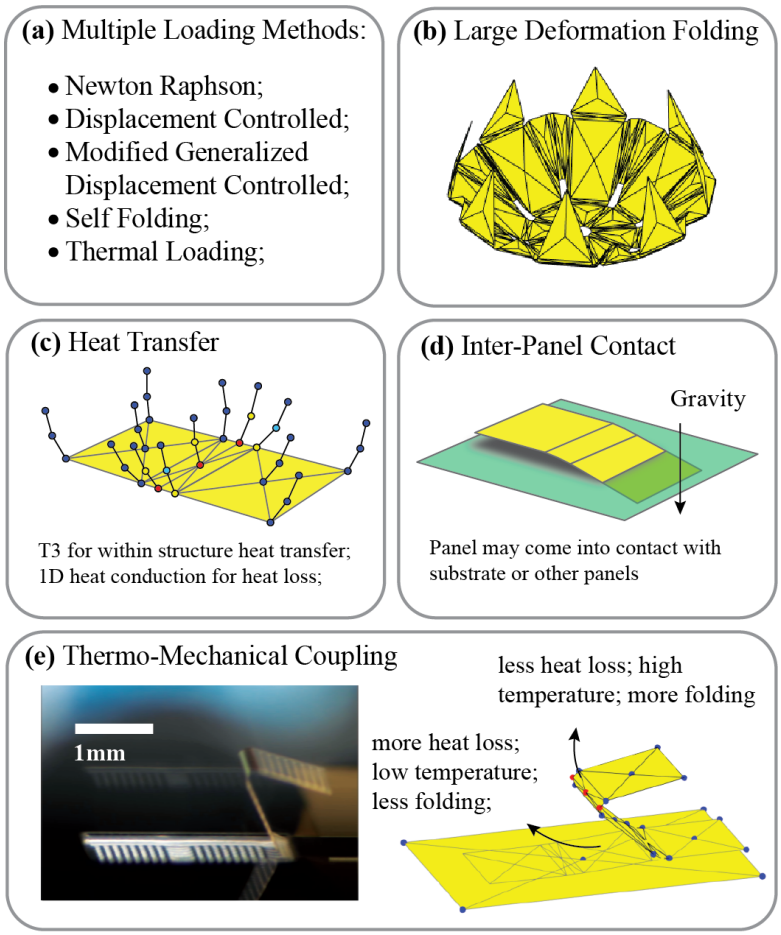
Tutorial on Using Sequentially Working Origami Multi-Physics Simulator (SWOMPS)

Authors: Yi Zhu, and Evgueni T. Filipov
Department of Civil and Environmental Engineering
University of Michigan at Ann Arbor

The SWOMPS program is a simulator built for solving active origami systems with complex multi-physics behaviors. The package can capture the large deformation folding, the heat transfer, the inter-panel contact, and the thermal-mechanically coupled actuation of active origami systems. The program provides five different loading methods and allows users to create loading schemes with arbitrary number and sequence of these methods. The users can also stop the solver at a specified step to switch to different methods and proceed the analysis.

In this document we will introduce how to sue the SWOMPS program to solve an origami structure with a sequence of different loading. The executing codes are provided in the boxes and explanatory figures are put next to the code boxes to provide guidance.



Package Access:

The package can be found on GitHub: https://github.com/zzhuyii/OrigamiSimulator or from our group web site at: https://drsl.engin.umich.edu/software/swomps-package/

Acknowledgement:

We would like to acknowledge the prior works from Ke Liu and Glaucio H. Paulino for establishing shared versions of nonrigid origami simulators. Their works paved the way for the new origami simulator, the origami contact, compliant crease, electro-thermal model presented in this package.

Reference:

[1] Y. Zhu, E. T. Filipov (2021). 'Sequentially Working Origami Multi- Physics Simulator (SWOMPS): A Versatile Implementation' (submitted)

[2] Y. Zhu, E. T. Filipov (2021). 'Rapid Multi-Physic Simulation for Electro-Thermal Origami Robotic Systems' (submitted)

[3] Y. Zhu, E. T. Filipov (2020). 'A Bar and Hinge Model for Simulating Bistability in Origami Structures with Compliant Creases' Journal of Mechanisms and Robotics, 021110-1.

[4] Y. Zhu, E. T. Filipov (2019). 'An Efficient Numerical Approach for Simulating Contact in Origami Assemblages.' Proc. R. Soc. A, 475: 20190366.

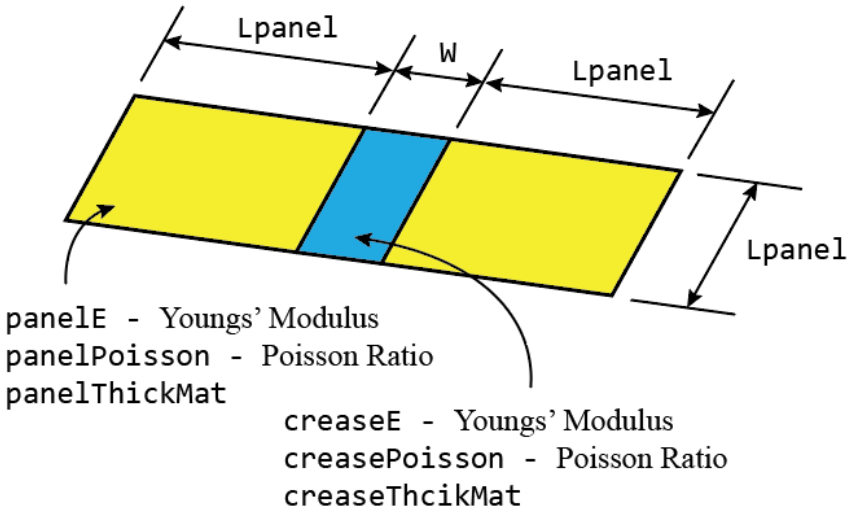
[5] Y. Zhu, E. T. Filipov (2019). 'Simulating compliant crease origami with a bar and hinge model.' IDETC/CIE 2019. 97119.

[6] K. Liu, G. H. Paulino (2018). 'Highly efficient nonlinear structural analysis of origami assemblages using the MERLIN2 software.' Origami^7.

[7] K. Liu, G. H. Paulino (2017). 'Nonlinear mechanics of non-rigid origami - An efficient computational approach.' Proc. R. Soc. A 473: 20170348.

[8] K. Liu, G. H. Paulino (2016). 'MERLIN: A MATLAB implementation to capture highly nonlinear behavior of non-rigid origami.' Proceedings of IASS Annual Symposium 2016.

Step 1
The example & Define some terms to be used latter

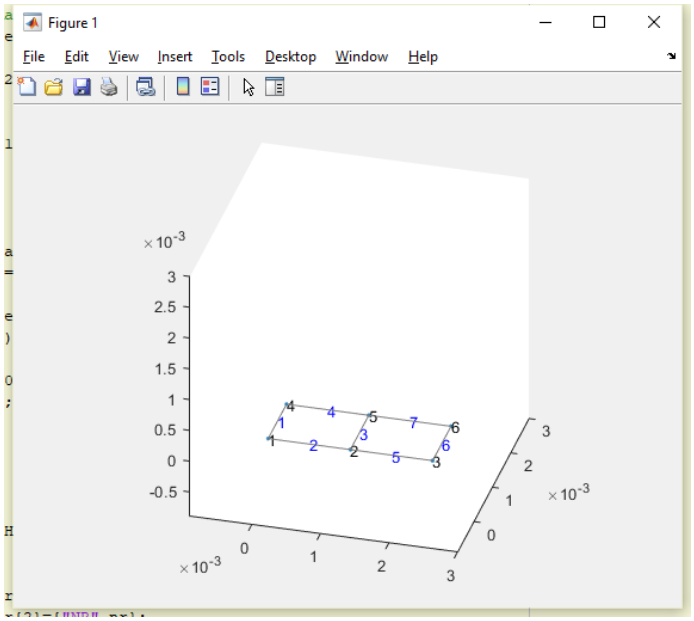



```
%% Define the Geometry of origami
% This section of code is used to generate the geometry of the origami
% pattern before meshing. First, we define the parameters that will be used
% in the modeling.

% thickness of two layers of material
t1=0.2*10^-6;
t2=0.80*10^-6;

% thickness of panel
tpanel=21*10^-6;

% density of panel
rho=1200;

% residual folding developed after fabrication (degree)
residualFold=10;

% width of crease
W=400*10^-6;

% length of panel
Lpanel=1*10^(-3);

% power input (mW)
qload=10;
```

Step 2
Define the input geometry

Here we first define the nodal coordinates and the panel information. The we analyze the input and plot the unmeshed geometry of the origami system.

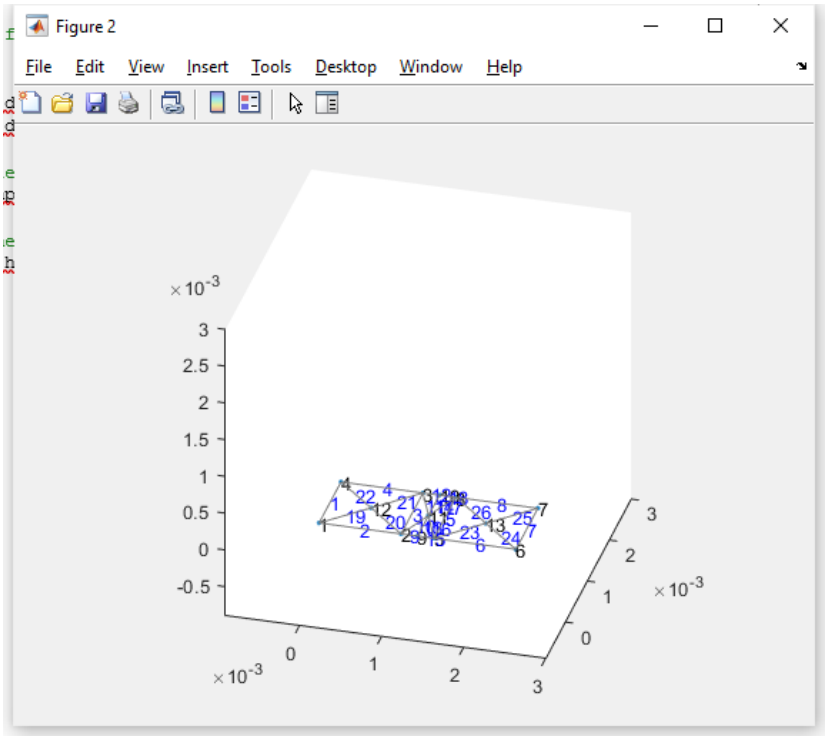
```
% create an instance of the solver class
ori=OrigamiSolver;

% define the Geometry
ori.node0=[0 0 0;
      Lpanel+W/2 0 0;
      2*Lpanel+W 0 0;
      0 Lpanel 0;
      Lpanel+W/2 Lpanel 0;
      2*Lpanel+W Lpanel 0;];

ori.panel0{1}=[1 2 5 4];
ori.panel0{2}=[2 3 6 5];


% Analyze the original pattern before proceeding to the next step
ori.Mesh_AnalyzeOriginalPattern();

% Plot the results for inspection
ori.displayRange=3*10^(-3); % plotting range
ori.displayRangeRatio=0.3; % plotting range in the negative axis

ori.Plot_UnmeshedOrigami(); % Plot the unmeshed origami for inspection;
```



Step 3
Generate the meshed geometry

After further providing the information about the width of the creases of the origami, we can generate the meshing of the origami system. We then plot the meshed origami for inspection.

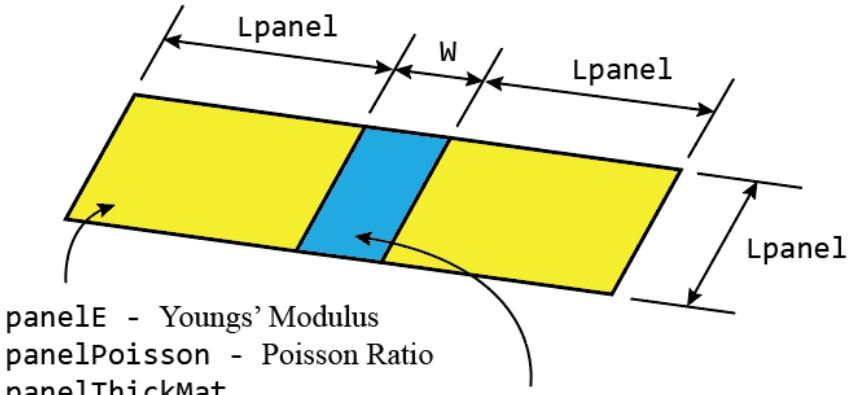
```
%% Meshing of the origami model

% Define the crease width
ori.creaseWidthVec=zeros(ori.oldCreaseNum,1);
ori.creaseWidthVec(3)=W;

% Compute the meshed geometry
ori.Mesh_Mesh()

% Plot the meshed origami for inspection;
ori.Plot_MeshedOrigami();
```

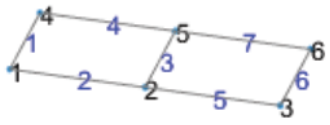


Step 4
Input the properties of the origami

Here, we first define the mechanical properties of the origami. The following figures shows how the variables are related to the origami system of interest.

```
%% Assign Mechanical Properties

ori.panelE=2*10^9;
ori.creaseE=2*10^9;
ori.panelPoisson=0.3;
ori.creasePoisson=0.3;
ori.panelW=W;
ori.diagonalRate=1000; % crease torsion stiffness/ bending stiffness

ori.panelThickVec=[tpanel;tpanel];
ori.creaseThickVec=zeros(ori.oldCreaseNum,1);
ori.creaseThickVec(3)=(t1+t2);
```

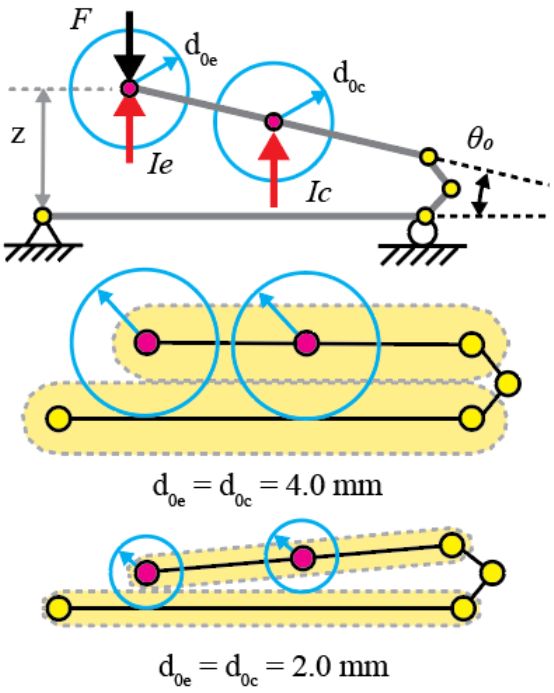


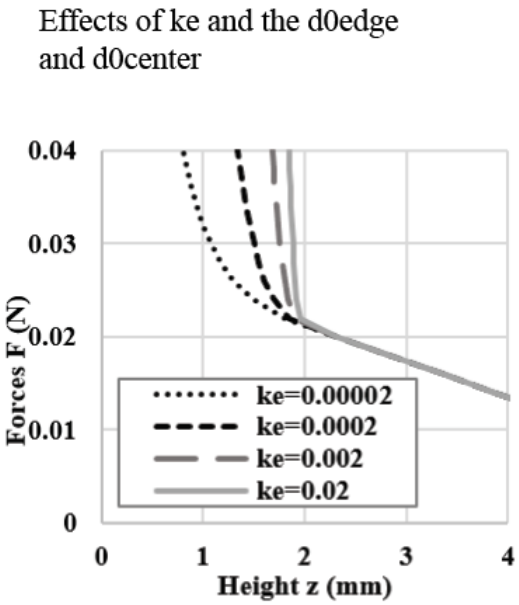


Use the generated numbering system for defining crease thickness matrix.

If we want to study the inter-panel contact of origami systems, we will need to open the contact by setting "`ori.contactOpen=1;`". The following figure shows how the three parameters will affects the behavior of this panel contact model.

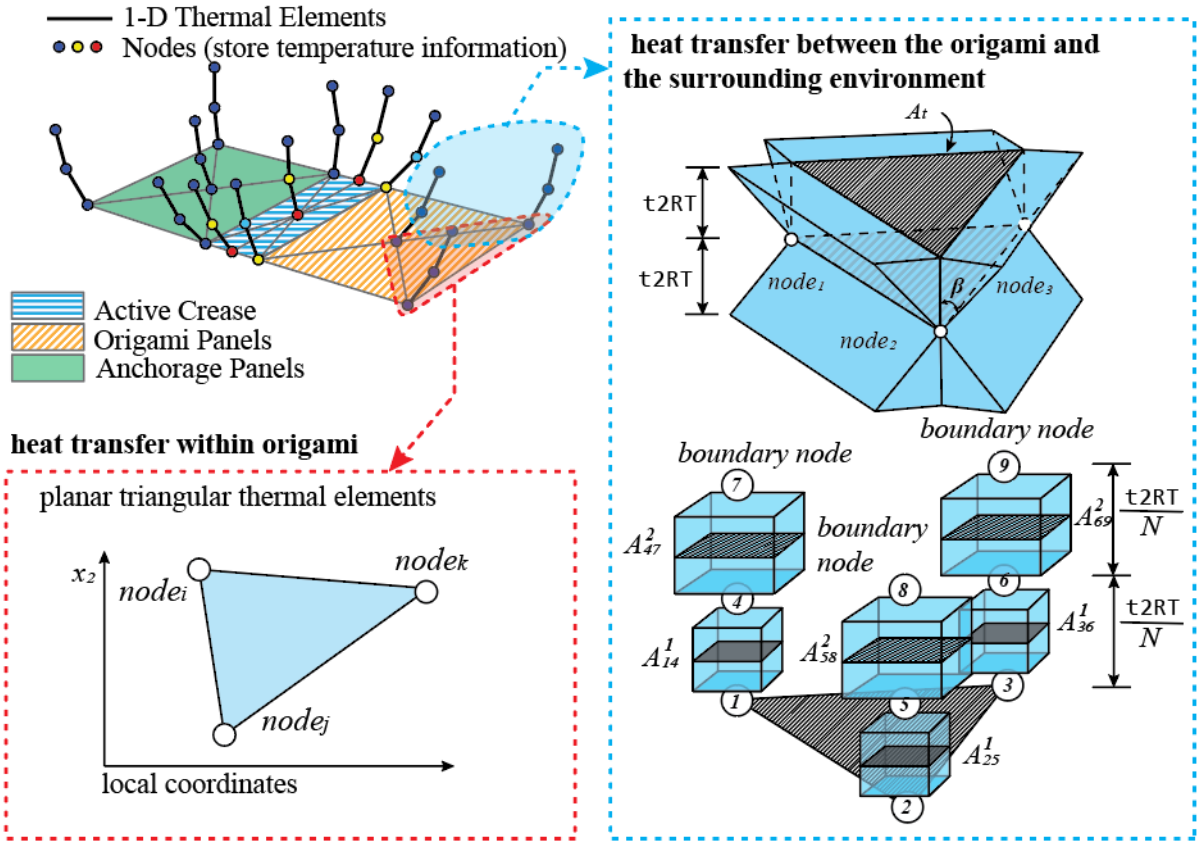
```
%% setup panel contact information

ori.contactOpen=1;
ori.ke=0.0001;
ori.d0edge=40*(10^(-6));
ori.d0center=40*(10^(-6));
```





Finally, we define the parameters for thermal properties. These include thermal conductivity of panels, creases, and the surrounding environments. We also need to specify how thick is the surrounding environment we want to consider.

```
%% Assign Thermal Properties

ori.panelThermalConductVec = [1.3;1.3];
ori.creaseThermalConduct=0.3;
ori.envThermalConduct=0.026;

% thickness of the submerged environment at RT
ori.t2RT=1500*10^(-6);
```

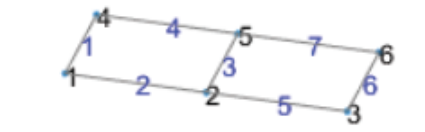


The model uses planar triangular thermal elements to capture the heat transfer within the origami structure and captures the heat transfer between the structure and the surrounding environments using a 1-D heat conduction model. The “t2RT” property is the thickness from the origami to the outer most layer of environment which is assumed to be at the “room temperature”.

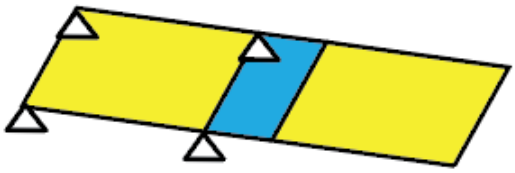
Step 5
Setup the loading step

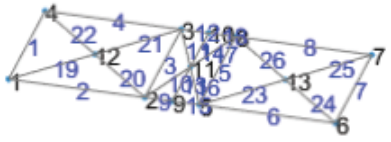
Here, we start setting up the loading steps. The first loading of this case is a self-stress induced folding. We assume that the system will develop 10 degrees of residual folding because of the residual stress generated during the fabrication. We use a total number of 80 incremental step to achieve the targe self-folding. We use the self folding solver to achieve this loading case.

```
%% Setup the loading controller

% applying the residual stress
selfFold=ControllerSelfFolding;

% Assign zero strain position for creases during self-folding
% 0-2pi, This matrix can be used to manually set the zero energy rotation
% angle of the crease hinge
selfFold.targetRotZeroStrain=pi*ones(ori.oldCreaseNum,1);

ratio=residualFold/180;
selfFold.targetRotZeroStrain(3)=pi+ratio*pi;

selfFold.supp=[1,0,0,1;
      2,0,0,1;
      3,0,1,1;
      4,1,1,1;];
selfFold.increStep=80;
selfFold.tol=4*10^-6;
selfFold.iterMax=25;
selfFold.videoOpen=0;
```



Use the generated numbering system of unmeshed origami to define crease self folding

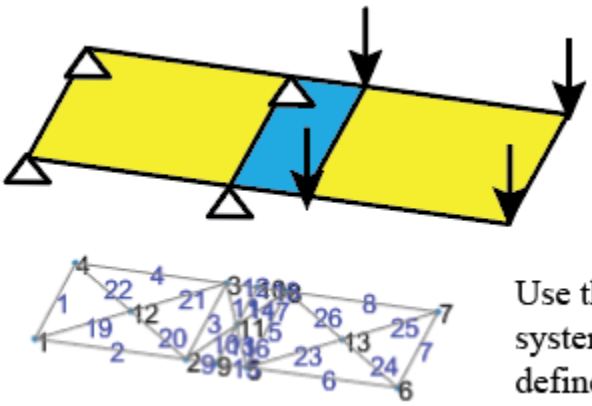




Use the generated numbering system of meshed origami to define support

The second loading of this case is the loading of gravity. We will calculate the self-weight using the geometry of the system and the density of the material. We use a total of 50 incremental step to achieve the target loading.

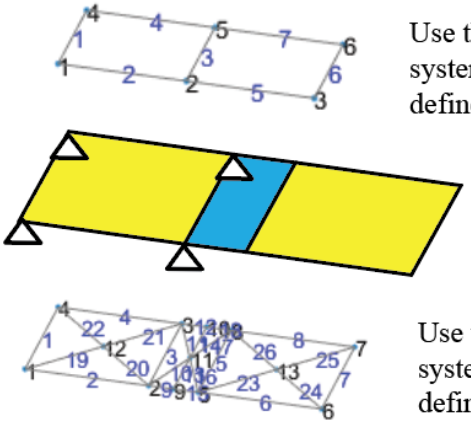
```
% applying the gravity loading
nr=ControllerNRLoading;

nr.increStep=50;
nr.tol=10^-6;
nr.iterMax=50;

nr.supp=[1,0,0,1;
      2,0,0,1;
      3,0,1,1;
      4,1,1,1;];

loadMag=0.9*(Lpanel*Lpanel*tpanel*rho*9.8)/6/nr.increStep;
nr.load=[5,0,0,-loadMag;
      6,0,0,-loadMag;
      7,0,0,-loadMag;
      8,0,0,-loadMag;
      13,0,0,-2*loadMag;];
nr.videoOpen=0;
```



Use the generated numbering system of meshed origami to define support and loading

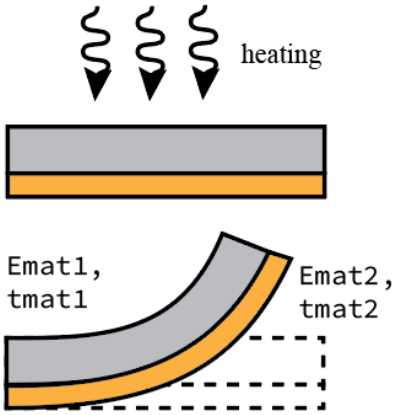
The final loading of this case is the thermal loading. We will input 10mW of power into the crease and calculate how much folding is generated. We use a total of 250 incremental step to achieve the target loading.

```
% applying the thermal loading
thermal=ControllerThermalLoading;
thermal.thermalStep=250;
thermal.tol=2*10^-6;
thermal.supp=[1,1,1,1;
      2,1,1,1;
      3,1,1,1;
      4,1,1,1;];

thermal.thermalBoundaryPanelVec=[];
thermal.roomTempNode=[];

thermal.deltaAlpha=zeros(ori.oldCreaseNum,1);
thermal.deltaAlpha(3)=(52-14)*10^(-6);

thermal.Emat1=39.5*10^9;
thermal.Emat2=2*10^9;
thermal.tmat1=t1;
thermal.tmat2=t2;
thermal.videoOpen=0;

% the target loading of crease heating
thermal.targetCreaseHeating=[3,qload/1000];
```



Use the generated numbering system of unmeshed origami to define crease heating

Use the generated numbering system of meshed origami to define support

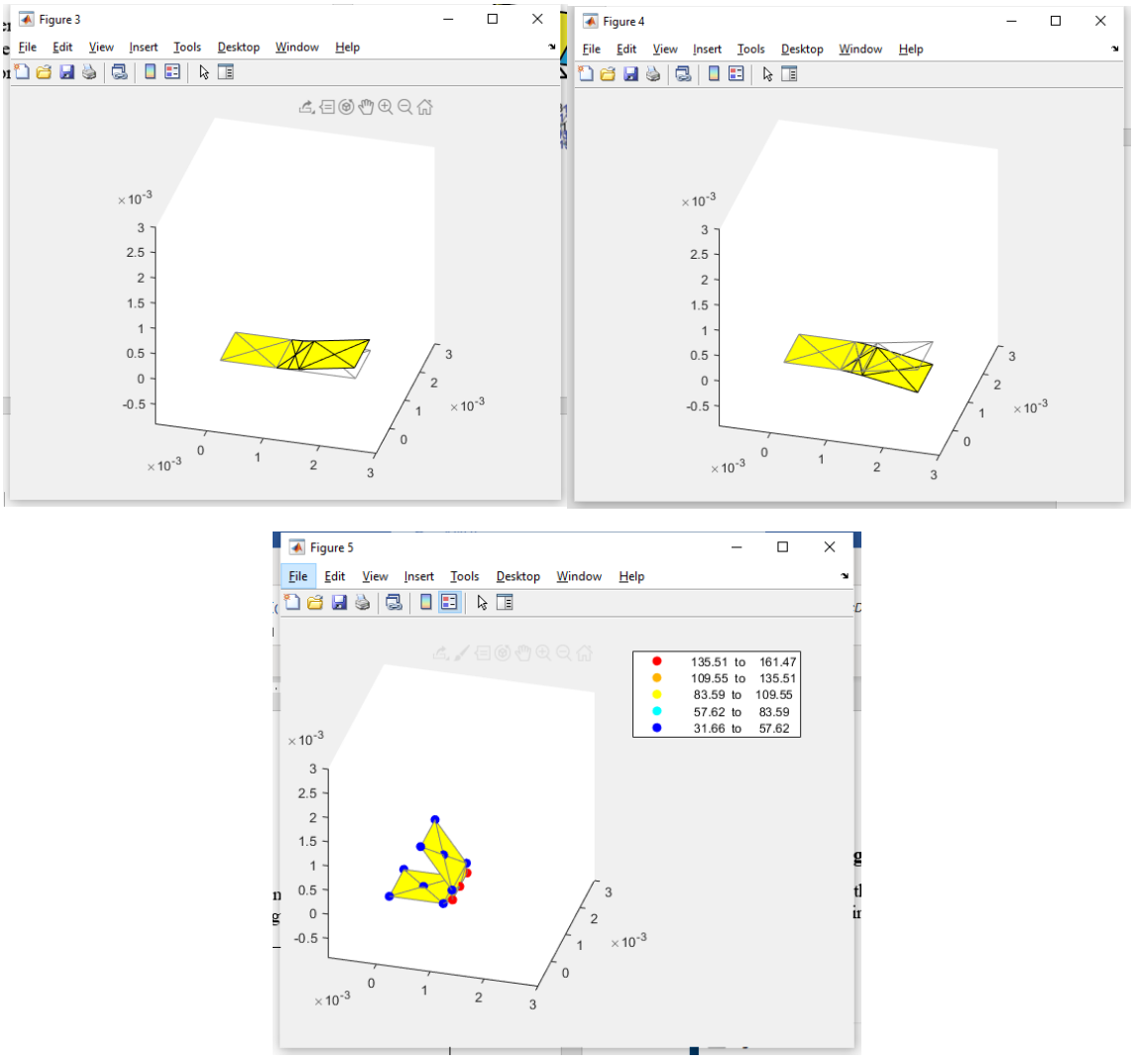


`deltaAlpha` difference in thermal expansion of two later

Step 6
Stack the different loading method and solve the system

After specifying the sequence of loading, we can solve the origami system. The solver will output the final results automatically as demonstrated in the following figures.

```
ori.loadingController{1}={"SelfFold",selfFold};
ori.loadingController{2}={"NR",nr};
ori.loadingController{3}={"ThermalLoading",thermal};


%% Solving the model
ori.Solver_Solve();
```



Step 7
Continuing loading

Suppose after running the three loading cases, we want to add another external load (with the same magnitude and direction as the gravity) we can use the continuing loading capability of the package by setting. Then we get the loaded system as shown in the final figure.

```
%% countiniuing loading
ori.continuingLoading=1;
ori.loadingController={};
ori.loadingController{1}={"NR",nr};
ori.Solver_Solve();
```