Technical MANUAL

***T**en**s**e**g**rity **F**inite **E**lement **M**ethod (TsgFEM) Software*

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

June 2021

Revision Sheet

Release No.	**Date**	**Revision Description**
Rev. 0.0	12/01/2019	Major functions for FEM statics and dynamics are developed.
Rev. 0.1	03/15/2020	1. Statics - Equilibrium in three forms. 2. Dynamics - Modified the codes, the codes run more efficiently. - Tested various boundary condition examples. - Modal analysis.
Rev. 0.2	8/15/2020	Code Revised Statics - Added global buckling analysis and examples. Dynamics - Add more elastic and plastic deformation examples.
Rev. 0.3	11/01/2020	User's Manual Created
Rev. 0.4	06/01/2021	Updated User's Manual
Version 1.1	06/09/2021	Submitted to JOSS (Journal of Open Source Software)
	09/09/2021	Revised according to JOSS reviews
	12/24/2021	Revised according to JOSS reviews
	01/21/2022	Revised according to JOSS reviews

Tensegrity Finite Element Method (TsgFEM)

Software Information

Software Goals:

The purpose of this software is to facilitate the statics and dynamics of Tensegrity systems based on FEM. The software allows performing modeling, structural design, nonlinear static, and dynamic FEM simulation of any tensegrity and truss systems.

- **Modeling**:
  1). Model any tensegrity structures by nodal coordinates and the node's connectivity information.
  2). Specify the constraints of nodal coordinates (restricted motions of the X-, X-, or/and Z-directions of some certain nodes).
  3). Normally, structure members in symmetric positions have the same force densities. They are allowed to be grouped in the software.
- **Statics**:
  1). Prestress modes and mechanism modes analysis by singular value decomposition of equilibrium matrix.
  2). Prestress design and minimal mass calculation considering yielding and buckling constraints.
  3). Stiffness and stability analysis.
  4). Solve the equilibrium equations with any given external forces considering nonlinear geometry and material.
  5). Simulate the forced motion by giving sequences of some nodes or changing the rest length of some members.
- **Dynamics**:
  1). Linearized dynamics.
  2). Modal analysis, calculate natural frequencies and mode shapes.
  3). Nonlinear dynamics simulation of any tensegrity structures or truss systems with linear elastic，multilinear elastic, or plastic materials.

TsgFEM 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@tamu.edu, Ph.D., Postdoctoral Researcher, Department of Aerospace Engineering, Texas A&M University, College Station, Texas, 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

1.0 GENERAL INFORMATION

1. GENERAL INFORMATION

1.1 System Overview

Undergraduate linear algebra, material mechanics/continuum mechanics, finite element method, and some basic knowledge of MATLAB are required to understand the codes well. This software is developed based on:

- 64-bit Windows
- MATLAB
- MATLAB Optimization Toolbox

Note: Win7/Win10/Mac OS/Linux/Win XP/Win Vista, the software is compatible with a MATLAB version later than 2009a. However, we encourage the user to run the software with the latest MATLAB release if possible. (More information about MATLAB versions can be found here: https://en.wikipedia.org/wiki/MATLAB). Since 'linprog' and 'fmincon' functions from MATLAB Optimization Toolbox are used in statics calculation, this toolbox should also be installed (more information can be found here: https://www.mathworks.com/products/optimization.html).
Since commercial software embedded a lot more material database, for research purposes, one may also use

- ANSYS

TsgFEM also provides an interface to ANSYS. The automatically generated interface file allows users to perform the FEM simulation in ANSYS automatically, and it is compatible with an ANSYS version later than 16.0. However, we also encourage the user to run the software with the latest ANSYS release if possible.

1.2 Project References

Tensegrity Finite Element Method (TsgFEM) software is created based on the theories developed in the following references.
[1] Muhao Chen and Robert E. Skelton. "A general approach to minimal mass tensegrity." Composite Structures 248 (2020): 112454.
[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

Our group focuses on the research of integrating structure and control design. Based on the tensegrity paradigm, we design tensegrity structures to meet the specified objectives. These objectives can vary from minimizing the mass of the structure to controlling the structure to meet certain performance. This software is intended to study the statics and dynamics of tensegrity systems based on FEM. The authors would like to make this open-source software to help other researchers who are also interested in this field.

In this user guide, we state every aspect of the software to make it more user-friendly. We appreciate your questions and any help in improving the software.

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@tamu.edu, **Shuo Ma**: mashuo@zju.edu.cn

1.5 Organization of the Manual

User's Manual v2.1.

2.0 SYSTEM SUMMARY

2. SYSTEM SUMMARY

2.1 System Configuration

This software does not have a specific APP user interface; a MATLAB .mat file is implemented as one. The user should make sure MATLAB (https://www.mathworks.com/products/matlab.html) is well installed. Following the steps mentioned below, one can perform the statics analysis and dynamics simulations for any 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):

2.2.1Tensegrity Topology

- ➢ **Specify Structure Configuration**: Draw the sketch and number all the nodes, bars, and strings in any desired manner.
- ➢ **Specify Node Positions**: Manual node matrix specification or Automatic node matrix generation for some given tensegrity topologies.
- ➢ **Specify Bar/String Connectivity**: Manual connectivity matrix generation or Automatic node matrix generation for some given tensegrity topologies.
- ➢ **Visualize the Tensegrity Structure**: Plot and check the tensegrity structure.

2.2.2 Statics Analysis

- ➢ **Specify Boundary Constraints:** Manual constrained nodal coordinate specification and automatic index matrix generation.
- ➢ **Specify Group Information:** Manual group information specification and automatic group matrix generation.
- ➢ **Prestress Design:** Calculate the equilibrium matrix and do singular value decomposition of the equilibrium matrix. Design prestress in external force and specified member force.
- ➢ **Cross-Sectional Area Design**: Minimal mass design of bars in compression and strings in tension. The safe coefficient and the thickness of hollow bars can be specified.
- ➢ **Specify External loads**: External loads include but are not limited to external force, boundary node movement, and change of rest length.
- ➢ **Static Analysis**: Static analysis is conducted by solving nonlinear equilibrium equations based on the modified Newton method.
- ➢ **Results Analysis**: The results of member force, nodal coordinate, and final configuration are plotted.
- ➢ **Make Video**: Make a video of the deformation process in every sub-step of the static analysis.
- ➢ **Exit System**: Click on Exit to close MATLAB.

2.2.3 Dynamics Analysis

- ➢ **Specify Structure Configuration**: Draw from sketch number all the nodes, bars, and strings in any desired manner.

- **Specify Node Positions**: Manual node matrix specification or Automatic node matrix generation for some given tensegrity topologies.
- **Specify Bar/String Connectivity**: Manual connectivity matrix generation or Automatic node matrix generation for some given tensegrity topologies.
- **Visualize the Tensegrity Structure**: Plot and check the tensegrity structure.
- **Specify Pinned Nodes**: Manual specify pinned nodes.
- **Assign Prestress in Strings**: Manual string rest length specification.
- **Specify External Force**: Manual nodes velocity and time-varying external forces.
- **Prepare for Simulation**: Manual ode solver time step and simulation time.
- **Perform the Simulation**: Click MATLAB "run" button to perform the simulation.
- **Results Analysis**: Plot the position and velocity of any desired nodes, force density in bars and strings, and generate a video simulating the motion of the structure.
- **Exit System**: Click on Exit to close MATLAB.

2.3 Become a Developer

To thoroughly understand the codes in this software and develop more functions for specific purposes, we highly encourage the user to read this paper: **Shuo Ma, Muhao Chen, Robert E. Skelton. Tensegrity system dynamics based on finite element method, arxiv.org/abs/2106.02176v1.** We are open to collaborating in tensegrity research projects and would also like to hear your opinions on both theoretical and practical problems.

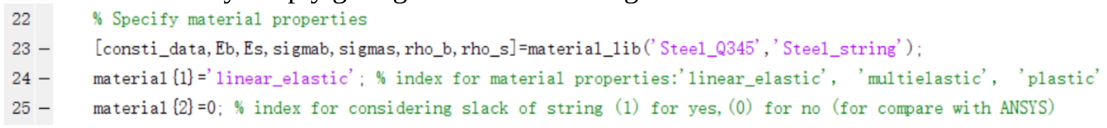
3.0 INSTRUCTION FOR TENSEGRITY TOPOLOGY

3. INSTRUCTION FOR TENSEGRITY MODELING

The software comes with a few example scripts to demonstrate different aspects of the functionality, which are explained in great detail here.

3.1 Specify material properties

We first choose the material type of bars and strings using the function 'material_lib.' The input information is the name of two materials, for example, 'Steel_Q345' and 'Steel_string.' The output data includes the stress-strain constitutive data, Young's modulus, the yielding stress, the material density of bars and strings. The material can be linear elastic, multilinear elastic, or plastic. And the slack of strings can be simulated by simply giving zero force of strings if the strain is less than zero.



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

3.2 Specify Structure Configuration

To analyze any tensegrity structure, we suggest the user draw the sketch and number all the nodes, bars, and strings in any desired manner. For example, to analyze a tensegrity tower structure shown in Figure 1. We suggest the user follow these steps:
1. Sketch the structure.
2. Label all nodes, bars, and strings in any order. Nodes, bars, and strings are shown in blue, black, and red, respectively.
3. Bars and strings are vectors; thus, each member has a direction.
4. For complex structures, find governing rules to generate Node Positions $\mathrm{N}, \mathrm{C_b}, C_s$ (described in Sections 3.2, 3.3, and 3.4) with respect to structural complexity.
5. Check the structure by plots (in Section 3.5).

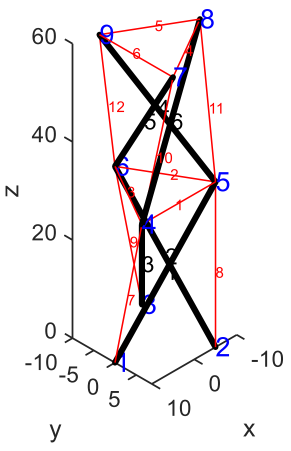
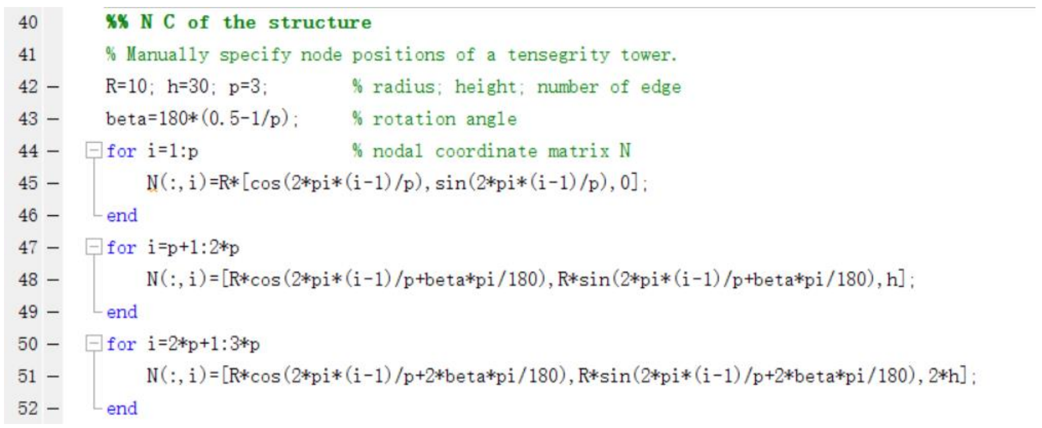


Figure 1. Tensegrity Tower Configuration

3.3 Specify Node Positions

Modeling of a tensegrity system begins with specifying node positions. In this software, node positions are stored in a [node matrix N]. Position coordinates are described in 3D space ($[x, y, z]^T$). If you are describing a 2D system, set all z-coordinates to zero. The following line of code defines the nodal position of a tensegrity tower. The tensegrity is made of two prisms with p edges connected by the top and bottom nodes.

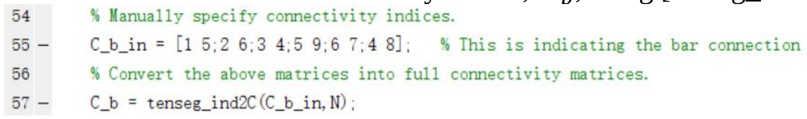


```
%% N C of the structure
% Manually specify node positions of a tensegrity tower.
R=10; h=30; p=3;          % radius; height; number of edge
beta=180*(0.5-1/p);       % rotation angle
for i=1:p                 % nodal coordinate matrix N
    N(:,i)=R*[cos(2*pi*(i-1)/p),sin(2*pi*(i-1)/p),0];
end
for i=p+1:2*p
    N(:,i)=[R*cos(2*pi*(i-1)/p+beta*pi/180),R*sin(2*pi*(i-1)/p+beta*pi/180),h];
end
for i=2*p+1:3*p
    N(:,i)=[R*cos(2*pi*(i-1)/p+2*beta*pi/180),R*sin(2*pi*(i-1)/p+2*beta*pi/180),2*h];
end
```

3.4 Specify Member Connectivity

3.4.1 Specify Bar Connectivity

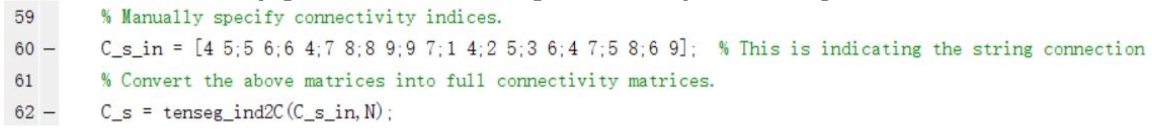
After specifying node positions, bar and string members can be defined in terms of connections between the nodes. TsgFEM provides an intuitive method for defining these members: an input matrix is defined by specifying which nodes connect to make a member. Below, six bars are defined. Bar1 is a vector from nodes 1 to 5, bar2 is a vector from nodes 2 to 6, bar3 is a vector from nodes 3 to 4, bar4 is a vector from nodes 5 to 9, bar5 is a vector from nodes 6 to 7, bar6 is a vector from nodes 4 to 8. This index notation is converted into a full bar connectivity matrix, C_b, using [tenseg_ind2C].



```
% Manually specify connectivity indices.
C_b_in = [1 5;2 6;3 4;5 9;6 7;4 8];   % This is indicating the bar connection
% Convert the above matrices into full connectivity matrices.
C_b = tenseg_ind2C(C_b_in,N);
```

3.4.2 Specify String Connectivity

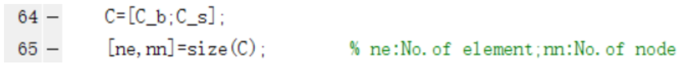
This same index connectivity notation can be used to define string members. The function tenseg_ind2C can also be applied to the string connectivity matrix. Below, twelve strings are defined: string1 is a vector from nodes 4 to 5, string2 is a vector from nodes 5 to 6, etc. This index notation is converted into a full bar connectivity matrix, C_s, using [tenseg_ind2C]. The following lines specify connectivity for two string members and similarly generate the full string connectivity matrix, C_s.



```
% Manually specify connectivity indices.
C_s_in = [4 5;5 6;6 4;7 8;8 9;9 7;1 4;2 5;3 6;4 7;5 8;6 9];  % This is indicating the string connection
% Convert the above matrices into full connectivity matrices.
C_s = tenseg_ind2C(C_s_in,N);
```

3.4.3 Connectivity matrix of the structure

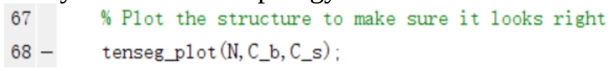
Bar connectivity and string connectivity matrix are combined to form a connectivity matrix of the structure.



```
C=[C_b;C_s];
[ne,nn]=size(C);          % ne:No.of element;nn:No.of node
```

3.5 Visualize the Tensegrity Structure

At this point, we can visualize what our structure looks like based on what we have specified. This visualization is performed with [tenseg_plot]. Using function [tenseg_plot], we can plot the structure to verify the structure topology.



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

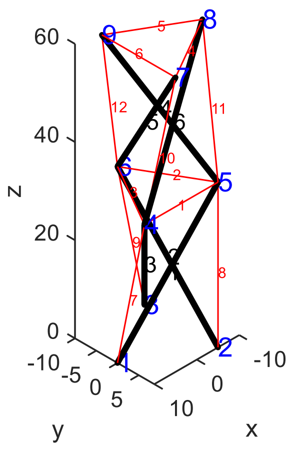


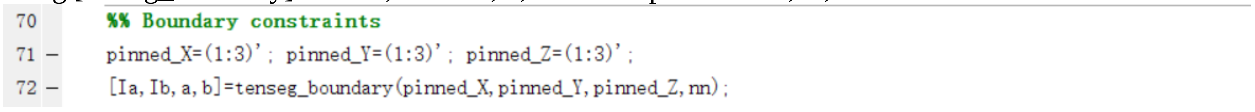
Figure 2 Tensegrity Tower Structure (Nodes, bars, and strings are in blue, black, and red, respectively.)

4.0 INSTRUCTIONS FOR TENSEGRITY STATICS

4. INSTRUCTION FOR TENSEGRITY STATICS

4.1 Specify Boundary Constraints

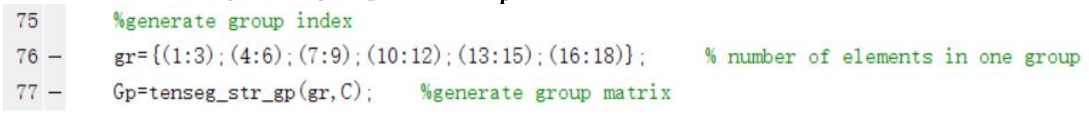
For static structures, some nodes might be pinned in some directions. We define three vectors pinned_X, pinned_Y, and pinned_Z, which contain the number of pinned nodes in X, Y, and Z direction. These three vectors are converted into an index matrix $\boldsymbol{I}_a, \boldsymbol{I}_b$, which locate the free and boundary nodal coordinate, using [tenseg_boundary]. Below, nodes 1, 2, and 3 are pinned in X, Y, and Z direction.



```
%% Boundary constraints
pinned_X=(1:3)'; pinned_Y=(1:3)'; pinned_Z=(1:3)';
[Ia,Ib,a,b]=tenseg_boundary(pinned_X,pinned_Y,pinned_Z,nn);
```

4.2 Specify Group Information

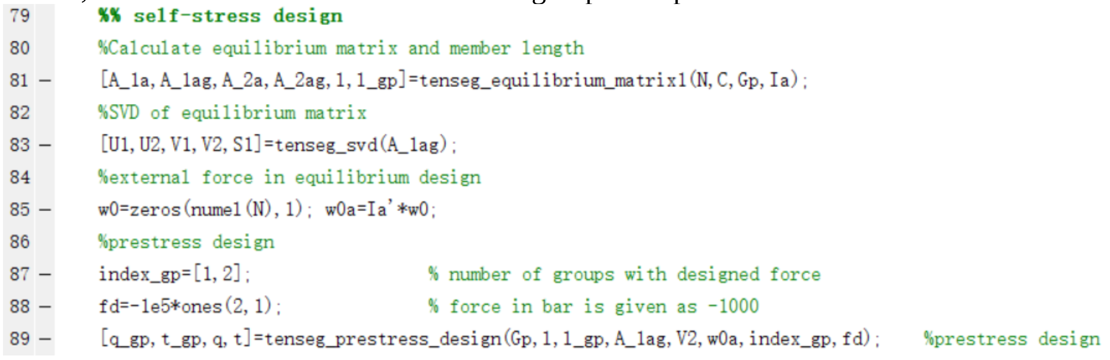
Some members in symmetrical positions have the same length and force, so they can be classified into the same group. Below, members 1 to 3, 4 to 6, 7 to 9, 10 to 12, 13 to 15, 16 to 18 are classified into six groups. We use a structure array 'gr' to record the information of groups. Then a function [tenseg_str_gp] is used to transfer 'gr' into group matrix $\mathbf{G}_{\boldsymbol{p}}$.



```
%generate group index
gr={(1:3);(4:6);(7:9);(10:12);(13:15);(16:18)};     % number of elements in one group
Gp=tenseg_str_gp(gr,C);    %generate group matrix
```

4.3 Prestress Design

In the prestress design, we first generate the equilibrium matrix considering boundary constraints and group information, using function [tenseg_equilibrium_matrix1]. The singular value decomposition of the equilibrium matrix is conducted to get the four subspaces. The prestress subspace is in the null space of the equilibrium matrix $\boldsymbol{V}_2$. The external force and prestress of specific groups are given, and the prestress is solved by function [tenseg_prestress_design]. Below, the external force in every free nodal coordinate is zero, and the initial force of the 1st and 2nd groups are specified as -10^5N.



```
%% self-stress design
%Calculate equilibrium matrix and member length
[A_1a,A_1ag,A_2a,A_2ag,l,l_gp]=tenseg_equilibrium_matrix1(N,C,Gp,Ia);
%SVD of equilibrium matrix
[U1,U2,V1,V2,S1]=tenseg_svd(A_1ag);
%external force in equilibrium design
w0=zeros(numel(N),1); w0a=Ia'*w0;
%prestress design
index_gp=[1,2];                    % number of groups with designed force
fd=-1e5*ones(2,1);                 % force in bar is given as -1000
[q_gp,t_gp,q,t]=tenseg_prestress_design(Gp,l,l_gp,A_1ag,V2,w0a,index_gp,fd);    %prestress design
```

4.4 Cross-Sectional Area Design

The cross-sectional area is designed by yielding or buckling constraints of strings and bars, using function [tenseg_minimass]. To ensure the safety of members, the cross-sectional area is designed so that the prestress is only a small portion, c_b for bars and c_s for strings, of the stress of ultimate bearing load. For example, $\text{c\_b} = 0.1, \text{c\_s} = 0.1$.

```
%% cross sectional design
index_b=find(t<0);              % index of bar in compression
index_s=setdiff(1:ne,index_b);  % index of strings
[A_b,A_s,A_gp,A,r_b,r_s,r_gp,radius,E,l0,rho,mass]=tenseg_minimass(t,l,Gp,sigmas,sigmab,Eb,Es,index_b,index_s,c_b,c_s,rho_b,rho_s,thick,hollow_solid);
```

The tensegrity can be plotted with the real radius from the above cross-sectional area. The following line of code plots the structure with radius information.

```
% Plot the structure with radius
R3Ddata.Bradius=interp1([min(radius),max(radius)],[0.2,0.8],r_b);
R3Ddata.Sradius=interp1([min(radius),max(radius)],[0.2,0.8],r_s);
R3Ddata.Nradius=ones(nn,1);
tenseg_plot(N,C_b,C_s,[],[],[],'Double layer prism',R3Ddata);
```

4.5 Specify External Loads

The external loads include but are not limited to external force, boundary node movement, and change of rest length. The external force can be defined at any node in any direction. For example, in this structure, we want to apply $F = 9{,}000\text{N}$ force at node 4 in the x-direction. Move the z-coordinate of 7, 8, 9 nodes by 5m, and change the 1st member rest length for -0.3m. The index of nodal coordinate in external force and the magnitude of external force is stored in vectors' ind_w' and 'w,' respectively. The index of moved nodal coordinate and moving distance are stored in vectors' ind_dnb' and 'dnb0', respectively. The index of members with changed rest length and magnitude of rest length difference is stored in vectors' ind_dl0' and 'dl0', respectively. The function [tenseg_load_prestress] is used to transfer the above constraint information into input data for static analysis.

```
%% external force, forced motion of nodes, shrink of strings
ind_w=4*3-2;w=9*1000;
ind_dnb=[3*[7:9]']; dnb0=5*ones(3,1);
ind_dl0=1; dl0=-0.3;
[w_t,dnb_t,l0_t,Ia_new,Ib_new]=tenseg_load_prestress(substep,ind_w,w,ind_dnb,dnb0,ind_dl0,dl0,l0,b,gravity,[0;9.8;0],C,mass);
```

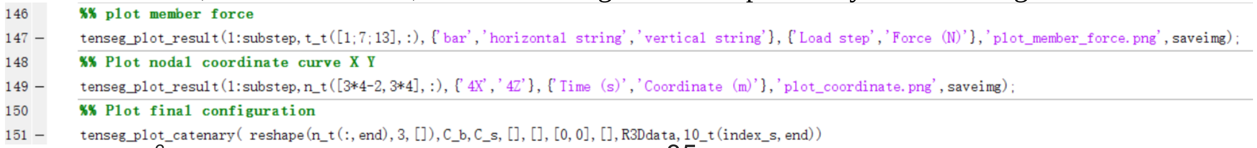
4.6 Static Analysis

At this point, we have given enough information to perform a simulation that we have defined the structure shape, prestress, cross-sectional area, and external loads that should induce motion. To perform a simulation, we need to create a data structure named 'data' here. This data structure requires, configuration(N), topology(C), number of element(ne), number of node(nn), information of free and pinned nodal coordinate(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), number of sub-steps(substep). The static analysis is performed by function [static_solver2]. These fields are populated as follows:

```
%% 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.E=E; data.A=A; data.l0=l0; 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_t;% forced movement of pinned nodes
data.substep=substep;    % substep
% nonlinear analysis
data_out=static_solver2(data);       %solve equilibrium using mNewton method
```

4.7 Results Analysis

By clicking the "run" button in MATLAB, we can obtain the static analysis result in 'data_out.' The results of member force, nodal coordinate, and final configuration are plotted by the following functions.



```
%% plot member force
tenseg_plot_result(1:substep,t_t([1;7;13],:),{'bar','horizontal string','vertical string'},{'Load step','Force (N)'},'plot_member_force.png',saveimg);
%% Plot nodal coordinate curve X Y
tenseg_plot_result(1:substep,n_t([3*4-2,3*4],:),{'4X','4Z'},{'Time (s)','Coordinate (m)'},'plot_coordinate.png',saveimg);
%% Plot final configuration
tenseg_plot_catenary( reshape(n_t(:,end),3,[]),C_b,C_s,[],[],[0,0],[],R3Ddata,l0_t(index_s,end))
```

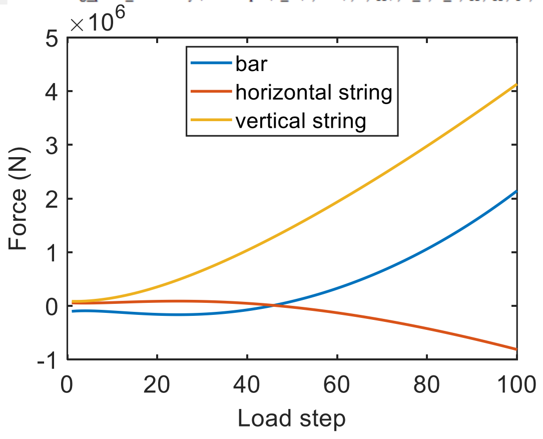


Figure 3 Member force in static analysis

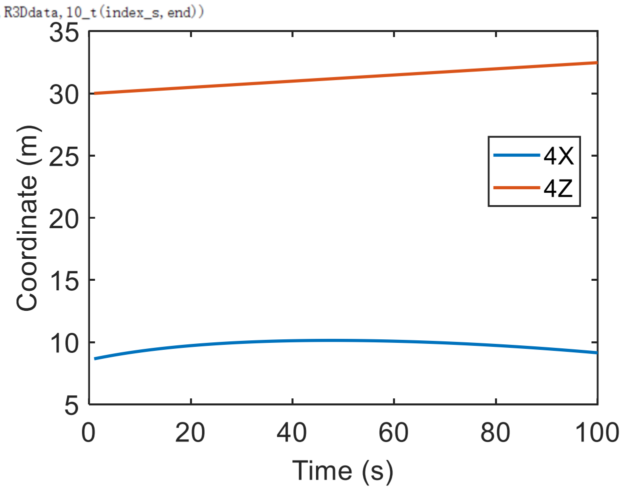


Figure 4 Nodal coordinate in static analysis

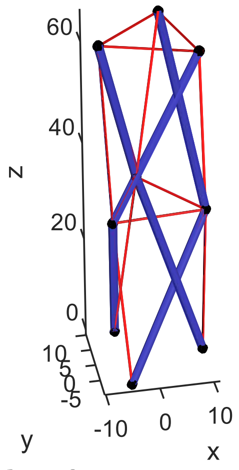
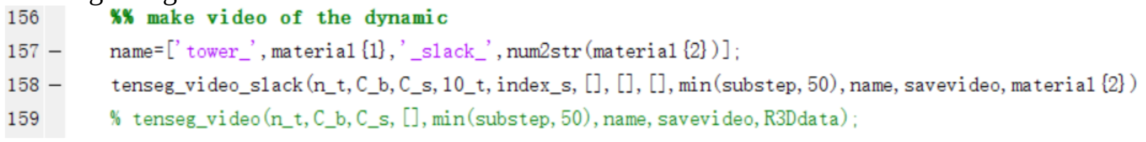


Figure 5 Final configuration in static analysis

4.8 Make video

The picture of structure configuration in every time step of static analysis is used to make a video. The function [tenseg_video_slack] makes videos for the structure considering the slack of strings, in which slacking strings are plotted by a catenary. The function [tenseg_video] makes videos without plotting slacking strings.



```
%% make video of the dynamic
name=['tower_',material{1},'_slack_',num2str(material{2})];
tenseg_video_slack(n_t,C_b,C_s,l0_t,index_s,[],[],[],min(substep,50),name,savevideo,material{2})
% tenseg_video(n_t,C_b,C_s,[],min(substep,50),name,savevideo,R3Ddata);
```

4.9 Exit System

Click on Exit to close MATLAB.

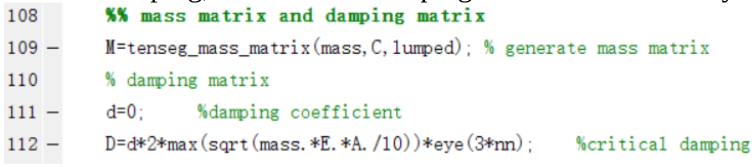
5.0 INSTRUCTION FOR TENSEGRITY DYNAMICS

5. INSTRUCTION FOR TENSEGRITY DYNAMICS

The structure modeling, boundary constraints, prestress design, cross-sectional area design for dynamic analysis are the same as that for static analysis. The difference between statics and dynamics is the velocity and acceleration of nodes need to be considered; thus, the inertia force and damping force should be considered.

5.1 Calculate mass matrix and damping matrix

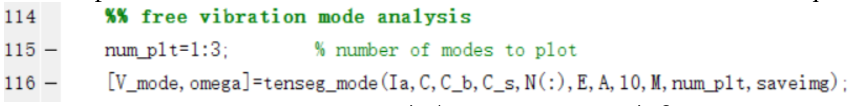
The mass matrix of the tensegrity structure is calculated by function [tenseg_mass_matrix], the input information is the mass of each member, the connectivity matrix, and the lumped coefficient. Lumped mass matrix or consistent mass matrix are given with 'lumped' equals 1 or 0, respectively. Damping matrix is viscous damping, in which the damping force exerted in every node is proportional to velocity.



```
%% mass matrix and damping matrix
M=tenseg_mass_matrix(mass,C,lumped); % generate mass matrix
% damping matrix
d=0;      %damping coefficient
D=d*2*max(sqrt(mass.*E.*A./10))*eye(3*nn);     %critical damping
```

5.2 Free vibration analysis

The undamped free vibration mode and natural frequency are calculated by the function [tenseg_mode]. The input information 'num_plt' is a vector indicating which free vibration mode shape picture to output. The output includes vibration mode matrix 'V_mode' and natural frequency 'omega.'



```
%% free vibration mode analysis
num_plt=1:3;          % number of modes to plot
[V_mode,omega]=tenseg_mode(Ia,C,C_b,C_s,N(:),E,A,10,M,num_plt,saveimg);
```

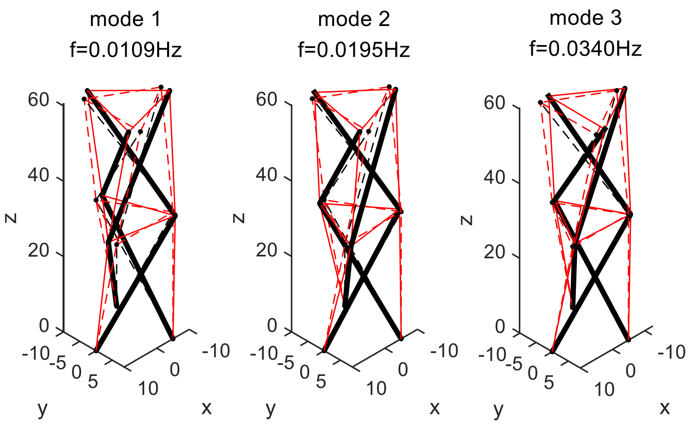
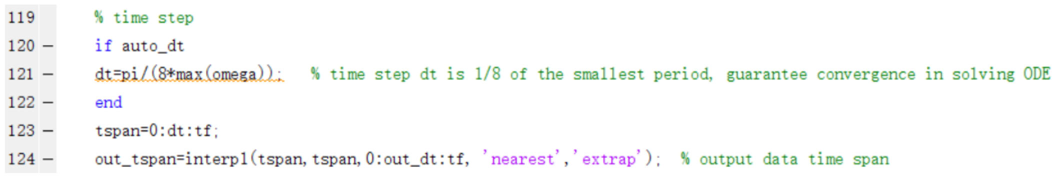


Figure 6 Free vibration mode and natural frequency

5.3 Prepare for Simulation

5.3.1 Timestep

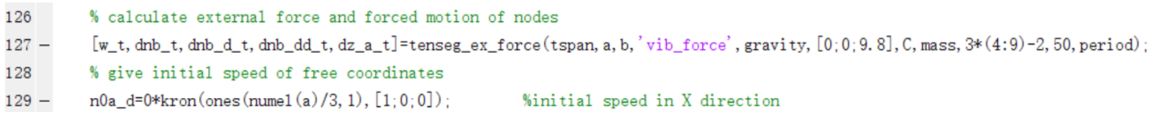
The time step of dynamic simulation should be less than $1/10$ of minimal vibration, period to avoid divergence of dynamic simulation. If 'auto_dt' is 1, the time step is automatically given; otherwise, the time step is given by 'dt.' The 'out_dt' is the time step for recording the output information in dynamic simulation.



```
% time step
if auto_dt
dt=pi/(8*max(omega));   % time step dt is 1/8 of the smallest period, guarantee convergence in solving ODE
end
tspan=0:dt:tf;
out_tspan=interp1(tspan,tspan,0:out_dt:tf, 'nearest','extrap');  % output data time span
```

5.3.2 External force and boundary constraints

The external force, motion, velocity, and acceleration of boundary nodes in every time step is given in ‘w_t’,’dnb_t’, ’dnb_d_t’, ’dnb_dd_t’ respectively by function [tenseg_ex_force]. The initial velocity of the free nodal coordinate is given in 'n0a_d'. Note that the above function [tenseg_ex_force] is used to give external force and boundary constraints in the earthquake, and other self-defined functions can also be used in a different situation

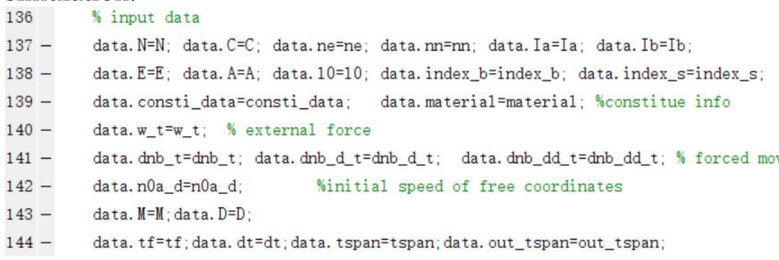


```
% calculate external force and forced motion of nodes
[w_t,dnb_t,dnb_d_t,dnb_dd_t,dz_a_t]=tenseg_ex_force(tspan,a,b,'vib_force',gravity,[0;0;9.8],C,mass,3*(4:9)-2,50,period);
% give initial speed of free coordinates
n0a_d=0*kron(ones(numel(a)/3,1),[1;0;0]);        %initial speed in X direction
```

5.4 Perform the Simulation

5.4.1 Input data

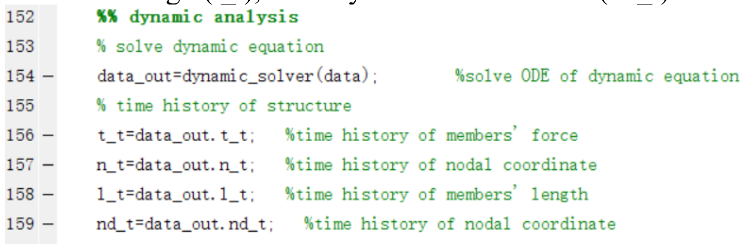
The information of structure and external force is stored in the structure 'data' as input of the dynamic simulation.



```
% input data
data.N=N; data.C=C; data.ne=ne; data.nn=nn; data.Ia=Ia; data.Ib=Ib;
data.E=E; data.A=A; data.l0=l0; 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; data.dnb_d_t=dnb_d_t;  data.dnb_dd_t=dnb_dd_t; % forced movement of pinned nodes
data.n0a_d=n0a_d;        %initial speed of free coordinates
data.M=M;data.D=D;
data.tf=tf;data.dt=dt;data.tspan=tspan;data.out_tspan=out_tspan;
```

5.4.2 Dynamic simulation

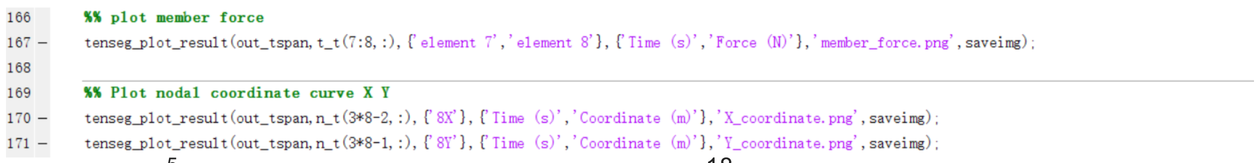
The dynamic simulation is performed by function [dynamic_solver]. The ordinary differential equation is solved by Runge-Kutta method. The output information includes member force(t_t), nodal coordinate(n_t), members’ length(l_t), velocity of nodal coordinate(nd_t) in every sub step.



```
%% dynamic analysis
% solve dynamic equation
data_out=dynamic_solver(data);        %solve ODE of dynamic equation
% time history of structure
t_t=data_out.t_t;   %time history of members' force
n_t=data_out.n_t;   %time history of nodal coordinate
l_t=data_out.l_t;   %time history of members' length
nd_t=data_out.nd_t;   %time history of nodal coordinate
```

5.5 Results Analysis

By clicking the "run" button in MATLAB, we can obtain the static analysis result in 'data_out.' The results of member force, nodal coordinate, and final configuration are plotted by the following function.



```
%% plot member force
tenseg_plot_result(out_tspan,t_t(7:8,:),{'element 7','element 8'},{'Time (s)','Force (N)'},'member_force.png',saveimg);

%% Plot nodal coordinate curve X Y
tenseg_plot_result(out_tspan,n_t(3*8-2,:),{'8X'},{'Time (s)','Coordinate (m)'},'X_coordinate.png',saveimg);
tenseg_plot_result(out_tspan,n_t(3*8-1,:),{'8Y'},{'Time (s)','Coordinate (m)'},'Y_coordinate.png',saveimg);
```

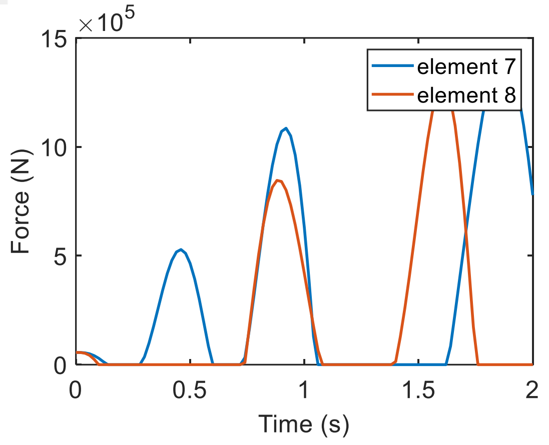


Figure 7 Member force in dynamic analysis

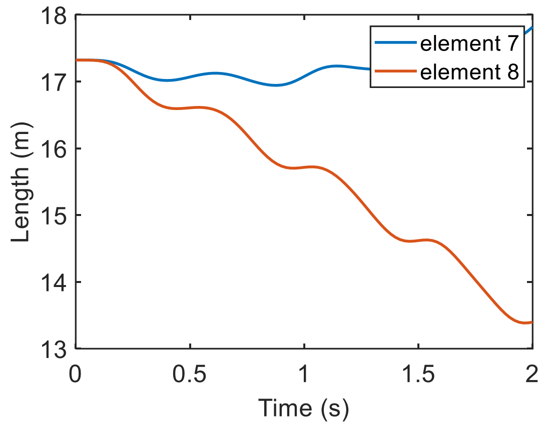


Figure 8 Member length in dynamic analysis

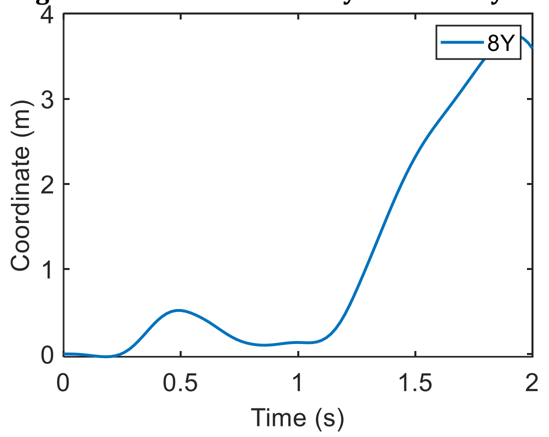


Figure 9 Nodal coordinate in dynamic analysis

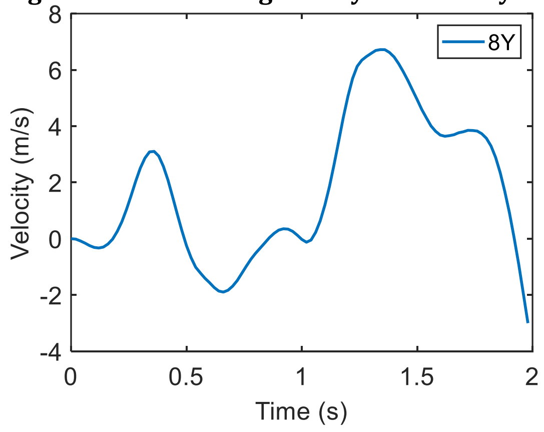


Figure 10 Nodal velocity in dynamic analysis

5.6 Exit System

Click on Exit to close MATLAB.

6.0 APPENDIX

6. APPENDIX

6.1 Verification of Statics Calculation

This section verifies the results obtained from the software's statics calculation. Here, we implement the same D-Bar structure described in Section 4 to check the theoretical solution and the simulation results.

6.1.1 Tensegrity Statics Theory

From statics, we have the following equation,

$$\boldsymbol{Kn} = \mathbf{w},$$

where $\boldsymbol{K} = \left(\boldsymbol{C}^T\widehat{\boldsymbol{q}}\mathbf{C}\right)\otimes\boldsymbol{I}_3$ is the stiffness matrix, $\mathbf{C}$ is the connectivity matrix, and $\boldsymbol{q}$ is the force densities vector containing the force density of bars $\boldsymbol{\lambda}$ and strings $\boldsymbol{\gamma}$. The static equation can also be written linearly with force density as

$$\boldsymbol{A_1 q} = \boldsymbol{w},$$

where $\boldsymbol{A_1} = (\boldsymbol{C}^T \otimes \boldsymbol{I}_3)b.d.(\boldsymbol{B})$ is the equilibrium matrix with force density as the variable.
Consider minimum mass subject to yielding and buckling condition, and assume same material for bars and strings,

$$M = \frac{\rho_s}{\sigma_s}\sum\gamma_i\left|\left|s_i\right|\right|^2 + \max\left(\frac{\rho_b}{\sigma_b}\sum\lambda_i\left|\left|b_i\right|\right|^2\ ,\ \sum 2\rho_b\lambda_i^{\frac{1}{2}}\left(\frac{||b_i||^5}{\pi E_b}\right)^{\frac{1}{2}}\right),$$

where ρ_b, ρ_s, σ_b, σ_s, E_b, s_i, b_i are density, yield strength, Young's modulus, and length of the members.

6.1.2 Analytical Solution

Take Aluminum ($\rho_b = \rho_s = 2700\ kg/m^3, \sigma_b = \sigma_s = 110e06\ Pa, E_b = 60e09\ Pa$) and $\mathrm{F} = 10{,}000$ N to calculate the mass of each member, as shown in Figure 11.

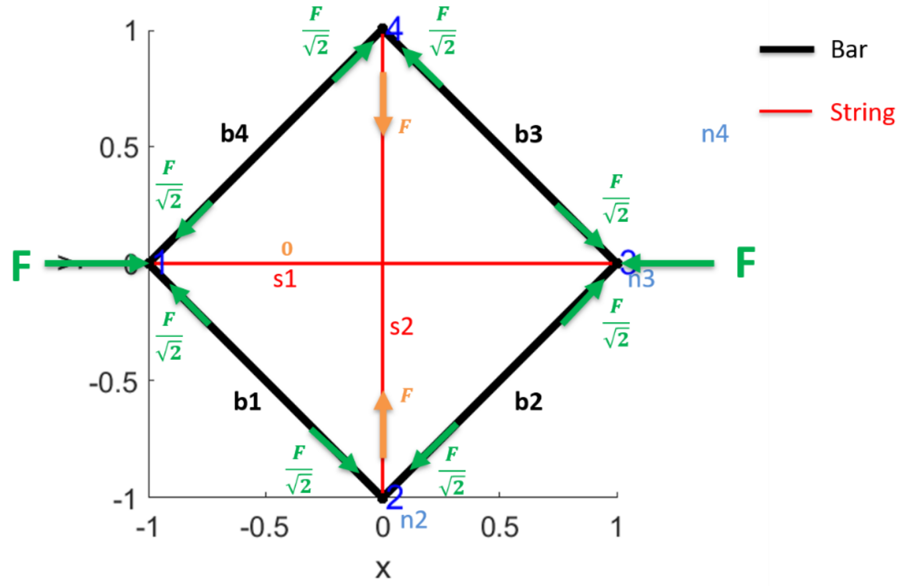


Figure 11 Analytical solution of D-Bar structure

According to the mass of one bar subject to yield,

$$M_{bY} = \frac{\rho_b}{\sigma_b}\lambda\left||b|\right|^2 = \frac{\rho_b}{\sigma_b}\frac{f_b}{||b||}\left||b|\right|^2 = \frac{\rho_b}{\sigma_b} f\left||b|\right|,$$

where $\mathrm{f_b} = \frac{\mathrm{F}}{\sqrt{2}}$ and $b = \sqrt{2}$. We can obtain the mass of one bar subject to yield: $M_{bY} = 0.2455$ kg.
According to the mass of one bar subject to buckling,

$$M_{bB} = 2\rho_b\lambda^{\frac{1}{2}}\left(\frac{||b||^5}{\pi E_b}\right)^{\frac{1}{2}} = 2\rho_b\left(\frac{f_b}{||b||}\right)^{\frac{1}{2}}\left(\frac{||b||^5}{\pi E_b}\right)^{\frac{1}{2}} = 2\rho_b\left(\frac{f_b}{\pi E_b}\right)^{\frac{1}{2}}\left||b|\right|^2.$$

We obtain the mass of one bar subject to buckling: $M_{bB} = 2.0918$ kg.
Buckling happens prior to yield, so the mass of one bar is subject to yield, and buckling is $\boldsymbol{M_b = 2.0918}$ **kg**.
According to the mass of one string subject to yield,

$$M_{sY} = \frac{\rho_b}{\sigma_b}\gamma\left||s|\right|^2 = \frac{\rho_b}{\sigma_b}\frac{f_s}{||s||}\left||s|\right|^2 = \frac{\rho_b}{\sigma_b} f_s\left||s|\right|,$$

where $\mathrm{f_s} = F$ and $s = 2$. We obtain the mass of one string subject to yield: $M_{sY} = 0.4909$ kg.
Mass of string 1 is $\mathbf{0}$, the mass of string 2 is $\boldsymbol{M_s = 0.4909}$ **kg**.

6.1.3 Results Analysis

Simulation results match well with the analytical solution, indicating the accuracy of the software.

6.2Verification of Dynamics Calculation

This section implements the dynamics of a Double pendulum (a simple multibody system) to verify the analytical solution and the simulation results, as well as a comparison with rigid body dynamics.

6.2.1 Double Pendulum Configuration

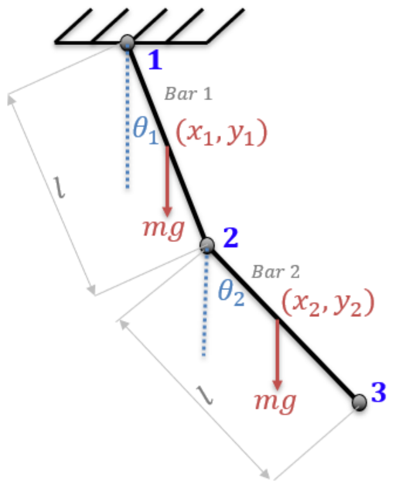


Figure 12 Configuration of a double pendulum.

We are interested in checking the nodal history of the two bars of the double pendulum to verify the results obtained from the TsgFEM.

6.2.2 Dynamics of the Double Pendulum

From geometric properties, one can get,

$$\mathrm{x}_1 = \frac{l}{2}\sin(\theta_1)\,, y_1 = -\frac{l}{2}\cos(\theta_1),$$

$$x_2 = l(\sin(\theta_1) + \frac{1}{2}\sin(\theta_2))\,, y_2 = -l(\cos(\theta_1) + \frac{1}{2}\cos(\theta_2)).$$

Define $\mathrm{L} = \mathrm{T} - \mathrm{V}$, where T and V are kinetic energy and potential energy of the system, then:

$$\mathrm{L} = \frac{m}{2}\left(\dot{x_1}^2 + \dot{y_1}^2 + \dot{x_2}^2 + \dot{y_2}^2\right) + \frac{1}{2}I\left(\dot{\theta_1}^2 + \dot{\theta_2}^2\right) - mg(y_1 + y_2),$$

Where $\mathrm{I} = \frac{1}{12}ml^2$ is the moment of inertia about the center of mass of the bar. Using Lagrange's Equation, we get:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = 0,$$

where q is the generalized coordinate (for this example, q is θ_1 *and* θ_2), $\mathrm{m}_1 = \mathrm{m}_2 = 1kg$, $\mathrm{b}_1 = \mathrm{b}_2 = 1m$, then we get,

$$8\ddot{\theta}_1 + 3\ddot{\theta}_2\cos(\theta_1 - \theta_2) + 3\dot{\theta_2}^2\sin(\theta_1 - \theta_2) + 9\frac{g}{l}sin(\theta_1) = 0,$$

$$2\ddot{\theta}_2 + 3\ddot{\theta}_1\cos(\theta_1 - \theta_2) - 3\dot{\theta_1}^2\sin(\theta_1 - \theta_2) + 3\frac{g}{l}\sin(\theta_2) = 0.$$

Let $\theta_1 = \theta_2 = \pi/4,\ g = 9.8\,m/m^2$. Time step 0.01s, and simulation time 10s, by solving the two odes, we can obtain the history of the two angles.

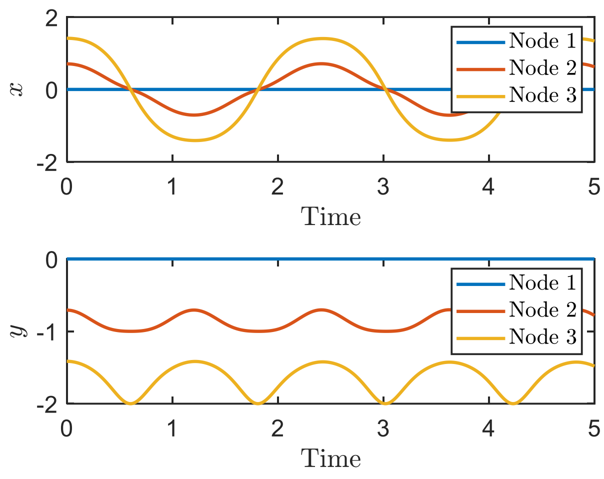


Figure 13 Analytical Solution of X and Y Coordinate Histories of Node 1, 2 and 3

6.2.3 Numerical Solution

Now, let us analyze the results by TsgFEM software. To perform the simulation, one can change the corresponding codes in the examples.

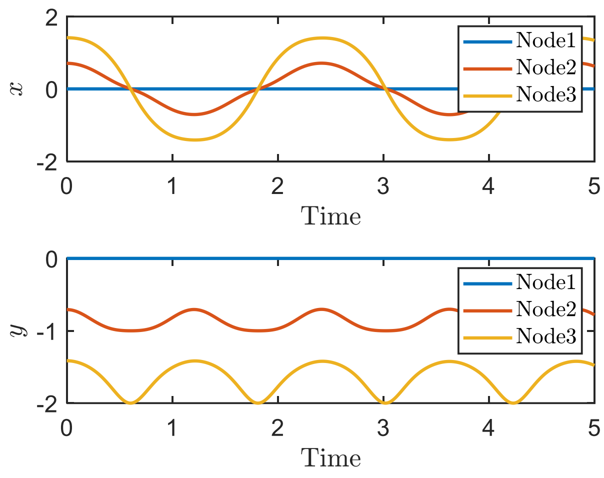


Figure 14 X and Y Coordinate Histories of Node 1, 2, and 3 by TsgFFEM Software

6.2.4 Results Analysis

Simulation results from Figure 14 in Section 6.2.3 match well with the analytical solution from Figure 13 in Section 6.2.2. The error of X and Y Coordinates of Node 2 and 3 by TsgFEM software and analytical results are shown in Figure 15. Figure 16 shows that the bar length changes with simulation time are very small, which indicates that this software gives an accurate solution.

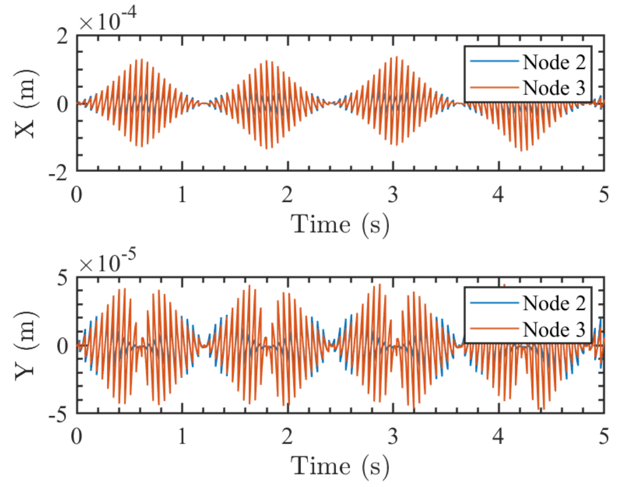


Figure 15 Error of X and Y Coordinates of Node 2 and 3 by TsgFEM Software and Analytical Results

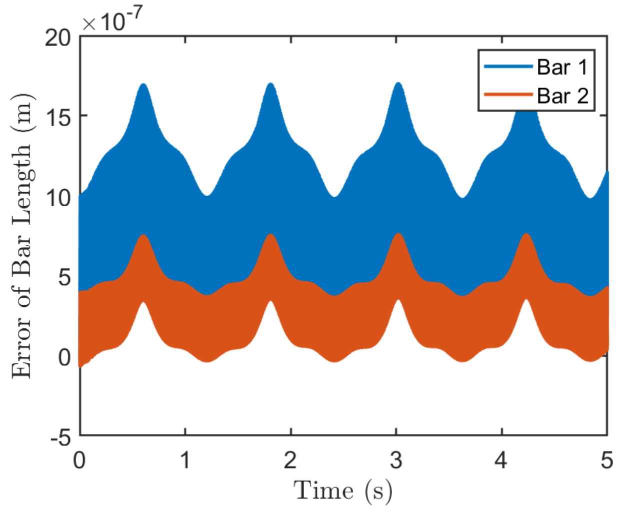


Figure 16 Bar Length Errors of the TsgFEM Software