Developing Python Code for Static and Modal Analysis of Plane Truss Structure

A project report submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

MECHANICAL ENGINEERING

BY

DUNGA DIVYA	(317126520191)
GOLAKOTI SRI NETHIKONDA	(318126520L49)
DEEPAK KUMAR BEHERA	(317126520190)
PASUPULETI BHARGAVI SRI	(317126520213)
DATLA SANTOSH SUNIL VARMA	(317126520189)

Under the esteemed guidance of

Mr. R. VARA PRASAD M.Tech (PhD)

Professor

DEPARTMENT OF MECHANICAL ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES (A)

(Affiliated to Andhra University, Accredited By NBA and NAAC with 'A' Grade)

SANGIVALASA, VISAKHAPATNAM (District) – 531162

2021

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCES (A)

(Affiliated to Andhra University, Approved by AICTE, Accredited by NBA & NAAC with A grade)
SANGIVALASA, VISAKHAPATNAM (District) – 531162

PRAGNANAM BRAHMA
ANITS

CERTIFICATE

This is to certify that the Project Report entitled "**DEVELOPING PYTHON CODE FOR STATIC AND MODAL ANALYSIS OF PLANE TRUSS STRUCTURE**" being submitted by DUNGA DIVYA (317126520191), GOLAKOTI SRI NETHIKONDA (318126520L49), PASUPULETI BHARGAVI SRI (317126520213), DEEPAK KUMAR BEHERA (317126520190), DATLA SANTOSH SUNIL VARMA (317126520189) in partial fulfillments for the award of degree of **BACHELOR OF TECHNOLOGY** in **MECHANICAL ENGINEERING**. It is the work of bona-fide, carried out under the guidance and supervision of **MR.R. VARA PRASAD**, Assistant Professor, Department Of Mechanical Engineering, ANITS during the academic year of 2017-2021.

Approved By

PROJECT GUIDE

HEAD OF THE DEPARTMENT

(MR.R. VARA PRASAD)
Assistant Professor
Mechanical Engineering Department
ANITS, Visakhapatnam.

(Dr. B. Naga Raju)
Head of the Department
Mechanical Engineering Department
ANITS, Visakhapatnam.

PROFESSOR & HEAD
Department of Mechanical Engineering
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY & SCIENCE
Sangivalasa-531 162 VISAKHAPATNAM Dist. A.P.

ACKNOWLEDGEMENTS

We express immensely our deep sense of gratitude **Mr.R.Vara Prasad**,asst Professor Department of Mechanical Engineering, Anil Neerukonda Institute of Technology & Sciences, Sangivalasa, Bheemunipatnam Mandal, Visakhapatnam district for his valuable guidance and encouragement at every stage of the work made it a successful fulfillment.

We were very thankful to **Prof.T.V.HanumanthaRao,** Principal and **Prof.B.NagaRaju,** Head of the Department, Mechanical Engineering Department, Anil Neerukonda Institute of Technology & Sciences for their valuable suggestions.

We express our sincere thanks to the members of non-teaching staff of Mechanical Engineering for their kind co-operation and support to carry on work.

Last but not the least, we like to convey our thanks to all who have contributed either directly or indirectly for the completion of our work.

DUNGA DIVYA (317126520191)

DEEPAK KUMAR BEHERA (317126520190)

GOLAKOTI SRI NETHIKONDA (318126520L49)

PASUPULETI BHARGAVI SRI (317126520213)

DATLA SANTOSH SUNIL VARMA (317126520189)

ABSTRACT

Python is a high-sophisticated, general- purposefulness, object-oriented, enhanced programming language that includes several general features like clean, easy and simple language, indicative language, dynamically typed, automatic memory management and interpreted and very best tool when comparing with MATLAB because of various rich libraries. In various fields of science, computational work and numerical calculations forms a bridge between theory and experimentation which leads to developing automation and simulation. Developing of automation or simulation has created several benefits like quality, high production rate, efficient use of materials, increased safety and decreases industry lead time. Therefore, the present work have been investigated to develop Python code for automation in structural and vibrational analysis of any Truss structure. In order to write python code, Railway bridge truss structure was considered and the data was taken from introduction to finite elements by chandrapatla. Fully working python code was developed using jupyter notebook and python 3.9.10 and investigated nodal displacements, element stresses, support reactions, free natural frequencies and mode shapes. These results are compared using ANSYS APDL R21. Therefore, using this developed python code, many researchers can do automation for analysis of any truss structure at rocket speed.

Then Modal analysis of truss is to be done by running python script. For validation purpose, the results obtained from python scripting are compared with the results obtained from ANSYS software and MATLAB. Therefore this project is going to show that Python is best suited for scientific applications when compared to ANSYS and MATLAB

Keywords:. *ANSYS APDL, Automation, Jupyter Notebook, MATLAB, Python, Truss.*

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER- I

INTRODUCTION

1.0 SCOPE AND BACKGROUND:

It is no secret that problems that were 10 years ago considered intractable except on dedicated supercomputers, now routinely run on modest commodity personal computers. However, it is not quite as well recognized that these advances have also been accompanied by important changes in programming languages, all of which aim at greatly reducing the effort of writing codes. Scientific applications are no exceptions; many legacy programs written in Fortran, C or even C++ could probably be rewritten in a more concise way using a variety of scripting languages, such as Matlab, Scilab, IDL, Mathematica and others.

Scripting languages have several advantages over 'conventional' languages.
(1) scripting languages produce portable codes,
(2) require little or nor memory management responsibility by the programmer and
(3) allow data types to be dynamically set at run time.

All of these factors contribute to making codes less bug prone and so ultimately lead to increased productivity and shorter code development cycles.

These advantages, unfortunately, have a price; scripting languages tend to be inferior in raw numerical performance when benchmarked against compiled codes. Our experience, however, has been that scripting languages, and in particular Python, often perform above expectation, and are thus well positioned to find the optimum between satisfactory execution time and acceptable program development cost.

It is convenient here to distinguish between matrix oriented scripting languages (Matlab, Scilab, IDL,...) geared towards engineering and scientific applications from symbolic languages (Mathematica, Maple,...) that are strong in manipulating mathematical expressions, and general purpose scripting languages (Python, Perl,...). Although this distinction has become increasingly blurred, Matlab now allows for some degree of abstract programming while Mathematica improved its raw numerical performance, it is generally true that languages in group one are superior in number crunching at the expense, perhaps, of limited programming flexibility. Languages in group three on the other hand are traditionally used for

web programming, as substitutes to shell scripts, for text processing and the production of graphical user interfaces. Python is one such language although one could also claim that, thanks to the NumPy module extension, Python has some overlap with languages in group one, too.

Although extremely useful for rapid prototyping, we have found that matrix oriented languages are not always suitable for scientific applications. Many objects such as trees, graphs and sparse matrices do not neatly fit into a matrix cast. In dealing with unstructured meshes for instance, there can be an arbitrary number of connections between a node and its neighbors, which may only be known at run time. These are typical cases where Python nested structures are best suited. We have also found the dictionary data type in Python, where values are accessed by keys, extremely useful. Dictionaries (or hash tables) can grow dynamically during a calculation. Keys can be (immutable) aggregate objects, including strings, integers, doubles, a list, or a mixture of these. Finally, Python is unique in fully supporting (yet not enforcing) object oriented programming. The usual arithmetic operators can be overloaded as in C++. However, in addition to C++ the slicing operators such as a(n:m) (available in Fortran 90 and Matlab) can also be overloaded.

In summary, Python contains features individually but not globally available in Fortran 90, Java and C++.

It is for the above reasons that we have selected Python as the programming scripting language for analysis of structural problems like plane truss structure.

1.1 INTRODUCTION OF TRUSS

A plane truss is defined as a two dimensional framework of straight prismatic members connected at their ends by frictionless hinged joints, and subjected to loads and reactions that act only at the joints and lie in the plane of the structure. The members of a plane truss are subjected to axial compressive or tensile forces.Trusses are widely used in bridges, buildings, and other infrastructures. The function of a truss is to provide turgidity to the skeleton. A truss is an assembly of metallic elements (bars, rods, pipes, etc). The elements of a truss are interdependent and exert force on one another, to survive the external load and burden. A truss is used instead of RCC and concrete beams. Trusses are of different types with regard to their designs and shapes.

1.2 TYPES OF TRUSS

Basically, there are two types of the truss on the basis of their design and working mechanism.

1. Pitched Truss
2. Parallel Truss

1.2.1 Pitched Truss

In pitched truss, the chord (upper stringer) and bottom (the lower stringer) are not parallel. The chord of the truss is extended outward like an arch or a cone. The extended chord of the truss provides extra strength to the truss. The pitched trusses are used in constructing roofs of the buildings, especially in the area of snowfall. The cone-shaped roofs do not allow the snowfall dump on the roof while making the snowfall slip down from the edges of the roof.

1.2.2 Parallel Truss

A parallel truss is made up of the parallel chord and bottom. The chord and bottom run straight in a parallel path. Both the stringers (chord and bottom) are interconnected by means of struts (the connecting rods). If compared, the pitched trusses are stronger than the parallel truss. A parallel truss is generally used instead of girders and beams.

1.3 BASIC TYPES OF TRUSS

1. Warren Truss
2. Octet Truss
3. Prat Truss
4. Bowstring Truss
5. King post Truss
6. Lenticular Truss
7. Town's Lattice Truss
8. Vierendeel Truss

1.3.1 Warren Truss:

It is a very simple type of trusses, in which the truss members form a series of equilateral triangles. These are included in the category of the parallel truss.

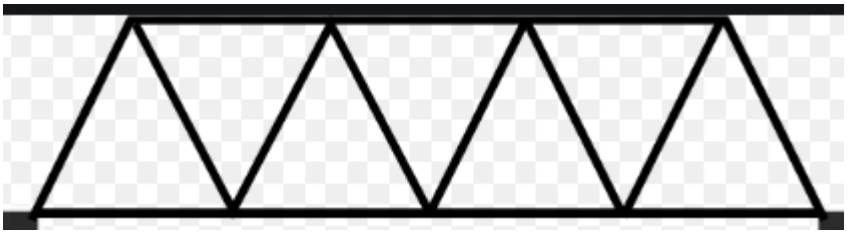


Fig 1.1 warren truss

1.3.2 Octet Truss:

In this type of trusses, the truss members are made up of all equivalent equilateral triangles. This is a very complicated truss, in which each triangle is associated with the other in multi-dimensions. This type of truss is strongest as compared to the rest of the types. This type of trusses is designed with very high skill and is very difficult to understand.

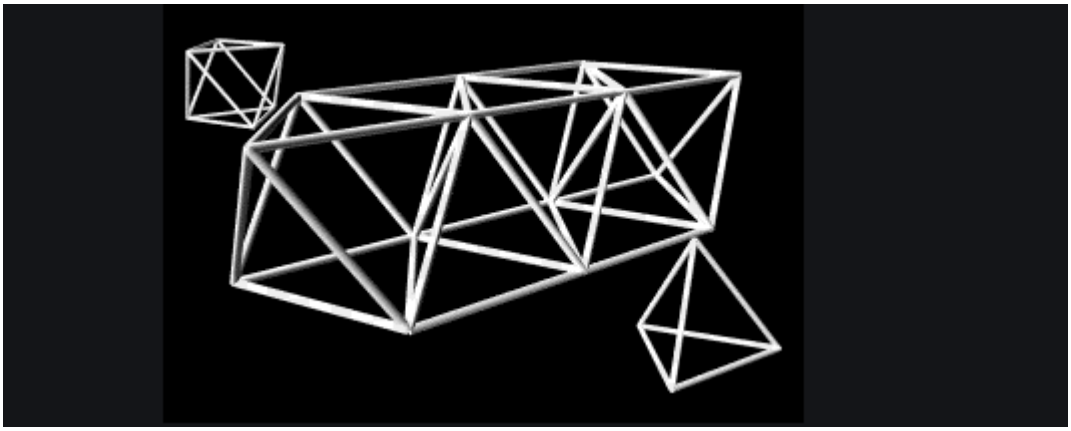


Fig 1.2 octet truss

1.3.3 Pratt Truss:

In 1844, the engineers of the Boston railway track designed it. Two types of members are used in this truss. One is vertical and the other is a diagonal member. The two types of members consecutively, follow one another. The vertical members are for compression and the diagonal members are for responding tension.

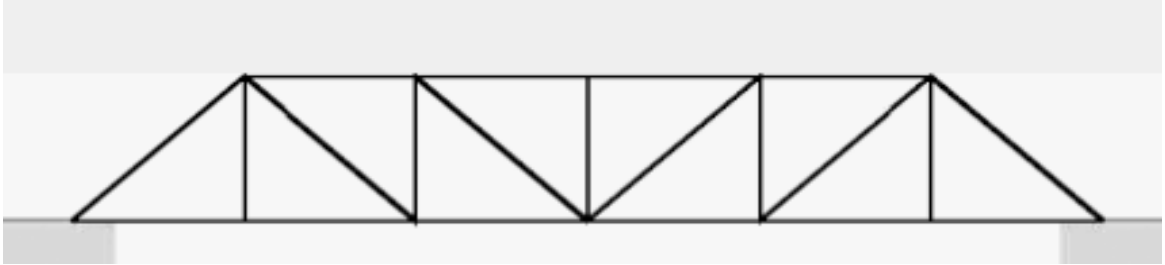


Fig 1.3 pratt truss

1.3.4 Bowstring Truss:

Bowing Strings are used in this type of trusses. The bowstrings act as an arch. These strings give extra turgidity to the truss. These were, first used in World War II. The need for such type of trusses was felt, the curved roof of aircraft was to be designed.

Fig 1.4 bowstring truss

1.3.5 King post Truss:

In this type of trusses, two angled members/struts support a vertical strut. It is very simple to design but frequently used truss. In this design, the vertical member/strut is called Kingpost.

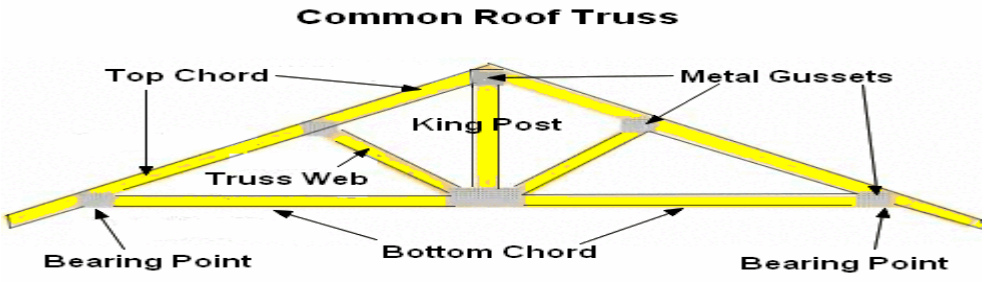


Fig 1.5 king poss truss

1.3.6 Lenticular Truss:

Lenticular Truss was, first time used in the Gaunless Railway bridge of Stockholm and Darlington in 1823. In this type the chord and the bottom, both are arched and connect with each other at both ends.

Fig 1.6 lenticular truss

1.3.7 Town's Lattice Truss:

In these trusses, the inclined members are used which cross over one another at frequent points. An American architect "Itheal Town" designed it, this is why is known after his name.

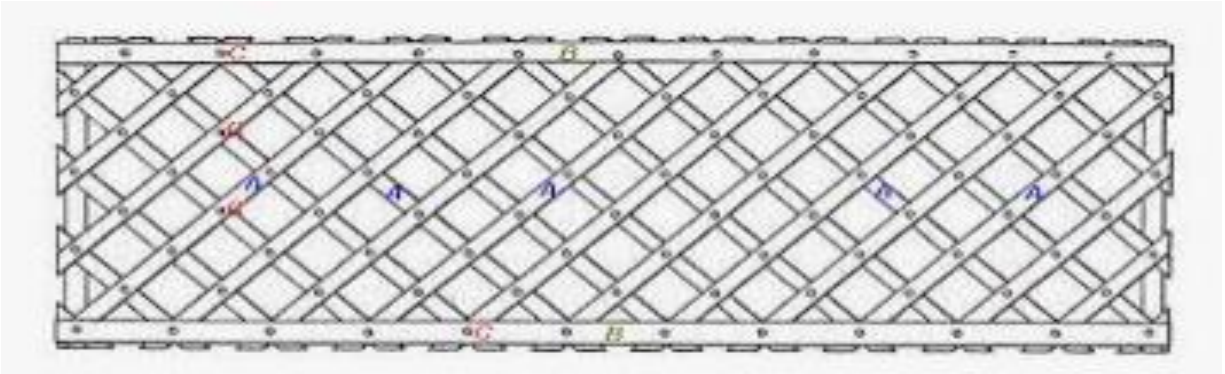


Fig 1.7 town's lattice truss

1.4 ELEMENTS OF TRUSS

Almost all the trusses are made up of three fundamental components. The chord, the bottom, and the members.

1. Upper stringer in a truss is called the chord.

2. The lower stringer of the truss is called the bottom.

3. Members, also called struts are the bars, rod, and strips that connect the chord and bottom of the truss.

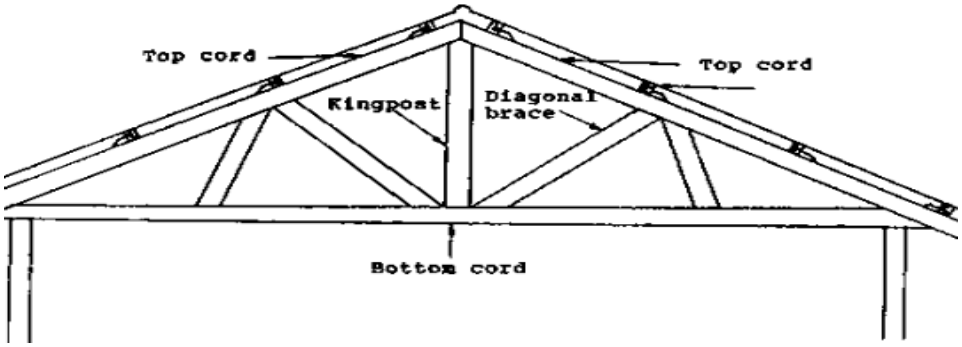


Fig 1.8 element of truss

1.5 WHAT IS FEA

Finite Element Analysis is a mathematical representation of a physical system comprising a part/assembly (model), material properties, and applicable boundary conditions {collectively referred to as pre-processing}, the solution of that mathematical representation {solving}, and the study of results of that solution {post-processing}. Simple shapes and simple problems can be, and often are, done by hand. Most real world parts and assemblies are far too complex to do accurately, let alone quickly, without use of a computer and appropriate analysis software.

1.6 MODAL ANALYSIS

A technique used to determine a structure's vibration characteristics: – Natural frequencies – Mode shapes – Mode participation factors (how much a given mode participates in a given direction). It is Most fundamental of all the dynamic analysis types.

1.6.1 Benefits of modal analysis:

Allows the design to avoid resonant vibrations or to vibrate at a specified frequency (speakers, for example). Gives engineers an idea of how the design will respond to different types of dynamic loads. Helps in calculating solution controls (time steps, etc.) for other dynamic analyses. Recommendation:

Because a structure's vibration characteristics determine how it responds to any type of dynamic load, always perform a modal analysis first before trying any other dynamic analysis.

1.6.2 Terminology & Concepts:

General equation of motion:

[M]{&u&} + [C]{u&} + [K]{u} = {F(t)}

Assume free vibrations and ignore damping:

[M]{&u&} +[K]{u} = {0}

Assume harmonic motion (i.e.u = Usin(ωt))

([K] [M]){u} {0}

The roots of this equation are ωi2 , the eigenvalues, where i ranges from 1 to number of DOF. Corresponding vectors are {u}i , the eigenvectors.

The square roots of the eigenvalues are ωi , the structure's natural circular frequencies (radians/sec). Natural frequencies fi are then calculated as fi = ωi /2π (cycles/sec). It is the natural frequencies fi that are input by the user and output by ANSYS.

The eigenvectors {u}i represent the mode shapes - the shape assumed by the structure when vibrating at frequency fi .

1.6.3 Mode Extraction:

It is the term used to describe the calculation of eigenvalues and eigenvectors.

1.6.4 Mode Expansion:

It has a dual meaning. For the reduced method, mode expansion means calculating the full mode shapes from the reduced mode shapes. For all other methods, mode expansion simply means writing mode shapes to the results file.

1.6.5 Mode Extraction Methods:

Several mode extraction methods are available in ANSYS:

Block Lanczos (default)

Subspace

PowerDynamics

Reduced

Unsymmetric

Damped (full)

QR Damped

Which method you choose depends primarily on the model size (relative to your computer resources) and the particular application.

Table 1.1 mode extraction methods

Extraction method	Linear Solver Used	Remarks
Block Lanczos	Sparse Matrix	Recommended for most applications; Most stable;
Powerdynamics	PCG solver	Same as subspace but with PCG solver; Can handle very large models; Lumped mass only; May miss modes; Modes cannot be used in subsequent spectrum and PSD analyses
Reduced	Frontal Solver	In general fastest; Accuracy depends on Master DOF selection; Limitations similar to Subspace; Not recommended due to expertise required in selecting Master DOF.

1.6.6 Modal Analysis Procedure:

Four main steps in a modal analysis:

Build the model

Choose analysis type and options

Apply boundary conditions and solve

Review results.

Build the model :

Remember density!

Linear elements and materials only. Nonlinearities are ignored.

Choose analysis type and options

Enter Solution and choose modal analysis.

Mode extraction options*

Mode expansion options*

Other options*

Mode extraction options

Method: Block Lanczos recommended for most applications.

Number of modes: Must be specified (except Reduced method).

Frequency range: Defaults to entire range, but can be limited to a desired range (FREQB to FREQE). Specification of a frequency range requires additional factorizations and it is typically faster to simply request a number of modes which will overlap the desired range.

Normalization:

Only the shape of the DOF solution has real meaning. It is therefore customary to normalize them for numerical efficiency or user convenience.

Modes are normalized either to the mass matrix or to a unit matrix (unity). – Normalization to mass matrix is the default, and is required for a spectrum analysis or if a subsequent mode superposition analysis is planned. – Choose normalization to unity when you want to easily compare relative values of displacements throughout the structure.

Modes normalized to unity cannot be used in subsequent mode superposition analyses (transient, harmonic, spectrum or random vibration).

Mode expansion:

You need to expand mode shapes if you want to do any of the following:

Have element stresses calculated.

Do a subsequent spectrum or mode superposition analysis.

1.6.7 Other analysis options:

Lumped mass matrix – Mainly used for slender beams and thin shells, or for wave propagation problems. – Automatically chosen for PowerDynamics method.

Pre-stress effects – For Pre-stressed modal analysis (discussed later).

Full damping – Used only if Damped mode extraction method is chosen. – Damping ratio, alpha damping, and beta damping are allowed. – BEAM4 and PIPE16 also allow gyroscopic damping.

QR damping – All types of damping are allowed.

Apply boundary conditions and solve

Displacement constraints

External loads

However, ANSYS creates a load vector which you can use in a subsequent mode superposition analysis.

Review results using POST1, the general postprocessor

List natural frequencies

View mode shapes

Review participation factors

Review modal stresses.

1.7 WHY PYTHON FOR SCENTIFIC COMPUTERING?

The design focus on the Python language is on productivity and code readability, for example through:

- Interactive python console
- Very clear, readable syntax through whitespace indentation
- Strong introspection capabilities
- Full modularity, supporting hierarchical packages
- Exception-based error handling

• Dynamic data types & automatic memory management

As Python is an interpreted language, and it runs many times slower than compiled code, one might ask why anybody should consider such a 'slow' language for computer simulations?

1.7.1 Numerical Python:

The NumPy package (read as NUMericalPYthon) provides access to

• a new data structure called arrays which allow efficient vector and matrix operations.

It also provides a number of linear algebra operations (such as solving of systems of linear equations, computation of Eigenvectors and Eigenvalues).

There are two other implementations that provide nearly the same functionality as NumPy. These are called "Numeric" and "numarray":

• Numeric was the first provision of a set of numerical methods (similar to Matlab) for Python. It evolved from a PhD project.

• Numarray is a re-implementation of Numeric with certain improvements (but for our purposes both Numeric and Numarray behave virtually identical).

• Early in 2006 it was decided to merge the best aspects of Numeric and Numarray into the Scientific Python (scipy) package and to provide (a hopefully "final") array data type under the module name "NumPy". We will use in the following materials the "NumPy" package as provided by (new) SciPy.

If for some reason this doesn't work for you, chances are that your SciPy is too old. In that case, you will find that either "Numeric" or "numarray" is installed and should provide nearly the same capabilities.

1.7.2 Visualisation in Python:

The Python library Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code. For more detailed information, check these links

• A very nice introduction in the object oriented Matplotlib interface, and summary of all important ways of changing style, figure size, linewidth, etc. This is a useful reference: http://nbviewer.ipython.org/urls/raw.github.com/jrjohansson/scientific-python-lectures/master/Lecture4-Matplotlib.ipynb

• Matplotlib tutorial http://matplotlib.sourceforge.net/users/index.html

• Matplotlib home page http://matplotlib.sourceforge.net

• List of simple screenshot examples http://matplotlib.sourceforge.net/users/screenshots.html

• Extended thumbnail gallery of examples http://matplotlib.sourceforge.net/gallery.html

1.7.3 Numerical Methods using Python SciPy:

scipy package (SCIentificPYthon) which provides a multitude of numerical algorithms

Many of the numerical algorithms available through scipy and numpy are provided by established compiled libraries which are often written in Fortran or C. They will thus execute much faster than pure Python code (which is interpreted). As a rule of thumb, we expect compiled code to be two orders of magnitude faster than pure Python code.

CHAPTER-II

LITERATURE REVIEW

Dorival et.al.,[1] investigated that the Generalized Finite Element Method (GFEM) is a numerical method based on the Finite Element Method (FEM), presenting as its main feature the possibility of improving the solution by means of local enrichment functions. In spite of its advantages, the method demands a complex data structure, which can be especially benefited by the Object-Oriented Programming (OOP). Even though the OOP for the traditional FEM has been extensively described in the technical literature, specific design issues related to the GFEM are yet little discussed and not clearly defined. Inthe present article it is described an Object-Oriented (OO) class design for the GFEM, aiming to achieve a computational code that presents a flexible class structure, circumventing the difficulties associated to the method characteristics. The proposed design is evaluated by means of some numerical examples, computed using a code implemented in Python programming language.

Aleveset.al.,[2]The objective computing project was created with the intention of writing a fully interactive-adaptive finite element program using the object oriented programming philosophy. In a first phase of the project, a graphical environment was developed to simplify and generalize the interactive programming concept. The resulting interface library gives the programmer easy access to graphical user interface tools such as windows, menus and dialogs. In order to improve programmer efficiency, the same interface library is being implemented to run under various existing toolboxes such as Macintosh, MS-Windows, OSF/Motif and others. During the second phase of the project, an innovative finite element data structure is developed which will be used as a finite element research platform

Bordaset.al.,[3]This paper presents and exercises a general structure for an object-oriented-enriched finite element code. The programming environment provides a robust tool for extended finite element (XFEM) computations and a modular and extensible system. The programme structure has been designed to meet all natural requirements for modularity, extensibility, and robustness. To facilitate mesh–geometry interactions with hundreds of enrichment items, a mesh generator and mesh database are included. The salient features of the programme are: flexibility in the integration schemes (subtriangles,

subquadrilaterals, independent near-tip, and discontinuous quadrature rules); domain integral methods for homogeneous and bi-material interface cracks arbitrarily oriented with respect to the mesh; geometry is described and updated by level sets, vector level sets or a standard method; standard and enriched approximations are independent; enrichment detection schemes: topological, geometrical, narrow-band, etc.; multi-material problem with an arbitrary number of interfaces and slip-interfaces; non-linear material models such as J2 plasticity with linear, isotropic and kinematic hardening. To illustrate the possible applications of our paradigm, we present 2D linear elastic fracture mechanics for hundreds of cracks with local near-tip refinement, and crack propagation in two dimensions as well as complex 3D industrial problems.

Duarteet.al.,[4]A new methodology to build discrete models of boundary-value problems is presented. The h-pcloud method is applicable to arbitrary domains and employs only a scattered set of nodes to build approximate solutions to BVPs. This new method uses radial basis functions of varying size of supports and with polynomialreproducing properties of arbitrary order. The approximating properties of the h-p cloud functions are investigated in this article and a several theorems concerning these properties are presented. Moving least squares interpolants are used to build a partition of unity on the domain of interest. These functions are then used to construct, at a very low cost, trial and test functions for Galerkinapproximations. The method exhibits a very high rate of convergence and has a greater -exibility than traditional h-p finite element methods. Several numerical experiments in I-D and 2-D are also presented.

Tom et.al.,[5]Partial differential equations (PDEs)—such as the Navier–Stokes equations in fluid mechanics, the Maxwell equations in electromagnetism, and the Schrödinger equation in quantum mechanics—are the basic building blocks of modern physics and engineering. The finite element method (FEM) is a flexible computational technique for the discretization and solution of PDEs, especially in the case of complex spatial domains. Conceptually, the FEM transforms a time-independent (or temporally discretized) PDE into a system of linear equations $Ax = b$. scikit-fem is a lightweight Python library for the creation, or *assembly*, of the finite element matrix A and vector b. The user loads a computational mesh, picks suitable basis functions, and provides the PDE's weak formulation (Logg, Mardal, Wells, & others, 2012). This results in sparse matrices and vectors compatible with the SciPy (Virtanen et al., 2020) ecosystem.

Cimrmanet.al.,[6]SfePy (simple finite elements in Python) is a software for solving various kinds of problems described by partial differential equations in one, two, or three spatial dimensions by the finite element method. Its source code is mostly (85%) Python and relies on fast vectorized operations provided by the NumPy package. For a particular problem, two interfaces can be used: a declarative application programming interface (API), where problem description/definition files (Python modules) are used to define a calculation,

and an imperative API, that can be used for interactive commands, or in scripts and libraries. After outlining the SfePy package development, the paper introduces its implementation, structure, and general features. The components for defining a partial differential equation are described using an example of a simple heat conduction problem. Specifically, the declarative API of SfePy is presented in the example. To illustrate one of SfePy's main assets, the framework for implementing complex multiscale models based on the theory of homogenization, an example of a two-scale piezoelastic model is presented, showing both the mathematical description of the problem and the corresponding code.

Hunter et.al.,[7]Matplotlib is a 2D graphics package for Python for application development, interactive scripting, and publication-quality image generation across user interfaces and operating systems.

2.0 OVERVIEW OF THE PROJECT

1. Doing model analysis for any structure is very difficult.
2. In this work, first we are going to investigate modal values for a plane truss structure using ANSYS software.
3. Then we will develop Python scripting for getting model values.
4. For validation we will compare results obtained by python scripting and ANSYS software.
5. If time persists we will compare with MATLAB to conclude that the Python Scriptting is best tool for analysis of structural problems.

CHAPTER-III

ANALYSIS OF TRUSS USING ANSYS SOFTWARE

3.0 ANSYS

ANSYS is a general purpose finite element modelling package for numerically solving a wide variety of mechanical problems. These problems include: static/dynamic structural analysis (both linear aswell as acoustic and electro-magnetic problems.Ansys develops and markets engineering simulation software for use across the product life cycle. Ansys Mechanical finite element analysis software is used to simulate computer models of structures, electronics, or machine components for analysing strength, toughness, elasticity, temperature distribution, electromagnetism, fluid flow, and other attributes. Ansys is used to determine how a product will function with different specifications, without building test products or conducting crash tests. For example , Ansys software may simulate how a bridge will hold up after years of traffic, how to best process salmon in a cannery to reduce waste, or how to design a slide that uses less material without sacrificing safety.Typically Ansys users break down larger structures into smallcomponents that are each modelled and tested individually.A user may start by defining thedimensions of an object, and then adding weight, pressure, temperature and other physical properties. Finally, the Ansys software simulates and analyses movement, fatigue, fractures, fluid flow, temperature distribution, electromagnetic efficiency and other effects over time.

Ansys also develops software for data management and backup, academic research and teaching. Ansys software is sold on an annual subscription basis.

In general, a finite element solution may be broken into the following three stages.This is a general guideline that can be used for setting up any finite element analysis.

1. Preprocessing defining the problem ; The major steps in preprocessing aregiven below:

→Define keypoints/lines/areas/volumes

→ Define element type and material/geometric properties

→ Mesh lines/areas/volumes as required

The amount of detail required will depend on the dimensionality of the analysis (i.e. 1D, 2D, axi-symmetric, 3D).

Solution: assigning loads, constraints and solving here we specify the loads (point or pressure), constraints (translational and rotational) and finally solve the resulting set of equations.

2.Postprocessing processing further and viewing of the results in this stage one

may wish to see:

→ Lists of nodal displacements

→ Element forces and moments

→Deflection plots

→Stress contour diagrams

ANSYS uses certain inputs and evaluates the product behaviour to the physics that you are testing it in. It is a general purpose software used to simulate the interactions between various physics like dynamics, statics, fluids, electromagnetic, thermal, and vibrations. ANSYS typically creates the user an opportunity to create a virtual environment to simulate the tests or working conditions of the products before manufacturing the prototypes. This would certainly reduce the cost of producing prototypes and mainly the time. In this competitive world the accuracy and time are the most deciding factors for the

company or the organization to sustain. ANSYS helps in increasing the accuracy and decreasing the time of outcome of the final product.

3.1 ADVANTAGES

1. ANSYS can import all kinds of CAD geometries (3D and 2D) from different CAD software's like Design Modeler and Space Claim which makes the work flow even smoother.
2. ANSYS has the capability of performing advanced engineering simulations accurately and realistic in nature by its variety of contact algorithms, time dependent simulations and non linear material models.
3. ANSYS has the capability of integrating various physics into one platform and perform the analysis. Just like integrating a thermal analysis with structural and integrating fluid flow analysis with thermal and structural,etc.,
4. ANSYS now has featured its development into a product called ANSYS AIM, which is capable of performing multi physics simulation. It is a single platform which can integrate all kinds of physics and perform simulations.
5. ANSYS has its own customization tool called ACT which uses python as a background scripting language and used in creating customized user required features in it.
6. ANSYS hasthe capability to optimize various features like the geometrical design, boundary conditions and analyse the behaviour of the product under various criterion's.

3.2 PROBLEM STATEMENT

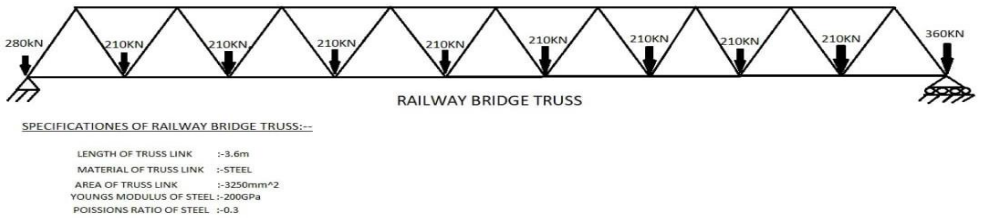


Fig 3.1 railway bridge truss

3.3 STRUCTURAL ANALYSIS OF TRUSS

Software used:- ANSYS 2021 R1.

Presteps:-Simulation environment: ANSYS

File management :working directory:-C\ANSYS\TRUSS

Job Name:-TRUSS01< Run

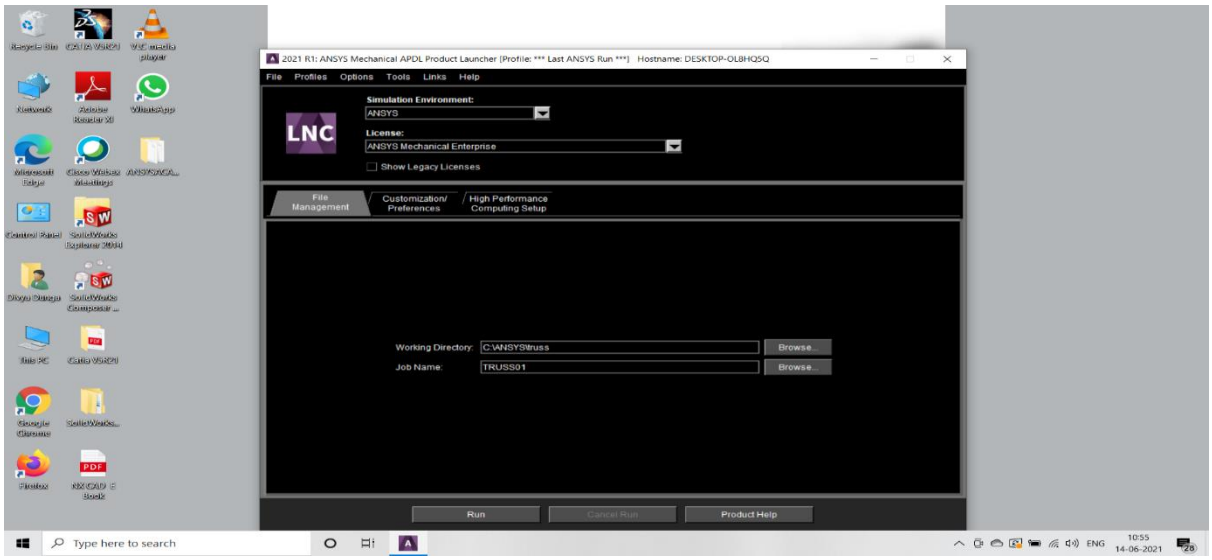


Fig 3.2 ansys file

Step 1:-firstly, we click on Preferences in main menu and then we select the type of analysis by choosing structural and proceed on by clicking Ok.

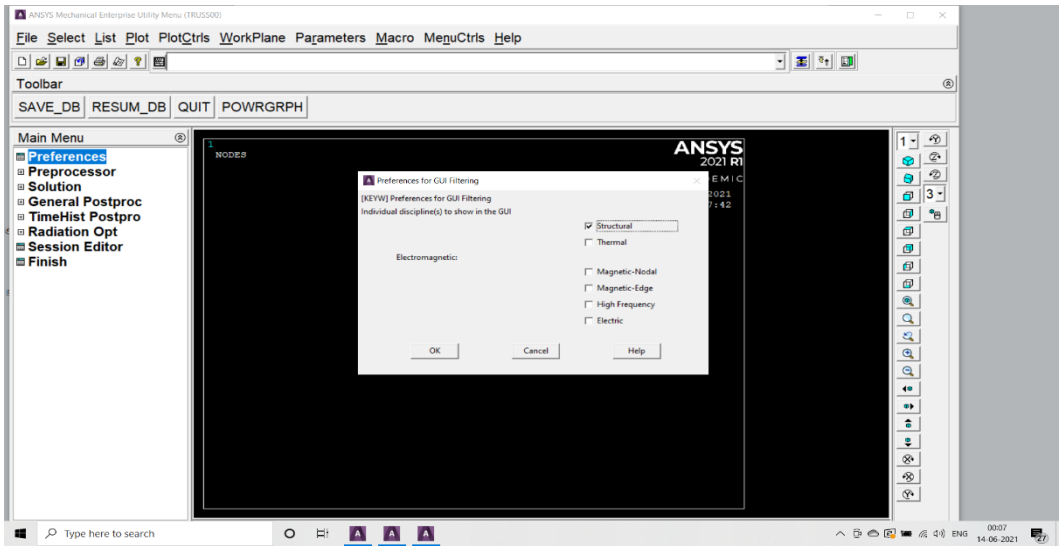


Fig 3.3 ansys preprocessor

Step 2:-Now considering preprocessor we select the Element type , and click on Add/Edit/Delete. We add link and select Link 3D finitstn 180 as element type.

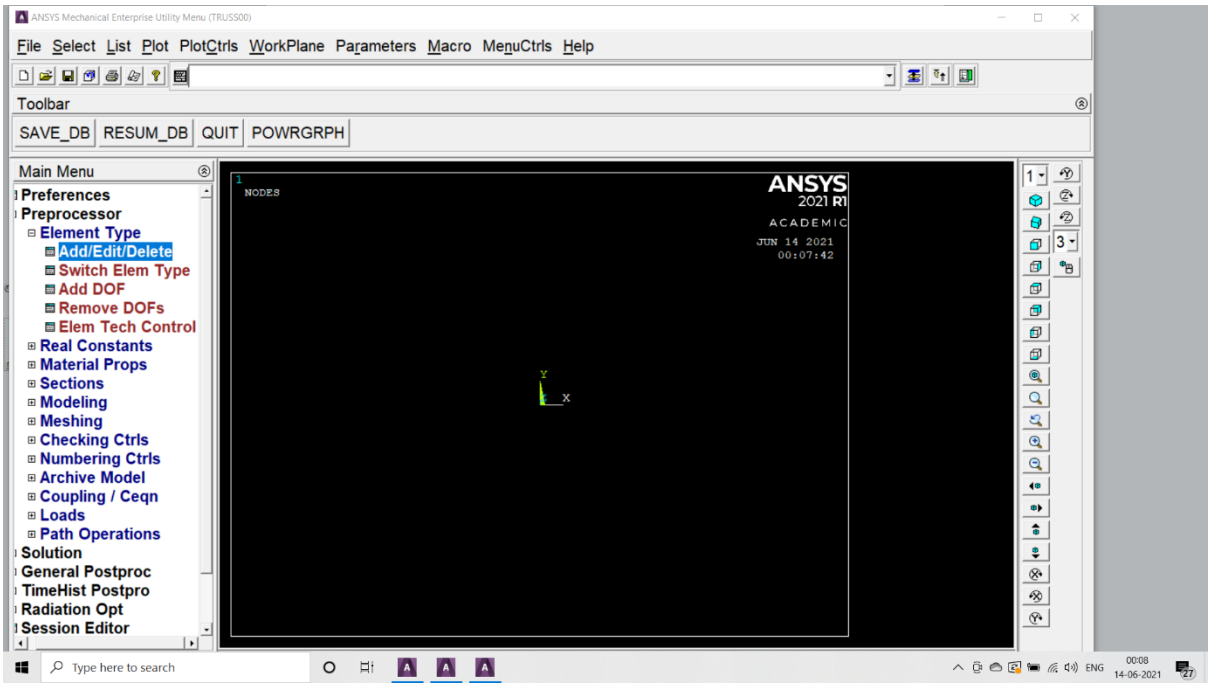


Fig 3.4 ansys element types

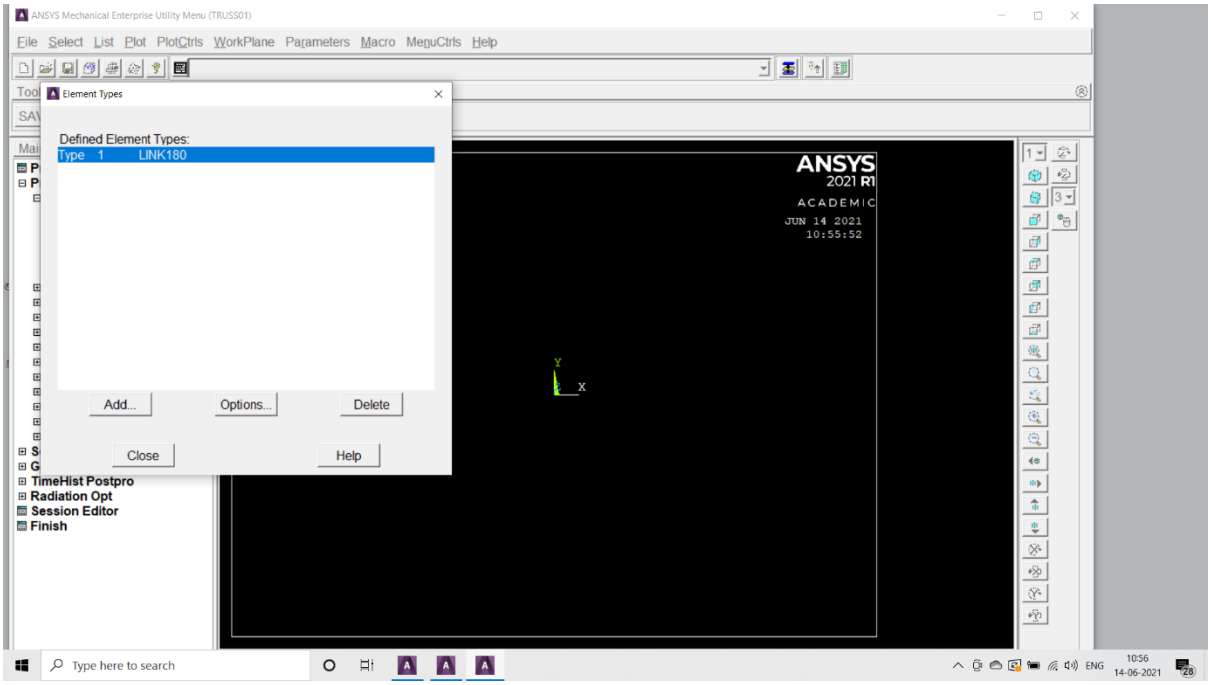


Fig 3.5 selection of link 180 element

Step 3:-In this we choose the Material props and Select the available Material models under Structural→Linear→Elastic→Isotropic and click Ok.

We assign the material property values for the material 1. Ex= 200GPA ,PRXY=0.3 → Ok

The material taken in steel

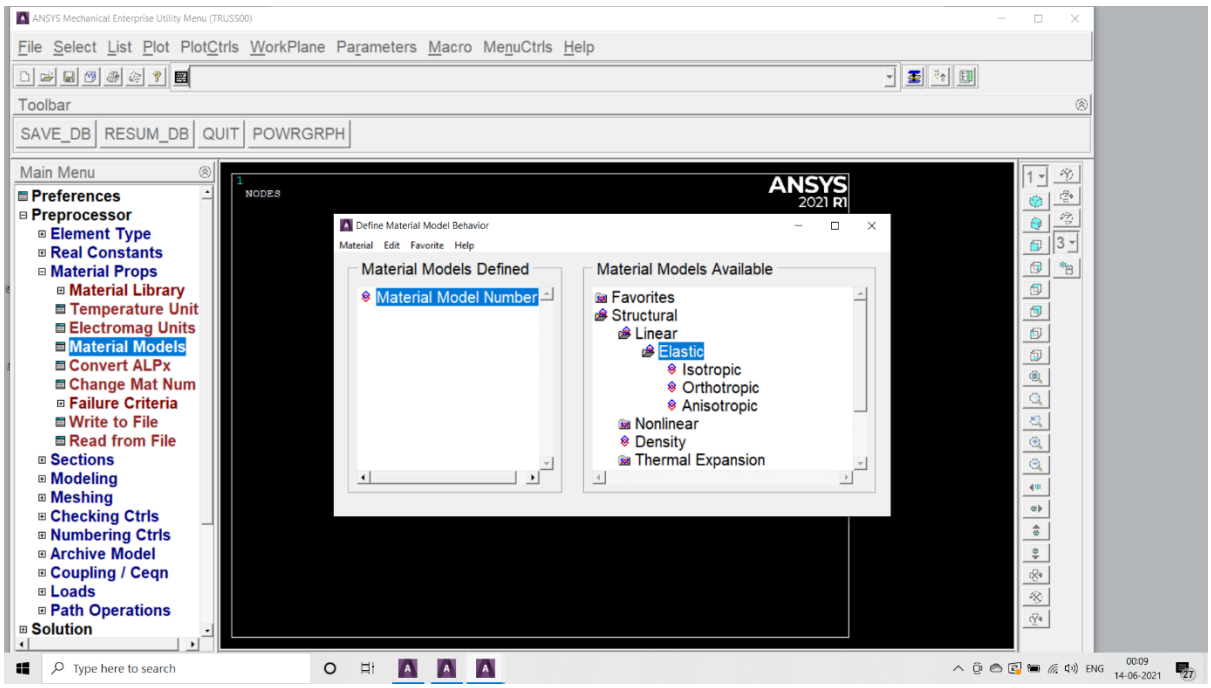


Fig 3.6 material in steel

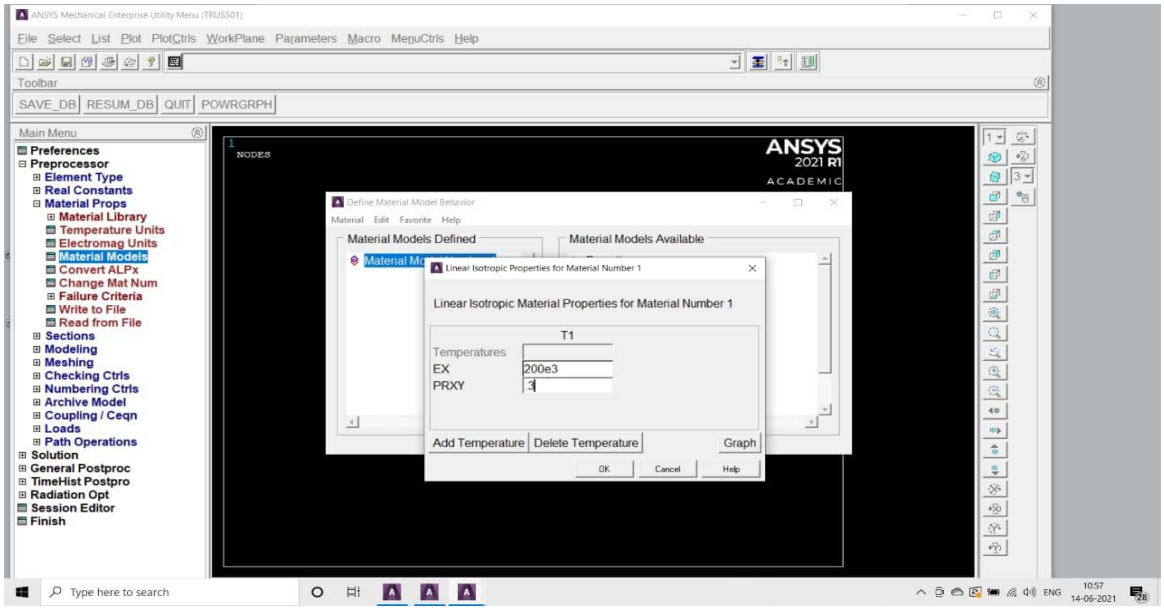


Fig 3.7 material property selection

Step 4:- We go toSections and under it we click on Link and then Add. We take the add link section with ID as 1.

And , we name the section as ‘Rect’ and take the Link area as ‘3250’ and click Ok.

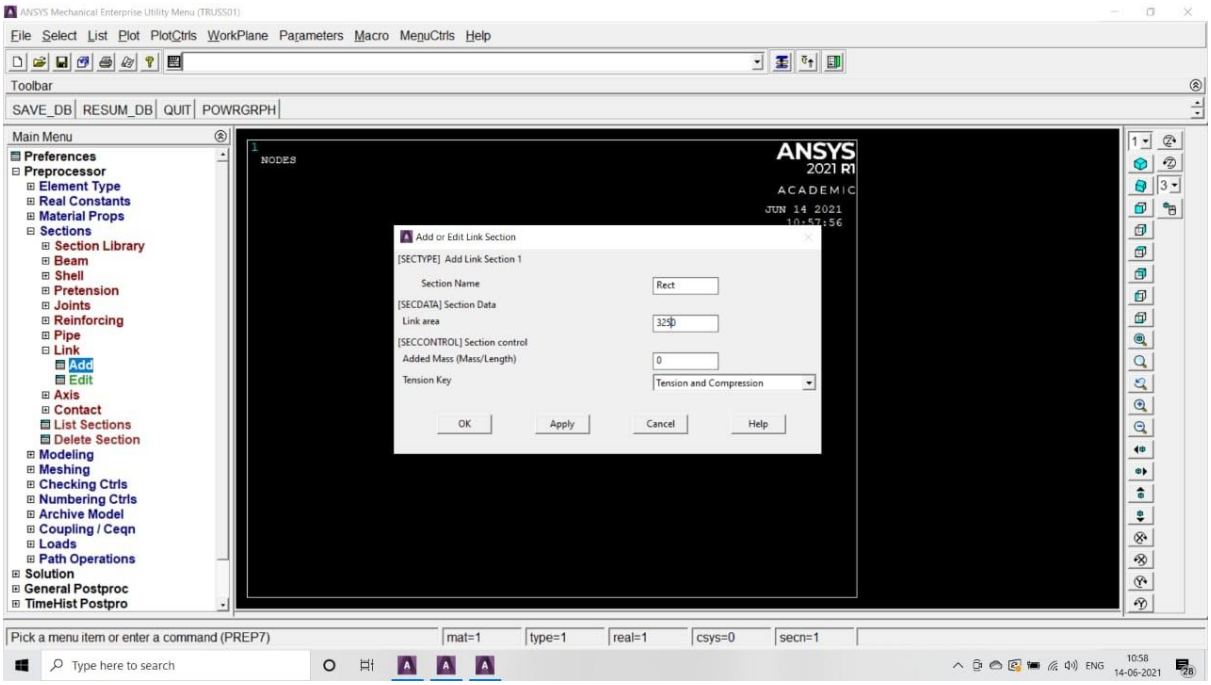


Fig 3.8 adding link section and link area

Step 5:-Now click on Modelling and select Create. We can see other functions below, select Nodes and then click on InactiveCS. A window will appear as shown in belowfig ,and we can choose the nodes and assign the values for the nodes.

Node number 1 at X = 0 , Y = 0 , Z = 0 → Apply

Node number 2 at X = 1800 , Y = 3118 , Z = 0 → Apply

Node number 3 at X = 3600 , Y = 0 , Z = 0 → Apply

Node number 4 at X = 5400 , Y = 3118 , Z = 0 → Apply

Node number 5 at X = 7200 , Y = 0 , Z = 0 → Apply

Node number 6 at X = 9000 , Y = 3118 , Z = 0 → Apply

Node number 7 at X = 10800 , Y = 0 , Z = 0 → Apply

Node number 8 at X = 12600 , Y = 3118 , Z = 0 → Apply

Node number 9 at X = 14400 , Y = 0 , Z = 0 → Apply

Node number 10 at X = 16200 , Y = 3118 , Z = 0 → Apply

Node number 11 at X = 18000 , Y = 0 , Z = 0 → Apply

Node number 12 at X = 19800 , Y = 3118 , Z = 0 → Apply

Node number 13 at X = 21600 , Y = 0 , Z = 0 → Apply

Node number 14 at X = 23400 , Y = 3118 , Z = 0 → Apply

Node number 15 at X = 25200 , Y = 0 , Z = 0 → Apply

Node number 16 at X = 27000 , Y = 3118 , Z = 0 → Apply

Node number 17 at X = 28800 , Y = 0 , Z = 0 → Apply

Node number 18 at X = 30600 , Y = 3118 , Z = 0 → Apply

Node number 19 at X = 32400 , Y = 0 , Z = 0 → Apply→ Ok

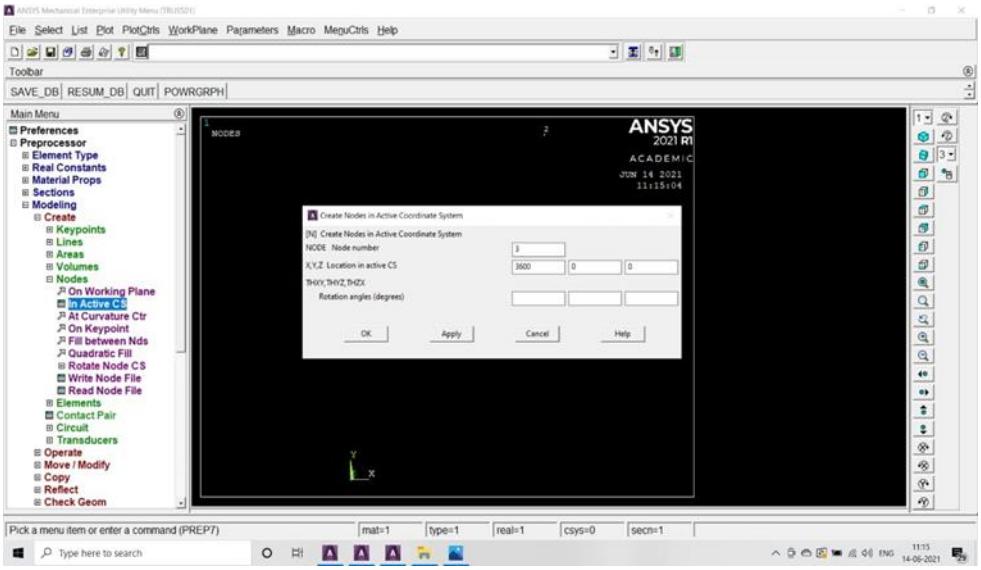


Fig 3.9 creating truss nodes in active coordinating system

Step 6:-We click on Element tool and under that we click on Auto numbered and below that we select Thru nodes and Select the nodes 1&2 and click on Apply . In the same way we select thenodes 1&3 and click on Apply.

We Repeat the same procedure.

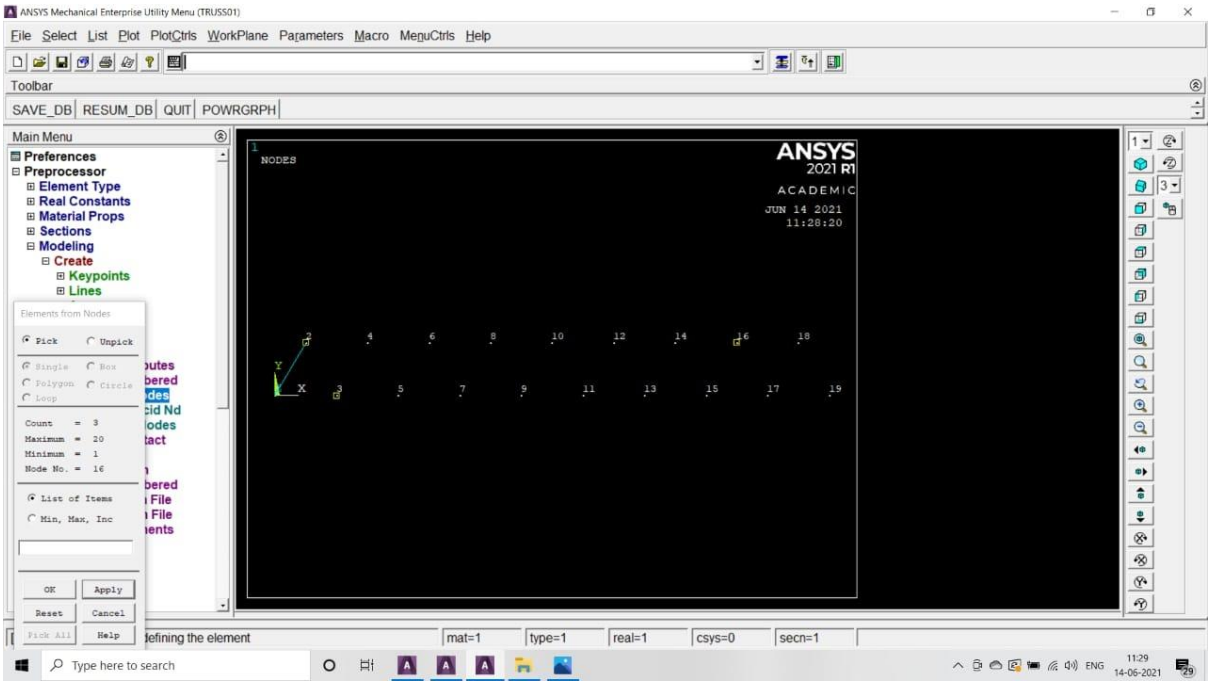


Fig 3.10 creating truss elements in between two nodes

Step 7:-Now on the main menu we select 'Loads' and Define the loads and click on Apply.Apply the'Structural' tool and select 'Displacement' and then 'On nodes'. Now select node 1 and clickOk . A pop-up window opens , the DOF'S needs to be constrained so we select All DOF'S and click Apply.

Then select node 7 → Ok → UY → Ok.

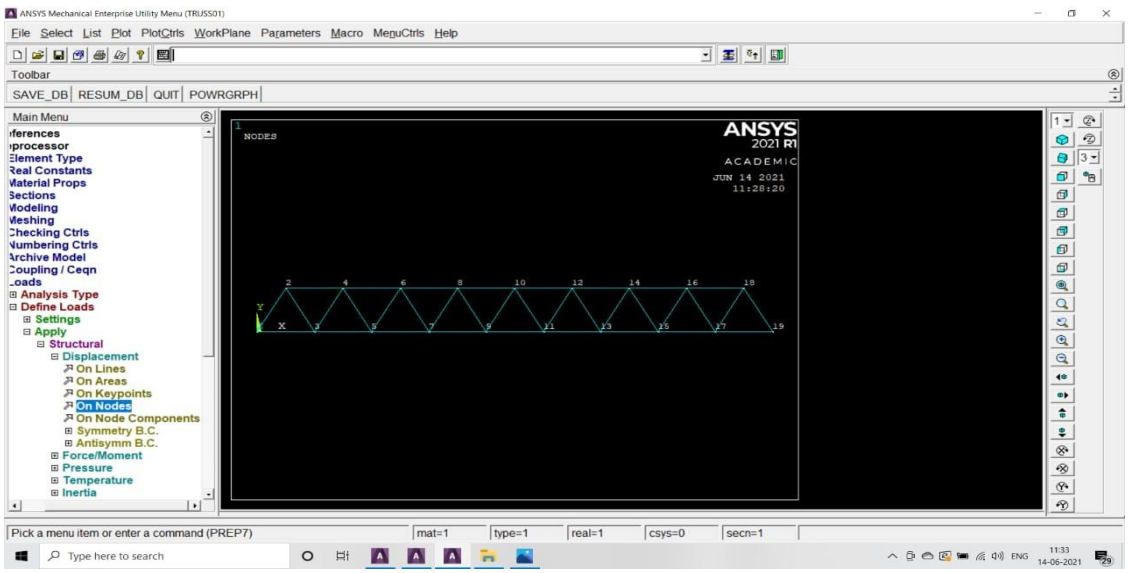


Fig 3.11 railway bridge truss in ansys interface

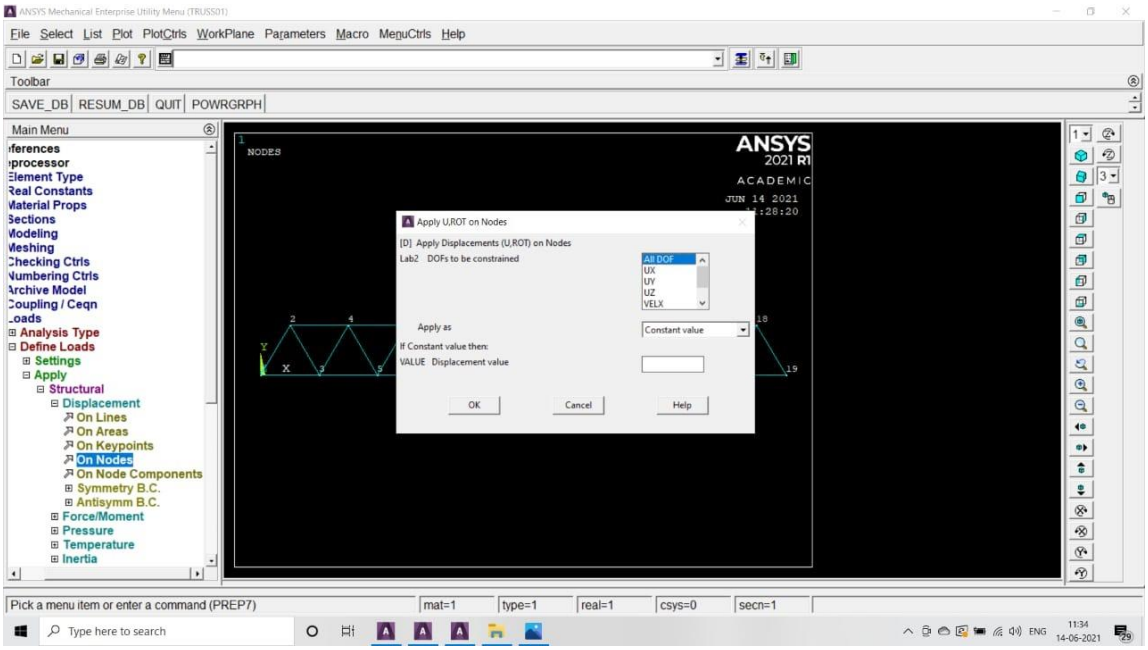


Fig 3.12 generate support reaction at nodes

Step 8:- Force / Moment → On nodes → Select Node 1 → Ok → FY → -280000 → Apply.

Select node 3 → Ok → FY → -210000 → Apply.

Select node 5 → Ok → FY → -210000 → Apply.

Select node 7 → Ok → FY → -210000 → Apply.

Select node 9 → Ok → FY → -210000 → Apply.

Select node 11 → Ok → FY → -210000 → Apply.

Select node 13 → Ok → FY → -210000 → Apply.

Select node 15 → Ok → FY → -210000 → Apply.

Select node 17 → Ok → FY → -210000 → Apply.

Select node 19 → Ok → FY → -360000 → Apply → Ok.

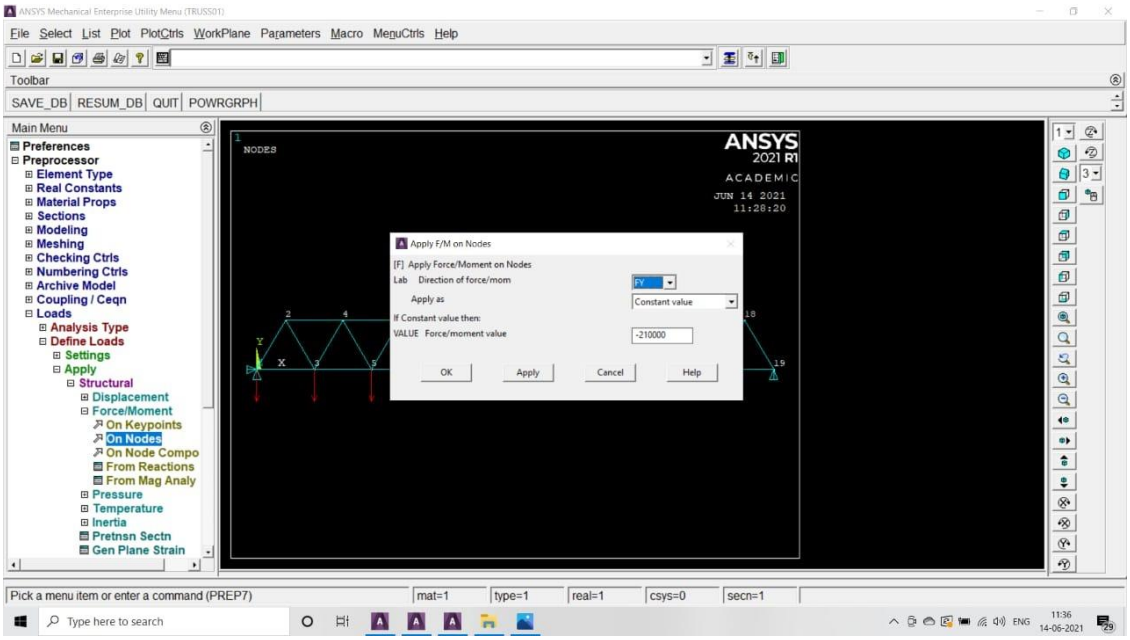


Fig 3.13 applying external loads on truss nodes

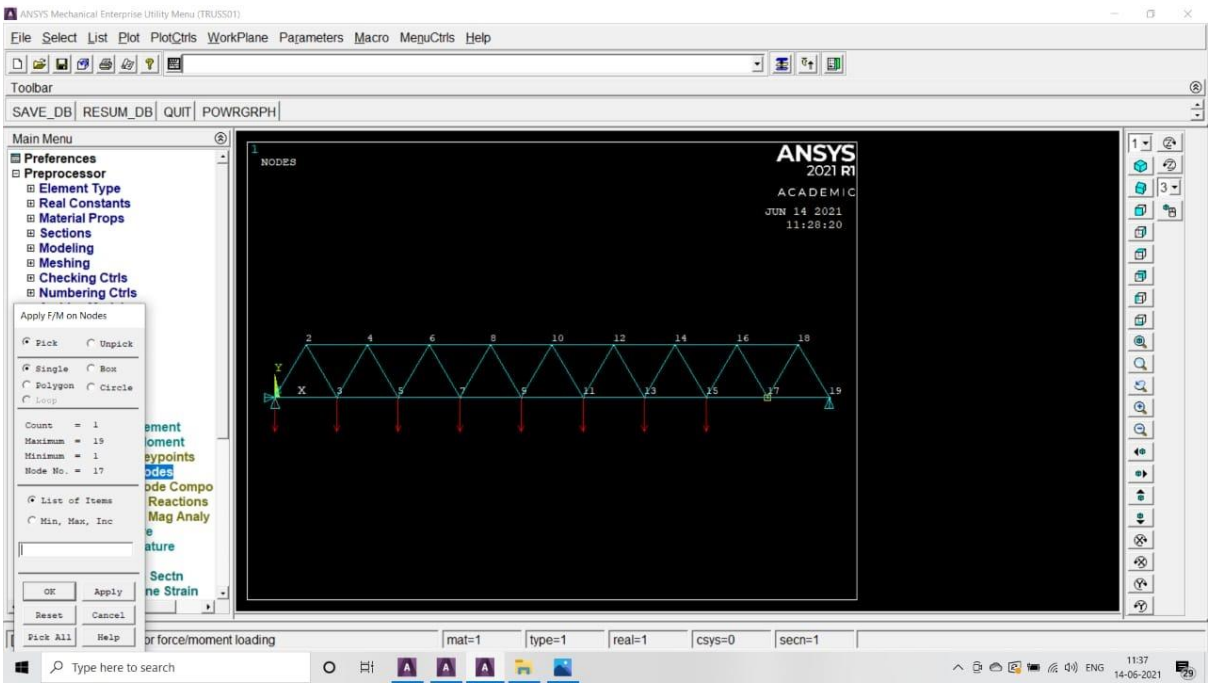


Fig 3.14 truss diagram with external loads

Step 9:- In ‘Solution’menu , we click on ‘Solve’and below it select‘Current LS’, a window will open showing solution options. Now press Ok to start the Solution . it finally shows ‘Solution is done’ and then we ‘Close’.

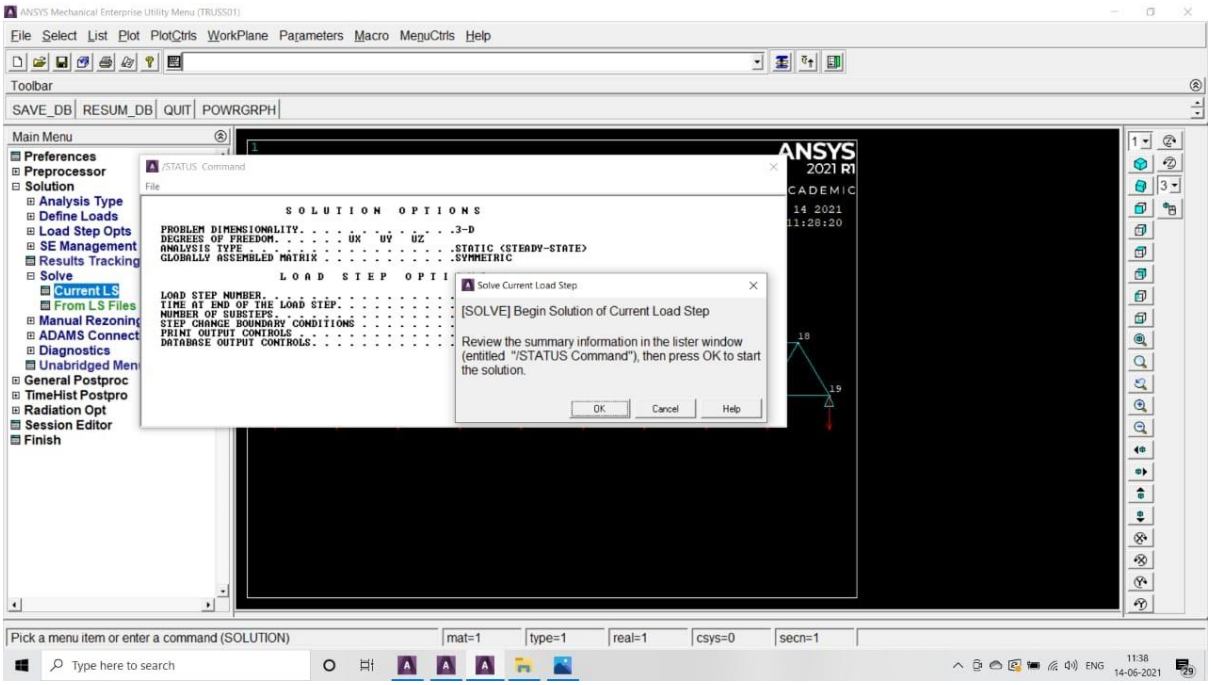


Fig 3.15 solving truss in current load step

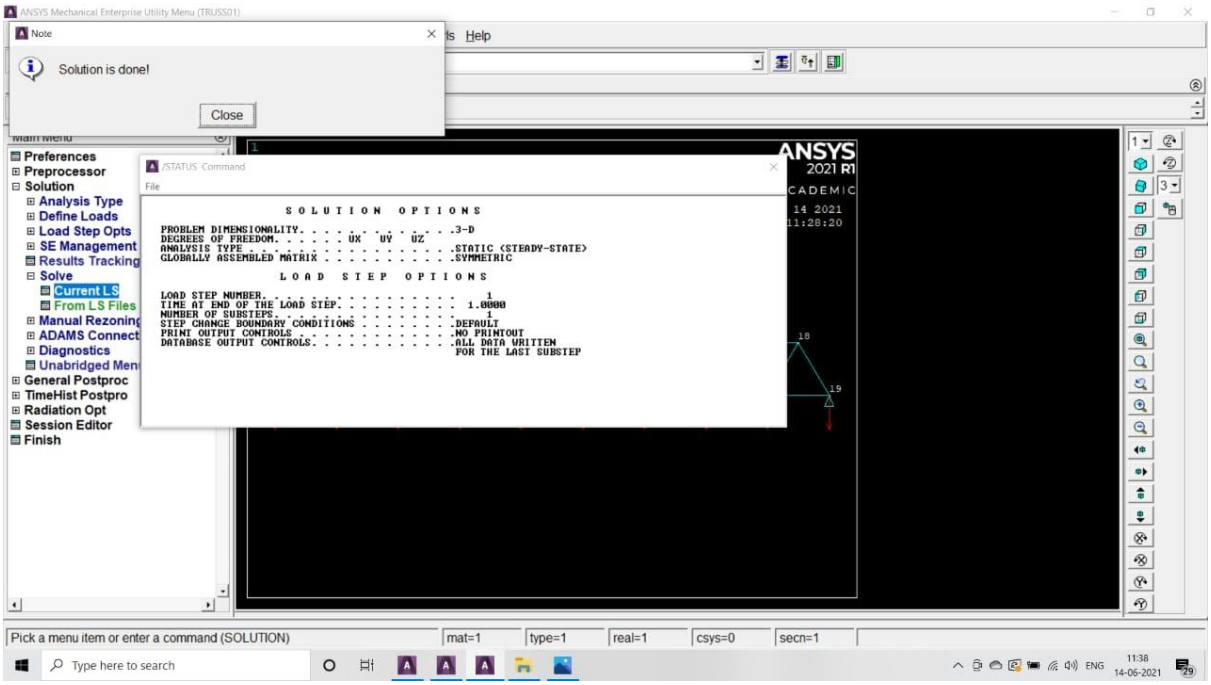


Fig 3.16 solution truss in current load step

Step 10:-In the above tool bar click on ‘Plot ctrlS’and select ‘Style’ in which we aagain select ‘Size & Shape’ and finally Display of element is done and click Ok.

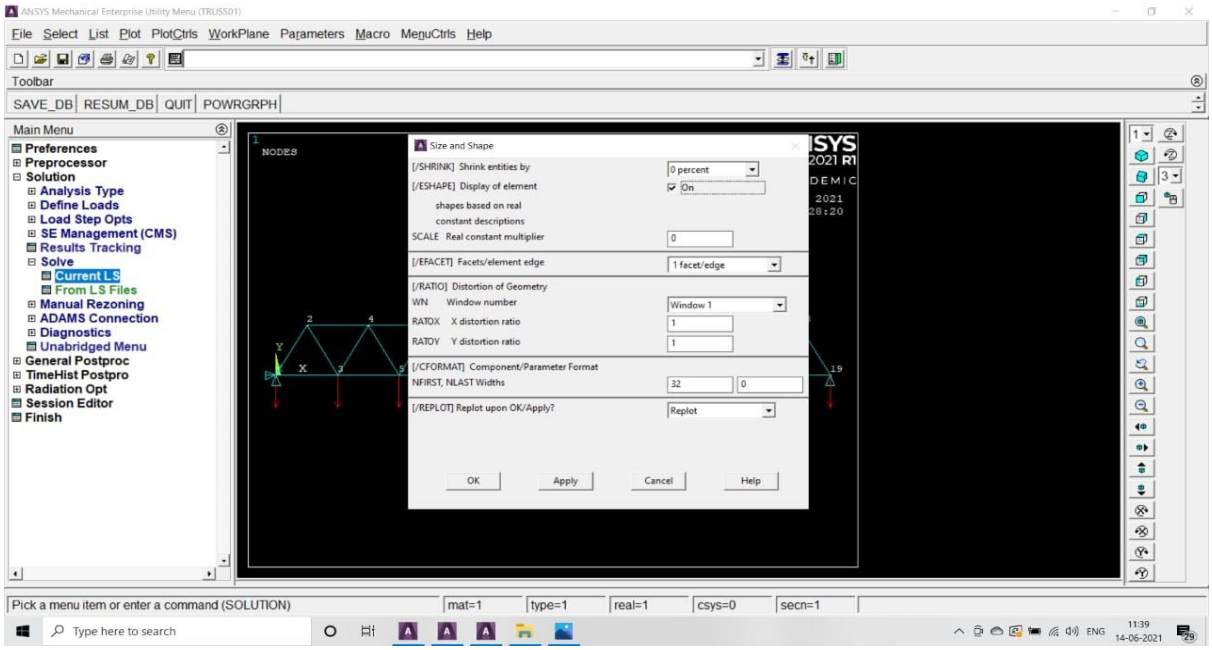


Fig 3.17 selection of style,size &shape

Step 11:-We click on **‘**Result viewer’ from menu and after that we chose a result item and select ‘DOF solution’then we check the ‘Displacement vector sum ‘ and finally Plot result.

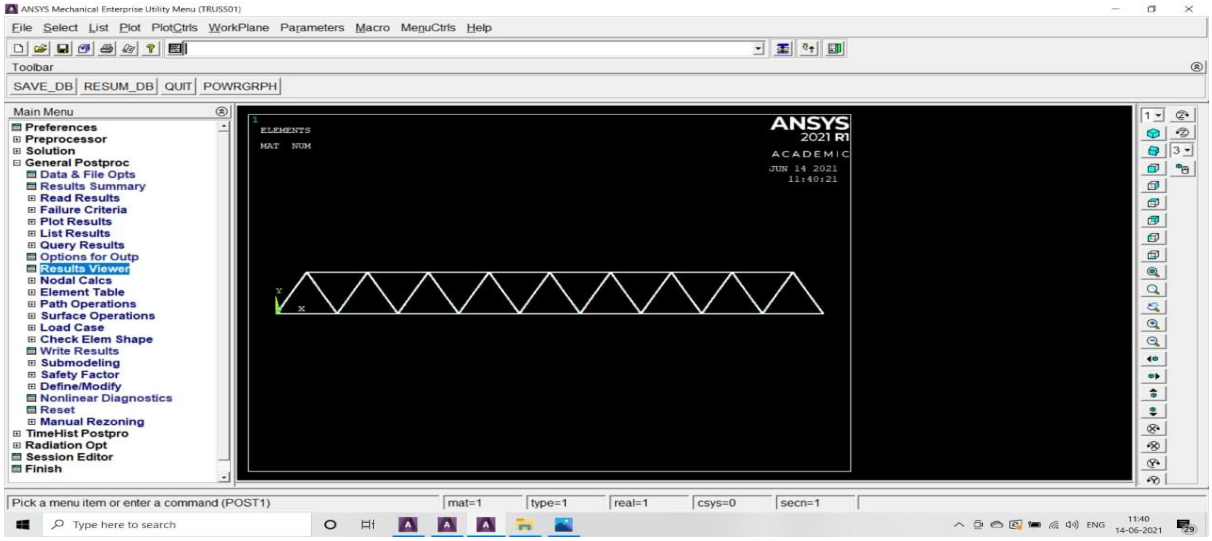


Fig 3.18 result viewer in ansys interface

Step 12:-Here we check the Stress loading on the truss structure and then we checkVon misses stress shown below and Plot the results.

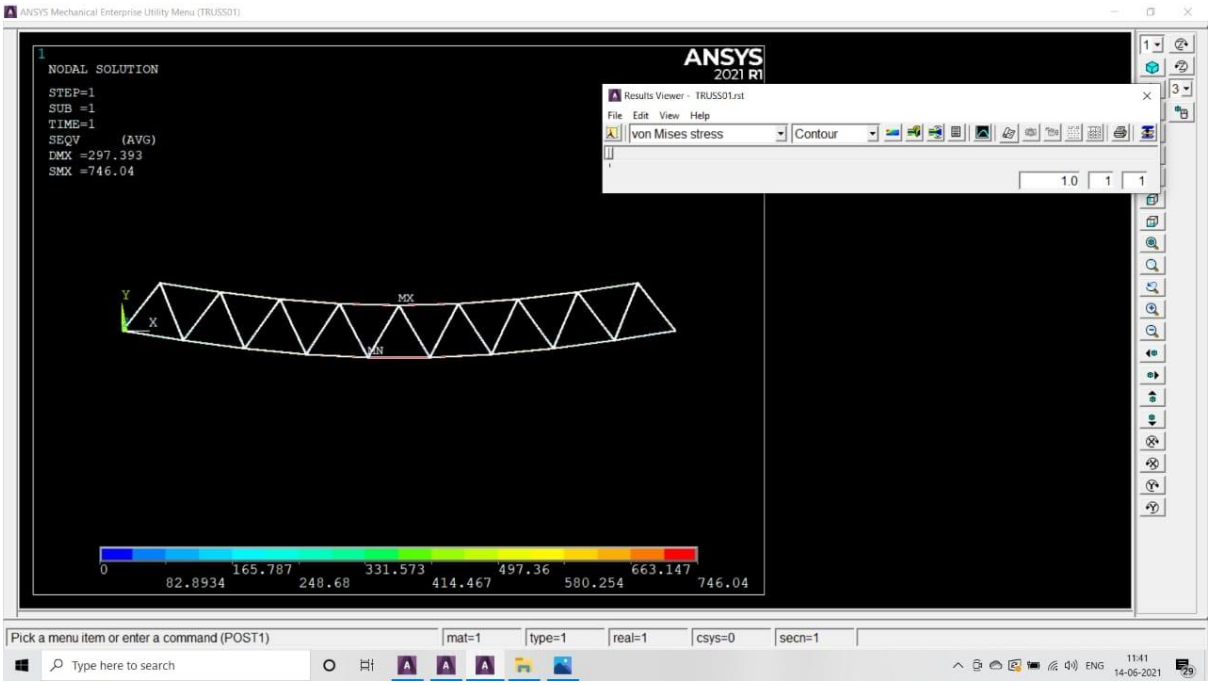


Fig 3.19 von mises stress

Step 13:-General post proc → Plot results → Deformed shape →Def+Undeformed→ Ok.

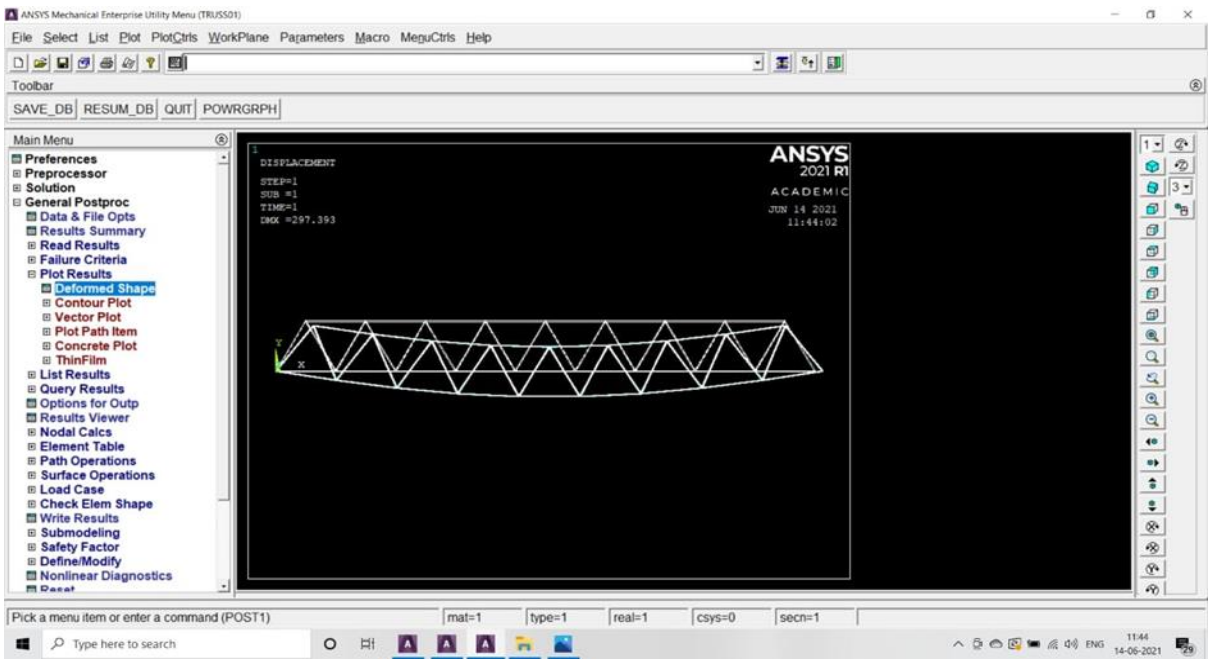


Fig 3.20 deformed shape of truss

Step 14:-At last we select ‘List results’ and under that we click on ‘Reaction solu’ which shows All items and total reaction solu listing and click Ok.Nodal

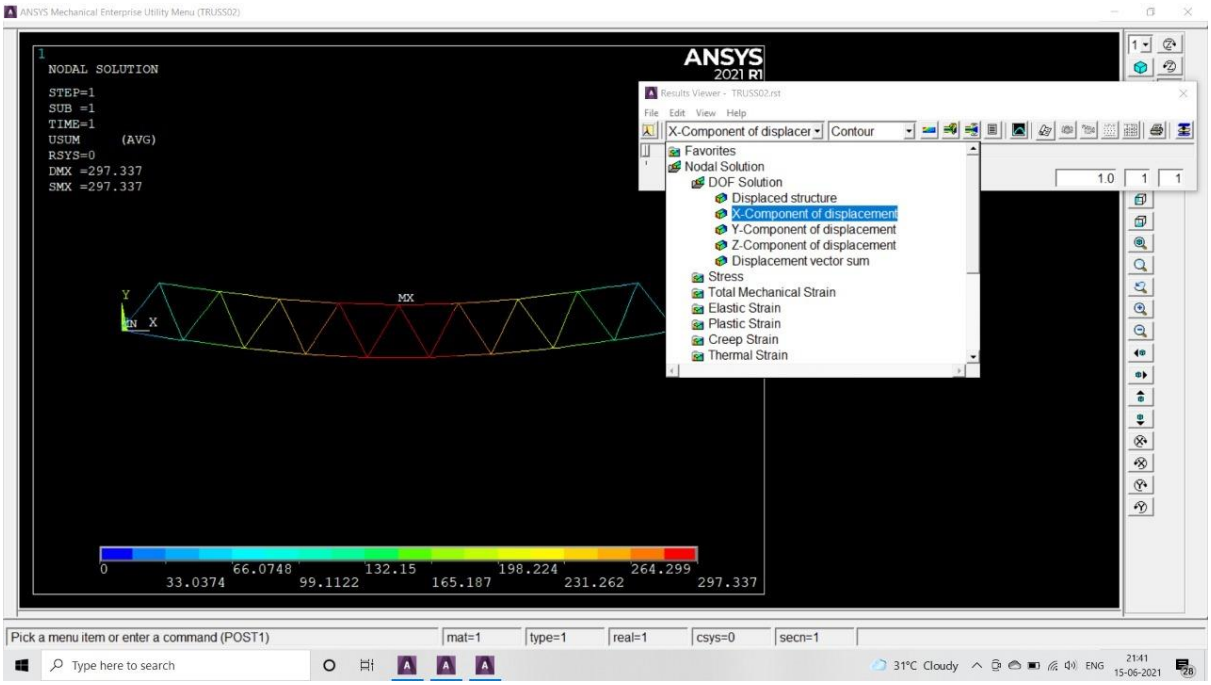


Fig 3.21 nodel displacement in x-componet

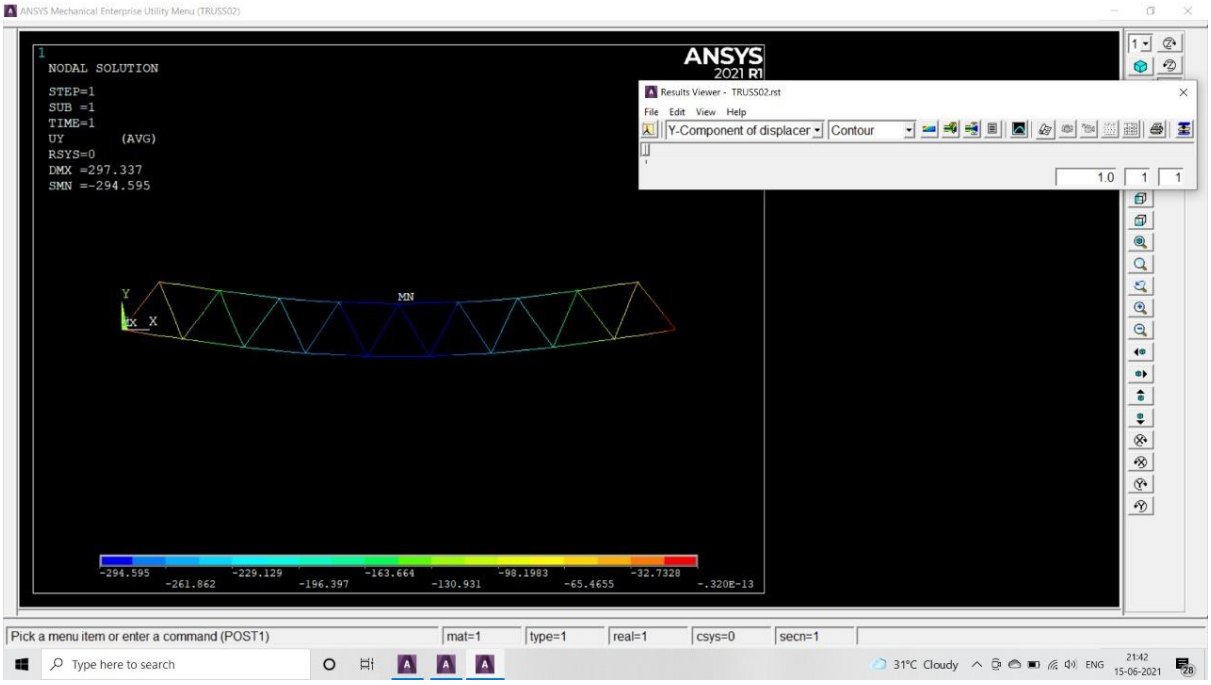


Fig 3.22 nodel displacement in y-component

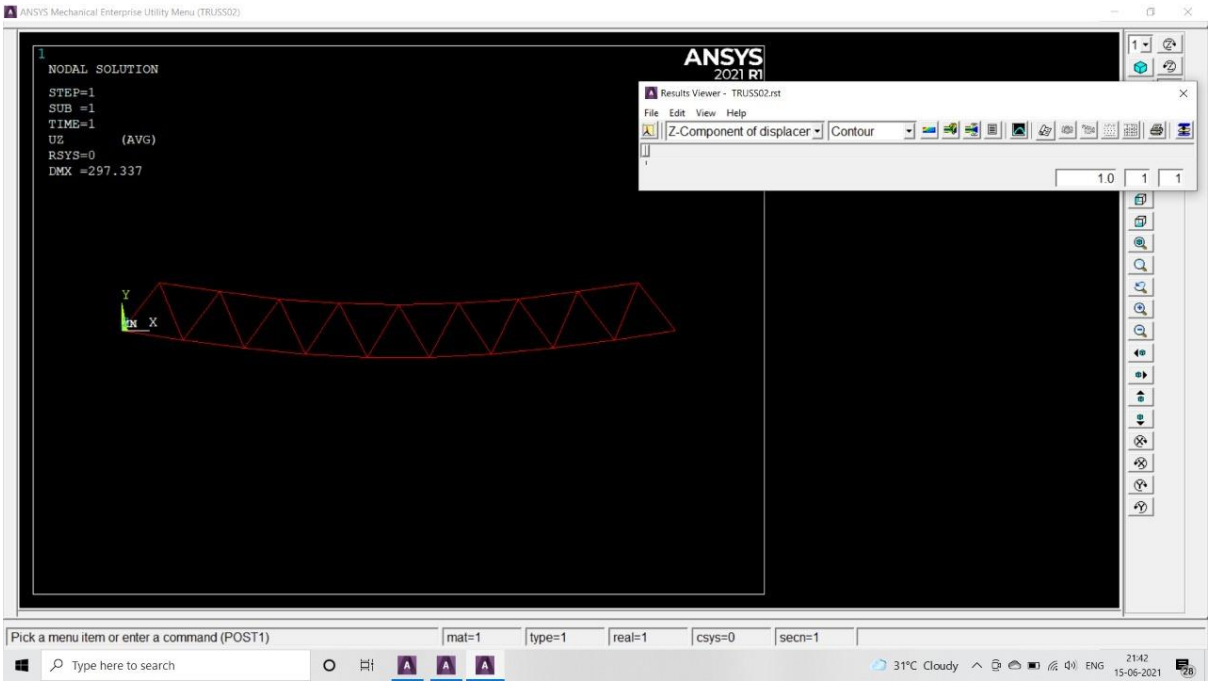


Fig 3.23 nodel displacement in z-component

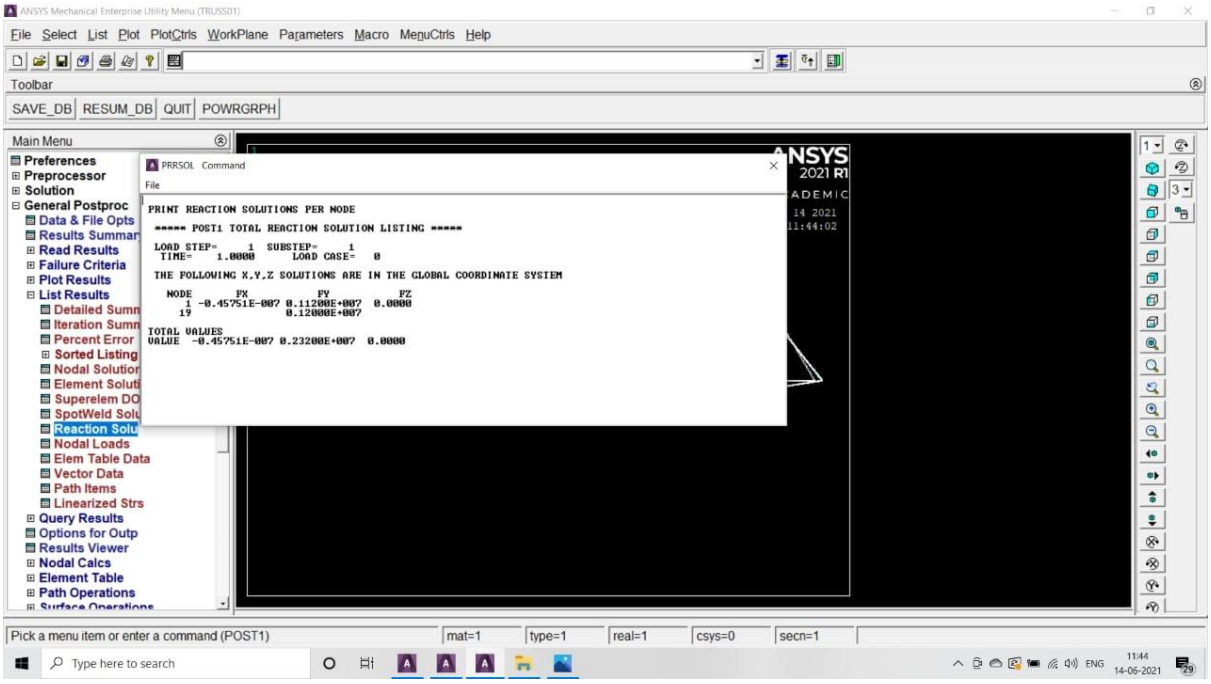


Fig 3.24 nodel solution at global coordinate system

3.3.1 Nodal displacement results

PRINT U NODAL SOLUTION PER NODE

***** POST1 NODAL DEGREE OF FREEDOM LISTING *****

LOAD STEP= 1 SUBSTEP= 1
TIME= 1.0000 LOAD CASE= 0

THE FOLLOWING DEGREE OF FREEDOM RESULTS ARE IN THE GLOBAL COORDINATE SYSTEM

NODE	UX	UY	UZ	USUM
1	0.0000	0.0000	0.0000	0.0000
2	80.572	-52.717	0.0000	96.286
3	2.6857	-103.88	0.0000	103.92
4	75.201	-150.40	0.0000	168.15
5	10.072	-192.65	0.0000	192.91
6	65.801	-227.92	0.0000	237.23
7	20.815	-257.00	0.0000	257.84
8	53.715	-277.54	0.0000	282.69
9	33.572	-290.72	0.0000	292.65
10	40.286	-294.59	0.0000	297.34
11	47.001	-290.72	0.0000	294.49
12	26.857	-277.54	0.0000	278.84
13	59.758	-257.00	0.0000	263.85
14	14.772	-227.92	0.0000	228.40
15	70.501	-192.65	0.0000	205.14
16	5.3715	-150.40	0.0000	150.49
17	77.887	-103.88	0.0000	129.84
18	0.79378E-012	-52.717	0.0000	52.717
19	80.572	0.0000	0.0000	80.572

MAXIMUM ABSOLUTE VALUES				
NODE	2	10	0	10
VALUE	80.572	-294.59	0.0000	297.34

3.3.2 nodal stress results

PRINT S NODAL SOLUTION PER NODE

***** POST1 NODAL STRESS LISTING *****

LOAD STEP= 1 SUBSTEP= 1
TIME= 1.0000 LOAD CASE= 0

NODE	S1	S2	S3	SINT	SEQV
1	0.0000	0.0000	-74.615	74.615	74.615
2	0.0000	0.0000	-99.472	99.472	99.472
3	158.53	0.0000	0.0000	158.53	158.53
4	0.0000	0.0000	-205.16	205.16	205.16
5	270.44	0.0000	0.0000	270.44	270.44

6	0.0000	0.0000	-298.42	298.42	298.42
7	345.05	0.0000	0.0000	345.05	345.05
8	0.0000	0.0000	-354.37	354.37	354.37
9	382.35	0.0000	0.0000	382.35	382.35
10	0.0000	0.0000	-373.02	373.02	373.02
11	382.35	0.0000	0.0000	382.35	382.35
12	0.0000	0.0000	-354.37	354.37	354.37
13	345.05	0.0000	0.0000	345.05	345.05
14	0.0000	0.0000	-298.42	298.42	298.42
15	270.44	0.0000	0.0000	270.44	270.44
16	0.0000	0.0000	-205.16	205.16	205.16
17	158.53	0.0000	0.0000	158.53	158.53
18	0.0000	0.0000	-99.472	99.472	99.472
19	0.0000	0.0000	-74.615	74.615	74.615

MINIMUM VALUES					
NODE	1	1	10	1	1
VALUE	0.0000	0.0000	-373.02	74.615	74.615

MAXIMUM VALUES					
NODE	9	1	3	9	9
VALUE	382.35	0.0000	0.0000	382.35	382.35

3.3.3 ELEMENTARY STRESS RESULTS

PRINT ELEMENT TABLE ITEMS PER ELEMENT

STAT ELEM	CURRENT SAXL
1	149.21
2	410.32
3	596.83
4	708.74
5	746.04
6	708.74
7	596.83
8	410.32
9	149.21
10	-298.42
11	-522.23
12	-671.44
13	-746.04
14	-746.04
15	-671.44
16	-522.23
17	-298.42
18	-298.44
19	298.44
20	-223.83
21	223.83
22	-149.22
23	149.22
24	-74.610
25	74.610
26	0.0000
27	0.27756E-011

```
        28          74.610
        29          -74.610
        30          149.22
        31          -149.22
        32          223.83
        33          -223.83
        34          298.44
        35          -298.44
MINUMUM VALUES
ELEM             13
VALUE          -746.04
MAXIMUM VALUES
ELEM              5
VALUE           746.04
```

3.4 MODAL ANALYSIS OF TRUSS

Software used:- ANSYS 2021 R1.

Presteps:-Simulation environment: ANSYS

File management :working directory:-C\ANSYS\TRUSS

Job Name:-TRUSS01< Run

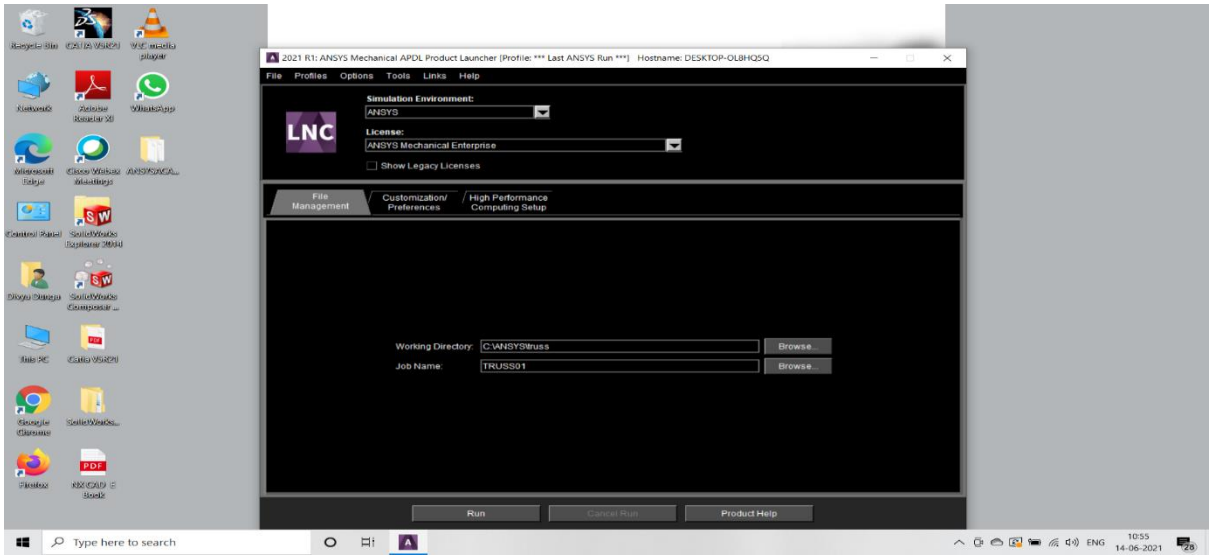


Fig 3.25 ansys file

Step 1:-firstly, we click on Preferences in main menu and then we select the type of analysis by choosing structural and proceed on by clicking Ok

.

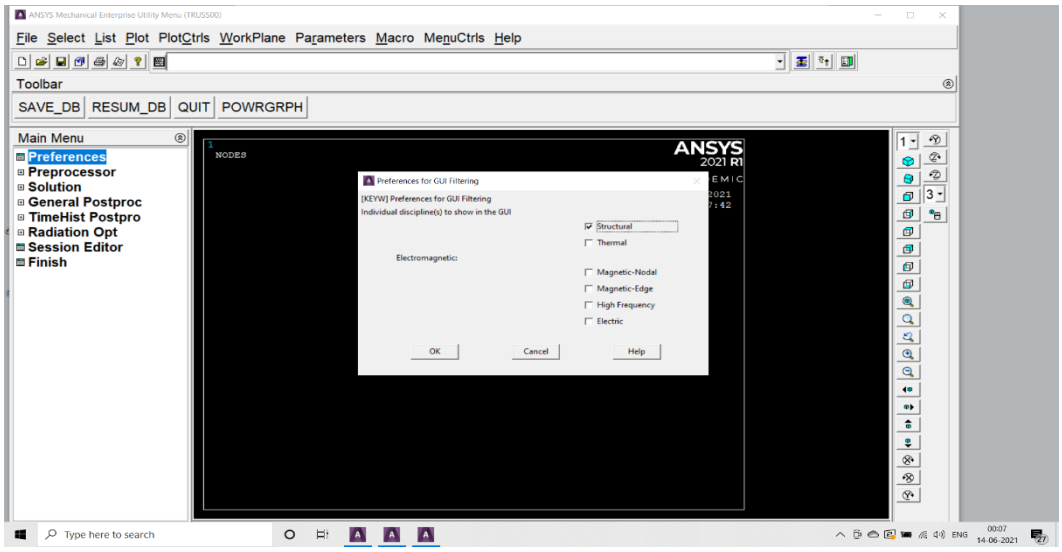


Fig 3.26 ansys preprocessor

Step 2:-Now considering preprocessor we select the Element type , and click on Add/Edit/Delete. We add link and select Link 3D finitstn 180 as element type.

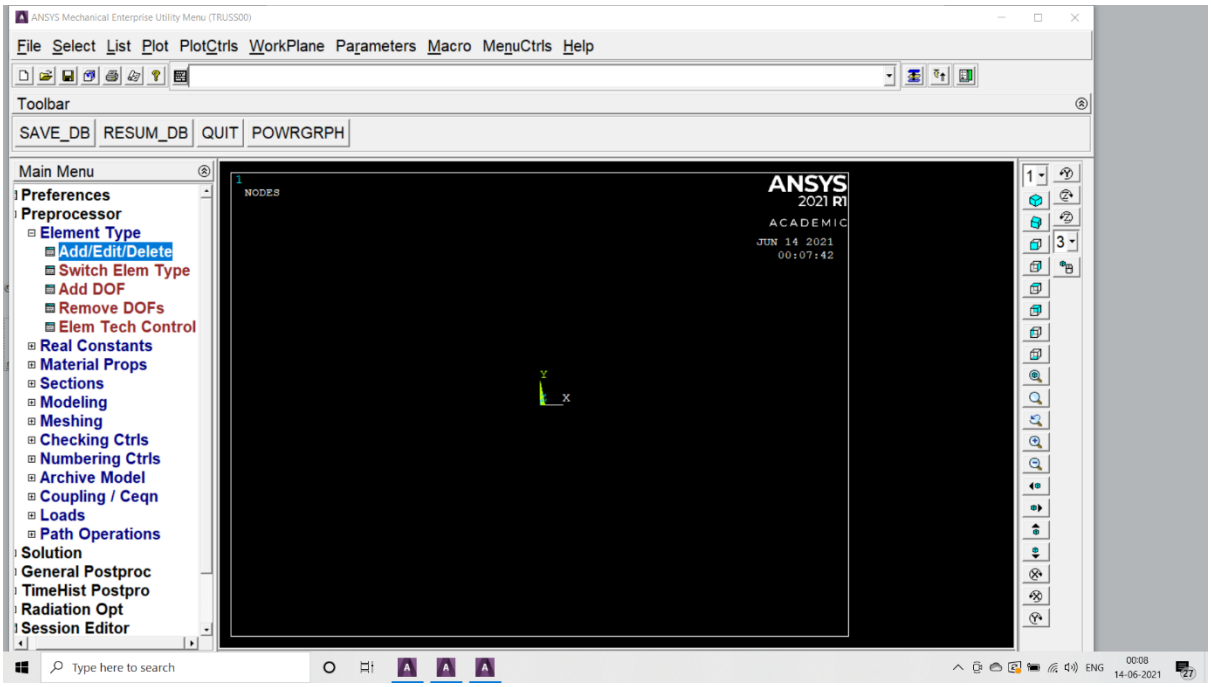


Fig 3.27 ansys element types

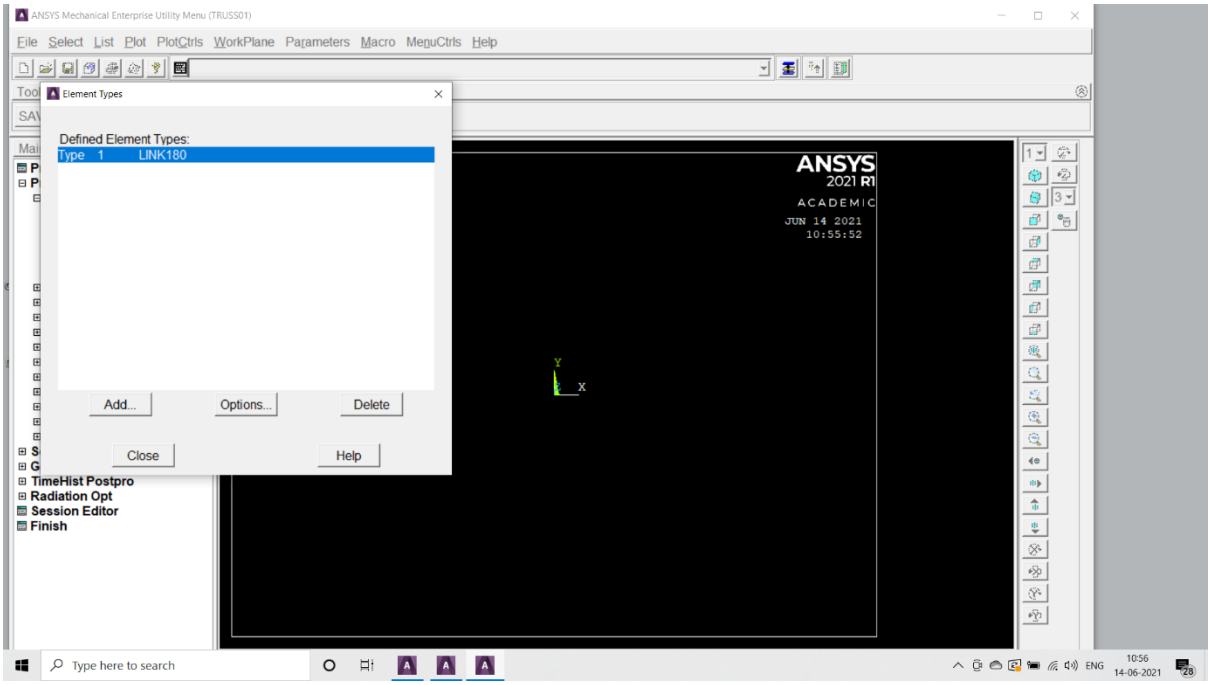


Fig 3.28 selection of link 180 element

Step 3:-In this we choose the Material props and Select the available Material models under Structural→Linear→Elastic→Isotropic and click Ok.

We assign the material property values for the material 1. Ex= 200GPA ,PRXY=0.3 → Ok

The material taken in steel

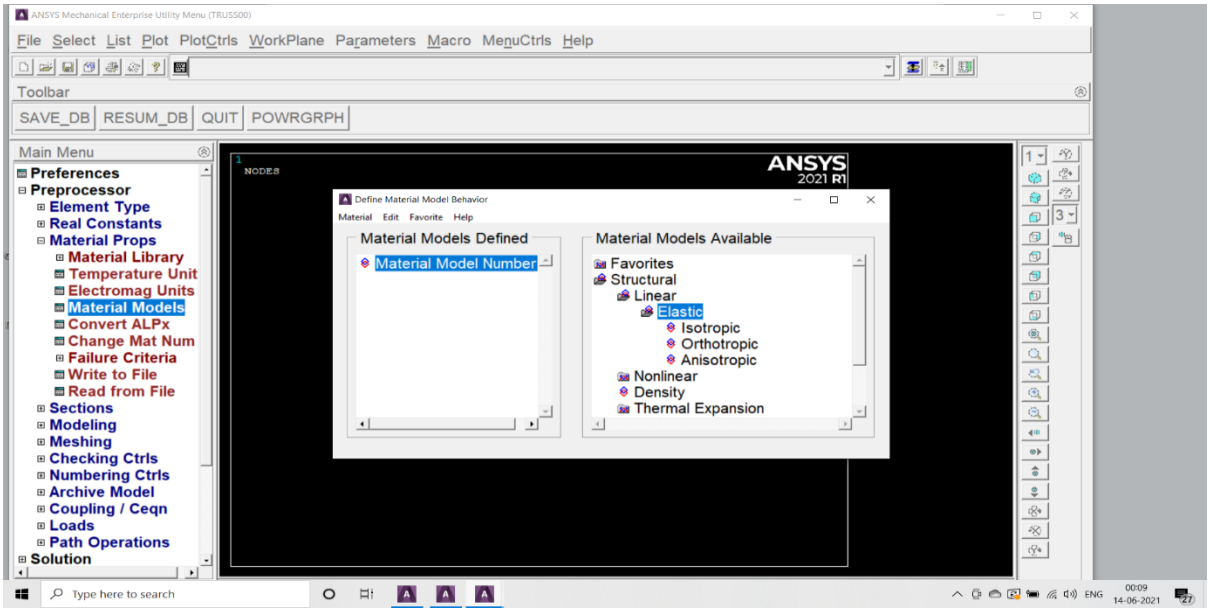


Fig 3.29 material in steel

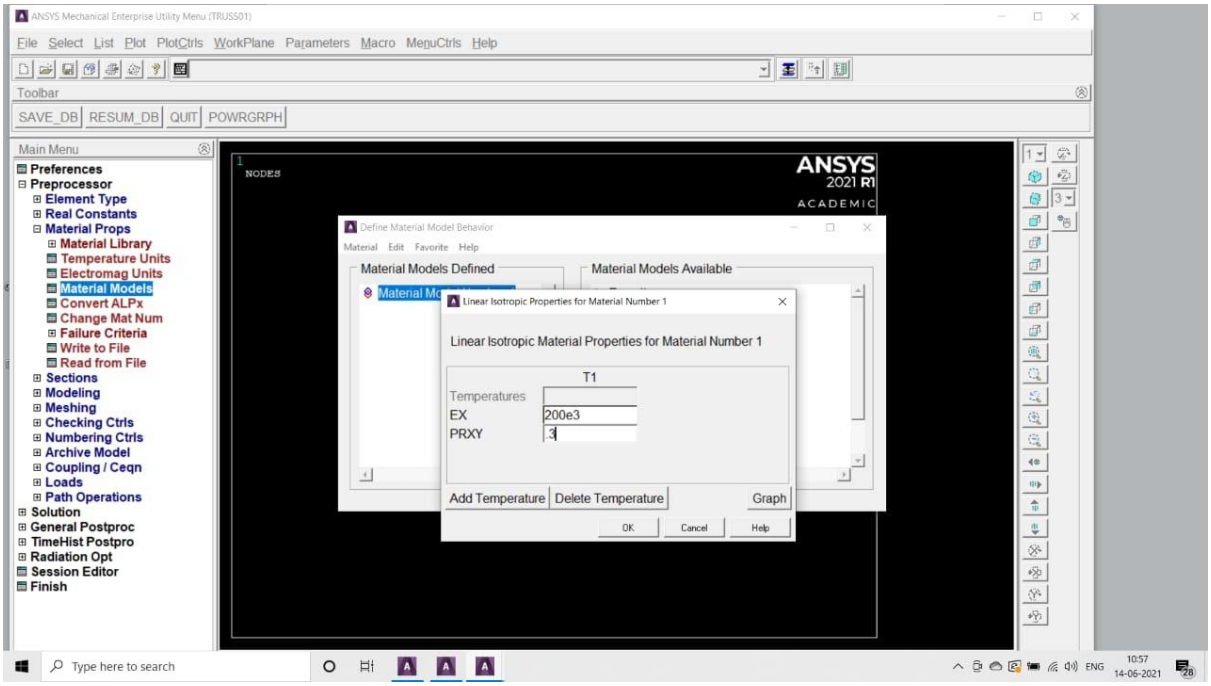


Fig 3.30 material property selection

Step 4:- We go toSections and under it we click on Link and then Add. We take the add link section with ID as 1.

And , we name the section as ‘Rect’ and take the Link area as ‘3250’ and click Ok.

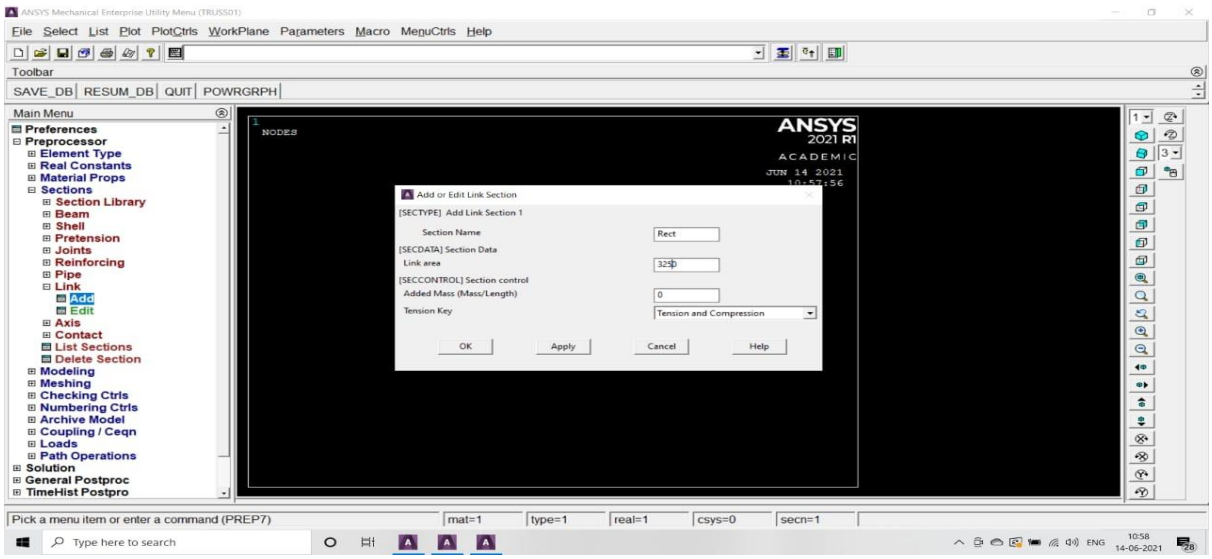


Fig 3.31 adding link section and link area

Step 5:-Now click on Modelling and select Create. We can see other functions below, select Nodes and then click on InactiveCS. A window will appear as shown in belowfig ,and we can choose the nodes and assign the values for the nodes.

Node number 1 at X = 0 , Y = 0 , Z = 0 → Apply

Node number 2 at X = 1800 , Y = 3118 , Z = 0 → Apply

Node number 3 at X = 3600 , Y = 0 , Z = 0 → Apply

Node number 4 at X = 5400 , Y = 3118 , Z = 0 → Apply

Node number 5 at X = 7200 , Y = 0 , Z = 0 → Apply

Node number 6 at X = 9000 , Y = 3118 , Z = 0 → Apply

Node number 7 at X = 10800 , Y = 0 , Z = 0 → Apply

Node number 8 at X = 12600 , Y = 3118 , Z = 0 → Apply

Node number 9 at X = 14400 , Y = 0 , Z = 0 → Apply

Node number 10 at X = 16200 , Y = 3118 , Z = 0 → Apply

Node number 11 at X = 18000 , Y = 0 , Z = 0 → Apply

Node number 12 at X = 19800 , Y = 3118 , Z = 0 → Apply

Node number 13 at X = 21600 , Y = 0 , Z = 0 → Apply

Node number 14 at X = 23400 , Y = 3118 , Z = 0 → Apply

Node number 15 at X = 25200 , Y = 0 , Z = 0 → Apply

Node number 16 at X = 27000 , Y = 3118 , Z = 0 → Apply

Node number 17 at X = 28800 , Y = 0 , Z = 0 → Apply

Node number 18 at X = 30600 , Y = 3118 , Z = 0 → Apply

Node number 19 at X = 32400 , Y = 0 , Z = 0 → Apply→ Ok

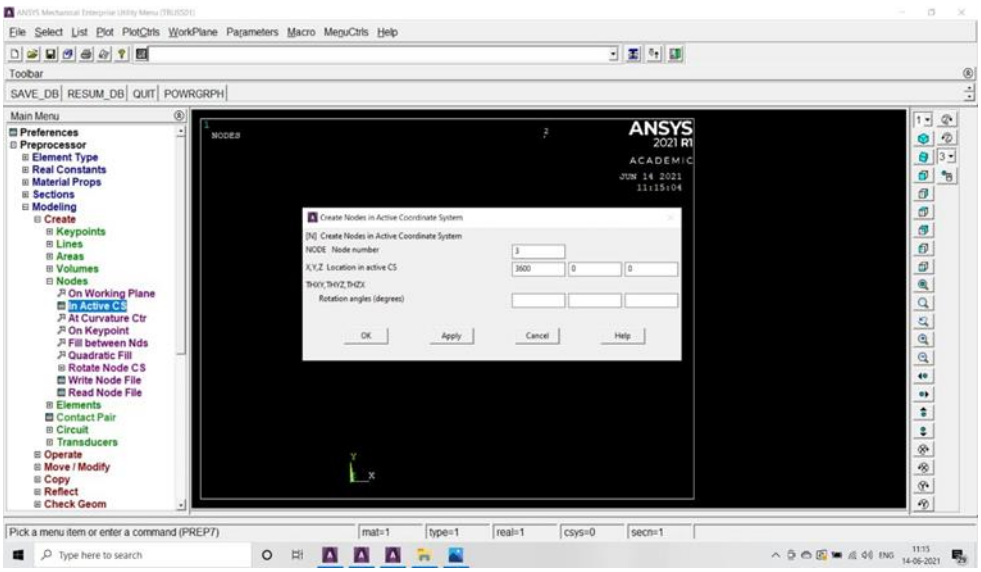


Fig 3.32 creating truss nodes in active coordinating system

Step 6:-We click on Element tool and under that we click on Auto numbered and below that we select Thru nodes and Select the nodes 1&2 and click on Apply . In the same way we select thenodes 1&3 and click on Apply.

We Repeat the same procedure.

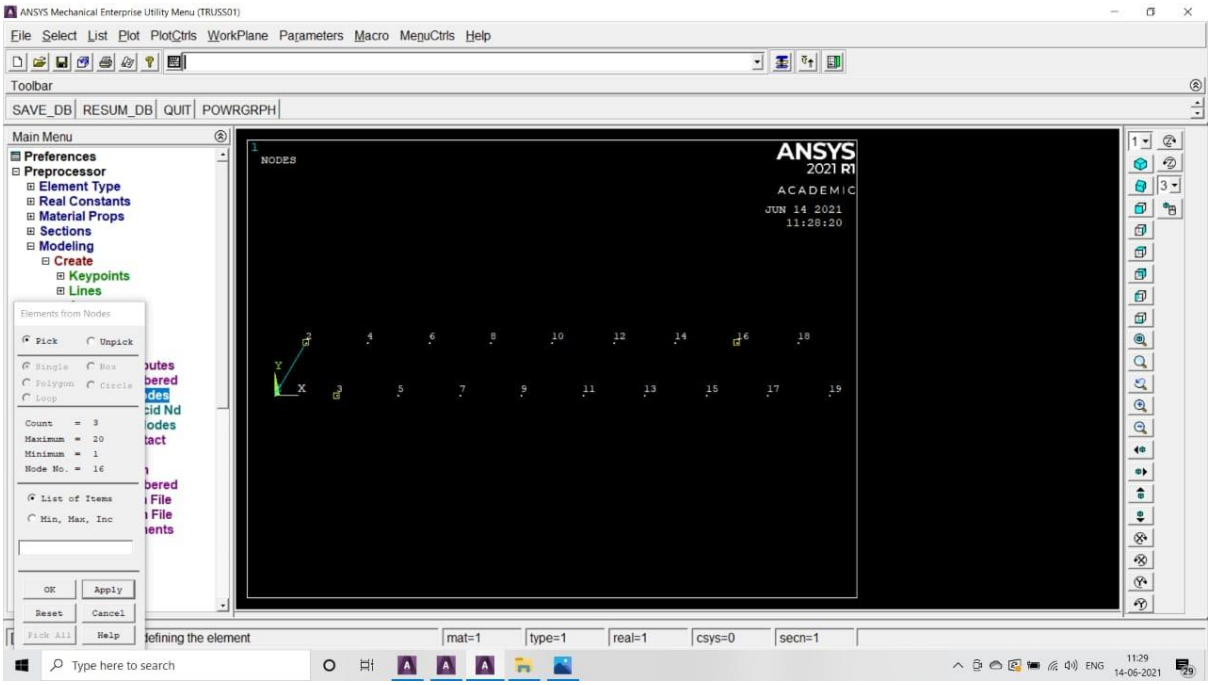


Fig 3.33 creating truss elements in between two nodes

Step-7:-Click on ‘Solution’ in main menu and select the ‘Analysis type’ as ‘New Analysis’ and then we select ‘modal’ and click Ok.

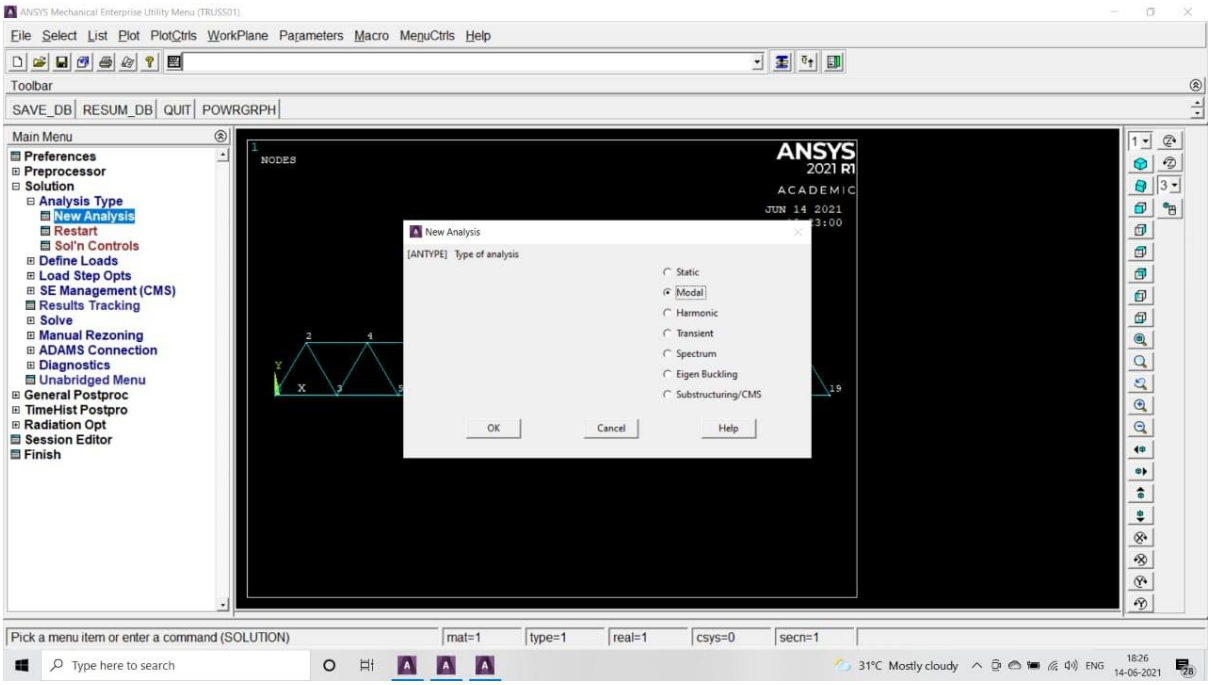


Fig 3.34 model analysis selection

Step-8:-Now we click on ‘Analysis option’ and then we select the mode extraction method as ‘Block lanczos’ we assign tthe No of modes to extract as 40 and NMOD E no of modes to expand- as and click Ok.

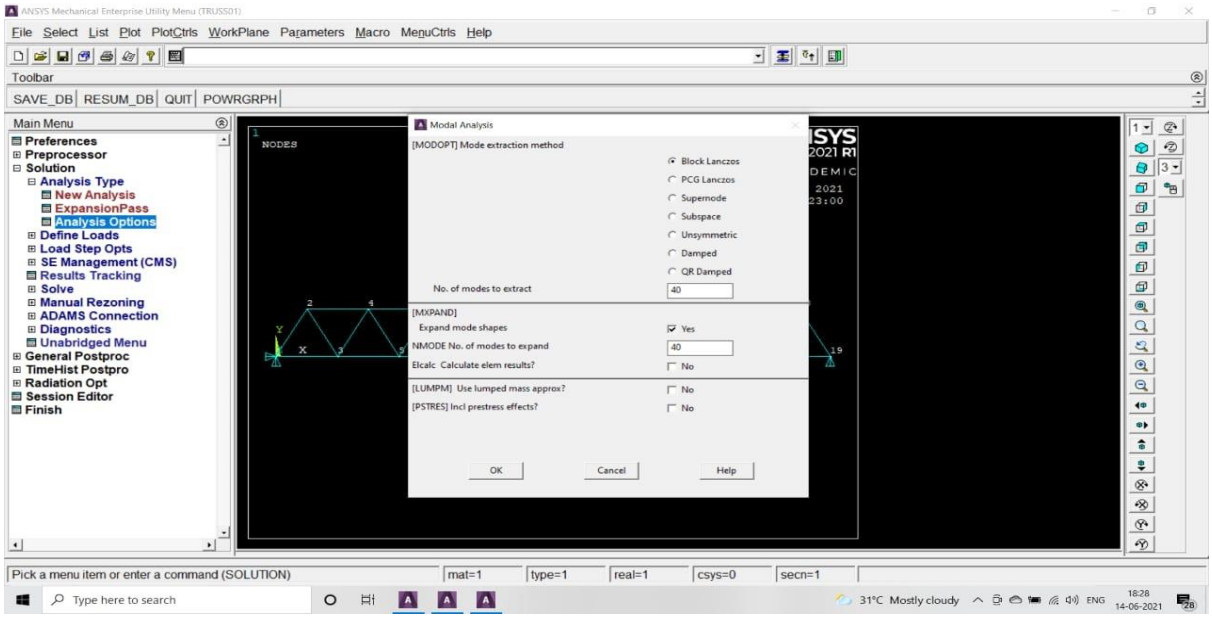


Fig 3.35 modl extraction method

Step-9:-In this step we assign the value 0 for FREQB start Freq(initial shift) and for FREQE End Freq as 0 and click on Ok.

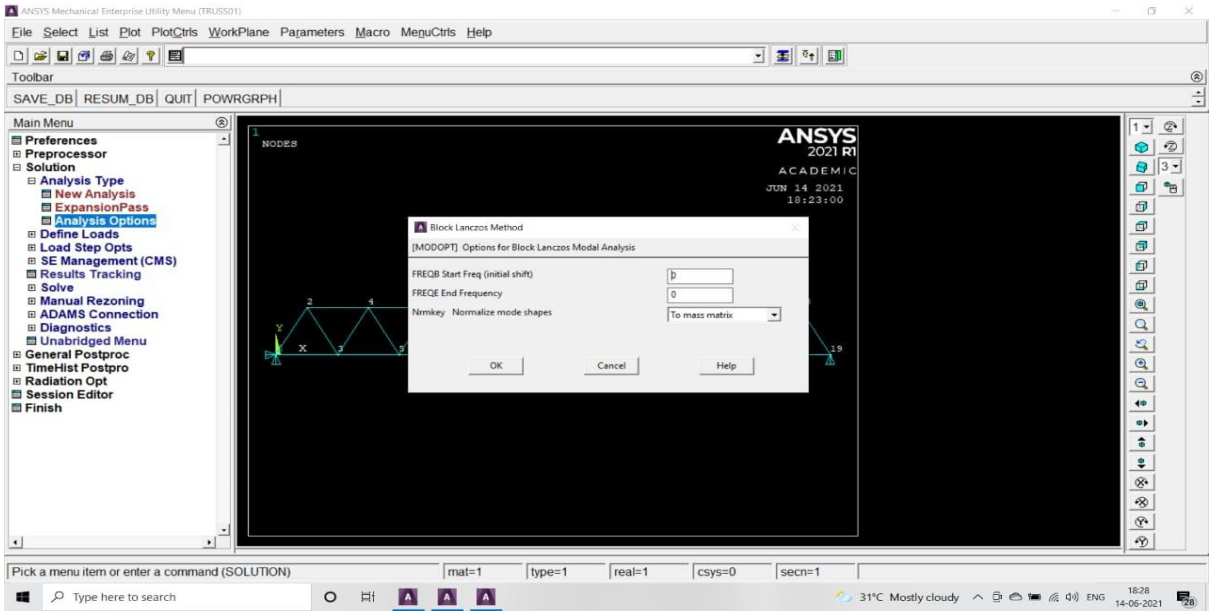


Fig 3.36 options in model analysis

Step-10:-Now on the main menu we select 'Loads' and Define the loads and click on Apply. Apply the 'Structural' tool and select 'Displacement' and then 'On nodes'. Now select node 1 and click Ok . A pop-up window opens , the DOF'S needs to be constrained so we select All DOF'S and click Apply.

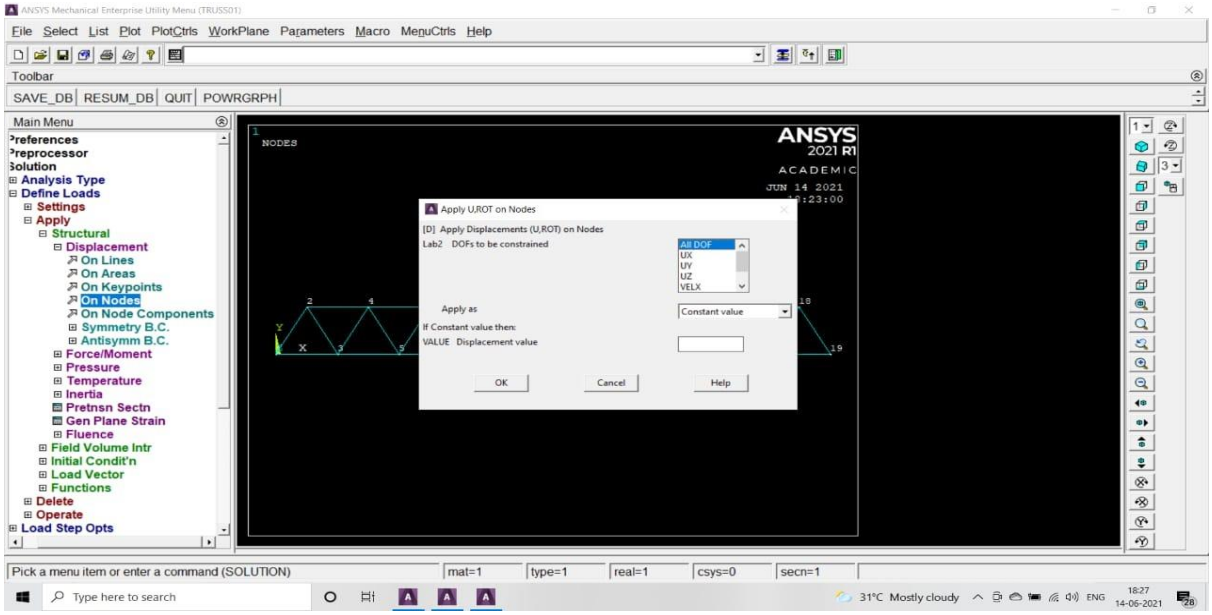


Fig 3.37 apply displacements on nodes

Step-11:- In ‘Solution’ menu , we click on ‘Solve’ and below it select ‘Current LS’ , a window will open showing solution options. Now press Ok to start the Solution . it finally shows ‘Solution is done’ and then we ‘Close’.

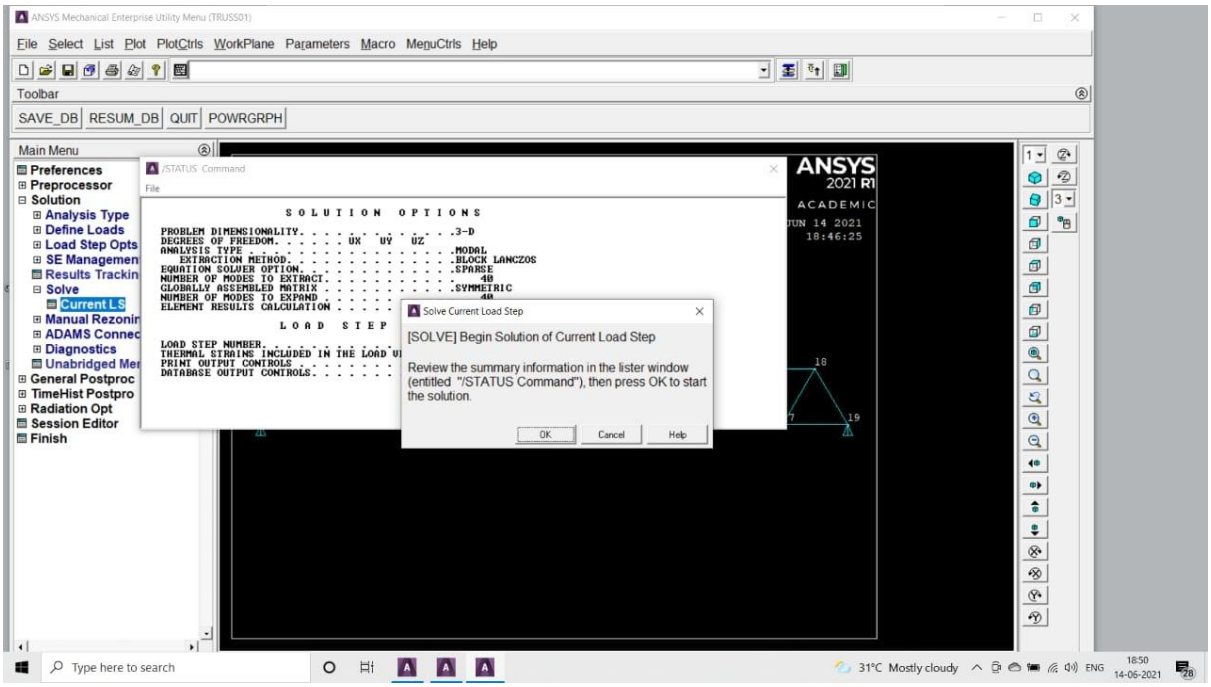


Fig 3.38solving the truss in current load step

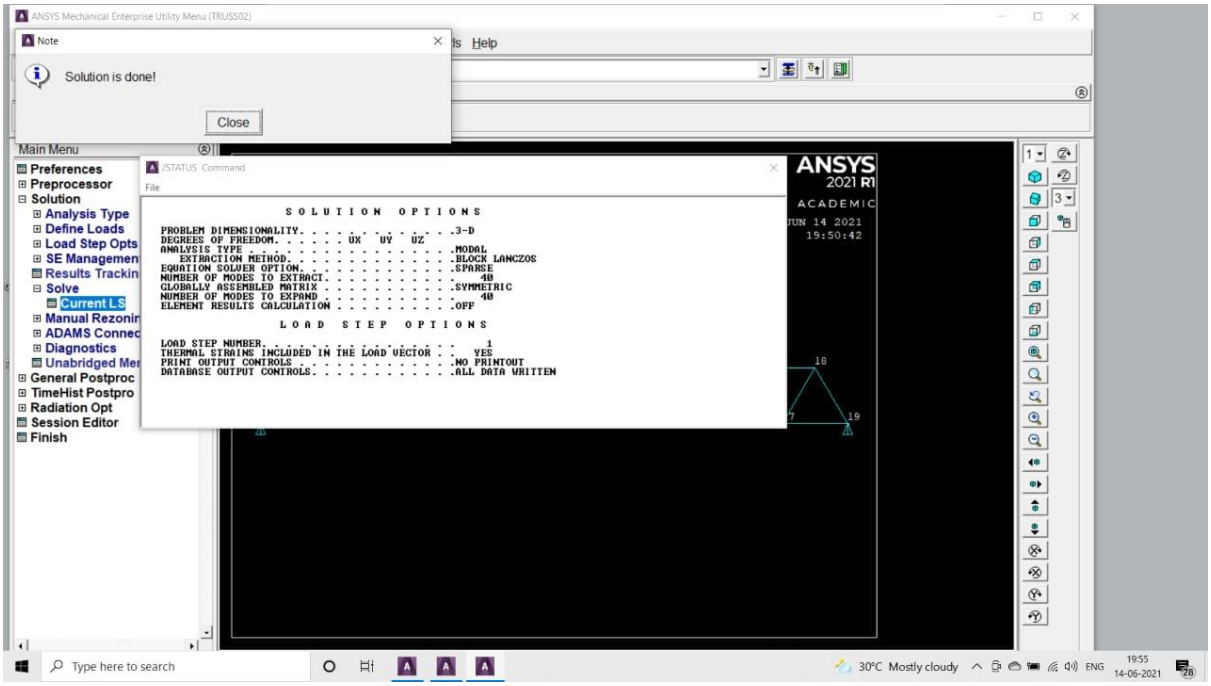
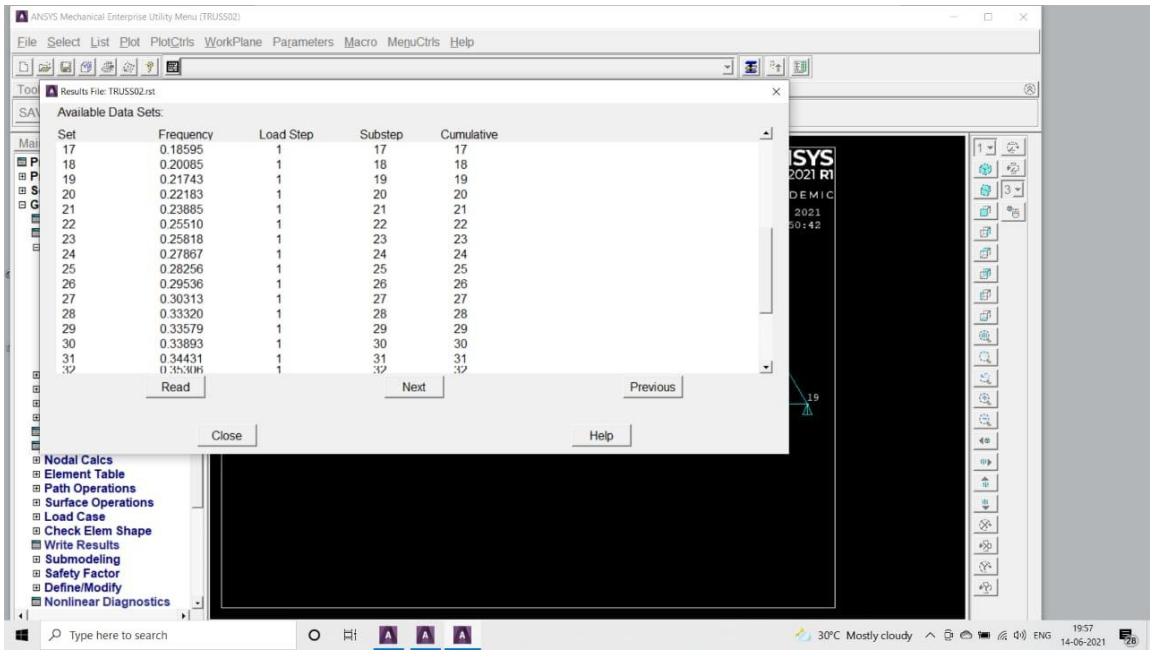


Fig 3.39 solution of truss in current load step

Step-12:-We click on General post proc and select ‘Read results’we see the results ‘By pick’ and select 4result and click onClose.



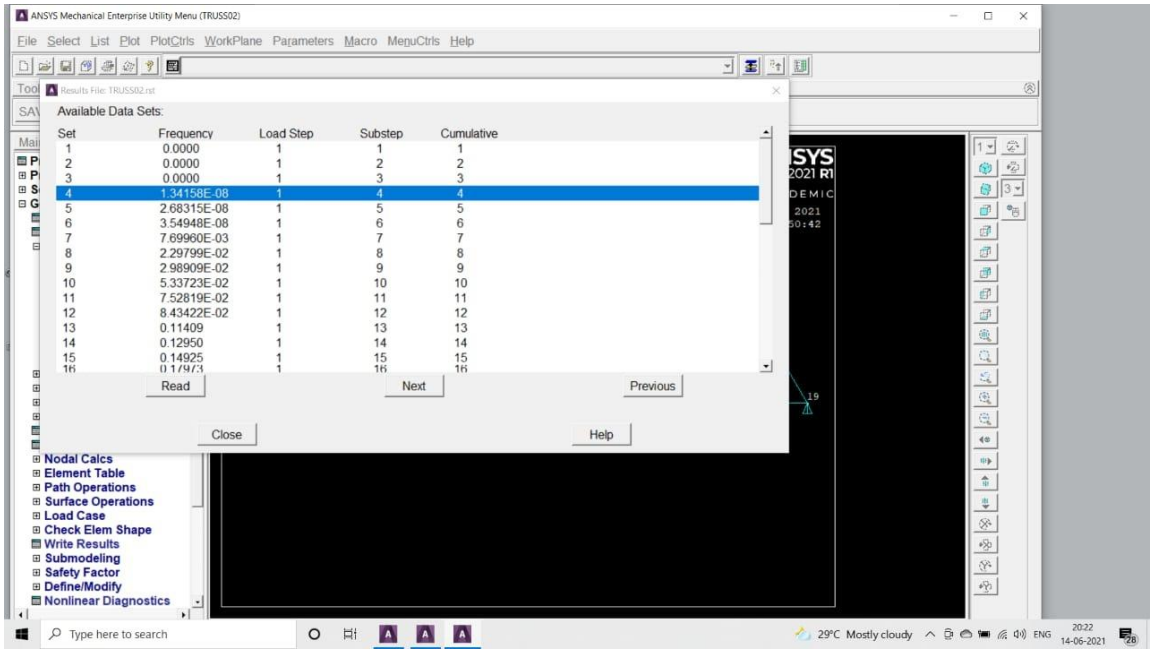


Fig 3.40 natural frequencies at modes

Step- 13:-We click on Plot results and select Deformed shape after that we again considerDef+Undeformedand click on Ok.

Step-14:-We read results By pic and select 5result and click ‘Close’.

Step-15:-We click on Plot results and select Deformed shape after that we again consider Def+Undeformed and click on Ok.

Step-16:-Now we repeat the steps 14 and 15 for the nodes 6-40.

Step-17:-Click on plot ctrls and below it click on deformed shape and select 'Animate' and chose Mode shape. Now we give the animation data, No of frames to create as 10 and Time delay(seconds) as 0.5 secs. We chose acceleration type as linear . the nodal solution data is given by selecting 'DOF solution' and then click on deformed shape and click Ok.

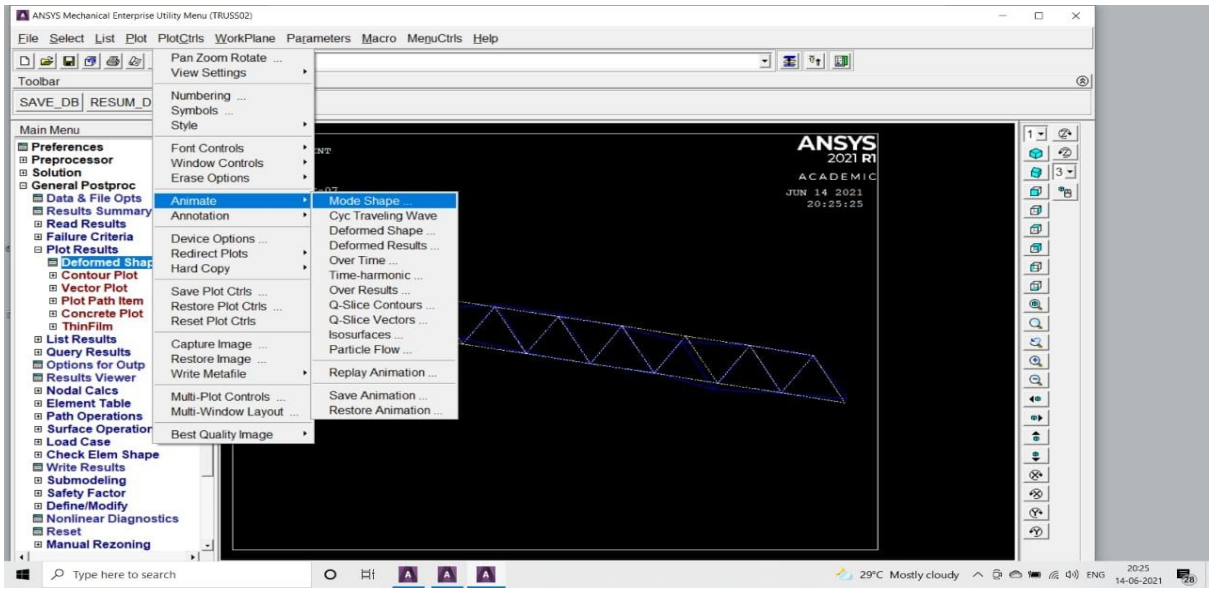


Fig 3.41 deformed shape of truss

3.4.1 REULTS OF MODAL ANALYSIS

```
*****  INDEX OF DATA SETS ON RESULTS FILE  *****

   SET    TIME/FREQ     LOAD STEP   SUBSTEP  CUMULATIVE
     1   0.0000            1         1         1

     2   0.0000            1         2         2

     3   0.0000            1         3         3

     4   0.0000            1         4         4

     5   0.0000            1         5         5
```

6	0.23237E-07	1	6	6
7	0.76996E-02	1	7	7
8	0.22980E-01	1	8	8
9	0.29891E-01	1	9	9
10	0.53372E-01	1	10	10
11	0.75282E-01	1	11	11
12	0.84342E-01	1	12	12
13	0.11409	1	13	13
14	0.12950	1	14	14
15	0.14925	1	15	15
16	0.17973	1	16	16
17	0.18595	1	17	17
18	0.20085	1	18	18
19	0.21743	1	19	19
20	0.22183	1	20	20
21	0.23885	1	21	21
22	0.25510	1	22	22
23	0.25818	1	23	23

24	0.27867	1	24	24
25	0.28256	1	25	25
26	0.29536	1	26	26
27	0.30313	1	27	27
28	0.33320	1	28	28
29	0.33579	1	29	29
30	0.33893	1	30	30
31	0.34431	1	31	31
32	0.35306	1	32	32
33	0.36434	1	33	33
34	0.37680	1	34	34
35	0.39635	1	35	35
36	0.42150	1	36	36
37	0.43633	1	37	37
38	0.45981	1	38	38
39	0.47060	1	39	39
40	0.48353	1	40	40

CHAPTER – IV

FEA ANALYSIS OF TRUSS STRUCTURE USING PYTHON.

4.1 NEED OF PYTHON

Science has basically been divided into theoretical and experimental disciplines, but during the last several decades computing has emerged as an important part of science. Scientific computing is closely based to theory, but it has many characteristics in common with experimental work. It is therefore often viewed as a new third branch of science. In most fields of science, computational work is a very important complement to both theory as well as experiments, and nowadays a large majority of both theoretical and experimental papers involve some numerical calculations, computer modeling or simulations. In theoretical and experimental sciences there are good established codes of conducts for how methods as well as results are published and made available to other scientists. For example, in theoretical sciences, derivations, proofs as well as results are published in full detail and in experimental sciences, the methods used as well as results are published. In computational sciences there are not yet any well established guidelines for how source code as well as generated data should be handled. For example, it is relatively rare that source code used in simulations for published papers are provided to readers, in contrast to the open nature of theoretical and experimental work. It is not uncommon that source code for simulation software is withheld as well as considered a competitive advantage. So, this problem has recently started to attract increasing attention, as well as a number of editorials in high-profile journals have called for increased openness in computational sciences. Some journals, including Science, have even started to demand of authors to provide the source code for simulation software used in publications to readers upon request. Replication (An author of a scientific paper that involves numerical calculations should be able to rerun the simulations as well as replicate the results upon request and other scientist should also be able to perform the same calculations as well as obtain the same results, given the information about the methods used in a publication.) and reproducibility (The results obtained from numerical simulations should be reproducible with an independent implementation of the method.) are two of the cornerstones in the scientific method. In short summary, a sound scientific result should be reproducible, as well as a sound scientific study should be replicable. Ensuring replicability as well as reproducibility of scientific simulations is a complicated problem, but there are some good tools to help with this such as Revision Control System (RCS) software (git, mercurial (hg) and subversion (svn)) as well as online repositories for source code, that available as both public and private repositories (GitHub, Bit bucket and Privately hosted repositories). Python is a modern, general-purpose, object-oriented, high-level programming language that includes some general

characteristics like clean and simple language, expressive language, dynamically typed, automatic memory management and interpreted. The main advantage is ease of programming, minimizing the time required to develop, debug, maintain the code, modular, object-oriented programming, good system for packaging, re-use of code, documentation tightly integrated with the code, large standard library, and a large collection of add-on packages. Python has a strong position in scientific computing because it contains extensive ecosystem of scientific libraries as well as environments like numpy, scipy, matplotlib etc. Python supports for parallel processing with processes and threads, Interprocess communication as well as GPU computing and suitable for use on high-performance computing clusters.

4.1.1 Reasons for using Python for Scientific Computing

- Clear, readable syntax through whitespace indentation
- Interactive python console
- Strong introspection capabilities
- Full modularity, supporting hierarchical packages
- Exception-based error handling
- Dynamic data types & automatic memory management

4.1.2 Reasons for using Python over MATLAB

- Python packages can do similarly everything Matlab can do for signal processing.
- Python is free and open source but Matlab is a closed-source commercial product.
- Python is a general-purpose language but Matlab is purely for scientific computing.
- Python is high level, easy to learn, and so many cool features.
- The demand for Python programmers is increasing Matlab is quite expensive, that means that code that is written in Matlab can only be used by people with sufficient funds to buy a license.
- Python contains great libraries, and yet more external libraries are being developed by Python programmers, scientists, mathematicians, and engineers.
- Similar every programming language other than Matlab, Python uses zero-based indexing.
- Python code tends to be more compact as well as more readable than Matlab code.
- OOP in Python is simple and elegant, that offers flexibility but Matlab's OOP scheme is complex and confusing.
- Python offers a wider set of choices in graphics packages and toolsets as well as easily integrates better with other languages.

4.2 Methodology of Implementing Python for current Project

4.2.1 Jupyter Notebook

- Jupyter Notebook (open source code), which began as the iPython Notebook project, is a development environment for writing and executing Python code. Jupyter Notebook is often used for exploratory data analysis and visualization.
- Project Jupyter is the top-level project name for all of the subprojects under development, which includes Jupyter Notebook. Jupyter Notebooks can also run code for other programming languages such as Julia and R.
- The key piece of Jupyter Notebook infrastructure is a web application that runs locally for creating and sharing documents that contain embedded code and execution results.
- IPython Notebook was the original project that proved that there was great demand among data scientists and programmers for an interactive, repeatable development environment. Jupyter Notebook became the new official name for the overall project during The Big Split after the IPython Notebook project matured into distinct submodules such as the interactive shell, notebook document format and user interface widgets tools. However, the IPython Notebook name sticks around as the Python backend for Jupyter Notebook which is seriously confusing if you are searching the internet and come across both current and old articles that use all of these names interchangeably.

4.2.2 Python Version

For this current project we used python 3.9.5 version for developing code. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

4.2.3 Python Libraries

A library is a collection of pre-combined codes that can be used iteratively to reduce the time required to code. They are particularly useful for accessing the pre-written frequently used codes, instead of writing them from scratch every single time. Similar to the physical libraries, these are a collection of reusable resources, which means every library has a root source. This is the foundation behind the numerous open-source libraries available in Python. There are many libraries but we are listing the libraries which we used during this current project.

- NumPy : NumPy is the basic package for scientific computing in Python that provides multi-dimensional arrays and matrices, broadcasting

functions, tools for integrating C/C++, Fortran code, mathematical, logical, shape manipulation, sorting, selecting, I/O, useful linear algebra, Fourier transform, random number capabilities, basic statistical operations and much more. NumPy can be used as an efficient multi-dimensional container of generic data and it is licensed under the BSD license.

- SciPy library : SciPy library is one of the core packages used by scientists, analysts, and engineers that provides many user-friendly and efficient numerical routines such as routines for numerical integration, optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing. SciPy builds on the NumPy array object as well as is part of the NumPy stack that includes tools like Matplotlib, pandas and SymPy. The SciPy library is currently distributed under the BSD license, as well as its development is supported and sponsored by an open community of developers. It is also supported by Numfocus that is a community foundation for supporting reproducible and accessible science.
- Matplotlib : Matplotlib is a 2D plotting library for the Python programming language that produces publication quality figures in a variety of hardcopy formats as well as
- interactive environments across platforms. Matplotlib tries to make easy things easy as well as hard things possible and provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits such as wxPython, Qt, or GTK+. Matplotlib was mainly written by John D. Hunter, has an active development community, and is distributed under a BSD-style license.

4.3 Developing Python code for planar truss structure:

Our project is to develop full working code for any type of truss structure for fea analysis. Inorder to develop pythode code using jupyter Notebook we considered Railway Bridge problem from the text book of Introduction to finite elements in Engineering by Tirupathi R. Chandrapatla and Ashok D Belugundu.

4.3.1 Formulation of Truss behind developing code for static analysis

There are two joints for an arbitrarily inclined single truss element (at an angle q, positive counter-clockwise from +ve x- axis). For each joint i, there are two degrees of freedom, i.e.,a joint can have horizontal displacement u(i) and vertical displacement v(i) . Hence, for a single truss element, there are 4 degrees of freedom. The nodal displacement degrees of freedom and the nodal force de grees of freedom are shown in the following figgure.

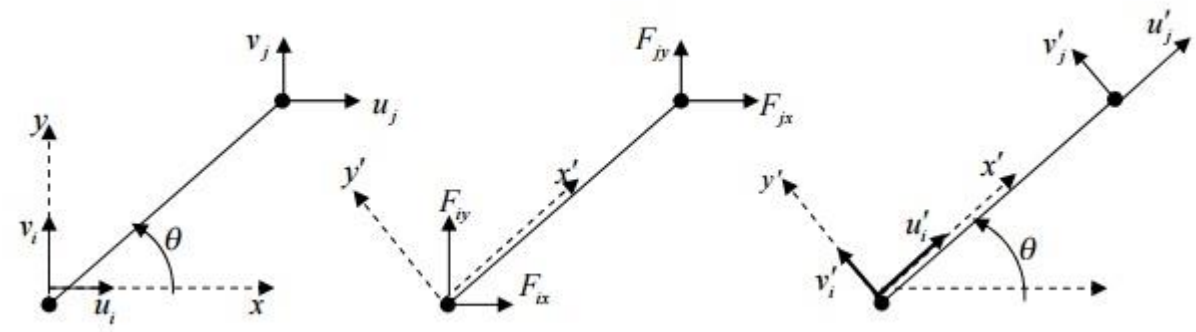
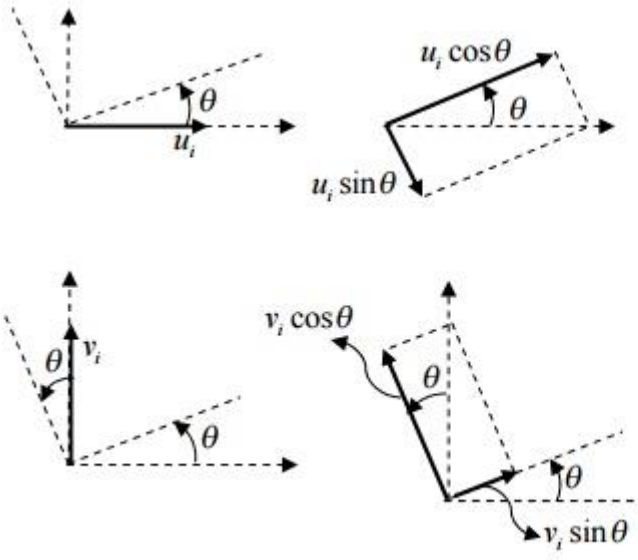


Fig 4.1 displacement of truss at nodes

Note that the deformations occurring in the truss members are so small that they are only axial. The axial displacement of the truss can be resolved along horizontal x-axis and vertical y-axis. But in our derivation, let us resolve the horizontal and vertical displacements (in xy-axes) of a joint along and perpendicular to the truss member (in x'y-axes). Refer to the Figure in the next page. Note ui sin*q* component acting towards negative y -direction and all other components acting towards in +ve x - and y -directions.



$$u'_i = u_i \cos\theta + v_i \sin\theta$$
$$v'_i = -u_i \sin\theta + v_i \cos\theta$$

$$u'_j = u_j \cos\theta + v_j \sin\theta$$
$$v'_j = -u_j \sin\theta + v_j \cos\theta$$

Fig 4.2 global displacement vector at nodes

The above equations can be written in the matrix form as follows

$$\begin{Bmatrix} u' \\ v'_i \\ u'_j \\ v'_j \end{Bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix}$$

$\{u'\} = [T]\{u\}$ where $[T]$ is the transformation matrix

$\{u¢\}=[T]\{u\}$

where $[T]$is the transformation matrix It is important to note that the displacements v ¢(i) and v(i) are both zero since there can be no displacements perpendicular to the length of the member. Also $[T]$-1 =$[T]T$

Similarly, we resolve forces along the length of the member (positive x direction) and perpendicular to the length of the member (positive y direction)

$$\begin{Bmatrix} F'_{ix} \\ F'_{iy} \\ F'_{jx} \\ F'_{jy} \end{Bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & \cos\theta & \sin\theta \\ 0 & 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{Bmatrix} F_{ix} \\ F_{iy} \\ F_{jx} \\ F_{jy} \end{Bmatrix}$$

$\{F'\} = [T]\{F\}$ where $[T]$ is the transformation matrix

$\{F'\}=[T]\{F\}$

where $[T]$is the transformation matrix

The arbitrarily inclined truss member can be thought of as a simple bar element oriented at the same angle q . Hence, we can write the finite element equation for this inclined bar element (in $x¢y$coordinate system) as

$$\begin{Bmatrix} F'_{ix} \\ F'_{iy} \\ F'_{jx} \\ F'_{jy} \end{Bmatrix} = \frac{AE}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} u'_i \\ v'_i \\ u'_j \\ v'_j \end{Bmatrix}$$

$$\{F'\} = [k']\{u'\}$$

Substituting $\{F ¢\}$and $\{u¢\}$from the previous equations, we can write

$[T]\{F\}=[k¢][T]\{u\}$

Pre-multiplying the above equation by $[T]$-1 ,

$[T]$-1 $[T]\{F\}=[T]$-1$[k¢][T]\{u\}$

But [T]-1 [T]=1 and the above equation can be written as

{F }=[k]{u} where [k] [=T]-1 [k¢][T]

Carrying out the matrix multiplication for [k , we obtain

$$\begin{Bmatrix} F_{ix} \\ F_{iy} \\ F_{jx} \\ F_{jy} \end{Bmatrix} = \frac{AE}{L} \begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix}$$

where c =cos2 q and s =sin2 q .

Computation of strain and stress in the truss element

The change in length of the truss member is equal to the change in axial displacement of the truss member in the x¢$y$co-ordinate system

$$\delta = u_j' - u_i'$$

$$\delta = \left(u_j \cos\theta + v_j \sin\theta\right) - \left(u_i \cos\theta + v_i \sin\theta\right)$$

$$\delta = \left\langle -\cos\theta \quad -\sin\theta \quad \cos\theta \quad \sin\theta \right\rangle \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix}$$

Strain in the truss element is given by $\varepsilon^e = \frac{\delta}{L}$, i.e.,

$$\varepsilon^e = \frac{\left\langle -\cos\theta \quad -\sin\theta \quad \cos\theta \quad \sin\theta \right\rangle}{L} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix}$$

Stress in the truss element is given by $\sigma^e = E\varepsilon^e$, i.e.,

$$\sigma^e = E \frac{\left\langle -\cos\theta \quad -\sin\theta \quad \cos\theta \quad \sin\theta \right\rangle}{L} \begin{Bmatrix} u_i \\ v_i \\ u_j \\ v_j \end{Bmatrix}$$

4.3.2 Formulation of Truss behind developing code for Modal analysis

The goal of modal analysis in structural mechanics is to determine the natural mode shapes and frequencies of an object or structure during free vibration. It is common to use the finite element method (FEM) to perform this analysis because, like other

calculations using the FEM, the object being analyzed can have arbitrary shape and the results of the calculations are acceptable. The types of equations which arise from modal analysis are those seen in eigensystems. The physical interpretation of the eigenvalues and eigenvectors which come from solving the system are that they represent the frequencies and corresponding mode shapes. Sometimes, the only desired modes are the lowest frequencies because they can be the most prominent modes at which the object will vibrate, dominating all the higher frequency modes.

It is also possible to test a physical object to determine its natural frequencies and mode shapes. This is called an Experimental Modal Analysis. The results of the physical test can be used to calibrate a finite element model to determine if the underlying assumptions made were correct (for example, correct material properties and boundary conditions were used).

4.3.3 Problem Data

The fig shows the Railway Bridge Structure having 19 nodes and 35 elements with equal spacing.

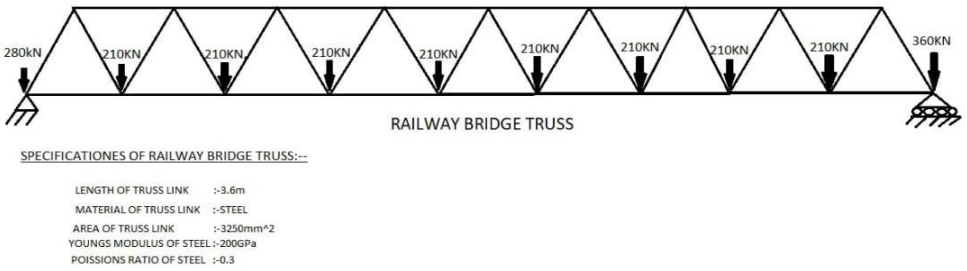


Fig 4.3 railway bridge truss

Table 4.1 Nodal coordinates:

Node	X(mm)	Y(mm)
1	0	0
2	1800	3118

3	3600	0
4	5400	3118
5	7200	0
6	9000	3118
7	10800	0
8	12600	3118
9	14400	0
10	16200	3118
11	18000	0
12	19800	3118
13	21600	0
14	23400	3118
15	25200	0
16	27000	3118
17	28800	0
18	30600	3118
19	32400	0

Table 4.2 Element connectivity:

Element	Starting Node	Ending Node
1	1	3
2	3	5
3	5	7
4	7	9

5	9	11
6	11	13
7	13	15
8	15	17
9	17	19
10	2	4
11	4	6
12	6	8
13	8	10
14	10	12
15	12	14
16	14	16
17	16	18
18	1	2
19	2	3
20	3	4
21	4	5
22	5	6
23	6	7
24	7	8
25	8	9
26	9	10
27	10	11
28	11	12

29	12	13
30	13	14
31	14	15
32	15	16
33	16	17
34	17	18
35	18	19

Table 4.3 Nodal Supports:

Node Number Having Support	Type of Support Condition
1	Hinged Support
19	Horizontal Roller Support

Table 4.4 Nodal Load:

Node Number having Loads	Horizontal Load F_x (N)	Vertical Load F_y (N)
1	0	-280000
3	0	-210000
5	0	-210000
7	0	-210000
9	0	-210000
11	0	-210000
13	0	-210000
15	0	-210000
17	0	-210000

19	0	-360000

Table 4.5 Material Properties:

Meterial Used	Structural Steel
Modulus of Elasticity	2e5 N/mm^2
Cross sectional Area of element	3250 mm^2
Density	0.00785 gm/mm^3
Poisson's ratio	0.3

4.3.4 Procedure for writing Code

After installing python and Jupyter notebook and import all required libraries for the project. This code contains two parts. 1st part is static analysis for finding nodal displacements, element stresses and support reactions. 2nd part is dynamic analysis for finding natural frequencies and Modal shapes for the truss structure. The following procedure is adopted for developing code for all truss structures.

- This code is written in Jupyter notebook and can run in any python3 version and this folder is saved as Project2021 and file as projectcode.ipynb.
- First import all libraries.
- Creating input data for entering all information regarding particular truss structure.
- Creating Transformation Matrix to find direction cosines length of element.
- Formulating element stiffness matrix, element mass matrix
- Assembling Global stiffness Matrix, Global Mass Matrix and Global load vector.
- Finding reduced global stiffness matrix, reduced global mass matix and reduced load vector.
- Applying Boundary conditions using gauss elimination method and find nodal displacements, element stresses and support reactions.
- Next creating equations of motion for undamped free linear vibrations to calculate eigen values and eigen vectors using inverse iteraton method.
- After inputing the data, run the code then the results are printed within seconds.

4.4 Python code for truss structure for Static and Dynamic Analysis

```
# importing libraries
import math
import matplotlib.pyplot as plot
import numpy as np
# creating path for a file
file = open(r"D:\Remo desktop\PhD work\Python Project 2021\out.txt","w+")
# creating input data and print
noofnodes=int(input("enter the total number of nodes"))
file.write("the total number of nodes= "+str(noofnodes)+ '\n')
noofelements=int(input("enter the total number of elements"))
file.write("the total number of elements= "+str(noofelements)+ '\n')
coordinates={}
for i in range(1,noofnodes+1):
coordinates[i]=list(map(int,input("enter coordinates for node "+ str(i)+ ' in mm :').split(",")))
file.write('coordinates for node' + str(i)+ 'in mm : '+ str(coordinates[i])+ '\n')
area=float(input("enter area in mm-square"))
file.write("area in mm-square= "+str(area)+ '\n')
elasticity=float(input("enter the modulus of elasticity in N/mm-square"))
file.write("modulus of elasticity in N/mm-square= "+str(elasticity)+ '\n')
startend={}
for i in range(1,noofelements+1):
startend[i]=list(map(int,input("enter the start node and end node for element "+ str(i)+ ' :').split(",")))
file.write('the start node and end node for element '+ str(i)+ ' :'+str(startend[i])+ '\n')
noofsuppnodes=int(input("enter the total number of nodes having supports"))
file.write('the total number of nodes having supports '+str(noofsuppnodes)+ '\n')
nodesandtypesupport=[]
for i in range(noofsuppnodes):
    x=int(input("enter the node number having support"))
    print("enter the type of support")
    print("h for hinged/fixed")
    print("hrs for horizontal roller support")
    print("vrs for vertical roller support")
    t=input()
    file.write('the node number having support is '+str(x)+' type of support is '+t+ '\n')
```

```
        nodesandtypesupport.append([x,t.lower()])
    print(nodesandtypesupport)
    noofloadnodes=int(input("enter total number of loaded nodes "))
    file.write('total number of loaded nodes '+str(noofloadnodes)+ '\n')
    loadnodes=[]
    typeofload={}
    for i in range(noofloadnodes):
        x=int(input("enter the node number having load "))
        loadnodes.append(x)
typeofload[x]=list(map(float,input('enter the horizontal and vertical loads in N ').split(',')))
file.write('the node number having load is '+str(x)+'  horizontal and vertical loads in N is '+str(typeofload[x])+ '\n')
    density=float(input("enter density in gm/mm-cube "))
    poission=float(input("enter poission's ratio "))
    file.write('density in gm/mm-cube is '+str(density)+ '\n')
    file.write('poissions ratio  is '+str(poission)+ '\n')
    # calculating length of element direction cosines
    lengthofelements={}
    cos={}
    sin={}
    for i in range(1,noofelements+1):
        x,y=startend[i][0],startend[i][1]
        a,b=coordinates[x][0],coordinates[x][1]
        c,d=coordinates[y][0],coordinates[y][1]
        l=math.sqrt((d-b)**2+(c-a)**2)
        lengthofelements[i]=l
        cos[i]=(c-a)/l
        sin[i]=(d-b)/l
    for i in range(1,noofelements+1):

        print("length of elements "+str(i)+": ",lengthofelements[i])
        print("cos value of elements "+str(i)+": ",cos[i])
        print("sin value of elements "+str(i)+": ",sin[i])
    # calculating element stiffness matrix and element mass matrix
    stiffness={}
    for i in range(1,noofelements+1):
        l=cos[i]
        m=sin[i]
```

```
    mat=[[l**2,l*m,-l**2,-l*m],[l*m,m**2,-l*m,-m**2],[-l**2,-l*m,l**2,l*m],[-l*m,-
m**2,l*m,m**2]]
    a=(area*elasticity)/lengthofelements[i]
    for j in range(4):
        for k in range(4):
            mat[j][k]=a*mat[j][k]
    print("stiffness matrix of element "+str(i))
    for j in range(4):
        for k in range(4):
            print(mat[j][k],end='    ')
        print()
    print()
    print()
    stiffness[i]=mat
mass={}
for i in range(1,noofelements+1):

    mat=[[2,0,1,0],[0,2,0,1],[1,0,2,0],[0,1,0,2]]
    a=(area*density*lengthofelements[i])/6
    for j in range(4):
        for k in range(4):
            mat[j][k]=a*mat[j][k]
    print("mass matrix of element "+str(i))
    for j in range(4):
        for k in range(4):
            print(mat[j][k],end='    ')
        print()
    print()
    print()
    mass[i]=mat
# calculating Global Stiffness matrix and global mass matrix
ndof = noofnodes * 2
print(ndof)
globstifmat=[]
for i in range(ndof):
    globstifmat.append([0]*ndof)
x=0
asinode={}
for i in range(1,noofnodes+1):
    asinode[i]=[x,x+1]
```

```
    x+=2
e=startend
n=asinode
d={}
for i in range(1,noofelements+1):
    x=[]
    p=e[i]
    q=p[0]
    r=p[1]
    x.extend([n[q][0],n[q][1],n[r][0],n[r][1]])
    d[i]=x
print(d)
    for i in range(ndof):
    for j in range(ndof):
        s=0
        for k in d:
            if(i in d[k] and j in d[k]):
                zzz=stiffness[k]
                s+=zzz[d[k].index(i)][d[k].index(j)]

        globstifmat[i][j]=s
print("global stiffness matrix ")
for j in range(ndof):
    for k in range(ndof):
        print(globstifmat[j][k],end='    ')
    print()
print()
print(len(globstifmat))
print(len(globstifmat[0]))
ndof = noofnodes * 2
print(ndof)
globmassmat=[]
for i in range(ndof):
    globmassmat.append([0]*ndof)
for i in range(ndof):
    for j in range(ndof):
        s=0
        for k in d:
            if(i in d[k] and j in d[k]):
                zzz=mass[k]
```

```
            s+=zzz[d[k].index(i)][d[k].index(j)]

        globmassmat[i][j]=s
print("global mass matrix ")
for j in range(ndof):
    for k in range(ndof):
        print(globmassmat[j][k],end='    ')
    print()
print()
# calculating global load vector
globloadvector=[0]*noofnodes*2


for i in typeofload:
    globloadvector[asinode[i][0]]=typeofload[i][0]
    globloadvector[asinode[i][1]]=typeofload[i][1]
print(globloadvector)
print(nodesandtypesupport)
coltorem=[]
rowtorem=[]
for i in nodesandtypesupport:
    if(i[1]=='h'):
        coltorem.extend(asinode[i[0]])
        rowtorem.extend(asinode[i[0]])
    elif(i[1]=='hrs'):
        coltorem.append(asinode[i[0]][1])
        rowtorem.append(asinode[i[0]][1])
    elif(i[1]=='vrs'):
        coltorem.append(asinode[i[0]][0])
        rowtorem.append(asinode[i[0]][0])
print(coltorem)
print(rowtorem)
npglobstifmat=np.array(globstifmat)
onpglobstifmat=npglobstifmat
print(npglobstifmat)
print(npglobstifmat.shape)
# Reducing Global Stiffness Matrix, global mass matrix and global load vector
#removing row
npglobstifmat=np.delete(npglobstifmat, rowtorem, 0)
print(npglobstifmat)
```

```
print(npglobstifmat.shape)
#removing column
npglobstifmat=np.delete(npglobstifmat, rowtorem, 1)
print(npglobstifmat)
rednpglobstifmat=npglobstifmat
print(npglobstifmat.shape)
#mass matrix removing rows nd columns
npglobmassmat=np.array(globmassmat)
onpglobmassmat=npglobmassmat
print(npglobmassmat)
npglobmassmat=np.delete(npglobmassmat, rowtorem, 0)
print(npglobmassmat)
npglobmassmat=np.delete(npglobmassmat, rowtorem, 1)
print(npglobmassmat)
rednpglobmassmat=npglobmassmat
print(npglobmassmat.shape)
print(globloadvector)
#coverting to column vector
npglobloadvector=np.array(globloadvector)
orignpglobloadvector=npglobloadvector
print(npglobloadvector.shape)
#removing rows
npglobloadvector=np.delete(npglobloadvector, rowtorem, 0)
print(npglobloadvector)
print(npglobloadvector.shape)
# Solving KQ = F  to determine Nodal Displacments
#linear eqn solving
a = npglobstifmat
b = npglobloadvector
nodaldisplacement = np.linalg.solve(a, b)
print("nodal displacement matrix in mm")
print(nodaldisplacement)
#changing dimension
xx=[]
zz=list(nodaldisplacement)
i=0
s=set(rowtorem)
for j in range(noofnodes*2):
   if(j not in s):
      xx.append(nodaldisplacement[i])
```

```
        i+=1
    else:
        xx.append(0)
print(xx)
file.write('nodal displacements in mm are ' + str(xx)+'\n')
#changing to col matrix
npnodaldisplacement=np.array(xx)
print(npnodaldisplacement.shape)
print('nodal displacement vector in mm')
print(npnodaldisplacement)
# calculating Element Stresses
elementstresses={}
for i in range(1,noofelements+1):
    l=cos[i]
    m=sin[i]
    m1=np.array([-l,-m,l,m]).reshape(1,-1)
    pq=[]
    for z in d[i]:
        pq.append(npnodaldisplacement[z])
    zz=np.array(pq).reshape(-1,1)
    a=(elasticity/lengthofelements[i])
    b=np.matmul(m1,zz)
    elementstresses[i]=a*b
print("element stresses in N/mm-square")
for i in elementstresses:
    print("element"+str(i)+"="+str(elementstresses[i][0][0]))

    file.write('element '+str(i)+' stress in N/mm-square=
'+str(elementstresses[i][0][0])+'\n')
#reaction of support

npglobstifmat=np.array(globstifmat)

x=np.matmul(npglobstifmat,npnodaldisplacement)
y=x-orignpglobloadvector
reactionmat=y
print("reaction at support nodes in N")
print(reactionmat)
j=0
for i in nodesandtypesupport:
```

```
    if(i[1]=='h'):
        print('reaction node at R'+str(i[0])+"X in N "+str(reactionmat[(i[0]-1)*2]))
        print('reaction node at R'+str(i[0])+"Y in N "+str(reactionmat[(i[0]-1)*2+1]))
        file.write('reaction node at R'+str(i[0])+"X in N "+str(reactionmat[(i[0]-1)*2])+
'\n')
        file.write('reaction node at R'+str(i[0])+"Y in N "+str(reactionmat[(i[0]-
1)*2+1])+ '\n')
    elif(i[1]=='hrs'):
        print('reaction node at R'+str(i[0])+"Y in N "+str(reactionmat[(i[0]-1)*2+1]))
        file.write('reaction node at R'+str(i[0])+"Y in N "+str(reactionmat[(i[0]-
1)*2+1])+ '\n')
    elif(i[1]=='vrs'):
        print('reaction node at R'+str(i[0])+"X in N "+str(reactionmat[(i[0]-1)*2]))
        file.write('reaction node at R'+str(i[0])+"X in N "+str(reactionmat[(i[0]-1)*2])+
'\n')
#modal analysis
#(globalstiffnessmatrix-(lambda)globalmassmatrix)X
class Eigen(object):

    def _init_(self, *args, **kwargs):
        return super()._init_(*args, **kwargs)

    def Rescale(self, x):
        max = x.max()
        min = x.min()
        s = x.shape
        n = s[0]
        amax = max
        if abs(min) > max: amax = abs(min)
        for i in range(n):
            x[i] = x[i] / max
        return x

    def RescaleEigenVectors(self, evec):
        dims = evec.shape
        ndofs = dims[0]
        for i in range(ndofs):
            evec[:,i] = self.Rescale(evec[:,i])
        return evec
```

```
def GetOrthogonalVector(self, ndofs, trial, mg, ev,evec):
    const = 0
    s = [ndofs]
    sumcu= np.zeros(s)
    for e in range(ev  ):
        U = evec[:,e]
        const += trial @ mg @ U
        cu = [x * const for x in U]
        sumcu += cu
    trial = trial - sumcu
    return trial
def Solve(self,kg, mg, tolerance = 0.00001 ):
    dims = kg.shape
    ndofs = dims[0]
    s = (ndofs,ndofs)
    evec = np.zeros(s)
    s = (ndofs)
    eval = np.zeros(ndofs)
    trial = np.ones(ndofs)
    eigenvalue0 = 0
    for ev in range(ndofs):
        print("Computing eigenvalue and eigen vector " + str(ev) + "... " , end="")

        converged = False
        uk_1 = trial
        k = 0
        while converged == False:
            k += 1

            if ev > 0:
                uk_1 = self.GetOrthogonalVector(ndofs,uk_1,mg,ev,evec)
            vk_1 = mg @ uk_1
            uhatk = np.linalg.solve(kg,vk_1)
            vhatk = mg @ uhatk
            uhatkt = np.transpose(uhatk)
            eigenvalue = (uhatkt @ vk_1)/(uhatkt @ vhatk)
            denominator = math.sqrt(uhatkt @ vhatk)
            uk = uhatk/denominator
            tol = abs((eigenvalue - eigenvalue0) / eigenvalue)
            if tol <= tolerance:
```

```
                converged = True
                evec[:,ev] = uk
                eval[ev] = eigenvalue
                print("Eigenvalue = " + str(eigenvalue))
                print('no of iterations= ',k)
            else:
                eigenvalue0 = eigenvalue
                uk_1 = uk

                if k > 1000:
                    evec[:,ev] = uk
                    eval[ev] = eigenvalue
                    print ("could not converge. Tolerance = " + str(tol))
                    break

        self.eigenvalues = eval
        return evec
rednpglobstifmat.shape
rednpglobmassmat.shape
# compute eigenvalues and eigen vectors
e = Eigen()
evec = e.Solve(rednpglobstifmat,rednpglobmassmat)
evect = e.RescaleEigenVectors(evec)

eval = e.eigenvalues
neval = len(eval)
print("eigen values")
eigvalues=e.eigenvalues
print(e.eigenvalues)
file.write("eigen values are "+str(eigvalues)+'\n')
print(len(e.eigenvalues))
print("eigen vectors")
print(len(evect))
print(evect)
for i in evect:
    file.write("eigen vectors are "+str(list(i))+'\n')
from scipy.linalg import eigvalsh
ab=eigvalsh(rednpglobstifmat,rednpglobmassmat)
print(ab[0])
f1=np.sqrt(ab[0])
```

```
f1
f2=np.sqrt(ab[0])
f2=f2/(2*3.14)
f2
#frequency
freq=[]
for i in eigvalues:
   freq.append(math.sqrt(i)/(2*3.14))
print("natural frequency in hertz",freq)
file.write("natural frequency in hertz "+str(freq)+ '\n')
import matplotlib.pyplot as plt
z=1
for i in evect:
   time        = np.array(i)
   amplitude   = np.sin(time)
   plot.plot(time, amplitude)
   plot.title('mode'+str(z))
   plot.xlabel('vector')
   plot.ylabel('sin(vector)')
   plot.grid(True, which='both')
   plot.axhline(y=0, color='k')
   plot.savefig(r"D:\Remo desktop\PhD work\Python Project 2021\mode"+str(z)+'.png', bbox_inches='tight')
   plot.show()
   plot.show()

   z+=1
file.close()
```

4.5 How the python prints the results

Now run the code and give the input data. Python gives all static and dynamic results with in a few milli seconds and print the results.

the total number of nodes= 19

the total number of elements= 35

coordinates for node1in mm : [0, 0]

coordinates for node2in mm : [1800, 3118]

coordinates for node3in mm : [3600, 0]

coordinates for node4in mm : [5400, 3118]

coordinates for node5in mm : [7200, 0]
coordinates for node6in mm : [9000, 3118]
coordinates for node7in mm : [10800, 0]
coordinates for node8in mm : [12600, 3118]
coordinates for node9in mm : [14400, 0]
coordinates for node10in mm : [16200, 3118]
coordinates for node11in mm : [18000, 0]
coordinates for node12in mm : [19800, 3118]
coordinates for node13in mm : [21600, 0]
coordinates for node14in mm : [23400, 3118]
coordinates for node15in mm : [25200, 0]
coordinates for node16in mm : [27000, 3118]
coordinates for node17in mm : [28800, 0]
coordinates for node18in mm : [30600, 3118]
coordinates for node19in mm : [32400, 0]
area in mm-square= 3250.0
modulus of elasticity in N/mm-square= 200000.0
the start node and end node for element 1 :[1, 3]
the start node and end node for element 2 :[3, 5]
the start node and end node for element 3 :[5, 7]
the start node and end node for element 4 :[7, 9]
the start node and end node for element 5 :[9, 11]
the start node and end node for element 6 :[11, 13]
the start node and end node for element 7 :[13, 15]
the start node and end node for element 8 :[15, 17]
the start node and end node for element 9 :[17, 19]
the start node and end node for element 10 :[2, 4]
the start node and end node for element 11 :[4, 6]
the start node and end node for element 12 :[6, 8]
the start node and end node for element 13 :[8, 10]
the start node and end node for element 14 :[10, 12]

the start node and end node for element 15 :[12, 14]
the start node and end node for element 16 :[14, 16]
the start node and end node for element 17 :[16, 18]
the start node and end node for element 18 :[1, 2]
the start node and end node for element 19 :[2, 3]
the start node and end node for element 20 :[3, 4]
the start node and end node for element 21 :[4, 5]
the start node and end node for element 22 :[5, 6]
the start node and end node for element 23 :[6, 7]
the start node and end node for element 24 :[7, 8]
the start node and end node for element 25 :[8, 9]
the start node and end node for element 26 :[9, 10]
the start node and end node for element 27 :[10, 11]
the start node and end node for element 28 :[11, 12]
the start node and end node for element 29 :[12, 13]
the start node and end node for element 30 :[13, 14]
the start node and end node for element 31 :[14, 15]
the start node and end node for element 32 :[15, 16]
the start node and end node for element 33 :[16, 17]
the start node and end node for element 34 :[17, 18]
the start node and end node for element 35 :[18, 19]
the total number of nodes having supports 2
the node number having support is 1 type of support is h
the node number having support is 19 type of support is hrs
total number of loaded nodes 10
the node number having load is 1 horizontal and vertical loads in N is [0.0, -280000.0]
the node number having load is 3 horizontal and vertical loads in N is [0.0, -210000.0]
the node number having load is 5 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 7 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 9 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 11 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 13 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 15 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 17 horizontal and vertical loads in N is [0.0, -210000.0]

the node number having load is 19 horizontal and vertical loads in N is [0.0, -360000.0]

density in gm/mm-cube is 0.00785

poissions ratio is 0.3

nodal displacements in mm are [0, 0, 80.57235900725235, -52.71710028209583, 2.6857453002416967, -103.88373823563647, 75.20086840676883, -150.39864392570678, 10.07154487590647, -192.6497782122503, 65.80075985592269, -227.92348674249067, 20.814526076873438, -256.9953459585102, 53.71490600483477, -277.5393170896713, 33.57181625302168, -290.7185921601952, 40.28617950362602, -294.5947479815832, 47.0005427542304, -290.71859216019516, 26.857453002417287, -277.5393170896711, 59.75783293037865, -256.9953459585099, 14.77159915132945, -227.9234867424904, 70.50081413134559, -192.64977821225014, 5.371490600483331, -150.39864392570675, 77.88661370701043, -103.88373823563649, -2.0155159541979338e-13, -52.71710028209584, 80.57235900725217, 0]

element 1 stress in N/mm-square= 149.2080722356498

element 2 stress in N/mm-square= 410.32219864804296

element 3 stress in N/mm-square= 596.8322889426094

element 4 stress in N/mm-square= 708.7383431193468

element 5 stress in N/mm-square= 746.0403611782623

element 6 stress in N/mm-square= 708.738343119347

element 7 stress in N/mm-square= 596.8322889426081

element 8 stress in N/mm-square= 410.3221986480463

element 9 stress in N/mm-square= 149.2080722356527

element 10 stress in N/mm-square= -298.4161444713062

element 11 stress in N/mm-square= -522.2282528247856

element 12 stress in N/mm-square= -671.4363250604399

element 13 stress in N/mm-square= -746.0403611782639

element 14 stress in N/mm-square= -746.0403611782631

element 15 stress in N/mm-square= -671.4363250604354

element 16 stress in N/mm-square= -522.2282528247845

element 17 stress in N/mm-square= -298.4161444713074

element 18 stress in N/mm-square= -298.43829460071817

element 19 stress in N/mm-square= 298.43829460071817

element 20 stress in N/mm-square= -223.82872095053725

element 21 stress in N/mm-square= 223.8287209505396

element 22 stress in N/mm-square= -149.21914730035712

element 23 stress in N/mm-square= 149.21914730035869

element 24 stress in N/mm-square= -74.60957365017461

element 25 stress in N/mm-square= 74.60957365018092

element 26 stress in N/mm-square= 0.0

element 27 stress in N/mm-square= -3.157733329710917e-12

element 28 stress in N/mm-square= 74.6095736501825

element 29 stress in N/mm-square= -74.60957365018092

element 30 stress in N/mm-square= 149.21914730035869

element 31 stress in N/mm-square= -149.21914730035553

element 32 stress in N/mm-square= 223.8287209505349

element 33 stress in N/mm-square= -223.82872095053645

element 34 stress in N/mm-square= 298.43829460071856

element 35 stress in N/mm-square= -298.43829460071817

reaction node at R1X in N 1.1816155165433884e-08

reaction node at R1Y in N 1119999.9999999981

reaction node at R19Y in N 1199999.9999999972

eigen values are [0.00234043 0.02084768 0.03527261 0.11245795 0.22372369 0.19907059

0.11219565 0.02924594 0.00643356 0.00449632 0.02088284 0.02231559

0.00370794 0.00785575 0.00288586 0.00249354 0.00266894 0.00245855

0.00255837 0.00243372 0.00249551 0.00241577 0.00245658 0.0024025

0.00243084 0.00244897 0.00241239 0.00242439 0.00239916 0.00240752

0.00238934 0.00239541 0.00238184 0.00238639 0.00238352]

eigen vectors are [-0.27799045565014263, 0.4802719325422103, -0.1296690082758511, -0.3867645404348355, 0.8798180497251236, 0.6830058652995395, 0.20043660942161073, 0.6230894275101142, 1.0, 1.0, -1.078998781421197, 0.532752541553521, -0.14664847887313584, 0.0032723966875199853, -0.1945351825034667, 1.0, -0.2137952511157689, 1.0, -0.22561650756410556, 1.0, -0.2337438760568063, 1.0, -0.23964958084113616, 1.0, 1.0, -0.24091511313716987, 1.0, -0.24534667690296313, 1.0, -0.24878409829719475, 1.0, -0.25153297519894685, 1.0, 1.0, -0.2545465577557507]

eigen vectors are [0.17803559634423766, -0.28167753351026253, 0.2332450632018002, 0.4843885758649944, -0.409921973289099, -0.2760920684694611, -0.38255900269679116, -0.2861343173731408, -1.2985268233637735, -0.8113276791527179, 0.5636096207662951, -0.2711156023029782, 0.10987895006449384, 0.029080008480762407, 0.13355954812524237, -0.6603540632499568, 0.14378690910305547, -0.6573397285394004, 0.1500159214703888, -0.6550675577299753, 0.15431504851525035, -0.6533148286556708, 0.15744787773038738, -0.651930132988549, -0.6546924097072699, 0.15811195365036507, -

0.6529212468230842, 0.1604761754956811, -0.6515472581347203, 0.16231196257269095, -0.6504506614023005, 0.16378135615474912, -0.6495552102459853, -0.6501023229189612, 0.16539404256567505]

eigen vectors are [-0.013771764084600773, 0.24060768642796307, 0.16436294401525556, -0.07268596292187587, 0.6698467898473073, 0.5152851775163946, -0.103017996765653, 0.48257507959915946, -2.7433245449641577, -0.6282460448928144, -0.5724693723912044, 0.31286643080630683, 0.05713077201012279, 0.13309821355939833, 0.030119975060005893, -0.04035066729716144, 0.020221191868914568, -0.027409518632921123, 0.013840019693586907, -0.01749427528626151, 0.009478703066800327, -0.009762387994732067, 0.006323834129690061, -0.0036057088314930185, -0.016137683949349148, 0.005637018343970947, -0.00815942089388527, 0.003290285818646873, -0.0019507382114112744, 0.0014731434703147176, 0.003017918540506312, 2.213021330447775e-05, 0.007084881421493825, 0.0045890034332496665, -0.0015657762068523463]

eigen vectors are [0.3476121584008677, -0.4182225632911257, 0.5505365870188006, 0.8661042073979331, -0.3574014655196348, -0.20298368523507754, -0.7943879411712989, -0.2704583845168411, -4.250846674882634, -1.9950393737743686, 0.7815887551146279, -0.3535014465627632, 0.25390022692009734, 0.13843438220449908, 0.28507321196405383, -1.3442036965946194, 0.2995596012684204, -1.3303997591533705, 0.30817427872773756, -1.3199014689213358, 0.31414721099668685, -1.3117548245682435, 0.3185144976393441, -1.3052907553978885, -1.3183330502622255, 0.3194291580528421, -1.3100039031042459, 0.32274624093715265, -1.303531351827773, 0.3253251268067214, -1.298357820474594, 0.32739149536331363, -1.2941276800194343, -1.2967187203643908, 0.3296622653351482]

eigen vectors are [-0.26262500787935433, 0.4727902859587274, 0.004607480286885616, -0.1625999134543082, 0.8684773657942036, 0.7012059976379708, -0.0017725524403347814, 0.654786192229127,

0.4827230010487541, 0.8185984666506072, -1.108730594140894, 0.552635613633201, -0.12712573440013453, 0.02498268205429096, -0.177343821211837, 0.9303490495997366, -0.19703309166628066, 0.9325010093254804, -0.20915771136105948, 0.9341293825806498, -0.2174842502343619, 0.9353887145285708, -0.2235295261288009, 0.9363854634886403, 0.9343899408040662, -0.22482958560765176, 0.9356676767006012, -0.22935772372356428, 0.9366597751457728, -0.23286893547573417, 0.9374521730862039, -0.23567604394327035, 0.9380996379408488, 0.9377037157851825, -0.2387523898743091]

eigen vectors are [0.507229724293509, -0.5394354336478332, 0.7648765893731594, 1.0, -0.21619773474470722, -0.09588614392004527, -1.0111593847976283, -0.24855778770210404, -6.9736666497470186, -3.079341858333644, 1.0, -0.4384766964245419, 0.38737167517391435, 0.23878164916378355, 0.4268916970767629, -1.9861680241132509, 0.44558835318440243, -1.9625046059405797, 0.4565745541858721, -1.9444786643403502, 0.4642067612072528, -1.9304755210520497, 0.4697952435032035, -1.9193558441537113, -1.9418408139233525, 0.4709605712929126, -1.9274921423121385, 0.4752158106827349, -1.9163386746243862, 0.4785258207921383, -1.907421462165865, 0.48117918212763316, -1.900128684149449, -1.9045977557492, 0.48409649834945667]

eigen vectors are [-0.04169441392933607, 0.4689600426071694, 0.20778581285015485, -0.25448312242279436, 1.0, 0.7343067079545679, -0.001965655546096626, 0.820321651855307, -4.602298384687668, -1.009970878946891, -1.0398716724016497, 0.565262508912681 8, 0.0874364870039814, 0.22683631138338717, 0.039244241403805974, -0.005296311148581218, 0.021096706478196246, 0.016856919467131977, 0.009398490412439354, 0.033882479828690484, 0.0013834367451650603, 0.04718565873252263, -0.004424811343662149, 0.05779386326935623, 0.036129217406917484, -0.00568187541188562, 0.04990513527537437, -0.010016918730667292, 0.060632039999055774, -

0.013375971288585215, 0.06922092084537836, -0.016059782624916558, 0.07625425342509927, 0.07193480656902497, -0.01899883648013852]
eigen vectors are [0.6491392165473799, -0.5146951018413798, 0.9688773183414469, 0.8920056002342169, 0.16043890372803812, 0.1466725237278416, -1.0630655305923462, -0.01727310379604383, -10.810330039322078, -4.382213729528509, 0.935673494984739, -0.37096077357573515, 0.5381610612288938, 0.3955303145153926, 0.573601945196995, -2.6039369914890496, 0.591494630882576, -2.5648856337903965, 0.6016172778457921, -2.5350553937593583, 0.6086834036509693, -2.5118399736857335, 0.6138757504343789, -2.4933806487352195, -2.530830323183909, 0.6149455084096932, -2.5069599218410983, 0.6189247615226668, -2.4883952527179085, 0.6220240853709786, -2.473546057367803, 0.624511252913146, -2.4613970897413067, -2.468847422622586, 0.6272493525835072]
eigen vectors are [-0.2336012465393183, 0.4729056312084427, 0.22776577540096643, 0.0915535615854031, 0.9010469195371557, 0.7418986660676631, -0.25493728934318605, 0.7318073651511968, -0.5742675637477538, 0.4653599466535145, -1.1776181427125045, 0.597233161349146, -0.08889051013220096, 0.06986733704740146, -0.14365895236179438, 0.7959510001148336, -0.16440713495964665, 0.8025305519554817, -0.1772717339294577, 0.8075242318727278, -0.1860913191813184, 0.8113938943548938, -0.19248622942761673, 0.8144611345798004, 0.8082952930674724, -0.1938863564283652, 0.8122377533185628, -0.19864575425415945, 0.8153004798207454, -0.20234819202433815, 0.8177478240170285, -0.20530690825383008, 0.8197483541027348, 0.8185239601199614, -0.20854766600470556]
eigen vectors are [0.7722589830935865, -0.47622872832708574, 1.0, 0.5156132389243878, 0.47133525520322206, 0.3009753079090322, -0.8464653644809262, 0.15644872482352792, -13.989315609958298, -5.461628382945342, 0.9134460472452036, -0.3306979413589713, 0.6645788922854671, 0.5245633510209592, 0.6991590617381712, -

3.1363970621337263, 0.7168383748709609, -3.0846609956746285, 0.7265003819544928, -3.045059067997189, 0.7332612570395747, -3.014196648539377, 0.7382382912117296, -2.989632725673801, -3.039591719894652, 0.7392584205504051, -3.0077758838975672, 0.7430841780833515, -2.98302220428749, 0.746065926858136, -2.9632160350387724, 0.7484600529806898, -2.9470067267615905, -2.9569524345970413, 0.7510973619245623]

eigen vectors are [-0.08099426285584785, 0.6581071612975151, 0.18846687490220593, -0.3123405081745887, 0.8756386751759025, 0.6085347952938813, 0.13258736788877998, 0.9741505847577681, -5.449775508889345, -1.1359081150799768, -1.3688321539468242, 0.739439915951773, 0.08994510810500624, 0.27456183476648144, 0.027304828624015155, 0.0998346747013552, 0.0031741160437175923, 0.12685788469743337, -0.012378060453272558, 0.14770506488492843, -0.023063089170329846, 0.164034531748442, -0.0308214824306311, 0.17707891626953381, 0.1503364551814622, -0.032490542803710394, 0.1673172429516685, -0.03830185396890458, 0.18054976899515945, -0.04280822472463939, 0.19115166818024995, -0.04641101530624279, 0.19983825028951743, 0.19449898442748903, -0.05035939750761931]

eigen vectors are [0.870271143622845, -0.31601012114335053, 0.9741940546854702, 0.034860706299077074, 0.5861307418591146, 0.26518666759523435, -0.44795310417379036, 0.38456680659966336, -17.54632399845788, -6.5833511958204065, 0.7139858857075008, -0.19701961583476138, 0.7886932592039336, 0.6763826736761646, 0.8151106182226693, -3.5974886989877204, 0.8292617873752459, -3.5306571043010266, 0.8363121588388768, -3.4793540243817027, 0.8412668819028394, -3.439297917391368, 0.8449268283189596, -3.4073737265493644, -3.4725082109535697, 0.8456684581208208, -3.4310752073184263, 0.848498969035409, -3.3988211468437193, 0.8507077624171379, -3.37300148950496, 0.8524830952457871, -3.3518620521609663, -3.3648417490594023, 0.8544411220799292]

eigen vectors are [-0.19446001531148918, 0.5033302111874072, 0.45999569849319566, 0.16039180673608652, 0.8936276571996518, 0.6903065610634841, -0.3424043875440735, 0.8285029822411419, -2.068472178729247, -0.02057332709862773, -1.2778610702412652, 0.6618524590357254, -0.036416776161161894, 0.1338429133970738, -0.09673449338058757, 0.6108830428923316, -0.11914327016406781, 0.6238168112012068, -0.1331852754280413, 0.633685488966157, -0.1428049501067117, 0.6413597669493518, -0.14977573215400314, 0.647458170140929, 0.635119408615788, -0.15128659834869526, 0.6429908488031442, -0.15648696382878258, 0.6491119689901043, -0.16051644643118768, 0.6540074114360647, -0.16373582260308747, 0.6580121231921753, 0.6555576434973791, -0.16726113095934209]
eigen vectors are [0.9429590274369207, -0.16485955410360875, 0.7702326172612566, -0.4903682520402279, 0.5517951273383631, 0.13501902968671475, 0.05949899217083694, 0.5131322974149841, -20.085369540941844, -7.386151381070984, 0.5913146847735459, -0.11127777228648109, 0.8777828930235444, 0.7844356863098476, 0.9004387922650894, -3.938580745005686, 0.9122719967320299, -3.860913441306575, 0.9175382620315556, -3.8011483906348786, 0.9212398794285842, -3.754411762767971, 0.9239759929203469, -3.7171210962389343, -3.7934090138016083, 0.9245299886771444, -3.744927762929283, 0.9266477607704576, -3.7071693232991416, 0.928300801206269, -3.676931384562472, 0.9296297186095555, -3.6521659943649447, -3.667380909547732, 0.9310956935573679]
eigen vectors are [-0.1274500671406974, 0.7841684551166858, 0.18318599453599105, -0.1287660020154853, 0.5102099800670818, 0.3390010251287388, 0.12486291637172642, 1.0, -5.451688938162227, -1.0518200146154064, -1.5761769531020164, 0.8449271077412134, 0.07053231821433055, 0.28325334472893393, -0.0013669080049726725, 0.2573243365325906, -0.02936767796216382, 0.2854429415942624, -0.04738099676527955, 0.3072054104915948, -0.05978281695454559,

0.3242875251705431, -0.06880113928100975, 0.3379534858901755, 0.3098515482394296, -0.07073325523465217, 0.3276752670318601, -0.07750565666981524, 0.3415739550267142, -0.0827603037673432, 0.3527158145810833, -0.08696333130244442, 0.36184918929632404, 0.35623153774232025, -0.09157204290415953]

eigen vectors are [0.9858724540034506, 0.06327881873556133, 0.5280371436395312, -0.843436450081474, 0.16684124606291267, -0.244553783806678, 0.5576215876549268, 0.6171322148778494, -22.469186352328723, -8.102982004881403, 0.3658514095251342, 0.025699216233159985, 0.9509678335154694, 0.8906280701861925, 0.9654160186193487, -4.1744513131776175, 0.9729357437764249, -4.085920577839474, 0.9751429051096129, -4.017608798555461, 0.9766925817372993, -3.964092994728274, 0.9778404130533375, -3.9213386382433915, -4.009057561860898, 0.9780710033565075, -3.9533703965982476, 0.9789628941632428, -3.909976349502751, 0.9796596077300048, -3.875209372100083, 0.9802200817964974, -3.8467233255028255, -3.864235123931945, 0.9808388477639745]

eigen vectors are [-0.14995109536348109, 0.577558829267646, 0.6214274383804375, 0.030086167559868073, 0.614013535604104, 0.3601812214645606, -0.16779248583180115, 0.8763247921496278, -3.7361530743258275, -0.5716740090658194, -1.3780865913705693, 0.7287506712215652, 0.02215367918366367, 0.2053241600369, -0.04271445950715352, 0.39809497999724763, -0.06696255803801024, 0.41829224402105786, -0.08236782589207285, 0.43381079808908823, -0.09293822937093464, 0.4459338558219488, -0.10060608308861406, 0.4555991688529143, 0.4358895275055424, -0.1022631185514102, 0.44842790277152283, -0.10799407063702993, 0.4581913194455559, -0.11243641721043447, 0.4660086748115482, -0.11598681106477737, 0.4724100437424387, 0.46847994364437956, -0.11987605806548378]

eigen vectors are [1.0, 0.24682933446266533, 0.1688554476818977, -1.0061524658492047, -0.10206909639150386, -0.4332836227216297,

0.8801445921824158, 0.6786793677809857, -23.774725429239542, -8.470530687065903, 0.20508869025629958, 0.1196231661214277, 0.9865509281752511, 0.950584697175644, 0.99560330076101, -4.273127081010934, 0.9999851241998766, -4.178393024722619, 1.0, -4.105153819325627, 1.0, -4.047705847119084, 1.0, -4.001768791458481, -4.0962109002101315, 1.0, -4.036300133497959, 1.0, -3.989597124024109, 1.0, -3.9521669886812782, 1.0, -3.9214903337663203, -3.9403572384968832, 1.0]

eigen vectors are [-0.17590819351909362, 0.8381707226908744, 0.2505957309563954, 0.1753797792066275, 0.18001935576541195, 0.11726950514543093, -0.021670275214446068, 0.9734216300962102, -4.883271470911847, -0.8213792482091452, -1.6921867811630082, 0.898919893651531, 0.03739402843018875, 0.265020488219093, -0.04019464363403665, 0.4423964802090548, -0.07037051620919042, 0.4688842664617278, -0.08970389691925933, 0.4894200474093052, -0.10302687536201208, 0.5055572295913918, -0.11272136720904675, 0.5184773884141946, 0.4918738958578656, -0.11479544342144982, 0.5087382585104041, -0.12208298665763481, 0.5218939501271284, -0.1277387577520697, 0.5324435522402993, -0.13226356511414994, 0.5410937614775424, 0.5357716922847133, -0.13722623465930905]

eigen vectors are [0.9830178383631668, 0.4546796969638217, -0.15927303504313184, -0.856638951466471, -0.531483469409781, -0.6814390005554869, 1.0, 0.7297578182305416, -24.67148659651854, -8.67929746771332, -0.023922095221403164, 0.24452256774338432, 1.0, 0.9966741792174799, 1.0, -4.2508619420864076, 1.0, -4.151213844138212, 0.9970860455393531, -4.074037301912874, 0.9950259258775088, -4.013430297156583, 0.9935051831363565, -3.9649267552614096, -4.064827009983164, 0.9931948025204531, -4.001496211296679, 0.9920214910694457, -3.952109214702161, 0.9911059862198731, -3.9125158793771657, 0.9903702337729073, -3.8800577420823616, -3.9000282771791603, 0.9895589861810473]

eigen vectors are [-0.10545680866683885, 0.6913092640011199, 0.6679389924126434, -0.10593321255024372, 0.042125691697348915, -0.19752143258949117, 0.12744507702138225, 0.8638860915394281, -5.4010538485651365, -1.136183104440396, -1.4715629536296004, 0.7929112149825562, 0.08071025381224955, 0.2768264024247328, 0.012335949937011328, 0.18219755203827562, -0.013801280912459787, 0.20983970414250527, -0.03065499728946047, 0.23121751655444042, -0.04225718613208197, 0.2479887475974026, -0.050692619010115826, 0.26140046712456083, 0.2338591252134628, -0.05250308958636611, 0.25133557480254876, -0.05883347460560478, 0.26496131254701516, -0.0637446782572373, 0.27588277770308717, -0.06767261215971793, 0.2848342819514197, 0.27933017684634803, -0.07197905214435134]
eigen vectors are [0.9372668859353239, 0.5845448836882491, -0.506934497128297, -0.5114984722421537, -0.49818920719860027, -0.458799430982391, 0.8131096212022175, 0.817743777331656, -24.56633285800569, -8.540292070387109, -0.20871050258096918, 0.3365851523559103, 0.977463585133239, 1.0, 0.9695558559036541, -4.090477145316475, 0.9661716203189107, -3.9904337506979095, 0.9611359416114603, -3.91290404240705, 0.9576011270449442, -3.8519951082849446, 0.9550003397962732, -3.803235598358337, -3.903727825624804, 0.954464103990057, -3.840037210736585, 0.9524686996414643, -3.7903628647246252, 0.9509134419476314, -3.7505345662958876, 0.9496646791186678, -3.7178805363569336, -3.737974522933653, 0.94828922637932]
eigen vectors are [-0.2209034921734199, 0.8308447022200377, 0.40195299522734396, 0.3570859808364809, -0.10056461199446935, -0.11816628063444032, -0.11650387974808933, 0.9027966796326077, -3.9169231241908653, -0.49329051794834444, -1.7212108822533758, 0.9052080357220876, -0.003589696806587998, 0.2271833259408199, -0.08300444602924532, 0.629926506682418, -0.11377100573022324, 0.6527769949564759, -0.1333787611672275, 0.670513273469655, -

0.14689608758678072, 0.6844606587598684, -0.1567346493284072, 0.6956331373198521, 0.6726152343173741, -0.15883898587129064, 0.6872028412284498, -0.16623724661830008, 0.6985855017661424, -0.1719795001096038, 0.7077152575920277, -0.17657384222052674, 0.7152025731171766, 0.7105953697040847, -0.18161314833165348]

eigen vectors are [0.8626016459745839, 0.6942086323017892, -0.7444708430630268, 0.007990423597836324, -0.4952759792220744, -0.24212384259727626, 0.4696150246199912, 0.8908228311122439, -23.837270582833302, -8.182854122306983, -0.4145226467153322, 0.4334910491250113, 0.9284993328696594, 0.9787829909525486, 0.9110551725572793, -3.8079863573278665, 0.9039463062600183, -3.7099921405086858, 0.8966623569322232, -3.6340224345600327, 0.8915723722818546, -3.5743241603412703, 0.8878364535556431, -3.5265250315213503, -3.6250723002350047, 0.8870594418728518, -3.5626232021465447, 0.8842060605698147, -3.5139124690271886, 0.8819839663604627, -3.474853624024154, 0.8802010580837899, -3.442828217221262, -3.4625370087880936, 0.8782389566472245]

eigen vectors are [-0.0663448903224016, 0.8219685286140248, 0.6102342048443596, -0.050558850077669915, -0.5227804724834907, -0.6649181949220836, 0.29173061040356585, 0.8603487932512462, -7.025995516969587, -1.6801691015182048, -1.582195165230026, 0.8653044843731785, 0.13614031419276448, 0.347352665028695, 0.06340270417351525, -0.013932482252544419, 0.03506838442396685, 0.02107338078529797, 0.016575610025744033, 0.04825286644242376, 0.003808543326140772, 0.06963003095997251, -0.005492321218696554, 0.08675618409509929, 0.051441618223281214, -0.007476590407987334, 0.07381709737818247, -0.014481303964526824, 0.091275781920563, -0.01991979529406437, 0.10527854073441575, -0.02427222131378559, 0.11676199808828883, 0.10969470317813337, -0.029047603672036525]

eigen vectors are [0.7626522270863698, 0.7096458904259071, -0.9149609877342385, 0.49486178409735704, -0.043596721950612025,

0.34308123215568576, -0.014849218555007352, 0.9846659675939458, -22.069325308715147, -7.472611259978803, -0.5526782420581802, 0.48545561754080063, 0.843950023402035, 0.9124827524380306, 0.8194111399917421, -3.398189544428958, 0.8099768967491986, -3.307004637629978, 0.8015050673834416, -3.2363645111083357, 0.795619748417052, -3.1808791255163116, 0.7913163173932379, -3.1364669037240422, -3.2279666870775814, 0.7904104560468146, -3.169969392030889, 0.787145559990731, -3.1247362385607134, 0.7846063890843282, -3.088469349825265, 0.7825713442748038, -3.0587355764572837, -3.0770314114529675, 0.7803346926990992]

eigen vectors are [-0.25731369808935456, 0.7891396115639155, 0.5954122225719293, 0.29043586424896056, -0.47863269158690025, -0.5223190573228217, -0.001775142725100294, 0.7647604120305623, -2.7147953013382526, -0.12491305264694891, -1.665357480481418, 0.8667329294475415, -0.04661429989810314, 0.176661645137044, -0.1234979951915225, 0.7948458291644638, -0.15340822849928437, 0.8128302363069317, -0.17237952117071031, 0.8268363125413764, -0.1854700603223333, 0.8378733683658033, -0.19500402761581362, 0.8467272978814554, 0.8284409478176615, -0.19704008868307948, 0.8400189238620558, -0.20421656829680937, 0.8490590720098982, -0.209787900585571, 0.8563138160607626, -0.21424628382695515, 0.8622660902649676, 0.858601498853419, -0.21913743750971176]

eigen vectors are [0.6392919605047511, 0.6816899146396852, -0.9325617583148019, 0.8757077429967813, 0.20435633389849256, 0.6771501802647297, -0.3961598395349214, 0.9843581988144812, -19.418870804200225, -6.493935618616271, -0.6354003880818665, 0.5017209052230468, 0.729080818693751, 0.8084772982208215, 0.6999597258625563, -2.878192251124834, 0.6891143215752527, -2.797494822083796, 0.6801093944861674, -2.7350162053744707, 0.6738751975490538, -2.6859595374945355, 0.6693267441837515, -2.6467031222999142, -2.7275304615481493, 0.668362148572447, -

2.676287081635206, 0.6649254500412739, -2.6363250241561795, 0.6622548140726133, -2.604286807989606, 0.6601158554726806, -2.578021683158761, -2.594181126172651, 0.6577668912958787]

eigen vectors are [-0.03732613008727977, 0.9358175213920019, 0.5057297386293966, 0.190178959438176, -0.7970211894749782, -0.809587793094245, 0.22842020871664953, 0.9166219202056586, -8.47385556571541, -2.1414605130773623, -1.7112234732595828, 0.9445898010678249, 0.1829960080776227, 0.4111033979480739, 0.10448429297018957, -0.16651174857542378, 0.0737620127738326, -0.12496709554276433, 0.053574333462694, -0.09267564678344992, 0.039620950693557096, -0.0672596091171003, 0.02944765112240393, -0.0468871637478318, -0.08894559271981187, 0.027282355039568846, -0.06230883210093679, 0.01960969704694733, -0.04152046953712978, 0.013650692732934584, -0.024843880395711306, 0.008880415402926398, -0.01116532552147943, -0.01958577686677536, 0.0036449508132047575]

eigen vectors are [0.49717443022479657, 0.5695497930570333, -0.8476605563360332, 0.9985838615687918, 0.555009387763469, 1.0, -0.642646973546415, 0.9158541693663548, -15.694387660106871, -5.184069284435926, -0.6081792671378393, 0.45224915239768265, 0.5806714632523593, 0.6567406817857699, 0.5521018425155892, -2.255449840859211, 0.5418226087988485, -2.1900849304210794, 0.5336558024761419, -2.139538741083307, 0.528025638523001, -2.0998817865357453, 0.52392955417595, -2.068164494886374, -2.1333896008176128, 0.5230531445877524, -2.092019909819157, 0.519974044610757, -2.0597648092657073, 0.5175837948409516, -2.033910080828492, 0.5156710776171166, -2.012717537799196, -2.0257527455456783, 0.5135727252001209]

eigen vectors are [-0.2809708105733507, 0.7461564940952752, 0.7573497995246958, 0.08817354627904381, -0.9090326441505543, -1.0031619925899709, 0.24200748896300373, 0.6033138345307255, -1.6475096724380458, 0.18033814549699917, -1.5735313897823677,

0.8115057295698472, -0.08124712660640625, 0.13031881833835302, -0.1535222009320517, 0.9101601486329483, -0.1819833277654065, 0.9237241708655033, -0.1999798708627855, 0.934364985489786, -0.21241696336710583, 0.9427882435252183, -0.22148471141860399, 0.9495665716812222, 0.9354829485892209, -0.22341534015049686, 0.9443802524930928, -0.2302533879397248, 0.9513366029234326, -0.23556397116633387, 0.9569252372269573, -0.23981501042749345, 0.9615147940496639, 0.9586855860145922, -0.24448036174966606]

eigen vectors are [0.3398407200586651, 0.41711019627641693, -0.6310835261682288, 0.8685363472105481, 0.5177106744605058, 0.8728672660907547, -0.6106948828549927, 0.6965329573945176, -11.081128659380132, -3.6298127347752334, -0.48282296272358405, 0.3459341679253006, 0.4052256279599891, 0.4656391806588685, 0.3822537851019475, -1.5526123510500118, 0.37410765126255746, -1.5063241801167724, 0.36780753295357854, -1.4705521424384587, 0.3634719646009427, -1.4424976075132636, 0.3603214789123158, -1.4200660535070966, -1.4661654681680507, 0.35964473018408855, -1.4369197901706248, 0.357281709437803, -1.414119947173929, 0.3554481540742728, -1.3958459037701578, 0.3539814637580838, -1.380868238096028, -1.3900795110192112, 0.3523731465814684]

eigen vectors are [-0.021892076509316266, 1.0, 0.42715283455405273, 0.4066790074639688, -0.815020881961226, -0.7421496205958702, 0.09249711424821383, 0.9872004765010773, -9.367008697074638, -2.4126336349696333, -1.8044500545476945, 1.0, 0.21075675792023685, 0.4507971422660544, 0.12779025163185917, -0.2507582794187477, 0.09542362204965904, -0.20522248017328631, 0.07410596211726742, -0.1698370287645353, 0.05937140211700667, -0.14198951010691832, 0.04862846601124682, -0.11967004580776487, -0.1657401221188529, 0.046341824838218325, -0.13656102765394437, 0.03823951546064537, -0.11378915527026558, 0.031946706328631995, -0.09552177698149748,

0.02690913664607076, -0.08053868493887521, -0.08976191098687084, 0.02138023115097929]

eigen vectors are [0.1725311904068664, 0.21631415434606466, -0.34063137776943236, 0.4951636799784924, 0.3569001149130205, 0.5470687253186791, -0.37548566763920516, 0.3844708366425129, -5.726194443605213, -1.8638741550248314, -0.26483759900236353, 0.1862755365372125, 0.20801509288560988, 0.2411345559758594, 0.19529768849300172, -0.7908977044198306, 0.19085443008166667, -0.7669717056713958, 0.18746574948192649, -0.7484958418741231, 0.18513852080782134, -0.7340132188626849, 0.1834497968315468, -0.7224374351441978, -0.7462084111042016, 0.18308548719733586, -0.7311237642677174, 0.18182207004056308, -0.7193655131601153, 0.1808422545094237, -0.7099424314165719, 0.18005883066983644, -0.7022199456916531, -0.7069684851001788, 0.1792002024882986]

eigen vectors are [-0.2891576758597114, 0.7285863169669518, 0.8200063862766722, -0.01634406194298168, -1.1070465455964957, -1.2244949171564008, 0.3661413139819448, 0.5299318810178555, -1.2148930284577546, 0.30006207947242036, -1.5293462560769877, 0.785665220536748 1, -0.09470080417885984, 0.11129670341190978, -0.16470706103879698, 0.9517914233371336, -0.19247100476223764, 0.9635525654178776, -0.2100084777644408, 0.9728261575960667, -0.22213840432099963, 0.9801902162173645, -0.2309873068876496, 0.9861289978648213, 0.9737378297942274, -0.23286816272838357, 0.9815539046827625, -0.2395478518685038, 0.987670415599645, -0.2447365012378604, 0.9925879906434496, -0.24889063322997193, 0.9966289978922684, 0.9941357413827956, -0.2534505229283462]

natural frequency in hertz [0.007703503204665013, 0.022991600605499613, 0.029906056449980054, 0.05339928477376626, 0.07531757807865841, 0.07104669894212147, 0.05333697346367255, 0.027231599568402928, 0.012772210975147915, 0.010677490254064573, 0.023010982615878518, 0.023787264146577062, 0.009696317064634398,

0.014113478513246161, 0.008554170055072698, 0.007951486848923174, 0.008226394161470186, 0.007895503869950136, 0.008054194602876177, 0.007855526590241307, 0.007954627092713318, 0.007826517972431668, 0.00789233437775372, 0.00780498893290061, 0.007850875842142314, 0.00788011335206704, 0.007821031537877629, 0.00784046858362362, 0.007799552608904261, 0.007813139572461643, 0.007783574311242421, 0.007793453617436156 6, 0.00777135850232812, 0.007778775096071561, 0.007774098757903454 5].

CHAPTER – V

RESULTS AND DISCUSSION

5.1 Results

This project carried out in ANSYS APDL Software and Python for validation purpose for both static and vibration analysis for any truss structure.

Fig 5.1 deformation of stress in model analaysis

5.1.1 Results comparison for validation between Ansys APDL and Python Code:

Static analysis results of python code were compared with Ansys APDL software. The below tables shows the comparision of nodal displacements , element stresses and reaction forces at the supports.

Table 5.1 Nodal displacements validation:

node Number	Nodal displacements in mm obtained from Ansys APDL in MM		Nodal displacements in mm obtained from Python Code in MM	
	Qx	Qy	Qx	Qy
1	0.000	0.000	0.000	0.000
2	80.572	-52.717	80.572	-52.717
3	2.686	-103.880	2.686	-103.880
4	75.201	-150.400	75.201	-150.400
5	10.072	-192.650	10.072	-192.650
6	65.801	-227.920	65.801	-227.920
7	20.815	-257.000	20.815	-257.00

				0
8	53.715	-277.540	53.715	-277.540
9	33.572	-290.720	33.572	-290.720
10	40.286	-294.590	40.286	-294.590
11	47.001	-290.720	47.001	-290.720
12	26.857	-277.540	26.857	-277.540
13	59.758	-257.000	59.758	-257.000
14	14.772	-227.920	14.772	-227.920
15	70.501	-192.650	70.501	-192.650
16	5.372	-150.400	5.372	-150.400
17	77.887	-103.880	77.887	-103.880
18	7.9378E-13	-52.717	7.9378E-13	-52.717
19	80.572	0.000	80.572	0.000

Table 5.2 Element Stresses validation:

Element Number	Element stresses from Ansys APDL in N/mm2	Element stresses from Python code in N/mm2
1	149.21	149.21

2	410.32	410.32
3	596.83	596.83
4	708.74	708.74
5	746.04	746.04
6	708.74	708.74
7	596.83	596.83
8	410.32	410.32
9	149.21	149.21
10	-298.42	-298.42
11	-522.23	-522.23
12	-671.44	-671.44
13	-746.04	-746.04
14	-746.04	-746.04
15	-671.44	-671.44
16	-522.23	-522.23
17	-298.42	-298.42
18	-298.44	-298.44
19	298.44	298.44
20	-223.83	-223.83
21	223.83	223.83
22	-149.22	-149.22
23	149.22	149.22
24	-74.61	-74.61
25	74.61	74.61
26	0	0
27	2.78E-12	2.78E-12
28	74.61	74.61
29	-74.61	-74.61
30	149.22	149.22
31	-149.22	-149.22
32	223.83	223.83
33	-223.83	-223.83
34	298.44	298.44
35	-298.44	-298.44

Table 5.3 Reactions forces validation:

Reaction forces	Reaction forces from	Reaction forces from

	Ansys APDL in N	Python Code in N
R1x	1.1816E-08	1.1816E-08
R1y	1119999.9	1119999.9
R19y	1199999.9	1199999.9

From the above comparison we found almost zero error between the results obtained using python code and ANSYS software.

From this we can confirm that the python code which we developed is working fully with almost zero error.

5.1.2 Modal Analysis using Python Code:

Before going to add modal results we are comparing natural frequencies obtained from python and ansys software for cross checking.

Table 5.4 Natural frequencies Validation:

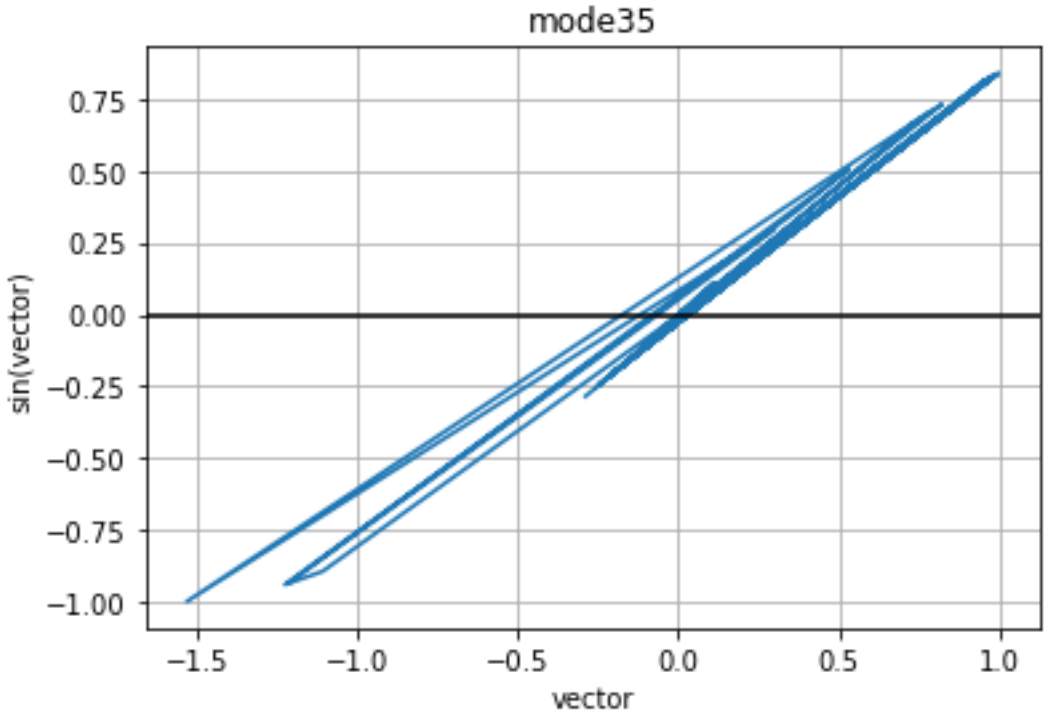
S.No	Natural frequency from Ansys APDl In Hertz	Natural frequency from Python Code In Hertz
1	0.00770	0.00770
2	0.02298	0.02299
3	0.02989	0.02991
4	0.05337	0.05340
5	0.07528	0.07532

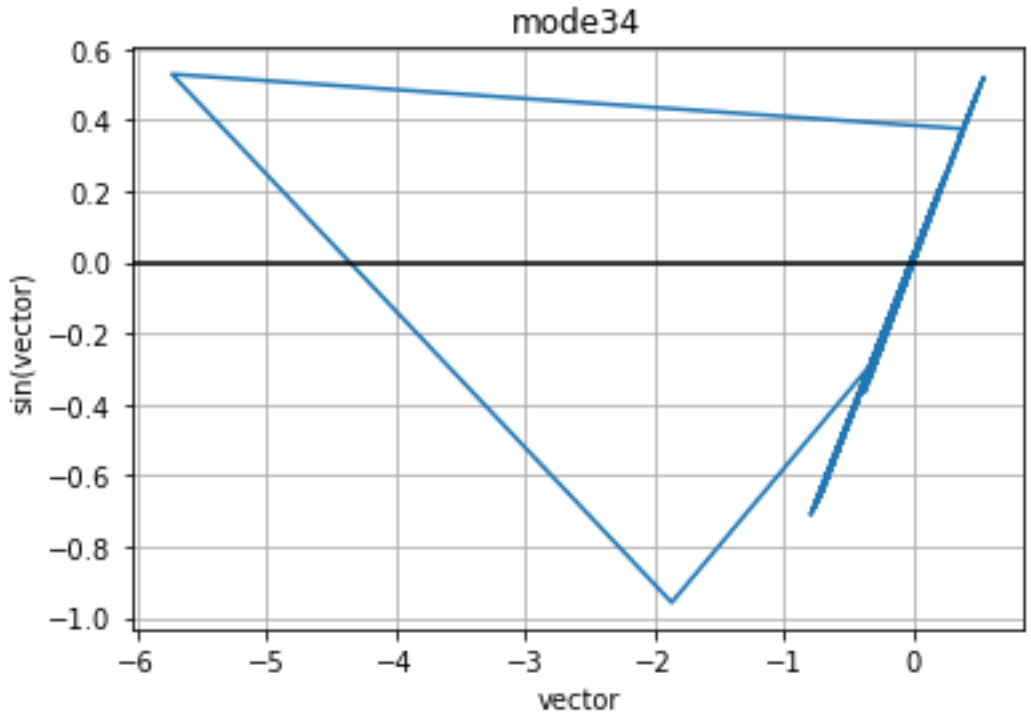
Just for validation we compared any of five natural frequencies from both python and ansys software and we found that the results obtained are very close.

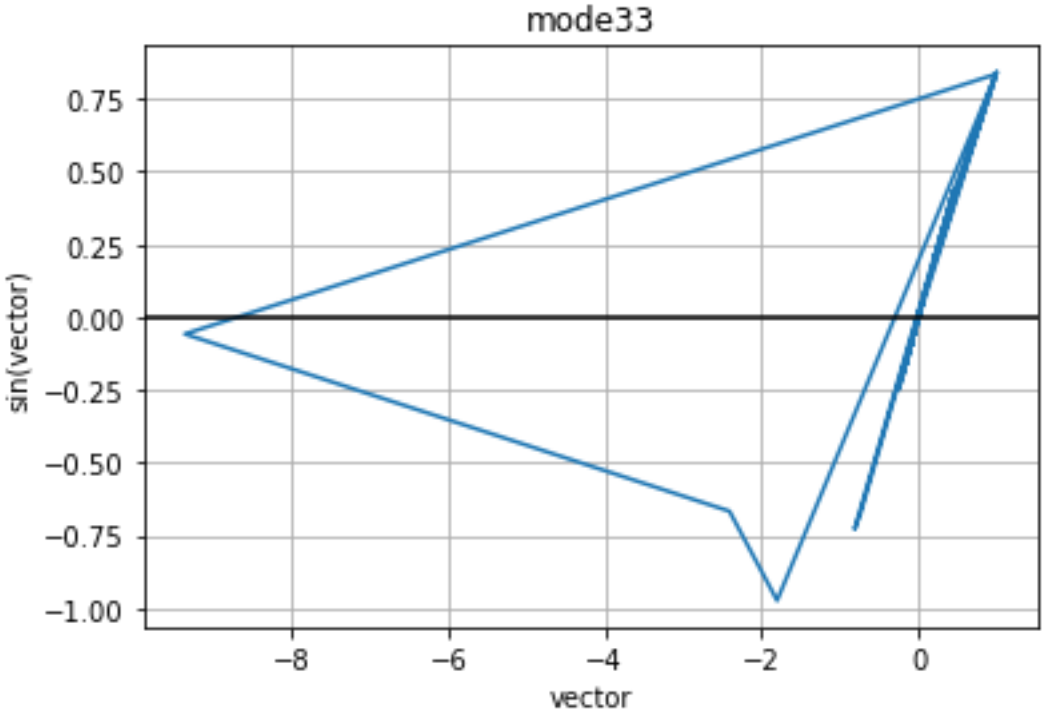
Already results obtained from ansys software were discussed in chapter 3. In this chapter we will show the results obtained from python code.

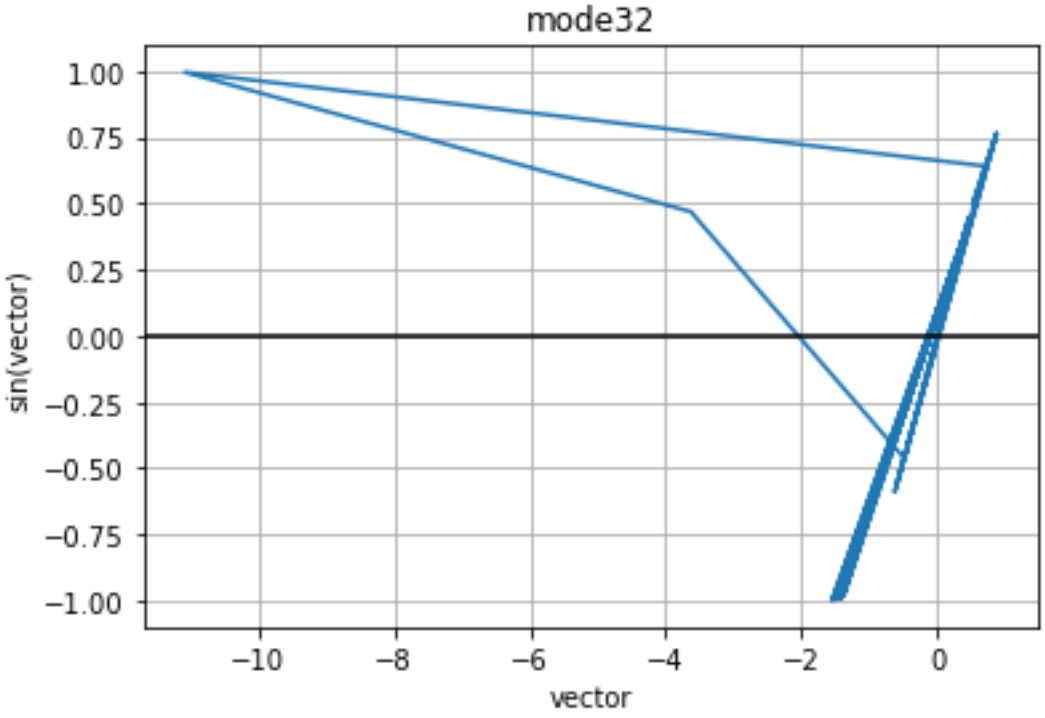
5.2 Plotting Mode Shapes:

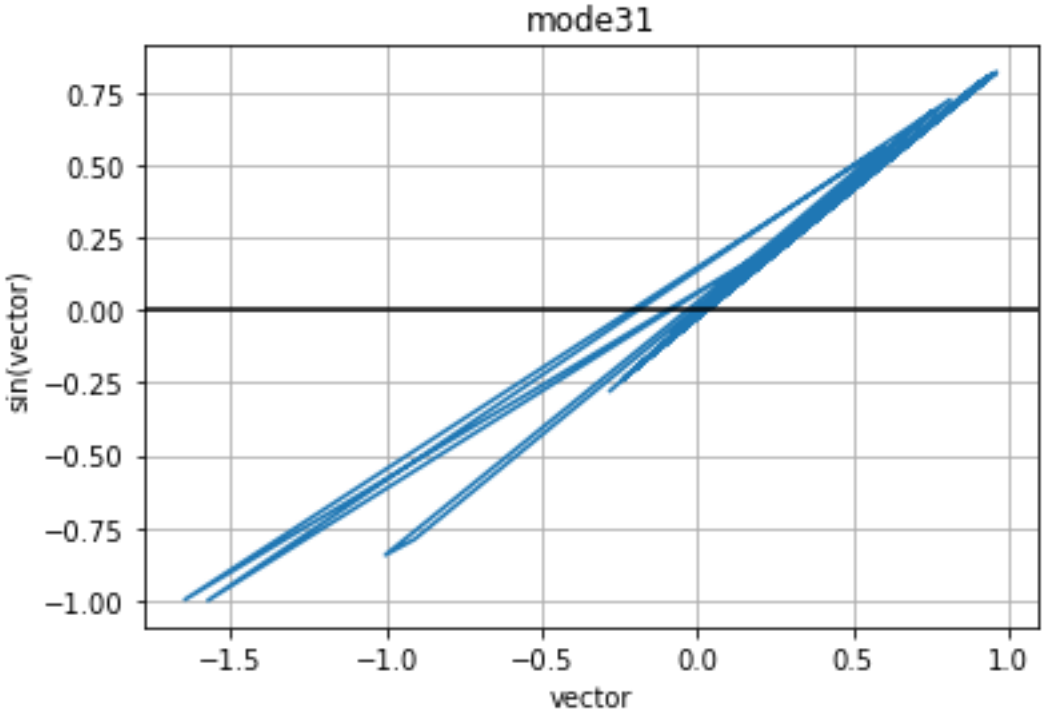
From the data of current project we obtained 35 mode shapes for the corresponding natural frequencies. The following fig shows the model shapes of truss structure.

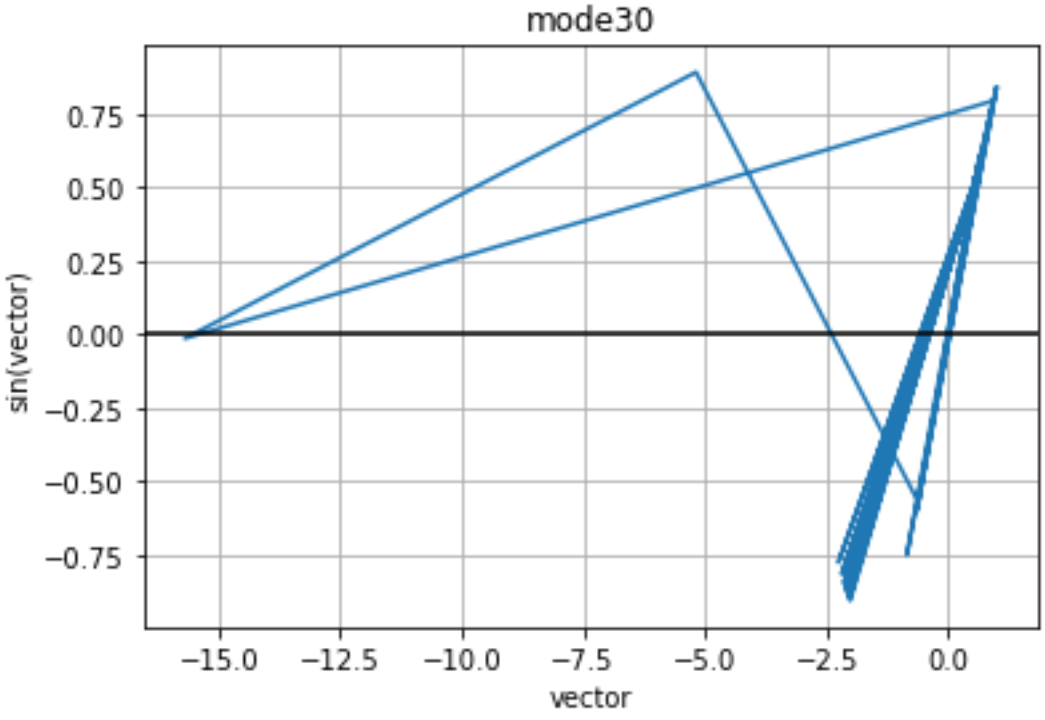


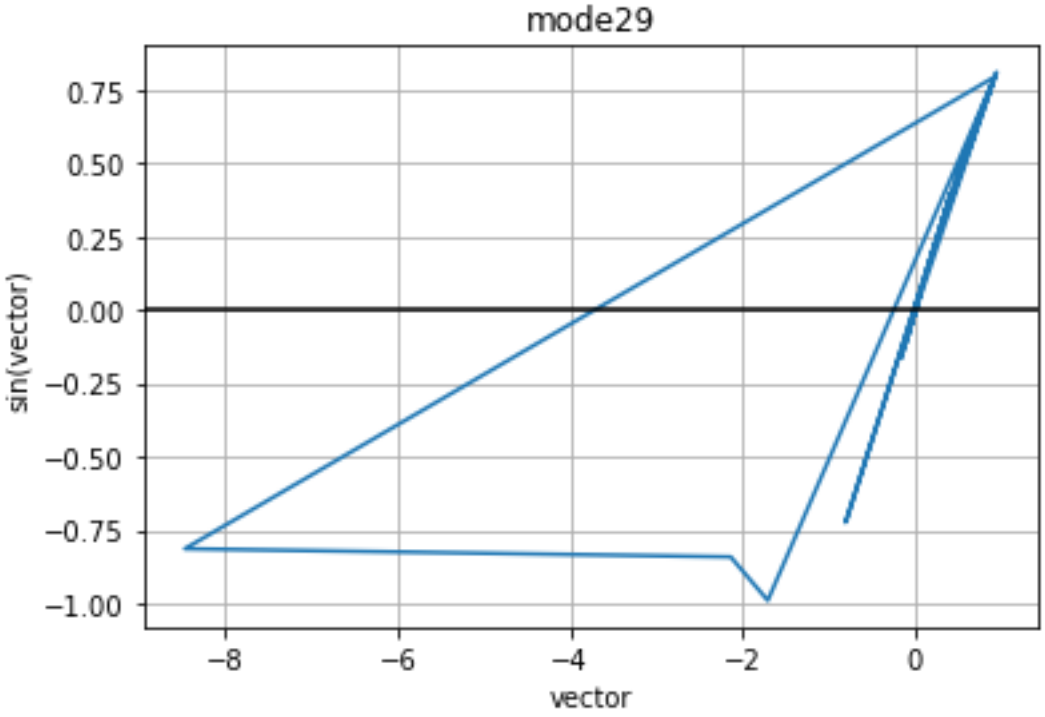


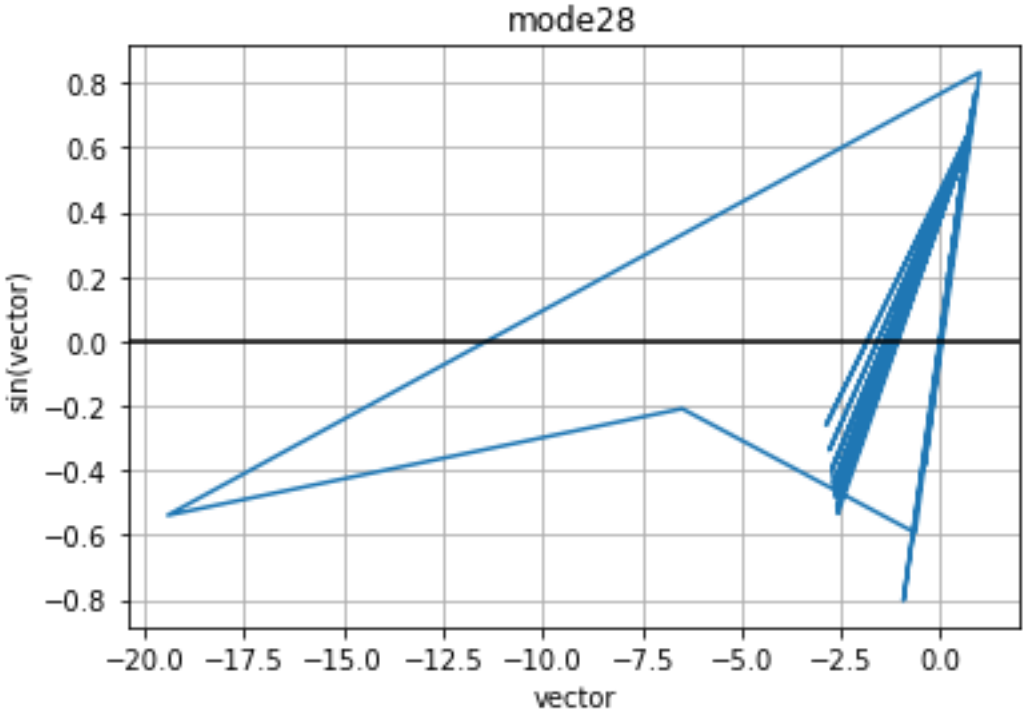

mode33
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−8
−6
−4
−2
0
vector

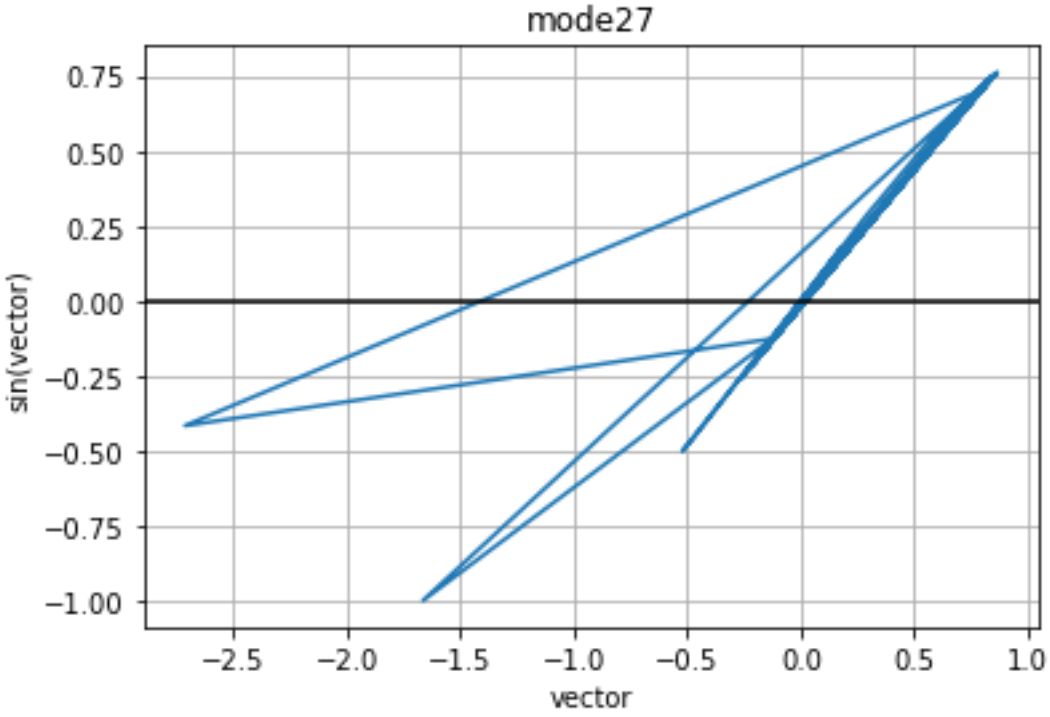

mode32
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−10
−8
−6
−4
−2
0
vector

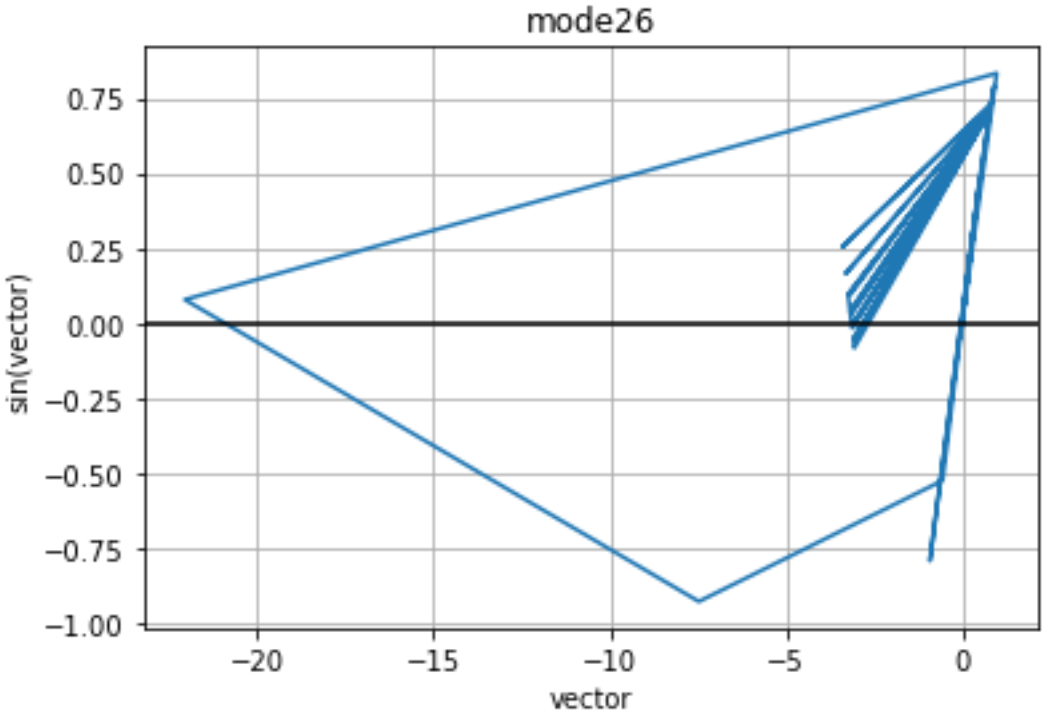

mode31
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−1.5
−1.0
−0.5
0.0
0.5
1.0
vector

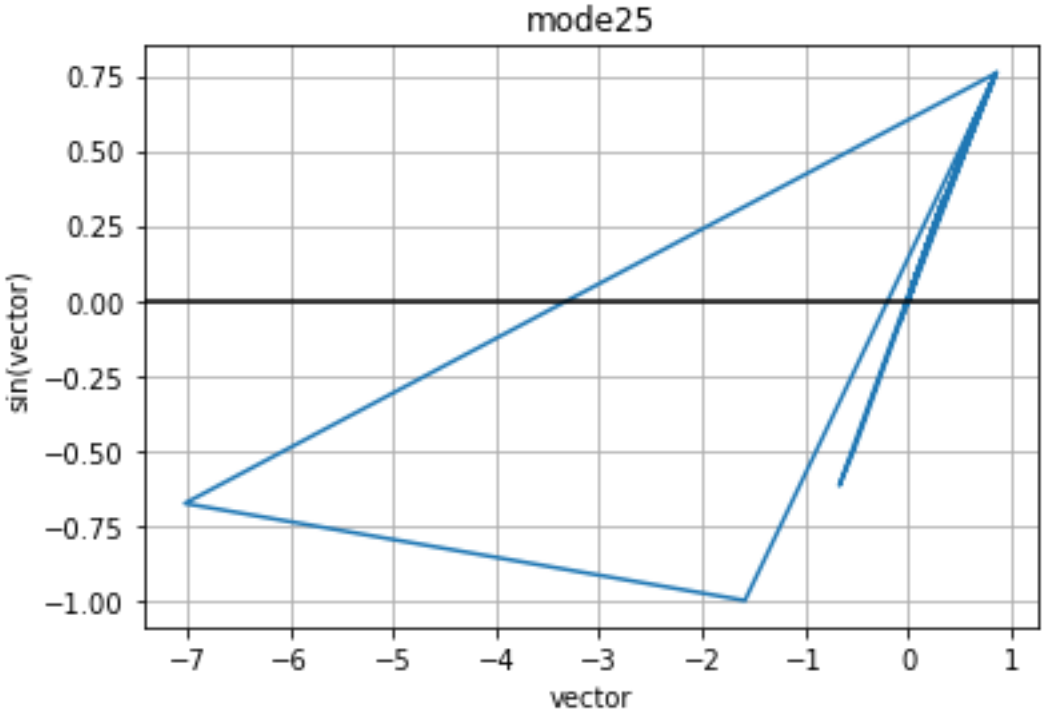

mode30
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
sin(vector)
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
vector

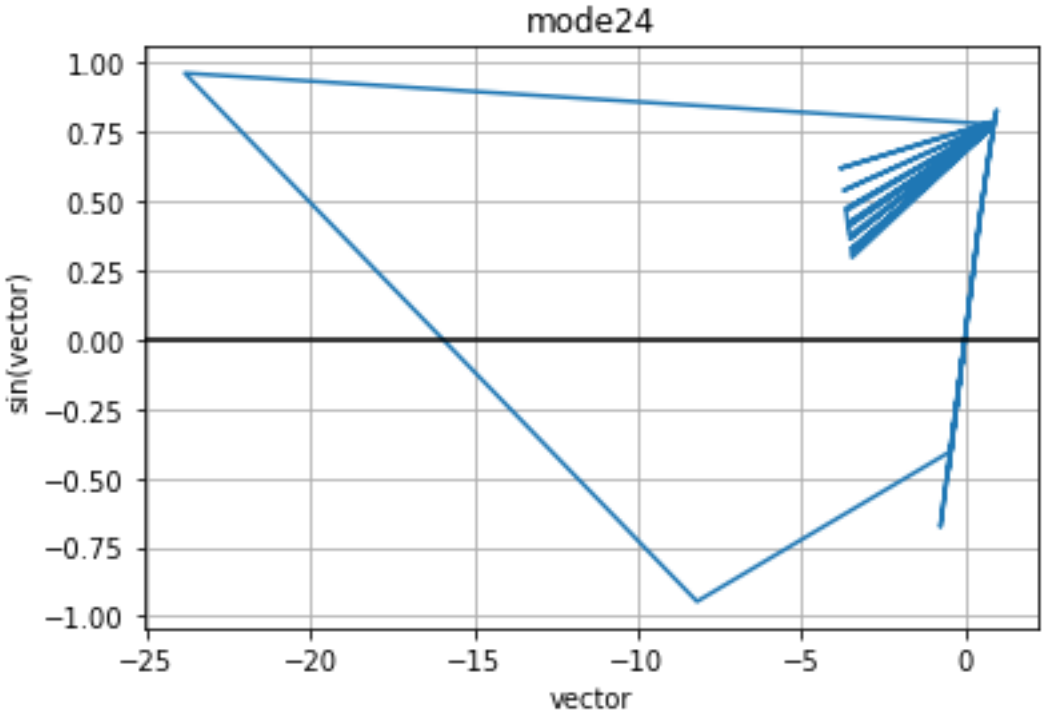

mode29
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−8
−6
−4
−2
0
vector

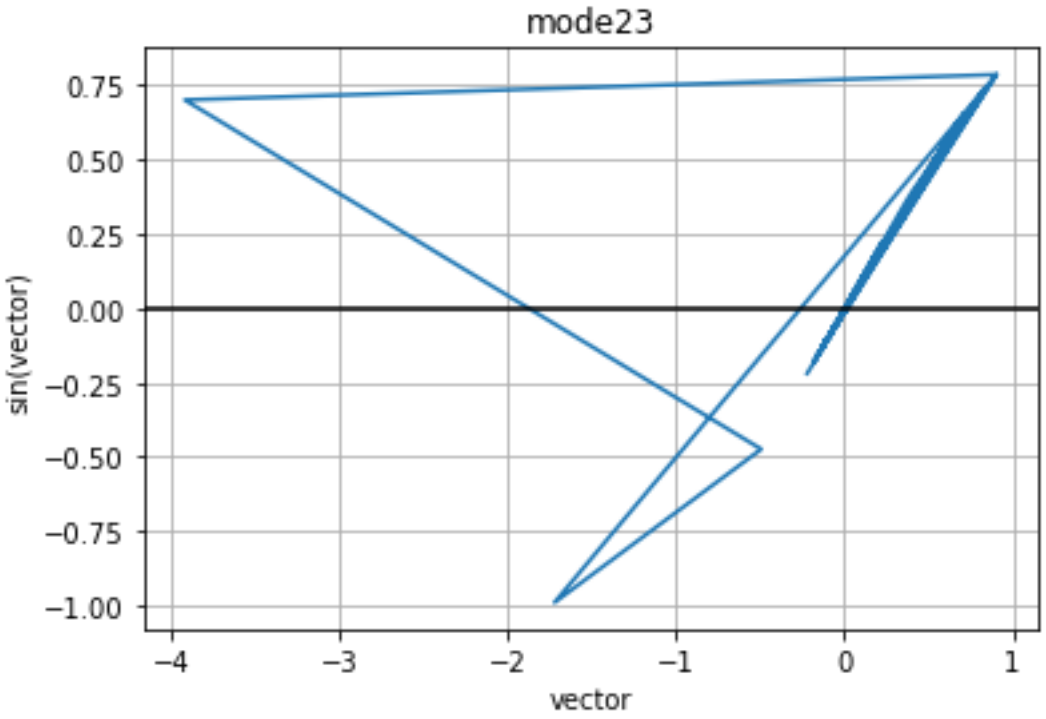

mode28
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
sin(vector)
−20.0
−17.5
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
vector

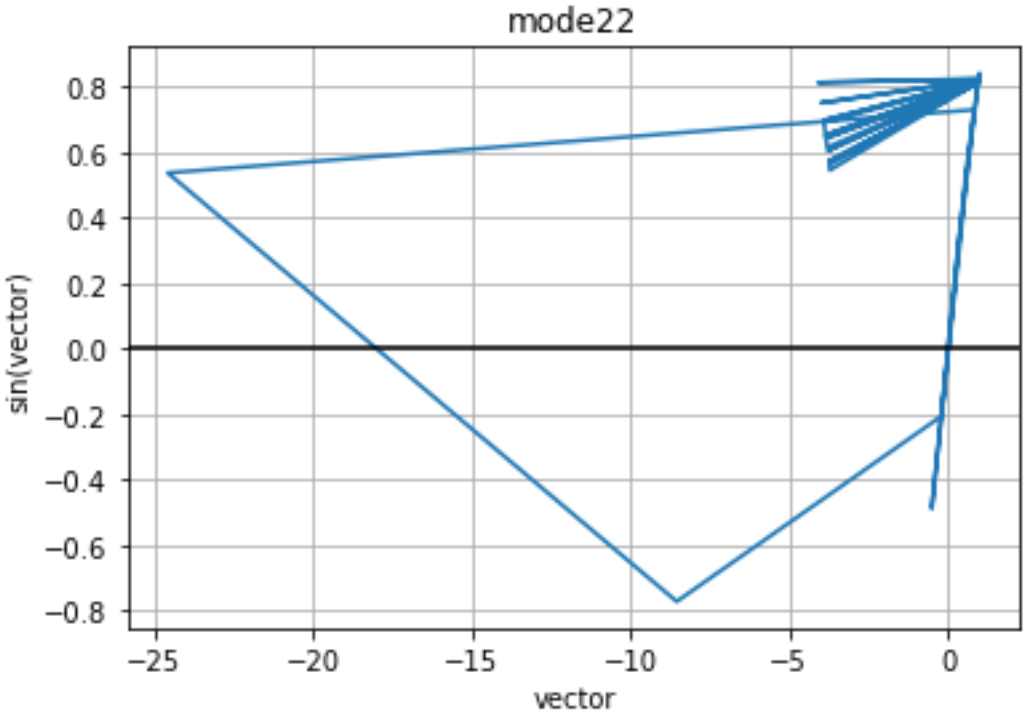

mode27
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
vector

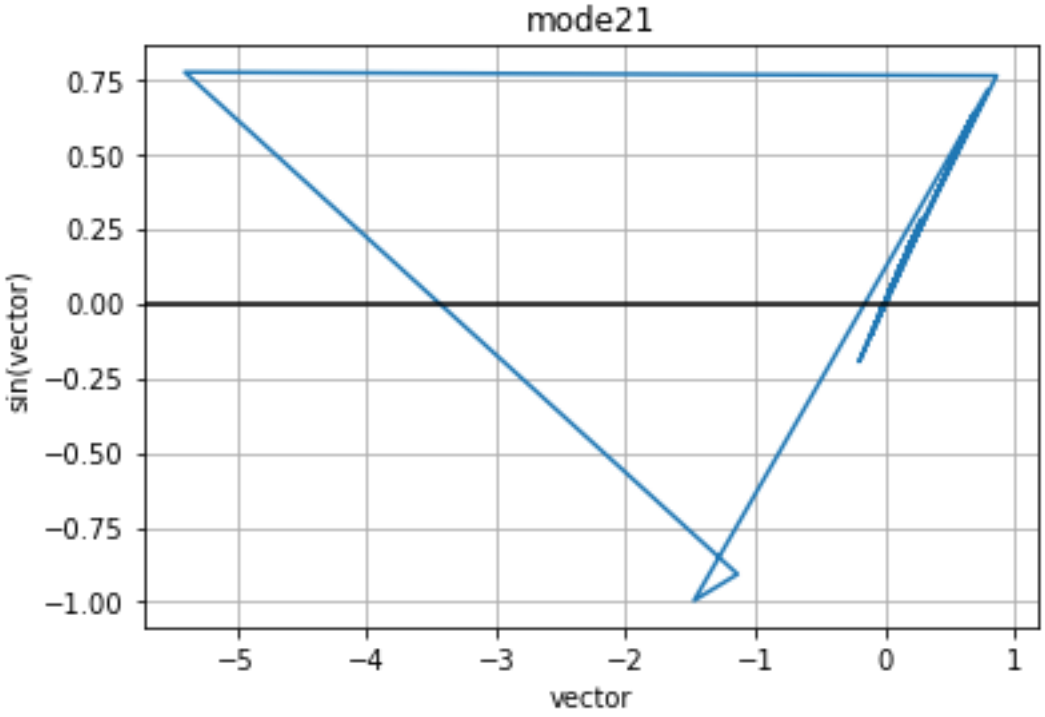

mode26
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−20
−15
−10
−5
0
vector

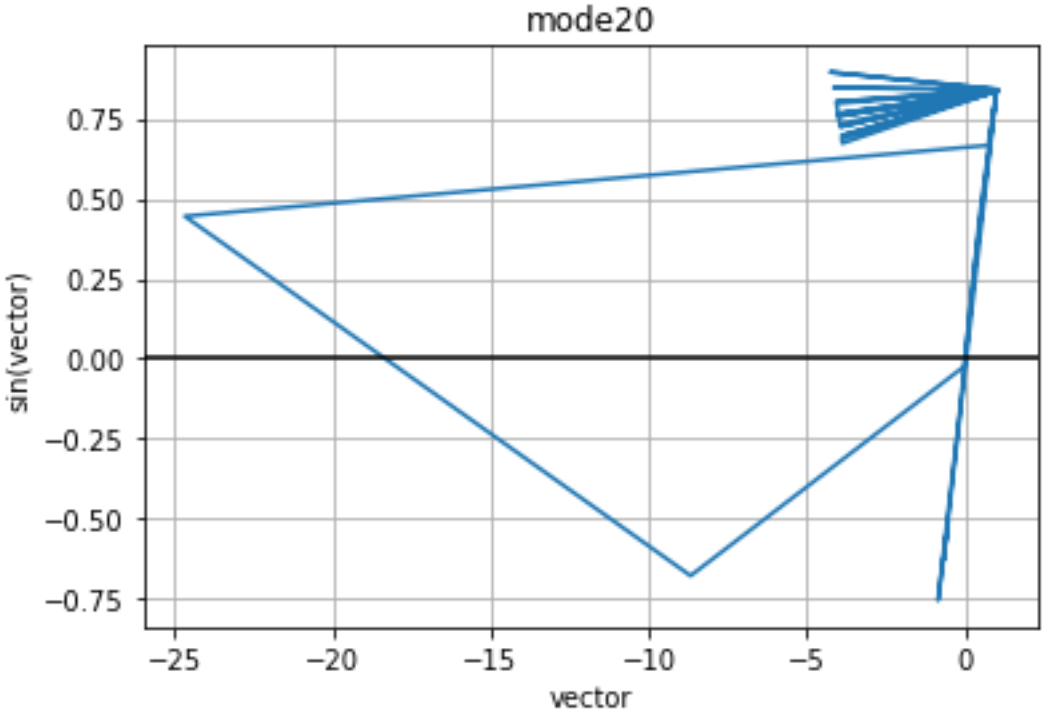

mode25
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−7
−6
−5
−4
−3
−2
−1
0
1
vector

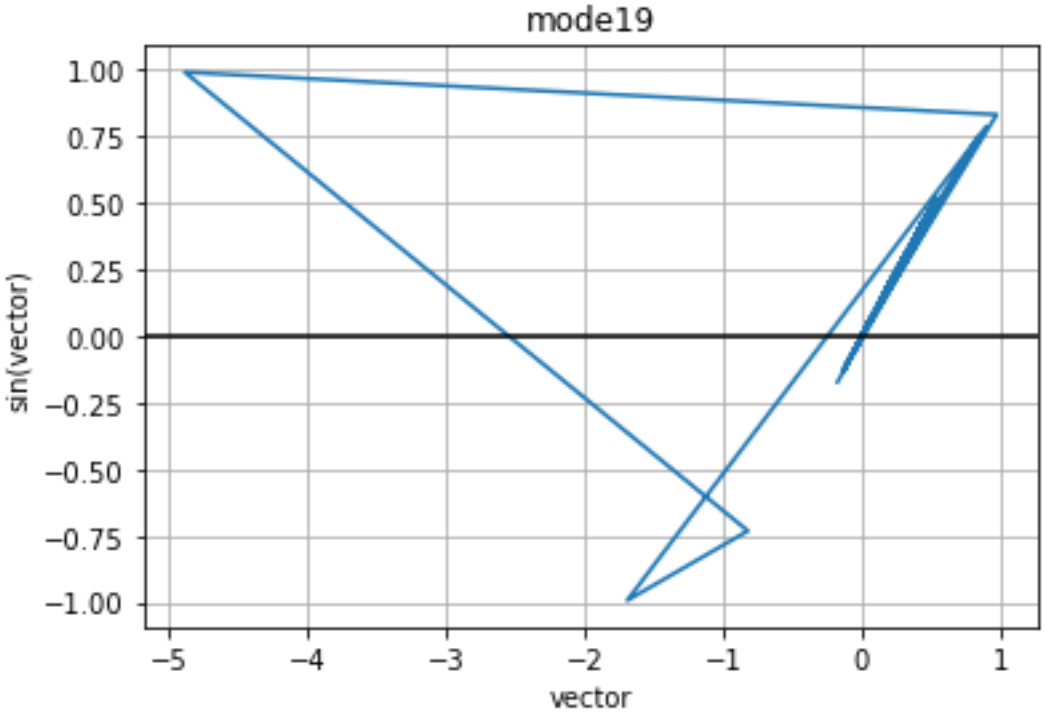

mode24
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−25
−20
−15
−10
−5
0
vector

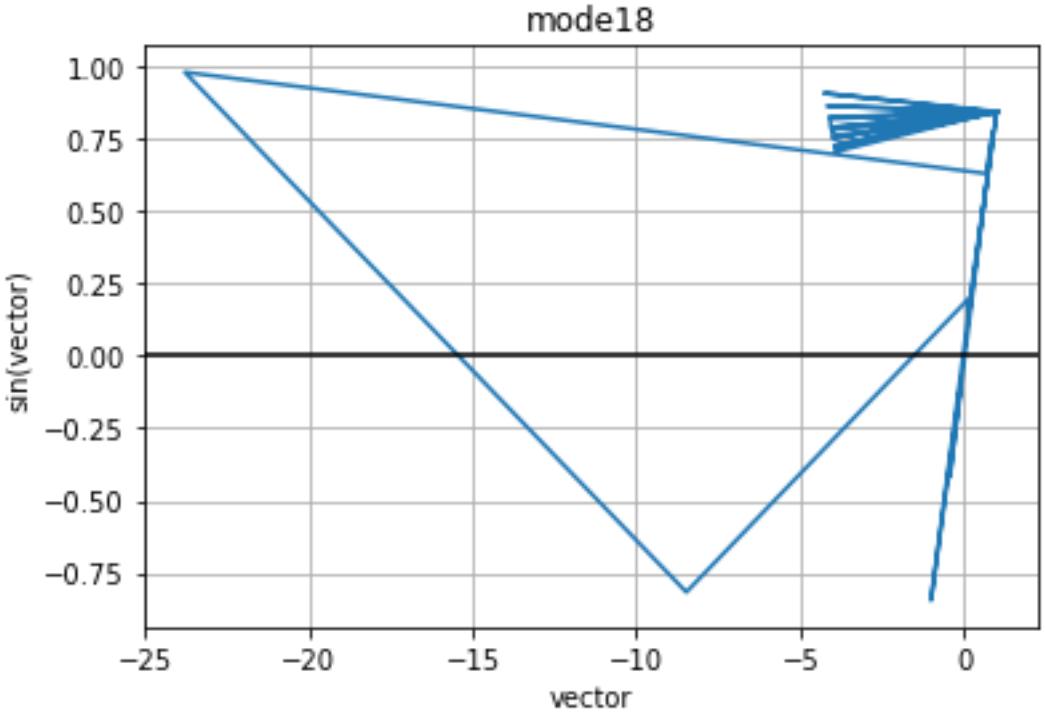

mode23
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−4
−3
−2
−1
0
1
vector

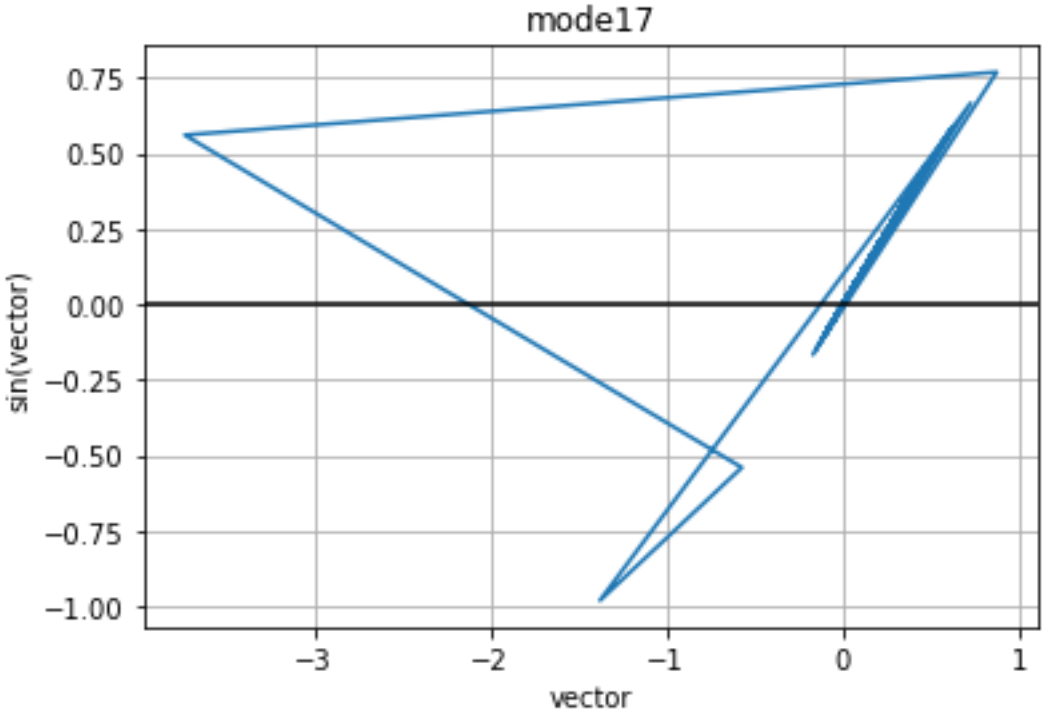

mode22
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
sin(vector)
−25
−20
−15
−10
−5
0
vector

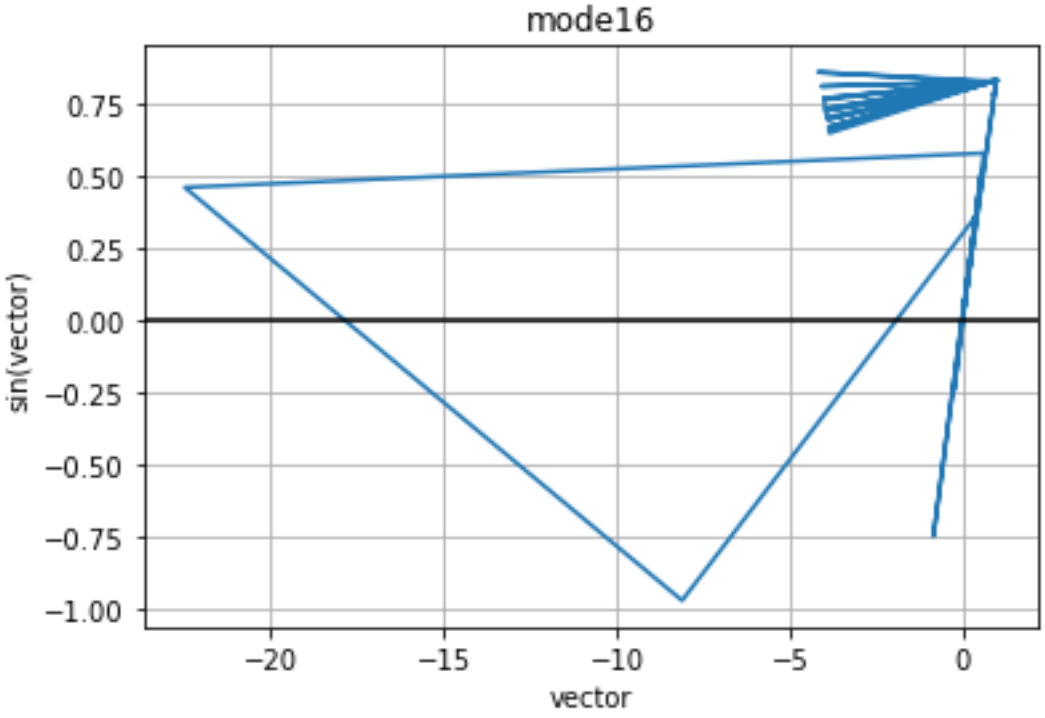

mode21
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−5
−4
−3
−2
−1
0
1
vector

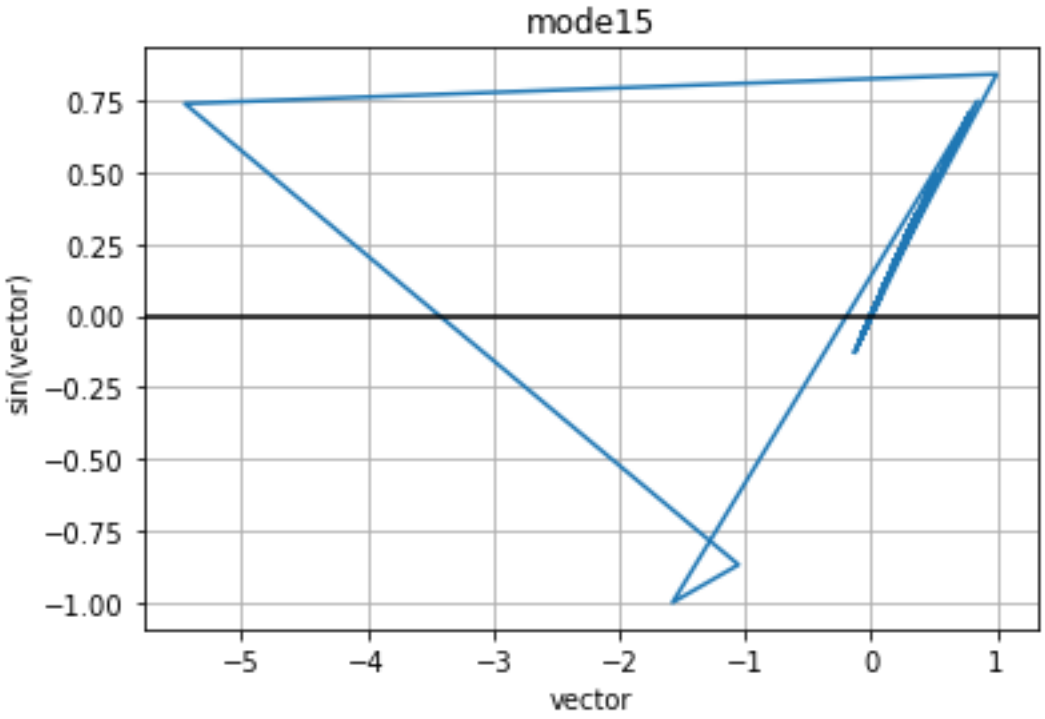

mode20
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
sin(vector)
−25
−20
−15
−10
−5
0
vector

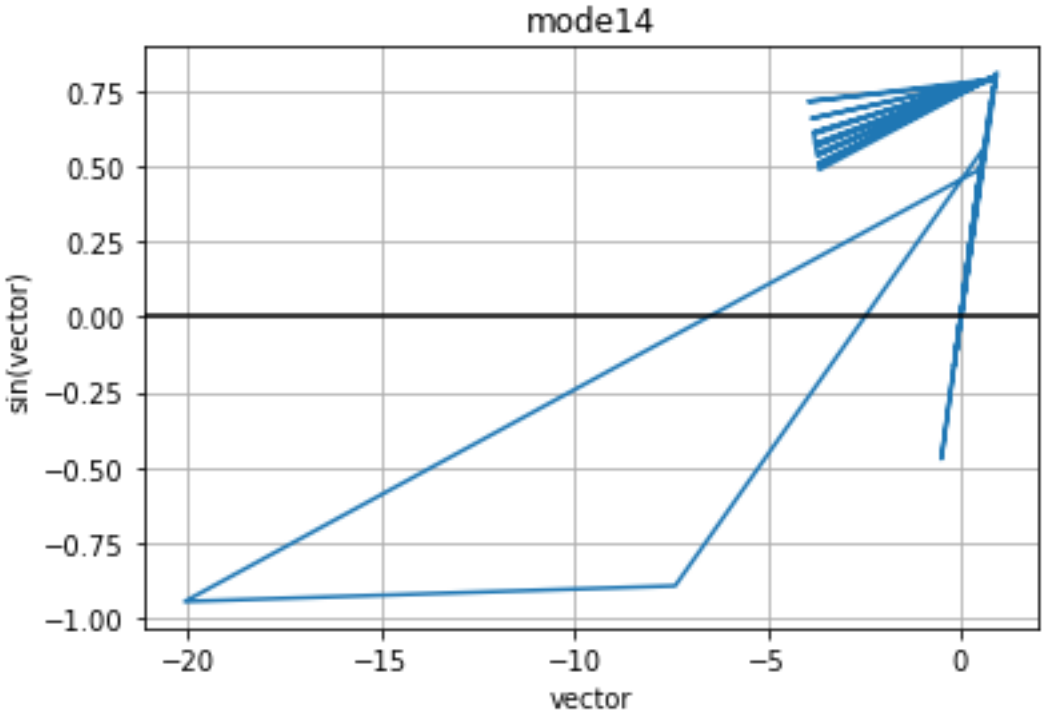

mode19
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−5
−4
−3
−2
−1
0
1
vector

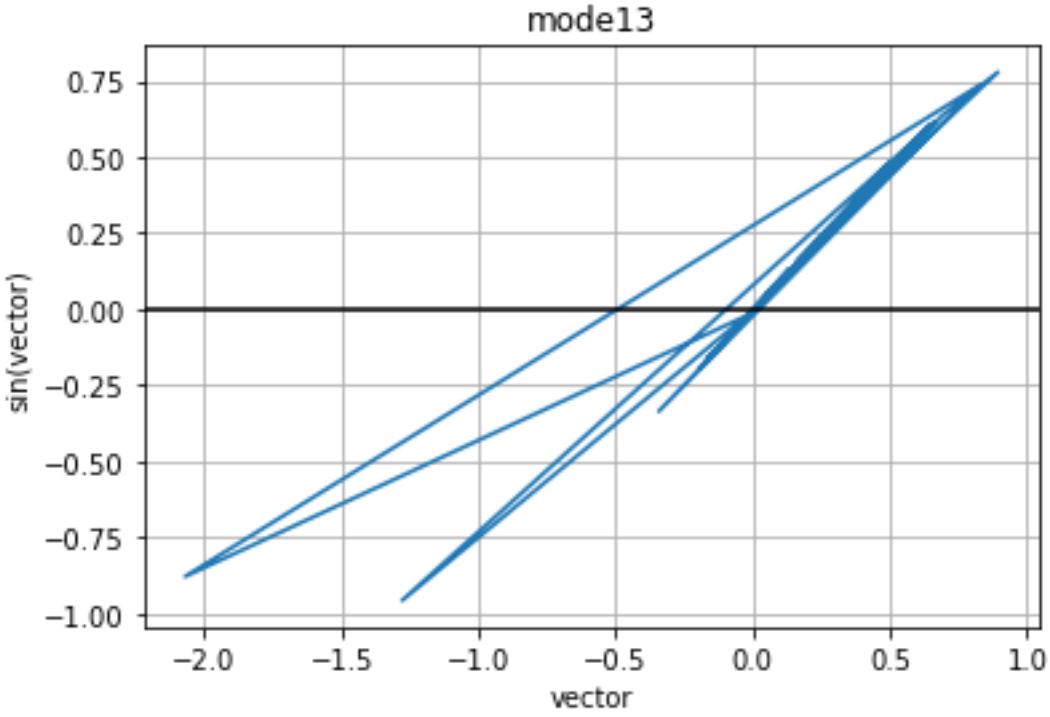

mode18
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
sin(vector)
−25
−20
−15
−10
−5
0
vector

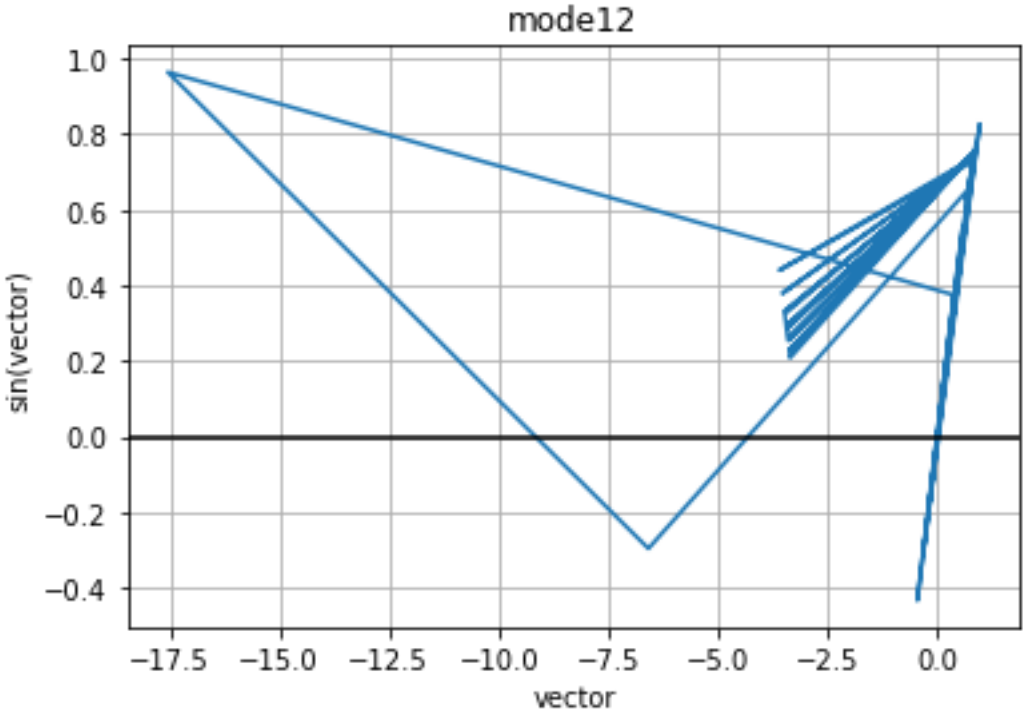

mode17
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−3
−2
−1
0
1
vector

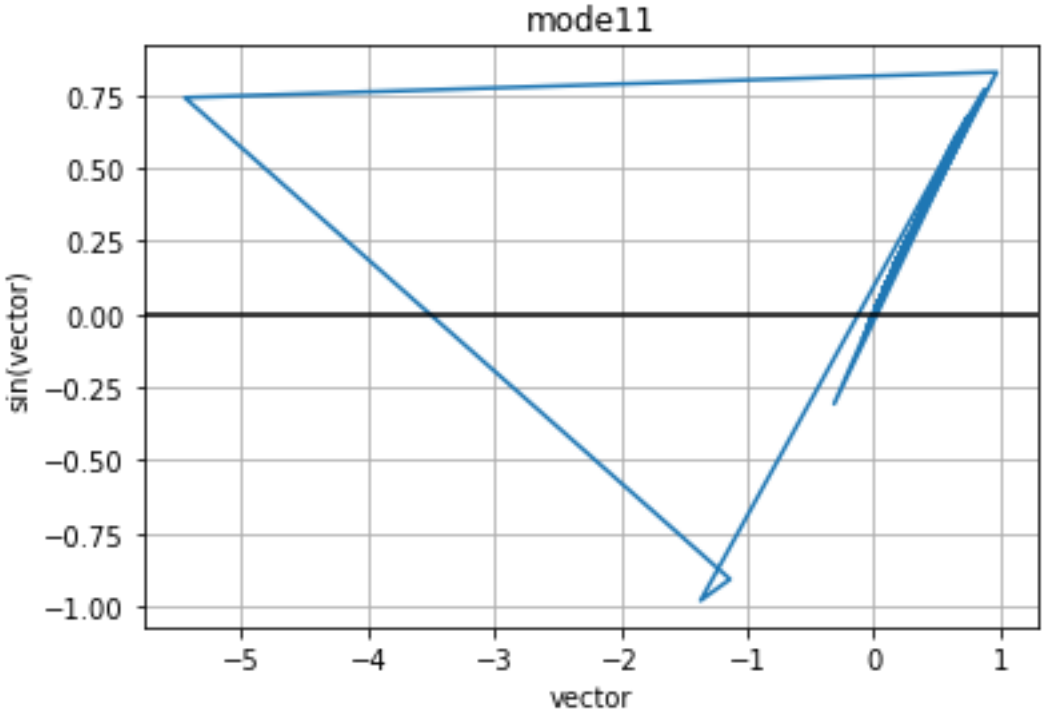

mode16
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−20
−15
−10
−5
0
vector

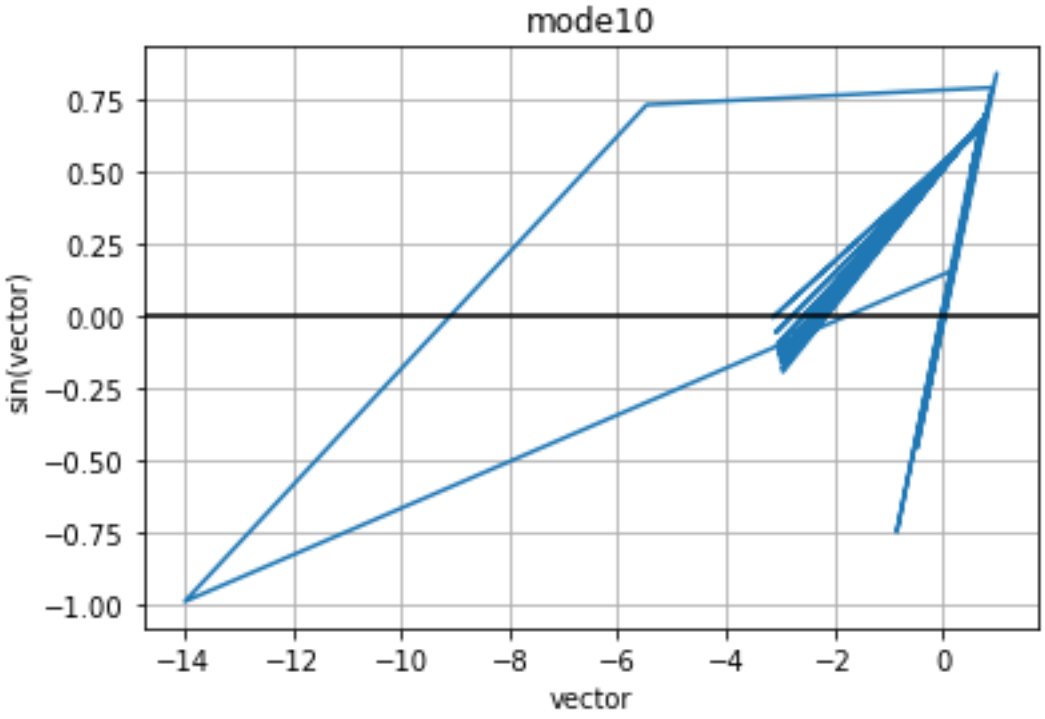

mode15
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−5
−4
−3
−2
−1
0
1
vector

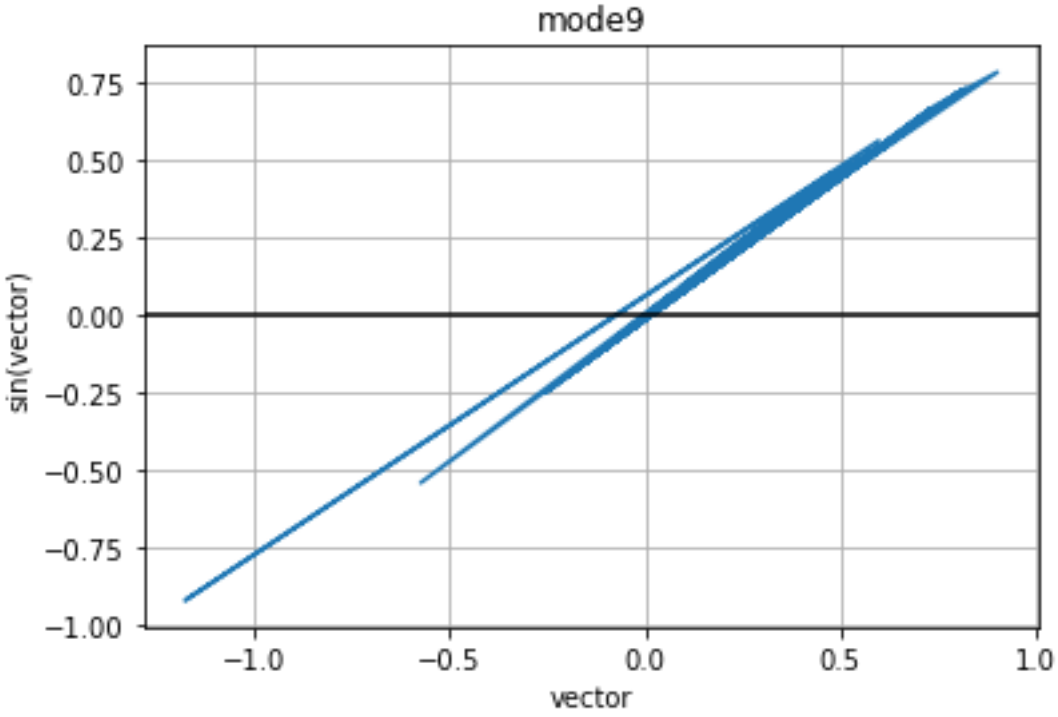

mode14
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−20
−15
−10
−5
0
vector

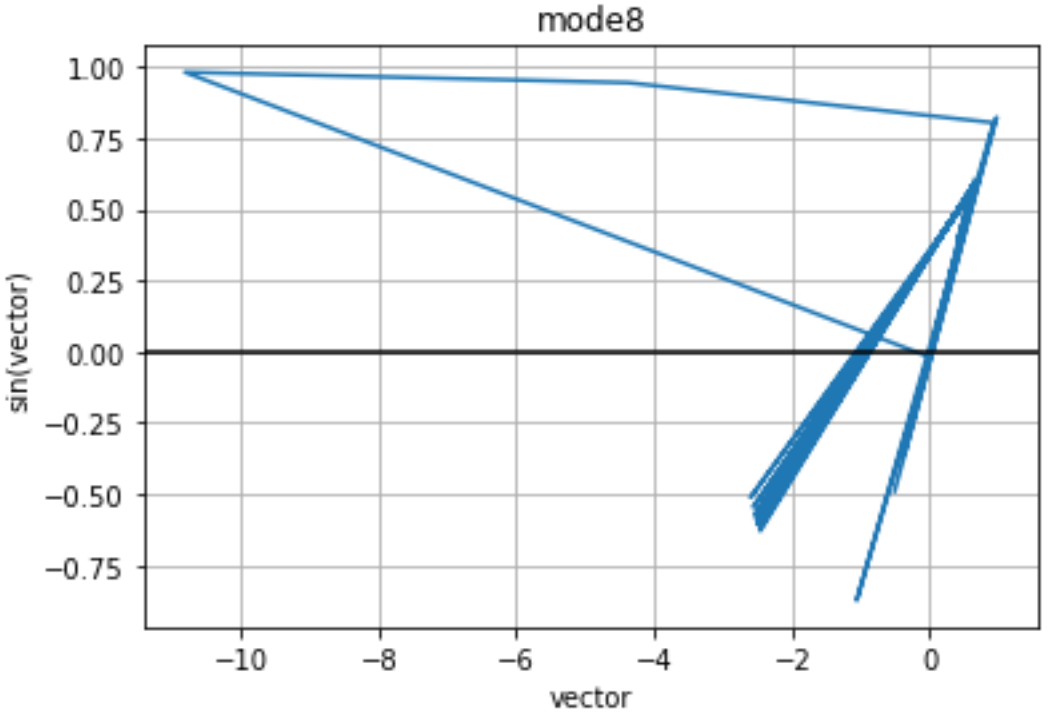

mode13
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
vector

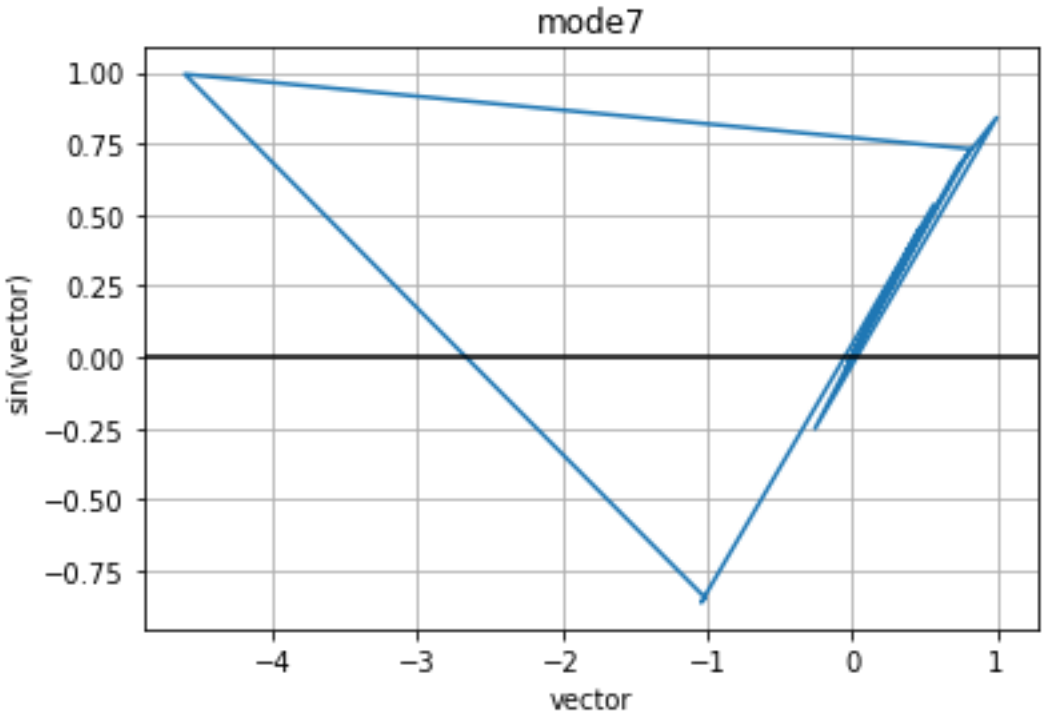

mode12
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
sin(vector)
−17.5
−15.0
−12.5
−10.0
−7.5
−5.0
−2.5
0.0
vector

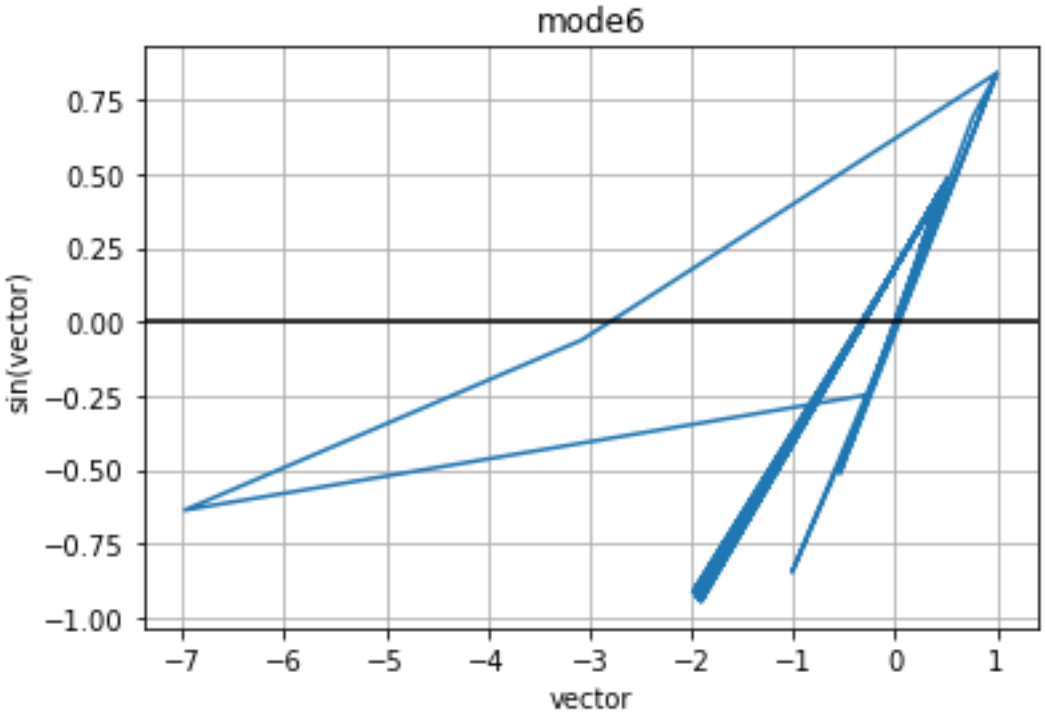

mode11
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−5
−4
−3
−2
−1
0
1
vector

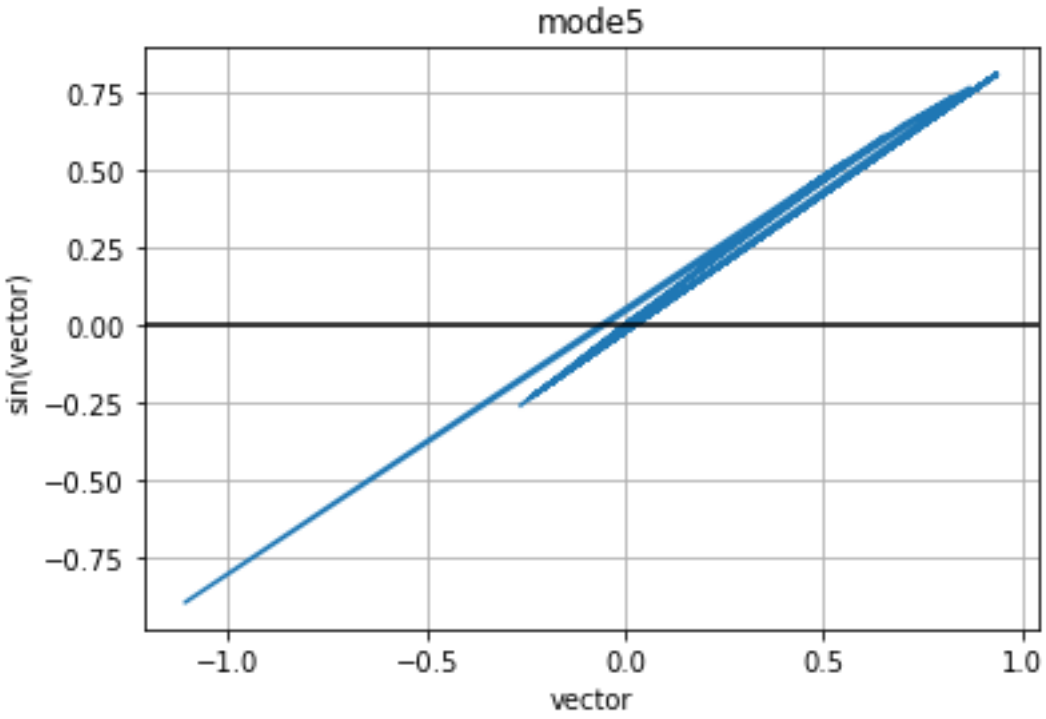

mode10
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−14
−12
−10
−8
−6
−4
−2
0
vector

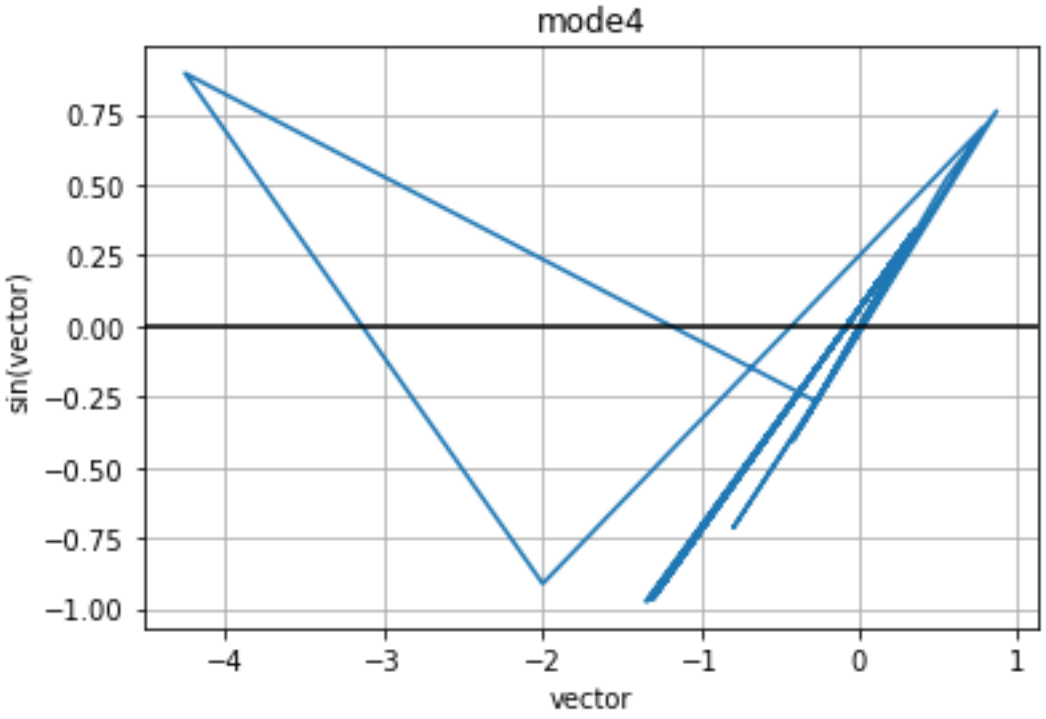

mode9
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−1.0
−0.5
0.0
0.5
1.0
vector

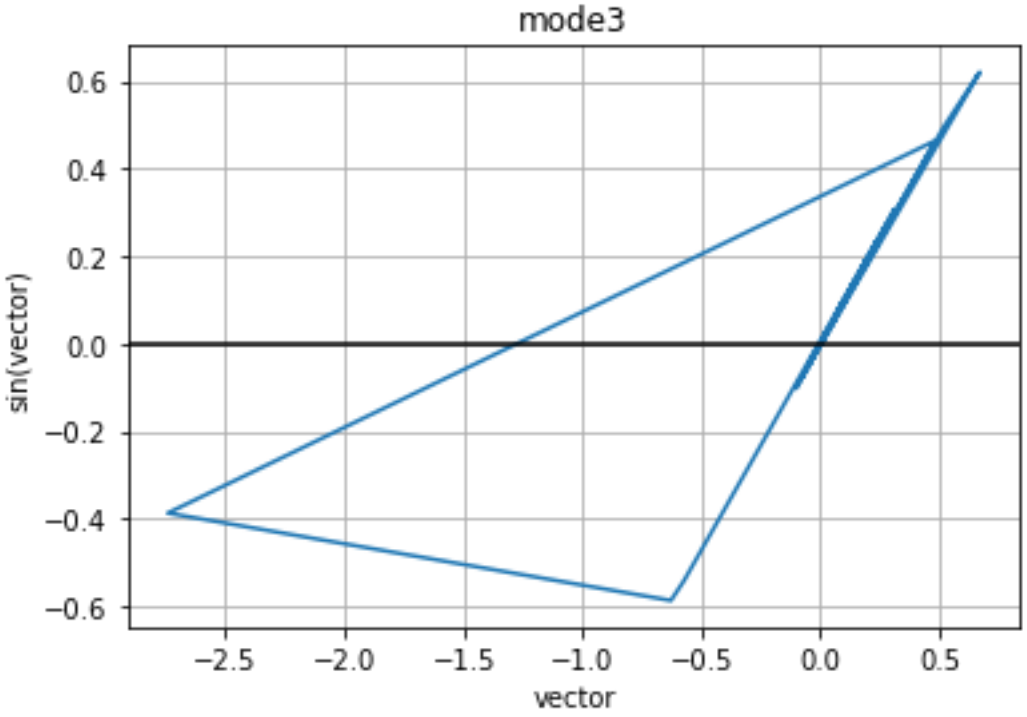

mode8
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
sin(vector)
−10
−8
−6
−4
−2
0
vector

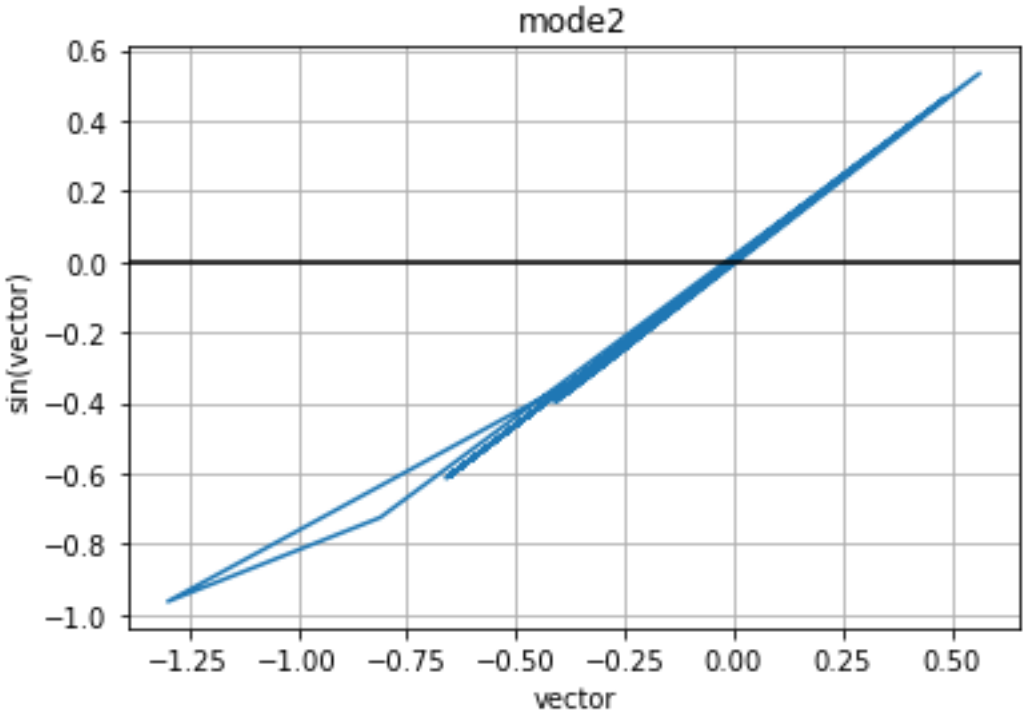

mode7
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
sin(vector)
−4
−3
−2
−1
0
1
vector


mode6
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−7
−6
−5
−4
−3
−2
−1
0
1
vector


mode5
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
sin(vector)
−1.0
−0.5
0.0
0.5
1.0
vector


mode4
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
sin(vector)
−4
−3
−2
−1
0
1
vector


mode3
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
sin(vector)
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
vector


mode2
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
sin(vector)
−1.25
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
vector

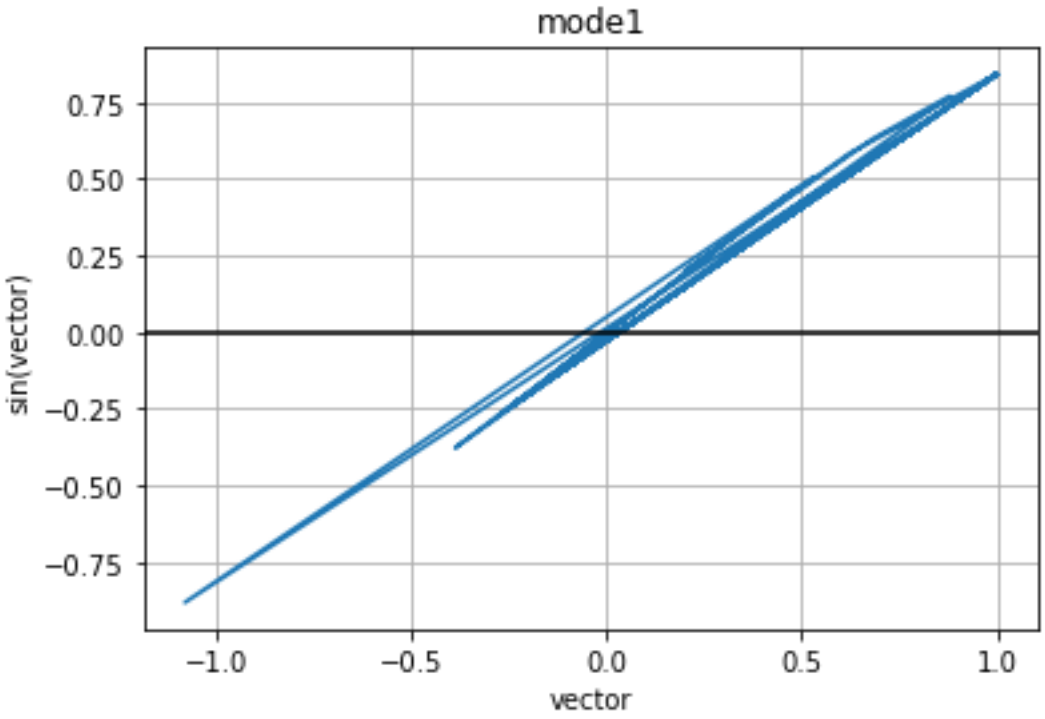


Fig 5.2 Figures of mode shapes

CHAPTER – VI
CONCLUSIONS AND FUTURE SCOPE

6.0 Conclusions

From the present investigation, it can be concluded that the results obtained by running python code were almost matched with the results obtained from ANSYS APDL software. The current project investigated static and vibrational analysis of Railway Bridge structure using Python code and results obtained are exceptional.

The python code which we developed is suitable for all truss structures with any material. Our code is restricted to linear properties only.The code which we developed is used for automation for conducting stress analysis and vibration analysis .

If for different materials this same experiment has to run, then the entire pre-processing process has to be changed and re-run the analysis. If in another case the truss structure was different or number of elements used are differnet, then the entire modeling work and then the pre & post works has to be carried out newly while using ANSYS APDL Software. With this Python code, the mentioned challenges can be solved very easily and at rocket speeds.

6.1 Future Scope

This Python code developed for any type of truss structure. In future it can be upgraded to work for all structural members like beams plates columns etc Also same procedure can be followed to develop python code that can work for plates with notches. Also we can implement in composite materials also.

CHAPTER VII

REFERENCE

[1] An Object-Oriented class design for the Generalized Finite Element Method programmingDorivalPiedadeNeto* ManoelDênis Costa FerreiraSergio PersivalBaronciniProença, latin American journal of solids and structures, 10(2013) 1267 – 1291

[2] Alves Filho, J. S. R., Devloo, P. R. B. (1991). Object Oriented programming in Scientific computations: the beginning of a new era. Engineering Computations, Vol. 8, Issue 1, pp. 81-87.

[3] Bordas, S. P. A., Nguyens, P. V., Dunant, C., Guidoum, A., Nguens-Dand, H. (2007). An extended finite element library, International Journal for Numerical Methods in Engineering, Vol. 71, pp. 703-732

[4] Duarte, C. A. and Oden, J. T. (1996b).Hpclouds – an hpmeshless method. Numerical Methods for Partial Differential Equations, Vol. 12, pp. 673–705.

[5] scikit-fem: A Python package for finite element assembly
Tom Gustafsson1 **and G. D. McBainDOI:** 10.21105/joss.02369

[6] Cimrman, R., Lukeš, V., & Rohan, E. (2019). Multiscale finite element calculations in Python using SfePy. *Advances in Computational Mathematics*. doi:10.1007/s10444-019 09666-0

[7] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &*
Engineering, *9*(3), 90–95. doi:10.1109/MCSE.2007.55

[8] Bathe, K. J. (1996). Finite Element Procedures, Prentice-Hall, Inc. Englewood Cliffs, New Jersey.http://matplotlib.sf.net/Matplotlib.pdf, (acessed April 2011).