Technical

MANUAL

***M**odeling **o**f **Te**nsegrity **S**tructures (MOTES) Software*

@ Department of Aerospace Engineering, Texas A&M University

Aug 2019

Revision Sheet

Release No.	Date	Revision Description
Rev. 1.0	02/09/2017	Major functions for statics and dynamics
Rev. 1.1	03/15/2018	1. Statics - Fix any node in any direction. - Minimum mass optimization for yielding constraints. 2. Dynamics - Gives a flag for incorrect initial velocities. - New Plotting functions 3. Bar length corrections and Class-k Constraints
Rev. 1.2	12/13/2018	Master Code Revised Statics - Minimum mass optimization for both yielding and buckling constraints Dynamics - Tensegrity Test Examples
Rev. 1.3	02/01/2019	User's Manual Created
Rev. 2.0	08/05/2019	Document reorganized according to The Journal of Open Source Software (JOSS) reviews.
Rev. 2.1	08/16/2019	JOSS paper modified according to The Journal of Open Source Software (JOSS) reviews.
Rev. 2.2	08/24/2019	Software modified according to The Journal of Open Source Software (JOSS) reviews.

Modeling of Tensegrity Structures (MOTES)

Software Information

Software Goals:

The purpose of this software is to facilitate the modeling and analysis of Tensegrity systems.
The software allows to minimize the mass of structure at the equilibrium condition (statics) and to perform the dynamics simulation of any tensegrity systems.

- **Modeling**: Tensegrity is a network of compressive and tensile members, where the compressive members are connected by tension members (strings, cables, tendons) forming a stable system. This software aims at generating any tensegrity structure Class 0 (the structure is pure tensile members), Class 1 (no compressive members touch each other), Class-k (k compressive members connected to a node).
- **Statics**: Minimal mass calculation and analysis for any tensegrity structure.
- **Dynamics**: Multibody dynamics simulation for any tensegrity structure.

MOTES Members and Information:

Raman Goyal
ramaniitrgoyal92@tamu.edu, Ph.D. Student, Department of Aerospace Engineering, Texas A&M University, College Station, Texas, USA.
Muhao Chen
muhaochen@tamu.edu, Ph.D. Student, Department of Aerospace Engineering, Texas A&M University, College Station, Texas, USA.
Manoranjan Majji
mmajji@tamu.edu, Assistant Professor, Director of LASR Laboratory, Department of Aerospace Engineering, Texas A&M University, College Station, Texas, USA.
Robert E. Skelton
bobskelton@tamu.edu, 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.

Acknowledgement:

MOTES members would like to thank NASA NIAC Phase II grant, Prof. Edwin Peraza Hernandez (Department of Mechanical and Aerospace Engineering, UC Irvine), Dr. Maziar Izadi, Mr. James Henrickson, and students from Integrating Structure and Control Design Group for their help and suggestions during the development of this software.

USER'S MANUAL

TABLE OF CONTENTS

Page #

1.0 GENERAL INFORMATION

A. GENERAL INFORMATION

1.1 System Overview

Undergraduate linear algebra and some basic knowledge of MATLAB is required to understand the codes well. This software is developed based on:

- 64-bit Windows
- MATLAB

Note: Win7/Win10/Mac OS/Linux/Win XP/Win Vista compatible with a MATLAB version later than 2009a should work fine. 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).

1.2 Project References

Modeling of Tensegrity Structures (MOTES) software is created based on the theory developed in the following references.

[1] Goyal, Raman, and Robert E. Skelton. "Tensegrity system dynamics with rigid bars and massive strings." Multibody System Dynamics (2019): 1-26.
[2] Skelton, Robert E., and Mauricio C. de Oliveira. Tensegrity systems. Vol. 1. New York: Springer, 2009.
[3] Raman Goyal and Robert E. Skelton. Dynamics of class 1 tensegrity systems including cable mass, Earth and Space (2018), Page: 868.
[4] Cheong, Joono, and Robert E. Skelton. "Nonminimal dynamics of general class k tensegrity systems." International Journal of Structural Stability and Dynamics 15.02 (2015): 1450042.
[5] Skelton, Robert. "Dynamics and control of tensegrity systems." IUTAM symposium on vibration control of nonlinear mechanisms and structures. Springer, Dordrecht, 2005.
[6] Nagase, K., and R. E. Skelton. "Minimal mass tensegrity structures." Journal of The International Association for Shell and Spatial Structures 55.1 (2014): 37-48.

1.3 Authorized Use Permission

1.4 Points of Contact

1.4.1 Information

MOTES focuses on integrating structure and control design using Tensegrity structure. The group focuses on designing the 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. The authors would like to make this as an 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
Raman Goyal: ramaniitrgoyal92@tamu.edu, **Muhao Chen**: muhaochen@tamu.edu.

1.5 Organization of the Manual

User's Manual v2.2.

2.0 SYSTEM SUMMARY

B. 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):

2.2.1 Tensegrity 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 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.
- **Specify Pinned Nodes**: Manually specify pinned nodes.
- **Specify External Force**: Manually specify external forces.
- **Perform the Analysis**: Click MATLAB “run” button to perform the 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 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: ***Goyal, Raman, and Robert E. Skelton. "Tensegrity system dynamics with rigid bars and massive strings." Multibody System Dynamics (2019): 1-26.*** We are open to collaborate in tensegrity research projects and would also like to hear your opinions on both theoretical and practical problems.

3.0 INSTRUCTION FOR TENSEGRITY TOPOLOGY

C. INSTRUCTION FOR TENSEGRITY TOPOLOGY

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

3.1 Specify Structure Configuration

To analyze any tensegrity structure, we suggest the user to draw the sketch and number all the nodes, bars, and strings in any desired manner. For example, to analyze a D-Bar structure shown in Figure 1. We suggest the user to 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 the Section 3.2, 3.3, and 3.4) with respect to structure complexity.
5. Check the structure by plots (in Section 3.5).

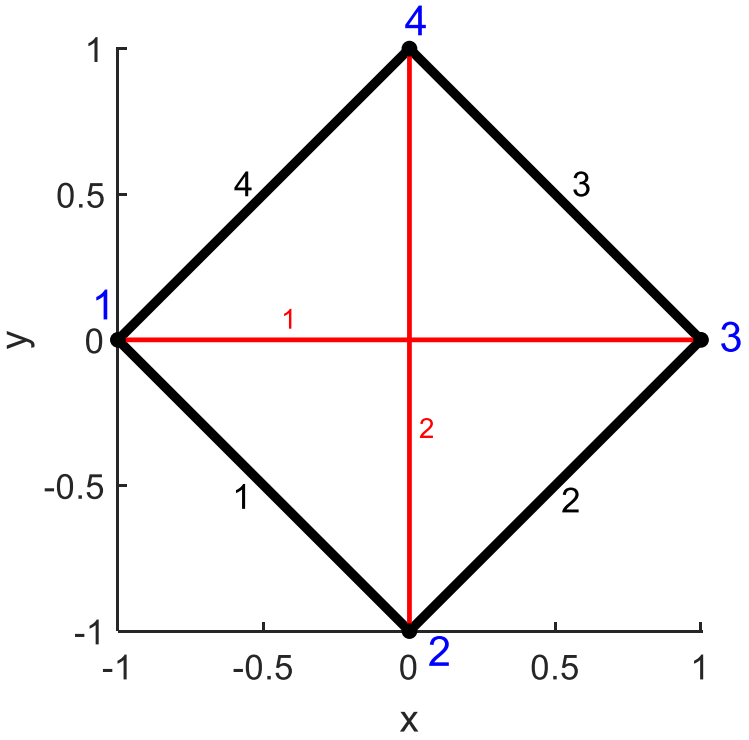


Figure 1. D-Bar Structure Configuration

3.2 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, simply set all z-coordinates to zero. The following line of code defines four node positions. Each column defines a node, and they are in order, node1, node2, node3, and node4.

```
1.  EXAMPLE:
```

```
% Initialization
clear all; clc; close all;
% Manually specify node positions (in 3D).
n1 = [-1 0 0]'; n2 = [0 -1 0]'; n3 = [1 0 0]'; n4 = [0 1 0]';
% Put node vectors in node matrix. Node matrix has to be 3xn for n nodes.
N = [n1 n2 n3 n4]; % N = [node1, node2, node3, node4]
```

3.3 Specify Bar Connectivity

After specifying node positions, bar and string members can be defined in terms of connections between the nodes. There are two ways to determine the connectivity matrix. The two methods are equivalent and depends on the user's choice for modeling structures. However, our suggestion for the new users is to use method 2 (See Chapter 3.3.2). Method 1 (See Chapter 3.3.1) is easier to understand the connectivity relationships and is a better way to go for complex structures.

3.3.1 Write the Connectivity Matrix Directly – Method 1

Now, we already have nodal matrix N, and we know which bar is connected to which two nodes. Thus, for example, we have four bars, bar1 is a vector from nodes 1 to 2, bar2 is a vector from nodes 2 to 3, bar3 is a vector from nodes 3 to 4, and bar4 is a vector from nodes 4 to 1. B = [bar1, bar2, bar3, bar4], according to $B = NC_b^T$, then we can write the connectivity matrix directly.

```
EXAMPLE:
% Manually specify connectivity indices.
C_b' = [ -1     0     0     1
         1    -1     0     0
         0     1    -1     0
         0     0     1    -1];
```

Bar 1 is from node 1 to node 2, corresponding to the first column of C_b^T, i.e. the first element -1 is the start node of bar1, the second element 1 is the end node of bar1.

3.3.2 Give Node Relationships to Generate Connectivity Matrix – Method 2

MOTES provides an intuitive method for defining these members: an input matrix is defined by specifying which nodes connects to make a member. Below, four bars are defined. Bar1 is a vector from nodes 1 to 2, bar2 is a vector from nodes 2 to 3, bar3 is a vector from nodes 3 to 4, and bar4 is a vector from nodes 4 to 1. This index notation is converted into a full bar connectivity matrix, C_b, using [tenseg_ind2C].

```
EXAMPLE:
C_b_in = [1 2;
          2 3;
          3 4;
          4 1]; % Bar 1 is from node 1 to node 2, and so on...
% Convert the above matrices into full connectivity matrices.
C_b = tenseg_ind2C(Cb_in,N);
C_b =[-1     1     0    0
       0    -1     1     0
       0     0    -1     1
       1     0     0    -1]; % method 1 and method 2 are equivalent
```

3.4 Specify String Connectivity

This same index connectivity notation can be used to define string members. Similarly, string connectivity also has two methods.

3.4.1 Write the Connectivity Matrix Directly – Method 1

Now, we already have nodal matrix N, and we know the connections between nodes to make a string. Thus, for example, we have two strings, string1 is a vector from nodes 1 to 3, string2 is a vector from nodes 2 to 4. S = [string1, string2], according to $S = NC_s^T$, then we can write the connectivity matrix directly.

```
EXAMPLE:
C_s' = [-1     0
          0    -1
          1     0
          0     1];
```

String1 is from node 1 to node 3, corresponding to the first column of C_s^T, i.e. the first element -1 is the start node of String1, the third element 1 is the end node of String1.

3.4.2 Give Node Relationships to Generate Connectivity Matrix – Method 2

The function tenseg_ind2C can also be applied to string connectivity matrix. Below, two strings are defined: string1 is a vector from nodes 1 to 3, string2 is a vector from nodes 2 to 4. 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.

```
EXAMPLE:
% Manually specify connectivity indices.
C_s_in = [1 3;
          2 4]; % This is indicating that string 1 is the vector from node 1 to node 3, a
nd that bar 2 connects node 2 to node 4.
% Convert the above matrices into full connectivity matrices.
C_s = tenseg_ind2C(Cs_in,N);
C_s = [ -1     0     1     0
          0    -1     0     1]; % method 1 and method 2 are equivalent
```

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.

```
EXAMPLE:
tenseg_plot(N,C_b,C_s);
```

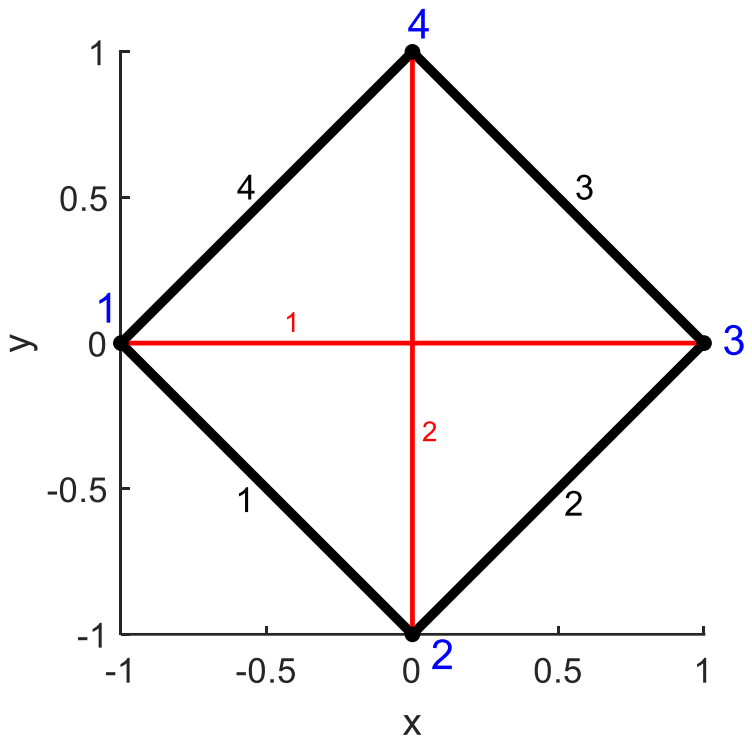


Figure 2 D-Bar Structure (Nodes, bars and strings are in blue, black, and red, respectively.)

3.5.1 Plot Structure Before String Segmentation

To integrate the string mass in the analysis, we distribute the mass of the strings on certain number of point masses along the string. This can be done by breaking each current string member into a set of string segments connected in series. This task is easily performed with [tenseg_string_segment]. In this case, by specifying a single value for `segments`, all string members will be split into three segments. Note that in doing this, it is essential to keep track of the "parent" of each generated string segment. This is useful not only for analysis, but it makes it easier to define parameters for the original parent string members that are inherited by their children string segments. We can again visualize our updated system with [tenseg_plot].

```
EXAMPLE:
segments = 3; % Divide strings into a number of segments
[N,C_b,C_s,parents] = tenseg_string_segment(N, C_b, C_s, segments);
tenseg_plot(N,C_b,C_s); % Plot structure to see what string segmentation looks like
```

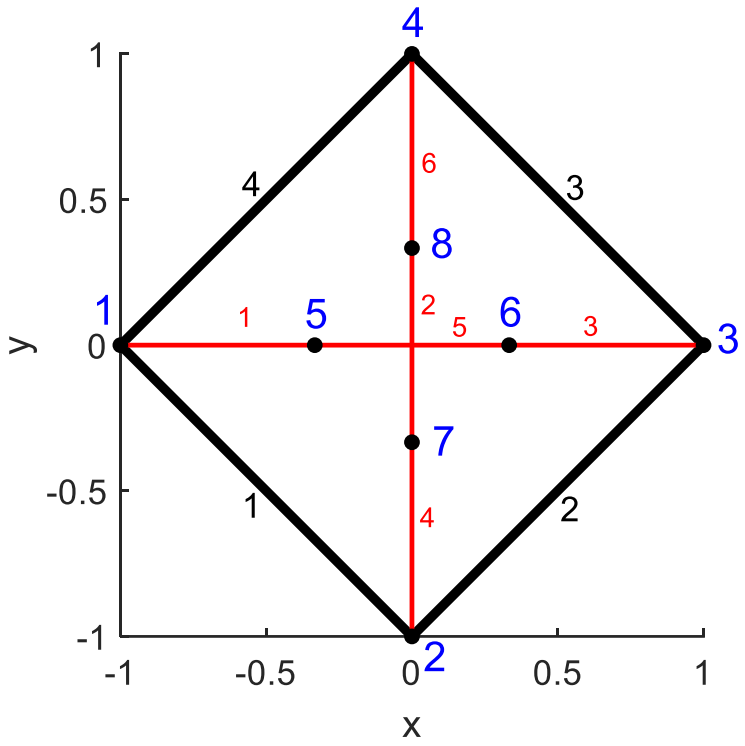


Figure 3 D-Bar Structure, each string is divided into 3 segments (Nodes, bars and strings are in blue, black, and red respectively.)

4.0 INSTRUCTIONS FOR TENSEGRITY STATICS

D. INSTRUCTION FOR TENSEGRITY STATICS

4.1 Specify Pinned Nodes

For static structures, some nodes might be pinned in some directions. We define a matrix Pinned_nodes = [Node_Number, X-direction, Y-direction, Z-direction], values in X-direction, Y-direction, Z-direction for 1 means that direction is fixed, and for 0 means that direction is free.

```
EXAMPLE:
Pinned_nodes = [2 1 1 0]; % Node 2's x, y direction is fixed to the ground
```

4.2 Specify External Force

The external force can be defined at any node in any direction. For example, in this structure, we want to apply $F = 10{,}000\text{N}$ force at nodes 1 and 3 in the x-direction.

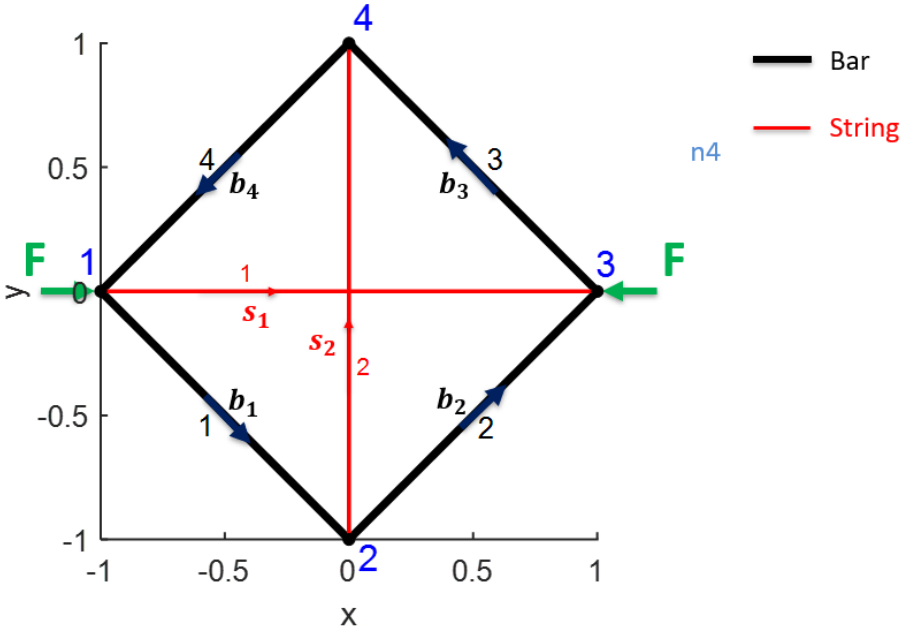


Figure 4 D-Bar Structure (Nodes, bars, strings, and force are in blue, black, red, and green respectively.)

```
EXAMPLE:
% Define the Force Matrix
W = zeros(size(N));
W(1,1) = 10000; W(1,3) = -10000;
```

4.3 Prepare for Analysis

At this point we have given enough information to perform a simulation -- we've defined the structure shape and indirectly assigned some string tensions that should induce motion. To perform a simulation,

we need to create a **simulation task input** data structure, named 'D_Bar' here. This data structure requires, at a minimum, structure definition and external force. These four fields are populated as follows:

```
EXAMPLE:
% Define a tensegrity data structure, named D_Bar
D_Bar.N = N;
D_Bar.C_b = C_b;
D_Bar.C_s = C_s;
D_Bar.W = W;
D_Bar.Pinned_nodes = Pinned_nodes;
D_Bar.bar_material='Aluminum'; % Specify bar material Aluminum, UHMWPE or Steel
D_Bar.string_material='Aluminum'; % Specify bar material Aluminum, UHMWPE or Steel
D_Bar.bar_failure='yielding_n_buckling'; % Bar failure mode 'yielding' or 'yielding_n_b
uckling'
```

4.4 Perform the Analysis

By clicking the "run" button in MATLAB, we can obtain the following results.

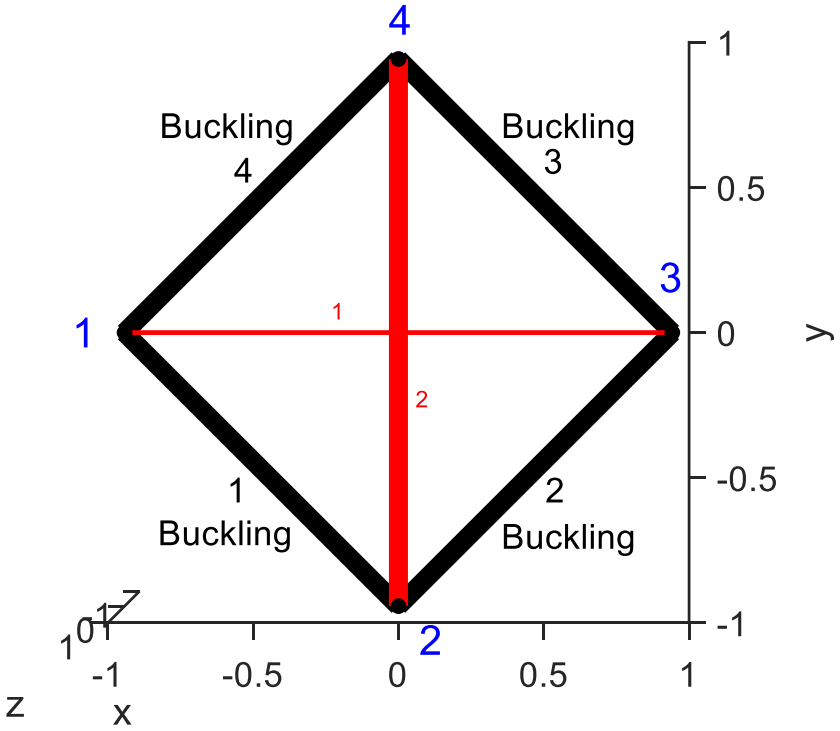


Figure 5 D-Bar Structure Result (Nodes, bars, and strings are in blue, black, and red respectively.)

```
EXAMPLE:
Mass_bar    = 2.0918    2.0918    2.0918    2.0918 kg
Mass_string = 0.0000    0.4909 kg
MIN_MASS    = 8.8580 kg
```

4.5 Exit System

Click on Exit to close MATLAB.

5.0 INSTRUCTION FOR TENSEGRITY DYNAMICS

E. INSTRUCTION FOR TENSEGRITY DYNAMICS

5.1 Specify Pinned Nodes

For dynamic analysis of some structures, nodes might be pinned in some directions. We define a matrix Pinned_nodes = [Node_Number, X-direction, Y-direction, Z-direction], values in X-direction, Y-direction, Z-direction such that 1 means that direction is pinned, 0 means that direction is free. By [tenseg_class_k_convert], the class k structure will convert to class 1 structure mathematically with constraint matrix P, and $N_{new}P = D$.

```
EXAMPLE:
pinned_nodes = [1]; % pinned_nodes
% Convert specified class k structure into a class 1 structure with constraints
[N_new,C_b_new,C_s_new,P,D,node_constraints] = tenseg_class_k_convert(N_simple,C_b,C_s,
pinned_nodes); % convert class k structure to class 1 structure
```

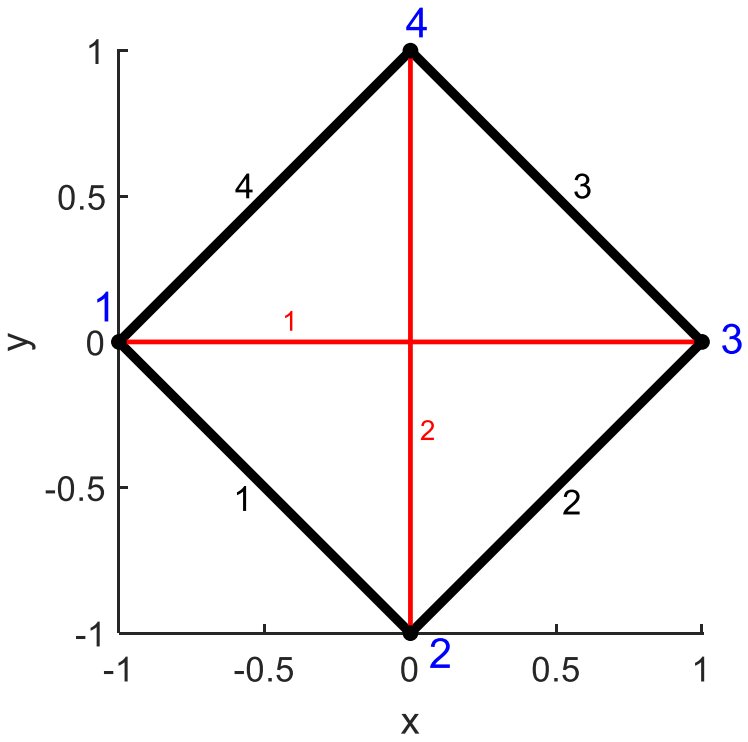


Figure 6 D-Bar Structure (Nodes, bars and strings are in blue, black, and red respectively.)

5.2 Assign Pre-Tension/Prestress in the Strings

The initial condition of the structure would include the rest length of all the strings. This is equivalent to assigning the tension or prestress in the strings. In this example, we do so by arbitrarily assigning rest lengths for each string member. Specifically, we specify these rest lengths as percentages of the strings' given lengths. That is, from N and C_s, we have already defined the initial lengths of our string members.

Specifying string rest length percentages will scale the string rest lengths based on their actual lengths (>=100% would mean there's no tension in the string). To specify these rest length percentages, we create a two-column matrix in which the first column gives string indices and the second column gives the corresponding rest length percentage. In the lines below, we specify string's rest length to be 70% of its current length and then convert that into actual rest lengths using [tenseg_percent2s0].

```
EXAMPLE:
%Specify resting string lengths
% Here, we're setting every string rest length to 70% of its given length
S_0_percent = [(1:size(C_s_new,1))',0.7*ones(size(C_s_new,1),1)]; % percent of initial
lengths
% This function converts those specified percentages into rest lengths
s_0 = tenseg_percent2s0(N_new,C_s_new,S_0_percent);
```

5.4 Prepare for Simulation

At this point we have given enough information to perform a simulation – we have defined the structure shape and indirectly assigned some string tensions that should induce motion. To perform a simulation, we need to create a **simulation task input** data structure, named ‘classK_test’ here. This data structure requires, at a minimum, structure definitions and string rest lengths. These four fields are populated as follows:

```
EXAMPLE:
% Create data structure of system BEFORE segmentation
classK_test.N = N_new;
classK_test.C_b = C_b_new;
classK_test.C_s = C_s_new;
classK_test.P = P; % P gives the constraint matrix
classK_test.D = D; % N_new*P=D
classK_test.s_0 = [1;0.5];
classK_test.tf = 1; % final time
classK_test.dt = .01; % time step
classK_test.video = 0; % 1: generate a video; 0: will not generate a video
```

5.5 Perform the Simulation

By clicking the “run” button in MATLAB, we can obtain the following results.

```
EXAMPLE:
% Perform simulation
[History,debug] = tenseg_sim_classkopen(classK_test);
```

5.6 Results Analysis

By clicking the “run” button in MATLAB, we can obtain the following results.

```
EXAMPLE:
%% Plotting position and velocity of specified node/axes
tenseg_plot_node(History,[1,2],[1,2,3])
tenseg_plot_velocity(History,[1,2],[1,2,3])
%% Create plots with 3D objects
classK_test.Bradius = [0.02; 0.02; 0.02; 0.02]; % Radius of bars [# bars x 1]
classK_test.Sradius = [0.01; 0.01]; % Radius of strings [# strings x 1]
classK_test.Nradius = 0.035*ones(size(classK_test.N,2),1); % Radius of node spheres [#
nodes x 1]
```

```
%% Plotting configurations
number_of_configurations = 3; % Number of configurations to plot
tenseg_plot_configurations(History, classK_test, number_of_configurations)
%% Create animation
tenseg_animation(History,classK_test,[],[],[],[],10)
```

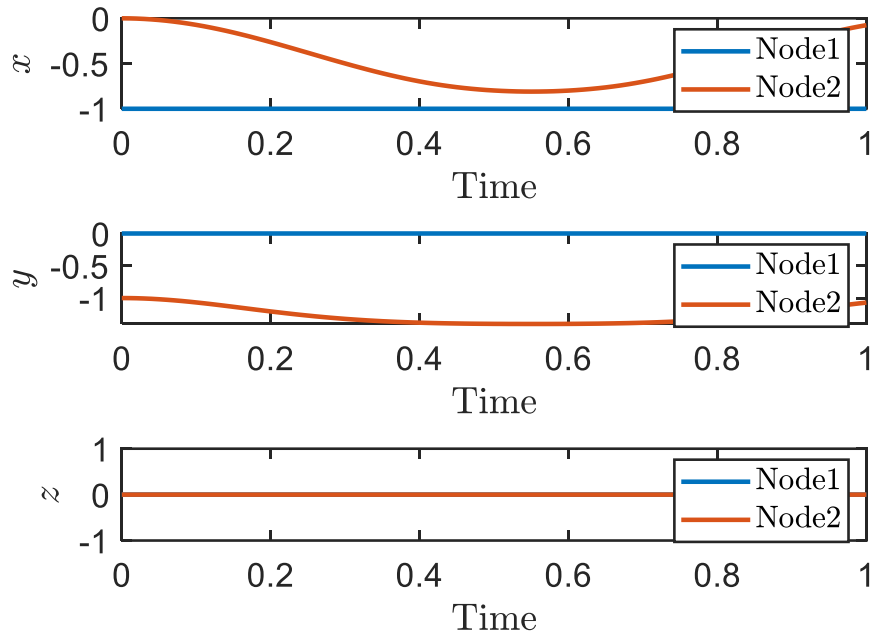


Figure 7 D-Bar Structure Dynamics Simulation Nodes Position History

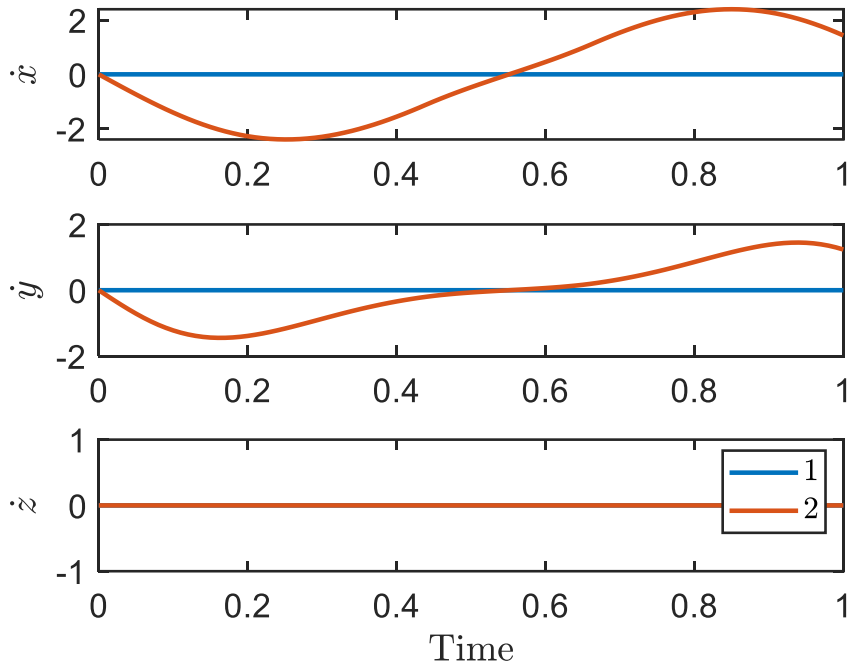


Figure 8 D-Bar Structure Dynamics Simulation Nodes Velocity History

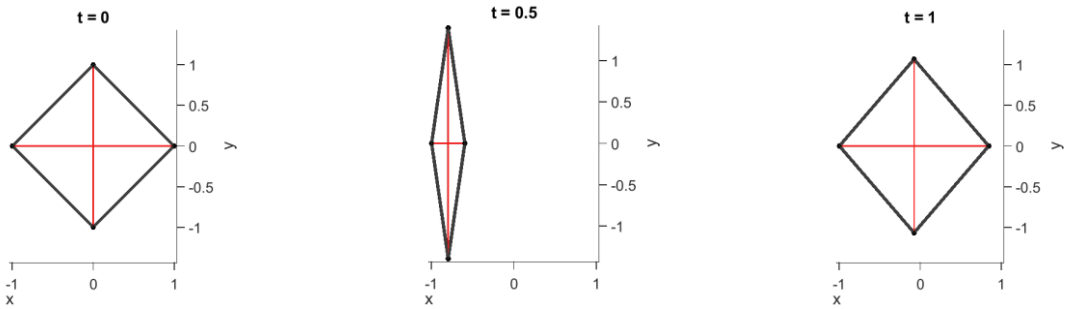


Figure 9 D-Bar Structure Dynamics Simulation Structure Configuration History

5.7 Exit System

Click on Exit to close MATLAB.

6.0 ADVANCED TENSEGRITY SYSTEM APPLICATION

F. ADVANCED TENSEGRITY SYSTEM APPLICATIONS

Tensegrity structures have various applications, especially in space exploration, because of its many advantages, such as mass-saving, deployability, controllability, and many more. This section provides some examples of our research projects developed with the help of the MOTES software. Tensegrity systems are classified into Class 0, Class 1, and Class K structures based on how many bar nodes are connected in the structure configuration. The following selected a few project examples to show their strength.

6.1 Class 0 Structure: DHT for Space Habitat

Mankind has been in space 60 years in the unhealthy environment of zero-g. The goal of this project is to provide artificial gravity and growth strategy to a large spacious habitat as opposed to currently limited space cubicles. The objective of habitat structure is to design a minimum mass cylinder to sustain load due to atmospheric pressure and centrifugal forces while providing the required stiffness for any unexpected change in pressure and consequently force.

6.1.2 DHT Configuration

A DHT is a class-2 tensegrity structure with one set of bars following a clockwise pattern and another set of bars following an anti-clockwise pattern (*Nagase, K., and R. E. Skelton. "Double-helix tensegrity structures." AIAA Journal 53.4 (2014): 847-862.*).

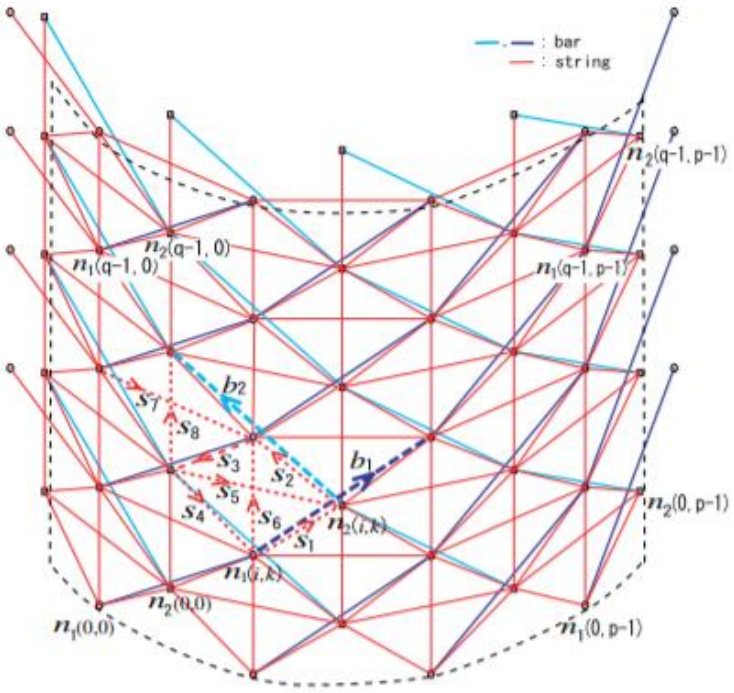


Figure 10 Complete set of DHT units
(p=4 number of points at bottom circle, q=4 number of points along the vertical sides)

6.1.2 Habitat Membrane

The static analysis shows no need for bars in the DHT cylinder as atmospheric pressure would provide enough outward force or in other words, would be working as a compressive member. This structure can provide radial and torsional stiffness to the structure due to all the diagonal strings shown in the structure.

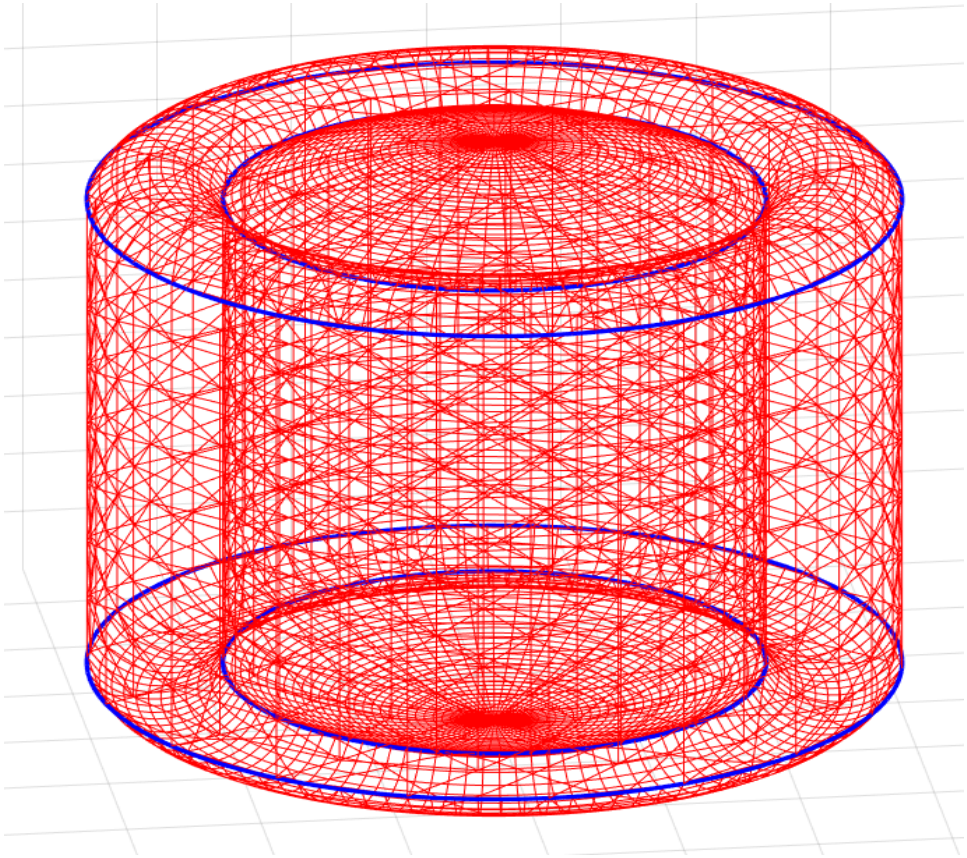


Figure 11 DHT Membrane for Space Habitat

6.2 Class 1 Structure: Lander for Planetary Exploration

For soft landing of many planetary exploratory vehicles, the parachute is not an option due to the absence of adequate atmosphere on that planet. Even though deceleration by rockets can solve the problem, it is very expensive, and the success of the project is also sensitive to landing areas. The tensegrity lander works as an air-bag and offers a good structure in absorbing and distributing impact energy while protecting the payload. The tensegrity lander can be well packed in a small volume in a rocket before landing. Once tensegrity lander lands on the planet, it will bounce, roll, and finally come to rest on the surface of the planet. Then, by actuating the strings of the lander, it can work as a primary mobility system and complete its required task of surface exploration.

6.2.1 Lander Configuration

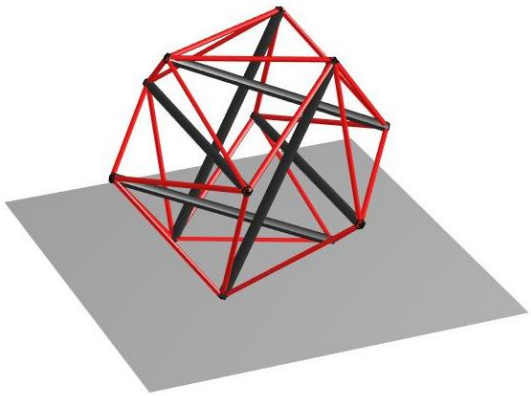


Figure 12 Tensegrity Lander for Planetary Exploration

6.2.2 Results

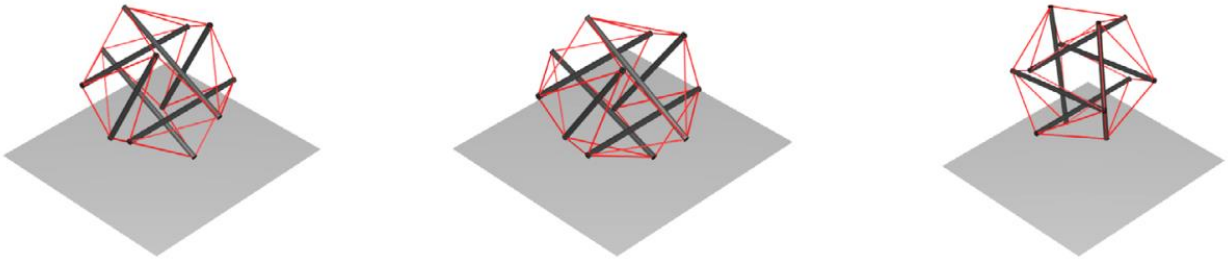


Figure 13 Simulation time-lapse of a tensegrity lander (t = 0.825s, 0.850s, 0.875s)

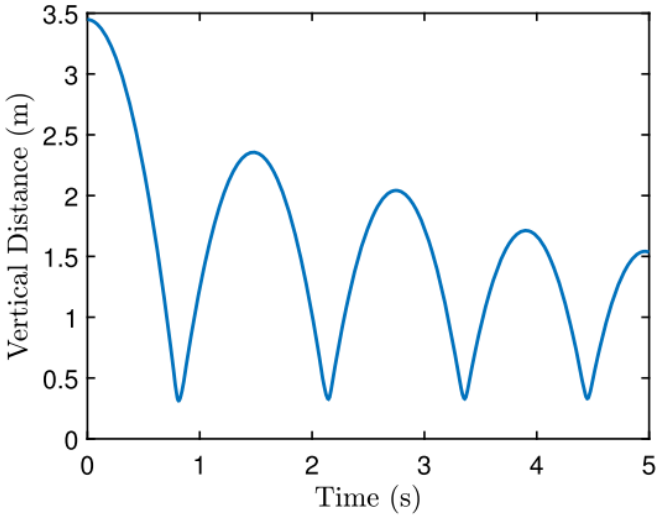


Figure 14 Vertical Distance of Center of Mass from the Ground
(*Goyal, Raman, and Robert E. Skelton. "Tensegrity system dynamics with rigid bars and massive strings." Multibody System Dynamics (2019): 1-26.*)

6.3 Class 1 Structure: Tensegrity plate for Space Telescope

Space telescope is one of the most important devices for astronomical study. A good space telescope should have properties like: lightweight, deployability, focal point adaptability, and precision etc. A tensegrity plate provides a solution to space telescope based on the properties mentioned above. The tensegrity plate can be packed in a small volume in the rocket. After launching into space, it can be spread out and deployed into a paraboloid. Each string works as an actuator, so it is easy to change shape to achieve a certain focal point. The smooth morphing process of tensegrity is changing the equilibrium of the structure instead of pushing the structure against its equilibrium. This provides both minimum mass and minimal energy requirement, a very important aspect in space study. We believe the tensegrity design to be the future for space telescope.

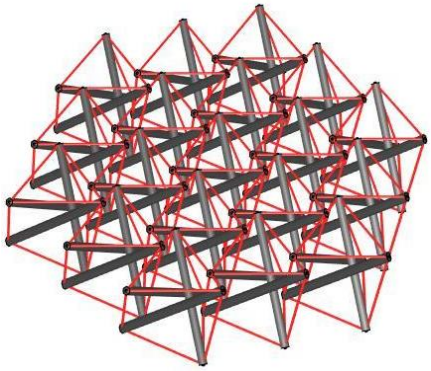


Figure 15 Tensegrity Plate Topology

6.3.1 Space Telescope Configuration

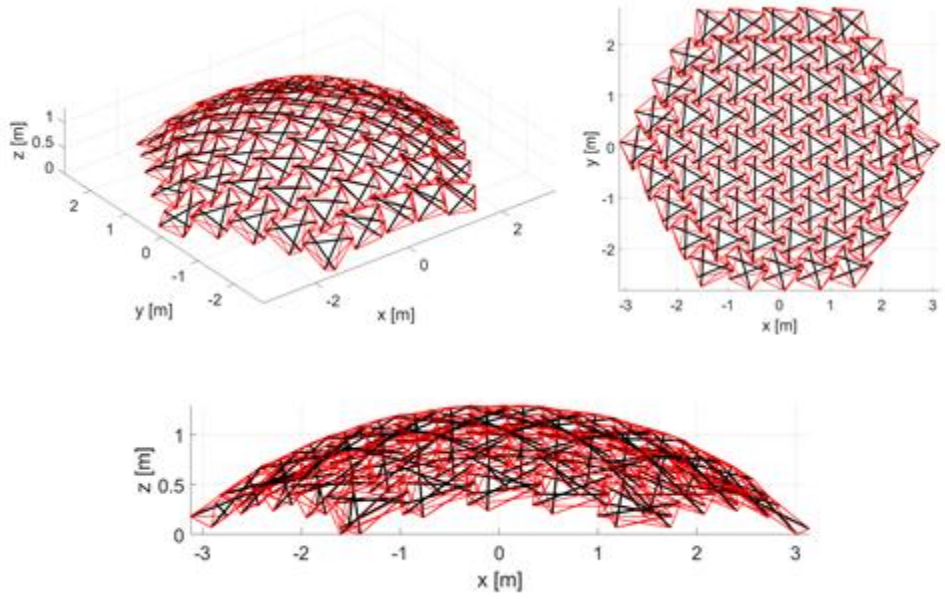


Figure 16 Top and Side View of the Space Telescope

6.3.2 Deployed Shape

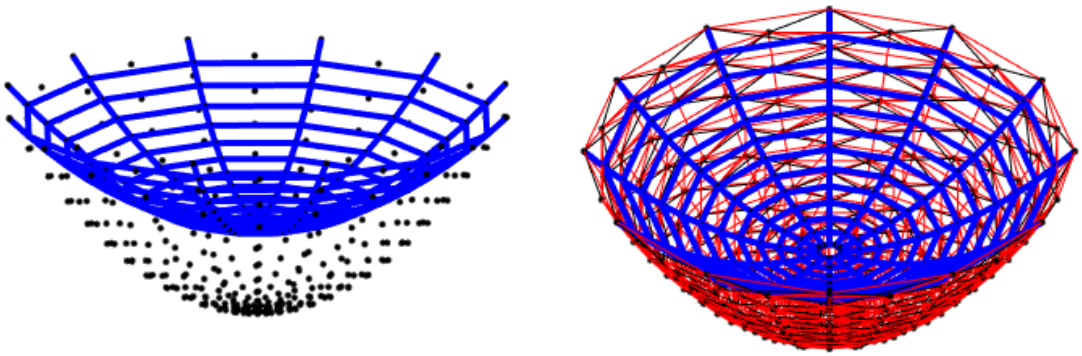


Figure 17 Parabolic Surface and Deployed Version

7.0 APPENDIX

G. APPENDIX

A. 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 verify the theoretical solution and the simulation results.

A.1 Tensegrity Statics Theory

From statics, we have the following equation,

$$NK = W,$$

where $K = C_s^T \hat{\gamma} C_S - C_B^T \hat{\lambda} C_B$, γ, and λ are force densities in bars and strings. Take the k^{th} column of the equation,

$$\widehat{SC_{S_k}}\gamma - \widehat{BC_{B_k}}\lambda = W_k.$$

Write into a matrix form in terms of γ and λ,

$$[\widehat{SC_{S_k}} \quad -\widehat{BC_{B_k}}]\begin{bmatrix}\gamma\\ \lambda\end{bmatrix} = W_k, \begin{bmatrix}\gamma\\ \lambda\end{bmatrix} \geq 0.$$

Stacking all the columns till nth column, we get,

$$\begin{pmatrix}\widehat{SC_s e_1} & -\widehat{BC_b e_1}\\ \widehat{SC_s e_2} & -\widehat{BC_b e_2}\\ \vdots & \vdots\\ \widehat{SC_s e_n} & -\widehat{BC_b e_n}\end{pmatrix}\begin{bmatrix}\gamma\\ \lambda\end{bmatrix} = \begin{bmatrix}\mathrm{W_{e_1}}\\ \mathrm{W_{e_2}}\\ \vdots\\ \mathrm{W_{e_n}}\end{bmatrix},$$

which can be simply written as,

$$A\begin{bmatrix}\gamma\\ \lambda\end{bmatrix} = W_{vec}, \begin{bmatrix}\gamma\\ \lambda\end{bmatrix} \geq 0.$$

Consider minimum mass subject to Yield and buckling condition, and assume same material for bars and strings,

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

where ρ_b, ρ_s, σ_b, σ_s, E_b are density, yield strength, and Young's modulus of the material.

A.2 Analytical Solution

Take Aluminum ($\rho_b = \rho_s = 2700\ kg/m^3, \sigma_b = \sigma_s = 110e06\ Pa, E_b = 60e09\ Pa$) and F = 10,000 N to calculate the mass of each members, see Figure 18.

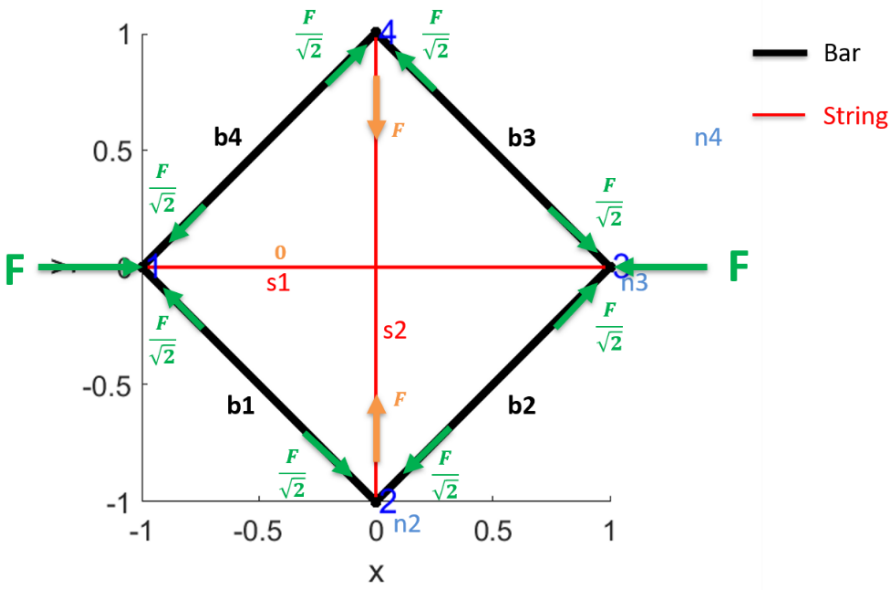


Figure 18 Analytical solution of D-Bar structure

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

According to 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 mass of one bar subject to buckling: $M_{bB} = 2.0918$ kg.

Buckling happens prior of Yield, so mass of one bar subject to Yield and buckling is $\boldsymbol{M_b = 2.0918}$ **kg**.

According to 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 $f_s = F$ and $s = 2$. We obtain mass of one string subject to yield: $M_{sY} = 0.4909$ kg.

Mass of string 1 is **0**, mass of string 2 is $\boldsymbol{M_s = 0.4909}$ **kg**.

A.3 Results Analysis

Simulation results from Section 4.5 matches well with the analytical solution in Section A.2, indicating the accuracy of the software.

B. VERIFICATION 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.

B.1 Double Pendulum Configuration

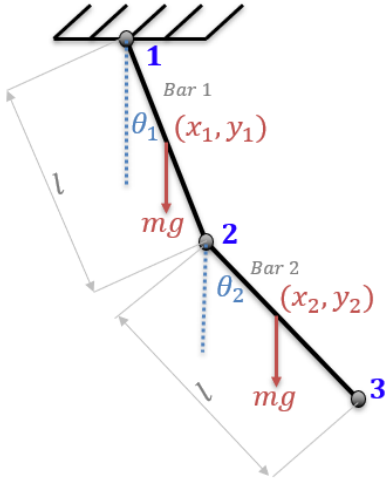


Figure 19 Double Pendulum Configuration

We are interested in checking the nodal history of the two bars of the double pendulum to verify the results obtained from the software. The analytical and simulation results would be given in Section B.2 and B.3.

B.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 moment of inertia about the center of mass of the bar.

Using Lagrange's Equation,

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

where q is the generalized coordinate (for this example q is $\theta_1 \, and \, \theta_2$), $\mathrm{m}_1 = \mathrm{m}_2 = 1kg$, $\mathrm{b}_1 = \mathrm{b}_2 = 1m$, 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.

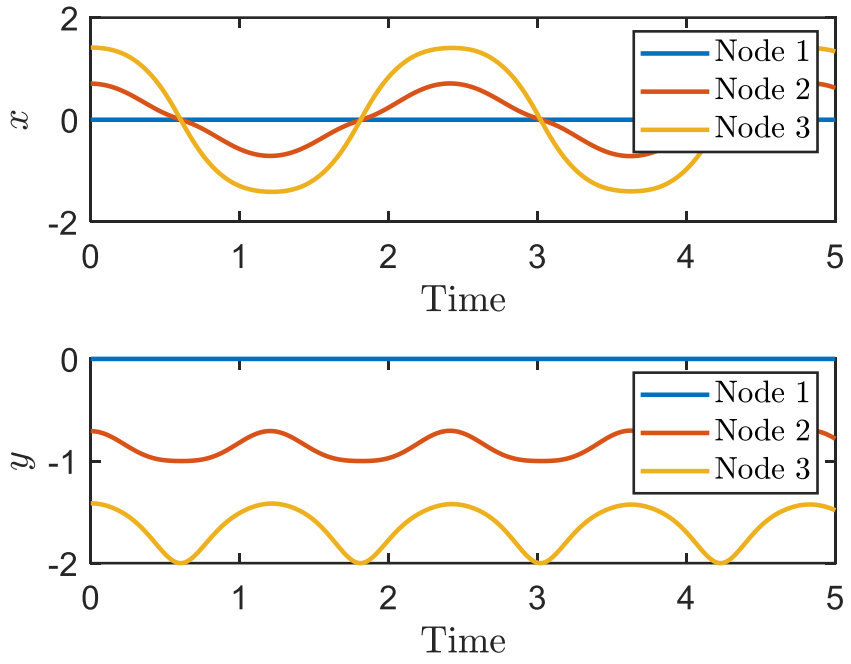


Figure 20 Analytical Solution of X and Y Coordinate Histories of Node 1, 2 and 3

B.3 Numerical Solution

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

```
% Specify the nodes
n1 = [0 0 0]';
n2 = [sqrt(2)/2 -sqrt(2)/2 0]';
n3 = [sqrt(2) -sqrt(2) 0]';
N_simple = [n1 n2 n3];
% Specify bar and string connectivity
C_b_in = [1 2; 2 3];
C_s_in = [1 2]; % at least one string need to be added for this software, we make a mas
    sless string coincides with bar1 such that it doesn't affect the results.
pinned_nodes = [1]; % pinned_nodes
W=zeros(3,length(N_new(1,:)));
W(2,1) = -0.5*9.8;
W(2,2) = -9.8;
W(2,3) = -0.5*9.8; % Consider gravity
classK_test.tf = 10; % Set simulation time
classK_test.dt = .05; % Set time step
% Plot Node x and y coordinates history of 1, 2, and 3
tenseg_plot_node(History,[1 2 3],[1 2])
```

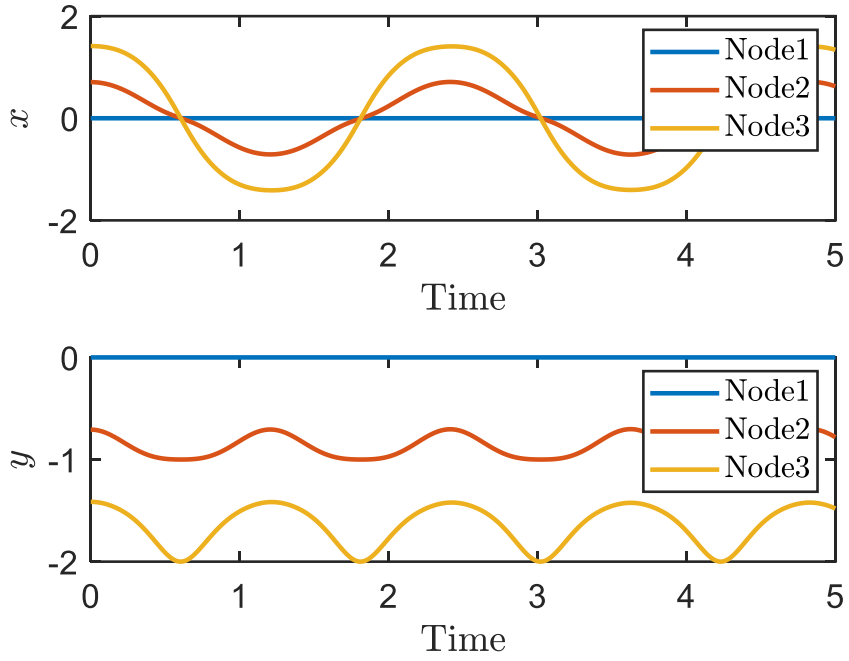


Figure 21 X and Y Coordinate Histories of Node 1, 2 and 3 by MOTES Software

B.4 Results Analysis

Simulation results from Figure 21 in Section B.3 matches well with the analytical solution from Figure 8 in Section B.2. Figure 20 shows the bar length changes with simulation time are very small, which indicates that this software gives an accurate solution.

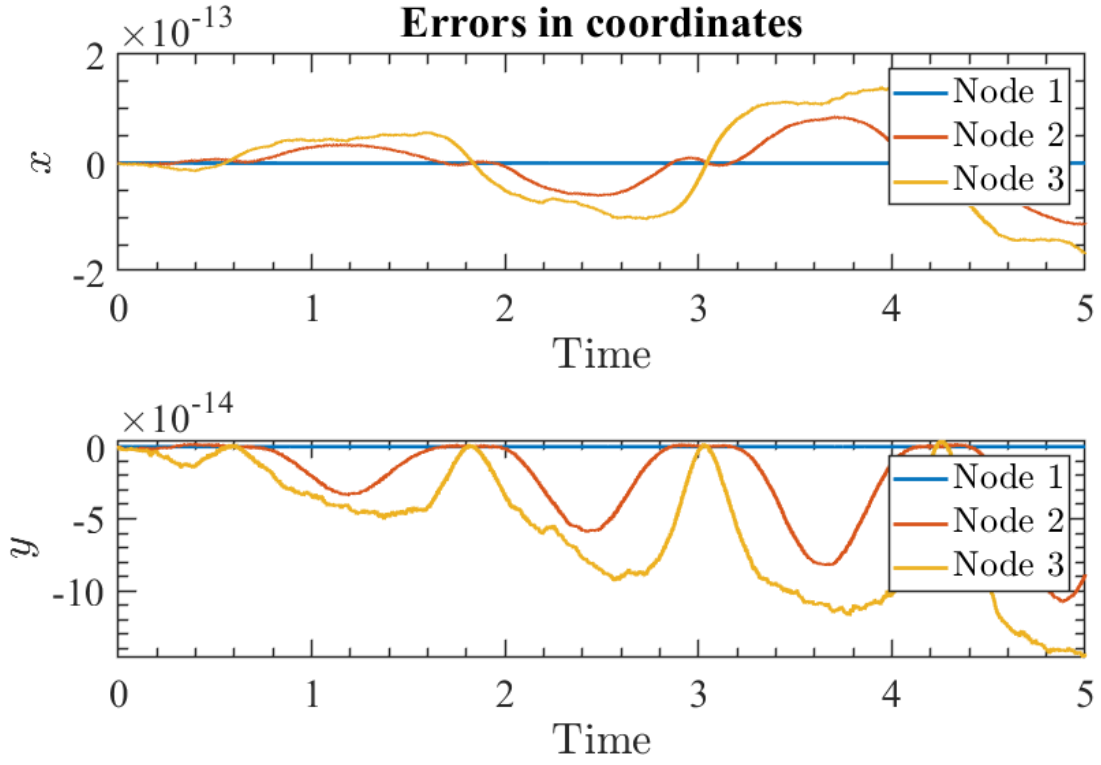


Figure 22 Error of X and Y Coordinates of Node 1, 2 and 3 by MOTES Software and Analytical Results

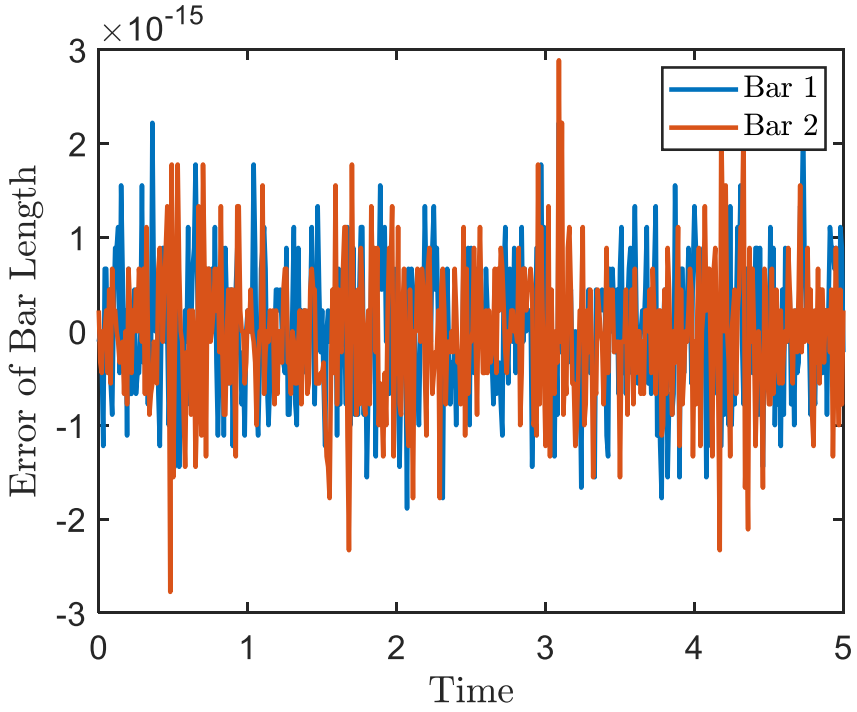
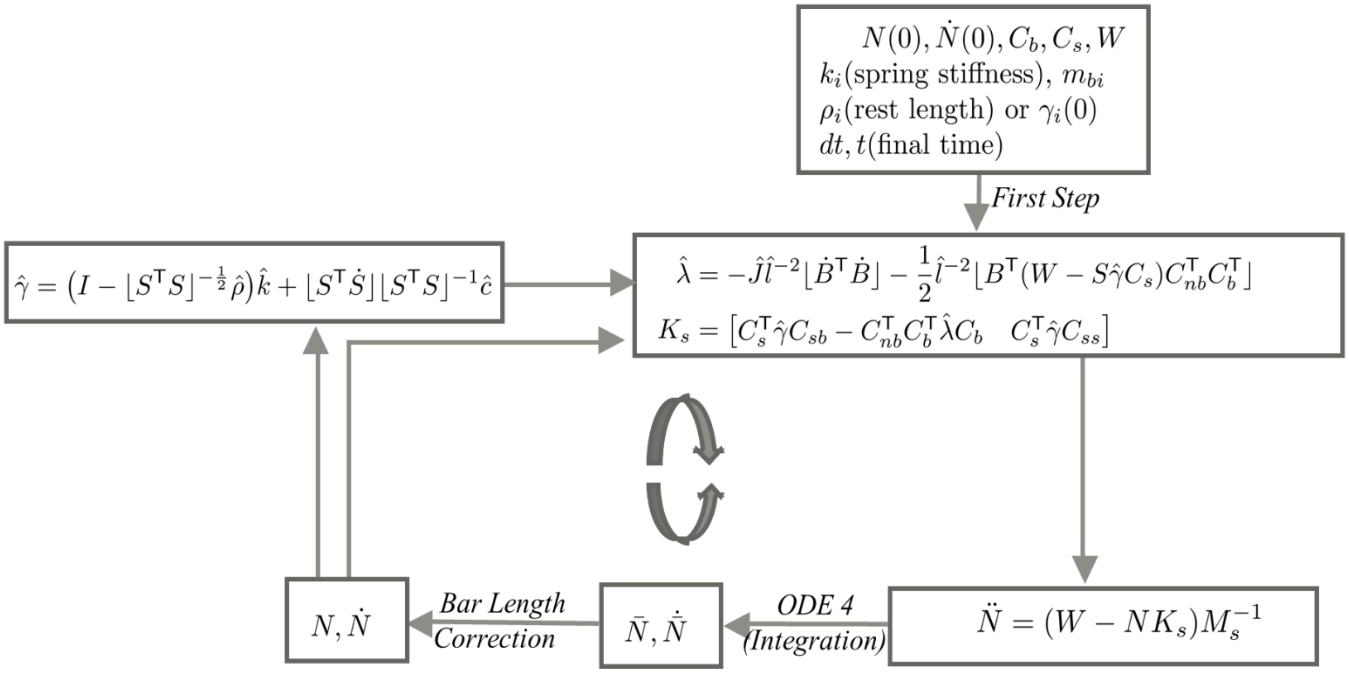


Figure 23 Bar Length Errors of the MOTES Software

C. SOFTWARE WORKFLOW DIAGRAM

C.1 Class-1 Dynamics Calculation Flow Chart



C.1 Class-K Dynamics Calculation Flow Chart