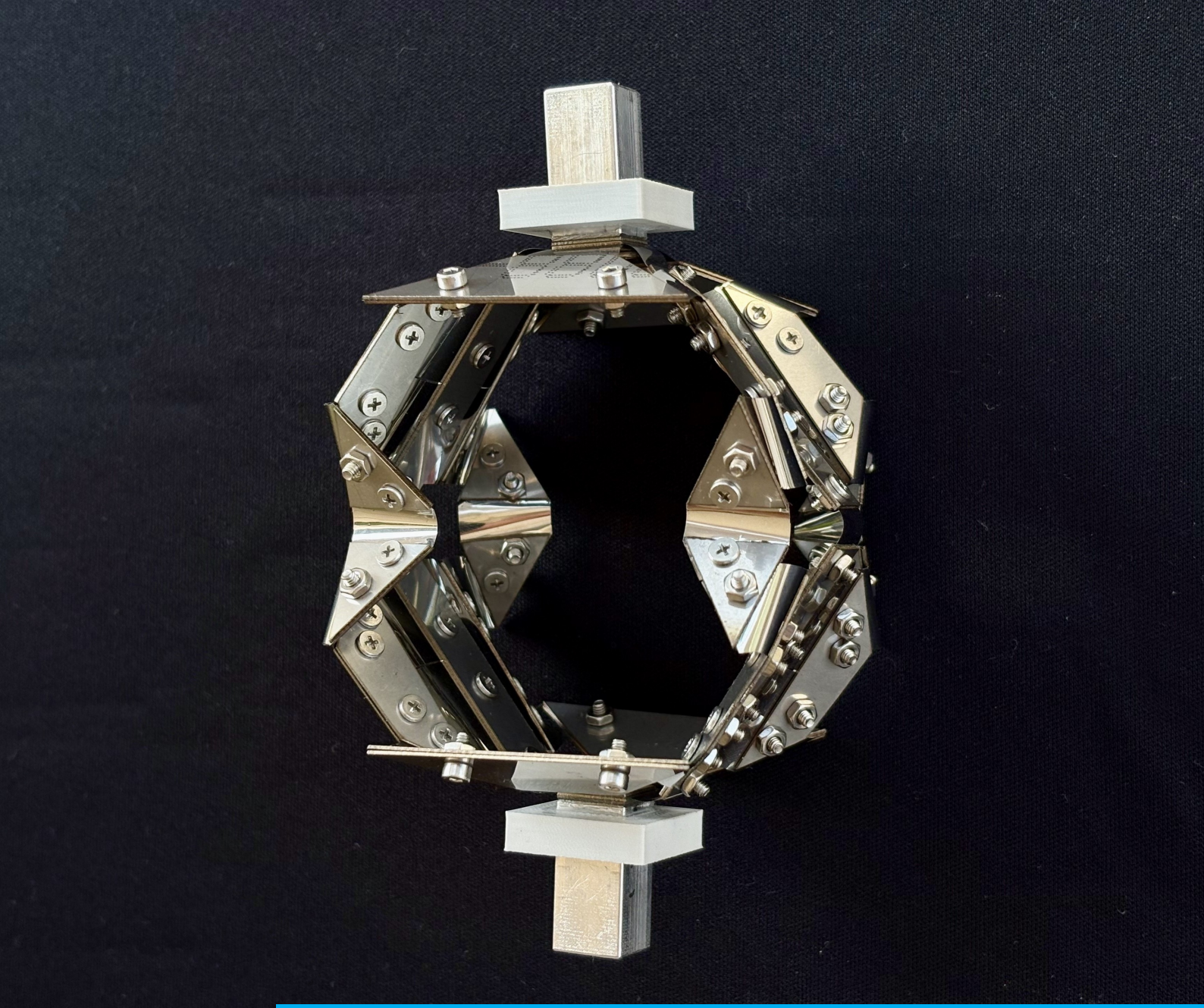

Department of Precision and Microsystems Engineering

Finite Element Modelling of Flexible non-Euclidean Origami

Max Benninga

Report no	: 2025.024
Coach	: Ph. D. candidate Mingkai Zhang
Professor	: Asst. Prof. Davood Farhadi Machekposhti
Specialisation	: MSD
Type of report	: Research paper
Date	: 20-06-2025

Preface

This thesis report signifies the end of my time as a student at the TU Delft. For the last one and a half years, I have been working on this project about flexible origami mechanisms. I faced some tough challenges along the way that heavily shaped the final result, but in the end I can truly say that I am proud of what I have achieved.

I could never have achieved what I did without the people around me. In that regard, I would first like to thank Davood Farhadi Machekposhti and Mingkai Zhang for their supervision and guidance throughout this project. You both repeatedly nudged me in the right direction, helping me reach my goal. I would also like to thank Gideon Emmaneel and Patrick van Holst from the technical support staff of the PME department. You helped me to fabricate and test my physical prototypes, which was an amazing learning opportunity for me. In a similar fashion, I want to express my gratitude to the Faculty Workshop of Mechanical Engineering for laser cutting hundreds of components for my prototypes.

From a more personal perspective, I also want to thank my girlfriend Sophie and my amazing family. You have all supported me in my toughest moments and I feel very lucky to have you. Finally, a special thanks to my fellow student and friend, Daan Roebroek. Sharing this journey with you, side by side for a year and a half, helped transform a potentially lonely individual project into a collaborative and joyful experience.

Max Benninga
Delft, June 2025

Contents

Finite Element Modelling of Flexible non-Euclidean Origami

Max Benninga

Abstract—Flexible origami is suitable for designing deployable mechanisms due to its ability to transform from a flat or compactly folded geometry to a more extended geometry. Non-Euclidean origami can help by splitting folding branches of origami vertices, leading to kinematically determinate behaviour. This paper presents two novel finite element modelling procedures for analysing flexible non-Euclidean origami, fabricated using 2D manufacturing techniques. The finite element modelling procedures presented in this work are a step towards the implementation of flexible non-Euclidean origami in functional applications, such as deployable mechanisms.

Index Terms—flexible origami, non-Euclidean origami, assembly, finite element modelling

I. Introduction

Deployable mechanisms play a crucial role in various industries, including aerospace [1]–[7], and medical industries [8]–[12]. Their ability to be compactly stored and transported, then expanded into a larger functional form, makes them highly valuable for applications where space efficiency and adaptability are essential [13].

Origami offers a way to go from a flat or compactly folded geometry to a more extended geometry in its deployed state. Origami is capable of doing this transformation by coupling the movements of the facets adjacent to a shared vertex. The result of these coupled movements is the three-dimensional motion of the origami model. Another advantage of origami is its manufacturability using only 2D fabrication techniques. These qualities are the main reason why origami is applied in real-world applications [14]–[19].

In this research, the focus is on the modelling of flexible origami; a class of origami that relies on the flexibility of its parts for movement, rather than using traditional rigid body joints [20]–[23]. In many of the sectors where deployability is needed, there are other special needs: vacuum compatibility in the aerospace industry and cleanliness in the medical sector. In the context of origami, flexible creases provide an effective solution to these specialised demands, which rigid body joints are typically unable to meet. Moreover, numerous studies have already explored the topic of origami with rigid body joints [24]–[29]. Flexible origami can be categorised based on how the facets are modelled: as rigid panels or as flexible members with specified thickness. This paper discusses both, with each having their own advantages. Rigid facet origami is more predictable and less computationally expensive to model, while flexible facet origami is better able to distribute stress and strain over its entire volume, leading to lower stress concentrations.

It is difficult to access the potential of flexible origami, as its origami vertices often have multiple folding branches. The concept of having multiple folding branches refers to the fact that a single set of input rotations can lead to multiple different output geometries. Due to the flexible creases, these folding branches will correspond to different energy levels. The higher-energy folding branch will be difficult to reach and maintain, as the mechanism naturally tends to return to its lower-energy state. This tendency is further supported by the flexibility of the creases, which enables the mechanism to deform more freely and return more readily to its lower-energy state.

As opposed to regular, Euclidean, origami, a more complex form of origami is non-Euclidean origami. Non-Euclidean origami can help by disconnecting the folding branches, making it possible to select folding branches that would otherwise not be employed [30], [31]. The difference between Euclidean and non-Euclidean origami lies in the sum of the sector angles, which are the angles between the creases of a vertex. In the case where this sum is exactly 360 degrees for all vertices, the origami is called Euclidean. Practically, this means that every vertex becomes flat when all the angles between the facets are zero. From now on, these angles will be called dihedral angles. In contrast to Euclidean origami, an origami tessellation that contains one or more vertices with a sector angle sum that equals more or less than 360 degrees is called non-Euclidean origami. Unlike their Euclidean counterparts, non-Euclidean vertices are inherently unable to reach a configuration in which all dihedral angles are simultaneously zero [30]. Figure 1 shows the differences between Euclidean and non-Euclidean origami in terms of appearance, elastic energy, and angular relations. The ability to disconnect folding branches using non-Euclidean origami is most evident in the angular relation plots in this figure. In addition to being used to split folding branches, non-Euclidean origami is also used in mechanisms that make use of multistabilities [32]–[34]. This potential is not explored in this paper, but it does make non-Euclidean origami even more versatile.

To yield the potential of flexible non-Euclidean origami, this paper presents two novel finite element modelling procedures to model this type of origami. The novelty of these procedures is the ability to use initially flat and unassembled non-Euclidean vertices, and model the process towards assembled non-Euclidean vertices. Due

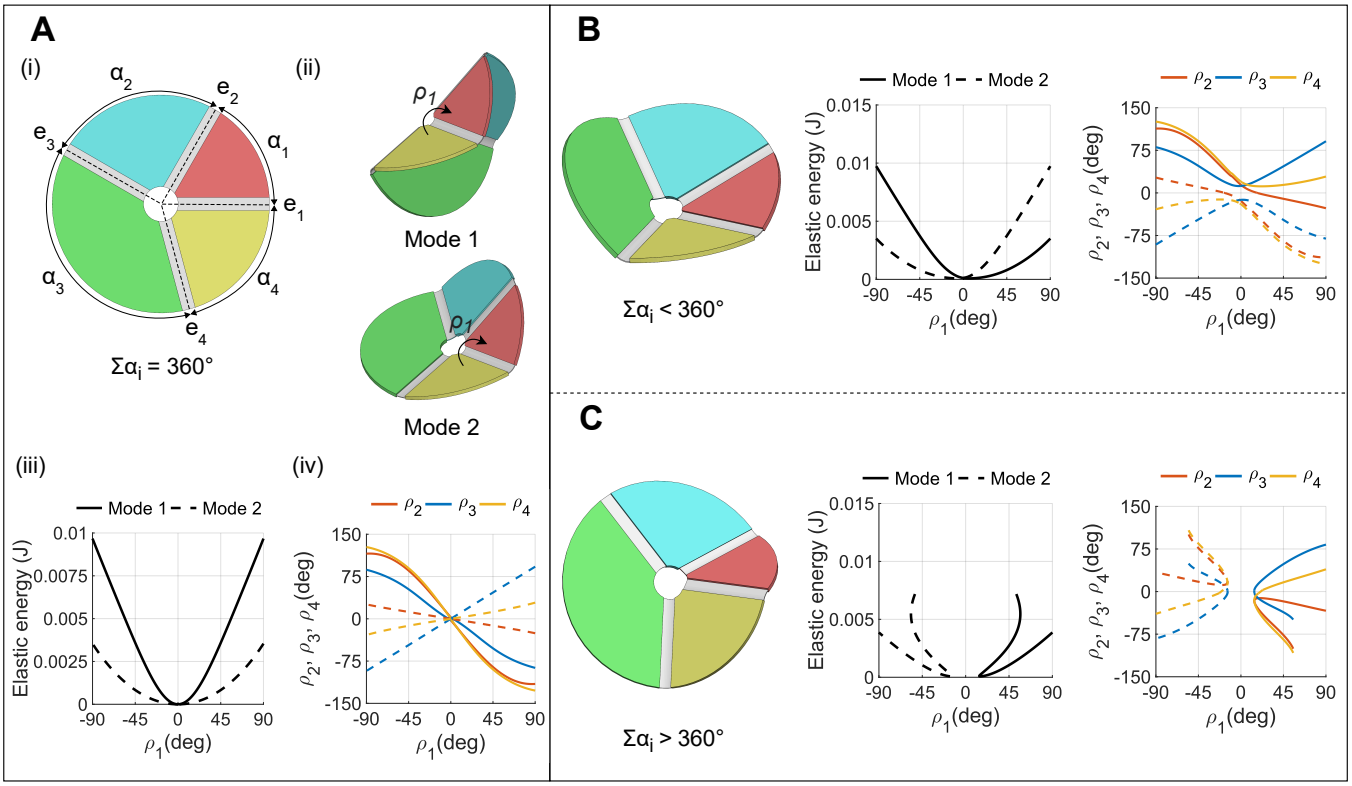

Fig. 1. Introduction of non-Euclidean vertices to split folding branches. A (i) Flexible Euclidean vertex. (ii) The two possible folding modes of the vertex, displayed with an equal ρ_1. (iii) Elastic energy of the vertex for both folding modes. (iv) Dihedral angular relations of the vertex for both folding modes. ρ_n correspond with the rotations around the axes e_n indicated in (i). B & C Images of the same vertex, with a sector angular deficit (B) and surplus (C), making it non-Euclidean vertices. Next to it, the energy and dihedral angular relation plots corresponding to the non-Euclidean vertices are shown.

to this approach, it is possible to evaluate mechanical responses as a result of assembly, while still being able to use 2D fabrication methods for manufacturing the flexible origami. Measurable mechanical responses consist of kinematics, elastic energy, internal stresses, and force-deflection behaviour during operation. The two modelling procedures differ in how they treat facets: the first assumes that the facets are infinitely rigid, while the second considers their flexibility.

In summary, this paper presents two novel finite element modelling procedures for analysing flexible non-Euclidean origami, fabricated using 2D manufacturing techniques. Non-Euclidean origami can be used either to split folding branches of vertices, resulting in kinematically determinate behaviour, or to exploit the multistability of non-Euclidean origami. The procedures introduced in this paper can be used in the process of developing better deployable mechanisms, making use of the advantages of flexible non-Euclidean origami.

In the next section, the two modelling procedures are introduced and explained in detail, and a fabrication method is presented. Afterwards, the Results section presents how both are applied to a single vertex and how the first procedure is applied to a more complex case study. After applying the modelling procedures, their possibilities and limitations are also reviewed. The subsequent Discussion section consists of two parts: a discussion of the discrepancies between the experimental and numerical data of the case study mechanism and suggestions for future research.

II. Methods

The numerical procedures for modelling flexible non-Euclidean are set up in ANSYS Mechanical APDL [35], but could be applied to other finite element analysis packages if these have the same capabilities. The two procedures correspond to two different levels of simplification of non-Euclidean origami. Both are novel because of their ability to capture the influence of non-Euclidean vertex assembly starting from a flat unassembled state. Figure 2 shows how both modelling methods relate to state-of-the-art methods for modelling non-Euclidean origami. This figure also lists the capturable mechanical responses for each simplification level of non-Euclidean origami. In the following two sections, the two previously mentioned numerical modelling procedures are explained in detail. At the end of the Methods section, a fabrication method is introduced that will be used for experimental validation of the first modelling procedure.

A. Procedure 1: rigid facet assembly

For this procedure, origami mechanisms are modelled as shell elements connected by rigid beams. The rigid beams

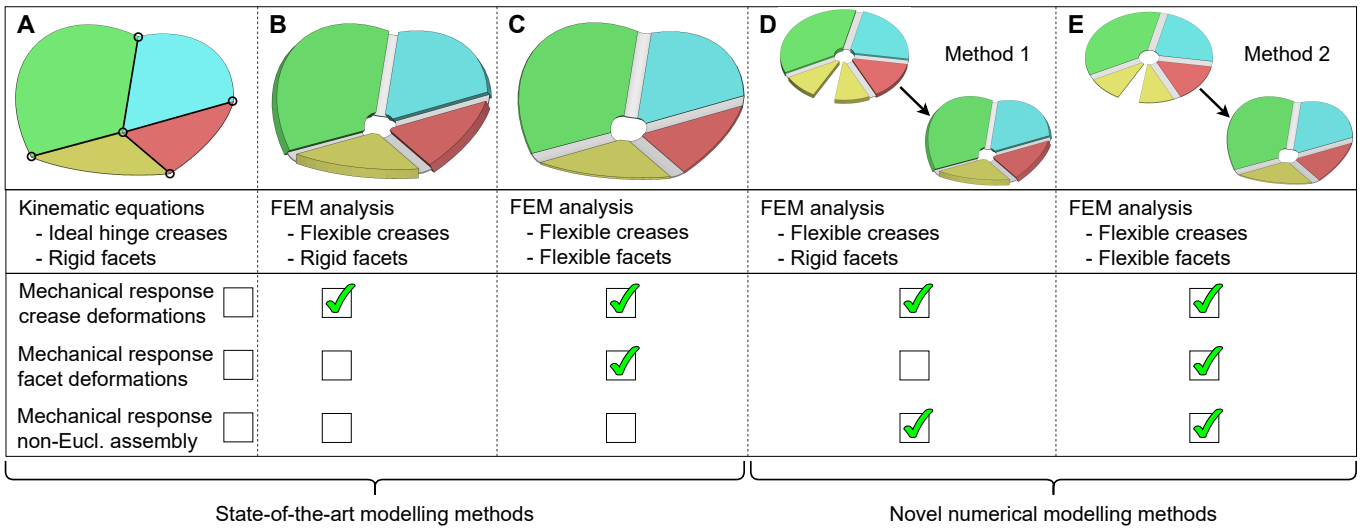

Fig. 2. Origami simplification levels along with their capturable mechanical responses. A Rigid facets combined with ideal hinge creases. B Rigid facets combined with flexible creases, not starting from a flat state. C Flexible facets combined with flexible creases, not starting from a flat state. D Rigid facets combined with flexible creases, starting from a flat state, so including non-Euclidean assembly. E Flexible facets combined with flexible creases, starting from a flat state, so including non-Euclidean assembly.

correspond to the rigid facets of the origami mechanism, while the shell elements correspond to the creases in between the facets.

In terms of element types and key options, this numerical framework uses the following:

- The SHELL281 element with its default element key options.
- The MPC184 element with KEYOPT(1) = 1 and KEYOPT(2) = 1.

SHELL281 was chosen for its suitability in large-strain nonlinear applications.

MPC184-Link/Beam element is chosen to construct rigid facets. Both of the chosen key options are essential for applying this procedure. The first key option defines the elements as rigid beams instead of links. The second key option selects the Lagrange multiplier method, instead of the direct elimination method. This is necessary for later use of the CP command, which is not compatible with the direction elimination method.

In preparation of modelling non-Euclidean origami, the geometry is built up with flat, unassembled non-Euclidean vertices. Therefore, one of the vertices' facets is split, creating two parts of a facet that need to be carefully aligned and coupled. The flat starting geometry is built up by creating areas for all creases and rigid facets. These areas are then meshed using the appropriate element types, creating a flat origami design. For creating the non-split facets, rigid beams should span from all the nodes of its adjacent lines to one central node. The position of this central node can be arbitrarily chosen, even outside of the facet area. However, for clarity, it is advised to choose a central position in the facet area.

For constructing the split facet, both parts should have their own central node, which lies on the facets' shared seam. For both sides of the split facet, two more nodes are created and rigidly connected to the central node. These nodes, adding to a total of three per side of the split, are used to align the parts of the facet. From now on, the to be coupled parts of the split facet are called panel A and panel B. Their respective central nodes are called A1 and B1, and the additional alignment nodes are called A2, A3, B2, and B3. To align the panels, the alignment nodes need to be in the same relative position, which means that if all three nodes coincide with their corresponding counterpart (A1 with B1, and so on), the facet will be whole.

Starting from the initially flat origami design, the model is loaded in steps to eventually form an assembled model. This procedure spans multiple load steps, which are as follows:

1) Fix the central node of one of the non-split facets in all 6 DOFs. This will be the connection of the mechanism to the ground. Rotate the central nodes A1 and B1 out of their original plane and toward each other. The closer they are, the easier the next steps will be.
2) Replace the rotations applied to A1 and B1 with their corresponding reaction moments. Simultaneously fix all three translational DOFs of A1 and B1.
3) Stepwise reduce the applied reaction moments to zero, while displacing B1 in all three translational DOFs to the location of A1, which is still fixed in space.
4) Couple all three translational DOFs of A1 to B1, while deleting the displacements applied to both. Replace the displacements applied to A1 with their corresponding reaction forces. Simultaneously fix two translational DOFs of A2 and B2.
5) Displace B2 in the same two translational DOFs to the location of A2, of which these two translational DOFs are still fixed in space.

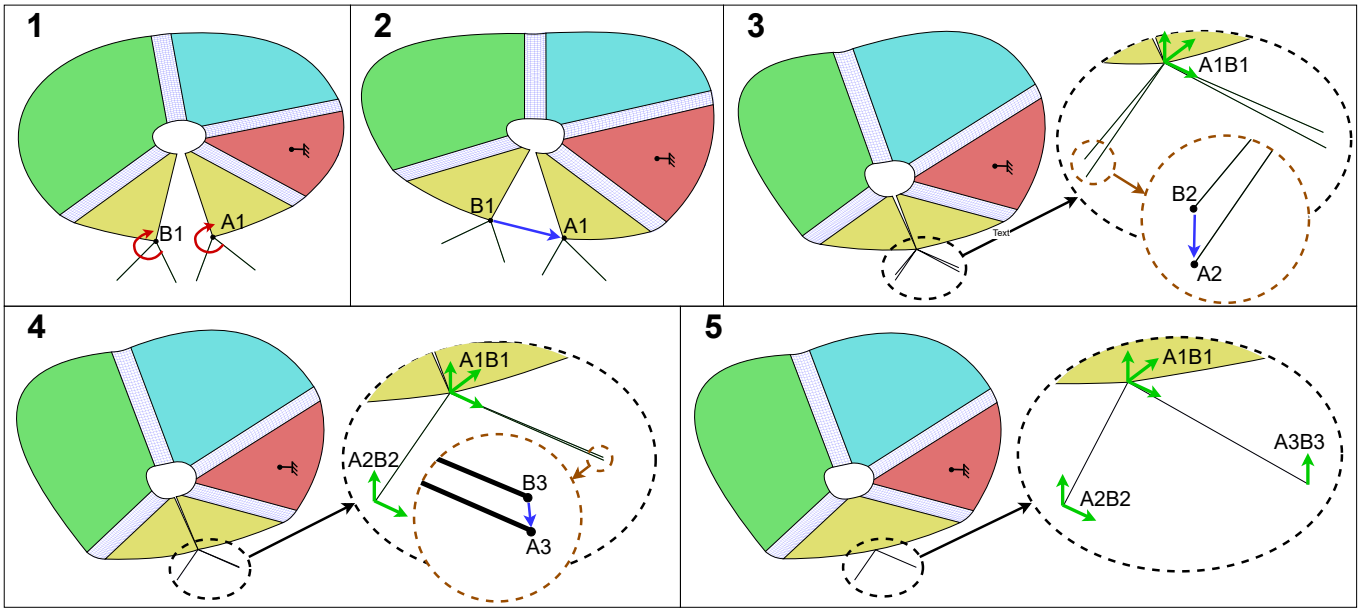

Fig. 3. Global overview of the assembly steps of procedure 1: assembling a rigid facet non-Euclidean vertex. 1 Starting from a flat unassembled vertex, fix the central node of the red facet. Rotate the central nodes A1 and B1 out of their original plane and toward each other. 2 Displace B1 in all translational DOFs to the location of A1, which is fixed in space. 3 Couple all three translational DOFs of A1 to B1. Displace B2 in two translational DOFs to the location of A2, which is fixed in space. 4 Couple the two aligned translational DOFs of A2 to B2. Displace B3 in one translational DOF to the location of A3, which is fixed in space. 5 Couple the aligned translational DOF of A3 to B3.

6) Couple the two aligned translational DOFs of A2 to B2, while deleting the displacements applied to both. Replace the displacements applied to A2 with their corresponding reaction forces. Simultaneously fix one translational DOF of A3 and B3.
7) Displace B3 in the same translational DOF to the location of A3, of which this translational DOF is still fixed in space. There is a possibility that B3 moves to another location than A3. If this is the case, go back to the last step and fix another translational DOF of A3 and B3.
8) Couple the aligned translational DOF of A3 to B3, while deleting the displacements applied to both. Replace the displacements applied to A3 with their corresponding reaction forces.
9) Stepwise reduce all remaining forces to zero. If needed, the connection to the ground could also be removed or changed.

After this procedure, panels A and B are coupled and can move freely together, forming an assembled rigid facet. A global overview of the assembly steps is shown in Figure 3. A more detailed explanation of the first procedure, including exemplary ANSYS APDL code, can be found in Appendix A.

B. Procedure 2: flexible facet assembly

This procedure uses the same element types as the first. However, the preparation of building the geometry is not the same. To understand the necessary preparation, the general goal should first be introduced. In the first procedure, it was sufficient to align and couple the split facet at one single location. This was the case because of the rigidity of the facets. Now that the facets will be modelled as flexible panels, it is necessary to couple the entire seam line of both sides of the facet to each other. To do so, a number of hard points need to be defined on the seam of either side of the split facet. These points will become the first alignment nodes (like A1 and B1 in procedure 1), and therefore need to be in the same relative position. The amount of hard points needed depends on the details of the simulation. Preferably, one would want to have as many as possible. However, the number of hard points is limited by the mesh size of the facets, as the areas cannot be meshed if there are too many hard points. A convergence study of the results can be done to determine the minimal amount of hard points required. Alternatively, a visual inspection can also be performed by checking the deformations of the facet between its coupled locations, and deciding whether or not they are acceptable. However, this second assessment method is subjective.

For the next step, for every hard point two additional nodes are created to serve as the two other alignment nodes (just like A2, A3, B2, and B3 in procedure 1). These alignment nodes are rigidly connected to their corresponding hard point. After this is done, the areas of the flat starting geometry are all meshed using shell elements. Generally, the facets are meshed using a different section, which is thicker than the creases' section. This completes the preparation for procedure 2.

To explain the load-step procedure, it helps to label the entities created in the preparation. Panels A and B each have a number of hard points along their seam line,

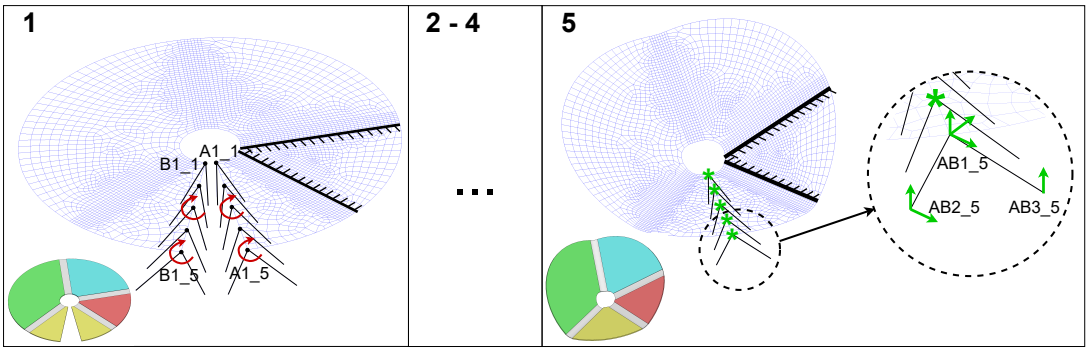

Fig. 4. First and last steps of the assembly of procedure 2: constructing a flexible facet non-Euclidean vertex. 1 Starting from a flat unassembled vertex, fix the bounding crease nodes of one of the other facets. Rotate the $A1_n$ and $B1_n$ alignment nodes out of their original plane and toward each other. 5 After aligning the $B3_n$ nodes with their corresponding $A3_n$ nodes in one translational direction, they are coupled in that direction. The intermediate steps for aligning and coupling the nodes are very similar to the steps in Figure 3.

labelled $A1_n$ and $B1_n$. Rigidly connected to these points are nodes called $A2_n$, $A3_n$, $B2_n$, and $B3_n$. After aligning all nodes of panel A with the corresponding counterpart of panel B ($A1_1$ with $B1_1$, and so on), the facet will be whole. The load steps needed for assembly are as follows:

1) Fix all 6 DOFs of one of the non-split facets, this will be the connection of the mechanism to the ground. It is sufficient to fix the facets' nodes that border the adjacent creases. Rotate the $A1_n$ and $B1_n$ alignment nodes out of their original plane and toward each other. The closer they are, the easier the next steps will be.
2) Replace the rotations applied to the $A1_n$ and $B1_n$ nodes with their corresponding reaction moments. Simultaneously fix all three translational DOFs of the $A1_n$ and $B1_n$ nodes.
3) Stepwise reduce the applied reaction moments to zero, while displacing the $B1_n$ nodes in all three translational DOFs to the locations of their corresponding $A1_n$ nodes, which are still fixed in space.
4) Couple all three translational DOFs of the $A1_n$ nodes to their corresponding $B1_n$ nodes, while deleting the displacements applied to both. Replace the displacements applied to the $A1_n$ nodes with their corresponding reaction forces. Simultaneously fix two translational DOFs of the $A2_n$ and $B2_n$ nodes.
5) Displace the $B2_n$ nodes in the same two translational DOFs to the locations of the $A2_n$ nodes, of which these two translational DOFs are still fixed in space.
6) Couple the two aligned translational DOFs of the $A2_n$ nodes to their corresponding $B2_n$ nodes, while deleting the displacements applied to both. Replace the displacements applied to the $A2_n$ nodes with their corresponding reaction forces. Simultaneously fix one translational DOF of the $A3_n$ and $B3_n$ nodes.
7) Displace the $B3_n$ nodes in the same translational DOF to the location of the $A3_n$ nodes, of which this translational DOF is still fixed in space. There is a possibility that the $B3_n$ nodes move to another location than the $A3_n$ nodes. If this is the case, go back to the previous step and fix another translational DOF of the $A3_n$ and $B3_n$ nodes.
8) Couple the aligned translational DOF of the $A3_n$ nodes to their corresponding $B3_n$ nodes, while deleting the displacements applied to both. Replace the displacements applied to the $A3_n$ nodes with their corresponding reaction forces.
9) Stepwise reduce all remaining forces to zero. If needed, the connection to the ground could also be removed or changed.

After this procedure, panels A and B are coupled and can move freely together, forming an assembled flexible facet. A global overview of this procedure is shown in Figure 4. Intermediate steps are omitted as they are very similar to the steps shown in Figure 3. A more detailed explanation of the first procedure, including exemplary ANSYS APDL code, can be found in Appendix B.

C. Fabrication method

To be able to experimentally validate numerical models, a sandwich technique is used to fabricate prototypes. This means that a thin flexible sheet is locally reinforced by sandwiching it between two significantly thicker plates. The clamping forces are exerted by bolts that extend through all three layers, fastening them securely. The reinforced regions represent rigid panels, while the creases are represented by the flexible areas. The flexible sheets are made from stainless steel 1.4310, while the thicker plates are also made from a stainless steel alloy. Both the flexible sheets and the reinforcement plates are manufactured by laser cutting. Due to the thickness of the flexible sheets, they are cut using a high-precision Lasea laser cutting machine. The reinforcements are cut using the Lion Alpha Metal XL laser cutting machine. Non-Euclidean surplus vertices cannot be made using a single flexible sheet. The flexible sheets and reinforcement plates are therefore assembled so that the reinforcement plates overlap the internal boundaries between the flexible sheets.

III. Results

A. Application of the procedures to a single origami vertex

Having established two procedures for modelling non-Euclidean origami, they are applied to the non-Euclidean vertices introduced in Figure 1. By doing so, it is possible to make comparisons between the presented modelling methods and state-of-the-art methods, as well as between both novel methods. These comparisons serve as a validation of the methods, but also show their potential.

The procedures are applied to an origami vertex with sector angles of 60°, 90°, 135°, and 75° for α_1, α_2, α_3, and α_4, respectively. To obtain the non-Euclidean vertices, a value of $0.05/2\pi$ rad (or $\approx 0.46°$) is added or subtracted from each sector angle to obtain the non-Euclidean surplus and deficit vertices. The numerical values mentioned above are chosen to be consistent with previous research on non-Euclidean origami [30], [31].

First, a comparison is made between rigid-facet non-Euclidean origami; comparing flexible crease vertices with ideal crease vertices. The dihedral angular relations of the flexible crease vertices are obtained by applying procedure 1 of this paper. The following dimensions for crease width, crease thickness, inner diameter, and outer diameter are used in the FEM analysis: $w_{crease} = 3mm$, $t_{crease} = 0.02mm$, $D_{in} = 8mm$, and $D_{out} = 50mm$. The angular relations of the rigid-body origami vertex are obtained from explicit kinematic equations published in the paper of Foschi et al. [31]. The results of this comparison are shown in Figure 5. The general trends in the dihedral angles from both simulations are consistent with each other. Small differences were found between the results due to unintended crease bending modes. These differences could be magnified by increasing the ratio between the width of the creases and the outer diameter of the vertex. It should also be noted that the absolute values of the dihedral angles were measured. Due to unintended bending modes, the angles never reach a value of zero in reality, leading to some inconsistent behaviour around zero.

A second comparison is made between the two procedures presented in this paper. For this comparison, the same dimensions are used as in the previous FEM analyses. However, one parameter is added for the flexible facet FEM analyses; the facet thickness. A value of $t_{facet} = 3t_{crease} = 0.06mm$ is chosen. The results of the comparison between procedures 1 and 2 are shown in Figure 6. In this instance, it is possible to show not only the differences in dihedral angular relations, but also the difference in elastic energy stored in the vertices.

The elastic energy plots of Figure 6 show that the flexibility of the facets decrease the elastic energy in the system. Adding flexible members enables a more efficient redistribution of strain, reducing internal stress, and thus total elastic energy. Similarly to the results in Figure 5, the discrepancies between the two modelling methods can be increased by changing the key dimensions. In this case, the differences could be magnified by increasing the ratio between the crease thickness and the facet thickness.

B. Case study

After applying the introduced procedures to one origami vertex at a time, a case study is executed showing how procedure 1 could be applied to a more complex non-Euclidean origami mechanism. The simulation of the case study was also experimentally validated.

The case study mechanism is based on the Sarrus mechanism, to which origami pitch hinges, found in the study by Nelson et al. [36], are applied. The resulting mechanism is a linear guide, and thus has one translational degree of freedom. Figure 7 globally shows the assembly steps of the mechanism, starting from a flat state. Note that the mechanism is displayed as a CAD model of a rigid body mechanism here, instead of a compliant mechanism. In the bottom right segment of Figure 7, the FEM simulation is shown in its fully assembled state. The actual assembly procedure of the FEM model is shown in Appendix C.

In order to validate the numerical model, the mechanism is loaded in two ways. First, the mechanism is loaded in its translational degree of freedom to obtain force-deflection data. Secondly, the mechanism is loaded in torsion around the axis of translational freedom. The results of these two load cases are compared to the results obtained from experimental tests, which will be discussed next.

The physical prototypes of the case study mechanism are fabricated as described in the Methods section. The steps for assembling the prototypes can be found in Appendix C.

The experimental validation of the finite-element model is carried out on a combined tension-compression and torsion test bench. The test bench consists of a motorised linear stage and a motorised rotary stage. The stages are equipped with a force load cell sensor and a torque load cell sensor, respectively. To ensure consistent testing, a test plan is set up. The tests are executed with three goals in mind:

1) Validating the force-deflection behaviour of the finite-element model along its translational degree of freedom (tension-compression).
2) Validating the torque-rotation behaviour of the finite-element model along the constraint rotation around its central axis (torsion).
3) Investigating the influence of prolonged pretension on the force-deflection behaviour along its translational degree of freedom. This is done by pretensioning the model by assembling it and testing the model before and after a period of 16 days of pretension.

The test plan includes two test moments. At the first test moment, the prototypes are only loaded in tension-compression. During the second test moment, this loading is repeated, and additionally the prototypes are loaded in torsion.

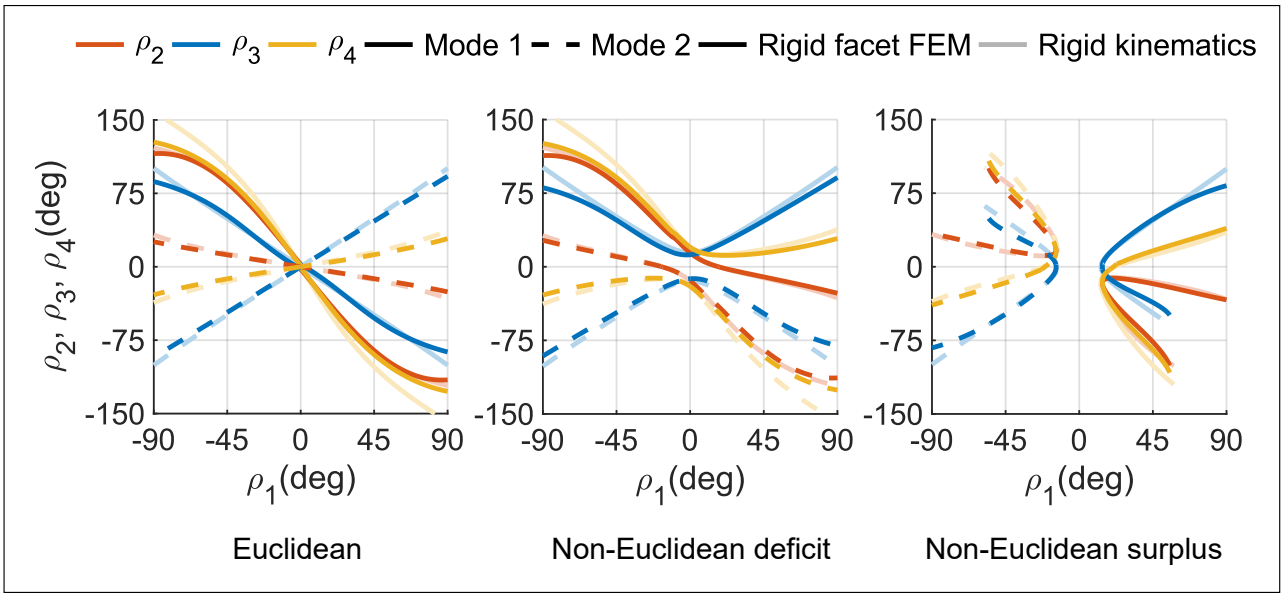

Fig. 5. Angular relations plots of Euclidean and non-Euclidean vertices, comparing rigid facet FEM analyses with rigid kinematic analyses.

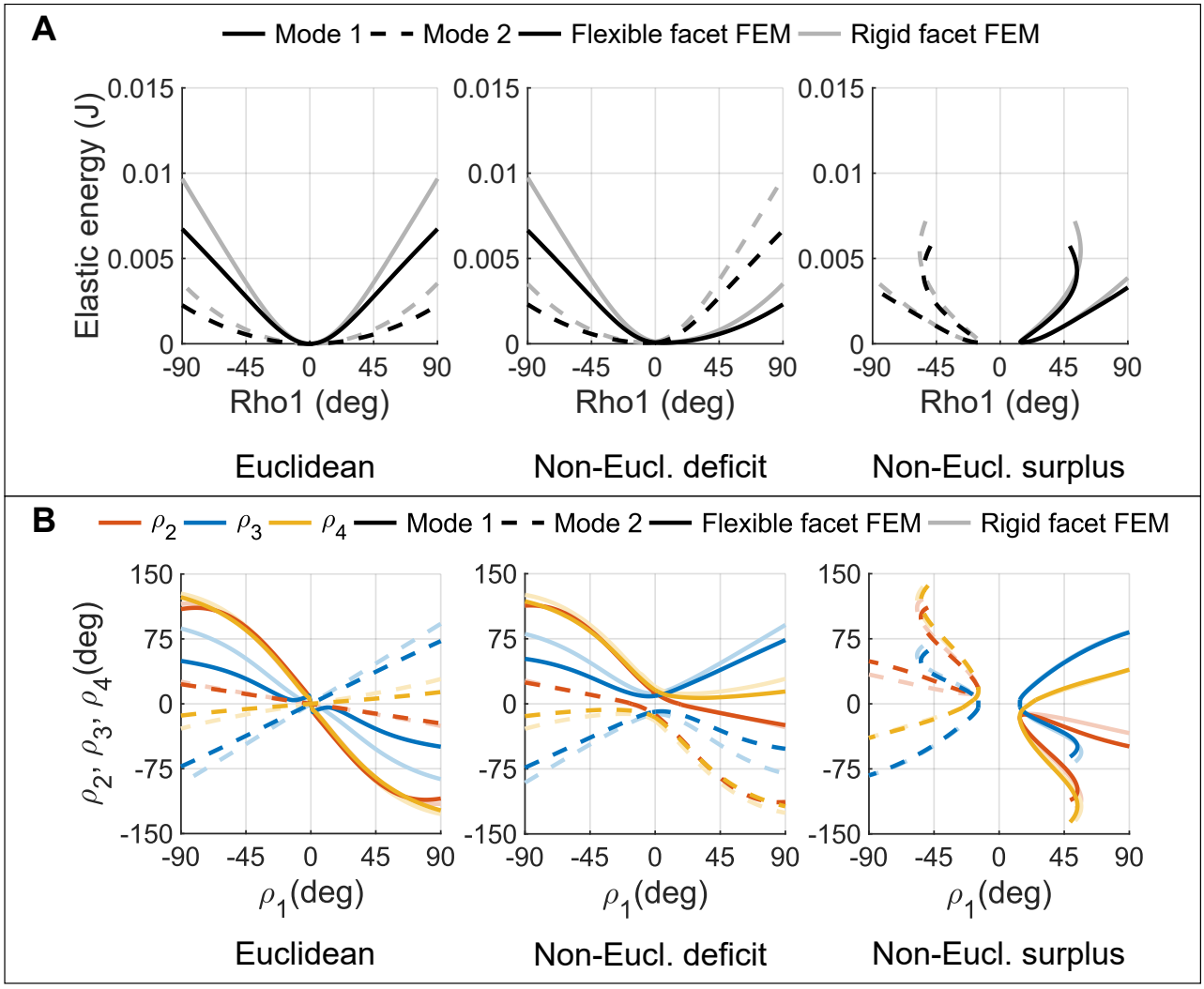

Fig. 6. Energy and angular relations plots of Euclidean and non-Euclidean vertices, comparing flexible facet FEM analyses with rigid facet FEM analyses. A Elastic energy. B Angular relations.

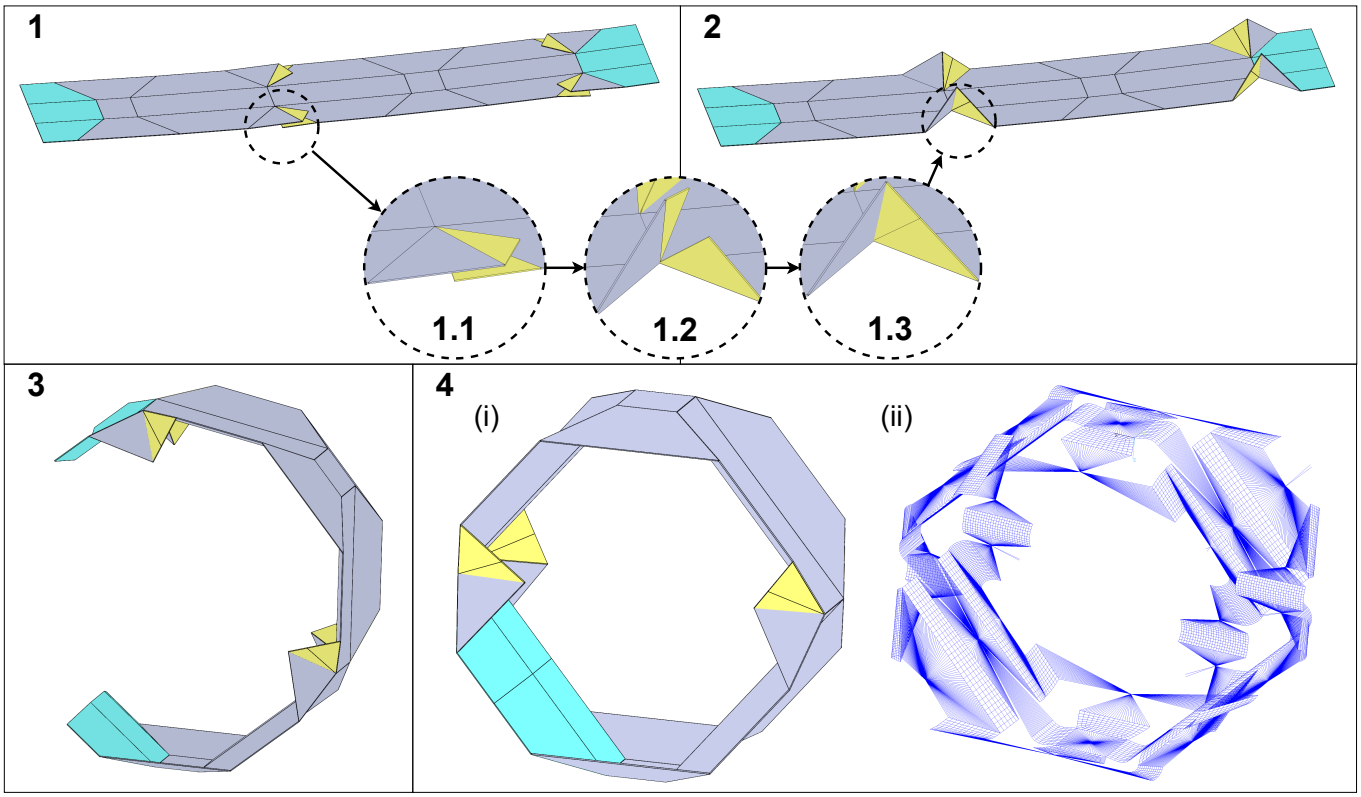

Fig. 7. Simplified assembly procedure of the case study mechanism, displayed with perfect hinge creases instead of flexible creases, concluding with an image of the fully assembled ANSYS APDL model. 1 Origami mechanism in its flat, unassembled state. 1.1 - 1.3 Intermediate steps for assembling the non-Euclidean vertices, by aligning the yellow panels. 2 Origami mechanism with assembled non-Euclidean vertices. 3 Intermediate step in process of connecting the ends of the mechanism, with the goal of aligning the cyan panels. 4 Fully assembled case study mechanism. (i) Case study mechanism with perfect hinge creases. (ii) ANSYS APDL model with flexible creases.

In total, three prototypes have been manufactured and tested. Multiple prototypes are tested to identify the influence of fabrication and assembly inconsistencies on the results. Each prototype is clamped into position three times. This is done to observe whether clamping inconsistencies influence the results. During each clamping set-up, the model is loaded six times. The results of the first run are discarded to allow the prototype to settle. Figure 8 offers an overview of the fabricated model, the experimental setups, and the graphs showing the experimental and numerical results of the case study mechanism. Discrepancies between the data from the numerical model and the experimental test data will be evaluated in the Discussion section, along with the reason why multiple material models were used in the FEM analyses. The test results showed that the pretensioning of the prototypes did not influence the force-deflection behaviour along its translational degree of freedom. The test data per prototype per test moment can be found in Appendix C.

C. Evaluation of introduced procedures

The main contribution of this work is the introduction of two finite element modelling procedures to model flexible non-Euclidean origami. These procedures are discussed in this section, touching upon their possibilities and limitations.

Starting with the most important potential of the modelling methods; the effects resulting from non-Euclidean assembly can be analysed. These effects include internal stresses, influenced force/deflection behaviour, and altered kinematics due to unintended bending modes. All of the effects mentioned above can be monitored during operation, after being affected by the non-Euclidean assembly.

Moving on to limitations of the introduced modelling methods, it has been found that the obtained simulations can suffer from convergence issues due to multistabilities. This proved to be mostly the case when loading mechanisms in constraint directions. A useful tool for solving this problem is the STABILIZE command. However, results obtained while using this command should be carefully checked, as the applied stabiliser can influence them. The STABILIZE command can also be used during the assembly procedure but should be removed after the assembly is complete. Another limitation of the proposed methods is their inability to be used for modal analyses. The couplings applied with the CP command are ignored during modal analyses, leading to meaningless results.

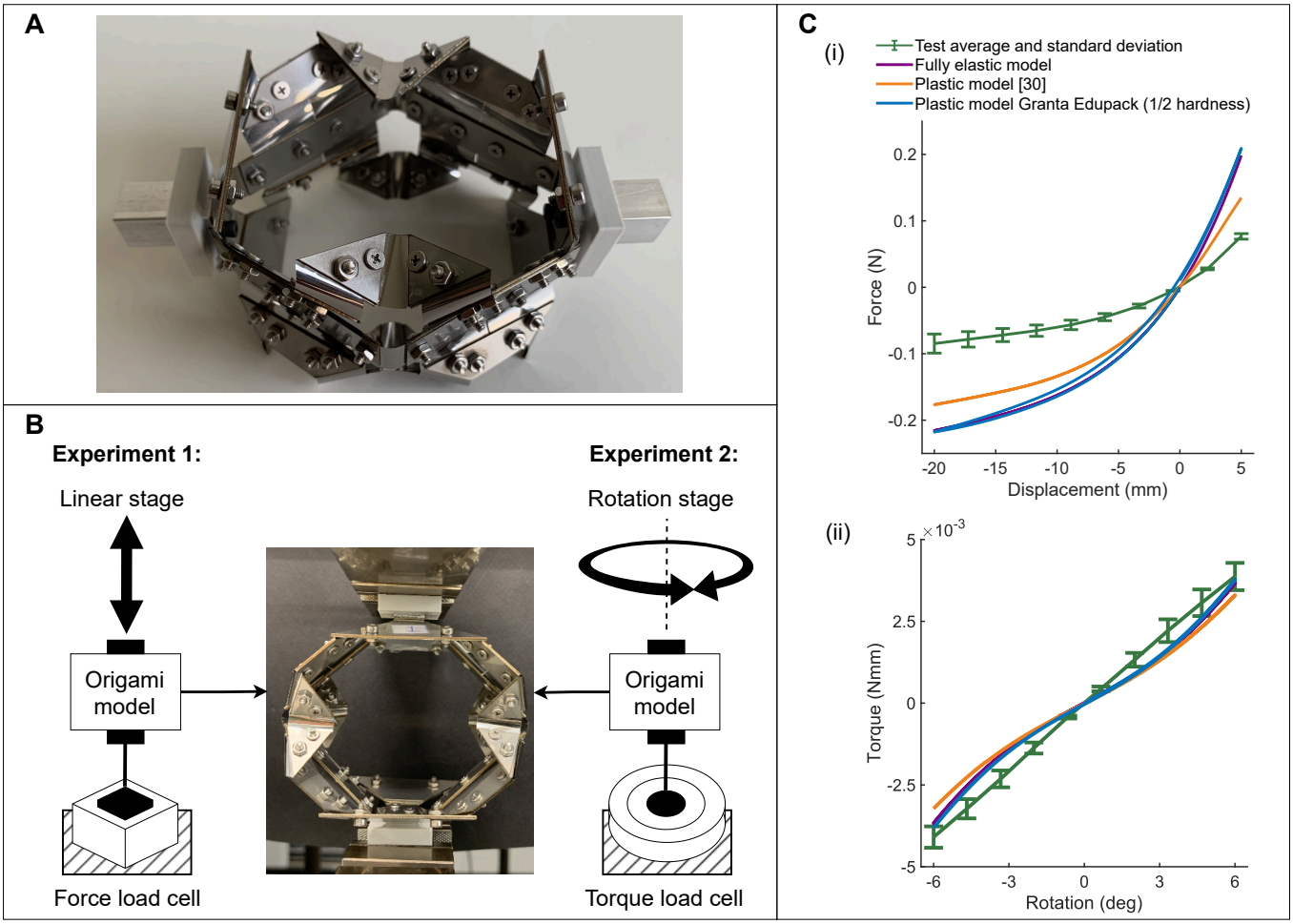

Fig. 8. Overview of fabricated model, experimental setups and result plots of case study mechanism. A Fabricated model including clamping parts. B Experimental setups. Tension-compression on the left, clamped-in model in the middle, and torsion on the right. C Plots displaying numerical and test data. Numerical data consists of three different material models. Test data average is shown in combination with its standard deviation. (i) Tension-compression. (ii) Torsion.

IV. Discussion

A. Discrepancies case study results

Significant discrepancies between the numerical and experimental data are identified in the result plots in Figure 8. These discrepancies are expected to be caused by plastic deformations as a result of assembly. Initially, the mechanism was not meant to deform plastically, but when the physical models were disassembled, it was observed that some plastic deformation occurred. In an attempt to reduce the discrepancy between the test results and the results of the elastic material model, two plastic material models were also evaluated. As the exact stress/strain behaviour of the material is unknown, two bilinear plastic material models were used, using Young's modulus, yield stress, and plastic tangent modulus as parameters. The first plastic model for this steel alloy was found in a work by Pavliuchenko et al. [37], while the other was found using Granta Edupack [38]. The stress/strain plot and parameter values of the three applied material models can be found in Appendix C. Both plastic material models did not produce results that were within the standard deviation of the test results. A third plastic material model could be implemented, with an even lower yield stress. However, this model could not be implemented due to time constraints. A more positive takeaway from these results is that the general trend of the test data was consistent with the trend found using numerical analyses.

In addition to plastic deformations, the test models appeared to be prone to unintended multistabilities. It proved to be difficult to force all prototypes to be in the same state during testing, leading to inconsistencies. Removing these inconsistencies could reduce the standard deviation in the test data.

B. Future work

As a result of this work, several new research challenges have emerged, which are listed below.

1) The second procedure introduced in this paper could be further improved. The coupling of the flexible facets is done by coupling multiple nodes along the seam. Due to this, the panels are perfectly coupled locally, but between the coupled locations the facet

does not behave as one. It should be investigated if it is possible to couple the seam in a more consistent matter. If this is not possible, it should be researched how many local couplings are needed to obtain sufficiently accurate results.

2) As mentioned in the Results section, the introduced modelling methods cannot be used for modal analyses, as the applied couplings are ignored. It might be possible to resolve this limitation by replacing the CP commands with the more general CE commands. As modal analyses were not the focus of this work, it was not attempted to implement this solution. The use of the CE command could also come at the cost of more computationally expensive simulations.
3) Both modelling methods introduced in this work have not yet been sufficiently validated by experiments. Experimental validation should be performed using simple models whose stresses remain well below the plastic deformation limit during assembly and loading. This is important to ensure that plastic deformations do not influence the results.
4) During this research, flexible origami creases were often found to deform in other ways than intended. The intended deformation of a crease is bending around its centreline, while some examples of unintended deformations are torsion or shearing of creases. These unintended deformations can result in unwanted stress concentrations and should therefore be avoided. Alternate creases could be implemented to reduce undesired crease deformations. This can be achieved by reducing the stiffness of the joint in bending and increasing the stiffness in other deformation modes, effectively forcing a certain deformation mode. In the case study mechanism, this was done by changing the 2D shape of its creases. Another way to avoid unintended crease bending modes is to implement more complex creases. Some options for this are lamina emergent joints [39], [40] or 3D compliant joints, such as circular groove joints [41]. The implementation of 3D compliant joints would negate the 2D manufacturability of flexible origami.
5) As a final recommendation for future work, the procedures presented in this paper should be applied to functional origami. Whether it is to approach previous research [32], [42]–[44] in a new way or to analyse new mechanisms, the application of these procedures is the best way to build on the foundation laid in this research.

V. Conclusion

The procedures introduced in this paper can be used to numerically analyse the mechanical responses of flexible non-Euclidean origami, after constructing it in 2D and virtually assembling it. This opens the door for using 2D fabrication techniques to make flexible non-Euclidean origami, which is simpler and cheaper than using 3D fabrication techniques.

The numerical modelling methods were validated by applying them to a flexible non-Euclidean vertex and comparing its dihedral angular relations with those of a vertex with ideal hinges. An attempt was also made to experimentally validate the first modelling method by applying it to a case study mechanism. The general trend of the force/deflection data was consistent for the experiment and the numerical model, but a discrepancy was found. This discrepancy is expected to be caused by plastic deformation, but should be further investigated. During analyses of flexible non-Euclidean origami, it was found that creases often deform differently than intended. These unintended bending modes can be detrimental to the operation of flexible origami and should be avoided by smart crease design.

The finite element modelling procedures presented in this paper are a step towards the implementation of flexible non-Euclidean origami in functional applications. The deployability of origami is highly valuable for applications where space efficiency and adaptability are essential. Non-Euclidean origami enables the splitting of folding branches, leading to mechanisms with kinematically determinate behaviour. Research into multistable mechanisms can also benefit from the procedures presented in this paper, as non-Euclidean origami contains implicit multistabilities that could be employed.

References

[1] T. Ye, Z. Chen, S. Huang, and F. Hu, "Review of Deployable Mechanism Test Technologies Oriented towards Deep Space Exploration," in IOP Conference Series: Materials Science and Engineering, vol. 1043, no. 5. IOP Publishing Ltd, 2 2021.

[2] B. Wang, J. Zhu, S. Zhong, W. Liang, and C. Guan, "Space deployable mechanics: A review of structures and smart driving," 1 2024.

[3] Z. Q. Liu, H. Qiu, X. Li, and S. L. Yang, "Review of Large Spacecraft Deployable Membrane Antenna Structures," pp. 1447–1459, 11 2017.

[4] X. Ma, T. Li, J. Ma, Z. Wang, C. Shi, S. Zheng, Q. Cui, X. Li, F. Liu, H. Guo, L. Liu, Z. Wang, and Y. Li, "Recent Advances in Space-Deployable Structures in China," pp. 207–219, 10 2022.

[5] L. Santo, F. Quadrini, A. Accettura, and W. Villadei, "Shape memory composites for self-deployable structures in aerospace applications," in Procedia Engineering, vol. 88. Elsevier Ltd, 2014, pp. 42–47.

[6] W. K. Belvin, M. Straubel, W. Keats Wilkie, M. E. Zander, J. M. Fernandez, and M. F. Hillebrandt, "Advanced Deployable Structural Systems for Small Satellites," Tech. Rep., 2016.

[7] S. Yue, "A Review of Origami-Based Deployable Structures in Aerospace Engineering," in Journal of Physics: Conference Series, vol. 2459, no. 1. Institute of Physics, 2023.

[8] K. Kuribayashi, K. Tsuchiya, Z. You, D. Tomus, M. Umemoto, T. Ito, and M. Sasaki, "Self-deployable origami stent grafts as a biomedical application of Ni-rich TiNi shape memory alloy foil," Materials Science and Engineering: A, vol. 419, no. 1-2, pp. 131–137, 3 2006.

[9] M. H. Akhtar and J. Ramkumar, "Origami inspired deployable structures: Future mobile healthcare for low resource settings." Alanya Hamdullah Emin Pasa Universitesi, 6 2023, pp. 209–219.

[10] Jahanshah Fathi, Timo J. C. Oude Vrielink, Mark S. Runciman, and George P. Mylonas, "A Deployable Soft Robotic Arm with Stiffness Modulation for Assistive Living Applications," 2019.

[11] W. He, D. Zhou, H. Gu, R. Qu, C. Cui, Y. Zhou, Y. Wang, X. Zhang, Q. Wang, T. Wang, and Y. Zhang, "A Biocompatible 4D Printing Shape Memory Polymer as Emerging Strategy for Fabrication of Deployable Medical Devices," Macromolecular Rapid Communications, vol. 44, no. 2, 1 2023.

[12] J. Gafford, Y. Ding, A. Harris, T. McKenna, P. Polygerinos, D. Holland, A. Moser, and C. Walsh, "Shape deposition manufacturing of a soft, atraumatic, deployable surgical grasper," Journal of Medical Devices, Transactions of the ASME, vol. 8, no. 3, 2014.

[13] G. E. Fenci and N. G. Currie, "Deployable structures classification: A review," pp. 112–130, 6 2017.

[14] C. Ynchausti, C. Roubicek, J. Erickson, B. Sargent, S. P. Magleby, and L. L. Howell, "Hexagonal Twist Origami Pattern for Deployable Space Arrays," ASME Open Journal of Engineering, vol. 1, 1 2022.

[15] K. Seymour, D. Burrow, A. Avila, T. Bateman, D. C. Morgan, S. P. Magleby, and L. L. Howell, "Origami-Based Deployable Ballistic Barrier," Tech. Rep., 2018. [Online]. Available: https://scholarsarchive.byu.edu/facpub

[16] Y. Zhu and E. T. Filipov, "Large-scale modular and uniformly thick origami-inspired adaptable and load-carrying structures," Nature Communications, vol. 15, no. 1, 12 2024.

[17] B. Sargent, J. Butler, K. Seymour, D. Bailey, B. Jensen, S. Magleby, and L. Howell, "An Origami-Based Medical Support System to Mitigate Flexible Shaft Buckling," Journal of Mechanisms and Robotics, vol. 12, no. 4, 8 2020.

[18] A. J. Taylor, Y. Chen, M. Fok, A. Berman, K. Nilsson, and Z. T. H. Tse, "Cardiovascular catheter with an expandable origami structure," Journal of Medical Devices, Transactions of the ASME, vol. 11, no. 3, 9 2017.

[19] S. J. Wu, H. Yuk, J. Wu, C. S. Nabzdyk, and X. Zhao, "A Multifunctional Origami Patch for Minimally Invasive Tissue Sealing," Advanced Materials, vol. 33, no. 11, 3 2021.

[20] H. C. Greenberg, M. L. Gong, S. P. Magleby, and L. L. Howell, "Identifying links between origami and compliant mechanisms," Mechanical Sciences, vol. 2, no. 2, pp. 217–225, 2011.

[21] Y. Feng, M. Wang, and X. Qiu, "A simplified mechanical model of the crease in the flexible origami structures," International Journal of Solids and Structures, vol. 241, 4 2022.

[22] J. Liu, Z. Chen, G. Wen, J. He, H. Wang, L. Xue, K. Long, and Y. M. Xie, "Origami Chomper-Based Flexible Gripper with Superior Gripping Performances," Advanced Intelligent Systems, vol. 5, no. 10, 10 2023.

[23] C. M. Wheeler and M. L. Culpepper, "Soft origami: Classification, constraint, and actuation of highly compliant origami structures," Journal of Mechanisms and Robotics, vol. 8, no. 5, 10 2016.

[24] L. Zimmermann, K. Shea, and T. Stanković, "Conditions for Rigid and Flat Foldability of Degree-n Vertices in Origami," Journal of Mechanisms and Robotics, vol. 12, no. 1, 2 2020.

[25] L. Zimmermann and T. Stanković, "Rigid and Flat Foldability of a Degree-Four Vertex in Origami," Journal of Mechanisms and Robotics, vol. 12, no. 1, 2 2020.

[26] E. T. Filipov, T. Tachi, G. H. Paulino, and D. A. Weitz, "Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials," Proceedings of the National Academy of Sciences of the United States of America, vol. 112, no. 40, pp. 12 321–12 326, 10 2015.

[27] L. . Zimmermann, K. . Shea, and T. Stankovic, "A Computational Design Synthesis Method for the Generation of Rigid Origami Crease Patterns," Journal of Mechanisms and Robotics, vol. 14, no. 3, 2021. [Online]. Available: https://doi.org/10.3929/ethz-b-000512541

[28] Y. Zhu, M. Schenk, and E. T. Filipov, "A Review on Origami Simulations: From Kinematics, to Mechanics, Toward Multiphysics," Applied Mechanics Reviews, vol. 74, no. 3, 5 2022.

[29] T. Tachi and T. C. Hull, "Self-foldability of rigid origami," Journal of Mechanisms and Robotics, vol. 9, no. 2, 4 2017.

[30] S. Waitukaitis, P. Dieleman, and M. Van Hecke, "Non-Euclidean Origami," Tech. Rep., 2020.

[31] R. Foschi, T. C. Hull, and J. S. Ku, "Explicit kinematic equations for degree-4 rigid origami vertices, Euclidean and non-Euclidean," Physical Review E, vol. 106, no. 5, 11 2022.

[32] Y. Li, P. Wang, Q. Zhang, K. Li, Y. Zhang, L. Kan, W. Xin, J. Feng, J. Cai, and C. Laschi, "Robots Evolved from Non-Euclidean Composite Origami," IEEE Robotics and Automation Letters, 2025.

[33] C. C. Addis, S. Rojas, and A. F. Arrieta, "Connecting the branches of multistable non-Euclidean origami by crease stretching," Physical Review E, vol. 108, no. 5, 11 2023.

[34] N. Bende, "Non-Euclidean Shells: A Study of Growth-Induced Fabrication and Mechanical Multi-Stability Item Type dissertation," 2017. [Online]. Available: https://hdl.handle.net/20.500.14394/17197

[35] M. K. Thompson and J. M. Thompson, ANSYS mechanical APDL for finite element analysis. Butterworth-Heinemann, 2017.

[36] T. G. Nelson, A. Avila, L. L. Howell, J. L. Herder, and D. F. Machekposhti, "Origami-inspired sacrificial joints for folding compliant mechanisms," Mechanism and Machine Theory, vol. 140, pp. 194–210, 10 2019.

[37] P. Pavliuchenko, M. Teller, and G. Hirt, "Analysis of influencing factors on the achievability of bistable fully closed shells by semi-analytical modelling," 3 2021. [Online]. Available: https://engrxiv.org/index.php/engrxiv/preprint/view/1541

[38] ANSYS, "Granta EduPack 2022 R1," 2022. [Online]. Available: https://www.ansys.com/products/materials/granta-edupack

[39] I. L. Delimont, S. P. Magleby, and L. L. Howell, "Evaluating compliant hinge geometries for origami-inspired mechanisms," Journal of Mechanisms and Robotics, vol. 7, no. 1, 2015.

[40] J. Keizer, "Design of a lamina emergent joint as an alternative for a groove joint," Tech. Rep., 2023.

[41] D. F. Machekposhti, N. Tolou, and J. L. Herder, "A review on compliant joints and rigid-body constant velocity universal joints toward the design of compliant homokinetic couplings," 2015.

[42] L. Huang, P. Zeng, L. Yin, B. Liu, Y. Yang, and J. Huang, "Design and kinematic analysis of a rigid-origami-based underwater sampler with deploying-encircling motion," Mechanism and Machine Theory, vol. 174, 8 2022.

[43] C. Shi, Q. Zhang, J. Feng, and J. Cai, "Mechanical performance of reconfigurable origami structures fabricated by cutting and planar assembly," Extreme Mechanics Letters, vol. 77, 6 2025.

[44] C. Shi, Q. Zhang, J. Feng, S. D. Kim, and J. Cai, "Tensile self-locking behavior of reconfigurable origami structure," Engineering Structures, vol. 335, 7 2025.

A

Detailed explanation of numerical procedure 1

This appendix includes the ANSYS Mechanical APDL code used to model a single rigid facet non-Euclidean vertex. The code is explained in blocks and contains comments in the script to guide the reader.

The first block of code contains the commands to define the element types (including key options), material model, and section used for the flexible creases.

```
1  FINISH !exit current processor
2  /CLEAR !clear APDL database
3
4  /PREP7 !initialize preprocessor
5
6    !!!=== Define element types and material properties
7
8  ET,1,281 !Element for panels (SHELL281)
9  ET, 2,MPC184 !Element for rigid connections
10 KEYOPT, 2, 1, 1 !Last number defines whether it is a link (0) or beam (1). Should always be
       beam for this application.
11 KEYOPT, 2, 2, 1 !Change kinematic constraint method to be able to use coupling (to Lagrange
       multiplier method)
12 MP,ex,1,1.85*10**11 !Define Young's Modulus
13 MP,nuxy,1,0.3 !Define Poisson's ratio
14
15   !!!=== Define section of creases
16
17 t=(0.02)*10**(-3) !Thickness used for creases
18
19 SECTYPE,1,shell,, !section of crease line
20 SECDATA,t,,,5 !defines the thickness, material ID, angle (for anisotropic materials) and
       number of integration points in layer, respectively.
21 SECOFFSET,MID !defines the section offset
```

Listing A.1: Definition of element types (including key options), material model, and section.

The second block of code contains the commands to define the geometry of the unassembled non-Euclidean vertex. The geometry is built up using keypoints, lines, and areas.

```
22   !!!=== Define geometry of the model
23
24 pi=3.14159265 !Definition of pi
25 Rsmall = 4*10**(-3) !Radius of inner circle of the vertex
26 Rbig = 25*10**(-3) !Radius of outer circle of the vertex
27 Creasewidth = 3*10**(-3) !Width of the creases of the vertex
28
29 alpha1_deg= 60 !Sector angle of the first facet
```

```
alpha1=(alpha1_deg/360)*2*pi !Conversion of alpha1 to radians

alpha2_deg= 90 !Sector angle of the second facet
alpha2=(alpha2_deg/360)*2*pi !Conversion of alpha2 to radians

alpha3_deg= 135 !Sector angle of the third facet
alpha3=(alpha3_deg/360)*2*pi !Conversion of alpha3 to radians

alpha4_deg= 75 !Sector angle of the fourth facet (the facet that is split)
alpha4=(alpha4_deg/360)*2*pi !Conversion of alpha4 to radians

alphasurplus_deg= -5 !Angle that is added or subtracted to each sector angle to get the non-Euclidean vertex. Negative for deficit, positive for surplus
alphasurplus=(alphasurplus_deg/360)*2*pi !Conversion of Alpha to radians

  !Intermediate variables, used to construct the model
Sdiagonal = sqrt((Rsmall)**2-(Creasewidth/2)**2)
Sdiagonal2 = sqrt((Rbig)**2-(Creasewidth/2)**2)

Radialline = Rbig - Rsmall - (Rbig-Sdiagonal2) + (Rsmall-Sdiagonal)

effective_angle3 = alpha1 + alpha2 + alpha3 + 3*alphasurplus - 3*pi/2


X3 = Creasewidth/2*sin(alpha1 + alphasurplus) + Sdiagonal*cos(alpha1 + alphasurplus)
Y3 = -Creasewidth/2*cos(alpha1 + alphasurplus) + Sdiagonal*sin(alpha1 + alphasurplus)

X4 = -Creasewidth/2*sin(alpha1 + alphasurplus) + Sdiagonal*cos(alpha1 + alphasurplus)
Y4 = Creasewidth/2*cos(alpha1 + alphasurplus) + Sdiagonal*sin(alpha1 + alphasurplus)

X5 = Creasewidth/2*cos(alpha1 + alpha2 + 2*alphasurplus - pi/2) - Sdiagonal*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2)
Y5 = Creasewidth/2*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2) + Sdiagonal*cos(alpha1 + alpha2 + 2*alphasurplus - pi/2)

X6 = -Creasewidth/2*cos(alpha1 + alpha2 + 2*alphasurplus - pi/2) - Sdiagonal*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2)
Y6 = -Creasewidth/2*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2) + Sdiagonal*cos(alpha1 + alpha2 + 2*alphasurplus - pi/2)

X7 = -Creasewidth/2*cos(-effective_angle3) - Sdiagonal*sin(-effective_angle3)
Y7 = Creasewidth/2*sin(-effective_angle3) - Sdiagonal*cos(-effective_angle3)

X8 = Creasewidth/2*cos(-effective_angle3) - Sdiagonal*sin(-effective_angle3)
Y8 = -Creasewidth/2*sin(-effective_angle3) - Sdiagonal*cos(-effective_angle3)


  !!!=== Keypoints

K,1, Sdiagonal, -Creasewidth/2, 0
K,2, Sdiagonal, Creasewidth/2, 0

K,3, X3, Y3, 0
K,4, X4, Y4, 0

K,5, X5, Y5, 0
K,6, X6, Y6, 0

K,7, X7, Y7, 0
K,8, X8, Y8, 0

K,9, Rsmall*sin((alpha4 + alphasurplus)/2 + effective_angle3), -Rsmall*cos((alpha4 + alphasurplus)/2 + effective_angle3), 0
K,10, Rsmall*cos((alpha4 + alphasurplus)/2), -Rsmall*sin((alpha4 + alphasurplus)/2), 0

K,11, Sdiagonal + Radialline, -Creasewidth/2, 0
K,12, Sdiagonal + Radialline, Creasewidth/2, 0

K,13, X3 + Radialline*cos(alpha1 + alphasurplus), Y3 + Radialline*sin(alpha1 + alphasurplus), 0
K,14, X4 + Radialline*cos(alpha1 + alphasurplus), Y4 + Radialline*sin(alpha1 + alphasurplus),
```

```
     0
94
95 K,15, X5 - Radialline*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2), Y5 + Radialline*cos(
       alpha1 + alpha2 + 2*alphasurplus - pi/2), 0
96 K,16, X6 - Radialline*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2), Y6 + Radialline*cos(
       alpha1 + alpha2 + 2*alphasurplus - pi/2), 0
97
98 K,17, X7 - Radialline*sin(-effective_angle3), Y7 - Radialline*cos(-effective_angle3), 0
99 K,18, X8 - Radialline*sin(-effective_angle3), Y8 - Radialline*cos(-effective_angle3), 0
100
101 K,19, (Rbig)*sin((alpha4 + alphasurplus)/2 + effective_angle3), -(Rbig)*cos((alpha4 +
       alphasurplus)/2 + effective_angle3), 0
102 K,20, (Rbig)*cos((alpha4 + alphasurplus)/2), -(Rbig)*sin((alpha4 + alphasurplus)/2), 0
103
104 K,21, 0,0,0 !Origin, used for center of curvature
105
106
107   !!!=== Lines between keypoints
108
109 !First ring arcs
110 LARC, 1, 2, 21, Rsmall
111 LARC, 2, 3, 21, Rsmall
112 LARC, 3, 4, 21, Rsmall
113 LARC, 4, 5, 21, Rsmall
114 LARC, 5, 6, 21, Rsmall
115 LARC, 6, 7, 21, Rsmall
116 LARC, 7, 8, 21, Rsmall
117 LARC, 8, 9, 21, Rsmall
118 LARC, 1, 10, 21, Rsmall
119
120
121 !Second ring arcs
122 LARC, 11, 12, 21, Rbig
123 LARC, 12, 13, 21, Rbig
124 LARC, 13, 14, 21, Rbig
125 LARC, 14, 15, 21, Rbig
126 LARC, 15, 16, 21, Rbig
127 LARC, 16, 17, 21, Rbig
128 LARC, 17, 18, 21, Rbig
129 LARC, 18, 19, 21, Rbig
130 LARC, 11, 20, 21, Rbig
131
132 !Radial lines
133 L,1,11
134 L,2,12
135 L,3,13
136 L,4,14
137 L,5,15
138 L,6,16
139 L,7,17
140 L,8,18
141 L,9,19
142 L,10,20
143
144   !!!=== Creating areas between lines
145
146 AL,1,10,19,20 !Area 1 (crease)
147 AL,2,11,20,21 !Area 2 (rigid facet)
148 AL,3,12,21,22 !Area 3 (crease)
149 AL,4,13,22,23 !Area 4 (rigid facet)
150 AL,5,14,23,24 !Area 5 (crease)
151 AL,6,15,24,25 !Area 6 (rigid facet)
152 AL,7,16,25,26 !Area 7 (crease)
153 AL,8,17,26,27 !Area 8 (rigid facet)
154 AL,9,18,19,28 !Area 9 (rigid facet)
```

Listing A.2: Building the geometry of the unassembled non-Euclidean vertex.

The third block of code contains the commands to create the central nodes and alignment nodes of the rigid panels, mesh the crease areas, and create rigid connections to the central nodes and alignment

nodes. After this block of code, a screenshot shows how the model looks after the preparation steps.

```
  !!!=== Create component containing the flexible areas (creases)

ASEL,S,AREA,,1,7,2
CM,flexible_areas,AREA !Create a component of the selected areas
ALLSEL !select everything (so reset selection done before)

  !!!=== Create central and aligning nodes used for rigid facets

!Base & other pre-assembled facets
N, 1, Rbig*cos((alpha1 + alphasurplus)/2), Rbig*sin((alpha1 + alphasurplus)/2), 0 !Central node for base rigid facet
N, 2, -Rbig*sin((alpha1 + alphasurplus) + (alpha2 + alphasurplus)/2 - pi/2), Rbig*cos((alpha1 + alphasurplus) + (alpha2 + alphasurplus)/2 - pi/2), 0 !Central node for second rigid facet
N, 3, -Rbig*cos((alpha1 + alphasurplus) + (alpha2 + alphasurplus) + (alpha3 + alphasurplus)/2 - pi), -Rbig*sin((alpha1 + alphasurplus) + (alpha2 + alphasurplus) + (alpha3 + alphasurplus)/2 - pi), 0 !Central node for third rigid facet


!Alignment nodes for panel A
N, 4, Rbig*cos((alpha4 + alphasurplus)/2), -Rbig*sin((alpha4 + alphasurplus)/2), 0 !A1
N, 5, (Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + pi/24), -(Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + pi/24), 0 !A2
N, 6, (Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 - pi/24), -(Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 - pi/24), 0 !A3

!Alignment nodes for panel B
N, 7, Rbig*sin((alpha4 + alphasurplus)/2 + effective_angle3), -Rbig*cos((alpha4 + alphasurplus)/2 + effective_angle3), 0 !B1
N, 8, (Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3 - pi/24), -(Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + effective_angle3 - pi/24), 0 !B2
N, 9, (Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3 + pi/24), -(Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + effective_angle3 + pi/24), 0 !B3

  !!!=== Mesh crease areas

CMSEL, S, flexible_areas !Select crease areas
AATT, 1, , 1, , 1 !Define associates element attributes with the selected, unmeshed areas. Entries are material ID, real constants, element type, coordinate system and section number respectively
ESIZE, Creasewidth/6 !Define element size as a function of crease width
AMESH,ALL !Mesh all selected areas
ALLSEL !Select everything (so reset selection done before)

    !!!=== Creating rigid connections between creases to central nodes to create the rigid panels

TYPE,2 !Select rigid beam element for creating new elements

  !!!=== Base facet

  !Select the nodes associated to the lines bordering a rigid facet
LSEL,S,LINE,,20,21 !Select lines surrounding rigid facet
NSLL,S,1 !Selects the nodes associated with the selected lines. Choose 1 to include nodes at the end of the lines.

*GET,base_surrounding_nodes_count,NODE,0,count !Get the number of nodes (count) in the selected set.
*DIM,base_surrounding_nodes_IDs,array,base_surrounding_nodes_count !Make parameter to store node IDs of the selected nodes
*VGET,base_surrounding_nodes_IDs,NODE,,nlist !Get the node IDs of selected nodes and store them in the parameter
ALLSEL !Select everything (so reset selection done before)

  !Loop to make rigid connections between the surrounding nodes to the central node of the rigid facet
*DO,i,1,base_surrounding_nodes_count,1 !Do loop continues for as long as we have surrounding nodes
      E, 1, base_surrounding_nodes_IDs(i) !Rigid connection is made between central node to
```

```
                all surrounding nodes
*ENDDO

  !facet 2

  !Select the nodes associated to the lines bordering a rigid facet
LSEL,S,LINE,,22,23 !Select lines surrounding rigid facet
NSLL,S,1 !Selects the nodes associated with the selected lines. Choose 1 to include nodes at
    the end of the lines.

*GET,facet2_surrounding_nodes_count,NODE,0,count !Get the number of nodes (count) in the
    selected set.
*DIM,facet2_surrounding_nodes_IDs,array,facet2_surrounding_nodes_count !Make parameter to
    store node IDs of the selected nodes
*VGET,facet2_surrounding_nodes_IDs,NODE,,nlist !Get the node IDs of selected nodes and store
    them in the parameter
ALLSEL !Select everything (so reset selection done before)


  !Loop to make rigid connections between the surrounding nodes to the central node of the
      rigid facet
*DO,i,1,facet2_surrounding_nodes_count,1 !Do loop continues for as long as we have
    surrounding nodes
        E, 2, facet2_surrounding_nodes_IDs(i) !Rigid connection is made between central node
            to all surrounding nodes
*ENDDO

  !facet 3

  !Select the nodes associated to the lines bordering a rigid facet
LSEL,S,LINE,,24,25 !Select lines surrounding rigid facet
NSLL,S,1 !Selects the nodes associated with the selected lines. Choose 1 to include nodes at
    the end of the lines.

*GET,facet3_surrounding_nodes_count,NODE,0,count !Get the number of nodes (count) in the
    selected set.
*DIM,facet3_surrounding_nodes_IDs,array,facet3_surrounding_nodes_count !Make parameter to
    store node IDs of the selected nodes
*VGET,facet3_surrounding_nodes_IDs,NODE,,nlist !Get the node IDs of selected nodes and store
    them in the parameter
ALLSEL !Select everything (so reset selection done before)


  !Loop to make rigid connections between the surrounding nodes to the central node of the
      rigid facet
*DO,i,1,facet3_surrounding_nodes_count,1 !Do loop continues for as long as we have
    surrounding nodes
        E, 3, facet3_surrounding_nodes_IDs(i) !Rigid connection is made between central node
            to all surrounding nodes
*ENDDO

  !!!=== Panels to couple

  !panel A

  !Select the nodes associated to the lines bordering a rigid panel
LSEL,S,LINE,,19 !Select lines surrounding rigid panel
NSLL,S,1 !Selects the nodes associated with the selected lines. Choose 1 to include nodes at
    the end of the lines.

*GET,panelA_surrounding_nodes_count,NODE,0,count !Get the number of nodes (count) in the
    selected set.
*DIM,panelA_surrounding_nodes_IDs,array,panelA_surrounding_nodes_count !Make parameter to
    store node IDs of the selected nodes
*VGET,panelA_surrounding_nodes_IDs,NODE,,nlist !Get the node IDs of selected nodes and store
    them in the parameter
ALLSEL !Select everything (so reset selection done before)


  !Loop to make rigid connections between the surrounding nodes to the central node of the
      rigid panel
```

```
256 *DO,i,1,panelA_surrounding_nodes_count,1 !Do loop continues for as long as we have
        surrounding nodes
257         E, 4, panelA_surrounding_nodes_IDs(i) !Rigid connection is made between central node
                to all surrounding nodes
258 *ENDDO
259
260   !panel B
261
262   !Select the nodes associated to the lines bordering a rigid panel
263 LSEL,S,LINE,,26 !Select lines surrounding rigid panel
264 NSLL,S,1 !Selects the nodes associated with the selected lines. Choose 1 to include nodes at
        the end of the lines.
265
266 *GET,panelB_surrounding_nodes_count,NODE,0,count !Get the number of nodes (count) in the
        selected set.
267 *DIM,panelB_surrounding_nodes_IDs,array,panelB_surrounding_nodes_count !Make parameter to
        store node IDs of the selected nodes
268 *VGET,panelB_surrounding_nodes_IDs,NODE,,nlist !Get the node IDs of selected nodes and store
        them in the parameter
269 ALLSEL !Select everything (so reset selection done before)
270
271
272   !Loop to make rigid connections between the surrounding nodes to the central node of the
        rigid panel
273 *DO,i,1,panelB_surrounding_nodes_count,1 !Do loop continues for as long as we have
        surrounding nodes
274         E, 7, panelB_surrounding_nodes_IDs(i) !Rigid connection is made between central node
                to all surrounding nodes
275 *ENDDO
276
277   !Create extra connections to second and third alignment nodes to align and couple panels A
        and B
278 E, 4, 5
279 E, 4, 6
280
281 E, 7, 8
282 E, 7, 9
283
284 FINISH !exit current processor
285
286       !!!=== Preparation complete
```

Listing A.3: Creating central and alignment nodes, mesh the crease areas, and create rigid connections between nodes.

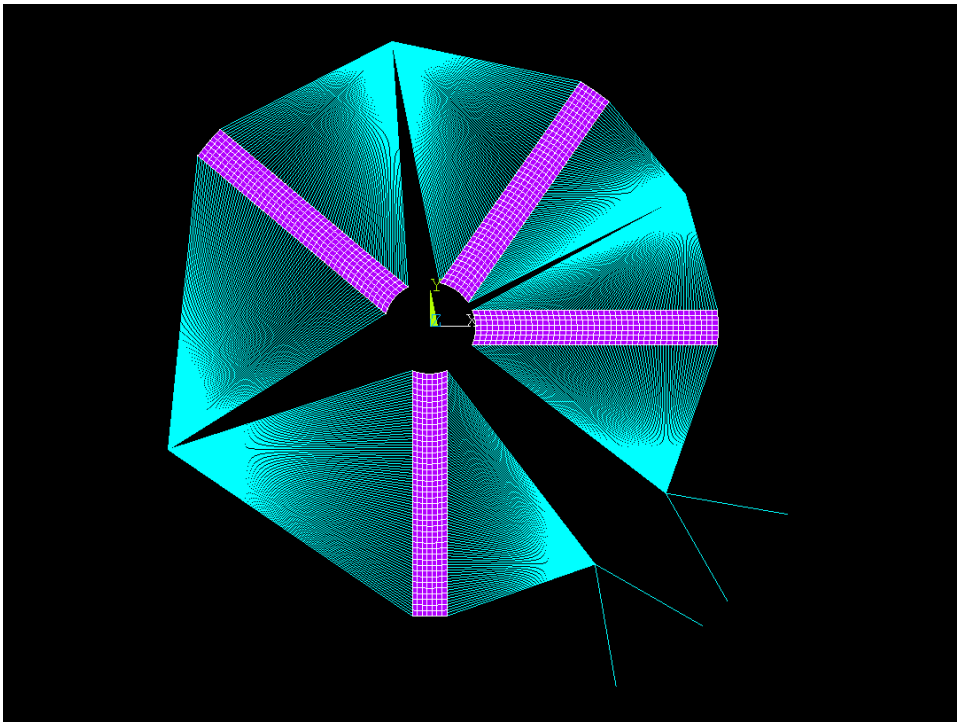

Figure A.1: Screenshot of the model after the preparation steps of procedure 1.

The last blocks of code contain the commands to stepwise assemble the vertex. The step numbers before each loadstep correspond to the step numbers in the main paper. In between the code, there are screenshots of how the model looks after every main displacement step.

```
287       !!!=== Non-Euclidean assembly starting here
288
289   !!!=== Step 1 of the procedure
290 /SOLU   !Initialize solution processor
291 ANTYPE, 0, new !Specifies a new static analysis
292 NLGEOM, on !Non-linear large deflection behavior on or off, including stress stiffening
293 OUTRES,all,all !Controls the solution data written to the database. All solution items at all
      substeps are written to the database.
294 AUTOTS, on !Use automatic time stepping
295 NEQIT, 1000 !Specifies the maximum number of equilibrium iterations for nonlinear analyses.
296 NSUBST,30,,10 !Specifies the size of substeps. If automatic time stepping is on, the second
    and third numbers specify the maximum and minimum amount of substeps.
297
298 D,1,ALL,0 !Fix the base node in 6 DOFs
299
300 D,4,ROTX, -pi/6 !Rotate the central nodes A1 and B1 out of their original plane and toward
    each other. The closer they get, the better it is.
301 D,7,ROTX, -pi/6 !Rotate the central nodes A1 and B1 out of their original plane and toward
    each other. The closer they get, the better it is.
302
303 SOLVE   !solve current study
304 FINISH !exit current processor
305
306 /POST1    !Initialize post1 processor
307 /DSCALE,ALL,1   !Scaling of displacement displays, set to 1 for true scale and 0 for auto
    scale
308 PLDISP,0  !Displays the displaced structure, key is used to show or not show the undisplaced
    structure
```

Listing A.4: Loadstep 1 of assembling the non-Euclidean vertex.

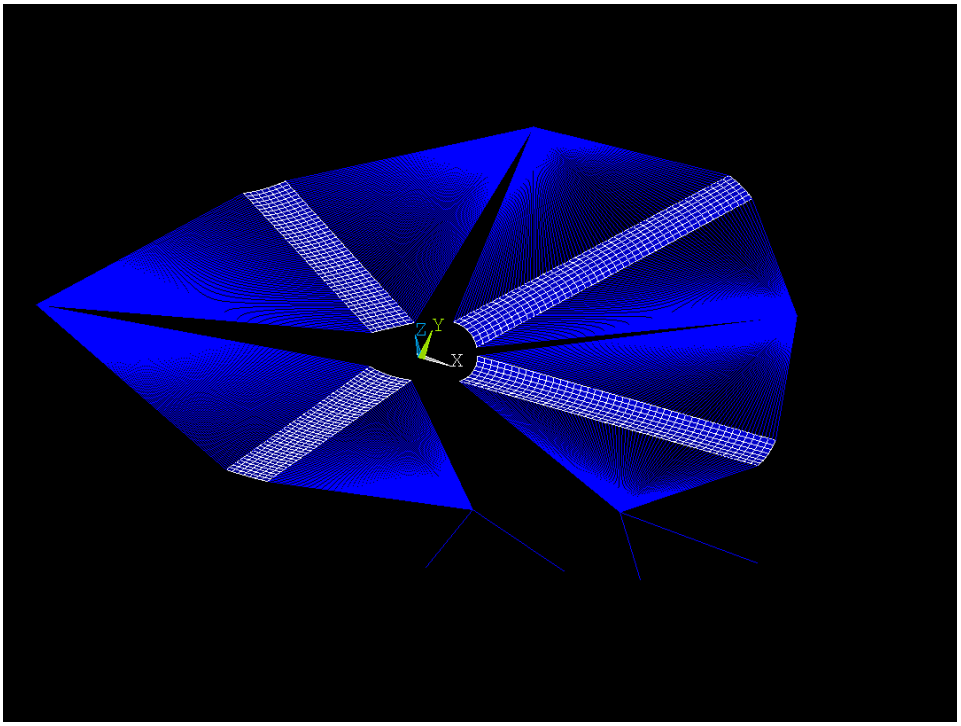

Figure A.2: Screenshot of the model after loadstep 1 of the assembly procedure.

```
309   !!!=== Step 2 of the procedure
310 /SOLU   !Initialize solution processor
311 ANTYPE, 0, restart, 1, last, continue !Continue onwards from the end of the last load step
312 AUTOTS, off !Switch auto timestepping off when applying reaction forces
313 NSUBST,1 !Use 1 substep when applying reaction forces
314
315 !Replace the rotation applied to A1 with its corresponding reaction moment
316 DDELE,4,ALL
317 *GET, Xmoment_4, NODE, 4, RF, MX
318 F,4,MX, Xmoment_4
319
320 !Replace the rotation applied to B1 with its corresponding reaction moment
321 DDELE,7,ALL
322 *GET, Xmoment_7, NODE, 7, RF, MX
323 F,7,MX, Xmoment_7
324
325 !Fix all three translational DOFs of A1 and B1
326 D,4,UX,%_FIX%
327 D,4,UY,%_FIX%
328 D,4,UZ,%_FIX%
329 D,7,UX,%_FIX%
330 D,7,UY,%_FIX%
331 D,7,UZ,%_FIX%
332
333 SOLVE   !Solve current study
334 FINISH !exit current processor
335
336
337   !!!=== Step 3 of the procedure
338 /SOLU   !Initialize solution processor
339 ANTYPE, 0, restart, 2, last, continue !Continue onwards from the end of the last load step
340 AUTOTS, on !Use automatic time stepping
341 NSUBST,50,,30 !Specifies the size of substeps. If automatic time stepping is on, the second
        and third numbers specify the maximum and minimum amount of substeps.
342
343 !Stepwise reduce the applied reaction moments to zero
344 F,4,MX, 0
345 F,7,MX, 0
346
```

```
!Get displacement of A1 relative to its initial position
*GET,UX4,NODE,4,U,X
*GET,UY4,NODE,4,U,Y
*GET,UZ4,NODE,4,U,Z

!Get initial position of A1 and B1
*GET,LOCX4,NODE,4,LOC,X
*GET,LOCX7,NODE,7,LOC,X
*GET,LOCY4,NODE,4,LOC,Y
*GET,LOCY7,NODE,7,LOC,Y
*GET,LOCZ4,NODE,4,LOC,Z
*GET,LOCZ7,NODE,7,LOC,Z

!Get distances between initial positions of A1 and B1
DLOCX7_4 = LOCX7 - LOCX4
DLOCY7_4 = LOCY7 - LOCY4
DLOCZ7_4 = LOCZ7 - LOCZ4

!Displace B1 based on the displacement of A1 relative to its initial position and the
    distances between initial positions of A1 and B1
D,7,UX,UX4 - DLOCX7_4
D,7,UY,UY4 - DLOCY7_4
D,7,UZ,UZ4 - DLOCZ7_4

SOLVE   !Solve current study
FINISH !exit current processor

/POST1 !Initialize post1 processor
PLDISP,0 !Check whether the nodes align properly
```

Listing A.5: Loadsteps 2-3 of assembling the non-Euclidean vertex.

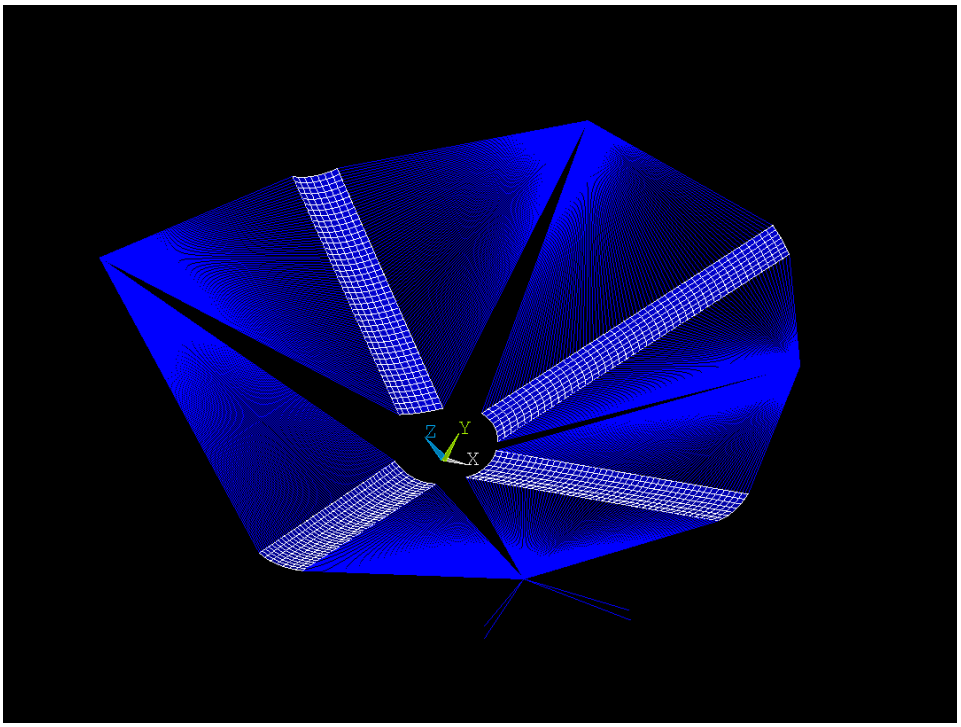

Figure A.3: Screenshot of the model after loadstep 3 of the assembly procedure.

```
  !!!=== Step 4 part 1 of the procedure (step needs to be divided over two loadsteps)
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 3, last, continue !Continue onwards from the end of the last load step
AUTOTS, off !Switch auto timestepping off when coupling nodes
NSUBST,1 !Use 1 substep when coupling nodes
```

```
380
381 FDELE, ALL !Delete all applied forces (as they are zero)
382 DDELE,7,ALL !Delete all displacements applied to B1, as A1 will lead this node from now on
383
384 CP,NEXT,UX,4,7 !Couple nodes in UX, first number is the primary node (A1), second number is
        the node for which the DOF is deleted (B1)
385 CP,NEXT,UY,4,7 !Couple nodes in UY, first number is the primary node (A1), second number is
        the node for which the DOF is deleted (B1)
386 CP,NEXT,UZ,4,7 !Couple nodes in UZ , first number is the primary node (A1), second number is
        the node for which the DOF is deleted (B1)
387
388 SOLVE   !Solve current study
389 FINISH !exit current processor
390
391
392   !!!=== Step 4 part 2 of the procedure (step needs to be divided over two loadsteps)
393 /SOLU   !Initialize solution processor
394 ANTYPE, 0, restart, 4, last, continue !Continue onwards from the end of the last load step
395 AUTOTS, off !Switch auto timestepping off when applying reaction forces
396 NSUBST,1 !Use 1 substep when applying reaction forces
397
398 !Replace all displacements applied to A1 with its corresponding reaction forces
399 DDELE,4,ALL
400 *GET, Xforce_4, NODE, 4, RF, FX
401 *GET, Yforce_4, NODE, 4, RF, FY
402 *GET, Zforce_4, NODE, 4, RF, FZ
403 F,4,FX,Xforce_4
404 F,4,FY,Yforce_4
405 F,4,FZ,Zforce_4
406
407 !Fix two translational DOFs of A2 and B2
408 D,5,UZ,%_FIX%
409 D,8,UZ,%_FIX%
410 D,5,UX,%_FIX%
411 D,8,UX,%_FIX%
412
413 SOLVE   !Solve current study
414 FINISH !exit current processor
415
416
417   !!!=== Loadstep 5 of the procedure
418 /SOLU   !Initialize solution processor
419 ANTYPE, 0, restart, 5, last, continue !Continue onwards from the end of the last load step
420 AUTOTS, on !Use automatic time stepping
421 NSUBST,30,,10 !Specifies the size of substeps. If automatic time stepping is on, the second
        and third numbers specify the maximum and minimum amount of substeps.
422
423 !Get displacement of A2 relative to its initial position
424 *GET,UX5,NODE,5,U,X
425 *GET,UZ5,NODE,5,U,Z
426
427 !Get initial position of A2 and B2
428 *GET,LOCX8,NODE,8,LOC,X
429 *GET,LOCX5,NODE,5,LOC,X
430 *GET,LOCZ8,NODE,8,LOC,Z
431 *GET,LOCZ5,NODE,5,LOC,Z
432
433 !Get distances between initial positions of A2 and B2
434 DLOCX8_5 = LOCX8 - LOCX5
435 DLOCZ8_5 = LOCZ8 - LOCZ5
436
437 !Displace B2 based on the displacement of A2 relative to its initial position and the
        distances between initial positions of A2 and B2
438 D,8,UX,UX5 - DLOCX8_5
439 D,8,UZ,UZ5 - DLOCZ8_5
440
441 SOLVE   !Solve current study
442 FINISH  !Exits normally from a processor
443
444 /POST1 !Initialize post1 processor
445 PLDISP,0 !Check whether the nodes align properly
```

Listing A.6: Loadsteps 4-5 of assembling the non-Euclidean vertex.

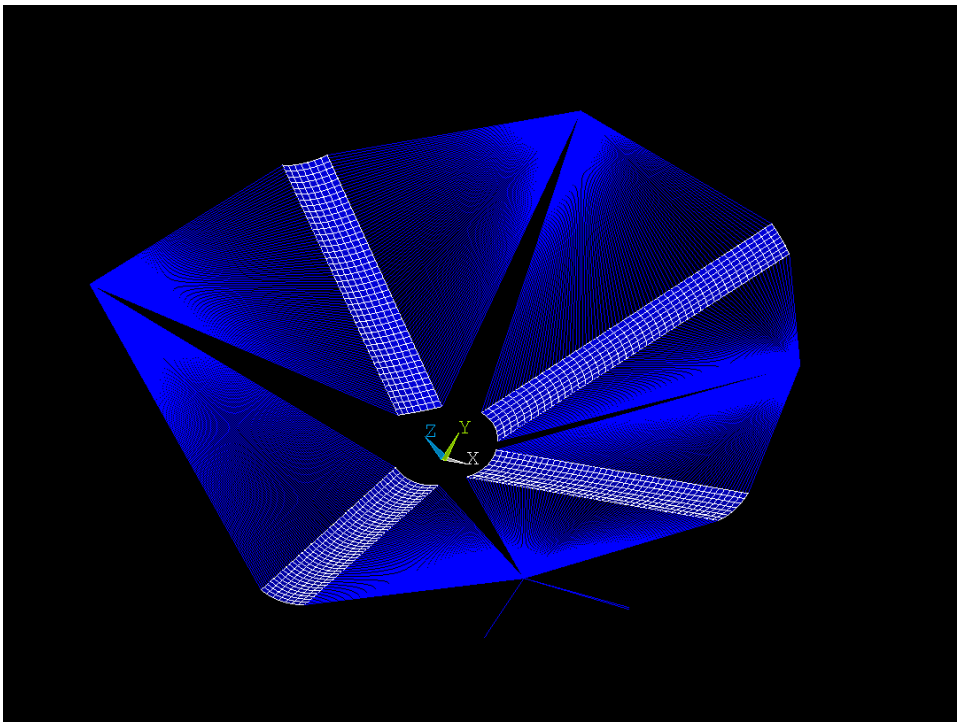

Figure A.4: Screenshot of the model after loadstep 5 of the assembly procedure.

```
446   !!!=== Step 6 part 1 of the procedure (step needs to be divided over two loadsteps)
447 /SOLU   !Initialize solution processor
448 ANTYPE, 0, restart, 6, last, continue !Continue onwards from the end of the last load step
449 AUTOTS, off !Switch auto timestepping off when coupling nodes
450 NSUBST,1 !Use 1 substep when coupling node
451
452 DDELE,8,ALL !Delete all displacements applied to B2, as A2 will lead this node from now on
453
454 CP,NEXT,UX,5,8 !Couple nodes in UX, first number is the primary node (A2), second number is
        the node for which the DOF is deleted (B2)
455 CP,NEXT,UZ,5,8 !Couple nodes in UZ, first number is the primary node (A2), second number is
        the node for which the DOF is deleted (B2)
456
457 SOLVE   !Solve current study
458 FINISH !exit current processor
459
460
461   !!!=== Step 6 part 2 of the procedure (step needs to be divided over two loadsteps)
462 /SOLU   !Initialize solution processor
463 ANTYPE, 0, restart, 7, last, continue !Continue onwards from the end of the last load step
464 AUTOTS, off !Switch auto timestepping off when applying reaction forces
465 NSUBST,1 !Use 1 substep when applying reaction forces
466
467 !Replace all displacements applied to A2 with its corresponding reaction forces
468 DDELE,5,ALL
469 *GET, Xforce_5, NODE, 5, RF, FX
470 *GET, Zforce_5, NODE, 5, RF, FZ
471 F,5,FX,Xforce_5
472 F,5,FZ,Zforce_5
473
474 !Fix one translational DOF of A3 and B3
475 D,6,UZ,%_FIX%
476 D,9,UZ,%_FIX%
```

```
477
478 SOLVE   !Solve current study
479 FINISH !exit current processor
480
481
482   !!!=== Loadstep 7 of the procedure
483 /SOLU   !Initialize solution processor
484 ANTYPE, 0, restart, 8, last, continue !Continue onwards from the end of the last load step
485 AUTOTS, on !Use automatic time stepping
486 NSUBST,30,,10 !Specifies the size of substeps. If automatic time stepping is on, the second
        and third numbers specify the maximum and minimum amount of substeps.
487
488 !Get displacement of A3 relative to its initial position
489 *GET,UZ6,NODE,6,U,Z
490
491 !Get initial position of A3 and B3
492 *GET,LOCZ9,NODE,9,LOC,Z
493 *GET,LOCZ6,NODE,6,LOC,Z
494
495 !Get distances between initial positions of A3 and B3
496 DLOCZ9_6 = LOCZ9 - LOCZ6
497
498 !Displace B3 based on the displacement of A3 relative to its initial position and the
        distances between initial positions of A3 and B3
499 D,9,UZ,UZ6 - DLOCZ9_6
500
501 SOLVE   !Solve current study
502 FINISH !exit current processor
503
504 /POST1 !Initialize post1 processor
505 PLDISP,0 !Check whether the nodes align properly
```

Listing A.7: Loadsteps 6-7 of assembling the non-Euclidean vertex.

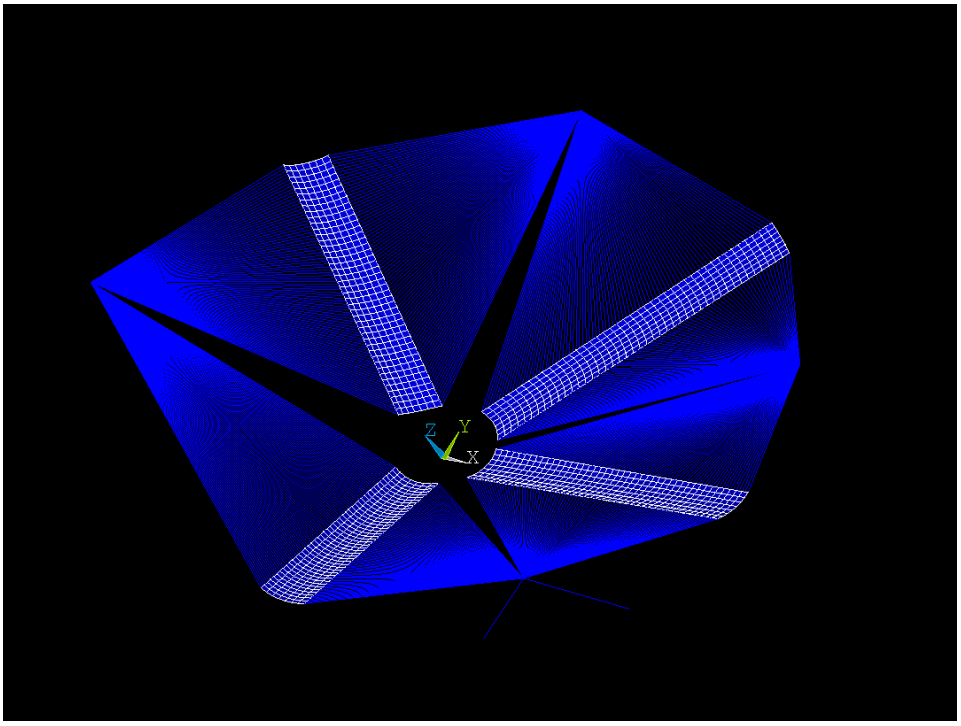

Figure A.5: Screenshot of the model after loadstep 7 of the assembly procedure.

```
506   !!!=== Step 8 part 1 of the procedure (step needs to be divided over two loadsteps)
507 /SOLU   !Initialize solution processor
508 ANTYPE, 0, restart, 9, last, continue !Continue onwards from the end of the last load step
```

```
AUTOTS, off !Switch auto timestepping off when coupling nodes
NSUBST,1 !Use 1 substep when coupling node

DDELE,9,ALL !Delete all displacements applied to B3, as A3 will lead this node from now on

CP,NEXT,UZ,6,9 !Couple nodes in UZ, first number is the primary node (A3), second number is
    the node for which the DOF is deleted (B3)

SOLVE   !Solve current study
FINISH !exit current processor


  !!!=== Step 8 part 2 of the procedure (step needs to be divided over two loadsteps)
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 10, last, continue !Continue onwards from the end of the last load step
AUTOTS, off !Switch auto timestepping off when applying reaction forces
NSUBST,1 !Use 1 substep when applying reaction forces

!Replace the displacement applied to A3 with its corresponding reaction force
DDELE,6,ALL
*GET, Zforce_6, NODE, 6, RF, FZ
F,6,FZ,Zforce_6

SOLVE   !Solve current study
FINISH !exit current processor


  !!!=== Loadstep 9 of the procedure
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 11, last, continue !Continue onwards from the end of the last load step
AUTOTS, on !Use automatic time stepping
NSUBST,100,,10 !Specifies the size of substeps. If automatic time stepping is on, the second
    and third numbers specify the maximum and minimum amount of substeps.

!Stepwise reduce all remaining forces to zero
F,4,FX,0
F,4,FY,0
F,4,FZ,0
F,5,FX,0
F,5,FZ,0
F,6,FZ,0

SOLVE   !Solve current study
FINISH !exit current processor

/POST1 !Initialize post1 processor
PLDISP,0 !Check whether the assembly of the vertex is complete and correct

    !!!=== Assembly of non-Euclidean vertex complete
```

Listing A.8: Loadsteps 8-9 of assembling the non-Euclidean vertex.

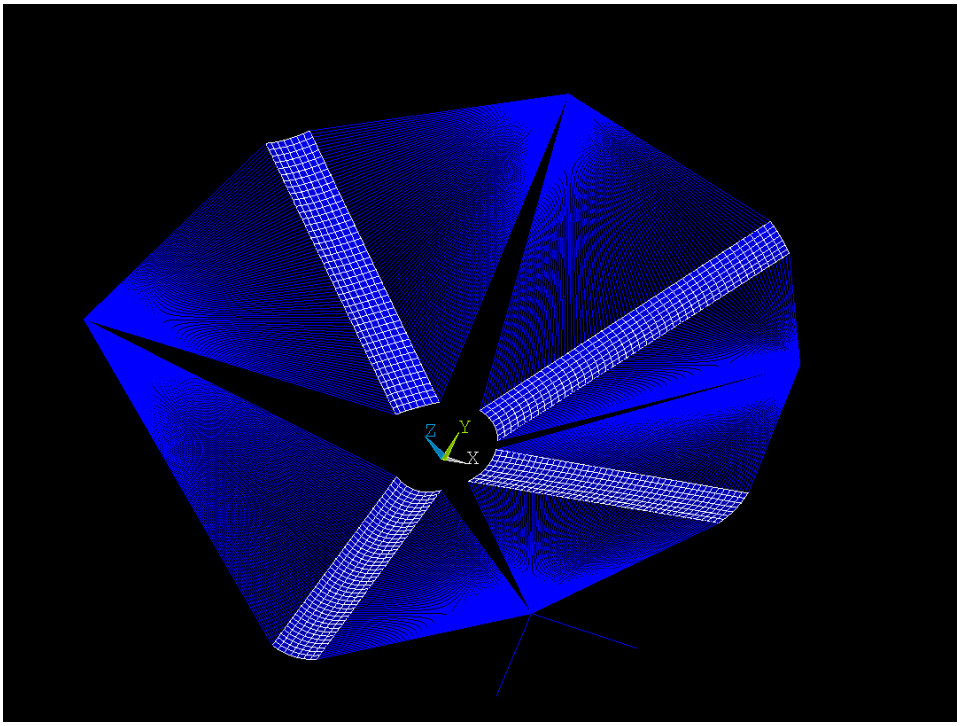

Figure A.6: Screenshot of the model after all loadsteps of the assembly procedure.

B

Detailed explanation of numerical procedure 2

This appendix includes the ANSYS Mechanical APDL code used to model a single rigid facet non-Euclidean vertex. The code is explained in blocks and contains comments in the script to guide the reader.

The first block of code contains the commands to define the element types (including key options), material model, and sections used for the flexible creases and facets.

```
1  FINISH !exit current processor
2  /CLEAR !clear APDL database
3
4  /PREP7 !initialize preprocessor
5
6    !!!=== Define element types and material properties
7
8  ET,1,281 !Element for panels (SHELL281)
9  ET, 2,MPC184 !Element for rigid connections
10 KEYOPT, 2, 1, 1 !Last number defines whether it is a link (0) or beam (1). Should always be
       beam for this application
11 KEYOPT, 2, 2, 1 !Change kinematic constraint method to be able to use coupling (to Lagrange
       multiplier method)
12 MP,ex,1,1.85*10**11 !Define Young's Modulus
13 MP,nuxy,1,0.3 !Define Poisson's ratio
14
15   !!!=== Define sections of creases and flexible facets
16
17 t=(0.02)*10**(-3) !Thickness used for creases
18
19 SECTYPE,1,shell,, !Section of creases
20 SECDATA,t,1,,5 !Defines the thickness, material ID, angle (for anisotropic materials) and
       number of integration points in layer, respectively
21 SECOFFSET,MID !Defines the section offset. Shell node will be offset to midplane of the
       section
22
23 SECTYPE,2,shell,, !Section of flexible facets
24 SECDATA,3*t,1,,3 !Defines the thickness (3 times the crease thickness), material ID, angle (
       for anisotropic materials) and number of integration points in layer, respectively
25 SECOFFSET,MID !Defines the section offset. Shell node will be offset to midplane of the
       section
```

Listing B.1: Definition of element types (including key options), material model, and sections.

The second block of code contains the commands to define the geometry of the unassembled non-Euclidean vertex. The geometry is built up using keypoints, lines, and areas.

```
26   !!!=== Define geometry of the model
```

```
27
28 pi=3.14159265 !Definition of pi
29 Rsmall = 4*10**(-3) !Radius of inner circle of the vertex
30 Rbig = 25*10**(-3) !Radius of outer circle of the vertex
31 Creasewidth = 3*10**(-3) !Width of the creases of the vertex
32
33 alpha1_deg= 60 !Sector angle of the first facet
34 alpha1=(alpha1_deg/360)*2*pi !Conversion of alpha1 to radians
35
36 alpha2_deg= 90 !Sector angle of the second facet
37 alpha2=(alpha2_deg/360)*2*pi !Conversion of alpha2 to radians
38
39 alpha3_deg= 135 !Sector angle of the third facet
40 alpha3=(alpha3_deg/360)*2*pi !Conversion of alpha3 to radians
41
42 alpha4_deg= 75 !Sector angle of the fourth facet (the facet that is split)
43 alpha4=(alpha4_deg/360)*2*pi !Conversion of alpha4 to radians
44
45 alphasurplus_deg= -5 !Angle that is added or subtracted to each sector angle to get the non-
       Euclidean vertex. Negative for deficit, positive for surplus
46 alphasurplus=(alphasurplus_deg/360)*2*pi !Conversion of Alpha to radians
47
48   !Intermediate variables, used to construct the model
49 Sdiagonal = sqrt((Rsmall)**2-(Creasewidth/2)**2)
50 Sdiagonal2 = sqrt((Rbig)**2-(Creasewidth/2)**2)
51
52 Radialline = Rbig - Rsmall - (Rbig-Sdiagonal2) + (Rsmall-Sdiagonal)
53
54 effective_angle3 = alpha1 + alpha2 + alpha3 + 3*alphasurplus - 3*pi/2
55
56
57 X3 = Creasewidth/2*sin(alpha1 + alphasurplus) + Sdiagonal*cos(alpha1 + alphasurplus)
58 Y3 = -Creasewidth/2*cos(alpha1 + alphasurplus) + Sdiagonal*sin(alpha1 + alphasurplus)
59
60 X4 = -Creasewidth/2*sin(alpha1 + alphasurplus) + Sdiagonal*cos(alpha1 + alphasurplus)
61 Y4 = Creasewidth/2*cos(alpha1 + alphasurplus) + Sdiagonal*sin(alpha1 + alphasurplus)
62
63 X5 = Creasewidth/2*cos(alpha1 + alpha2 + 2*alphasurplus - pi/2) - Sdiagonal*sin(alpha1 +
       alpha2 + 2*alphasurplus - pi/2)
64 Y5 = Creasewidth/2*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2) + Sdiagonal*cos(alpha1 +
       alpha2 + 2*alphasurplus - pi/2)
65
66 X6 = -Creasewidth/2*cos(alpha1 + alpha2 + 2*alphasurplus - pi/2) - Sdiagonal*sin(alpha1 +
       alpha2 + 2*alphasurplus - pi/2)
67 Y6 = -Creasewidth/2*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2) + Sdiagonal*cos(alpha1 +
       alpha2 + 2*alphasurplus - pi/2)
68
69 X7 = -Creasewidth/2*cos(-effective_angle3) - Sdiagonal*sin(-effective_angle3)
70 Y7 = Creasewidth/2*sin(-effective_angle3) - Sdiagonal*cos(-effective_angle3)
71
72 X8 = Creasewidth/2*cos(-effective_angle3) - Sdiagonal*sin(-effective_angle3)
73 Y8 = -Creasewidth/2*sin(-effective_angle3) - Sdiagonal*cos(-effective_angle3)
74
75
76   !!!=== Keypoints
77
78 K,1, Sdiagonal, -Creasewidth/2, 0
79 K,2, Sdiagonal, Creasewidth/2, 0
80
81 K,3, X3, Y3, 0
82 K,4, X4, Y4, 0
83
84 K,5, X5, Y5, 0
85 K,6, X6, Y6, 0
86
87 K,7, X7, Y7, 0
88 K,8, X8, Y8, 0
89
90 K,9, Rsmall*sin((alpha4 + alphasurplus)/2 + effective_angle3), -Rsmall*cos((alpha4 +
       alphasurplus)/2 + effective_angle3), 0
91 K,10, Rsmall*cos((alpha4 + alphasurplus)/2), -Rsmall*sin((alpha4 + alphasurplus)/2), 0
```

```
92
93  K,11, Sdiagonal + Radialline, -Creasewidth/2, 0
94  K,12, Sdiagonal + Radialline, Creasewidth/2, 0
95
96  K,13, X3 + Radialline*cos(alpha1 + alphasurplus), Y3 + Radialline*sin(alpha1 + alphasurplus),
        0
97  K,14, X4 + Radialline*cos(alpha1 + alphasurplus), Y4 + Radialline*sin(alpha1 + alphasurplus),
        0
98
99  K,15, X5 - Radialline*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2), Y5 + Radialline*cos(
        alpha1 + alpha2 + 2*alphasurplus - pi/2), 0
100 K,16, X6 - Radialline*sin(alpha1 + alpha2 + 2*alphasurplus - pi/2), Y6 + Radialline*cos(
        alpha1 + alpha2 + 2*alphasurplus - pi/2), 0
101
102 K,17, X7 - Radialline*sin(-effective_angle3), Y7 - Radialline*cos(-effective_angle3), 0
103 K,18, X8 - Radialline*sin(-effective_angle3), Y8 - Radialline*cos(-effective_angle3), 0
104
105 K,19, (Rbig)*sin((alpha4 + alphasurplus)/2 + effective_angle3), -(Rbig)*cos((alpha4 +
        alphasurplus)/2 + effective_angle3), 0
106 K,20, (Rbig)*cos((alpha4 + alphasurplus)/2), -(Rbig)*sin((alpha4 + alphasurplus)/2), 0
107
108 K,21, 0,0,0 !Origin, used for center of curvature
109
110
111   !!!=== Lines
112
113 !First ring arcs
114 LARC, 1, 2, 21, Rsmall
115 LARC, 2, 3, 21, Rsmall
116 LARC, 3, 4, 21, Rsmall
117 LARC, 4, 5, 21, Rsmall
118 LARC, 5, 6, 21, Rsmall
119 LARC, 6, 7, 21, Rsmall
120 LARC, 7, 8, 21, Rsmall
121 LARC, 8, 9, 21, Rsmall
122 LARC, 1, 10, 21, Rsmall
123
124 !Second ring arcs
125 LARC, 11, 12, 21, Rbig
126 LARC, 12, 13, 21, Rbig
127 LARC, 13, 14, 21, Rbig
128 LARC, 14, 15, 21, Rbig
129 LARC, 15, 16, 21, Rbig
130 LARC, 16, 17, 21, Rbig
131 LARC, 17, 18, 21, Rbig
132 LARC, 18, 19, 21, Rbig
133 LARC, 11, 20, 21, Rbig
134
135 !Radial lines
136 L,1,11
137 L,2,12
138 L,3,13
139 L,4,14
140 L,5,15
141 L,6,16
142 L,7,17
143 L,8,18
144 L,9,19
145 L,10,20
146
147   !!!=== Creating areas between lines
148
149 AL,1,10,19,20 !Area 1 (crease)
150 AL,2,11,20,21 !Area 2 (flexible facet)
151 AL,3,12,21,22 !Area 3 (crease)
152 AL,4,13,22,23 !Area 4 (flexible facet)
153 AL,5,14,23,24 !Area 5 (crease)
154 AL,6,15,24,25 !Area 6 (flexible facet)
155 AL,7,16,25,26 !Area 7 (crease)
156 AL,8,17,26,27 !Area 8 (flexible facet)
157 AL,9,18,19,28 !Area 9 (flexible facet)
```

Listing B.2: Building the geometry of the unassembled non-Euclidean vertex.

The third block of code contains the commands to create the necessary hard points (which will turn into the first alignment nodes) and the second and third alignment nodes. After that, the crease and facet areas are meshed and rigid connections are created from the first alignment nodes to the second and third alignment nodes. There is also a piece of code that creates the components consisting of the first alignment nodes. After this block of code, a screenshot shows how the model looks after the preparation steps.

```
158   !!!=== Create component containing the creases
159
160 ASEL,S,AREA,,1,7,2
161 CM,crease_areas,AREA !Create a component of the selected areas
162 ALLSEL !select everything (so reset selection done before)
163
164   !!!=== Create hard points and nodes used for aligning the facets
165
166   !Alignment hard points and nodes for panel A. A1_1 and A1_5 are made automatically as they
          are at a corner of a meshed area.
167
168 HPTCREATE,LINE,28,,COORD, ((1 - 0.25)*Rsmall + 0.25*Rbig)*cos((alpha4 + alphasurplus)/2),
      -((1 - 0.25)*Rsmall + 0.25*Rbig)*sin((alpha4 + alphasurplus)/2), 0 !A1_2
169 HPTCREATE,LINE,28,,COORD, ((1 - 0.5)*Rsmall + 0.5*Rbig)*cos((alpha4 + alphasurplus)/2), -((1
      - 0.5)*Rsmall + 0.5*Rbig)*sin((alpha4 + alphasurplus)/2), 0 !A1_3
170 HPTCREATE,LINE,28,,COORD, ((1 - 0.75)*Rsmall + 0.75*Rbig)*cos((alpha4 + alphasurplus)/2),
      -((1 - 0.75)*Rsmall + 0.75*Rbig)*sin((alpha4 + alphasurplus)/2), 0 !A1_4
171
172 N, 1, (Rsmall + 10e-3)*cos((alpha4 + alphasurplus)/2 + pi/24), -(Rsmall + 10e-3)*sin((alpha4
      + alphasurplus)/2 + pi/24), 0 !A2_1
173 N, 2, ((1 - 0.25)*Rsmall + 0.25*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + pi/24), -((1 -
      0.25)*Rsmall + 0.25*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + pi/24), 0 !A2_2
174 N, 3, ((1 - 0.5)*Rsmall + 0.5*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + pi/24), -((1 -
      0.5)*Rsmall + 0.5*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + pi/24), 0 !A2_3
175 N, 4, ((1 - 0.75)*Rsmall + 0.75*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + pi/24), -((1 -
      0.75)*Rsmall + 0.75*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + pi/24), 0 !A2_4
176 N, 5, (Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 + pi/24), -(Rbig + 10e-3)*sin((alpha4 +
      alphasurplus)/2 + pi/24), 0 !A2_5
177
178 N, 6, (Rsmall + 10e-3)*cos((alpha4 + alphasurplus)/2 - pi/24), -(Rsmall + 10e-3)*sin((alpha4
      + alphasurplus)/2 - pi/24), 0 !A3_1
179 N, 7, ((1 - 0.25)*Rsmall + 0.25*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 - pi/24), -((1 -
      0.25)*Rsmall + 0.25*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 - pi/24), 0 !A3_2
180 N, 8, ((1 - 0.5)*Rsmall + 0.5*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 - pi/24), -((1 -
      0.5)*Rsmall + 0.5*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 - pi/24), 0 !A3_3
181 N, 9, ((1 - 0.75)*Rsmall + 0.75*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 - pi/24), -((1 -
      0.75)*Rsmall + 0.75*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 - pi/24), 0 !A3_4
182 N, 10, (Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 - pi/24), -(Rbig + 10e-3)*sin((alpha4 +
      alphasurplus)/2 - pi/24), 0 !A3_5
183
184   !Alignment hard points and nodes for panel B. B1_1 and B1_5 are made automatically as they
          are at a corner of a meshed area.
185
186 HPTCREATE,LINE,27,,COORD, ((1 - 0.25)*Rsmall + 0.25*Rbig)*sin((alpha4 + alphasurplus)/2 +
      effective_angle3), -((1 - 0.25)*Rsmall + 0.25*Rbig)*cos((alpha4 + alphasurplus)/2 +
      effective_angle3), 0 !B1_2
187 HPTCREATE,LINE,27,,COORD, ((1 - 0.5)*Rsmall + 0.5*Rbig)*sin((alpha4 + alphasurplus)/2 +
      effective_angle3), -((1 - 0.5)*Rsmall + 0.5*Rbig)*cos((alpha4 + alphasurplus)/2 +
      effective_angle3), 0 !B1_3
188 HPTCREATE,LINE,27,,COORD, ((1 - 0.75)*Rsmall + 0.75*Rbig)*sin((alpha4 + alphasurplus)/2 +
      effective_angle3), -((1 - 0.75)*Rsmall + 0.75*Rbig)*cos((alpha4 + alphasurplus)/2 +
      effective_angle3), 0 !B1_4
189
190 N, 11, (Rsmall + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3 - pi/24), -(Rsmall +
       10e-3)*cos((alpha4 + alphasurplus)/2 + effective_angle3 - pi/24), 0 !B2_1
191 N, 12, ((1 - 0.25)*Rsmall + 0.25*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 +
      effective_angle3 - pi/24), -((1 - 0.25)*Rsmall + 0.25*Rbig  + 10e-3)*cos((alpha4 +
      alphasurplus)/2 + effective_angle3 - pi/24), 0 !B2_2
```

```
192 N, 13, ((1 - 0.5)*Rsmall + 0.5*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3
      - pi/24), -((1 - 0.5)*Rsmall + 0.5*Rbig  + 10e-3)*cos((alpha4 + alphasurplus)/2 +
    effective_angle3 - pi/24), 0 !B2_3
193 N, 14, ((1 - 0.75)*Rsmall + 0.75*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 +
    effective_angle3 - pi/24), -((1 - 0.75)*Rsmall + 0.75*Rbig  + 10e-3)*cos((alpha4 +
    alphasurplus)/2 + effective_angle3 - pi/24), 0 !B2_4
194 N, 15, (Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3 - pi/24), -(Rbig + 10e
    -3)*cos((alpha4 + alphasurplus)/2 + effective_angle3 - pi/24), 0 !B2_5
195
196 N, 16, (Rsmall + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3 + pi/24), -(Rsmall +
     10e-3)*cos((alpha4 + alphasurplus)/2 + effective_angle3 + pi/24), 0 !B3_1
197 N, 17, ((1 - 0.25)*Rsmall + 0.25*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 +
    effective_angle3 + pi/24), -((1 - 0.25)*Rsmall + 0.25*Rbig + 10e-3)*cos((alpha4 +
    alphasurplus)/2 + effective_angle3 + pi/24), 0 !B3_2
198 N, 18, ((1 - 0.5)*Rsmall + 0.5*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3
      + pi/24), -((1 - 0.5)*Rsmall + 0.5*Rbig + 10e-3)*cos((alpha4 + alphasurplus)/2 +
    effective_angle3 + pi/24), 0 !B3_3
199 N, 19, ((1 - 0.75)*Rsmall + 0.75*Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 +
    effective_angle3 + pi/24), -((1 - 0.75)*Rsmall + 0.75*Rbig + 10e-3)*cos((alpha4 +
    alphasurplus)/2 + effective_angle3 + pi/24), 0 !B3_4
200 N, 20, (Rbig + 10e-3)*sin((alpha4 + alphasurplus)/2 + effective_angle3 + pi/24), -(Rbig + 10e
    -3)*cos((alpha4 + alphasurplus)/2 + effective_angle3 + pi/24), 0 !B3_5
201
202
203   !!!=== Mesh crease and facet areas
204
205 CMSEL, S, crease_areas !Select crease areas
206 AATT, 1, , 1, , 1 !Define associates element attributes with the selected, unmeshed areas.
    Entries are material ID, real constants, element type, coordinate system and section
    number respectively
207 ESIZE, Creasewidth/6 !Define element size as a function of crease width
208 AMESH, ALL !Mesh all selected areas
209 ALLSEL !Select everything (so reset selection done before)
210
211 CMSEL, S , crease_areas !Select crease areas
212 ASEL, INVE !Inverse current selection, so select all areas instead of the crease areas
213 AATT, 1, , 1, , 2 !Define associates element attributes with the selected, unmeshed areas.
    Entries are material ID, real constants, element type, coordinate system and section
    number respectively
214 ESIZE, Creasewidth/2 !Define element size as a function of crease width
215 AMESH, ALL !Mesh all selected areas
216 ALLSEL !Select everything (so reset selection done before)
217
218
219   !!!=== Find the node numbers from nodes created using hard points
220
221 TOL = 1e-6   !Tolerance of the area we look for the nodes at. Could be decreased even further
     if more than 1 node is selected.
222
223 XLOC_A1_1 = Rsmall*cos((alpha4 + alphasurplus)/2) !The X location at which the node should be
     located
224 YLOC_A1_1 = -Rsmall*sin((alpha4 + alphasurplus)/2) !The Y location at which the node should
    be located
225 NSEL,S,LOC,X,XLOC_A1_1-TOL,XLOC_A1_1+TOL !Select the nodes around the correct X location
226 NSEL,R,LOC,Y,YLOC_A1_1-TOL,YLOC_A1_1+TOL !Reselect the node that is also at the correct Y
    location. This should be the correct node, reduce TOL if multiple nodes are selected.
227 CM,NODE_A1_1,NODE !Create a component the found node
228 *GET, NODE_A1_1_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node
    number
229 ALLSEL !Select everything (so reset selection done before)
230
231 XLOC_A1_2 = ((1 - 0.25)*Rsmall + 0.25*Rbig)*cos((alpha4 + alphasurplus)/2) !The X location at
     which the node should be located
232 YLOC_A1_2 = -((1 - 0.25)*Rsmall + 0.25*Rbig)*sin((alpha4 + alphasurplus)/2) !The Y location
    at which the node should be located
233 NSEL,S,LOC,X,XLOC_A1_2-TOL,XLOC_A1_2+TOL !Select the nodes around the correct X location
234 NSEL,R,LOC,Y,YLOC_A1_2-TOL,YLOC_A1_2+TOL !Reselect the node that is also at the correct Y
    location. This should be the correct node, reduce TOL if multiple nodes are selected.
235 CM,NODE_A1_2,NODE !Create a component the found node
236 *GET, NODE_A1_2_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node
    number
```

```
ALLSEL !Select everything (so reset selection done before)

XLOC_A1_3 = ((1 - 0.5)*Rsmall + 0.5*Rbig)*cos((alpha4 + alphasurplus)/2) !The X location at which the node should be located
YLOC_A1_3 = -((1 - 0.5)*Rsmall + 0.5*Rbig)*sin((alpha4 + alphasurplus)/2) !The Y location at which the node should be located
NSEL,S,LOC,X,XLOC_A1_3-TOL,XLOC_A1_3+TOL !Select the nodes around the correct X location
NSEL,R,LOC,Y,YLOC_A1_3-TOL,YLOC_A1_3+TOL !Reselect the node that is also at the correct Y location. This should be the correct node, reduce TOL if multiple nodes are selected.
CM,NODE_A1_3,NODE !Create a component the found node
*GET, NODE_A1_3_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node number
ALLSEL !Select everything (so reset selection done before)

XLOC_A1_4 = ((1 - 0.75)*Rsmall + 0.75*Rbig)*cos((alpha4 + alphasurplus)/2) !The X location at which the node should be located
YLOC_A1_4 = -((1 - 0.75)*Rsmall + 0.75*Rbig)*sin((alpha4 + alphasurplus)/2) !The Y location at which the node should be located
NSEL,S,LOC,X,XLOC_A1_4-TOL,XLOC_A1_4+TOL !Select the nodes around the correct X location
NSEL,R,LOC,Y,YLOC_A1_4-TOL,YLOC_A1_4+TOL !Reselect the node that is also at the correct Y location. This should be the correct node, reduce TOL if multiple nodes are selected.
CM,NODE_A1_4,NODE !Create a component the found node
*GET, NODE_A1_4_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node number
ALLSEL !Select everything (so reset selection done before)

XLOC_A1_5 = Rbig*cos((alpha4 + alphasurplus)/2) !The X location at which the node should be located
YLOC_A1_5 = -Rbig*sin((alpha4 + alphasurplus)/2) !The Y location at which the node should be located
NSEL,S,LOC,X,XLOC_A1_5-TOL,XLOC_A1_5+TOL !Select the nodes around the correct X location
NSEL,R,LOC,Y,YLOC_A1_5-TOL,YLOC_A1_5+TOL !Reselect the node that is also at the correct Y location. This should be the correct node, reduce TOL if multiple nodes are selected.
CM,NODE_A1_5,NODE !Create a component the found node
*GET, NODE_A1_5_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node number
ALLSEL !Select everything (so reset selection done before)


XLOC_B1_1 = Rsmall*sin((alpha4 + alphasurplus)/2 + effective_angle3) !The X location at which the node should be located
YLOC_B1_1 = -Rsmall*cos((alpha4 + alphasurplus)/2 + effective_angle3) !The Y location at which the node should be located
NSEL,S,LOC,X,XLOC_B1_1-TOL,XLOC_B1_1+TOL !Select the nodes around the correct X location
NSEL,R,LOC,Y,YLOC_B1_1-TOL,YLOC_B1_1+TOL !Reselect the node that is also at the correct Y location. This should be the correct node, reduce TOL if multiple nodes are selected.
CM,NODE_B1_1,NODE !Create a component the found node
*GET, NODE_B1_1_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node number
ALLSEL !Select everything (so reset selection done before)

XLOC_B1_2 = ((1 - 0.25)*Rsmall + 0.25*Rbig)*sin((alpha4 + alphasurplus)/2 + effective_angle3) !The X location at which the node should be located
YLOC_B1_2 = -((1 - 0.25)*Rsmall + 0.25*Rbig)*cos((alpha4 + alphasurplus)/2 + effective_angle3) !The Y location at which the node should be located
NSEL,S,LOC,X,XLOC_B1_2-TOL,XLOC_B1_2+TOL !Select the nodes around the correct X location
NSEL,R,LOC,Y,YLOC_B1_2-TOL,YLOC_B1_2+TOL !Reselect the node that is also at the correct Y location. This should be the correct node, reduce TOL if multiple nodes are selected.
CM,NODE_B1_2,NODE !Create a component the found node
*GET, NODE_B1_2_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node number
ALLSEL !Select everything (so reset selection done before)

XLOC_B1_3 = ((1 - 0.5)*Rsmall + 0.5*Rbig)*sin((alpha4 + alphasurplus)/2 + effective_angle3) !The X location at which the node should be located
YLOC_B1_3 = -((1 - 0.5)*Rsmall + 0.5*Rbig)*cos((alpha4 + alphasurplus)/2 + effective_angle3) !The Y location at which the node should be located
NSEL,S,LOC,X,XLOC_B1_3-TOL,XLOC_B1_3+TOL !Select the nodes around the correct X location
NSEL,R,LOC,Y,YLOC_B1_3-TOL,YLOC_B1_3+TOL !Reselect the node that is also at the correct Y location. This should be the correct node, reduce TOL if multiple nodes are selected.
CM,NODE_B1_3,NODE !Create a component the found node
```

```
285 *GET, NODE_B1_3_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node
        number
286 ALLSEL !Select everything (so reset selection done before)
287
288 XLOC_B1_4 = ((1 - 0.75)*Rsmall + 0.75*Rbig)*sin((alpha4 + alphasurplus)/2 + effective_angle3)
         !The X location at which the node should be located
289 YLOC_B1_4 = -((1 - 0.75)*Rsmall + 0.75*Rbig)*cos((alpha4 + alphasurplus)/2 + effective_angle3
        ) !The Y location at which the node should be located
290 NSEL,S,LOC,X,XLOC_B1_4-TOL,XLOC_B1_4+TOL !Select the nodes around the correct X location
291 NSEL,R,LOC,Y,YLOC_B1_4-TOL,YLOC_B1_4+TOL !Reselect the node that is also at the correct Y
        location. This should be the correct node, reduce TOL if multiple nodes are selected.
292 CM,NODE_B1_4,NODE !Create a component the found node
293 *GET, NODE_B1_4_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node
        number
294 ALLSEL !Select everything (so reset selection done before)
295
296 XLOC_B1_5 = Rbig*sin((alpha4 + alphasurplus)/2 + effective_angle3) !The X location at which
        the node should be located
297 YLOC_B1_5 = -Rbig*cos((alpha4 + alphasurplus)/2 + effective_angle3) !The Y location at which
        the node should be located
298 NSEL,S,LOC,X,XLOC_B1_5-TOL,XLOC_B1_5+TOL !Select the nodes around the correct X location
299 NSEL,R,LOC,Y,YLOC_B1_5-TOL,YLOC_B1_5+TOL !Reselect the node that is also at the correct Y
        location. This should be the correct node, reduce TOL if multiple nodes are selected.
300 CM,NODE_B1_5,NODE !Create a component the found node
301 *GET, NODE_B1_5_NUM, NODE, 0, NUM, MIN !Create a parameter that contains the found node
        number
302 ALLSEL !Select everything (so reset selection done before)
303
304
305   ===!!! Create extra connections to second and third alignment nodes to align and couple
          panels A and B
306
307 TYPE,2 !Select rigid beam element for creating new elements
308
309 E, NODE_A1_1_NUM, 1
310 E, NODE_A1_1_NUM, 6
311
312 E, NODE_A1_2_NUM, 2
313 E, NODE_A1_2_NUM, 7
314
315 E, NODE_A1_3_NUM, 3
316 E, NODE_A1_3_NUM, 8
317
318 E, NODE_A1_4_NUM, 4
319 E, NODE_A1_4_NUM, 9
320
321 E, NODE_A1_5_NUM, 5
322 E, NODE_A1_5_NUM, 10
323
324
325 E, NODE_B1_1_NUM, 11
326 E, NODE_B1_1_NUM, 16
327
328 E, NODE_B1_2_NUM, 12
329 E, NODE_B1_2_NUM, 17
330
331 E, NODE_B1_3_NUM, 13
332 E, NODE_B1_3_NUM, 18
333
334 E, NODE_B1_4_NUM, 14
335 E, NODE_B1_4_NUM, 19
336
337 E, NODE_B1_5_NUM, 15
338 E, NODE_B1_5_NUM, 20
339
340 FINISH
341
342       !!!=== Preparation complete
```

Listing B.3: Create alignment nodes, mesh the crease and facet areas, and create rigid connections between alignment nodes.

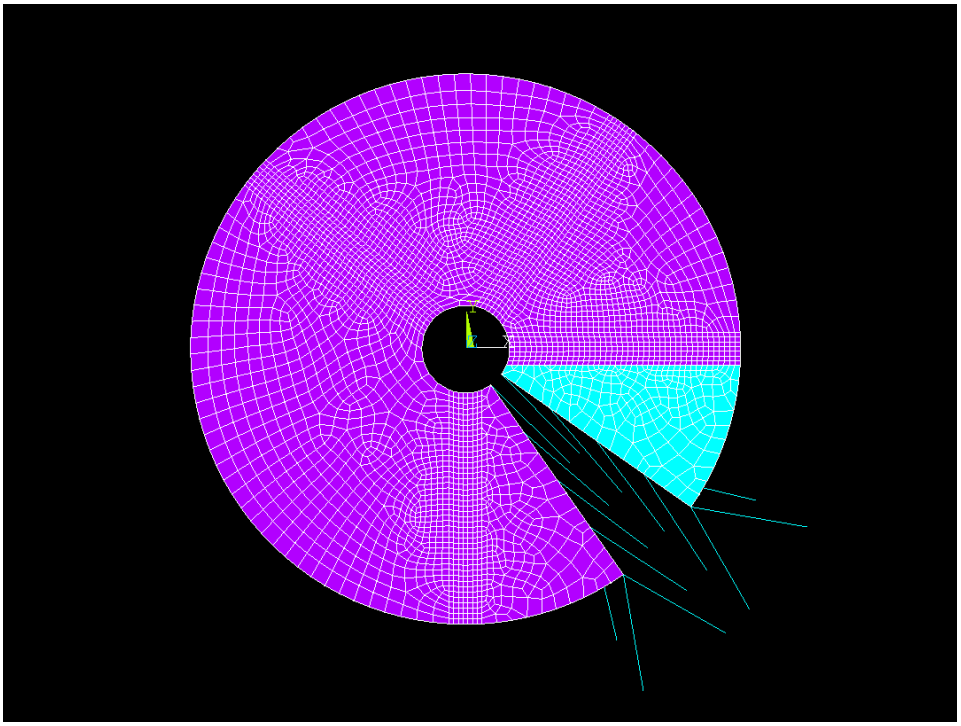

Figure B.1: Screenshot of the model after the preparation steps of procedure 2.

The last blocks of code contain the commands to stepwise assemble the vertex. The step numbers before each loadstep correspond to the step numbers in the main paper. In between the code, there are screenshots of how the model looks after every main displacement step.

```
343       !!!=== Non-Euclidean assembly starting here
344
345   !!!=== Step 1 of the procedure
346 /SOLU   !Initialize solution processor
347 ANTYPE, 0, new !Specifies a new static analysis
348 NLGEOM, on !Non-linear large deflection behavior on or off, including stress stiffening
349 OUTRES,all,all !Controls the solution data written to the database. All solution items at all
      substeps are written to the database.
350 AUTOTS, on !Use automatic time stepping
351 NEQIT, 1000 !Specifies the maximum number of equilibrium iterations for nonlinear analyses.
352 NSUBST,30,,10 !Specifies the size of substeps. If automatic time stepping is on, the second
     and third numbers specify the maximum and minimum amount of substeps.
353
354   !!!=== Fix the base facet in 6 DOFs
355
356 LSEL,S,LINE,,20,21 !Select the outer lines of the base facet that also border a crease
357 NSLL,S,1 !Selects the nodes associated with the selected lines. Choose 1 to include nodes at
     the end of the lines.
358 D,ALL,ALL,0 !Fix all the selected nodes in 6 DOFs
359 ALLSEL !Select everything (so reset selection done before)
360
361   !!!=== Rotate the central nodes A1_n and B1_n out of their original plane and toward each
       other. The closer they get, the better it is.
362
363 D,NODE_A1_1,ROTX, -pi/4
364 D,NODE_A1_2,ROTX, -pi/4
365 D,NODE_A1_3,ROTX, -pi/4
366 D,NODE_A1_4,ROTX, -pi/4
367 D,NODE_A1_5,ROTX, -pi/4
368
369 D,NODE_B1_1,ROTX, -pi/4
370 D,NODE_B1_2,ROTX, -pi/4
371 D,NODE_B1_3,ROTX, -pi/4
372 D,NODE_B1_4,ROTX, -pi/4
```

```
D,NODE_B1_5,ROTX, -pi/4

SOLVE   !Solve current study
FINISH  !Exits normally from a processor

/POST1    !Initialize post1 processor
/DSCALE,ALL,1   !Scaling of displacement displays, set to 1 for true scale and 0 for auto
    scale
PLDISP,0  !Displays the displaced structure, key is used to show or not show the undisplaced
    structure
```

Listing B.4: Loadstep 1 of assembling the non-Euclidean vertex.

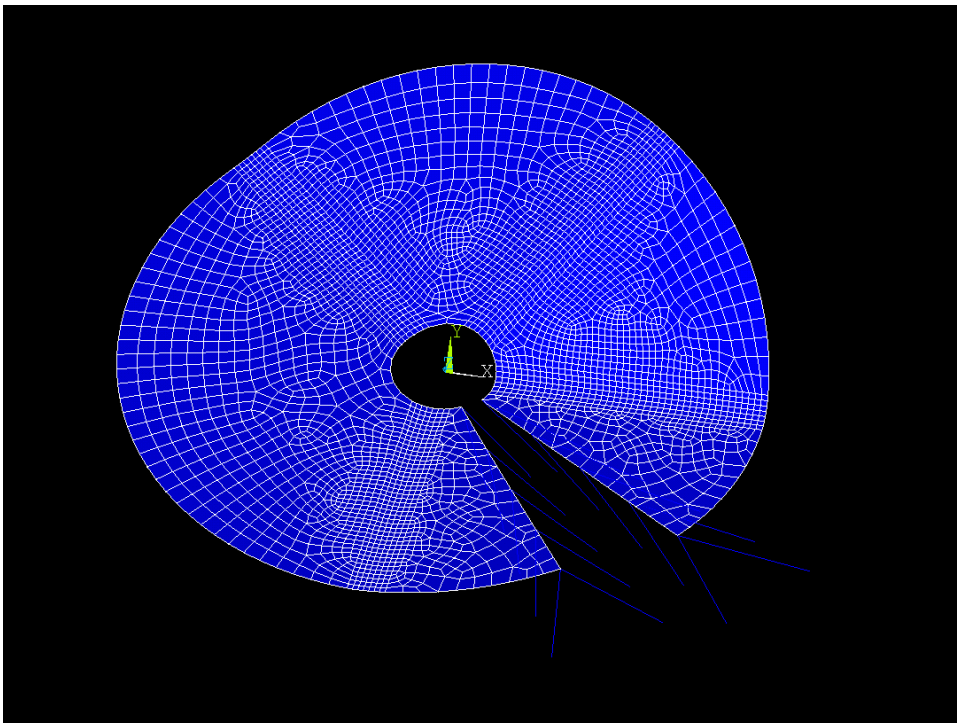

Figure B.2: Screenshot of the model after loadstep 1 of the assembly procedure.

```
  !!!=== Step 2 of the procedure
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 1, last, continue !Continue onwards from the end of the last load step
AUTOTS, off !Switch auto timestepping off when applying reaction forces
NSUBST,1 !Use 1 substep when applying reaction forces

  !!!=== Replace the rotations applied to A1_n with their corresponding reaction moments

DDELE,NODE_A1_1,ALL
DDELE,NODE_A1_2,ALL
DDELE,NODE_A1_3,ALL
DDELE,NODE_A1_4,ALL
DDELE,NODE_A1_5,ALL

*GET, Xmoment_NODE_A1_1, NODE, NODE_A1_1_NUM, RF, MX
F,NODE_A1_1,MX, Xmoment_NODE_A1_1

*GET, Xmoment_NODE_A1_2, NODE, NODE_A1_2_NUM, RF, MX
F,NODE_A1_2,MX, Xmoment_NODE_A1_2

*GET, Xmoment_NODE_A1_3, NODE, NODE_A1_3_NUM, RF, MX
F,NODE_A1_3,MX, Xmoment_NODE_A1_3

*GET, Xmoment_NODE_A1_4, NODE, NODE_A1_4_NUM, RF, MX
```

```
405 F,NODE_A1_4,MX, Xmoment_NODE_A1_4
406
407 *GET, Xmoment_NODE_A1_5, NODE, NODE_A1_5_NUM, RF, MX
408 F,NODE_A1_5,MX, Xmoment_NODE_A1_5
409
410   !!!=== Replace the rotations applied to B1_n with their corresponding reaction moments
411
412 DDELE,NODE_B1_1,ALL
413 DDELE,NODE_B1_2,ALL
414 DDELE,NODE_B1_3,ALL
415 DDELE,NODE_B1_4,ALL
416 DDELE,NODE_B1_5,ALL
417
418 *GET, Xmoment_NODE_B1_1, NODE, NODE_B1_1_NUM, RF, MX
419 F,NODE_B1_1,MX, Xmoment_NODE_B1_1
420
421 *GET, Xmoment_NODE_B1_2, NODE, NODE_B1_2_NUM, RF, MX
422 F,NODE_B1_2,MX, Xmoment_NODE_B1_2
423
424 *GET, Xmoment_NODE_B1_3, NODE, NODE_B1_3_NUM, RF, MX
425 F,NODE_B1_3,MX, Xmoment_NODE_B1_3
426
427 *GET, Xmoment_NODE_B1_4, NODE, NODE_B1_4_NUM, RF, MX
428 F,NODE_B1_4,MX, Xmoment_NODE_B1_4
429
430 *GET, Xmoment_NODE_B1_5, NODE, NODE_B1_5_NUM, RF, MX
431 F,NODE_B1_5,MX, Xmoment_NODE_B1_5
432
433   !!!=== Fix all three translational DOFs of A1_n and B1_n
434
435 D,NODE_A1_1,UX,%_FIX%
436 D,NODE_A1_1,UY,%_FIX%
437 D,NODE_A1_1,UZ,%_FIX%
438
439 D,NODE_A1_2,UX,%_FIX%
440 D,NODE_A1_2,UY,%_FIX%
441 D,NODE_A1_2,UZ,%_FIX%
442
443 D,NODE_A1_3,UX,%_FIX%
444 D,NODE_A1_3,UY,%_FIX%
445 D,NODE_A1_3,UZ,%_FIX%
446
447 D,NODE_A1_4,UX,%_FIX%
448 D,NODE_A1_4,UY,%_FIX%
449 D,NODE_A1_4,UZ,%_FIX%
450
451 D,NODE_A1_5,UX,%_FIX%
452 D,NODE_A1_5,UY,%_FIX%
453 D,NODE_A1_5,UZ,%_FIX%
454
455 D,NODE_B1_1,UX,%_FIX%
456 D,NODE_B1_1,UY,%_FIX%
457 D,NODE_B1_1,UZ,%_FIX%
458
459 D,NODE_B1_2,UX,%_FIX%
460 D,NODE_B1_2,UY,%_FIX%
461 D,NODE_B1_2,UZ,%_FIX%
462
463 D,NODE_B1_3,UX,%_FIX%
464 D,NODE_B1_3,UY,%_FIX%
465 D,NODE_B1_3,UZ,%_FIX%
466
467 D,NODE_B1_4,UX,%_FIX%
468 D,NODE_B1_4,UY,%_FIX%
469 D,NODE_B1_4,UZ,%_FIX%
470
471 D,NODE_B1_5,UX,%_FIX%
472 D,NODE_B1_5,UY,%_FIX%
473 D,NODE_B1_5,UZ,%_FIX%
474
475 SOLVE   !Solve current study
```

```
476 FINISH  !Exits normally from a processor
477
478
479   !!!=== Step 3 of the procedure
480 /SOLU   !Initialize solution processor
481 ANTYPE, 0, restart, 2, last, continue !Continue onwards from the end of the last load step
482 AUTOTS, on !Use automatic time stepping
483 NSUBST,200,,100 !Specifies the size of substeps. If automatic time stepping is on, the second
        and third numbers specify the maximum and minimum amount of substeps.
484
485   !!!=== Stepwise reduce the applied reaction moments to zero
486 F,NODE_A1_1,MX,0
487 F,NODE_B1_1,MX,0
488
489 F,NODE_A1_2,MX,0
490 F,NODE_B1_2,MX,0
491
492 F,NODE_A1_3,MX,0
493 F,NODE_B1_3,MX,0
494
495 F,NODE_A1_4,MX,0
496 F,NODE_B1_4,MX,0
497
498 F,NODE_A1_5,MX,0
499 F,NODE_B1_5,MX,0
500
501   !!!=== Displace B1_n to A1_n. Process explained for B1_1 and A1_1, and repeated for the
        rest.
502
503 !Get displacement of A1_1 relative to its initial position
504 *GET,UXNODE_A1_1,NODE,NODE_A1_1_NUM,U,X
505 *GET,UYNODE_A1_1,NODE,NODE_A1_1_NUM,U,Y
506 *GET,UZNODE_A1_1,NODE,NODE_A1_1_NUM,U,Z
507
508 !Get initial position of A1_1 and B1_1
509 *GET,LOCXNODE_A1_1,NODE,NODE_A1_1_NUM,LOC,X
510 *GET,LOCXNODE_B1_1,NODE,NODE_B1_1_NUM,LOC,X
511 *GET,LOCYNODE_A1_1,NODE,NODE_A1_1_NUM,LOC,Y
512 *GET,LOCYNODE_B1_1,NODE,NODE_B1_1_NUM,LOC,Y
513 *GET,LOCZNODE_A1_1,NODE,NODE_A1_1_NUM,LOC,Z
514 *GET,LOCZNODE_B1_1,NODE,NODE_B1_1_NUM,LOC,Z
515
516 !Get distances between initial positions of A1_1 and B1_1
517 DLOCXNODE_B1_1_NODE_A1_1 = LOCXNODE_B1_1 - LOCXNODE_A1_1
518 DLOCYNODE_B1_1_NODE_A1_1 = LOCYNODE_B1_1 - LOCYNODE_A1_1
519 DLOCZNODE_B1_1_NODE_A1_1 = LOCZNODE_B1_1 - LOCZNODE_A1_1
520
521 !Displace B1_1 based on the displacement of A1_1 relative to its initial position and the
      distances between initial positions of A1_1 and B1_1
522 D,NODE_B1_1,UX,UXNODE_A1_1 - DLOCXNODE_B1_1_NODE_A1_1
523 D,NODE_B1_1,UY,UYNODE_A1_1 - DLOCYNODE_B1_1_NODE_A1_1
524 D,NODE_B1_1,UZ,UZNODE_A1_1 - DLOCZNODE_B1_1_NODE_A1_1
525
526
527 *GET,UXNODE_A1_2,NODE,NODE_A1_2_NUM,U,X
528 *GET,UYNODE_A1_2,NODE,NODE_A1_2_NUM,U,Y
529 *GET,UZNODE_A1_2,NODE,NODE_A1_2_NUM,U,Z
530
531 *GET,LOCXNODE_A1_2,NODE,NODE_A1_2_NUM,LOC,X
532 *GET,LOCXNODE_B1_2,NODE,NODE_B1_2_NUM,LOC,X
533 *GET,LOCYNODE_A1_2,NODE,NODE_A1_2_NUM,LOC,Y
534 *GET,LOCYNODE_B1_2,NODE,NODE_B1_2_NUM,LOC,Y
535 *GET,LOCZNODE_A1_2,NODE,NODE_A1_2_NUM,LOC,Z
536 *GET,LOCZNODE_B1_2,NODE,NODE_B1_2_NUM,LOC,Z
537
538 DLOCXNODE_B1_2_NODE_A1_2 = LOCXNODE_B1_2 - LOCXNODE_A1_2
539 DLOCYNODE_B1_2_NODE_A1_2 = LOCYNODE_B1_2 - LOCYNODE_A1_2
540 DLOCZNODE_B1_2_NODE_A1_2 = LOCZNODE_B1_2 - LOCZNODE_A1_2
541
542 D,NODE_B1_2,UX,UXNODE_A1_2 - DLOCXNODE_B1_2_NODE_A1_2
543 D,NODE_B1_2,UY,UYNODE_A1_2 - DLOCYNODE_B1_2_NODE_A1_2
```

```
D,NODE_B1_2,UZ,UZNODE_A1_2 - DLOCZNODE_B1_2_NODE_A1_2


*GET,UXNODE_A1_3,NODE,NODE_A1_3_NUM,U,X
*GET,UYNODE_A1_3,NODE,NODE_A1_3_NUM,U,Y
*GET,UZNODE_A1_3,NODE,NODE_A1_3_NUM,U,Z

*GET,LOCXNODE_A1_3,NODE,NODE_A1_3_NUM,LOC,X
*GET,LOCXNODE_B1_3,NODE,NODE_B1_3_NUM,LOC,X
*GET,LOCYNODE_A1_3,NODE,NODE_A1_3_NUM,LOC,Y
*GET,LOCYNODE_B1_3,NODE,NODE_B1_3_NUM,LOC,Y
*GET,LOCZNODE_A1_3,NODE,NODE_A1_3_NUM,LOC,Z
*GET,LOCZNODE_B1_3,NODE,NODE_B1_3_NUM,LOC,Z

DLOCXNODE_B1_3_NODE_A1_3 = LOCXNODE_B1_3 - LOCXNODE_A1_3
DLOCYNODE_B1_3_NODE_A1_3 = LOCYNODE_B1_3 - LOCYNODE_A1_3
DLOCZNODE_B1_3_NODE_A1_3 = LOCZNODE_B1_3 - LOCZNODE_A1_3

D,NODE_B1_3,UX,UXNODE_A1_3 - DLOCXNODE_B1_3_NODE_A1_3
D,NODE_B1_3,UY,UYNODE_A1_3 - DLOCYNODE_B1_3_NODE_A1_3
D,NODE_B1_3,UZ,UZNODE_A1_3 - DLOCZNODE_B1_3_NODE_A1_3


*GET,UXNODE_A1_4,NODE,NODE_A1_4_NUM,U,X
*GET,UYNODE_A1_4,NODE,NODE_A1_4_NUM,U,Y
*GET,UZNODE_A1_4,NODE,NODE_A1_4_NUM,U,Z

*GET,LOCXNODE_A1_4,NODE,NODE_A1_4_NUM,LOC,X
*GET,LOCXNODE_B1_4,NODE,NODE_B1_4_NUM,LOC,X
*GET,LOCYNODE_A1_4,NODE,NODE_A1_4_NUM,LOC,Y
*GET,LOCYNODE_B1_4,NODE,NODE_B1_4_NUM,LOC,Y
*GET,LOCZNODE_A1_4,NODE,NODE_A1_4_NUM,LOC,Z
*GET,LOCZNODE_B1_4,NODE,NODE_B1_4_NUM,LOC,Z

DLOCXNODE_B1_4_NODE_A1_4 = LOCXNODE_B1_4 - LOCXNODE_A1_4
DLOCYNODE_B1_4_NODE_A1_4 = LOCYNODE_B1_4 - LOCYNODE_A1_4
DLOCZNODE_B1_4_NODE_A1_4 = LOCZNODE_B1_4 - LOCZNODE_A1_4

D,NODE_B1_4,UX,UXNODE_A1_4 - DLOCXNODE_B1_4_NODE_A1_4
D,NODE_B1_4,UY,UYNODE_A1_4 - DLOCYNODE_B1_4_NODE_A1_4
D,NODE_B1_4,UZ,UZNODE_A1_4 - DLOCZNODE_B1_4_NODE_A1_4


*GET,UXNODE_A1_5,NODE,NODE_A1_5_NUM,U,X
*GET,UYNODE_A1_5,NODE,NODE_A1_5_NUM,U,Y
*GET,UZNODE_A1_5,NODE,NODE_A1_5_NUM,U,Z

*GET,LOCXNODE_A1_5,NODE,NODE_A1_5_NUM,LOC,X
*GET,LOCXNODE_B1_5,NODE,NODE_B1_5_NUM,LOC,X
*GET,LOCYNODE_A1_5,NODE,NODE_A1_5_NUM,LOC,Y
*GET,LOCYNODE_B1_5,NODE,NODE_B1_5_NUM,LOC,Y
*GET,LOCZNODE_A1_5,NODE,NODE_A1_5_NUM,LOC,Z
*GET,LOCZNODE_B1_5,NODE,NODE_B1_5_NUM,LOC,Z

DLOCXNODE_B1_5_NODE_A1_5 = LOCXNODE_B1_5 - LOCXNODE_A1_5
DLOCYNODE_B1_5_NODE_A1_5 = LOCYNODE_B1_5 - LOCYNODE_A1_5
DLOCZNODE_B1_5_NODE_A1_5 = LOCZNODE_B1_5 - LOCZNODE_A1_5

D,NODE_B1_5,UX,UXNODE_A1_5 - DLOCXNODE_B1_5_NODE_A1_5
D,NODE_B1_5,UY,UYNODE_A1_5 - DLOCYNODE_B1_5_NODE_A1_5
D,NODE_B1_5,UZ,UZNODE_A1_5 - DLOCZNODE_B1_5_NODE_A1_5

SOLVE   !Solve current study
FINISH  !Exits normally from a processor

/POST1 !Initialize post1 processor
PLDISP,0 !Check whether the nodes align properly
```

Listing B.5: Loadstep 2-3 of assembling the non-Euclidean vertex.

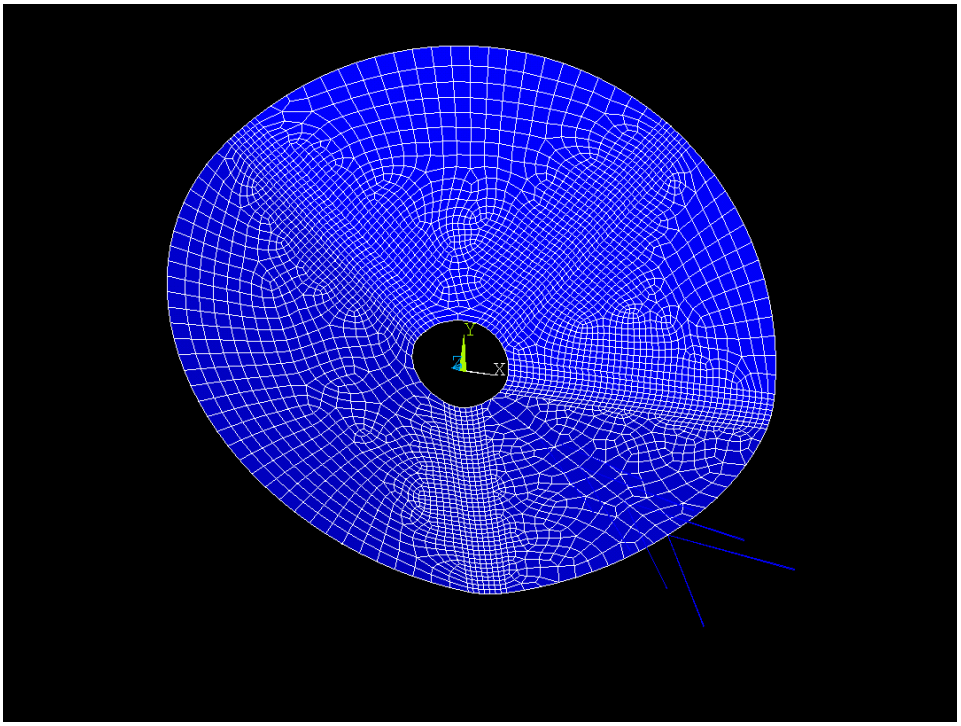

Figure B.3: Screenshot of the model after loadstep 3 of the assembly procedure.

```
611   !!!=== Step 4 part 1 of the procedure (step needs to be divided over two loadsteps)
612 /SOLU   !Initialize solution processor
613 ANTYPE, 0, restart, 3, last, continue !Continue onwards from the end of the last load step
614 AUTOTS, off !Switch auto timestepping off when coupling nodes
615 NSUBST,1 !Use 1 substep when coupling nodes
616
617 FDELE,ALL !Delete all applied forces (as they are zero)
618
619   !!!=== Delete all displacements applied to B1_n, as A1_n will lead these nodes from now on
620
621 DDELE,NODE_B1_1,ALL
622 DDELE,NODE_B1_2,ALL
623 DDELE,NODE_B1_3,ALL
624 DDELE,NODE_B1_4,ALL
625 DDELE,NODE_B1_5,ALL
626
627   !!!=== Coupling of A1_n to B1_n. Process explained for A1_1 and B1_1, and repeated for the
          rest.
628
629 CP,NEXT,UX,NODE_A1_1_NUM,NODE_B1_1_NUM !Couple nodes in UX, first number is the primary node
      (A1_1), second number is the node for which the DOF is deleted (B1_1)
630 CP,NEXT,UY,NODE_A1_1_NUM,NODE_B1_1_NUM !Couple nodes in UY, first number is the primary node
      (A1_1), second number is the node for which the DOF is deleted (B1_1)
631 CP,NEXT,UZ,NODE_A1_1_NUM,NODE_B1_1_NUM !Couple nodes in UZ, first number is the primary node
      (A1_1), second number is the node for which the DOF is deleted (B1_1)
632
633 CP,NEXT,UX,NODE_A1_2_NUM,NODE_B1_2_NUM
634 CP,NEXT,UY,NODE_A1_2_NUM,NODE_B1_2_NUM
635 CP,NEXT,UZ,NODE_A1_2_NUM,NODE_B1_2_NUM
636
637 CP,NEXT,UX,NODE_A1_3_NUM,NODE_B1_3_NUM
638 CP,NEXT,UY,NODE_A1_3_NUM,NODE_B1_3_NUM
639 CP,NEXT,UZ,NODE_A1_3_NUM,NODE_B1_3_NUM
640
641 CP,NEXT,UX,NODE_A1_4_NUM,NODE_B1_4_NUM
642 CP,NEXT,UY,NODE_A1_4_NUM,NODE_B1_4_NUM
643 CP,NEXT,UZ,NODE_A1_4_NUM,NODE_B1_4_NUM
644
645 CP,NEXT,UX,NODE_A1_5_NUM,NODE_B1_5_NUM
```

```
CP,NEXT,UY,NODE_A1_5_NUM,NODE_B1_5_NUM
CP,NEXT,UZ,NODE_A1_5_NUM,NODE_B1_5_NUM

SOLVE   !Solve current study
FINISH  !Exits normally from a processor


  !!!=== Step 4 part 2 of the procedure (step needs to be divided over two loadsteps)
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 4, last, continue !Continue onwards from the end of the last load step
AUTOTS, off !Switch auto timestepping off when applying reaction forces
NSUBST,1 !Use 1 substep when applying reaction forces

  !!!=== Replace all displacements applied to A1_n with their corresponding reaction forces

DDELE,NODE_A1_1,ALL
DDELE,NODE_A1_2,ALL
DDELE,NODE_A1_3,ALL
DDELE,NODE_A1_4,ALL
DDELE,NODE_A1_5,ALL

*GET, Xforce_NODE_A1_1, NODE, NODE_A1_1_NUM, RF, FX
*GET, Yforce_NODE_A1_1, NODE, NODE_A1_1_NUM, RF, FY
*GET, Zforce_NODE_A1_1, NODE, NODE_A1_1_NUM, RF, FZ
F,NODE_A1_1,FX,Xforce_NODE_A1_1
F,NODE_A1_1,FY,Yforce_NODE_A1_1
F,NODE_A1_1,FZ,Zforce_NODE_A1_1

*GET, Xforce_NODE_A1_2, NODE, NODE_A1_2_NUM, RF, FX
*GET, Yforce_NODE_A1_2, NODE, NODE_A1_2_NUM, RF, FY
*GET, Zforce_NODE_A1_2, NODE, NODE_A1_2_NUM, RF, FZ
F,NODE_A1_2,FX,Xforce_NODE_A1_2
F,NODE_A1_2,FY,Yforce_NODE_A1_2
F,NODE_A1_2,FZ,Zforce_NODE_A1_2

*GET, Xforce_NODE_A1_3, NODE, NODE_A1_3_NUM, RF, FX
*GET, Yforce_NODE_A1_3, NODE, NODE_A1_3_NUM, RF, FY
*GET, Zforce_NODE_A1_3, NODE, NODE_A1_3_NUM, RF, FZ
F,NODE_A1_3,FX,Xforce_NODE_A1_3
F,NODE_A1_3,FY,Yforce_NODE_A1_3
F,NODE_A1_3,FZ,Zforce_NODE_A1_3

*GET, Xforce_NODE_A1_4, NODE, NODE_A1_4_NUM, RF, FX
*GET, Yforce_NODE_A1_4, NODE, NODE_A1_4_NUM, RF, FY
*GET, Zforce_NODE_A1_4, NODE, NODE_A1_4_NUM, RF, FZ
F,NODE_A1_4,FX,Xforce_NODE_A1_4
F,NODE_A1_4,FY,Yforce_NODE_A1_4
F,NODE_A1_4,FZ,Zforce_NODE_A1_4

*GET, Xforce_NODE_A1_5, NODE, NODE_A1_5_NUM, RF, FX
*GET, Yforce_NODE_A1_5, NODE, NODE_A1_5_NUM, RF, FY
*GET, Zforce_NODE_A1_5, NODE, NODE_A1_5_NUM, RF, FZ
F,NODE_A1_5,FX,Xforce_NODE_A1_5
F,NODE_A1_5,FY,Yforce_NODE_A1_5
F,NODE_A1_5,FZ,Zforce_NODE_A1_5

  !!!=== Fix two translational DOFs of A2_n and B2_n

D,1,UX,%_FIX%
D,1,UY,%_FIX%
D,2,UX,%_FIX%
D,2,UY,%_FIX%
D,3,UX,%_FIX%
D,3,UY,%_FIX%
D,4,UX,%_FIX%
D,4,UY,%_FIX%
D,5,UX,%_FIX%
D,5,UY,%_FIX%

D,11,UX,%_FIX%
D,11,UY,%_FIX%
```

```
D,12,UX,%_FIX%
D,12,UY,%_FIX%
D,13,UX,%_FIX%
D,13,UY,%_FIX%
D,14,UX,%_FIX%
D,14,UY,%_FIX%
D,15,UX,%_FIX%
D,15,UY,%_FIX%

SOLVE   !Solve current study
FINISH  !Exits normally from a processor


  !!!=== Loadstep 5 of the procedure
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 5, last, continue !Continue onwards from the end of the last load step
AUTOTS, on !Use automatic time stepping
NSUBST,50,,30 !Specifies the size of substeps. If automatic time stepping is on, the second and third numbers specify the maximum and minimum amount of substeps.

  !!!=== Displace B2_n to A2_n. Process explained for B2_1 and A2_1, and repeated for the rest.

!Get displacement of A2_1 relative to its initial position
*GET,UX_A2_1,NODE,1,U,X
*GET,UY_A2_1,NODE,1,U,Y

!Get initial position of A2_1 and B2_1
*GET,LOCX_A2_1,NODE,1,LOC,X
*GET,LOCY_A2_1,NODE,1,LOC,Y
*GET,LOCX_B2_1,NODE,11,LOC,X
*GET,LOCY_B2_1,NODE,11,LOC,Y

!Get distances between initial positions of A2_1 and B2_1
DLOCXAB2_1 = LOCX_B2_1 - LOCX_A2_1
DLOCYAB2_1 = LOCY_B2_1 - LOCY_A2_1

!Displace B2_1 based on the displacement of A2_1 relative to its initial position and the distances between initial positions of A2_1 and B2_1
D,11,UX,UX_A2_1 - DLOCXAB2_1
D,11,UY,UY_A2_1 - DLOCYAB2_1


*GET,UX_A2_2,NODE,2,U,X
*GET,UY_A2_2,NODE,2,U,Y

*GET,LOCX_A2_2,NODE,2,LOC,X
*GET,LOCY_A2_2,NODE,2,LOC,Y
*GET,LOCX_B2_2,NODE,12,LOC,X
*GET,LOCY_B2_2,NODE,12,LOC,Y

DLOCXAB2_2 = LOCX_B2_2 - LOCX_A2_2
DLOCYAB2_2 = LOCY_B2_2 - LOCY_A2_2

D,12,UX,UX_A2_2 - DLOCXAB2_2
D,12,UY,UY_A2_2 - DLOCYAB2_2


*GET,UX_A2_3,NODE,3,U,X
*GET,UY_A2_3,NODE,3,U,Y

*GET,LOCX_A2_3,NODE,3,LOC,X
*GET,LOCY_A2_3,NODE,3,LOC,Y
*GET,LOCX_B2_3,NODE,13,LOC,X
*GET,LOCY_B2_3,NODE,13,LOC,Y

DLOCXAB2_3 = LOCX_B2_3 - LOCX_A2_3
DLOCYAB2_3 = LOCY_B2_3 - LOCY_A2_3

D,13,UX,UX_A2_3 - DLOCXAB2_3
D,13,UY,UY_A2_3 - DLOCYAB2_3
```

```
785
786
787 *GET,UX_A2_4,NODE,4,U,X
788 *GET,UY_A2_4,NODE,4,U,Y
789
790 *GET,LOCX_A2_4,NODE,4,LOC,X
791 *GET,LOCY_A2_4,NODE,4,LOC,Y
792 *GET,LOCX_B2_4,NODE,14,LOC,X
793 *GET,LOCY_B2_4,NODE,14,LOC,Y
794
795 DLOCXAB2_4 = LOCX_B2_4 - LOCX_A2_4
796 DLOCYAB2_4 = LOCY_B2_4 - LOCY_A2_4
797
798 D,14,UX,UX_A2_4 - DLOCXAB2_4
799 D,14,UY,UY_A2_4 - DLOCYAB2_4
800
801
802 *GET,UX_A2_5,NODE,5,U,X
803 *GET,UY_A2_5,NODE,5,U,Y
804
805 *GET,LOCX_A2_5,NODE,5,LOC,X
806 *GET,LOCY_A2_5,NODE,5,LOC,Y
807 *GET,LOCX_B2_5,NODE,15,LOC,X
808 *GET,LOCY_B2_5,NODE,15,LOC,Y
809
810 DLOCXAB2_5 = LOCX_B2_5 - LOCX_A2_5
811 DLOCYAB2_5 = LOCY_B2_5 - LOCY_A2_5
812
813 D,15,UX,UX_A2_5 - DLOCXAB2_5
814 D,15,UY,UY_A2_5 - DLOCYAB2_5
815
816 SOLVE   !Solve current study
817 FINISH  !Exits normally from a processor
818
819 /POST1 !Initialize post1 processor
820 PLDISP,0 !Check whether the nodes align properly
```

Listing B.6: Loadstep 4-5 of assembling the non-Euclidean vertex.

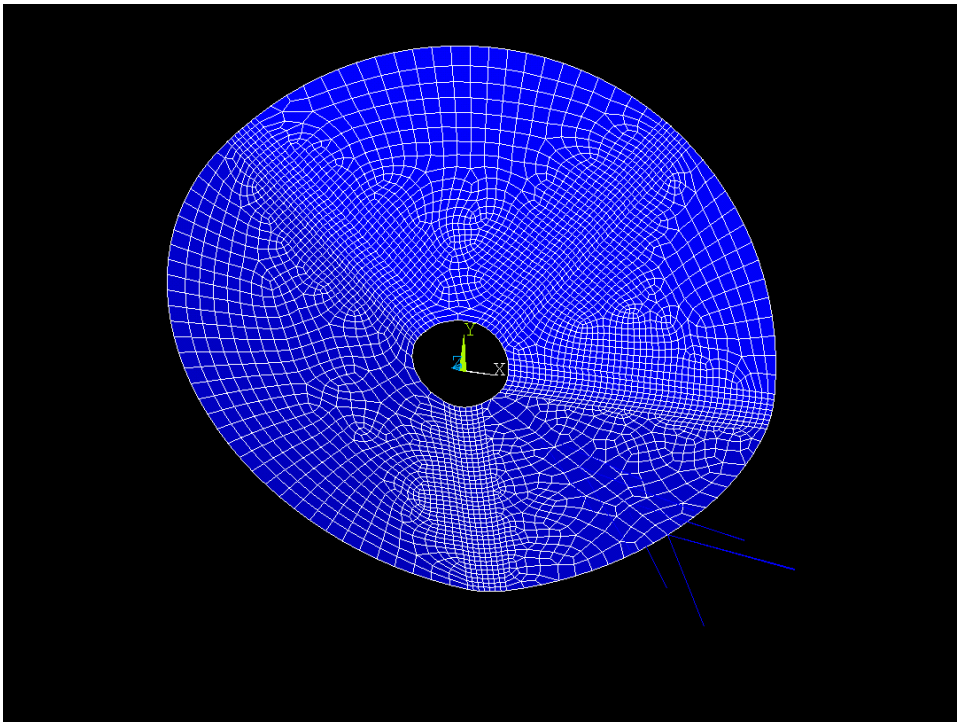

Figure B.4: Screenshot of the model after loadstep 5 of the assembly procedure.

```
  !!!=== Step 6 part 1 of the procedure (step needs to be divided over two loadsteps)
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 6, last, continue !Continue onwards from the end of the last load step
AUTOTS, off !Switch auto timestepping off when coupling nodes
NSUBST,1 !Use 1 substep when coupling nodes

  !!!=== Delete all displacements applied to B2_n, as A2_n will lead this node from now on

DDELE,11,ALL
DDELE,12,ALL
DDELE,13,ALL
DDELE,14,ALL
DDELE,15,ALL

  !!!=== Coupling of A2_n to B2_n. Process explained for A2_1 and B2_1, and repeated for the
      rest.

CP,NEXT,UX,1,11 !Couple nodes in UX, first number is the primary node (A2_1), second number
    is the node for which the DOF is deleted (B2_1)
CP,NEXT,UY,1,11 !Couple nodes in UY, first number is the primary node (A2_1), second number
    is the node for which the DOF is deleted (B2_1)

CP,NEXT,UX,2,12
CP,NEXT,UY,2,12

CP,NEXT,UX,3,13
CP,NEXT,UY,3,13

CP,NEXT,UX,4,14
CP,NEXT,UY,4,14

CP,NEXT,UX,5,15
CP,NEXT,UY,5,15

SOLVE   !Solve current study
FINISH  !Exits normally from a processor


  !!!=== Step 6 part 2 of the procedure (step needs to be divided over two loadsteps)
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 7, last, continue !Continue onwards from the end of the last load step
AUTOTS, off !Switch auto timestepping off when applying reaction forces
NSUBST,1 !Use 1 substep when applying reaction forces

  !!!=== Replace all displacements applied to A2_n with their corresponding reaction forces

DDELE,1,ALL
DDELE,2,ALL
DDELE,3,ALL
DDELE,4,ALL
DDELE,5,ALL

*GET, Xforce_1, NODE, 1, RF, FX
*GET, Yforce_1, NODE, 1, RF, FY
F,1,FX,Xforce_1
F,1,FY,Yforce_1

*GET, Xforce_2, NODE, 2, RF, FX
*GET, Yforce_2, NODE, 2, RF, FY
F,2,FX,Xforce_2
F,2,FY,Yforce_2

*GET, Xforce_3, NODE, 3, RF, FX
*GET, Yforce_3, NODE, 3, RF, FY
F,3,FX,Xforce_3
F,3,FY,Yforce_3

*GET, Xforce_4, NODE, 4, RF, FX
*GET, Yforce_4, NODE, 4, RF, FY
F,4,FX,Xforce_4
```

```
F,4,FY,Yforce_4

*GET, Xforce_5, NODE, 5, RF, FX
*GET, Yforce_5, NODE, 5, RF, FY
F,5,FX,Xforce_5
F,5,FY,Yforce_5

  !!!=== Fix one translational DOF of A3_n and B3_n

D,6,UZ,%_FIX%
D,7,UZ,%_FIX%
D,8,UZ,%_FIX%
D,9,UZ,%_FIX%
D,10,UZ,%_FIX%

D,16,UZ,%_FIX%
D,17,UZ,%_FIX%
D,18,UZ,%_FIX%
D,19,UZ,%_FIX%
D,20,UZ,%_FIX%

SOLVE   !Solve current study
FINISH  !Exits normally from a processor


  !!!=== Loadstep 7 of the procedure
/SOLU   !Initialize solution processor
ANTYPE, 0, restart, 8, last, continue !Continue onwards from the end of the last load step
AUTOTS, on !Use automatic time stepping
NSUBST,30,,10 !Specifies the size of substeps. If automatic time stepping is on, the second
    and third numbers specify the maximum and minimum amount of substeps.

  !!!=== Displace B3_n to A3_n. Process explained for B3_1 and A3_1, and repeated for the
      rest.

*GET,UZ6,NODE,6,U,Z !Get displacement of A3_1 relative to its initial position
*GET,LOCZ16,NODE,16,LOC,Z !Get initial position of B3_1
*GET,LOCZ6,NODE,6,LOC,Z !Get initial position of A3_1
DLOCZ16_6 = LOCZ16 - LOCZ6 !Get distances between initial positions of A3_1 and B3_1
D,16,UZ,UZ6 - DLOCZ16_6 !Displace B3_1 based on the displacement of A3_1 relative to its
    initial position and the distances between initial positions of A3_1 and B3_1

*GET,UZ7,NODE,7,U,Z
*GET,LOCZ17,NODE,17,LOC,Z
*GET,LOCZ7,NODE,7,LOC,Z
DLOCZ17_7 = LOCZ17 - LOCZ7
D,17,UZ,UZ7 - DLOCZ17_7

*GET,UZ8,NODE,8,U,Z
*GET,LOCZ18,NODE,18,LOC,Z
*GET,LOCZ8,NODE,8,LOC,Z
DLOCZ18_8 = LOCZ18 - LOCZ8
D,18,UZ,UZ8 - DLOCZ18_8

*GET,UZ9,NODE,9,U,Z
*GET,LOCZ19,NODE,19,LOC,Z
*GET,LOCZ9,NODE,9,LOC,Z
DLOCZ19_9 = LOCZ19 - LOCZ9
D,19,UZ,UZ9 - DLOCZ19_9

*GET,UZ10,NODE,10,U,Z
*GET,LOCZ20,NODE,20,LOC,Z
*GET,LOCZ10,NODE,10,LOC,Z
DLOCZ20_10 = LOCZ20 - LOCZ10
D,20,UZ,UZ10 - DLOCZ20_10

SOLVE   !Solve current study
FINISH  !Exits normally from a processor

/POST1 !Initialize post1 processor
PLDISP,0 !Check whether the nodes align properly
```

Listing B.7: Loadstep 6-7 of assembling the non-Euclidean vertex.

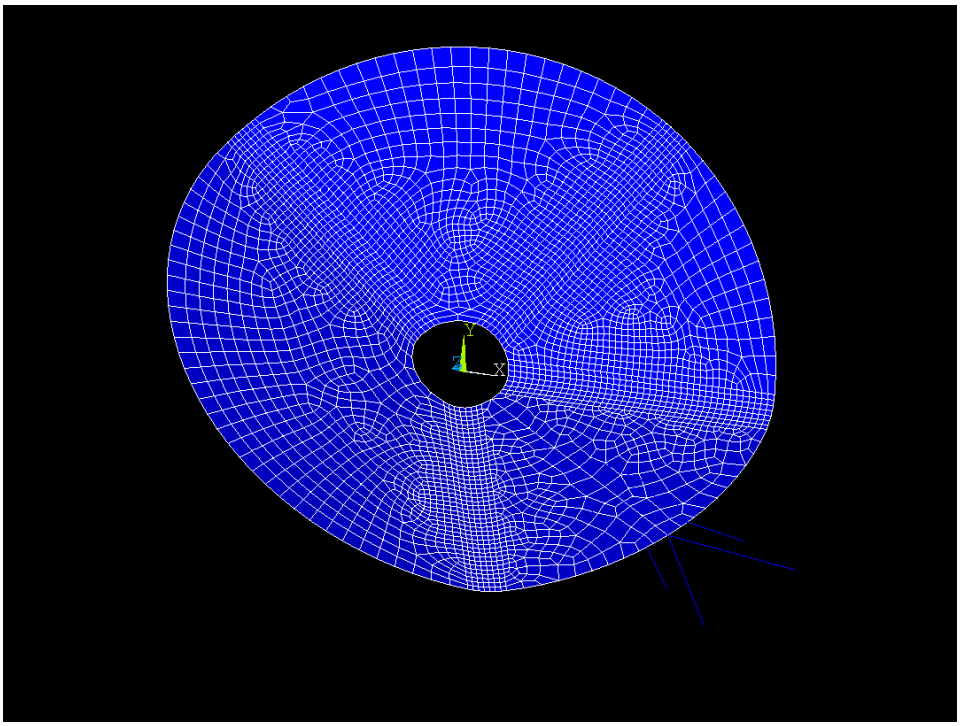

Figure B.5: Screenshot of the model after loadstep 7 of the assembly procedure.

```
956   !!!=== Step 8 part 1 of the procedure (step needs to be divided over two loadsteps)
957 /SOLU   !Initialize solution processor
958 ANTYPE, 0, restart, 9, last, continue !Continue onwards from the end of the last load step
959 AUTOTS, off !Switch auto timestepping off when coupling nodes
960 NSUBST,1 !Use 1 substep when coupling nodes
961
962   !!!=== Delete all displacements applied to B3_n, as A3_n will lead this node from now on
963
964 DDELE,16,ALL
965 DDELE,17,ALL
966 DDELE,18,ALL
967 DDELE,19,ALL
968 DDELE,20,ALL
969
970   !!!=== Coupling of A3_n to B3_n. Process explained for A3_1 and B3_1, and repeated for the
          rest.
971
972 CP,NEXT,UZ,6,16 !Couple nodes in UZ, first number is the primary node (A3_1), second number
      is the node for which the DOF is deleted (B3_1)
973 CP,NEXT,UZ,7,17
974 CP,NEXT,UZ,8,18
975 CP,NEXT,UZ,9,19
976 CP,NEXT,UZ,10,20
977
978 SOLVE   !Solve current study
979 FINISH  !Exits normally from a processor
980
981
982   !!!=== Step 8 part 2 of the procedure (step needs to be divided over two loadsteps)
983 /SOLU   !Initialize solution processor
984 ANTYPE, 0, restart, 10, last, continue !Continue onwards from the end of the last load step
985 AUTOTS, off !Switch auto timestepping off when applying reaction forces
986 NSUBST,1 !Use 1 substep when applying reaction forces
```

```
987
988   !!!=== Replace all displacements applied to A3_n with their corresponding reaction forces
989
990 DDELE,6,ALL
991 DDELE,7,ALL
992 DDELE,8,ALL
993 DDELE,9,ALL
994 DDELE,10,ALL
995
996 *GET, Zforce_6, NODE, 6, RF, FZ
997 F,6,FZ,Zforce_6
998
999 *GET, Zforce_7, NODE, 7, RF, FZ
1000 F,7,FZ,Zforce_7
1001
1002 *GET, Zforce_8, NODE, 8, RF, FZ
1003 F,8,FZ,Zforce_8
1004
1005 *GET, Zforce_9, NODE, 9, RF, FZ
1006 F,9,FZ,Zforce_9
1007
1008 *GET, Zforce_10, NODE, 10, RF, FZ
1009 F,10,FZ,Zforce_10
1010
1011 SOLVE   !Solve current study
1012 FINISH  !Exits normally from a processor
1013
1014
1015   !!!=== Loadstep 9 of the procedure
1016 /SOLU   !Initialize solution processor
1017 ANTYPE, 0, restart, 11, last, continue !Continue onwards from the end of the last load step
1018 AUTOTS, on !Use automatic time stepping
1019 NSUBST,50,,30 !Specifies the size of substeps. If automatic time stepping is on, the second
        and third numbers specify the maximum and minimum amount of substeps.
1020
1021   !!!=== Stepwise reduce all remaining forces to zero
1022
1023 F,NODE_A1_1,FX,0
1024 F,NODE_A1_1,FY,0
1025 F,NODE_A1_1,FZ,0
1026 F,NODE_A1_2,FX,0
1027 F,NODE_A1_2,FY,0
1028 F,NODE_A1_2,FZ,0
1029 F,NODE_A1_3,FX,0
1030 F,NODE_A1_3,FY,0
1031 F,NODE_A1_3,FZ,0
1032 F,NODE_A1_4,FX,0
1033 F,NODE_A1_4,FY,0
1034 F,NODE_A1_4,FZ,0
1035 F,NODE_A1_5,FX,0
1036 F,NODE_A1_5,FY,0
1037 F,NODE_A1_5,FZ,0
1038
1039 F,1,FX,0
1040 F,1,FY,0
1041 F,2,FX,0
1042 F,2,FY,0
1043 F,3,FX,0
1044 F,3,FY,0
1045 F,4,FX,0
1046 F,4,FY,0
1047 F,5,FX,0
1048 F,5,FY,0
1049
1050 F,6,FZ,0
1051 F,7,FZ,0
1052 F,8,FZ,0
1053 F,9,FZ,0
1054 F,10,FZ,0
1055
1056 SOLVE   !Solve current study
```

```
1057 FINISH  !Exits normally from a processor
1058
1059 /POST1 !Initialize post1 processor
1060 PLDISP,0 !Check whether the assembly of the vertex is complete and correct
1061
1062     !!!=== Assembly of non-Euclidean vertex complete
```

Listing B.8: Loadstep 8-9 of assembling the non-Euclidean vertex.

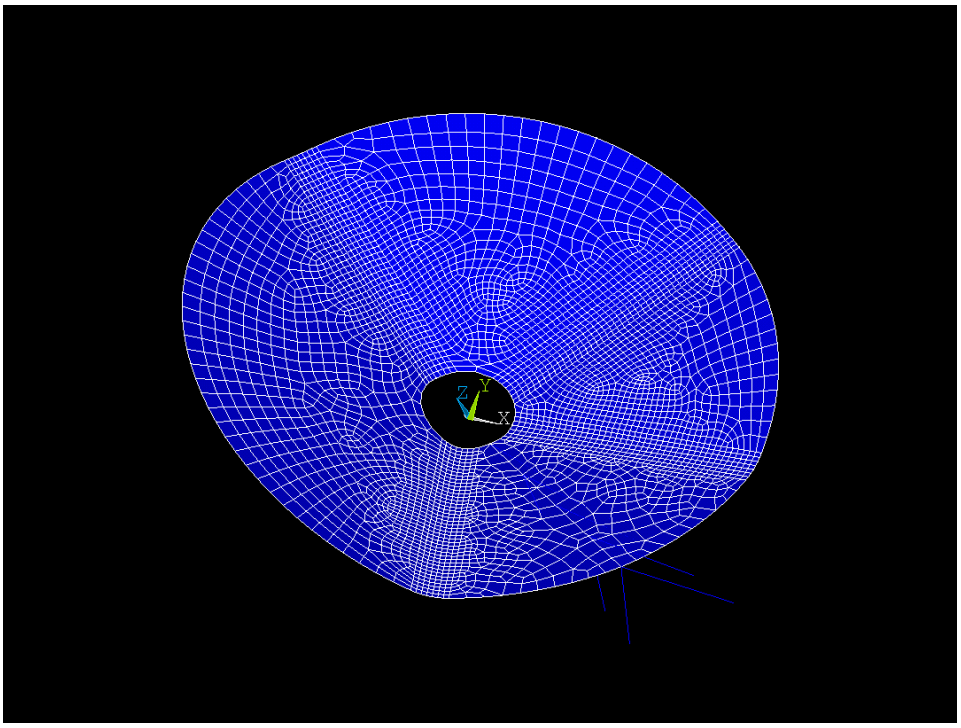

Figure B.6: Screenshot of the model after all loadsteps of the assembly procedure.

C

Additional information case study

This section of the appendix provides additional information on the case study discussed in the main paper. The first part is focussed on details of the finite element method model, while the second part provides details regarding fabrication and testing.

C.1. Finite Element Method

Figure C.1 shows the steps that were applied to assemble the case study mechanism in ANSYS APDL. This assembly order was chosen to avoid multistabilities during the assembly process, but an assembly order starting with assembling the non-Euclidean vertices would also work.

Figure C.2 shows the stress/strain plots of the three material models that were applied to the FEM analysis. Table C.1 lists the parametric values of the same material models.

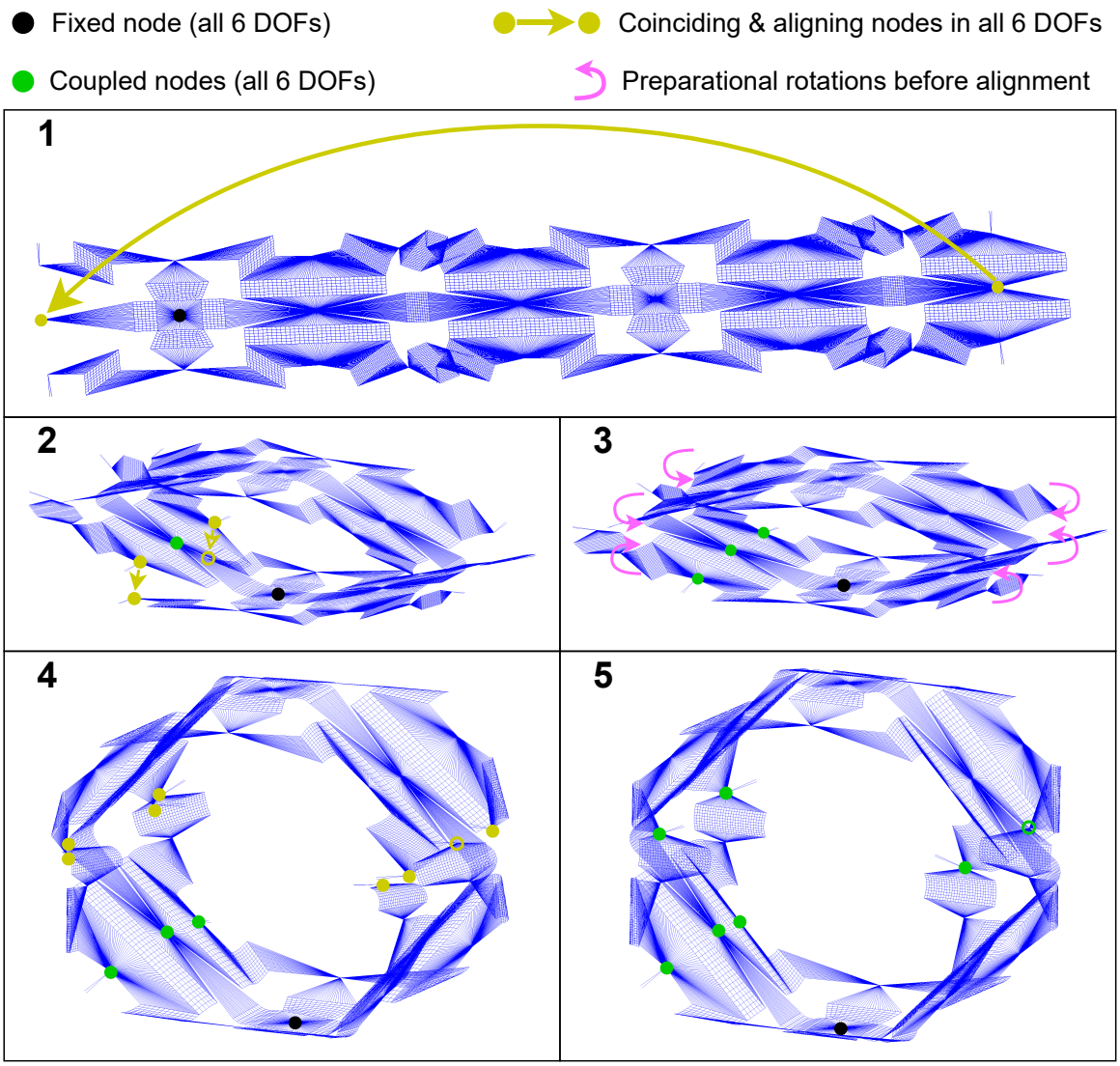

Figure C.1: Stepwise visualisation of the actual assembly procedure in FEM leading to the fully assembled case study mechanism.

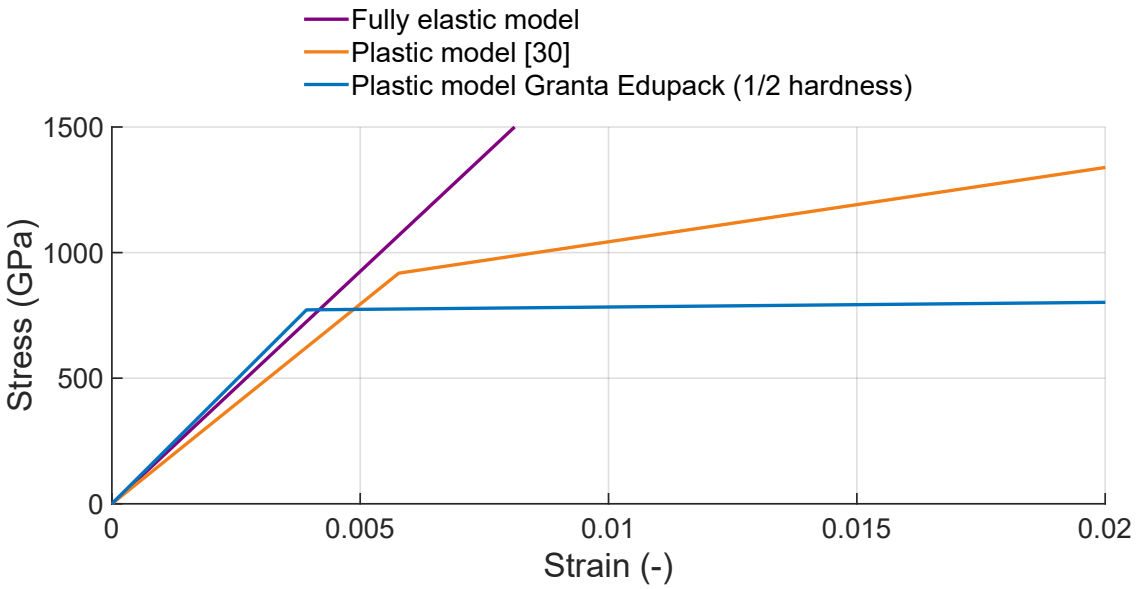

Figure C.2: Stress/strain plots of used material models in case study FEM analysis. The stainless steel alloy is 1.4310.

Material Model	Young's Modulus (GPa)	Yield Stress (MPa)	Tangent Modulus (GPa)
Fully Elastic Model	185	–	–
Plastic Model [30]	159	918	30.5
Plastic Model Granta Edupack	197	772	2.64

Table C.1: Parameter values of used material models in case study FEM analysis. The stainless steel alloy is 1.4310.

C.2. Fabrication & Testing

Figure C.3 shows how the components of the case study mechanism are assembled to form the complete prototype.

Figure C.4 shows the average test data per prototype per test moment. The average is taken from a total of 15 loading cycles, as the prototypes are clamped three times, and loaded five times per clamping setup.

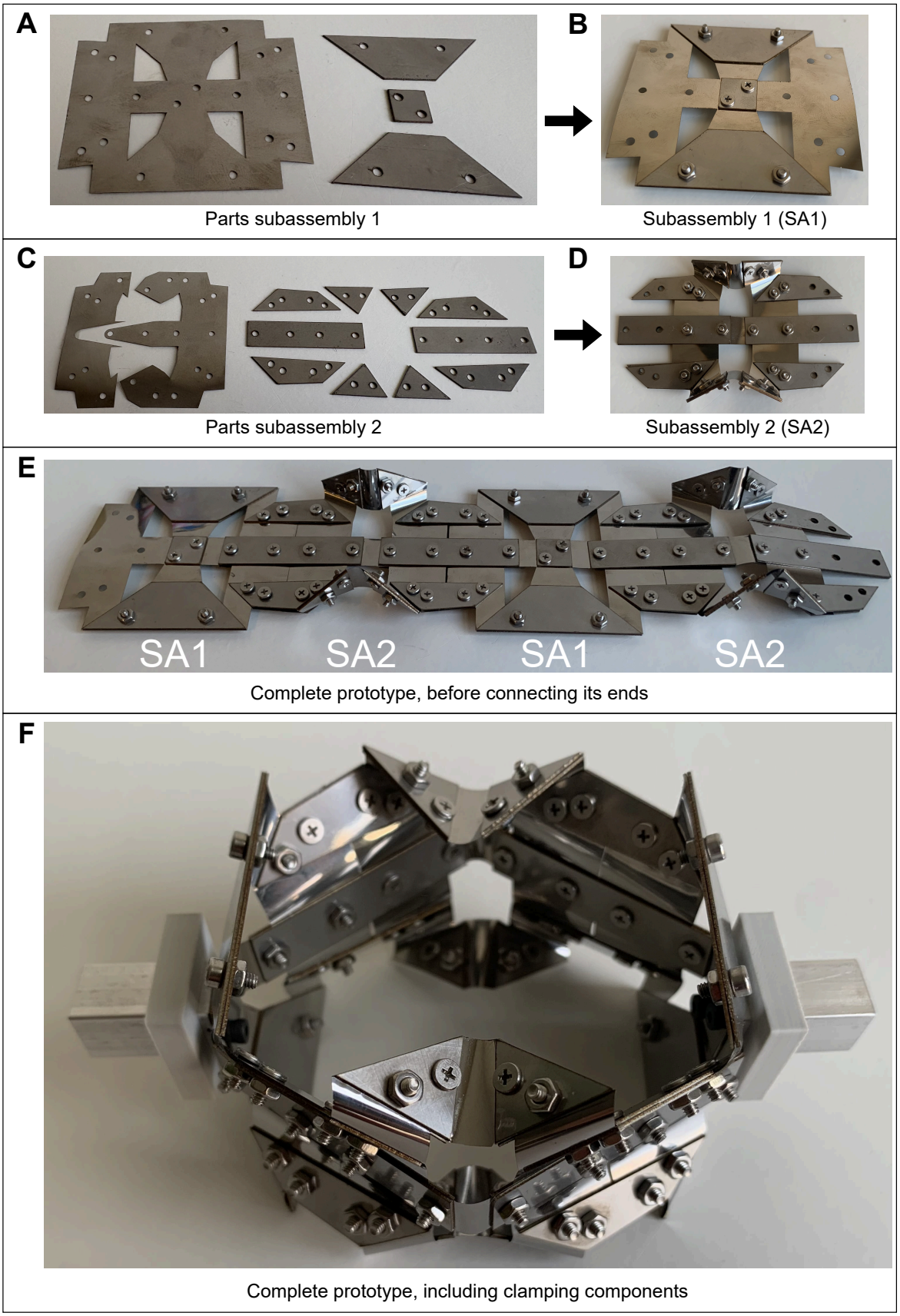

Figure C.3: Overview of components and physical assembly steps needed for a fully assembled case study prototype.

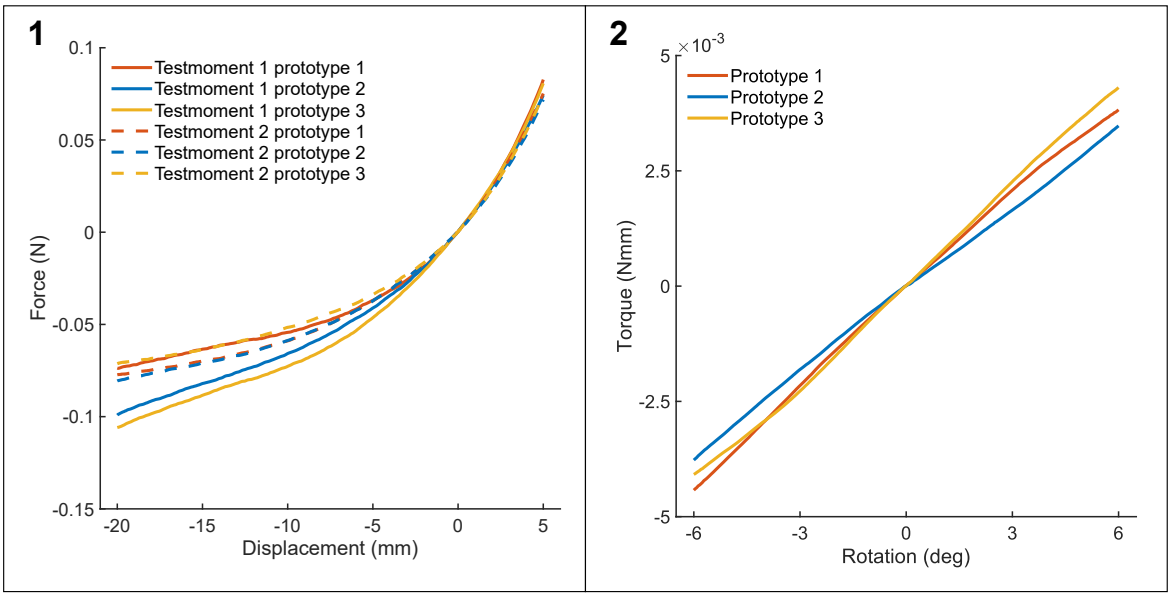

Figure C.4: Average test data per prototype per test moment.

D

Literature review

Literature Review for Designing an Origami-Inspired Linear Guide

by

Max Benninga

Student Name	Student Number
Max Benninga	4956710

Supervisor: D. Farhadi Machekposhti
Supervisor: M. Zhang
Faculty: Faculty of Mechanical Engineering, Delft

TU Delft

Abstract

Compliant linear guides are vital components in high-tech machinery due to their vacuum compatibility and high-precision. Unfortunately, all current state of the art compliant linear guides suffer from one or more drawbacks. This is why countless researchers work on developing novel and superior compliant linear guides. Whereas these researchers often investigate initially curved flexure mechanisms, this paper explores the possibility to employ origami mechanisms as possible solutions.

This literature review discusses all necessary subjects to start designing, analysing, manufacturing and testing origami-inspired linear guides. Besides this, the current state of the art of compliant linear guides is also elaborated for comparing purposes. This paper serves as a stepping-stone to create an origami linear guide that should rival or surpass current state of the art compliant linear guides. Additionally, it lays the groundwork for future research into a wider variety of subjects related to origami mechanisms.

Contents

1

Introduction

Today's industry is in constant need of more advanced mechanisms to accommodate higher precision, speed and special needs such as vacuum compatibility. This also applies to one of the most fundamental building blocks in machines, linear guides. Linear guides are mechanisms that allow for a linear translation of its End-Effector (EE), while constraining all other translations and rotations of the end-effector. The performance characteristics of an ideal linear guide are: zero parasitic displacements, zero stiffness in the degree of freedom, infinite stiffness in the degrees of constraint and infinite range of motion along the degree of freedom. Originally, linear guides consist of ball bearings, roller bearings or slide bearings. All of these examples generate particles in operation and can therefore not be used in vacuum environments. The current main solution for this issue is the use of compliant mechanisms as linear guides, like parallelogram joints [1, 2], folded leaf spring mechanisms [3, 4], diaphragm mechanisms [5, 6] or initially curved flexure mechanisms [4, 7, 8]. All of these individual solutions have their own strengths and weaknesses, which means that there is still room for novel and better solutions. The strengths and weaknesses of these compliant linear guides will be discussed in more detail in Chapter 2, Literature.

The ancient art of origami has gradually worked its way into science and engineering. The practice, originally performed for aesthetic and traditional reasons, is used as an inspiration for a wide variety of applications. Advancements in mathematics help to investigate origami in a more scientific way, which opened the door for functional origami. The most straight-forward functionality of origami is its capacity to be stored and transported in a flat configuration, while it can be transformed into an arbitrary 3D shape when needed. If this transformation happens only during deployment, this is called *static origami*. These static origami models are, for example, used in structural applications [9], space applications [10, 11], emergency housing [12] or safety equipment [13].

Another important functionality of origami is its ability to use its crease lines as hinges, which makes them behave like compliant mechanisms. Going by the definition of Howell, "Compliant mechanisms gain their motion from the deflection of flexible members rather than from traditional bearings and hinges" [14]. When, at least some of, the creases of an origami model are constantly acting as hinges during operation, the model is called *kinematic origami*. Examples of applications for kinematic origami are: actuators [15], medical devices [16, 17] and locomotion robotics [18, 19].

In this literature review, kinematic origami is examined as a potential new solution for compliant linear guides. Kinematic origami solutions are expected to perform well because of the ability to use coupled secondary deformations to maintain high support stiffness over a large range of motion. The goal of this research is to evaluate literature relevant for the design of an origami-inspired linear guide.

2

Literature

The literature in this chapter was found in a combination of ways. First of all, the search engine Google Scholar was used to find valuable papers for this literature research. Keywords to input in the search engine are mentioned at the start of each section. Next to this main strategy, a lot of papers were found in the references of previously discovered literature. Lastly, the platform *ResearchRabbit* was used occasionally to find similar, earlier and later work of previously discovered literature.

2.1. State of the Art of Compliant Linear Guides

Keywords: compliant/flexure mechanism, linear guide, parallelogram, folded leaf spring, diaphragm, initially curved flexure, constraint stiffness, parasitic displacement, range of motion.

2.1.1. Parallelogram Joints

One of the most used compliant linear guides is the (double) parallelogram joint. The difference between a parallelogram joint and a double parallelogram joint is the addition of a secondary stage. The purpose of this secondary stage is to prevent parasitic displacements. However, the addition of this extra stage does affect the stiffness of the joint in both the degree of freedom (DOF) and the degrees of constraint (DOCs). An undesired effect of adding the secondary stage is the significant decrease in stiffness in the degrees of constraint.

Both these parallelogram flexure modules lose a significant share of their constraint stiffness when displaced in their degree of freedom. This is the main problem with these linear guides. It is found that improving one performance characteristic of beam-based flexure modules, such as range of motion, constraint stiffness or parasitic motion, often is accompanied by a deterioration of another performance characteristic [1, 2]. Figure 2.1 shows how a displacement in x causes a parasitic displacement and decrease in constraint stiffness in y.

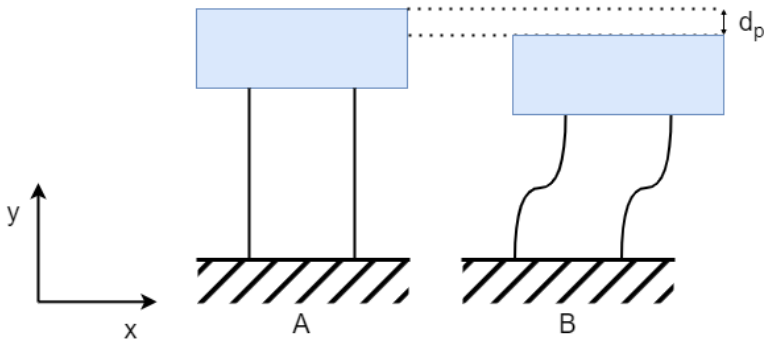

Figure 2.1: Parallelogram joint in its neutral (A) and displaced (B) configuration. When displaced, a parasitic displacement in the y direction takes place. The constraint stiffness in the y direction decreases when the end-effector is displaced in x.

2.1.2. Folded Leaf Spring Mechanisms

A big portion of compliant linear guides consists of folded leaf springs. A normal folded leaf spring constraints just one translation, along its fold line. As a rigid body in 3D space has a total of six degrees of freedom, at least five folded leaf springs are needed to create a linear guide. A variation on an original folded leaf spring is the torsion reinforced folded leaf spring, which constraints one translation and two rotations. Due to these extra constraints, just two torsion reinforced folded leaf springs are needed to create a linear guide [3].

Both folded leaf spring mechanisms possess good range of motion properties due to their long flexure elements. Unfortunately, folded leaf spring mechanisms also lose a significant portion of their constraint stiffness when displaced over their range of motion, just like the compliant parallelogram mechanisms.

2.1.3. Diaphragm Mechanisms

The third type of commonly used linear guide is the diaphragm mechanism. These mechanisms are generally made from a single sheet of material, and can be made using just 2D machining methods. The biggest drawback of this type of mechanism is its relatively small range of motion. This is due to the fact that the material of the diaphragm needs to elongate to move in the out-of-plane direction. This drawback can be resolved by mounting multiple flat diaphragms in series to increase the range of motion of the combined system. It should be noted that many flat diaphragms mounted in series effectively construct a bellows. The downside of combining multiple diaphragms is the reduced constraint stiffness [6].

Another potential disadvantage of diaphragm mechanisms is its susceptibility to rotational parasitic motions. Awtar and Slocum developed two diaphragm mechanisms in an attempt to remove the parasitic rotations of a traditional diaphragm mechanism. The first design was made symmetric, effectively freeing it of parasitic rotations. However, this design suffered from over-constraints, which in turn compromised the range of motion. Their second design was insensitive to parasitic rotations and offered a larger range of motion, but compromised with respect to the constraint stiffness. In general, the design objectives of diaphragm mechanisms are mutually conflicting [5].

2.1.4. Initially Curved Flexure Mechanisms

The current best solution to the problem of decreasing constraint stiffness over a range of motion is the use of initially curved flexures instead of straight flexures. This finding was shown in multiple studies [4, 7, 8].

Initially curved flexures help to maintain constraint stiffness due to their ability to provide a relatively high support stiffness, called a stiffness singularity, at a specific point in their range of motion. Generally, this stiffness singularity takes place at the point where the flexure is deformed to its straight configuration. Using multiple initially curved flexures, which all reach their stiffness singularity at a different point in the mechanism's range of motion, can reduce the loss of constraint stiffness over the mechanism's range of motion. A good example of this design strategy is provided by Rommers and Herder [4].

Modelling initially curved flexure mechanisms does pose a harder challenge than modelling straight flexures, but research has been done to build these models [20, 21].

2.1.5. Summarising State of the Art Solutions

To conclude the state of the art of compliant linear guides, the current solutions are summarised and their corresponding strengths and weaknesses are qualitatively evaluated in Table 2.1. The term 'relative constraint stiffness' is used to describe the ratio between the lowest stiffness of all degrees of constraint divided by the stiffness in the degree of freedom.

It can be observed in table 2.1 that none of the current mechanisms scores positive in all three criteria.

Initially curved flexures are currently used to help maintain constraint stiffness over the range of motion of a mechanism. In this way, initially curved flexures could potentially be used to make a linear guide that scores positive in all three aspects. However, in this literature review another approach is considered. In the following sections it is explored how origami mechanisms can be used as compliant linear guides that outperform the current state of the art.

Table 2.1: Qualitative evaluation of state of the art compliant linear guides. The design criteria Range of Motion, Parasitic Displacement and Relative Constraint Stiffness are abbreviated to RoM, PD and RCS respectively.

Mechanism	RoM	PD	RCS
Parallelogram joint	+	-	-
Double parallelogram joint	+	+	- -
Folded leaf spring mechanism	+	+	-
Torsion reinforced folded leaf spring mechanism	+	+	-
Traditional diaphragm mechanism	-	-	+
Over-constrained diaphragm mechanism	-	+	+
Low constraint stiffness diaphragm mechanism	+	+	-

2.2. General Origami Principles

Keywords: origami, Graph Theory, Euclidean, developable, rigid foldable, flat foldable, Principle of Three Units, bi/tri/multi-stability, stable state.

2.2.1. Origami Definitions

In order to understand general origami principles, some definitions need to be introduced. Origami consists of two main components: *facets* and *creases*. Facets are the surfaces in between creases. The creases themselves are the lines around which the facets can rotate. A third important concept of origami is the intersection of multiple creases, called a *vertex*. The amount of creases that intersect at a vertex is defined as the *degree* of a vertex.

Lastly, there are two types of angles present in origami models. *Sector angles* describe the angles between creases. *Dihedral angles* represent the angles between the facets of a an origami model. During folding, sector angles stay the same, while dihedral angles change. Figure 2.2 visually explains the concepts mentioned in this subsection.

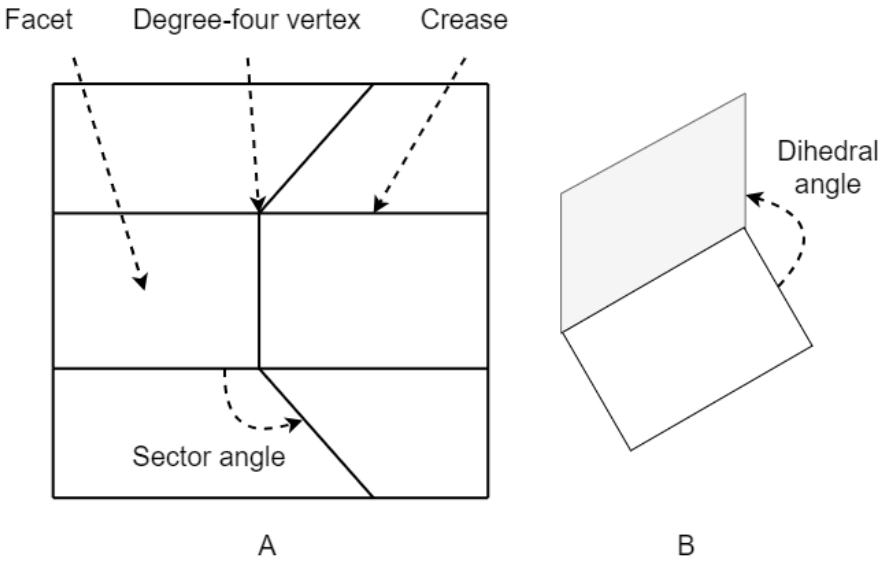

Figure 2.2: Visual explanation of the following concepts: facet, crease, (degree of a) vertex, sector angle (A) and dihedral angle (B).

2.2.2. Links Between Origami and Compliant Mechanisms

As mentioned in the introduction of this literature review, kinematic origami mechanisms are compliant mechanisms. The most important requirement for an origami model to be a compliant mechanism is the presence of hinge creases. As the name suggests, these creases acts as hinges during operation of the mechanism [22].

Next to hinge creases, the other type of origami crease is the construction crease. This type of crease is only subjected to folding when the origami model is deployed, after which its mobility is 'sacrificed'. The concept of sacrificial hinges in origami is introduced and researched in 2019 by Nelson *et al.* [23]. It should be noted that kinematic origami can include both construction and hinge creases or only hinge creases. An origami model that contains only construction creases is classified as static origami.

Origami mechanisms are highly inter-dependent compliant mechanisms. Every dihedral angle of a crease line can influence other dihedral angles of creases intersecting at the same vertex. If these secondary creases are in turn connected to other vertices, the effect of a single fold can propagate through an entire origami model.

A useful method to evaluate origami mechanisms as compliant mechanisms is the *Graph Theory*, initially presented by Marcus in the year 2020 [24]. A graph consists of points, called vertices, and lines between points, called edges. When applying this theory to origami, the vertices correspond with the (near-)rigid facets of an origami model, while the edges correspond with the compliant creases. It should be noted that the definition of a vertex in the Graph Theory does not correspond with the definition of an origami vertex. Graphs Theory allows for a better understanding of the interaction between motion and structure of origami, and it can help to predict complex motion and to develop mechanisms [22].

2.2.3. Foldability

Counterintuitively, not all origami models start as a flat sheet of material. The distinction can be made between *Euclidean* and *non-Euclidean* origami. Euclidean origami, also referred to as *developable* origami, is folded from a planar developable surface. This means that the sector angles of a Euclidean vertex always add up to 360 degrees. On the other hand, non-Euclidean origami is a subset of origami that is folded from non-planar developable surfaces [25]. The sector angles of a non-euclidean vertex add up to a total of more, or less than 360 degrees. Figure 2.3 illustrates the differences between Euclidean and non-Euclidean origami. Non-Euclidean origami is mainly used for its multi-stable behaviour, which will be further discussed in the next subsection.

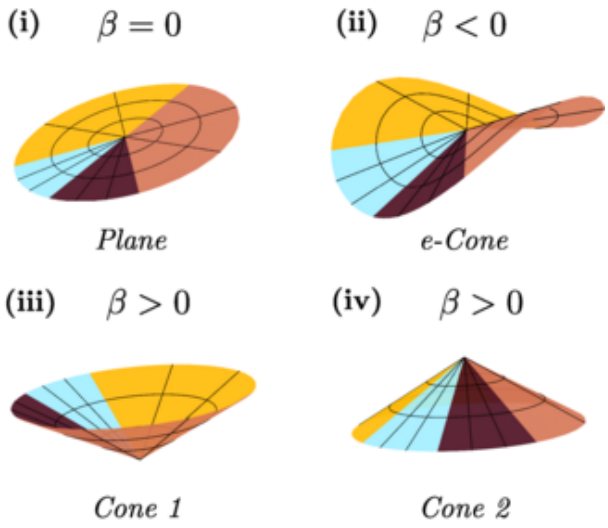

Figure 2.3: Visualisation of Euclidean and non-Euclidean origami, in which β represents the angular deficiency. (i) shows a Euclidean vertex (zero angular deficiency), while (ii), (iii) and (iv) show non-Euclidean vertices. The sum of the sector angles of (ii) is bigger than 360 degrees (negative angular deficiency), while the sum of the sector angles of (iii) and (iv) is smaller than 360 degrees (positive angular deficiency). This figure is adapted from a paper of Addis *et al.*[25].

An origami that can fold continuously without any deformation in its facets is called *rigid foldable* [26]. Global rigid foldability can only be the case when every vertex of the model is rigid foldable. Both non-Euclidean and Euclidean origami can be rigid foldable up to a certain dihedral angle, but non-Euclidean can never fold rigidly to a configuration where all dihedral angles are Pi radians (flat).

A special case of rigid foldability is *flat foldability*. This classification applies to origami models of which every crease can be folded over an angle of Pi radians. This entails starting from a flat configuration and consequently ending in a dihedral angle of 0 radians. Note that it is never possible to fold further than this, as at this point the facets coincide with each other, and they are not able to penetrate each other. Non-Euclidean origami is never able to fold flat in a rigid-foldable manner [25].

Zimmermann and Stanković [26] presented an article in which they derived a necessary and sufficient condition for the rigid foldability of a developable degree-four vertex. In later research, by Zimmermann *et al.*, the research was extended to degree-n vertices. In this later work Zimmerman *et al.* presented the *Principle of Three Units* (PTU) which provides an efficient approach to model the kinematics of degree-n vertices [27].

2.2.4. Multi-Stability in Origami Mechanisms

Origami mechanisms can be multi-stable. Stable states can be found by looking into the energy landscape, as these stable states are represented by (local) energy minima. This energy landscape also shows the energy barrier that needs to be overcome to transition between stable states [28].

As mentioned in the last subsection, non-Euclidean origami exhibits multi-stability naturally. This can be explained by the impossibility of flat foldability of non-Euclidean origami. This results in two disconnected configurations, each with the same dihedral angles but opposite handedness [25]. Waitukaitis *et al.* realized an origami inverter that physically demonstrates tri-stable behaviour [29].

Regular, Euclidean origami can also exhibit multi-stability. Waitukaitis *et al.* [30] show that a Euclidean, rigid, degree-four vertex can have up to six minima in its energy landscape, which causes it to behave hexa-stable.

When examining non-rigid origami, Silverberg *et al.* [31] show how the traditional square twist crease pattern, which has zero degrees of freedom, can be folded due to panel bending deformations. These hidden degrees of freedom result in bi-stability of the origami mechanism.

The unit cell of the most researched origami tessellation, the Miura-Ori tessellation, is naturally bi-stable. The unit cell is either in its original configuration, or it is 'popped'. These popped unit cells are called Pop Through Defects (PTDs). The PTDs can influence the stiffness characteristics of the Miura-Ori sheet. Also, the interaction between multiple PTDs has an effect on the stiffness characteristics. For example, a column of PTDs on alternating vertices behaves like a hinge while a column of PTDs on consecutive vertices generates a rigid corrugated structure [32].

2.3. Synthesis of Origami Linear Guides

Keywords: origami mechanism, kirigami mechanism, local support stiffness, stiffness singularity, tessellation, algorithmic/computational synthesis, reinforced mechanism, Inverse Design Strategy, orimimetrics, pseudo-rigid body model, linear guide, straight-line mechanism.

2.3.1. Classification and Design Possibilities of Compliant Linear Guides

The possibilities for designing compliant linear guides are infinite. This classification aims to structure some of the design possibilities of compliant linear guides and to offer inspiration for synthesising a design of an origami linear guide.

Spacial Placement with Respect to the DOF

The first and foremost classification of compliant linear guides is its spacial placement with respect to its degree of freedom. This classification can be divided in axial and transverse placement with respect to the DOF. Axial placement means that the EE of a compliant linear guide translates axially by expanding or contracting of the mechanism in this direction.

Transverse placement with respect to the DOF is the opposite of axial placement. In this case, the mechanism is placed in a plane perpendicular to the DOF. The EE of the mechanism moves along the DOF, while the base of the mechanism remains in the original plane of placement. The (double) parallelogram joint is an example of a transversely placed compliant linear guide.

Making Use of Multi-Stability

Multi-stability can be used as a tool to develop compliant linear guides with high constraint stiffness. One way to harness multi-stability, is by using it in a discrete configuration mechanism. Multi-stability could be used to create a mechanism with just a few stable configurations. After this is done, it is possible to add a second mechanism with high local constraint stiffness at the locations of the stable configurations. Rommers and Herder developed a method with which high local support stiffness could be achieved [4].

Another way of harnessing multi-stability for high support stiffness of a compliant linear guide is to use multi-stability to alter stiffness characteristics of a mechanism. An example of a mechanism that can change its stiffness characteristics due to its multi-stability is the Miura-Ori tessellation presented by Silverberg *et al.* [32].

Making use of Stiffness Singularities

A stiffness singularities is a local increase in stiffness at a specific position. When relating this to constraint stiffness, a stiffness singularity can be useful, but also deceiving. It can be deceiving as a designer of a compliant mechanism might report the constraint stiffness value at the stiffness singularity, while the constraint stiffness drops significantly at other places in its range of motion. This problem could be observed in the example of the (double) parallelogram joint [2].

However, stiffness singularities can also be used and combined to gain a high constraint stiffness over a longer range of motion. This method was used by Rommers and Herder when designing a compliant mechanism with initially curved folded leaf springs [4].

Making use of Origami Pitch Hinges for Improved Stiffness Characteristics

Origami enables to design origami hinges with coupled secondary deformations. These secondary deformations could help to maintain or increase the constraint stiffness of the hinge over its range of motion. Some origami hinge examples are developed by Nelson *et al.* in their research concerning sacrificial joints [23]. Out of their developed hinges, the pitch hinges are deemed most interesting due to their simplicity.

Schematic Overview of Design Possibilities of Compliant Linear Guides

Summarising the classification and design possibilities of compliant linear guides, a schematic overview can be found in Figure 2.4. Note that this schematic just gives an indication of the possibilities, and therefore does not include every possible solution. The schematic will be discussed in more detail in chapter 3, Discussion.

2.3.2. Existing Origami Tessellation-Based Synthesis

Having established an overview of design possibilities, it will now be explored how to move from a conceptual design of an origami linear guide to a detailed design.

The first method to synthesise origami mechanisms is by using existing origami or kirigami tessellations. Kirigami is a practice related to origami. When practicing kirigami, the practitioner is allowed to use cuts in addition to folds. In this literature review, the tessellations are categorized in three groups, cylindrical origami, planar origami and kirigami. The categories cylindrical and planar origami represent their approximate spacial shape in their folded state.

The tessellations in the first subset of origami are approximately cylindrical in their folded state. The reviewed origami tessellations are listed below, in combination with the sources where each tessellation was mentioned.

- Miura Ori Tube [33]
- Miura Cylinder [15, 28]
- Tachi–Miura Polyhedron Cylinder [10]
- Waterbomb Cylinder [28, 34]
- Triangulated Cylinder [35]
- Kresling Cylinder [10, 15, 28]

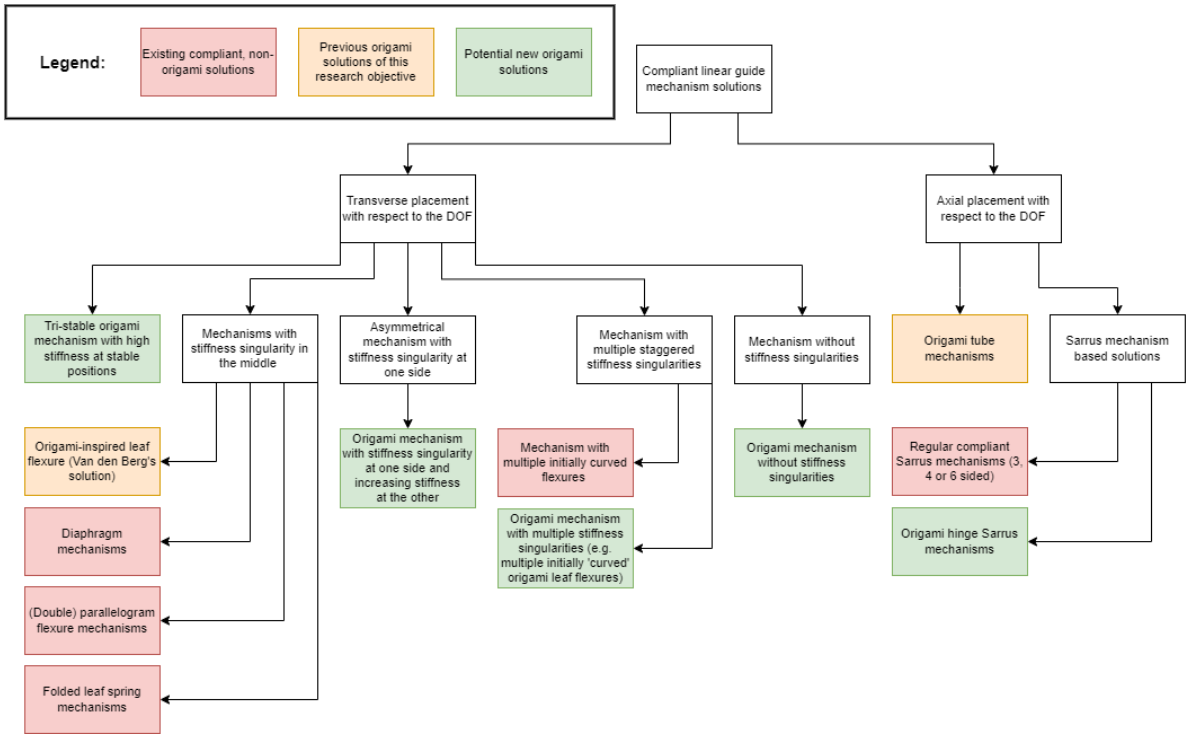

Figure 2.4: Schematic overview of the design possibilities of compliant linear guides.

- Accordion Cylinder [10, 15]
- Yoshimura Cylinder [15, 28]

The tessellations in the second category of origami are approximately planar in their folded state.

- Miura Ori sheet [32, 34, 36, 37]
- Square-twist pattern [28, 31]

The last subset of reviewed tessellations is kirigami tessellations. A list of found kirigami tessellations with their corresponding sources can be found below.

- Miura kirigami [38]
- Cubic kirigami [38]
- Parallel cuts kirigami [34]
- Fractal cut kirigami [34]
- Square array cut kirigami [38]

All of the tessellations mentioned before can be used as a starting point to synthesise origami mechanisms. The tessellations can be optimised, adapted or combined to yield optimal properties for the intended mechanism.

2.3.3. Algorithmic Computational Synthesis

Another method for synthesising origami designs is by generating them computationally. This has been done by Zimmermann *et al.* [39], after which Walker and Stankovic continued the work [40]. In both of these papers, however, the research is strictly focused on kinematics. This is not ideal for synthesising a linear guide, as maximising the constraint stiffness of the linear guide is one of the prime objectives, which is not considered in these algorithmic syntheses. An example of an origami mechanism synthesised by algorithmic computation is the origami-based Constant-height Walking system made by Sluijter [18].

One strategy that could be applied, is to first generate an origami mechanism computationally, after which the mechanism is reinforced to gain better constraint stiffness characteristics. Rommers *et al.* [3] developed a torsion-reinforced compliant linear guide, which could offer inspiration for developing reinforced structures. Van Manen used another method to reinforce compliant mechanisms, the *Inverse Design Strategy* (IDS). The strategy uses known mechanisms with very low stiffness in the desired direction, and 'inverses' them to get a mechanism with a very high stiffness in the desired direction. This method may not always yield good results, but it does help the designer to potentially get new insights [41].

2.3.4. Synthesis by Applying Orimimetrics to Existing Mechanisms

The final method for synthesising origami mechanism designs is applying orimimetrics to existing mechanisms. Orimimetrics is the practice to use the concept of folding to solve problems. The seed mechanisms, to which the concept of orimimetrics is applied, can both be traditional rigid-body mechanisms or compliant mechanisms. The method works as follows: a seed mechanism with desired kinematic properties is found, after which its rotational joints are replaced by origami hinges. In the case of applying orimimetrics to a compliant mechanism, a Pseudo-Rigid Body Model (PRBM) should first be made of the mechanism, in order to replace individual rotational joints with origami hinges [14].

The inspiration for the origami hinges is provided by Nelson *et al.* in their research paper 'origami-inspired sacrificial joints for folding compliant mechanisms' [23]. From this paper, the degree-four and degree-five pitch joints are selected due to their simplicity. A third type of pitch joint is added, which can be called a degree-six pitch joint. The degree-four,-five and -six pitch joints have one, two and three degrees of freedom respectively.

An example of applying orimimetrics to a compliant mechanism is provided by Van den Berg [42]. In his research, he used degree-four origami pitch joints to replace the two rotational joints in the PRBM of a single leaf flexure with a fixed-guided boundary condition.

In the case of synthesising origami linear guides, it is possible to use straight-line mechanisms as seed mechanisms. However, in many cases the end-effector of a straight line mechanism rotates during its translational motion. This is not desired, and therefore this issue needs to be resolved. This can, for example, be done by adding an extra rotational joint at the end-effector and mirroring the straight line mechanism to the other side of the end-effector. This possibility is illustrated in Figure 2.5.

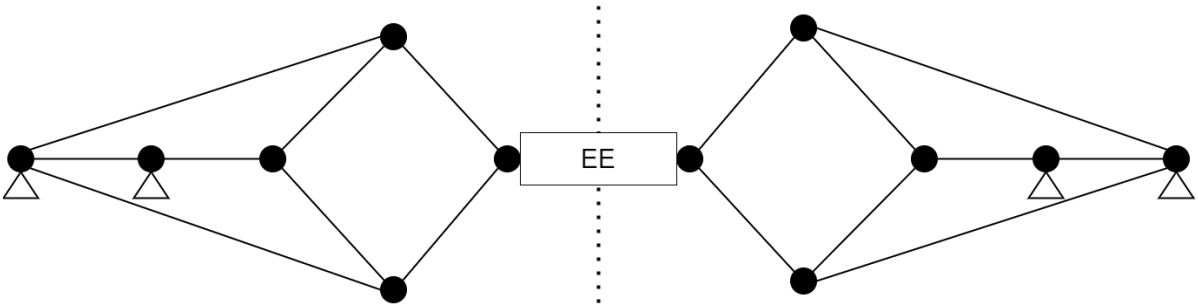

Figure 2.5: Linear guide mechanism, constructed from two Peaucellier-Lipkin inversor straight-line mechanisms

2.4. Origami Simulations

Keywords: origami simulation/modelling, kinematics, mechanics, finite element analysis, bar and hinge model, shell elements, parametric study.

2.4.1. General Overview of Origami Simulations

After synthesising an origami linear guide, it should be evaluated in an origami simulation. This is needed to be able to assess a design, and possibly to optimise it.

Origami simulations can be subdivided in kinematics-, mechanics- and multi-physics-based simulations. In this classification, the former does not offer enough detail and the latter is too extensive, so in this section the focus will be on mechanics-based simulations [43].

There are two main methods to mechanically simulate origami designs, namely using bar and hinge models or Finite Element Analysis (FEA) solvers.

As the name suggests, bar and hinge models consist of two main elements, bars and (spherical) hinges. The bars of the model can only deform axially and represent the in-plane stiffness of origami panels, such as stretching and shearing. The hinges connect the bars to each other and are accompanied by rotational springs. The hinges are used to capture out-of-plane behaviour, such as crease bending and panel bending [44].

FEA solvers are a more general tool for mechanics-based simulations. Any structure or mechanism can be meshed into finite elements and can subsequently be analysed. For origami models, it is generally possible to use the computationally more favorable *shell elements* for the panels and creases instead of *solid elements* [43].

2.4.2. Bar and Hinge Models

In this review, MERLIN, MERLIN2 and SWOMPS were evaluated as viable bar and hinge models. All three of these are related to each other, with the first developed package being MERLIN and the newest package being SWOMPS.

It is clear that MERLIN was the first developed package. The software offers the least customisation possibilities, making it difficult to model complex origami behaviour effectively. Although this software package is limited by itself, it did lay the foundation for later packages to excel [45].

Logically, MERLIN2 followed MERLIN as its successor. There are two most important changes between these software packages. The first one being the possibility to use the generalized N5B8 panel model on top of just N4B5. This new panel model can better capture bending of panels and allows for consideration of polygonal panels. The other extension of the software is the possibility to use displacement loading next to force loading. This allows for more flexibility in simulations [46, 47].

The final considered bar and hinge model is SWOMPS. The SWOMPS software package made several big improvements to the simulation possibilities. The first notable extension is panel contact, which can be simulated by SWOMPS. Another new possibility in this software package is the consideration of compliant creases. Compliant creases allow for more realistic crease line modelling, as this extension enables the user to specify the width of a crease. Thirdly, SWOMPS supports residual stress specifications in a design. This could be very helpful when modelling a design in its neutral, but already assembled state. Lastly, SWOMPS support sequential loading, which is crucial for multi-stage simulations and could be very helpful for determining stiffness characteristics [48, 49].

SWOMPS also offers multi-physics simulations, but this functionality is likely not needed for the analysis of origami linear guides.

2.4.3. Commercial FEA Software Packages

There are countless commercial FEA software packages available to use. In this section, two popular options are evaluated, ANSYS Mechanical APDL and COMSOL multiphysics. ANSYS Mechanical APDL offers users a versatile programming language to create parametric models for systemic analyses. COMSOL multiphysics is a more intuitive FEA software package, but lacks the parametric capabilities of ANSYS Mechanical APDL [50].

2.4.4. Comparison of the Methods

Using bar and hinge models for origami analyses has some drawbacks and limits. One of these limits is its incapability to deform in every arbitrary way, like a real system does. The degrees of freedom of the individual bars and hinges are a simplification of the degrees of freedom of actual origami creases and panels. Due to this, it is for example not possible to capture all possible panel bending modes.

Another drawback of bar and hinge models is its inability to capture localized behaviours such as crease buckling, panel buckling, stress concentrations, and local material plasticity in the origami mechanisms. All of these can be captured by FEA [43].

The last drawback of bar and hinge models is its lack of options in the current analysis software. It is for example not possible to constrain rotations of nodes, and forces and displacements can only be

located at a few discrete points.

FEA solvers also have some drawbacks compared to bar and hinge models. Generally, FEA solvers require longer computation time and require a more extensive model. Due to this reason, global behaviour of simple origami mechanisms can be studied easier with bar and hinge models.

Lastly, FEA solvers are made more general, so they do not offer all data that is specifically relevant for origami models. An example of this is SWOMPS's possibility to plot the energy associated to the specific deformation modes of origami, being: crease bending, crease stretching, panel bending and panel stretching. This information can give crucial insights in the kinetics of an origami mechanism, and could aid in improving a design.

2.5. Fabrication and Testing of Origami Linear Guides

Keywords: origami mechanism, fabrication/manufacturing, emergent lamina mechanism/joint, groove joint, 3D printing, testing/experiment.

2.5.1. Fabrication Options

Origami-inspired mechanisms can be fabricated in multiple different ways. In this subsection, three fabrication methods are elaborated.

The first way of fabricating origami mechanisms is to only use 2D manufacturing methods. This is done by using Emergent Lamina Joints (ELJs) for the origami crease lines. These joints consist of a pattern of flexure elements and holes, and should be flexible when folding, while being as stiff as possible when loaded in other ways. The big advantage of this fabrication method is the simplicity of 2D manufacturing methods. Disadvantages of this fabrication method are the lower constraint stiffness and limited range of motion of ELJs [31, 42, 51].

Another popular fabrication method is 'sandwiching' a flexible sheet of material in between thicker parts. These thicker parts will then serve as the panels of the origami model, while the flexible sheets between the thicker parts function as crease lines. In this way, groove joints are created instead of emergent lamina joints. The advantage of this fabrication method is the better performance of groove joints compared to emergent lamina joints. The main disadvantage of this method is the need for assembly, which makes it more prone to fabrication errors. This is especially concerning when there are over-constraints in play.

Lastly, 3D printing can also be considered as a fabrication method. Mak *et al.* used this technique to print an origami-inspired actuator [15]. The main advantage of 3D printing origami mechanisms is the possibility to print the model in its folded configuration. Normally, origami mechanisms are produced in a flat configuration, after which they are folded to the shape of the mechanism. Due to this, the mechanism is already pre-stressed in the neutral position. This issue can be prevented by 3D printing. 3D printing origami also has a number of disadvantages, such as anisotropic material behaviour and printing limitations.

2.5.2. Experimental Validation

After fabricating an origami model, it is often desired to experimentally validate the carried out simulations. It is important to know that origami hinges, being compliant mechanisms, have shifting centers of rotation. It is therefore important to avoid over-constraints during testing, as these could lead to incorrect stiffness observations. The over-constraints in the test setup make the mechanism stiffer than it is in reality. Grey *et al.* propose a test setup for testing the fold stiffness of a single origami hinge in their research, without over-constraining it [52]. A drawback of their proposed setup is it's inability to analyse origami hinges around their flat configuration.

3

Discussion

A broad review of literature, relevant for designing an origami-inspired linear guide, is executed in this paper. This review serves as a stepping-stone to design, analyse, manufacture and test an origami linear guide that should rival or surpass current state of the art compliant linear guides. To this regard, subsection 2.3.1, Classification and Design Possibilities of Compliant Linear Guides, offers insight in the core considerations for conceptually designing an origami linear guide. Figure 2.4, on page 8, illustrates the key design choices for the conceptual design. In this figure, the red blocks represent the current state of the art compliant linear guides. These mechanisms are not origami-inspired. The orange blocks contain origami solutions that were previously found as a result of the same research objective as in this paper. The green blocks consist of potential new origami linear guide solutions, and can be considered as knowledge gaps.

To consider the knowledge gaps and to choose a conceptual design to research, the green blocks are evaluated in detail. The green blocks are elaborated from left to right.

The leftmost green block is the only solution in this figure that makes use of multi-stability. As this concept is only stable in three positions in its range of motion, the applicability of this solution is limited compared to the rest of the solutions. One potential use for this concept is a pick-and-place machine, using the three stable positions for picking, moving and placing.

The second green block from the left consists of a concept which makes use of one stiffness singularity. This stiffness singularity is placed asymmetrically, which makes it different from the origami-inspired leaf flexure developed by Van den Berg [42]. The side opposite from the stiffness singularity should gain constraint stiffness in another way, for example by using coupled origami deformations that increase the surface moment of inertia of the design.

The origami linear guide concept in the middle is very similar to mechanisms consisting of multiple initially curved flexures. The difference is that these initially curved flexures would be replaced by initially curved origami leaf flexures. This concept would, however, be prone to over-constraints.

The second to last concept is very general, being an origami linear guide without stiffness singularities. The drawback of this concept is that it misses the constraint stiffness benefits that stiffness singularities provide.

The final and rightmost green block contains origami hinge Sarrus mechanisms. This is the only concept which is axially placed with respect to the DOF. Symmetric axial mechanisms are insusceptible to parasitic motions, which is a big advantage. Several physical models have already been made for this concept, and the initial signs are promising. Due to this reason, this concept is chosen to be investigated further in a following research. This study will explore whether origami hinge Sarrus mechanisms can rival or even excel current state of the art compliant linear guides. The plan for this research will be discussed in Chapter 4 and, in a more detailed manner, in Appendix A.

4

Research Plan

As mentioned at the end of chapter 3, Discussion, it was chosen to research origami-inspired Sarrus mechanisms.

In essence, the research will consist of three parts:

1. Make a decision on key practical considerations. These decisions include: the prototype's material, the fabrication method and the simulation method.
2. Perform an analysis of the building blocks of the several Sarrus mechanisms: the line hinge and the origami hinges (degree-four, -five and -six). Experimentally validate the simulation results. This step serves as proof of concept for the actual research performed in the step 3.
3. Perform an analysis of a normal compliant Sarrus mechanism and several origami-inspired Sarrus mechanisms. Experimentally validate the simulation results.

Paper models of three origami-inspired Sarrus mechanisms can be seen in Figure 4.1. The three mechanisms differ due to the varying configurations of their origami pitch hinges.

A more detailed, step-by-step research plan can be found in Appendix A. The corresponding research planning can be found in Appendix B.

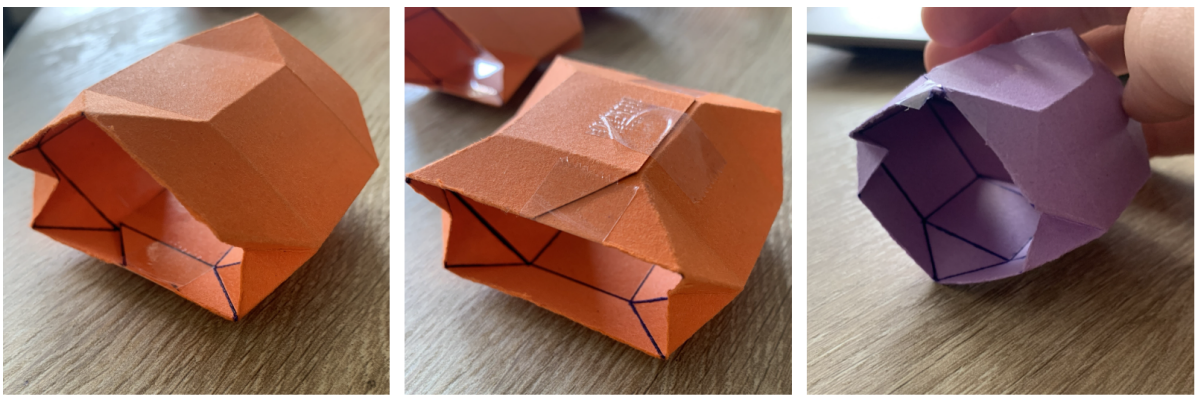

Figure 4.1: Paper models of origami-inspired Sarrus mechanisms. (A) point-in-point-out (B) points-in (C) points-out

5

Conclusion

The goal of this literature review was to evaluate literature relevant for the design of an origami-inspired linear guide. The paper discusses all necessary subjects to start designing, analysing, manufacturing and testing origami-inspired linear guides. Next to this, the current state of the art of compliant linear guides is also elaborated for comparing purposes.

In this paper's discussion, five separate origami linear guide concepts were identified and evaluated. Out of these five concepts, one is selected to be researched further. However, the other four concepts can just as well be used as a starting point for further research.

Further research could also focus on a specific subject discussed in this literature review, rather than focusing on designing an origami linear guide. This could for example be the development of a more advanced bar-and-hinge simulation software, targeting the drawbacks described in this paper. Another example of a possible future research subject would be the creation of a novel fabrication method for origami mechanisms. It can also be concluded that this literature review lays the groundwork for future research into a wide variety of subjects related to origami mechanisms.

References

[1] Shorya Awtar. *Synthesis and Analysis of Parallel Kinematic XY Flexure Mechanisms*. Tech. rep. 1998.

[2] Shorya Awtar, Alexander H. Slocum, and Edip Sevincer. “Characteristics of beam-based flexure modules”. In: *Journal of Mechanical Design* 129.6 (2007), pp. 625–639. ISSN: 10500472. DOI: 10.1115/1.2717231.

[3] J. Rommers et al. “A Flexure-Based Linear Guide With Torsion Reinforcement Structures”. In: *Journal of Mechanisms and Robotics* 14.3 (June 2022). ISSN: 19424310. DOI: 10.1115/1.4052971.

[4] J. Rommers and J. L. Herder. “Design of a Folded Leaf Spring with high support stiffness at large displacements using the Inverse Finite Element Method”. In: *Mechanisms and Machine Science*. Vol. 73. Springer Science and Business Media B.V., 2019, pp. 2109–2118. DOI: 10.1007/978-3-030-20131-9{_}209.

[5] Shorya Awtar and Alexander H Slocum. *Flexure Systems based on a Symmetric Diaphragm Flexure*. Tech. rep. 2005.

[6] Mario Di Giovanni. *Flat and Corrugated Diaphragm Design Handbook*. 1982.

[7] F W F Colin. *Variable thickness and initially curved flexures for improved flexure mechanisms*. Tech. rep. 2023. URL: http://repository.tudelft.nl/.

[8] N K Meinders. *Compensating parasitic motions and cross-couplings in compliant mechanisms*. Tech. rep. 2021. URL: http://repository.tudelft.nl/..

[9] Yi Zhu and Evgueni T. Filipov. “Large-scale modular and uniformly thick origami-inspired adaptable and load-carrying structures”. In: *Nature Communications* 15.1 (Dec. 2024). ISSN: 20411723. DOI: 10.1038/s41467-024-46667-0.

[10] Jessica Morgan, Spencer P. Magleby, and Larry L. Howell. “An approach to designing origami-adapted aerospace mechanisms”. In: *Journal of Mechanical Design* 138.5 (May 2016). ISSN: 10500472. DOI: 10.1115/1.4032973.

[11] Collin Ynchausti et al. “Hexagonal Twist Origami Pattern for Deployable Space Arrays”. In: *ASME Open Journal of Engineering* 1 (Jan. 2022). DOI: 10.1115/1.4055357.

[12] David Melancon et al. “Multistable inflatable origami structures at the metre scale”. In: *Nature* 592.7855 (Apr. 2021), pp. 545–550. ISSN: 14764687. DOI: 10.1038/s41586-021-03407-4.

[13] K Seymour et al. *Origami-Based Deployable Ballistic Barrier*. Tech. rep. 2018, pp. 763–778. URL: https://scholarsarchive.byu.edu/facpub.

[14] Larry L. Howell, Spencer P. Magleby, and Brian M. Olsen, eds. *Handbook of Compliant Mechanisms*. Wiley, Feb. 2013. ISBN: 9781119953456. DOI: 10.1002/9781118516485.

[15] Yoeko X. Mak, Alexander Dijkshoorn, and Momen Abayazid. “Design Methodology for a 3D Printable Multi□Degree of Freedom Soft Actuator Using Geometric Origami Patterns”. In: *Advanced Intelligent Systems* (May 2024). ISSN: 2640-4567. DOI: 10.1002/aisy.202300666. URL: https://onlinelibrary.wiley.com/doi/10.1002/aisy.202300666.

[16] Brandon Sargent et al. “An Origami-Based Medical Support System to Mitigate Flexible Shaft Buckling”. In: *Journal of Mechanisms and Robotics* 12.4 (Aug. 2020). ISSN: 19424310. DOI: 10.1115/1.4045846.

[17] Marco Salerno et al. “A Novel 4-DOF Origami Grasper With an SMA-Actuation System for Minimally Invasive Surgery”. In: *IEEE Transactions on Robotics* 32.3 (June 2016), pp. 484–498. ISSN: 15523098. DOI: 10.1109/TRO.2016.2539373.

[18] Jim Sluijter. *The Development and Geometric Analysis of an Origami-based Constant-height Walking Locomotion System*. Tech. rep. 2023.

[19] S. Felton et al. “A method for building self-folding machines”. In: *Science* 345.6197 (Aug. 2014), pp. 644–646. ISSN: 10959203. DOI: 10.1126/science.1252610.

[20] Guimin Chen et al. “Modeling large deflections of initially curved beams in compliant mechanisms using chained beam constraint model”. In: *Journal of Mechanisms and Robotics* 11.1 (Feb. 2019). ISSN: 19424310. DOI: 10.1115/1.4041585.

[21] Steven E Boer et al. “Modelling and Optimization of a Curved Hinge Flexure”. In: 2010.

[22] H. C. Greenberg et al. “Identifying links between origami and compliant mechanisms”. In: *Mechanical Sciences* 2.2 (2011), pp. 217–225. ISSN: 2191916X. DOI: 10.5194/ms-2-217-2011.

[23] Todd G. Nelson et al. “Origami-inspired sacrificial joints for folding compliant mechanisms”. In: *Mechanism and Machine Theory* 140 (Oct. 2019), pp. 194–210. ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2019.05.023.

[24] Daniel A Marcus. *Graph theory*. Vol. 53. American Mathematical Soc., 2020.

[25] Clark C. Addis, Salvador Rojas, and Andres F. Arrieta. “Connecting the branches of multistable non-Euclidean origami by crease stretching”. In: *Physical Review E* 108.5 (Nov. 2023). ISSN: 24700053. DOI: 10.1103/PhysRevE.108.055001.

[26] Luca Zimmermann and Tino Stanković. “Rigid and Flat Foldability of a Degree-Four Vertex in Origami”. In: *Journal of Mechanisms and Robotics* 12.1 (Feb. 2020). ISSN: 19424310. DOI: 10.1115/1.4044737.

[27] Luca Zimmermann, Kristina Shea, and Tino Stanković. “Conditions for Rigid and Flat Foldability of Degree-n Vertices in Origami”. In: *Journal of Mechanisms and Robotics* 12.1 (Feb. 2020). ISSN: 19424310. DOI: 10.1115/1.4045249.

[28] Lu Lu, Sophie Leanza, and Ruike Renee Zhao. “Origami With Rotational Symmetry: A Review on Their Mechanics and Design”. In: *Applied Mechanics Reviews* 75.5 (Sept. 2023). ISSN: 0003-6900. DOI: 10.1115/1.4056637.

[29] Scott Waitukaitis, Peter Dieleman, and Martin Van Hecke. *Non-Euclidean Origami*. Tech. rep.

[30] Scott Waitukaitis et al. “Origami multistability: From single vertices to metasheets”. In: *Physical Review Letters* 114.5 (Feb. 2015). ISSN: 10797114. DOI: 10.1103/PhysRevLett.114.055503.

[31] Jesse L. Silverberg et al. “Origami structures with a critical transition to bistability arising from hidden degrees of freedom”. In: *Nature Materials* 14.4 (2015), pp. 389–393. ISSN: 14764660. DOI: 10.1038/nmat4232.

[32] Jesse L Silverberg et al. *Using origami design principles to fold reprogrammable mechanical metamaterials*. Tech. rep. 2014. URL: https://www.science.org.

[33] Evgueni T. Filipov et al. “Origami tubes assembled into stiff, yet reconfigurable structures and metamaterials”. In: *Proceedings of the National Academy of Sciences of the United States of America* 112.40 (Oct. 2015), pp. 12321–12326. ISSN: 10916490. DOI: 10.1073/pnas.1509465112.

[34] Sebastien J.P. Callens and Amir A. Zadpoor. *From flat sheets to curved geometries: Origami and kirigami approaches*. Apr. 2018. DOI: 10.1016/j.mattod.2017.10.004.

[35] Zirui Zhai, Yong Wang, and Hanqing Jiang. “Origami-inspired, on-demand deployable and collapsible mechanical metamaterials with tunable stiffness”. In: *Proceedings of the National Academy of Sciences of the United States of America* 115.9 (Feb. 2018), pp. 2032–2037. ISSN: 10916490. DOI: 10.1073/pnas.1720171115.

[36] Mark Schenk and Simon D. Guest. “Geometry of Miura-folded metamaterials”. In: *Proceedings of the National Academy of Sciences of the United States of America* 110.9 (Feb. 2013), pp. 3276–3281. ISSN: 00278424. DOI: 10.1073/pnas.1217998110.

[37] Z. Y. Wei et al. “Geometric mechanics of periodic pleated origami”. In: *Physical Review Letters* 110.21 (May 2013). ISSN: 00319007. DOI: 10.1103/PhysRevLett.110.215501.

[38] Ahmad Rafsanjani and Katia Bertoldi. “Buckling-Induced Kirigami”. In: *Physical Review Letters* 118.8 (Feb. 2017). ISSN: 10797114. DOI: 10.1103/PhysRevLett.118.084301.

[39] Luca ; Zimmermann et al. “A Computational Design Synthesis Method for the Generation of Rigid Origami Crease Patterns”. In: *Journal of Mechanisms and Robotics* 14.3 (2021). DOI: 10.3929/ethz-b-000512541. URL: https://doi.org/10.3929/ethz-b-000512541.

[40] Andreas Walker and Tino Stankovic. “Algorithmic design of origami mechanisms and tessellations”. In: *Communications Materials* 3.1 (Dec. 2022). ISSN: 26624443. DOI: 10.1038/s43246-022-00227-5.

[41] R Van Manen. *Rotational stiffness in compliant mechanisms: theory, method and application*. Tech. rep. 2017.

[42] Y Van den Berg. “Design of an origami-inspired leaf flexure as an alternative to classical 2D flexures”. In: (2023).

[43] Yi Zhu, Mark Schenk, and Evgueni T. Filipov. “A Review on Origami Simulations: From Kinematics, to Mechanics, Toward Multiphysics”. In: *Applied Mechanics Reviews* 74.3 (May 2022). ISSN: 00036900. DOI: 10.1115/1.4055031.

[44] Yi Zhu and Evgueni T. Filipov. “A Bar and Hinge Model for Simulating Bistability in Origami Structures with Compliant Creases”. In: *Journal of Mechanisms and Robotics* 12.2 (Jan. 2020). ISSN: 19424310. DOI: 10.1115/1.4045955.

[45] Ke Liu and Glaucio H Paulino. *MERLIN: A MATLAB implementation to capture highly nonlinear behavior of non-rigid origami*. Tech. rep. 2016.

[46] Ke Liu and Glaucio H Paulino. *Highly efficient nonlinear structural analysis of origami assemblages using the MERLIN2 software*. Tech. rep.

[47] Sofie E. Leon et al. “On the effect of constraint parameters on the generalized displacement control method”. In: *Mechanics Research Communications* 56 (Mar. 2014), pp. 123–129. ISSN: 00936413. DOI: 10.1016/j.mechrescom.2013.12.009.

[48] Yi Zhu and Evgueni T. Filipov. “Sequentially working origami multi-physics simulator (SWOMPS): A versatile implementation”. In: *Proceedings of the ASME Design Engineering Technical Conference*. Vol. 8B-2021. American Society of Mechanical Engineers (ASME), 2021. ISBN: 9780791885451. DOI: 10.1115/DETC2021-68042.

[49] K. Liu and G. H. Paulino. “Nonlinear mechanics of non-rigid origami: An efficient computational approach”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2206 (Oct. 2017). ISSN: 14712946. DOI: 10.1098/rspa.2017.0348.

[50] Mary Kathryn Thompson and John Martin Thompson. *ANSYS mechanical APDL for finite element analysis*. Butterworth-Heinemann, 2017.

[51] Julian Keizer. *Design of a lamina emergent joint as an alternative for a groove joint*. Tech. rep. 2023.

[52] Steven W. Grey, Fabrizio Scarpa, and Mark Schenk. “Mechanics of paper-folded origami: A cautionary tale”. In: *Mechanics Research Communications* 107 (July 2020). ISSN: 00936413. DOI: 10.1016/j.mechrescom.2020.103540.

A

Detailed Research Plan

General first steps

1. Make a decision on material/fabrication method.
2. Make a decision on simulation method.

Hinge Analysis

1. Decide on the experimental setups for determining all stiffnesses, both in the DOF and all DOCs of the hinges.
2. Model line hinges and origami hinges (degree 4, 5, and 6). Keep the total size, crease width, and material thicknesses the same. Numerically determine all stiffnesses, both in the DOF and all DOCs, over the entire range of motion. Ensure that the hinge does not surpass the yield stress (or maybe even the fatigue stress) during its range of motion.
   (a) *Possibility: Do a parametric study or optimization for all three origami hinges by varying the origami angle and the width of the flaps.*
3. Experimentally validate the results, using an arbitrary value for the origami angle and side flaps, or using the optimized hinge models.

Sarrus Mechanism Analysis

1. Perform kinematic simulations in SolidWorks to evaluate current origami Sarrus mechanism designs and potentially find new designs. The mechanisms should have one DOF in theory.
2. Decide on the experimental setups for determining all stiffnesses, both in the DOF and all DOCs of the Sarrus mechanisms.
3. Model a normal compliant Sarrus mechanism in ANSYS APDL. Limit the size to the project description. Pick values for crease width and material thicknesses, ensuring that the mechanism's range of motion is sufficient according to the project description without surpassing the yield stress (or maybe even the fatigue stress). Numerically determine all stiffnesses, both in the DOF and all DOCs, over the entire range of motion.
   (a) *Possibility: Find out which normal compliant Sarrus mechanism is most suitable: 3-, 4-, or 6-sided Sarrus mechanism.*
4. Model the different origami-inspired Sarrus mechanisms in ANSYS APDL. Use the size from the project description and an optimal tessellation. Use the same values for crease width and material thicknesses as for the normal compliant Sarrus mechanism. Ensure that the mechanism's range of motion is sufficient according to the project description without surpassing the yield stress (or maybe even the fatigue stress). Numerically determine all stiffnesses, both in the DOF and all DOCs, over the entire range of motion.

5. For the next steps, there are two options for moving forward. This choice has not been made yet:
   (a) *Option 1: Try to find out which origami-inspired Sarrus mechanism has the best performance, using standard values for the origami angle and flap width for all of them.*
      i. *Possibility: Do a parametric study or optimization for the best origami-inspired Sarrus mechanism by varying the origami angle and the width of the flaps.*
   (b) *Option 2: Do a parametric study or optimization for all origami-inspired Sarrus mechanism concepts by varying the origami angle and the width of the flaps. Afterwards select the best origami-inspired Sarrus mechanism.*
6. Experimentally validate the results. Do this for both the normal compliant Sarrus mechanism and the best origami-inspired Sarrus mechanism. For the origami-inspired Sarrus mechanism, use an arbitrary value for the origami angle and side flaps, or use the optimized model.

B

Research Planning

	July 2024	August 2024	September 2024	October 2024	November 2024	December 2024		January 2025	February 2025 (half a month)
General first steps	Decide on material, fabrication method and simulation method	Holiday (entire month)					Holiday (half a month)		
Hinge Analysis	Decide on the experimental setups. Model and analyse line hinges and origami hinges.		Model and analyse line hinges and origami hinges. Do a parametric study or optimisation for all three origami hinges.	Experimentally validate the hinge analysis results.					
Sarrus mechanism analysis	Perform kinematic simulations to evaluate origami Sarrus mechanism designs.		Decide on the experimental setups.	Model and analyse a normal compliant Sarrus mechanism. Find out which Sarrus mechanism is most suitable (3,4 or 6 sided).	Model and anallyse the different origami inspired Sarrus mechanisms.	Find out which origami inspired Sarrus mechanism has the best performance. Do a parametric study or optimisation for the best origami inspired Sarrus mechanism.		Do a parametric study or optimisation for the best origami inspired Sarrus mechanism. Experimentally validate the Sarrus mechanism analysis results.	
Write research paper								Write research paper.	Write research paper. Hand-in the research paper on Friday, February 14th.

Figure B.1: Research planning. The yellow boxes represent optional steps that will be executed if time allows for it.