Indian Institute of Technology Palakkad
भारतीय प्रौद्योगिकी संस्थान पालक्काड
Under Ministry of Human Resource Development, Govt. of India
मानव संसाधन विकास मंत्रालय के अधीन, भारत सरकार

ME 3522 – Introduction to Finite Element Methods

Project Report

Problem Statement-

To develop a MATLAB App (GUI, Graphical User Interface) which implements finite element modelling, solution and further processing of 3-D truss structure made of all 2-noded link elements (bar oriented in 3-D).

Members

Divyansh Khare (131701009)
Shubham Wakchaure (131701026)
Mechanical Engineering

Mentor

Rd. K.V.N. Surendra
Mechanical Engineering

Introduction-

The task assigned was to Develop a MATLAB App (GUI, Graphical User Interface) which implements finite element modelling, solution and further processing of 3-D truss structure made of all 2-noded link elements (bar oriented in 3-D).

Approach-

Direct stiffness method was used to model the truss problem.
The following steps were included in the program-

1. User's inputs
    a. Young's modulus
    b. Density
    c. Number of elements
    d. X, Y, and Z coordinates of the nodes
    e. Area of the elements
    f. Declaring the first and second node of each element
    g. Number of nodes on which the forces act
    h. Node numbers on which specific forces act
    i. X, Y, Z components of forces.
    j. Nodes on which the displacement constraints act
2. Calculation and 3-D plot
3. Table view

After the program, all the required commands would be fed in the app through the App designer feature of MATLAB.

Execution-

MATLAB program

1. Recording the Material's properties-
   The initial part of the program simply records the material properties (Young's modulus and density) in two variables, namely 'Modulus' and 'Dens'

```
%% Material Constants

Modulus=input('Enter the Youngs Modulus in MPa\n'); % Entering the Young's Modulus
Dens=input('Enter the Density in Kg/m^3\n'); % Entering the density
```

2. Recording the geometrical features of the truss
   The number of elements is recorded in a variable named 'E' and the coordinates of the nodes in an array named 'N'.

```
%% Geometry of Truss

E=input('Enter the Number of Element\n'); % Entering the number of elements

for i=1:1000
    e=10;
    N(i,1:3)= input('Enter the Node X, Y, and Z location in unit of mm(millimeter) as the [X,Y,Z] format (Enter e to exit/stop)\n'); % Node locations
    if N(i,1:3)==10
        N(i,:)=[];
        break;
    end
end
[mmm,nnn]=size(N); % Essentially recording the number of rows to get the number of nodes, 'nnn' is just a dummy variable.
```

3. Recording the area of the elements
   Two provisions have been made for recording the area of cross section. The first provision allows the user to assign same area to all the elements and records it and the second provision allows the user to input different areas for different elements. The data is stored in an array named 'A'.
4. Recording the first and second nodes of the elements
   The first and second nodes of the elements are recorded to get the exact positions of the elements in an array named 'P'.

```
fprintf('Enter 1- If area of all the truss element should be same\nEnter 2- For different area for each truss element\n');
Sel=input('Enter the choice\n');
if Sel==1
    Area = input('Enter the Area in mm^2\n');
    A=Area*ones(1,E) ;
elseif Sel==2
    for i=1:E
        fprintf('Enter the Area of Element in mm^2 %d\n',i);
        A(i)=input('');
    end
end
P=zeros(2,E);
fprintf('Next you have to enter the node 1 and node 2 of each element\n')
for i=1:E
    P(1,i)=input('Enter the First Node\n');
    P(2,i)=input('Enter the Second Node\n');
end
```

5. Recording the boundary conditions
   a. Forces
      First the number of nodes on which the forces act is recorded in a variable named 'nf'. Then the forces are recorded individually by recording the node number and the components of the force in the three directions (X, Y, Z). The following formula enters the three components of the force acting on the 'nnfth' node in the 'nnfth' section of the force matrix. For example, if a force is acting on the 4th node, then the following formula will enter it in the 4th section of the force matrix with the x component being entered in 10th row, y in the 11th and z in the 12th. 'nnf*3-2' gives the starting row number of the respective node section in the force matrix.

      F((nnf*3)-2:(nnf*3), 1) = F((nnf*3)-2:(nnf*3), 1) + FF

```
%% Boundary Conditions

fprintf('Start with the Forces on the Truss\n');
nf=input('Enter the total number of nodes on which forces act\n');

% Defining the force vector. The force vector 'F' defined below is a matrix
% which has one column and rows, thrice the number of nodes. This has been
% done to accomodate the three components of a force applied on an element.
% In other way, it can be said that the force vector has a row section for
% each element which has further been divided into three sun sections(rows)
% to make space for the three components of the force applied to a
% particular element.

F=zeros(mmm*3,1); % Setting the forces to zero initially
for i=1:nf
    fprintf('Enter the Node number of which force %d acts\n',i);
    nnf=input('');
    FF=input('Enter the x, y, z componenet of force in Newton on the node in [Fx;Fy;Fz] format\n');

    % The following formula enters the three components of the force acting
    % on the nnf_th node in the nnf_th section of the force matrix. For
    % example, if a force is acting on the 4_th node, then the following
    % formula will enter it in the 4_th section of the force matrix with
    % the x component being enterd in the 10_th row, y in the 11_th and z
    % in the 12_th. 'nnf*3-2' gives the starting row number of the
    % respective node secion in the force matrix.

    F((nnf*3)-2:(nnf*3),1)=F((nnf*3)-2:(nnf*3),1)+FF;
end
```

b. Displacement constraints
The total number of nodes on which displacement constraints (displacement is 0) are applied is recorded. A provision has been given for the user to constraint the required nodes in all the three directions separately. The values '0' and '1' are used to represent the constrained and unconstrained condition of the node respectively in the respective direction.

```
B=ones(mmm*3,1);
fprintf('Displacement Contraints on Truss\n');
nb=input('Enter the total number of nodes on which displacement constraints are applied\n');
for i=1:nb
    fprintf('Enter the Node number on which constraint %d acts\n',i);
    nnb=input('');
    BB(1:3,1)=input('Enter 0 if x or y or z is constrained 1 if not, on node in [bx;by;bz] format,Enter just 0 to constraint all 3\n');
    B((nnb*3)-2:(nnb*3),1)=BB;
end
```

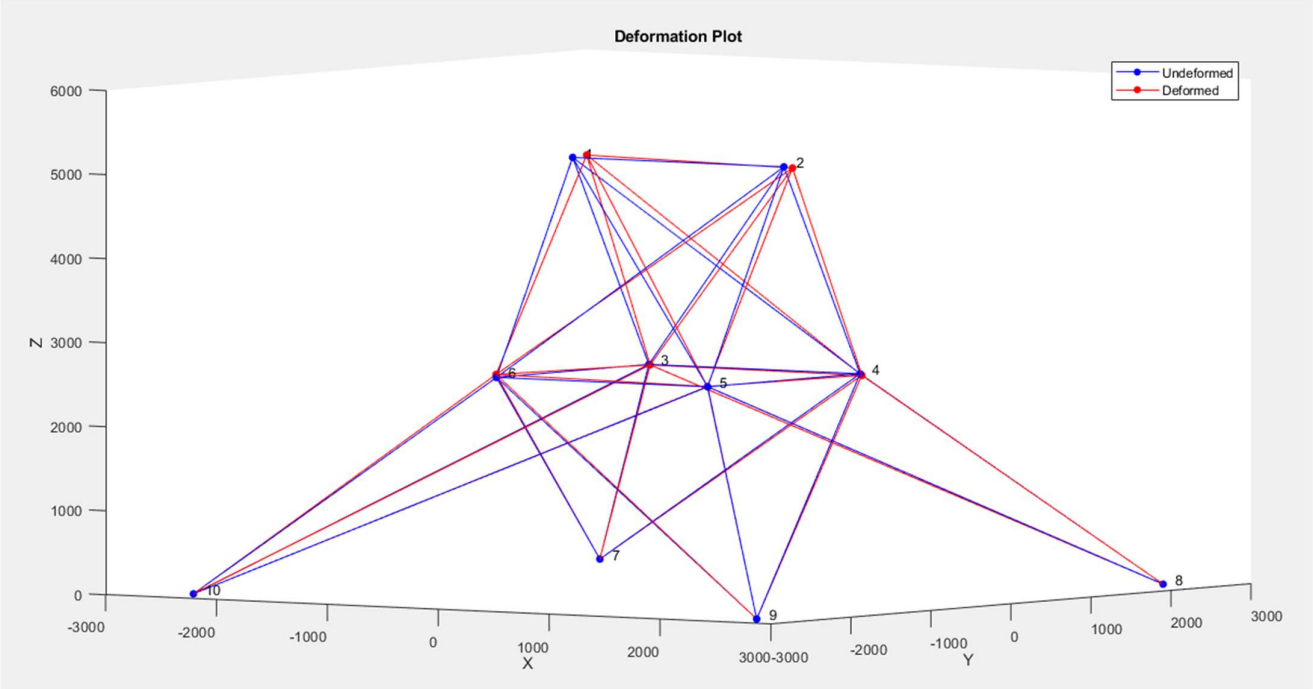
6. Settings
In this section the scaling factor for the plot are defined.

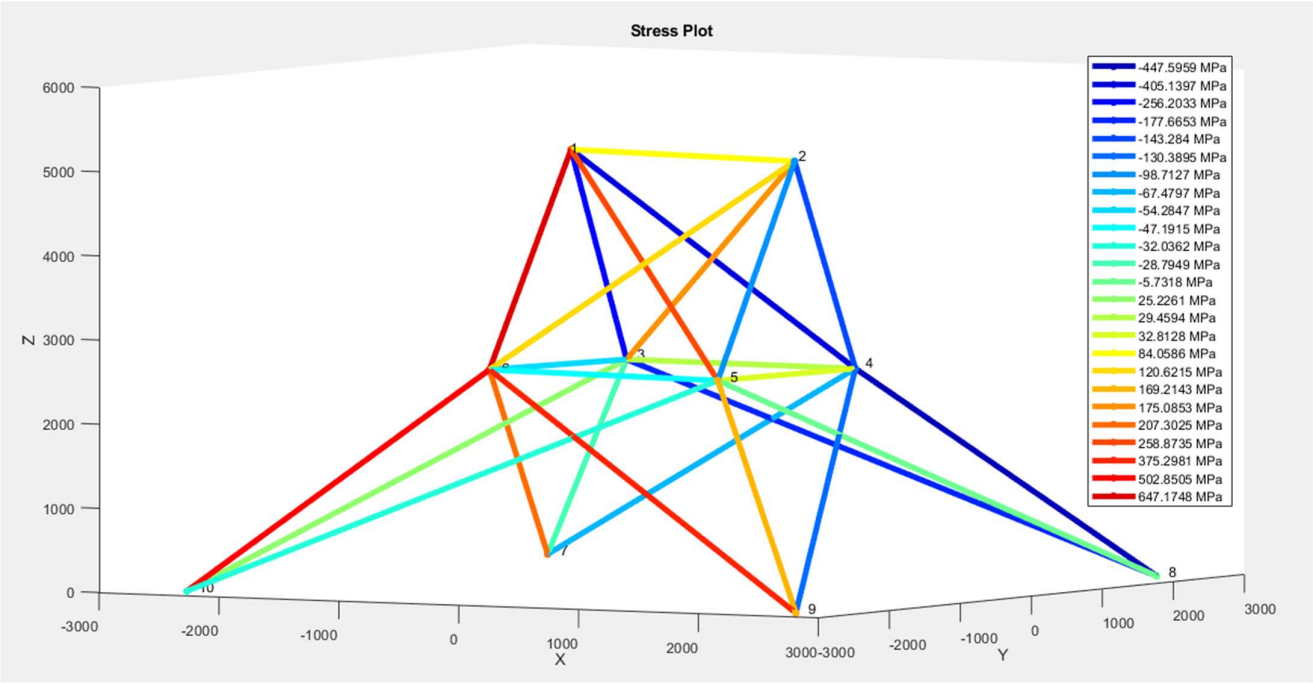
```
%% Settings

% Defining the scaling factors

if max(max(N))<=100 % Finding the maximum element of the node coordinate matrix N
    scale=max((max(N)/5));
    set=min(max(N)/40);
else
    scale=max((max(N)/20));
    set=min(max(N)/40);
end

factor=4/max(A);
```

7. Calculating the displacements and plotting them
Matrices named 'GLK', 'k_elem', and 'L' are defined to represent the global stiffness matrix, element stiffness matrix and the length of the elements respectively. The direction cosines are stored in 'l', 'm' and 'n' matrices and the global stiffness matrix is assembled.

```
%% Truss calulation and plotting results

GLK=zeros(3*mmm); % Global stiffness matrix
k_elem=cell(1,E); % Stiffness matrix for elements
L=zeros(1,E); % Length matrix for elements

% P is the matrix that contains the first and second nodes for all the
% elements. Refer to line 24 of file 'B_GEOMETRY_of_TRUSS.m'.

for i=1:E
    L(i)=sqrt((N(P(2,i),1)-N(P(1,i),1))^2+(N(P(2,i),2)-N(P(1,i),2))^2+(N(P(2,i),3)-N(P(1,i),3))^2);
    % Finding the lenght of the elements from their respective nodes
    l(i)=(N(P(2,i),1)-N(P(1,i),1))/L(i); % Direction cosines
    m(i)=(N(P(2,i),2)-N(P(1,i),2))/L(i); % Direction cosines
    n(i)=(N(P(2,i),3)-N(P(1,i),3))/L(i); % Direction cosines
    k_elem{i} = (Modulus*A(i))/L(i)*[l(i)*l(i), l(i)*m(i), l(i)*n(i); % Calculating the stiffness matrix for element i
                                     m(i)*l(i), m(i)*m(i), m(i)*n(i);
                                     n(i)*l(i), n(i)*m(i), n(i)*n(i)];
end

for j=1:E % Assembling the global stiffness matrix
    GLK(((P(1,j)*3)-2):P(1,j)*3,((P(1,j)*3)-2):P(1,j)*3)= GLK(((P(1,j)*3)-2):P(1,j)*3,((P(1,j)*3)-2):P(1,j)*3)+k_elem{j};
    GLK(((P(1,j)*3)-2):P(1,j)*3,((P(2,j)*3)-2):P(2,j)*3)= GLK(((P(1,j)*3)-2):P(1,j)*3,((P(2,j)*3)-2):P(2,j)*3)-k_elem{j};
    GLK(((P(2,j)*3)-2):P(2,j)*3,((P(1,j)*3)-2):P(1,j)*3)= GLK(((P(2,j)*3)-2):P(2,j)*3,((P(1,j)*3)-2):P(1,j)*3)-k_elem{j};
    GLK(((P(2,j)*3)-2):P(2,j)*3,((P(2,j)*3)-2):P(2,j)*3)= GLK(((P(2,j)*3)-2):P(2,j)*3,((P(2,j)*3)-2):P(2,j)*3)+k_elem{j};
end
```

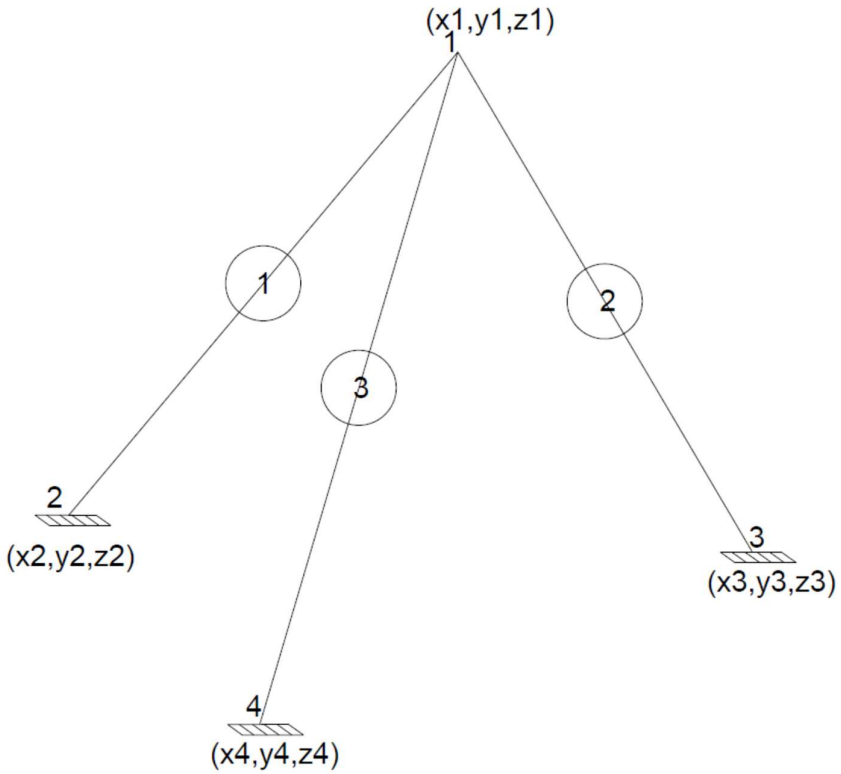
The required calculations are made and the results are plotted. Two plots are produced, namely, the ‘Deformation Plot’ (displaying the deformed and original geometries) and the ‘Stress Plot’ (displaying the stress variation in the members.





App Designing

We have used MATLAB App Designer for designing the app. Due to limitation of app designer like variable GUI is not possible in MATLAB app designer. We have designer app for specific problem. The problem is

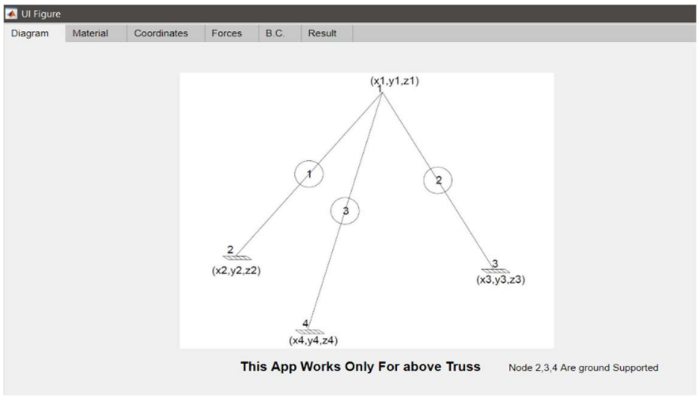


Where node 2,3,4 are ground supported. In given problem you can vary coordinates of nodes and young's modulus, area of cross section of material.
App is divided into 6 tabs they are

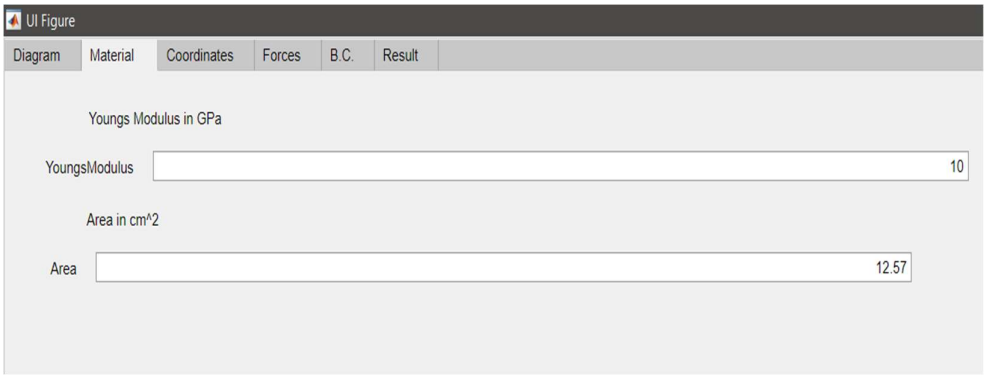
1) Diagram
2) Material

3) Coordinates
4) Forces
5) Boundary condition
6) Result

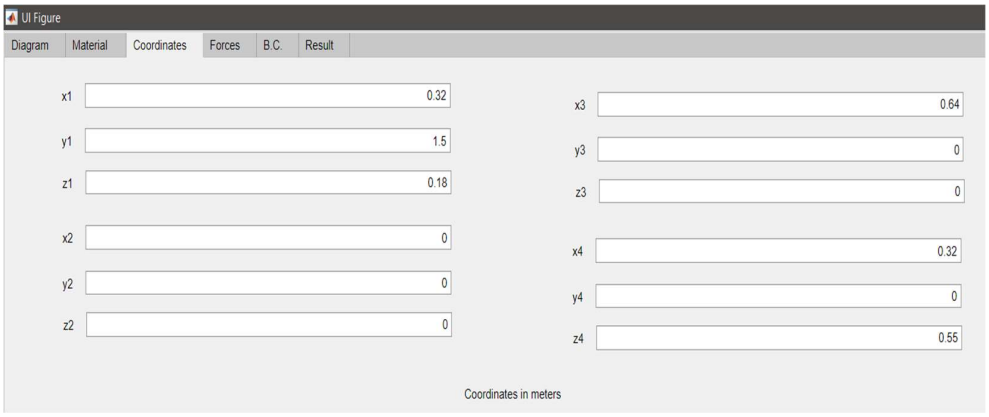
1) Diagram tab include the diagram of truss



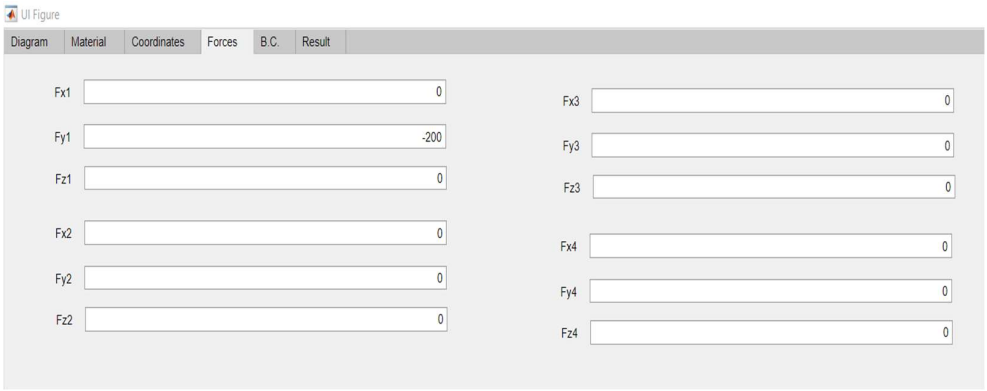
2)Material Properties tab includes the properties of materials like Youngs modulus material and area of cross section of material



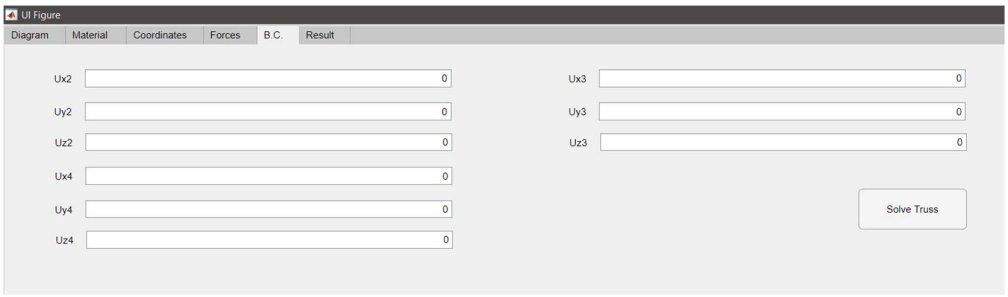
3)Coordinates tab includes taking the coordinates of different nodes from user.



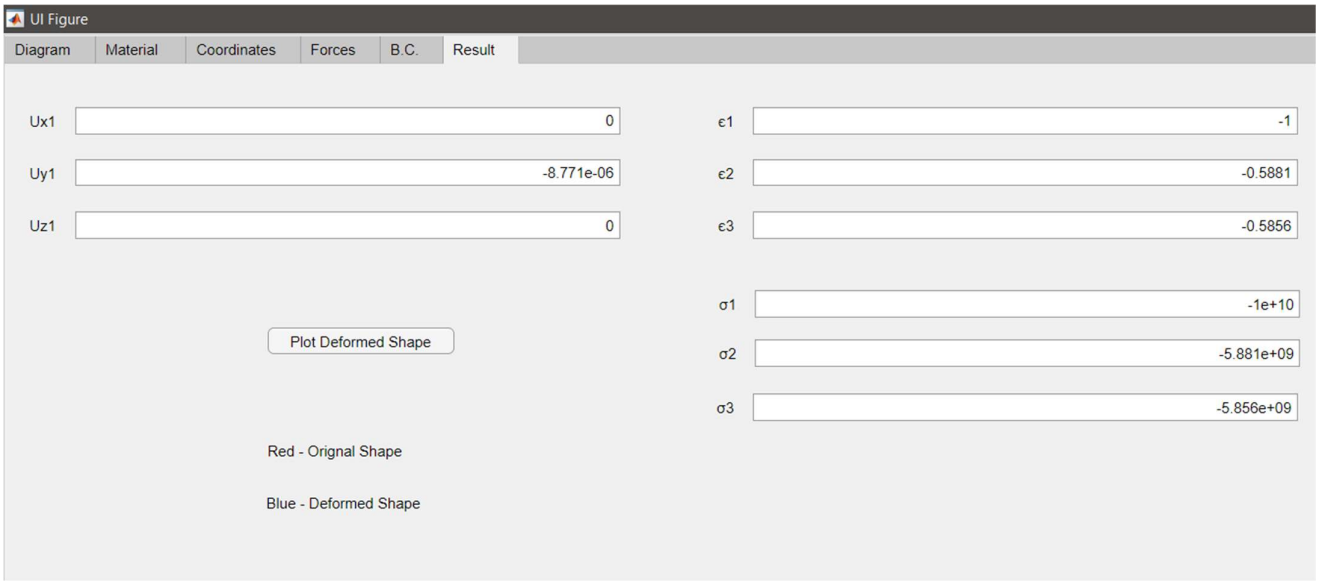
4)Forces tab includes the external forces acting on nodes.



5) B.C. tab includes the Boundary condition of various nodes.



6)Result tab includes the result of truss analysis: displacement of nodes, stress and strain in elements, plot of original shape and deformed shape.

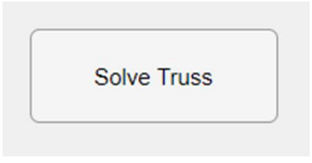


Blue - Deformed Shape Vs Red - Actual Shape

Behind Working of App

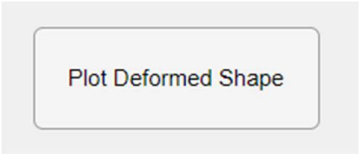
We took the inputs from user in specified format. There are two buttons in app which does the all the calculation the are

1) Solve Truss -



It calculates all length, cos(x), cos(y), cos(z), stiffness matrix, displacement, stress and strain of each element.

2) Plot Deformed Shape –



It Plot the Actual shape of truss (Red) vs Deformed shape of Graph (Blue). For clearly visibility of actual and deformed it is recommended to give large value of forces.