Technical

MANUAL

***O**rigami and **T**ensegrity **E**quilibrium and **F**orm-finding **A**nalysis (OriTsgEFA) Software*

@ College of Civil Engineering, Zhejiang University of Technology, China
@ Department of Mechanical and Aerospace Engineering, University of Kentucky, USA
@ Department of Aerospace Engineering, Texas A&M University, USA

August 2023

Revision Sheet

Release No.	Date	Revision Description
Rev. 1.0	29/08/2023	Major functions for OriTsgEFA static analysis are developed. Statics - Equilibrium in forms. - Static deformation. - Stiffness analysis.

Origami and Tensegrity Equilibrium and Form-finding Analysis (OriTsgEFA)

Software Information

Software Goals:

The software is designed to facilitate the process of integrated Origami and Tensegrity (OriTsg) systems. It provides capabilities for modeling, structural design, and conducting nonlinear static simulations for any combined origami-tensegrity systems.

- **Modeling**:
  1). Create models of origami and tensegrity structures using nodal coordinates, connectivity data between nodes, and the hinge angles of individual panels.
  2). Define constraints on nodal coordinates, limiting movements in the X-, Y-, or/and Z-axes for particular nodes.
  3). Facilitate the organization of structural elements into groups within the software.

- **Statics**:
  1). Analyze prestress and mechanism modes using singular value decomposition of the equilibrium matrix.
  2). Conduct analyses for structural stiffness and stability.
  3). Address equilibrium equations under specified external forces, accounting for geometric and material nonlinearities.
  4). Model forced movements by altering the positions of selected nodes or changing the rest length of particular structural elements.

OriTsgEFA Members and Information:

Shuo Ma
mashuo@zjut.edu.cn, Ph.D., Assistant Professor, College of Civil Engineering, Zhejiang University of Technology, Hangzhou, Zhejiang, China.

Muhao Chen
muhaochen@uky.edu, Ph.D., Assistant Professor, Department of Mechanical and Aerospace Engineering, University of Kentucky, Lexington, KY, USA.

Robert E. Skelton
bobskelton@tamu.edu, Ph.D., TEES Eminent Professor, Member National Academy of Engineering, Department of Aerospace Engineering, Joint Faculty in Department of Ocean Engineering, Texas A&M University, College Station, Texas, USA.

TABLE OF CONTENTS

Page #

1.0 GENERAL INFORMATION

1. GENERAL INFORMATION

1.1 System Overview

A foundational understanding of undergraduate-level linear algebra, materials or continuum mechanics, the finite element method, and basic MATLAB skills is necessary to grasp the code thoroughly. The software has been developed based on:

- 64-bit Windows
- MATLAB

The software is compatible with MATLAB versions released after 2009a and can run on various operating systems, including Win7, Win10, Mac OS, Linux, Win XP, and Win Vista. While it will function on older versions, we recommend users operate the software with the most recent release of MATLAB for optimal performance. For more information on MATLAB versions, please visit https://en.wikipedia.org/wiki/MATLAB.

1.2 Project References

The OriTsgEFA software, designed for Origami and Tensegrity Equilibrium and Form-finding Analysis, is built upon the theories outlined in the cited references.

[1] Shuo Ma, Muhao Chen, Hongying Zhang, and Robert E. Skelton. "Statics of integrated origami and tensegrity systems." *International Journal of Solids and Structures*, 279 (2023): 112361.
[2] Shuo Ma, Muhao Chen, Robert E. Skelton. "Finite element analytic formulation for nonlinear tensegrity dynamics," *Composite Structures* 280 (2022): 114838.

1.3 Authorized Use Permission

1.4 Points of Contact

1.4.1 Information

This subject area centers on combining origami and tensegrity structures for static analysis. The analytical framework enables the modeling and examining of origami and tensegrity elements as an integrated system. Descriptions of tensegrity and origami components are based on nodal coordinates and hinge angles between origami panels. Nonlinear static equations for the combined system are derived using the Lagrangian method. The authors aim to provide this software as open-source to benefit other researchers

interested in this specialized field. This user guide covers all facets of the software to enhance its user-friendliness. We welcome your questions and contributions to improve the software further.

1.4.2 Help Desk

We are open and willing to answer any question. Please state your problem clearly and use the following emails to contact: **Muhao Chen**: muhaochen@uuky.edu, **Shuo Ma**: mashuo@zju.edu.cn.

1.5 Organization of the Manual

User's Manual v1.0.

2.0 SYSTEM SUMMARY

2. SYSTEM SUMMARY

2.1 System Configuration

The software does not feature a standalone app interface but utilizes a MATLAB .mat file. Users should ensure that MATLAB (https://www.mathworks.com/products/matlab.html) is correctly installed on their systems. Following the subsequent steps, you can conduct static analyses simulations for any origami and tensegrity structure.

2.2 Analysis Steps

To analyze the structure, the user should follow these steps (more details will be provided in the following chapters):

2.2.1Origami-Tensegrity Topology

- **Specify Structure Configuration**: Create a sketch and label all the elements, including nodes, bars, strings, hinges, and panels, in the preferred arrangement.
- **Specify Node Positions**: Manually specify the node matrix or automatically generate it for predefined tensegrity topologies.
- **Specify Member Connectivity**: Either manually generate a connectivity matrix or automatically create a node matrix for specified origami-tensegrity configurations.
- **Visualize the Structure**: Visualize and verify the integrated origami-tensegrity structure.

2.2.2 Statics Analysis

- **Calculate the Hinge Transformation Matrix:** Specify nodal coordinates related to manual hinges while automatically generating transformation matrices.
- **Specify Boundary Constraints:** Specify nodal coordinates manually with constraints while the software automatically generates the index matrix.
- **Specify Group Information:** Specify group information manually and generate group matrices automatically.
- **Calculate Equilibrium Matrix:** The equilibrium matrices for the tensegrity and origami components are computed independently and combined to create the equilibrium matrix for the entire structure.
- **Specify Member Initial Parameters:** Configure parameters for string prestress, initial hinge moment, member cross-sectional dimensions, rest length, and initial hinge angles.
- **Tangent Stiffness Analysis**: The structure's stiffness is determined by the tensegrity framework and the hinge mechanism. The tangent stiffness for both components is provided, and the eigenvectors corresponding to the overall structural stiffness matrix are computed and graphically represented.
- **Modal Analysis Interface**: Provide a structural mass matrix and integrate it with the stiffness matrix to conduct the modal analysis.
- **Specify External loads**: External loads comprise various elements, such as external forces, nodal displacement, movements of boundary nodes, hinge angle changes, and alterations in rest length.
- **Static Analysis**: Static analysis is performed through the resolution of a nonlinear equilibrium equation, utilizing the modified Newton method.

- **Results Analysis**: The results for member forces, member length, nodal coordinates, hinge moment, and final structure configuration are graphically displayed.
- **Make Video**: Create a video to capture the deformation changes occurring during each sub-step of the static analysis.
- **Exit System**: Click on Exit to close MATLAB.

2.3 Become a Developer

For a comprehensive understanding of the software's code and to explore the development of additional specialized functions, we strongly recommend users read the following paper: **Shuo Ma, Muhao Chen, Hongying Zhang, and Robert E. Skelton. "Statics of integrated origami and tensegrity systems." International Journal of Solids and Structures, 279 (2023): 112361.** We welcome collaborations in integrated origami and tensegrity research and are eager to hear your thoughts on theoretical and practical challenges.

3.0 INSTRUCTION TO ORIGAMI &TENSEGRITY TOPOLOGY

3. INSTRUCTION TO ORIGAMI & TENSEGRITY MODELING

The software includes several sample scripts to showcase its various features, each elaborated in detail. For this description, we will focus on the '2kresling' example.

3.1 Specify tensegrity material properties

Initially, the material types for bars and strings are selected through the 'material_lib' function. Users input the names of two materials, for example, 'Steel_Q345' for bars and 'Steel_string' for strings. The output will provide details like stress-strain constitutive data, Young's modulus, yield stress, and material density for both bars and strings. Materials can be categorized as linearly elastic, multi-linear elastic, or plastic. Assign a zero force to the strings when the strain is negative to simulate string slack.

```
clc; clear all; close all;
% Global variable
[consti_data,Eb,Es,sigmab,sigmas,rho_b,rho_s]=material_lib('Steel_Q345','Steel_string');
Material{1}='linear_elastic'; % index for material properties: multielastic, plastic.
Material{2}=0; % index for considering slack of string (1) for yes,(0) for no (for compare with ANSYS)
```

3.2 Specify Node Positions

The initial step in modeling a tensegrity system involves defining the positions of the nodes. This software keeps track of node positions in a node matrix labeled as [N]. These positions are expressed in 3D coordinates ($[x, y, z]^T$). For 2D systems, set all the z-coordinates to zero. To illustrate, the subsequent code line establishes the nodal positions for a Kresling structure's topology.

```
%%  N C of the structure
% Manually specify node positions
R=10; h=10; p=5; level=3;       % radius; height; number of edge, level
% beta=(0.5-1/p)*pi;  % rotation angle
beta=15*pi/180;       % rotation angle
angle=kron(ones(1,level+1),2*pi*((1:p)-1)./p)+kron(0:level,beta*ones(1,p));
N=R*[cos(angle); sin(angle); zeros(1,p*(level+1))]+kron(0:level,[0;0;h]*ones(1,p));
```

3.3 Specify Member Connectivity

3.3.1 Specify Bar/String Connectivity

After setting the positions for the nodes, one can establish bars and string members based on the connections between these nodes. OriTsgEFA offers a user-friendly approach for defining these elements: an input matrix identifies which nodes are connected to form a specific member. In the example below, fifty bars are defined. Bar1 extends from node 1 to node 2, Bar2 from node 2 to node 3, and so on. This index-based notation is then translated into a comprehensive bar or string connectivity matrix, denoted as C_b for bars and C_s for strings, using the function [tenseg_ind2C]. In the absence of strings in the given structure, the bar connectivity matrix C_b is identical to the overall connectivity matrix C.

```
% Manually specify connectivity indices.
C_b_in_h=[(1:p*(level+1))',kron(ones(level+1,1),[2:p,1]')+kron((0:level)',p*ones(p,1))];% bottom bar
C_b_in_v=[(1:p*level)',(p+1:(level+1)*p)'];% vertical bar
C_b_in_d=[(1:p*level)',kron(ones(level,1),[2:p,1]')+kron((1:level)',p*ones(p,1))];% diagonal bar
C_in = [C_b_in_h;C_b_in_v;C_b_in_d];   % This is indicating the bar connection
% Convert the above matrices into full connectivity matrices.
C = tenseg_ind2C(C_in,N);
[ne,nn]=size(C);       % ne:No.of element;nn: No.of node
```

```
C_b=C;C_s=[];
% Plot the structure to make sure it looks right
tenseg_plot(N,C_b,[]);
```

3.3.2 Specify Hinge Connectivity

We consider this bar-and-hinge structure as a type of truss model. Given that the truss model is a specific instance of the tensegrity model, the relationship between tensegrity and origami structures becomes evident. The hinge or rigid hinge connectivity matrices, denoted as matrices C_h and C_{rh} are extracted from the bar connectivity matrix C_b.

```
%% define hinge, rigid hinge
% C_in_h is the connectivity of hinges and can be written in a function
C_in_h=[C_b_in_h(p+1:(level)*p,:);C_b_in_v;C_b_in_d];
n_h=size(C_in_h,1);        % number of hinge

[~,index_h]=ismember(C_in_h,C_in,'rows');   % index_h is the index number of hinge
index_rh=[];index_rh_in_h=[];
% [~,index_rh]=ismember(C_in_2,C_in,'rows');   % index_h is the index number of rigid hinge
% [~,index_rh_in_h]=ismember(C_in_2,C_in_h,'rows');   % index_h is the index of rigid hinge in all hinge

C_h=tenseg_ind2C(C_in(setdiff([1:ne]',index_rh),:),N);     % connectivity matrix of all edges
C_rh=tenseg_ind2C(C_in(index_rh,:),N); % connectivity matrix of rigid edges
% Plot the structure to make sure it looks right
tenseg_plot(N,C_b,C_s);
```

3.3.3 Connectivity Matrix of the Origami Panel

In our origami structure, triangular panels serve as the basic elements. Each of these triangles is comprised of three nodes and three edges. To simulate the hinges between adjacent panels, we employ rotational springs. The connectivity of the panel is denoted by C_a.

```
%% connectivity of triangle element Ca
% Ca can be written in a function
Ca=kron(ones(1,level),[[1:p;[2:p,1];[p+2:2*p,p+1]],[1:p;[p+2:2*p,p+1];p+1:2*p]])+kron(0:level-1,p*ones(3,p*2));
% Ca=generate_Ca(C_in,N);
% Ca=zeros(3,1)
[~,np]=size(Ca);       % ne: No.of element; np: No.of plate
```

3.4 Visualize the Origami & Tensegrity Structure

The structure can be visualized based on the specified parameters at this stage. This visualization is executed using the [tenseg_plot_ori] function. By employing [tenseg_plot_ori], you can graphically represent the structure to confirm its topology.

```
% plot the origami configuration
tenseg_plot_ori(N,[],[],C_h,C_rh,[],[],[],[] ,[],Ca);
```

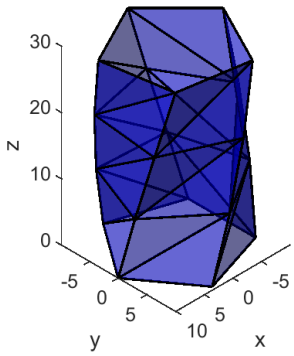

Figure 1. Kresling Structure. The bars and panels are in black and blue.

4.0 INSTRUCTIONS TO STRUCTURE STATICS

4. INSTRUCTIONS TO STRUCTURE STATICS

4.1 Calculate the Hinge Transformation Matrix

It is necessary to link the nodal coordinates associated with a hinge to the overall nodal coordinates of the structure. The 'E_n_total' correlates hinge-specific nodes with the comprehensive node relationships within the structure.

```
%% transformation matrix from element to structure

E_n=cell(1,n_h);           %transformation matrix from element node to total node
node_in_hinge=zeros(n_h,4);
I=eye(3*nn);

for i=1:n_h
node2=C_in_h(i,1);  % start node of the hinge
node3=C_in_h(i,2);  % end node of the hinge
for j=1:np
    if (node2==Ca(1,j)&node3==Ca(2,j))|(node2==Ca(2,j)&node3==Ca(3,j))|(node2==Ca(3,j)&node3==Ca(1,j))
        node1=setdiff(Ca(:,j),[node2;node3]);
    elseif (node2==Ca(2,j)&node3==Ca(1,j))|(node2==Ca(3,j)&node3==Ca(2,j))|(node2==Ca(1,j)&node3==Ca(3,j))
        node4=setdiff(Ca(:,j),[node2;node3]);
    end
end
node_in_hinge(i,:)=[node1,node2,node3,node4];
E_n{i}=I(:,kron(node_in_hinge(i,:),3*ones(1,3))-kron(ones(1,4),[2,1,0]));
end
E_n_total=cell2mat(E_n);       % transformation matrix of the whole structure
```

4.2 Specify Boundary Constraints

Specific nodes may be fixed or "pinned" in specific directions in the context of static structures. To capture this, we use three vectors: pinned_X, pinned_Y, and pinned_Z, which enumerate the nodes pinned in the X, Y, and Z axes. These vectors are then transformed into an index matrix, denoted as $\boldsymbol{I}_a$and $\boldsymbol{I}_b$. This matrix is used to identify the free and boundary nodal coordinates through the [tenseg_boundary] function. In this setup, nodes labeled 1, 2, …, p are considered to be pinned in the X, Y, and Z directions.

```
%% Boundary constraints
pinned_X=[1:p]'; pinned_Y=[1:p]'; pinned_Z=[1:p]';
[Ia,Ib,a,b]=tenseg_boundary(pinned_X,pinned_Y,pinned_Z,nn);
```

4.3 Specify Group Information

Members positioned symmetrically may share identical lengths and forces, warranting their classification into the same group. To capture this group information, we utilize a structure array named 'gr.' Subsequently, the function [tenseg_str_gp] is employed to convert the 'gr' array into the group matrix, denoted as $\mathbf{G}_{\boldsymbol{p}}$.

```
%% generate group index for tensegrity torus structure
gr=[];                    %no group is used
Gp=tenseg_str_gp(gr,C);    %generate group matrix
S=Gp';                    % clustering matrix
```

4.4 Calculate Equilibrium Matrix

The overall structure can formulate a linear equilibrium equation related to axial force $\boldsymbol{t}$ and hinge moment $\boldsymbol{M}$. The equilibrium matrix for bar elements, termed 'A_2,' is generated by the function [tenseg_equilibrium_matrix_CTS]. Similarly, the equilibrium matrix for hinge elements, referred to as 'phTpn,' is computed using [jacobian_ori]. The final result, 'A_o', is the equilibrium matrix for the combined origami and tensegrity systems.

```
%% equilibrium matrix
% equilibrium matrix of bar
[A_1,A_1c,A_1a,A_1ac,A_2,A_2c,A_2a,A_2ac,l,l_c]=tenseg_equilibrium_matrix_CTS(N,C,S,Ia);
% [A_1,A_1g,A_2,A_2g,l,l_gp]=tenseg_equilibrium_matrix2(N,C,Gp,Ia);
% equilibrium matrix of hinge
[phpn_e,phTpn,theta]=jacobian_ori(node_in_hinge,N,E_n_total);      % Jacobian matrix
A_o=[A_2,phTpn];
A_o_a=Ia'*A_o;
```

4.5 Specify Member Initial Parameters

4.5.1 Specify Member Self-Stress Design

Subsequently, the bar prestresses and the initial moment at the hinge are set to zero.

```
%% self-stress design (of the bar)
t=zeros(ne,1);     %member force
q=t./l;            % force density
%% self-stress design (of hinge)
M=zeros(n_h,1);
```

4.5.2 Specify Member Cross-Sectional / Rest Length / Angle

Using vectors, we explicitly defined the cross-sectional area, Young's modulus, and rest length for both bar and string elements. With the initial angle of the hinge provided, we then computed its rotational stiffness.

```
%% cross-sectional design (of bars and strings)
A_c=1e-4*ones(ne,1);
E_c=1e9*ones(ne,1);
index_b=[1:ne]';             % index of bar in compression
index_s=setdiff(1:size(S,1),index_b);       % index of strings
%% hinge section design  (of hinge)
k_h=1/12*E_c(index_h).*l(index_h)*thick^3;
k_h(index_rh_in_h)=1e2*1/12*E_c(index_rh).*l(index_rh)*thick^3;     % increase stiffness of rigid hinge
%% rest length (of strings), initial angle (of hinge)
l0_c=l;                   %rest length of truss
theta_0=theta;    % initial angle of the hinge
```

4.6 Tangent Stiffness Analysis

The overall structure's tangent stiffness is derived from the truss elements of the tensegrity structure and the hinge elements of the origami structure. The code initially computes the tangent stiffness matrix for the tensegrity using the function [tenseg_stiff_CTS], while the function [hessian] generates the Hessian matrix for the origami structure. Subsequently, the tangent stiffness matrix for the complete structure is obtained. Additionally, the function [plot_mode_ori] is available to visualize the eigenvectors of the tangent stiffness matrix. The first and second-order eigenvectors are displayed accordingly.

```
%% tangent stiffness matrix of bars
% [Kt_aa,Kg_aa,Ke_aa,K_mode,k]=tenseg_stiff_CTS(Ia,C,S,q,A_1a,E_c,A_c,l_c);
```

```
[Kt_aa,Kg_aa,Ke_aa,K_mode,k]=tenseg_stiff_CTS2(Ia,C,q,A_2ac,E_c,A_c,l0_c);

%% tangent stiffness matrix of hinge

%ph2px2 is the hessian matrix of theta to nodal coordinate
[ph2pn2_e,ph2pn2]=hessian_ori(node_in_hinge,N,E_n);        % calculate hessian matrix
G=cell2mat(ph2pn2);

%% tangent stiffness of the whole origami

K_t_oa=Kt_aa+Ia'*(phTpn*diag(k_h)*phTpn'+G*kron(M,eye(3*nn)))*Ia;

[K_mode,D1] = eig(K_t_oa);        % eigenvalue of tangent stiffness matrix
k=diag(D1);
% plot the mode shape of the tangent stiffness matrix
num_plt=1:6;
plot_mode_ori(K_mode,k,N,Ia,[],[],C_h,C_rh,l,'tangent stiffness matrix',...
    'Order of Eigenvalue','Eigenvalue of Stiffness (N/m)',num_plt,0.2,saveimg,3,Ca);
```

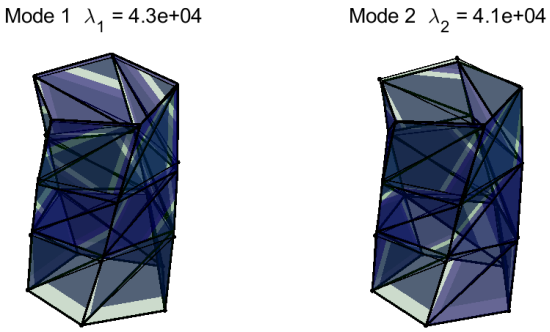

Figure 2. The first and second eigenvectors of the Kresling structure.

4.7 Modal Analysis Interface

The function [plot_mode_ori] can visualize modal shapes and frequencies.

```
%% mass matrix and damping matrix
rho=1;
mass=rho.*A_c.*l0_c;
M=tenseg_mass_matrix(mass,C,lumped); % generate mass matrix
% damping matrix
d=0;     %damping coefficient
D=d*2*max(sqrt(mass.*E_c.*A_c./l0_c))*eye(3*nn);    %critical damping
%% vibration mode analysis
[V_mode,D1] = eig(K_t_oa,Ia'*M*Ia);        % calculate vibration mode
w_2=diag(D1);                                 % eigenvalue of
omega=real(sqrt(w_2))/2/pi;                  % frequency in Hz
if 0
plot_mode_ori(K_mode,k,N,Ia,[],[],C_h,C_rh,l,'natrual vibration',...
    'Order of Vibration Mode','Frequency (Hz)',num_plt,0.2,saveimg,3,Ca);
end
```

4.8 Specify External Loads

External loads in the software can comprise various cases, such as external force, movement of boundary nodes, changes in hinge angles, and alterations in rest length. For instance, one can specify an external force of F=-500,000N to be applied at the top nodes in the Z-direction. The vectors 'ind_w' and 'w' store the index of the nodal coordinate where the external force is applied and the magnitude of that force, respectively. Similarly, vectors 'ind_dnb' and 'dnb0' hold information about moved nodal coordinates and the displacement of their movement. The index and magnitude of changes in rest length are stored in 'ind_dl0_c'

and 'dl0_c'. For rotational hinge angles, the index and magnitude of the angle difference are stored in 'ind_theta_0' and 'dtheta_0'. The function [tenseg_load_prestress_ori] converts all this constraint information into input data suitable for static analysis.

```
%% external force, forced motion of nodes, shrink of strings
% calculate external force and
ind_w=[3*(level*p+1:(level+1)*p)];w=-5e5*ones(p,1);   %external force in Z
ind_dnb=[]; dnb0=[];
ind_dl0_c=[]; dl0_c=[];
ind_theta_0=[]; dtheta_0=[];       % initial angel change with time
[w_t,dnb_t,l0_ct,theta_0_t,Ia_new,Ib_new]=tenseg_load_prestress_ori(substep,ind_w,w,ind_dnb,dnb0,ind_dl0_c,dl0_c,
ind_theta_0,dtheta_0,theta_0,l0_c,b,gravity,[0;9.8;0],C,mass);
```

4.9 Static Analysis

At this point, sufficient details have been provided to run a simulation, including defining the structure's topology, prestress conditions, cross-sectional areas, and the external forces that trigger movement. To execute the simulation, a 'data' structure must be created. This structure should include the following fields: configuration (N), topology(C), number of elements (ne), number of nodes (nn), information of free and pinned nodal coordinates (Ia_new, Ib_new), Young's modulus vector(E), cross-sectional area vector(A), rest length vector(l0), index of bars and strings(index_b, index_s), constitutive relation of members(consti_data), material elastoplastic properties(material), external force(w_t), movement of pinned nodes(dnb_t), rest length of members(l0_t), angle of hinges(theta_0_t), stiffness of hinge(k_h), transformation matrix from hinge to structure(E_n), node in hinge(node_in_hinge), maximum number of iterations(MaxIcr), number of sub-steps (substep). The static analysis is carried out by the function [static_solver_ori_2].

```
%% Step1: equilibrium calculation
% input data
data.N=N; data.C=C; data.ne=ne; data.nn=nn; data.Ia=Ia_new; data.Ib=Ib_new;data.S=S;
data.E=E_c; data.A=A_c; data.index_b=index_b; data.index_s=index_s;
data.consti_data=consti_data;   data.material=material; %constitue info
data.w_t=w_t;  % external force
data.dnb_t=dnb_t;          % forced movement of pinned nodes
data.l0_t=l0_ct;           % forced change of rest length
data.theta_0_t=theta_0_t;   % forced change of initial angle
data.k_h=k_h;              % stiffness of hinge
data.E_n=E_n;              % transfer matrix from matrix to structure
data.node_in_hinge=node_in_hinge;      % node in triangle element in hinge
data.substep=substep;    % substep
data.InitialLoadFactor=0.001;
data.MaxIcr=1000;
data.LoadType='Force'; % 'Force' or 'Displacement'
data.StopCriterion=@(U)(norm(U)>20);

% nonlinear analysis
data_out1=static_solver_ori_2(data);
```

4.10 Results Analysis

By clicking the "run" button in MATLAB, the static analysis outcome will be stored in 'data_out.' Subsequent functions are available to visualize various results, including member forces, member lengths, hinge moments, nodal coordinates, and the final structural configuration.

```
%% Plot final configuration
 j=linspace(1e-5,1,4);
for i=1:4
    num=ceil(j(i)*size(n_t,2));
```

```
tenseg_plot_ori(reshape(n_t(:,num),3,[]),[],[],C_h,C_rh,[],[],[30,30],[] ,[],Ca);
%  axis off;
end
%% plot member force
tenseg_plot_result(Fhis,t_t([1,10,20],:),{'truss1','truss10','truss20'},{'Load factor','Force /
N'},fullfile(savePath,'plot_member_force.png'),saveimg);
%% plot member length
tenseg_plot_result(Fhis,l_out([1,10,20],:),{'truss1','truss10','truss20'},{'Load factor','length /
m'},fullfile(savePath,'plot_member_length.png'),saveimg);

%% plot hinge moment
tenseg_plot_result(Fhis,M_out,{'rgd1','rgd2','rgd3','sft1','sft2'},{'Load factor','Moment / N \times
m'},fullfile(savePath,'plot_hinge_moment.png'),saveimg);

%% Plot nodal coordinate curve X Y Z
tenseg_plot_result(Fhis,n_t(3*(level*p+1),:),{'top node Z'},{'Load factor','Coordinate
/m)'},fullfile(savePath,'plot_coordinate.png'),saveimg);
```

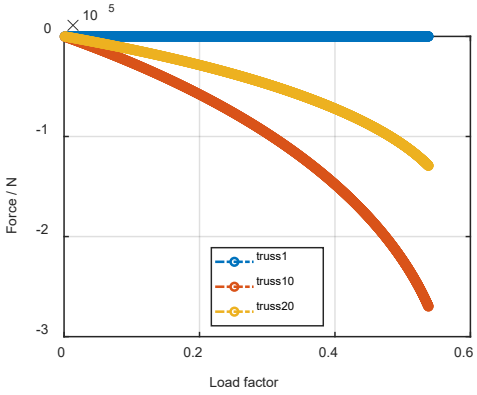

Figure 1 Member forces in static analysis.

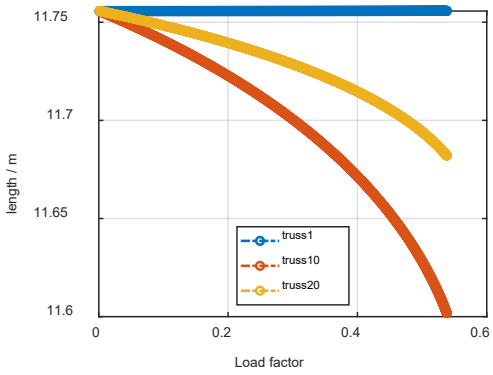

Figure 2 Member lengths in static analysis.

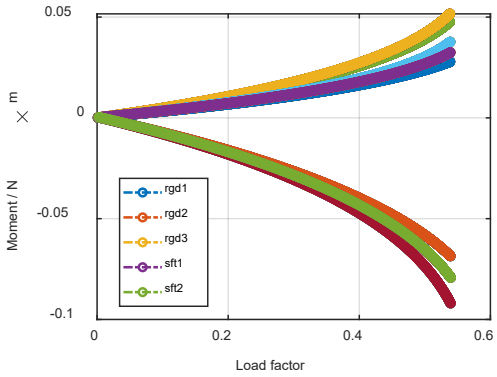

Figure 3 Hinge moments in static analysis.

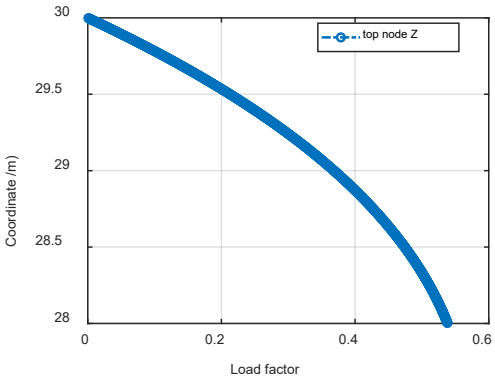

Figure 4 Nodal coordinates in static analysis.

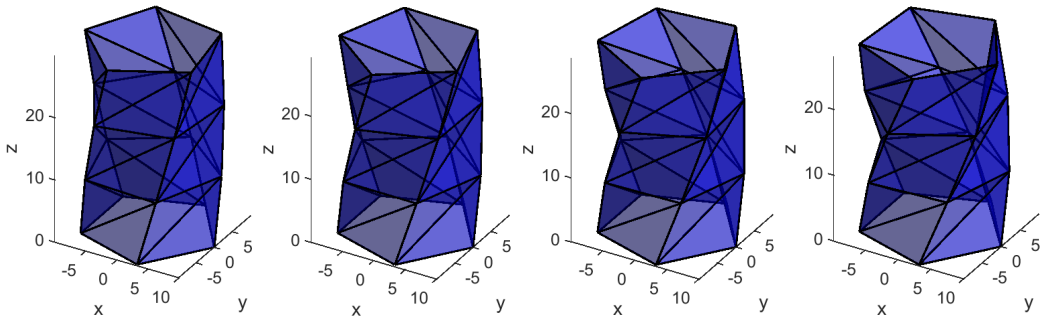

Figure 5 Final configurations in the static analysis.

4.11 Make video

Images of the structure's configuration at each static analysis step are compiled to create a video. The function [tenseg_video_ori] generates these videos as 'gif' files for the origami-tensegrity structure.

```
%% Make a video of the dynamic
name=fullfile(savePath,'origami_kresling');
% name=['origami_kresling'];
% tenseg_video(n_t,C_b,C_s,[],min(substep,50),name,savevideo,R3Ddata);
% tenseg_video_slack(n_t,C_b,C_s,l0_ct,index_s,[],[],[],min(substep,50),name,savevideo,material{2});
tenseg_video_ori(n_t,[],[],C_h,C_rh,Ca,[],min(icrm,50),name,savevideo,[]);
```

4.12 Exit System

Click on Exit to close MATLAB.