i \partial y_j} & \frac{\partial^2 U_{str}}{\partial y_i \partial y_j} & \frac{\partial^2 U_{str}}{\partial z_i \partial y_j} \\ \frac{\partial^2 U_{str}}{\partial x_i \partial z_j} & \frac{\partial^2 U_{str}}{\partial y_i \partial z_j} & \frac{\partial^2 U_{str}}{\partial z_i \partial z_j} \end{bmatrix}, H_4 = \begin{bmatrix} \frac{\partial^2 U_{str}}{\partial x_j^2} & \frac{\partial^2 U_{str}}{\partial y_j \partial x_j} & \frac{\partial^2 U_{str}}{\partial z_j \partial x_j} \\ \frac{\partial^2 U_{str}}{\partial x_j \partial y_j} & \frac{\partial^2 U_{str}}{\partial y_j^2} & \frac{\partial^2 U_{str}}{\partial z_j \partial y_j} \\ \frac{\partial^2 U_{str}}{\partial x_j \partial z_j} & \frac{\partial^2 U_{str}}{\partial y_j \partial z_j} & \frac{\partial^2 U_{str}}{\partial z_j^2} \end{bmatrix}$$

Note that the four matrices H_1, H_2, H_3, and H_4 take the general form H as shown below:

$$H = \begin{bmatrix} \frac{\partial^2 U_{str}}{\partial x_n \partial x_m} & \frac{\partial^2 U_{str}}{\partial y_n \partial x_m} & \frac{\partial^2 U_{str}}{\partial z_n \partial x_m} \\ \frac{\partial^2 U_{str}}{\partial x_n \partial y_m} & \frac{\partial^2 U_{str}}{\partial y_n \partial y_m} & \frac{\partial^2 U_{str}}{\partial z_n \partial y_m} \\ \frac{\partial^2 U_{str}}{\partial x_n \partial z_m} & \frac{\partial^2 U_{str}}{\partial y_n \partial z_m} & \frac{\partial^2 U_{str}}{\partial z_n \partial z_m} \end{bmatrix} \tag{16}$$

When $m = n = i$, H_1 is formed, when $m = n = j$, H_4 is created, when $m = i, n = j$, H_2 is found, and when $m = j, n = i$, H_3 is generated. It turns out that these four matrices are related to each other and can be reduced to the same matrix H_1 such that $H_2 = -H_1$, $H_3 = -H_1$, and $H_4 = H_1$, and H_1 is given by:

$$H_1 = \frac{k_{str}}{l_{final}^3} \begin{bmatrix} [(l_{final} - s)\Delta x^2 + s l_{final}^2] & [(l_{final} - s)\Delta x \Delta y] & [(l_{final} - s)\Delta z \Delta x] \\ [(l_{final} - s)\Delta x \Delta y] & [(l_{final} - s)\Delta y^2 + s l_{final}^2] & [(l_{final} - s)\Delta z \Delta y] \\ [(l_{final} - s)\Delta z \Delta x] & [(l_{final} - s)\Delta z \Delta y] & [(l_{final} - s)\Delta z^2 + s l_{final}^2] \end{bmatrix}$$

Now we have analytical expressions for the double derivatives corresponding to the stretching energy systematically arranged and here is how they should be placed in the global tangent stiffness matrix (K). Looking back at the general form of the matrix H (Equation 16), it is placed in K such that $K(3m-2:3m, 3n-2:3n) = K(3m-2:3m, 3n-2:3n) + H$. For instance, if we have a bar element with ends nodes 1 (node i) and 2 (node j), then the four matrices will be placed as: $K(1:3,1:3) = K(1:3,1:3) + H_1, K(1:3,4:6) = K(1:3,4:6) +$

$H_2, K(4{:}6,1{:}3) = K(4{:}6,1{:}3) + H_3$, and $K(4{:}6,4{:}6) = K(4{:}6,4{:}6) + H_4$. Therefore, using the formulations derived above, one can easily obtain the contribution of every bar element and place it accordingly to fill the global tangent stiffness matrix.

4.4.2.2. Contribution of torsional spring energy

As we are aware that the torsional spring energy $U_{rot} = \frac{1}{2} k_{rot}(\theta - \theta_0)^2$ is dependent on the nodal positions of four nodes as shown in Fig. 28. This means that to obtain its contribution in the global tangent stiffness matrix K, one can take the double derivatives with respect to every possible pair of every DOF associated with the four related nodes. Considering displacement in three directions as DOF, we have a total of 12 DOF. Therefore, in total there are 12×12 i.e., 144 pairs possible and these are the 144 double derivatives we need to obtain in order to fill K. Here is how they will look like and how they can be evaluated:

In general, suppose we have two different DOF, say, u_i^k and u_j^l, these can be read as the displacements of node k ($k \in \{a, b, c, d\}$) along the i axis (this can be X or Y or Z axis) and of node l along the j axis. Therefore, one of the two double derivatives of U_{rot} with respect to these two DOF will take the form:

$$\frac{\partial^2 U_{rot}}{\partial u_i^k \partial u_j^l} = k_{rot} \frac{\partial}{\partial u_i^k}\left[(\theta - \theta_0)\frac{\partial \theta}{\partial u_j^l}\right] = k_{rot}\left[\frac{\partial \theta}{\partial u_i^k}\frac{\partial \theta}{\partial u_j^l} + (\theta - \theta_0)\frac{\partial^2 \theta}{\partial u_i^k \partial u_j^l}\right]$$

We already know the analytical expressions for the single derivatives in the above expression from the Section 4.4.1.2. Now, we need to figure out how to obtain the double derivative of dihedral angle θ with respect to the two DOF, i.e., $\frac{\partial^2 \theta}{\partial u_i^k \partial u_j^l}$. As writing down all the 144 expressions is not the most efficient way to express these derivatives, they are clubbed together such that, for instance, instead of calculating $\frac{\partial^2 \theta}{\partial x_a \partial y_b}$ i.e., double derivative of dihedral angle with respect to displacement of node a along the X-axis and displacement of node b along the

Y-axis, we formulate $\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_b}$. Here, $\vec{r}_a$ and $\vec{r}_b$ are the position vectors of nodes a and b, i.e., $\vec{r}_a = (x_a, y_a, z_a)$. Because $\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_b}$ is a double derivative with respect to vectors, it turns out to be a 3×3 matrix which can be directly placed in the global tangent stiffness matrix as discussed in Section 4.4.2.1. From [23], these derivatives are obtained one by one and are listed below ($\otimes$ denotes a tensor product):

$$\frac{\partial^2\theta}{\partial\vec{r}_a^2} = -\frac{\vec{n}_{abc}}{|\vec{r}_{ac}||\vec{n}_{abc}|^2}\otimes\left(\left(1-2\frac{\vec{r}_{ab}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\right)\vec{r}_{ac}+\vec{r}_{ab}\right)+\left(\frac{\vec{r}_{ab}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}-1\right)\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_a}$$
$$+\left(\frac{\vec{r}_{dc}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_a}+\frac{\vec{n}_{acd}}{|\vec{r}_{ac}||\vec{n}_{acd}|^2}\otimes\left(\vec{r}_{dc}-2\frac{\vec{r}_{dc}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\vec{r}_{ac}\right)\right)$$

$$\frac{\partial^2\theta}{\partial\vec{r}_b^2} = \frac{|\vec{r}_{ac}|}{|\vec{n}_{abc}|^4}(\vec{n}_{abc}\otimes(\vec{r}_{ac}\times\vec{n}_{abc})+(\vec{r}_{ac}\times\vec{n}_{abc})\otimes\vec{n}_{abc})$$

$$\frac{\partial^2\theta}{\partial\vec{r}_c^2} = \frac{\vec{n}_{acd}}{|\vec{r}_{ac}||\vec{n}_{acd}|^2}\otimes\left(\left(2\frac{\vec{r}_{dc}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}-1\right)\vec{r}_{ac}-\vec{r}_{dc}\right)+\left(\frac{\vec{r}_{dc}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}-1\right)\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_c}$$
$$-\left(\frac{\vec{r}_{ab}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_c}-\frac{\vec{n}_{abc}}{|\vec{r}_{ac}||\vec{n}_{abc}|^2}\otimes\left(2\frac{\vec{r}_{ab}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\vec{r}_{ac}-\vec{r}_{ab}\right)\right)$$

$$\frac{\partial^2\theta}{\partial\vec{r}_d^2} = -\frac{|\vec{r}_{ac}|}{|\vec{n}_{acd}|^4}(\vec{n}_{acd}\otimes(\vec{r}_{ac}\times\vec{n}_{acd})+(\vec{r}_{ac}\times\vec{n}_{acd})\otimes\vec{n}_{acd})$$

$$\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_a} = -\frac{|\vec{r}_{ac}|}{|\vec{n}_{abc}|^4}(\vec{n}_{abc}\otimes((\vec{r}_{ac}-\vec{r}_{ab})\times\vec{n}_{abc})+((\vec{r}_{ac}-\vec{r}_{ab})\times\vec{n}_{abc})\otimes\vec{n}_{abc})-\frac{\vec{n}_{abc}\otimes\vec{r}_{ac}}{|\vec{n}_{abc}|^2|\vec{r}_{ac}|}$$

$$\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_c} = -\frac{|\vec{r}_{ac}|}{|\vec{n}_{abc}|^4}(\vec{n}_{abc}\otimes(\vec{r}_{ab}\times\vec{n}_{abc})+(\vec{r}_{ab}\times\vec{n}_{abc})\otimes\vec{n}_{abc})+\frac{\vec{n}_{abc}\otimes\vec{r}_{ac}}{|\vec{n}_{abc}|^2|\vec{r}_{ac}|}$$

$$\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_d} = \begin{bmatrix}0&0&0\\0&0&0\\0&0&0\end{bmatrix}$$

$$\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_a} = \frac{|\vec{r}_{ac}|}{|\vec{n}_{acd}|^4}(\vec{n}_{acd}\otimes(\vec{r}_{dc}\times\vec{n}_{acd})+(\vec{r}_{dc}\times\vec{n}_{acd})\otimes\vec{n}_{acd})+\frac{\vec{n}_{acd}\otimes\vec{r}_{ac}}{|\vec{n}_{acd}|^2|\vec{r}_{ac}|}$$

$$\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_c}=\frac{|\vec{r}_{ac}|}{|\vec{n}_{acd}|^4}\left(\vec{n}_{acd}\otimes\left((\vec{r}_{ac}-\vec{r}_{dc})\times\vec{n}_{acd}\right)+\left((\vec{r}_{ac}-\vec{r}_{dc})\times\vec{n}_{acd}\right)\otimes\vec{n}_{acd}\right)-\frac{\vec{n}_{acd}\otimes\vec{r}_{ac}}{|\vec{n}_{acd}|^2|\vec{r}_{ac}|}$$

$$\frac{\partial^2\theta}{\partial\vec{r}_c\partial\vec{r}_a}=\frac{\vec{n}_{acd}}{|\vec{r}_{ac}||\vec{n}_{acd}|^2}\otimes\left(\vec{r}_{dc}-2\frac{\vec{r}_{dc}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\vec{r}_{ac}\right)+\left(\frac{\vec{r}_{dc}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}-1\right)\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_a}$$

$$-\left(\frac{\vec{r}_{ab}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_a}-\frac{\vec{n}_{abc}}{|\vec{r}_{ac}||\vec{n}_{abc}|^2}\otimes\left(\left(1-2\frac{\vec{r}_{ab}\cdot\vec{r}_{ac}}{|\vec{r}_{ac}|^2}\right)\vec{r}_{ac}+\vec{r}_{ab}\right)\right)$$

The rest are related to the above-mentioned expressions due to the nature of a double derivative such that:

$$\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_d}=\left(\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_a}\right)^T,\frac{\partial^2\theta}{\partial\vec{r}_c\partial\vec{r}_b}=\left(\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_c}\right)^T,\frac{\partial^2\theta}{\partial\vec{r}_c\partial\vec{r}_d}=\left(\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_c}\right)^T$$

$$\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_b}=\left(\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_a}\right)^T,\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_c}=\left(\frac{\partial^2\theta}{\partial\vec{r}_c\partial\vec{r}_a}\right)^T,\frac{\partial^2\theta}{\partial\vec{r}_d\partial\vec{r}_b}=\left(\frac{\partial^2\theta}{\partial\vec{r}_b\partial\vec{r}_d}\right)^T$$

As discussed in Section 4.4.2.1, the above 3×3 mini matrices can be directly placed in specific regions of K such as $\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_d}$ gets placed in K so that $K(3d-2{:}3d,3a-2{:}3a)=K(3d-2{:}3d,3a-2{:}3a)+\frac{\partial^2\theta}{\partial\vec{r}_a\partial\vec{r}_d}$.

Therefore, with this, we have finally obtained analytical expressions for the residual force vector $\vec{R}$ and also for the global tangent stiffness matrix K. Note that there is no influence of the external loads on K as the double derivative of the work done W with respect to any pair of DOF turns out to be zero. At the end, looking back at Equation 10, numerically speaking, this turns out to be a set of simple linear algebraic equations to be solved to obtain the set of new DOF that define a new configuration of the system under consideration that is a mechanically stable one.

As explained in Section 4.3, the above numerical procedure is the Newton-Raphson method. This technique is able to capture and solve for the equilibrium for most of the origami patterns.

However, there are cases when Newton's method is not good enough and it fails to capture all the mechanical behaviours. Mathematically speaking, the Newton's method is used to find out the roots of an equation. Similar to Equation 8, the expression whose roots are to be found is written in terms of its value and its derivative at a starting point. Graphically, this portrays figuring out the intersection point of the tangent at this starting point with the horizontal axis. As we iterate this procedure, we get closer to the root. The technique may fail if the starting point is too far away from the root as the tangent's intersection point ends up diverging from the root instead of coming closer to it in successive iterations. Therefore, choosing an accurate starting point is crucial for the method to succeed. A second reason for failure can be that the tangent obtained at a point may turn out to be horizontal which will not ever intersect the horizontal axis and hence, the root will be undefined.

Looking from the mechanics point of view, for an origami these failures can be experienced when the starting configuration is chosen to be far from equilibrium. An origami is a complicated structure and practically many configurations are possible which may not be even close to its stable state. Therefore, if the starting value in the Newton's method represents any such configuration, there are chances of diverging from the correct transition to the equilibrium state. Also, the Newton's method might not work when sudden mechanical transitions take place. For instance, if locally there is a node that snaps from one position to another and both these positions indicate equilibrium, then Newton's method may not be able to capture the second equilibrium state. This is because if we observe the residual force during this snap process, it seems like it suddenly rises first followed by a fall. This means there will be local maximum or a possibility for the tangent to be horizontal in the Newton's method. In case the starting point is close to this peak, then there are chances that this transition is not captured.

In the method discussed here, the mathematical formulations made are complex but good enough for the numerical implementation in the code established. The program developed is

able to effectively translate all the theory discussed above into relevant lines of code and include the influence of every element of the bar and hinge model on the total energy U_T and solve for the equilibrium of a given origami structure which is an asset in assessing the stable configurations of the design being considered. Now, it's important to transition to the other significant part of this work, i.e., modelling the self-folding of an origami and this what the next section entails.

5. SELF-FOLDING – A DETAILED ANALYSIS

In the equilibrium setup developed in the sections before, how do we exactly capture an external stimulus responsible for self-folding of an origami, is a pivotal question to start with. If only self-folding is the focus, then there are no external physical forces acting on the nodes of the reduced-order system, which means that the third term in the expression for U_T (Equation 4) is no longer important. However, the first two terms are still there as they correspond to the resistance offered by the system to the unwanted external changes and therefore, these too must be the ones capturing the effect of external stimulus. Consider a simple crease separating two panels. Physically speaking, due to a change in the intensity of the external stimulus, the angle between these two planes, i.e., the dihedral angle will change. If we had a bigger complex origami, we could still define these dihedral angles for every fold crease in the pattern which would have changed as a function of the external stimulus. Therefore, the effect leading to self-folding is captured by the preferred dihedral angle θ_0 (Equation 5) of every folding crease in an origami. There must be a relationship between the external stimulus and θ_0 guiding the self-folding (as derived in Section 5.3.2) such that the deviation in θ_0 from its initial value changes the total energy U_T defining a new system to be solved for equilibrium. Note that θ_0 for the bending creases is not influenced by the external stimulus and it conforms with the value it had in the initial system. However, to understand how is θ_0 of every folding crease in an origami is related to the external stimuli, we move on to the next section.

5.1. Common self-actuation methods and its smart use in origami

A well-known concept exploited in a self-folding origami is the concept of a bilayer which can be actuated using various kinds of external stimuli which was clear through the examples seen in Section 1.2. In a bilayer, one of the two layers has slightly different properties related to the linear expansion of the individual layers and once they bilayer is placed under the influence of

the external stimulus, one of the layers expands more than the other. But as they are theoretically bonded perfectly, the bilayer bends in order to release the stored energy. In this work, we have focused on using temperature as an external stimulus leading to restricted thermal expansion of the two bonded layers leading to a bending of the bilayer (Fig. 29). Here, the difference in the property leading to the bending of bilayer are the different coefficients of thermal expansion (α) of the blue and the brown-colored layers.

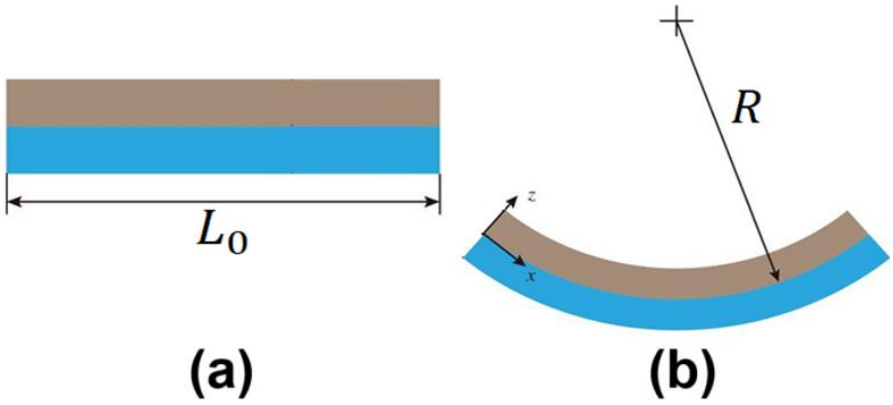


Fig. 29. (a) Thermal expansion of a bilayer (comprising of a blue and brown-colored layer) (b) Bending of bilayer due to a difference in the coefficients of thermal expansion.

It is interesting to know how the concept of bilayers is effectively used for self-folding of an origami. For instance, in Fig. 30a, it can be seen that a tri-layer like structure with a white-colored layer sandwiched between two pink-colored layers has some gaps cut out at specific regions across the length of the strip [27]. The white-colored layer expands more than the pink-colored one and the way these cuts have been made influence the direction of bending at those regions. As seen in Fig. 30b, the strip takes a deviation from the flat state to a bent zigzag like pattern where the function of these gaps in crafting a mountain or a valley crease is clear. Inspired from this concept, one can smartly cut the gaps in any origami tri-layer like pattern to define mountain and valley folds that will lead to a specific pattern of folding. A thought experiment yields Fig. 31, where a Miura origami unit cell is shown to be built on the same idea

as discussed here. However, having a closer look at the schematic in Fig. 31 uncovers the assumption that bending occurs only about the gaps or the creases formed. In reality, when two layers are bonded together and made to bend, they do not necessarily bend in a single direction to result in a single curvature. The final shape should highly depend on the dimensions of the two layers and there must be a way to control them in order to tune the resulting curvatures. This discussion reduces to a simple bilayer plate bending problem which has been discussed in the next section.

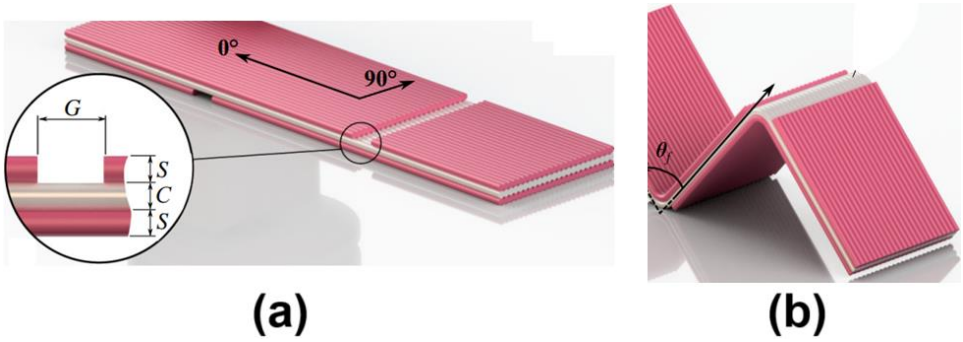


Fig. 30. (a) A concept using bilayer phenomenon for origami self-folding (b) A change in external stimuli leads to a specific folding pattern of the origami guided by the location and size of gaps (inset). Edited from [27]

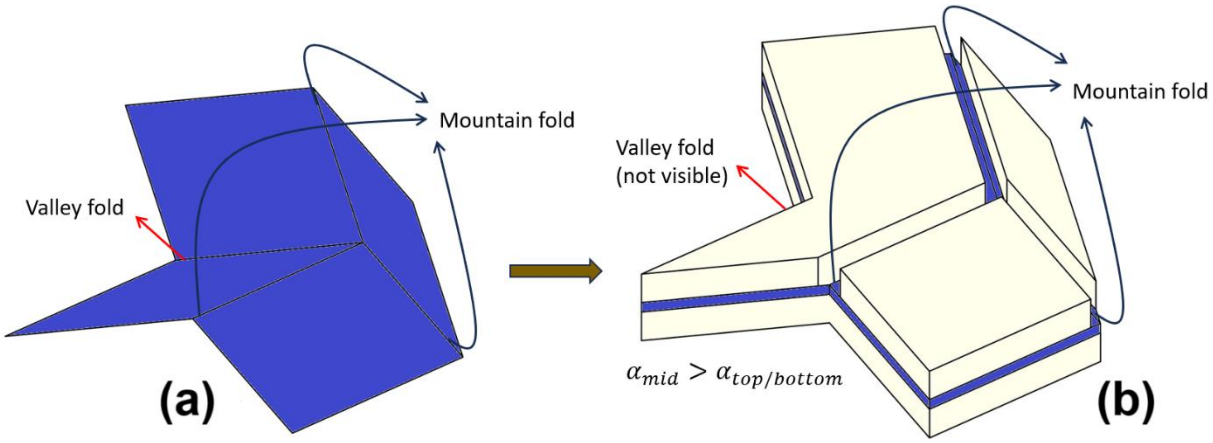


Fig. 31. (a) A Miura origami unit cell with defined mountain and valley creases (b) Corresponding tri-layer like structure to be actuated by change in temperature with gaps cut accordingly to mimic mountain and valley folds. (α is the coefficient of thermal expansion).

5.2. Bending of a bilayer plate – a mathematical solution

The bending of a bilayer plate has been researched in the past. It started with the classical model of bilayer bending by Timoshenko in 1925 [12] which assumed that a bilayer bends only in one direction. Results for bending in two directions were presented for a specific arrangement of bilayer plate [28, 29]. Here, the plates were elliptical in shape and had a lens shaped cross-section (similar to an inverted bowl). The bending was observed due to a thermal gradient through the thickness, and it was studied analytically. It was found that initially the bilayer plate assumes a spherical shape followed by choosing one curvature as the temperature climbed further. The point at which the spherical shape starts to transition was also studied numerically for a circular bilayer plate [30]. This section deals with coming up with analytical expressions for the two resulting curvatures when a bilayer plate is kept under uniform thermal gradient and how they are related to the geometrical and material properties of both the plates.

5.2.1. Free body diagrams

A schematic of a bilayer plate is shown in Fig. 32a. Note that it is called a plate because of the comparable length and width dimensions i.e., $(l \approx b)$. This would have been called a strip if either of the dimensions would have been very small compared to the other one. Focusing back on the bilayer plate, the two plates, plate I and II have thicknesses a_1 and a_2 which add up to a value signified by h and have coefficients of thermal expansion α_1 and α_2, respectively. We assume that the bilayer plate bends in two directions resulting in dual radii of curvatures about the Y (ρ_1) and the X-axis (ρ_2) as shown in Fig. 32b.

In order to solve for these two curvatures, we need to draw the free body diagrams of a section of the bilayer plate. First considering the radius of curvature ρ_1, to understand what forces and moments lead to the bending in this direction, a free body diagram of the relevant section is shown in Fig. 33a and the same has been done for ρ_2 in Fig. 33b. A common observation in

both the free body diagrams is that the top layer experiences a tensile internal force (P_1 in Fig. 33a and P_2 in Fig. 33b) whereas the bottom layer resists through compressive force (P_3 in Fig. 33a and P_4 in Fig. 33b). This happens, as, because of the positive difference in the coefficients of thermal expansion between the bottom and the top layer ($\alpha_2 > \alpha_1$), the bilayer tends to bend in the shown manner. Apart from the forces, there are also internal moments associated as bending is taking place. Therefore, based on the universal sign convention from the theory of bending, $M_1, M_2, M_3,$ and M_4 are shown in Fig. 33. Once the free body diagrams have been made, we can go ahead and write the equations for force and moment balance which are easier to interpret with a common free body diagram as shown in Fig. 34.

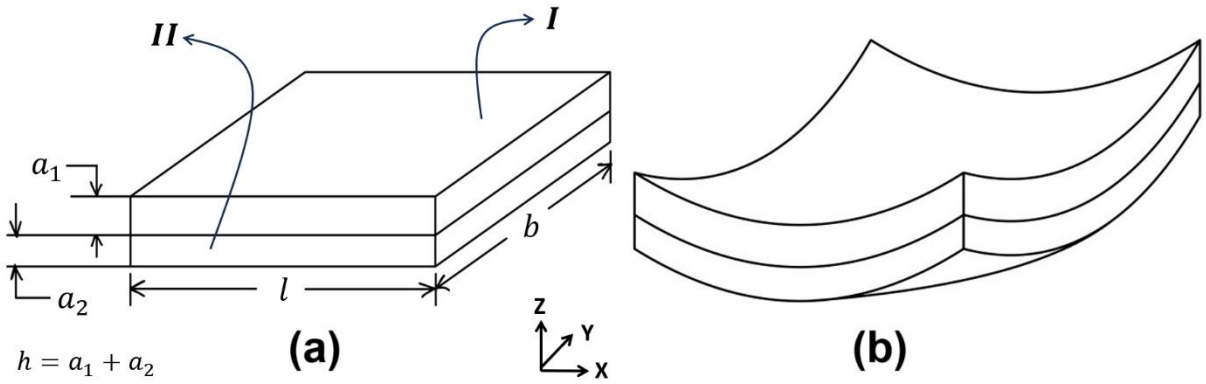


Fig. 32. Schematic of bilayer plate bending: (a) An unbent bilayer plate with shown dimensions ($l \approx b$) (b) Resulting bent structure of the bilayer plate with dual curvatures.

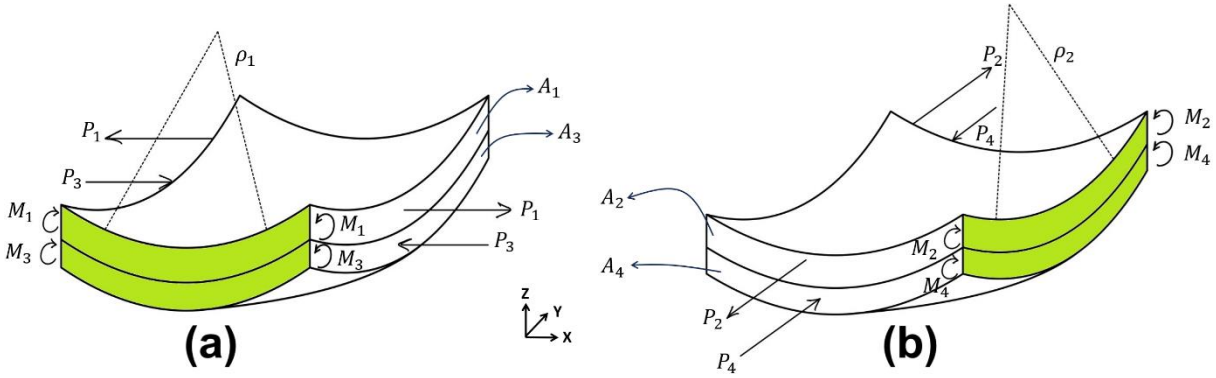


Fig. 33. Free body diagrams of the relevant sections of the bilayer plate for calculating the radii of curvatures: (a) ρ_1 (b) ρ_2.

5.2.2. Assembling the equilibrium equations

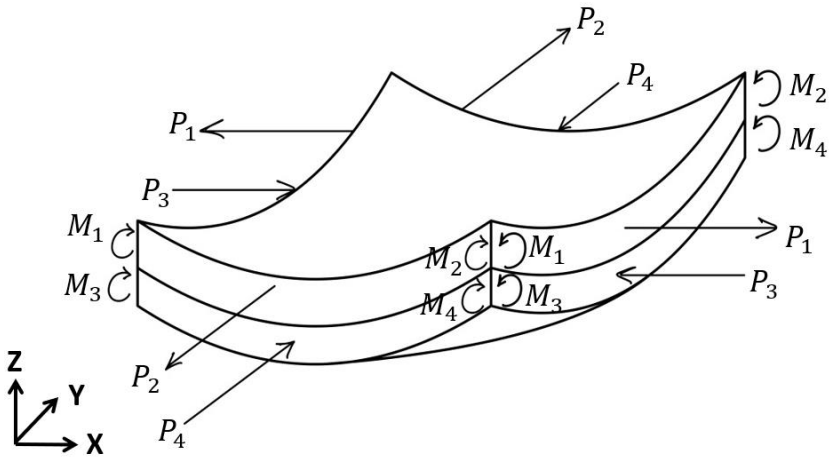


Fig. 34. Free body diagram of the bilayer plate used for writing equilibrium equations.

Balancing forces and moments (about a point at the interface) in and about the X and the Y directions leads to the following equations:

$$P_1 = P_3 \tag{17}$$

$$P_2 = P_4 \tag{18}$$

$$M_1 + M_3 = \frac{P_1 a_1}{2} + \frac{P_3 a_2}{2} = \frac{P_1 h}{2} \tag{19}$$

$$M_2 + M_4 = \frac{P_2 a_1}{2} + \frac{P_4 a_2}{2} = \frac{P_2 h}{2} \tag{20}$$

From the relationship between internal moment and the moment of inertia, we have the following equations:

$$M_1 = E_1 I_{1y}/\rho_1, \qquad M_3 = E_2 I_{3y}/\rho_1, \qquad M_2 = E_1 I_{2x}/\rho_2, \qquad M_4 = E_2 I_{4x}/\rho_2$$

Here, E_1 and E_2 are the Young's modulus of the two layers, and I_{1y} and I_{3y} are the moments of inertia of layer I and layer II about the Y-axis (the numbers 1 and 3 in the subscript denote the subscripts of the forces acting on those faces, for example, I_{1y} is the moment of inertia about Y-axis of the face that showcases an internal force of P_1). Similarly, I_{2x} and I_{4x} are also

moments of inertia of layers I and II about X-axis. These can be expressed in terms of the respective plate dimensions and are given by $I_{1y} = \frac{ba_1^3}{12}$, $I_{3y} = \frac{ba_2^3}{12}$, $I_{2x} = \frac{la_1^3}{12}$, and $I_{4x} = \frac{la_2^3}{12}$.

5.2.3. Common strain

However, the above equations are not sufficient to solve for ρ_1 and ρ_2. At the same time, it is crucial to observe that the displacement or the strain of a point at the interface of the two layers is the same when calculated with reference to layer I and layer II. Note that there are three key strains contributing to the total strain of any point on a layer and those are the normal strain due to the internal forces, bending strain due to the bending moment and then the thermal strain due to the change in temperature (ΔT). Also, the effect of Poisson's ratio too needs to be considered. Taking all of these into account, there are a total of four strains that can be calculated based on which two new relations can be established. These four strains include the strain of a layer I and layer II in the X and the Y directions, i.e., ϵ_{x_I}, ϵ_{y_I}, $\epsilon_{x_{II}}$, and $\epsilon_{y_{II}}$, given by:

$$\epsilon_{x_I} = \frac{P_1}{A_1E_1} + \frac{a_1}{2\rho_1} + \alpha_1\Delta T, \qquad \epsilon_{y_I} = \frac{P_2}{A_2E_1} + \frac{a_1}{2\rho_2} + \alpha_1\Delta T$$

$$\epsilon_{x_{II}} = -\frac{P_3}{A_3E_2} - \frac{a_2}{2\rho_1} + \alpha_2\Delta T, \quad \epsilon_{y_{II}} = -\frac{P_4}{A_4E_2} - \frac{a_2}{2\rho_2} + \alpha_2\Delta T$$

With the expressions for strains obtained, we can equate the relevant ones to obtain new equations. For instance, the strain (with the effect of Poisson's ratio considered) of layer I and layer II is the same in both the X and Y directions. This translates to the following equations:

$$\frac{P_1}{A_1E_1} + \frac{a_1}{2\rho_1} + \alpha_1\Delta T - \mu_1\epsilon_{y_I} = -\frac{P_3}{A_3E_2} - \frac{a_2}{2\rho_1} + \alpha_2\Delta T - \mu_2\epsilon_{y_{II}} \tag{21}$$

$$\frac{P_2}{A_2E_1} + \frac{a_1}{2\rho_2} + \alpha_1\Delta T - \mu_1\epsilon_{x_I} = -\frac{P_4}{A_4E_2} - \frac{a_2}{2\rho_2} + \alpha_2\Delta T - \mu_2\epsilon_{x_{II}} \tag{22}$$

Here, μ_1 and μ_2 are the Poisson's ratio of layers I and II. All the relations established above in equations 17 to 22 can be used to calculate the two radii of curvatures which turn out to be equal to one another and are given by:

$$\frac{1}{\rho_1} = \frac{1}{\rho_2} = \frac{\left((1-\mu_2)\alpha_2 - (1-\mu_1)\alpha_1\right)\Delta T}{\frac{1}{6h}\left(\frac{E_1 a_1^3}{1-\mu_1^2} + \frac{E_2 a_2^3}{1-\mu_2^2}\right)\left(\frac{1-\mu_1}{a_1 E_1} + \frac{1-\mu_2}{a_2 E_2}\right) + \frac{1}{2}(h - \mu_1 a_1 - \mu_2 a_2)} \tag{23}$$

The above calculations imply that a bilayer plate exhibits two curvatures which are equal to one another. However, it is important to note that the solution is only an approximate one and also only valid for the linear regime. Simple assumptions like stress is linearly proportional to the strain, relation between bending strain and moment, how thermal strain and change in temperature are linearly related, etc. make sure that this solution is only possibly true for small displacements. Thankfully this problem has been solved through energy calculations to know how and why does a bilayer bend and more importantly, how do the dimensions play an important role in guiding that.

5.2.4. Bilayer plate curvatures - an energy point of view

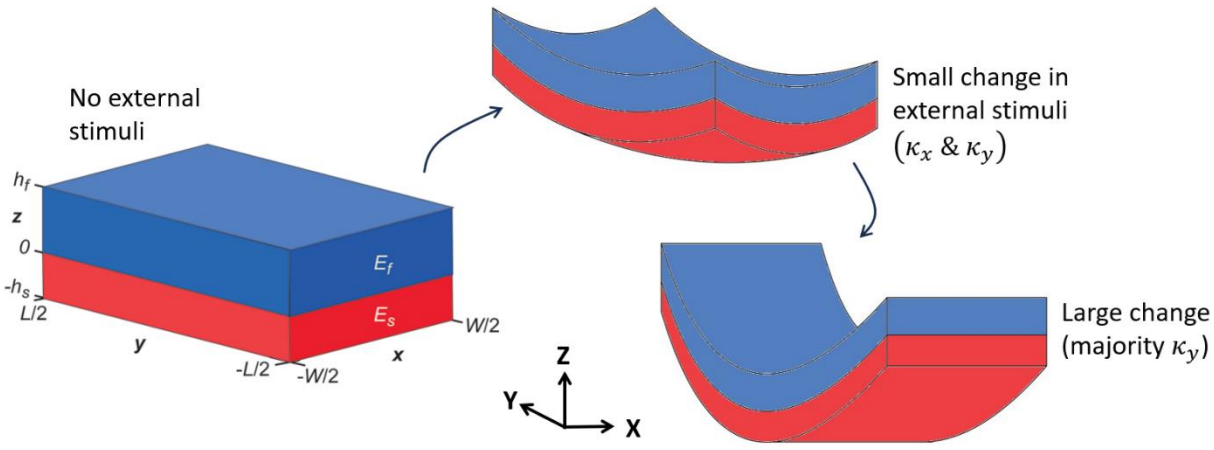


Fig. 35. Effect of external stimuli intensity on the resulting curvature(s) of a bilayer plate.

Edited from [31]

It has been found that initially when the change in external stimuli is small in intensity, a bilayer plate bends in two directions resulting in two curvatures [31] that also matches with our calculations in Section 5.2.3 (Equation 23). However, as this intensity rises, the bilayer tends to take a single curvature which is dependent on the ratio of the length and the width of the bilayer i.e., the aspect ratio. Moreover, higher is the aspect ratio, quicker this change takes place, and the bending takes place along the longer dimension. This transition has been shown in Fig. 35.

Therefore, as a conclusion in order to bend a bilayer quickly in a single direction, one needs a high aspect ratio or a thin bilayer strip. This takes us back to the thought experiment carried out to generate Fig. 31 about using a tri-layer like structure to self-fold a Miura origami unit cell. Based on our conclusion of using thin bilayer strips instead, a schematic is presented in Fig. 36 that shows how thin bilayer strips can be used across the creases to act as a mountain or a valley crease for effective self-folding of the whole structure. It is also important to note that the bilayer plate analysis done above considers an isolated bilayer. However, in case of an origami, we have rigid panels all around connected with each other. Therefore, the panels too will make sure that the bending of the bilayer stirp used to capture self-folding occurs only in one direction. And this direction will be the one the crease is originally supposed to fold.

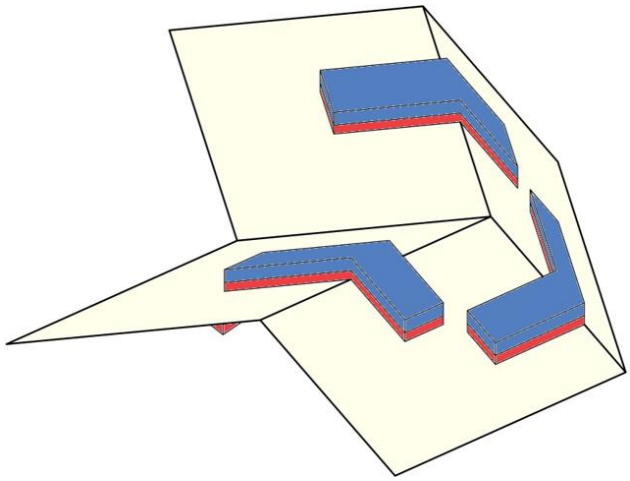


Fig. 36. Schematic of using thin bilayer strips as mountain and valley folds for overall self-folding of the Miura origami unit cell.

5.3. Single curvature of a bilayer strip

5.3.1. Radius of curvature

Similar to the calculations carried out in Section 5.2, one can easily obtain the relationship between the single radius of curvature of a thin bilayer strip and its dimensions [12]. As showcased in Fig. 37, one of the dimensions (width b) is very small as compared to the other dimension (length l) which makes sure that a single curvature is obtained due to the influence of an external stimulus. Other parameters include the thicknesses a_1 and a_2 of the two layers I and II which add up to a quantity h. Moreover, the layers I and II have coefficients of thermal expansion α_1 and α_2, respectively, such that $\alpha_2 > \alpha_1$. A free body diagram can be made following the procedure in Section 5.2.1 and it can be found that the radius of curvature ρ is given by:

$$\frac{1}{\rho} = \frac{6(\alpha_2-\alpha_1)(\Delta T)(1+m)^2}{h\left(3(1+m)^2+(1+mn)\left(m^2+\frac{1}{mn}\right)\right)} \tag{24}$$

Here, $m = a_1/a_2$ and $n = E_1/E_2$, in which E_1 and E_2 are the Young's modulus of the two layers.

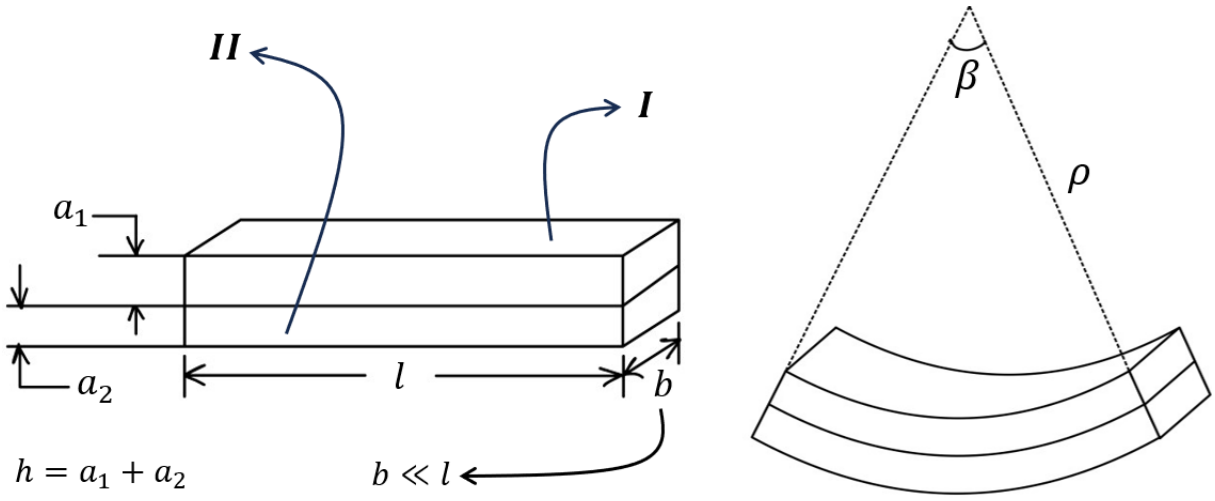


Fig. 37. A thin bilayer strip and its single radius of curvature.

Note that the above analysis is done considering a thermal gradient or a change in temperature. However, similar effects of bilayer strip bending can be obtained using some other stimulus. Mathematically, the procedure to attain an approximate analytical expression for the curvature will remain the same as discussed in Section 5.2. The only difference accounting for the external stimulus will be reflected in the equations for common strain. As seen in Section 5.2.3, the thermal strain is considered in Equations 21 and 22. For some other stimulus or even a combination of stimuli, the term corresponding to the stimuli strain will change and ultimately, a different expression for the curvature will be obtained.

5.3.2. Establishing connection between curvature and dihedral angle

With the single radius of curvature in place, how does one translate this change in curvature to the change in the dihedral angle, is a question worth answering. All we need to do is make a connection between the preferred dihedral angle θ_0 and the radius of curvature ρ, which is in turn related to the external stimuli, completing our story. Note in Fig. 37, in the curved bilayer strip there is an angle subtended by the bent bilayer β which is actually related to θ_0 as clarified through the schematic in Fig. 38. It can be observed that the bilayer strip is bonded across the crease common to the two planes which establish θ_0. Therefore, the sum of β and θ_0 must be π, that is:

$$\theta_0 = \pi - \beta = \pi - \frac{l'}{\rho} \tag{25}$$

Here, l' is the strained length of the interface of the bilayer strip and ρ is the radius of curvature. l' can be expressed in terms of the total strain and the initial length l while keeping in mind that the total strain includes contributions from the normal, bending, and the thermal strains. Therefore, $l' = l\left(1 + \frac{a_1}{2\rho} + \frac{E_1 a_1^3 + E_2 a_2^3}{6h\rho a_1 E_1} + \alpha_1 \Delta T\right)$, in which the second last term is the simplified form of the normal strain resulting in an expression only dependent on known parameters.

Hence, all of these expressions can be arranged together to form the following relationship between θ_0 and change in external stimulus (ΔT):

$$\theta_0 = \pi - \frac{6(\alpha_2-\alpha_1)(\Delta T)(1+m)^2 l}{h\left(3(1+m)^2+(1+mn)\left(m^2+\frac{1}{mn}\right)\right)}\left[1+\alpha_1\Delta T+\left(\frac{a_1}{2}+\frac{E_1a_1^3+E_2a_2^3}{6ha_1E_1}\right)\left(\frac{6(\alpha_2-\alpha_1)(\Delta T)(1+m)^2}{h\left(3(1+m)^2+(1+mn)\left(m^2+\frac{1}{mn}\right)\right)}\right)\right] \quad (26)$$

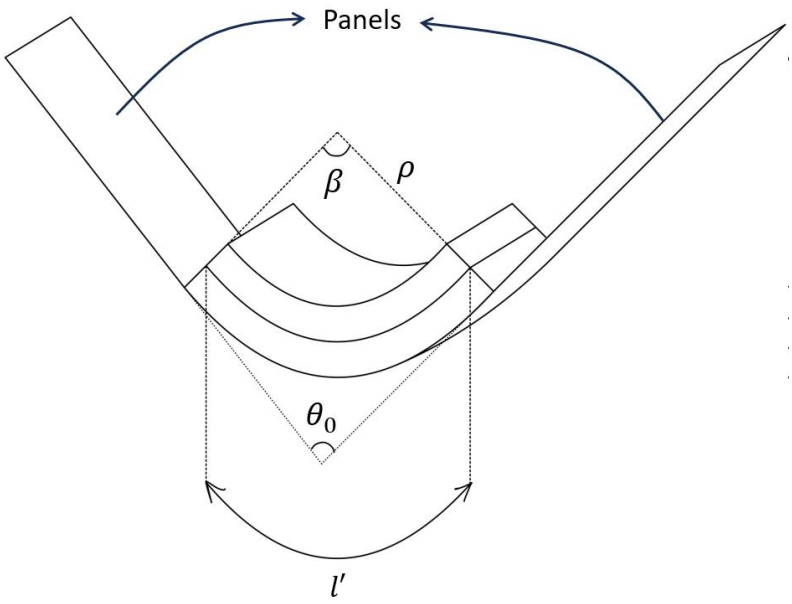


Fig. 38. A schematic of a bilayer strip attached to two panels uncovering relationship between preferred dihedral angle θ_0 and the radius of curvature ρ.

Hence, now we have an established framework that connects how external stimuli can influence the total energy U_T of the reduced-order system and all of the theory developed in the section about self-folding origami has been efficiently added to the code. This way the program developed is able to recognize all the folding creases and assign a unique relationship for the change in the preferred dihedral angle of that crease with the global change in external stimulus. Once this is done, it transfers to the strongly established section of solving for equilibrium where the general theories developed are carried on flawlessly and as a result, we are able to model the self-folding of an origami.

6. GIST OF THE CODE DEVELOPED

With this, we come to an end of the detailed narrative of how starting from scratch we established a smooth pathway between a CAD software and MATLAB to take a simple user input which was processed to identify the key elements for the reduced-order bar and hinge model that included two major tasks of panel identification and discretization for which a thorough recipe was developed and discussed. This was followed by a structured and intuitive theory on how the total energy captures every physical process during the folding of an origami and how it can be expressed in a way to make it easier for further procedure to be carried out.

A mathematics heavy series of steps were elaborated to come up with analytical formulations that drastically reduced the computation time of the numerical model developed. Additionally, to have a wider span of application, the aspect of modelling self-folding was also added with an in-depth analysis of how one can physically actuate an origami using an external stimuli and how it affects but also automatically aligns with the previously developed theories regarding energy minimization for finding out the mechanical stable states of an origami being folded through external loads or a stimulus.

In the next section, two simple examples are shown that show the ability of the code to model self-folding of a simple fold due to a rise in temperature and the mechanical folding process of a Miura origami unit cell. As an output, the code spits out 3D animation of the folding process, however, here, snapshots of the folding process at different time stamps have been included.

7. SOME BASIC RESULTS

7.1. Miura origami unit cell under the influence of external loads

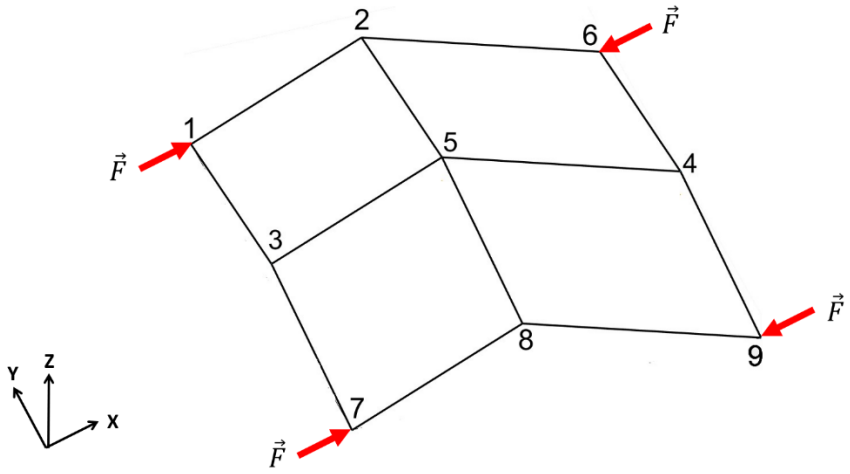


Fig. 39. Loading conditions of a Miura unit cell to be simulated.

Fig. 39 shows the phenomenon to be simulated using the code developed. It can be observed that the Miura unit cell is being loaded at the corner nodes such that nodes 1 and 7 experience force along the positive X direction meanwhile nodes 6 and 9 are under a force in the opposite direction. It is crucial to state the boundary conditions so as to avoid any rigid body motion and, in this case, the boundary conditions are explained through Fig. 40.

As seen in the said figure, the nodes that lie in the light blue-colored plane i.e., nodes 1,2,6,9,8, and 7 are restricted to displace along the Z-direction. Meanwhile, the nodes 2,5, and 8 that lie on the light green-colored region are not allowed to move in the X-direction. Lastly, node 2 is also fixed for translation in the Y-direction. All these boundary conditions are necessary to avoid any physically not possible folding process. For this example, in the code, a stretchable bar is configured with an area of cross-section of $A = 10^{-3}\, m^3$ and a Young's modulus of $E = 10^6 \frac{N}{m^2}$. The ratio of the folding and the bending stiffnesses of the respective torsional springs is taken as 10^{-3}. The force applied on the four nodes has a magnitude of $\left|\vec{F}\right| = \frac{70}{EA}$ each.

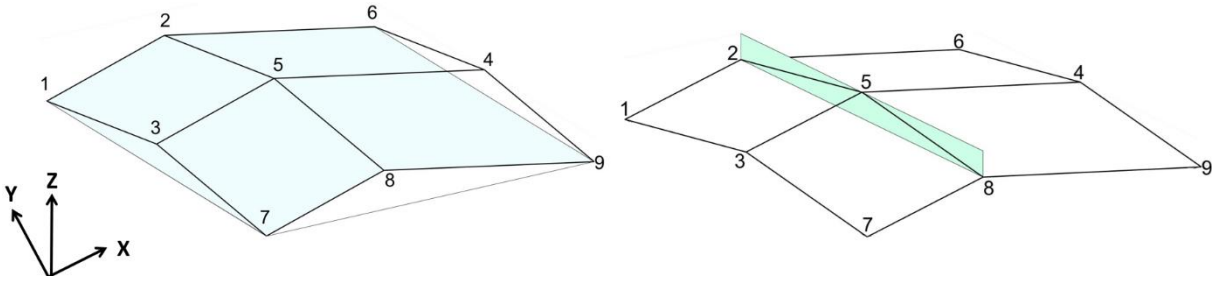


Fig. 40. Boundary conditions of the Miura unit cell to be modelled. Nodes lying in the colored planes are not allowed to move in a specific direction.

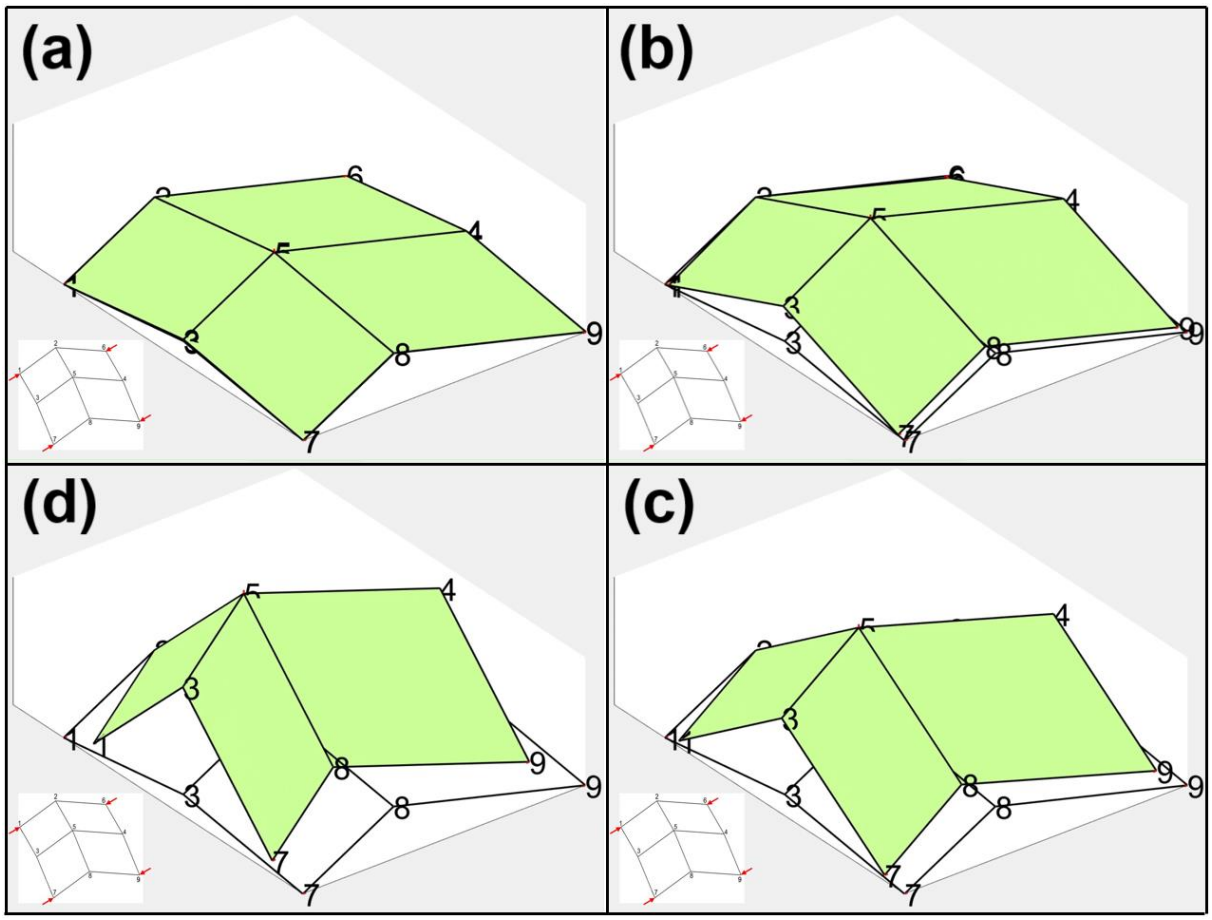


Fig. 41. Snapshots from the 3D animation of the folding process of a Miura unit cell generated through the code developed for the forementioned set of boundary conditions and loads. Time rises going from image (a) to image (d).

As seen in Fig. 41, a clear set of images showcase how the code presents the folding process of the Miura origami unit cell under the set of discussed boundary conditions. This folding process

matches with the folding process of a physical model if done by hand under similar circumstances.

7.2. Self-folding of a simple fold under rising temperature

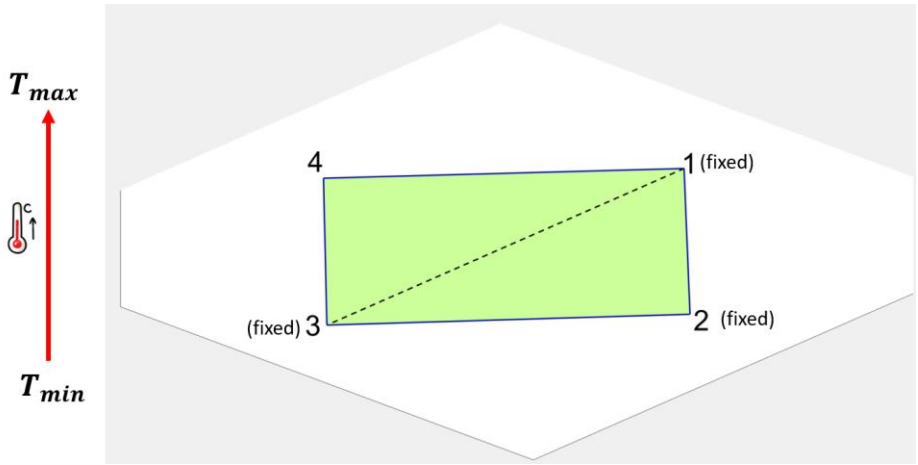


Fig. 42. A simple fold with three nodes fixed and ambient temperature is increased without any external loads applied.

As seen in Fig. 42, a simple fold is shown which basically means a folding crease separating two panels. Three out of the four nodes shown are fixed in all the three directions of translation. However, the fourth node is free to move anywhere. Note that there is no external force acting on an of the nodes and in order to actuate such a simple fold, one would need an external stimuli. In this example, ambient temperature is augmented from a minimum to a maximum value and the effect of that is presented as an output of the code in the form a 3D animation of the self-folding of this simple fold. For the thesis, snapshots of the self-folding simple fold at different values of temperature are shown in Fig. 43.

For this specific case, the bars have an area of cross-section of $A = 10^{-3}\ m^3$ and a Young's modulus of $E = 10^6\ \frac{N}{m^2}$. The ratio of the folding and the bending stiffnesses of the respective torsional springs is taken as 10^{-3}. For the bilayer strip present at the folding crease, the thicknesses, and Young's moduli for both the layers are taken as $10^{-3}\ m$ and $10^5\ \frac{N}{m^2}$,

respectively. Meanwhile, the coefficients of thermal expansions are considered to be $10^{-6}/°C$ and $4 \times 10^{-6}/°C$ for the two strips in the bilayer, respectively.

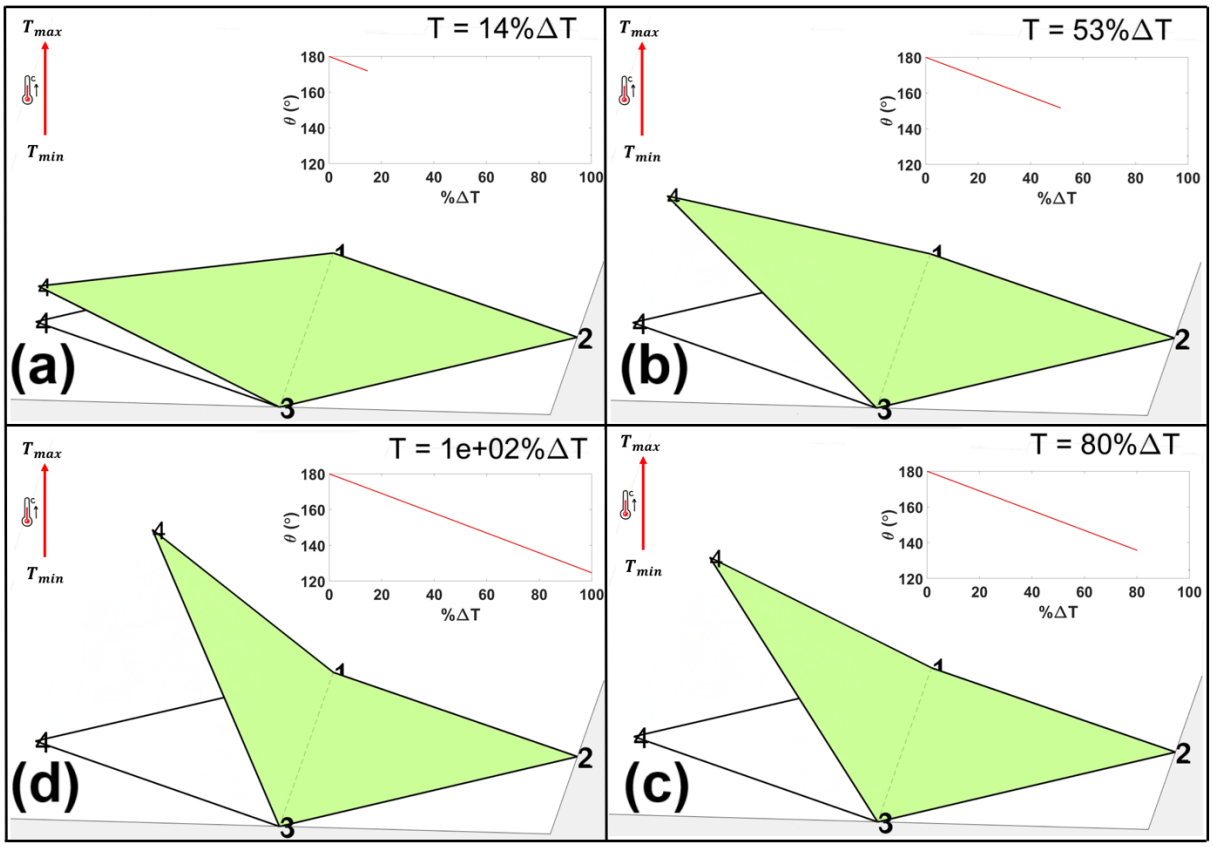


Fig. 43. Snapshots from the 3D animation of the self-folding process of a simple fold generated through the code developed for the forementioned set of boundary conditions. Temperature rises going from image (a) to image (d) as also clear from the $\%\Delta T$ attained.

As it can be observed from Fig. 43, the simple fold folds itself due to the rise in temperature which is clear from the movement of node 4 such that its traces an arc of a circle about the axis between nodes 1 and 3. Moreover, for a better understanding, the inset in each of the four images plots the dihedral angle with the percentage change in the temperature from the initial temperature. The dihedral angle decreases from 180° to about 120° as the temperature rises from T_{min} to T_{max}. This is an expected result and a full-proof example showing that the theory developed for self-folding is working correctly.

8. FUTURE DIRECTIONS AND IMPROVEMENTS

The present work is a testimony and a step towards the broad goal of designing a lockable self-folding origami. As discussed in the introduction, in order to generate special designs that are multi-stable and can self-fold in a specific sequence to lock themselves in one of the metastable states, we needed a computational framework that is able to quickly tell us if a particular design is favourable towards developing a lockable origami or not. Using the code developed, one can get to know the number and the configurations corresponding to the stable states of a design considered.

The code is a quick way to figure out if an origami is multi-stable or not based on which the possibility of locking can be explored. As concluded earlier, the kinetics of the energy landscape during self-folding is a way to know if a design is self-lockable or not. Because of the inclusion of self-folding in the code, one can observe the self-folding process and if the origami retains its shape in a stable state even after the external stimulus is removed. This can be done visually based on the strategic output of the code in the form of full 3D animation of the self-folding process of an origami.

Therefore, employing the code will quicken the process of identifying origami designs with multi-stable states and if any of these states can be retained even after the gradual absence of the stimuli. Moreover, even if a small change in the design can lead to swings in the mechanical behaviour and hence, the possibility of influencing the stable configurations to physically change how a stable state looks, this numerical method based computational platform is an asset in answering these questions.

With the above narrated purpose in mind, this work was carried out from scratch with a vision of keeping it as generalized as possible. The same has been reflected at different steps throughout the thesis. Especially, in the earlier sections that span over the coherent transfer of

user input for using the information so obtained to build the reduced-order model, a special focus has been made. Due to that, the code so written is able to identify panels in any origami design input, be it 2D or 3D, which is the first and most important step. Even the steps following identification which include panel discretization is based on a specific algorithm to keep the contribution of bending energy minimum in total energy functional which is an effort to keep the numerical technique close to the physically observed scenario.

The energy formulation and minimization is also done in a detailed fashion to address the contribution of every energy type popping up in the bar and hinge model. Despite the complicated derivatives involved, analytical expressions have been derived and used to minimize the time of computation, hence, meeting the primary goal of quickness in outputs. Most importantly, the major aspect of self-folding has been given a separate section and a detailed connected theory is developed considering the physically possible ways to actuate self-folding and how it is mathematically related to the change in the total energy functional of the system while making sure that the developments fit in seamlessly in the generalized theory discussed.

With all this, the program developed is currently in its raw stage and still contains regions of improvement for handling much more complicated designs showcasing special mechanical responses such as a snap-through behaviour. Present numerical method i.e., the Newton-Raphson method is able to capture many physical scenarios, however, in order to make the code versatile, a more sensitive numerical method can be employed. Other robust methods include the arc-length control method (ALCM) which is a technique in which instead of tracing the roots of the tangent to a curve, the direction along the tangent of the curve is followed [32, 33]. This eliminates the possibility of diverging from the true root of the curve which is a drawback of the Newton's method. Another type of solution scheme includes the generalized displacement control method (GDCM) which is known for its numerical stability while

handling non-linear systems [34]. A working application of a modified generalized displacement control method (MGDCM) in capturing snap-through behaviour in an origami structure can be found in [23].

Not only this, but the aspect of contact between panels is also unexplored in this work. However, contact mechanics in general is a well-researched field of study and one can bridge ideas from there to further improve the current state of the code. For instance, Liu et al. developed a simple contact model to avoid penetration in complicated membrane systems [35]. Specifically, the authors figured out potential points in the geometry of a solar sail that might come in contact with the hub during the spinning deployment process and calculated the possible penetration depth which was constrained to be always greater than or equal to zero. Sun et al. considered the contact force apart from the penetration depth to detect frictionless segment to segment contact [36]. Two contact constraint conditions were defined which implied that the gap between two segments is less than zero i.e., the segments have penetrated, and the normal contact force is positive. Penalty method was used to enforce these conditions where the size of the penalty parameter guided the contact threshold [37].

Therefore, in the future, above explained methods can be coupled with the bar and hinge model to simulate origami folding while avoiding contact between panels. A further advancement can be made in modelling the self-folding process of origami, where it will be interesting to see how the rigid panels around a crease influence the bending of the bilayer established on that crease to translate effect of external stimulus. Additionally, as every crease contains such a bilayer strip, it will be worth studying how one bilayer strip also influences the other physically.

That said, the code established so far is based on a robust base and is a tested starting point for achieving the major goal of designing self-folding lockable origami. The code developed is also capable of generating stored energy plots as a function of a DOF apart from the 3D animation

of the folding process. Using the energy plots, one can clearly identify the states of locally minimum energy which represent the stable states of the origami. This way it can be effectively used to quickly test new designs' multi-stability and observe how minute local changes can generate new paths to stability. To start with, current literature [9, 10] can be taken as an inspiration and detailed analysis can be made about how the current designs can be modified in order to entail a locking phenomenon. Special designs can be combined together to uncover multiplied degrees of freedom that may influence the behaviour of the overall system and hence, be a suitable candidate for the main goal.

Unique design features such as defining a local vertex at the centre of one of the panels has proven to be effective in rising the index of multi-stability of an origami [10]. This is ascribed to the local snap-through behaviour of that vertex during the global folding process. Such ideas can be combined together in many ways to possibly come up with origami patterns that can be guided externally to self-fold even in a desired sequence and the code developed as part of this work, can turn out to be an asset.

REFERENCES

[1] Tolley, M.T., Felton, S.M., Miyashita, S., Aukes, D., Rus, D. and Wood, R.J., 2014. Self-folding origami: shape memory composites activated by uniform heating. Smart Materials and Structures, 23(9), p.094006.

[2] Nishiyama, Y., 2012. Miura folding: Applying origami to space exploration. International journal of pure and applied mathematics, 79(2), pp.269-279.

[3] Thrall, A.P. and Quaglia, C.P., 2014. Accordion shelters: A historical review of origami-like deployable shelters developed by the US military. Engineering structures, 59, pp.686-692.

[4] Fernandes, R. and Gracias, D.H., 2012. Self-folding polymeric containers for encapsulation and delivery of drugs. Advanced drug delivery reviews, 64(14), pp.1579-1589.

[5] Wu, W. and You, Z., 2011. A solution for folding rigid tall shopping bags. Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 467(2133), pp.2561-2574.

[6] Ahmed, S., Ounaies, Z. and Arrojado, E.A.F., 2017. Electric field-induced bending and folding of polymer sheets. Sensors and Actuators A: Physical, 260, pp.68-80.

[7] Ze, Q., Wu, S., Nishikawa, J., Dai, J., Sun, Y., Leanza, S., Zemelka, C., Novelino, L.S., Paulino, G.H. and Zhao, R.R., 2022. Soft robotic origami crawler. Science advances, 8(13), p.eabm7834.

[8] Cheng, Y.C., Lu, H.C., Lee, X., Zeng, H. and Priimagi, A., 2020. Kirigami-based light-induced shape-morphing and locomotion. Advanced Materials, 32(7), p.1906233.

[9] Melancon, D., Gorissen, B., García-Mora, C.J., Hoberman, C. and Bertoldi, K., 2021. Multistable inflatable origami structures at the metre scale. Nature, 592(7855), pp.545-550.

[10] Melancon, D., Forte, A.E., Kamp, L.M., Gorissen, B. and Bertoldi, K., 2022. Inflatable origami: Multimodal deformation via multistability. Advanced Functional Materials, 32(35), p.2201891.

[11] Manna, R.K., Laskar, A., Shklyaev, O.E. and Balazs, A.C., 2022. Harnessing the power of chemically active sheets in solution. Nature Reviews Physics, 4(2), pp.125-137.

[12] Timoshenko, S., 1925. Analysis of bi-metal thermostats. Josa, 11(3), pp.233-255.

[13] Mishra, A.K., Wallin, T.J., Pan, W., Xu, A., Wang, K., Giannelis, E.P., Mazzolai, B. and Shepherd, R.F., 2020. Autonomic perspiration in 3D-printed hydrogel actuators. Science Robotics, 5(38), p.eaaz3918.

[14] Felton, S., Tolley, M., Demaine, E., Rus, D. and Wood, R., 2014. A method for building self-folding machines. Science, 345(6197), pp.644-646.

[15] Hanna, B.H., Lund, J.M., Lang, R.J., Magleby, S.P. and Howell, L.L., 2014. Waterbomb base: a symmetric single-vertex bistable origami mechanism. Smart Materials and Structures, 23(9), p.094009.

[16] Qiu, C., Zhang, K. and Dai, J.S., 2016. Repelling-screw based force analysis of origami mechanisms. Journal of Mechanisms and Robotics, 8(3), p.031001.

[17] Brunck, V., Lechenault, F., Reid, A. and Adda-Bedia, M., 2016. Elastic theory of origami-based metamaterials. Physical Review E, 93(3), p.033005.

[18] Zienkiewicz, O.C., Taylor, R.L. and Zhu, J.Z., 2005. The finite element method: its basis and fundamentals. Elsevier.

[19] Schenk, M., 2012. Folded shell structures (Doctoral dissertation, University of Cambridge).

[20] Wei, Z.Y., Guo, Z.V., Dudte, L., Liang, H.Y. and Mahadevan, L., 2013. Geometric mechanics of periodic pleated origami. Physical review letters, 110(21), p.215501.

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

[22] Filipov, E.T., Liu, K., Tachi, T., Schenk, M. and Paulino, G.H., 2017. Bar and hinge models for scalable analysis of origami. International Journal of Solids and Structures, 124, pp.26-45.

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

[24] Dias, M.A., Dudte, L.H., Mahadevan, L. and Santangelo, C.D., 2012. Geometric mechanics of curved crease origami. Physical review letters, 109(11), p.114301.

[25] Silvela, J. and Portillo, J., 2001. Breadth-first search and its application to image processing problems. IEEE Transactions on Image Processing, 10(8), pp.1194-1199.

[26] Verbeke, J. and Cools, R., 1995. The newton-raphson method. International Journal of Mathematical Education in Science and Technology, 26(2), pp.177-193.

[27] Baker, A.B., Bates, S.R., Llewellyn-Jones, T.M., Valori, L.P., Dicker, M.P. and Trask, R.S., 2019. 4D printing with robust thermoplastic polyurethane hydrogel-elastomer trilayers. Materials & Design, 163, p.107544.

[28] Mansfield, E.H., 1962. Bending, buckling and curling of a heated thin plate. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 268(1334), pp.316-327.

[29] Mansfield, E.H., 1965. Bending, buckling and curling of a heated elliptical plate. Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 288(1414), pp.396-417.

[30] Freund, L.B., 2000. Substrate curvature due to thin film mismatch strain in the nonlinear deformation range. Journal of the Mechanics and Physics of Solids, 48(6-7), pp.1159-1174.

[31] Alben, S., Balakrisnan, B. and Smela, E., 2011. Edge effects determine the direction of bilayer bending. Nano letters, 11(6), pp.2280-2285.

[32] Crisfield, M.A., 1981. A fast incremental/iterative solution procedure that handles "snap-through". In Computational methods in nonlinear structural and solid mechanics (pp. 55-62). Pergamon.

[33] Ramm, E., 1981. Strategies for tracing the nonlinear response near limit points. In Nonlinear Finite Element Analysis in Structural Mechanics: Proceedings of the Europe-US Workshop Ruhr-Universität Bochum, Germany, July 28–31, 1980 (pp. 63-89). Springer Berlin Heidelberg.

[34] Yang, Y.B. and Shieh, M.S., 1990. Solution method for nonlinear problems with multiple critical points. AIAA journal, 28(12), pp.2110-2116.

[35] Liu, C., Tian, Q., Yan, D. and Hu, H., 2013. Dynamic analysis of membrane systems undergoing overall motions, large deformations and wrinkles via thin shell elements of ANCF. Computer Methods in Applied Mechanics and Engineering, 258, pp.81-95.

[36] Sun, D., Liu, C. and Hu, H., 2019. Dynamic computation of 2D segment-to-segment frictionless contact for a flexible multibody system subject to large deformation. Mechanism and Machine Theory, 140, pp.350-376.

[37] Laursen, T.A., 2003. Computational contact and impact mechanics: fundamentals of modeling interfacial phenomena in nonlinear finite element analysis. Springer Science & Business Media.