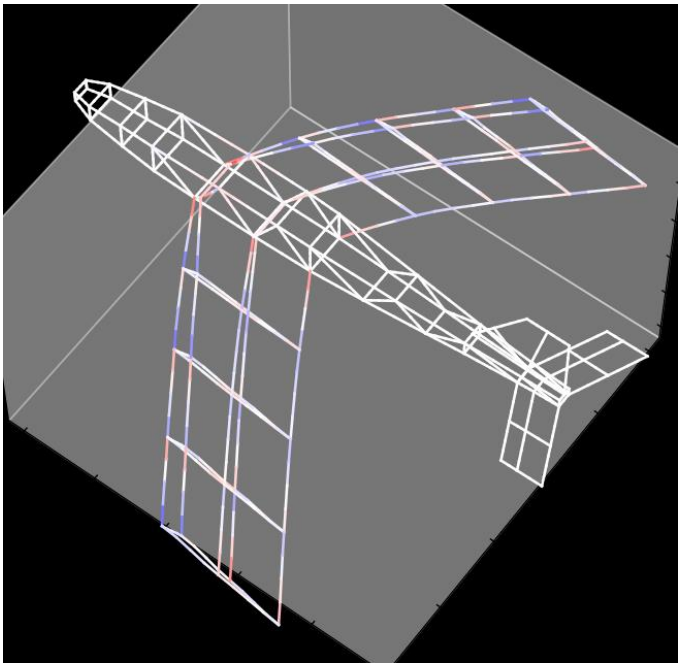
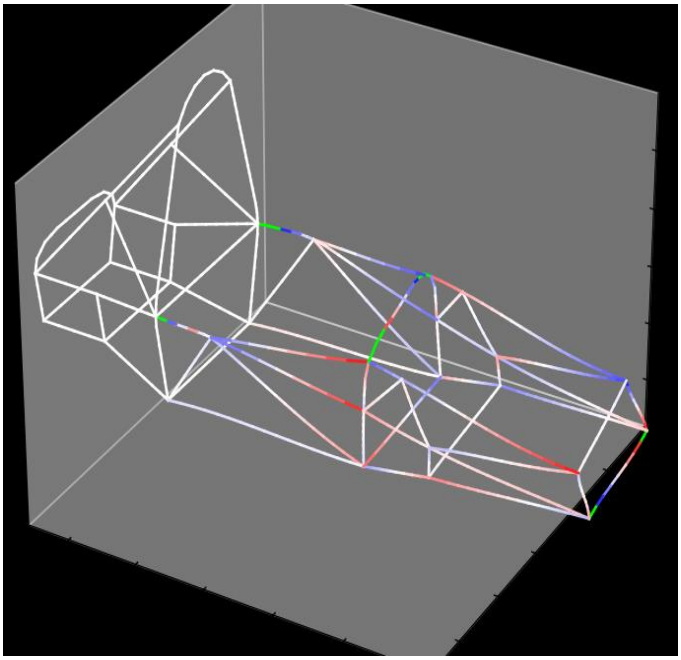




Introduction

To

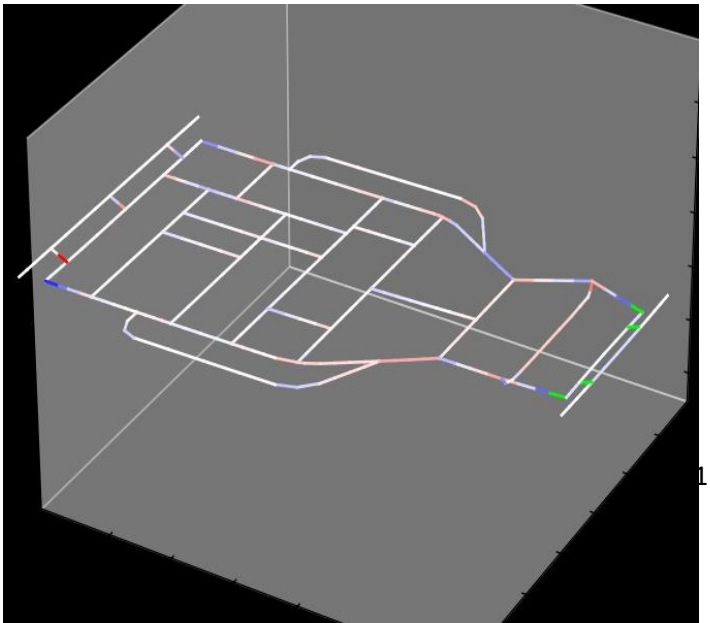
FrameFEA:

A complete finite element solver

for frame structures in Blender

Created By:

Karthikeyan. C



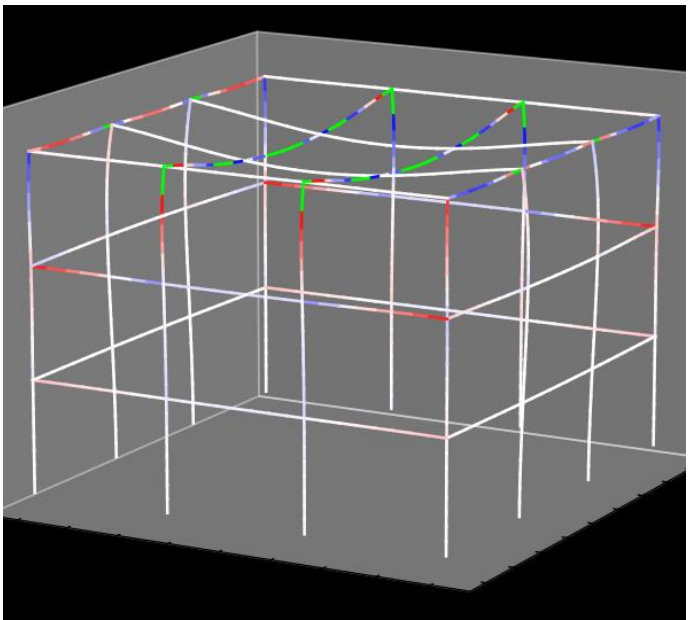


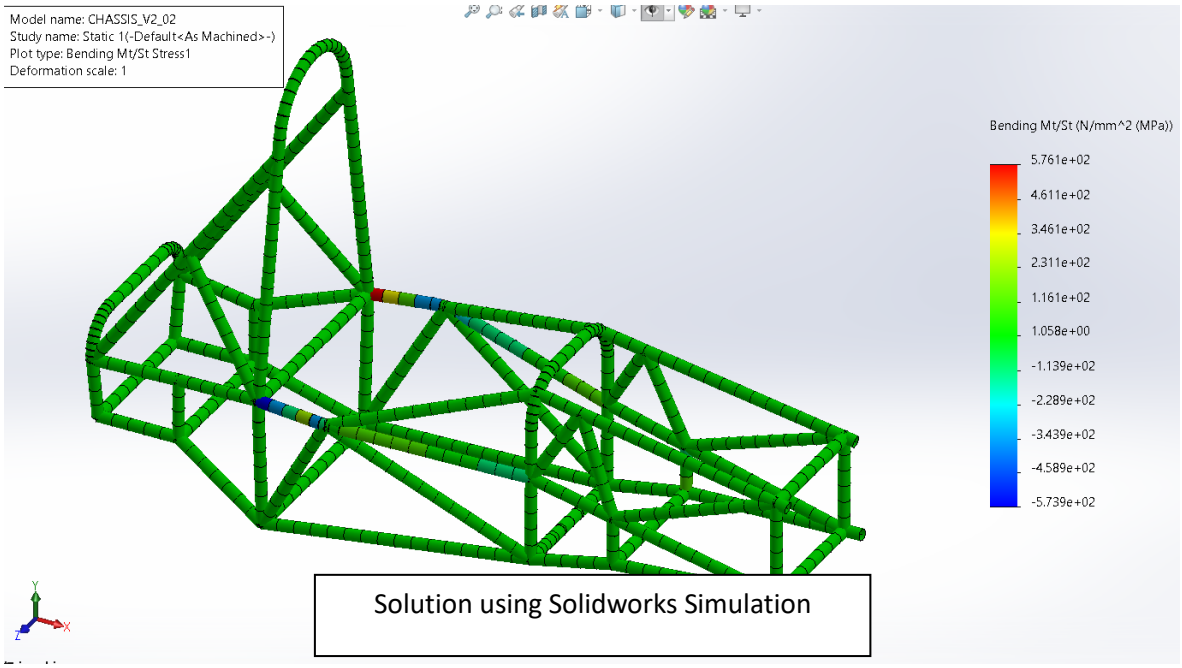
Table of Contents

CONTENTS

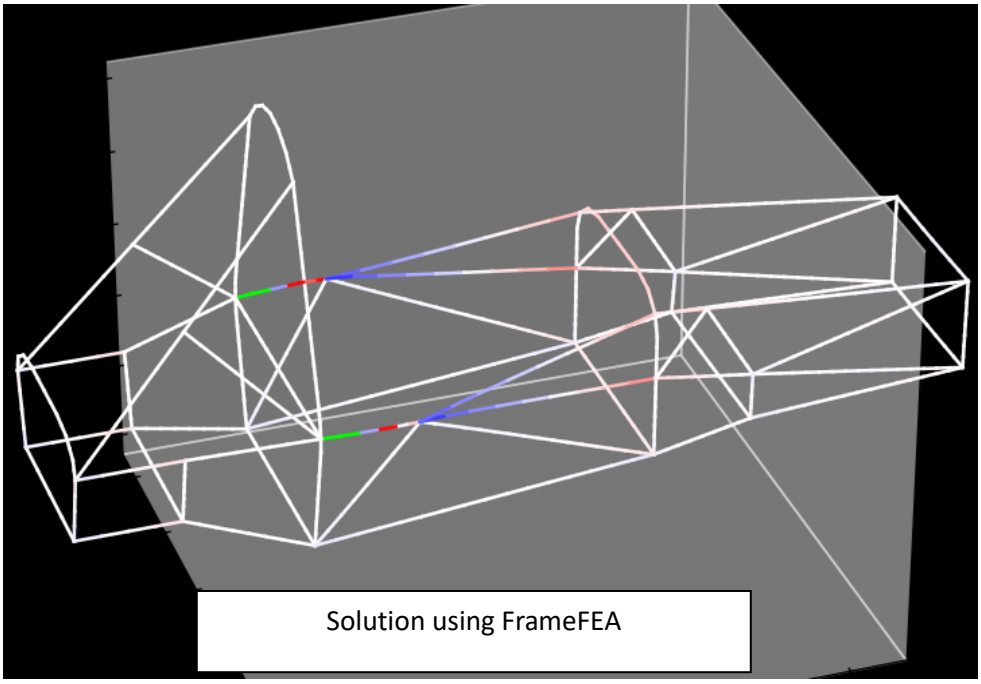
Introduction to FrameFEA

FrameFEA is a finite element solver for naturally discretised and shear deformable frame structures (as dictated by the Timoshenko beam theory). It promises a seamless experience between **modelling, meshing and stress analysis** on the modelled structure all **integrated into a single module**. My college FSAE team **Team Precisio** uses FrameFEA as their primary tool for rapid frame design iterations, providing a hassle-free and user-friendly experience

It allows users to model the structure, mesh the structure and refine it, and apply boundary conditions and forces in Blender itself. Blender has no implicit functionality for implementing FEA but has amazing modelling capabilities. The deformation and color plots are shown through 3D Matplotlib. To visualise the stresses, the color plotter function uses the nodal displacements and the transformation matrix corresponding to the deformed state to plot the stresses local to each element. The stresses were in good agreement with Solidworks simulations.



Solution using Solidworks Simulation



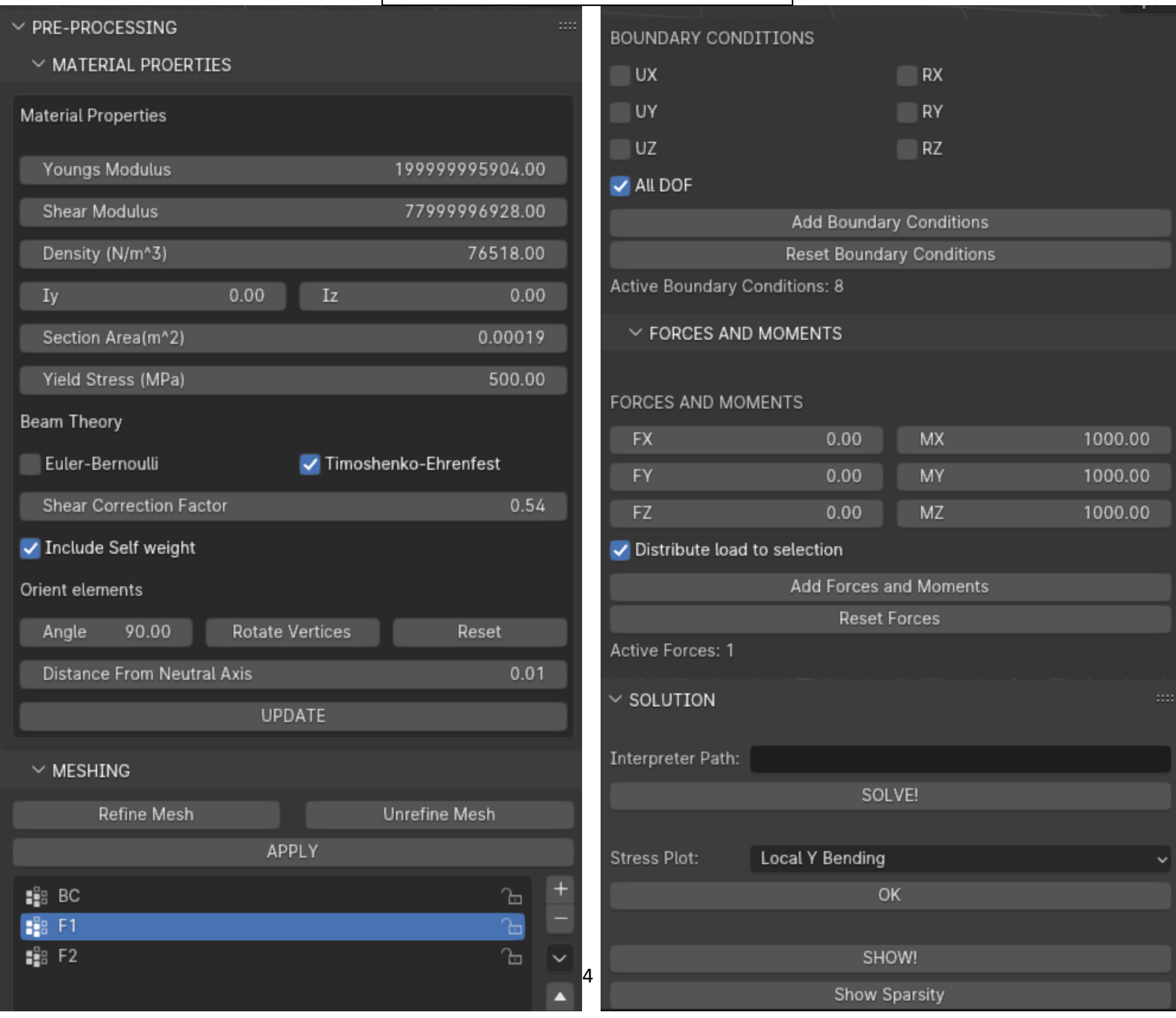
Solution using FrameFEA

Capabilities

The user can switch between regular Euler-Bernoulli and Timoshenko beam theory assumptions. They can also locally rotate any beam along its local axis to any specified angle. The UI is designed in an intuitive and user-friendly way. Users can pre-select the nodes and store them. This is useful when complex loading conditions are encountered. Execution by a specific interpreter in your system is also possible but requires the necessary packages installed beforehand.

Sparsity plot for the stiffness matrix is also available for the curious. Different types of stresses can be viewed after solution

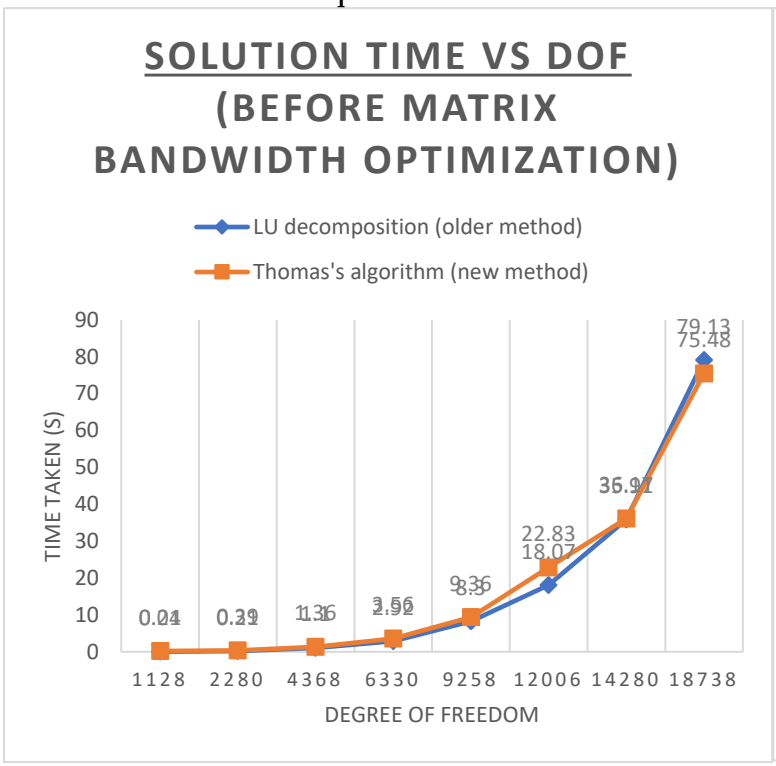
Some of the UI elements implemented in FrameFEA

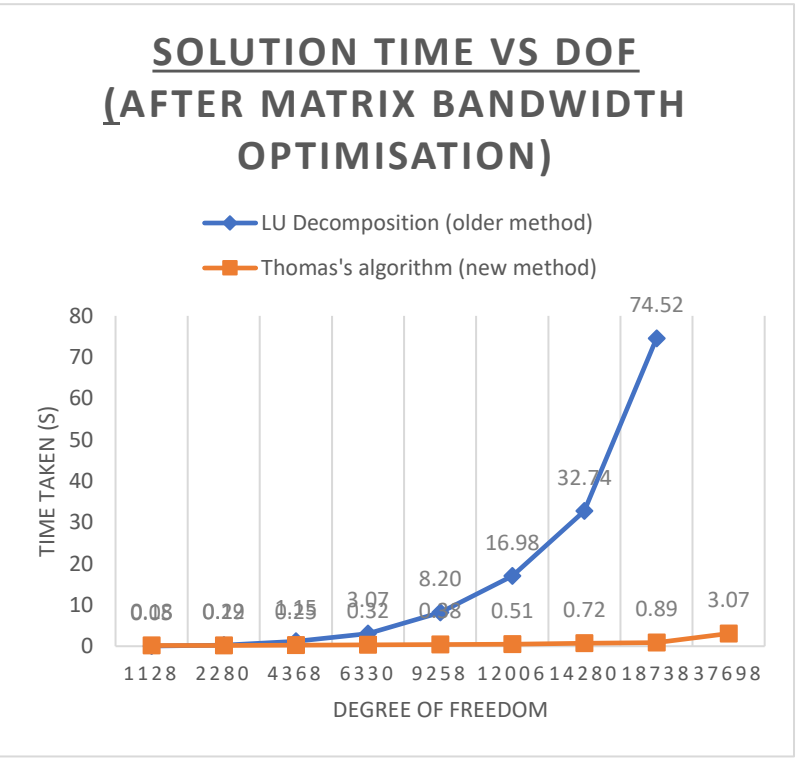


Optimization

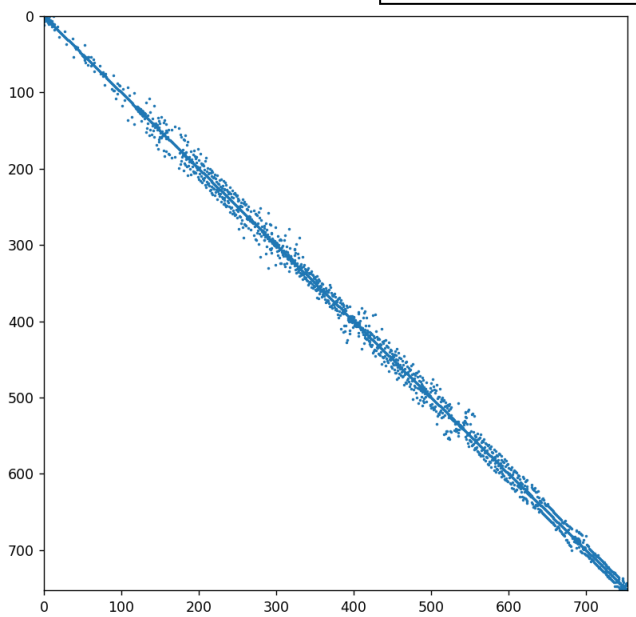
Naturally discretised systems have unusually high sparsity due to less nodal interconnectivity (exceeded 99% in most cases). Hence, solutions using standard Gauss elimination and its likes such as LU decomposition **were highly inefficient** as they mostly processed through zeros.

I found that banded matrix solutions significantly improved processing time during my research. Hence, I used the positive definite and symmetric nature of well-conditioned stiffness matrices to use Thomas's algorithm through functions from the **SciPy** Python module. This required a banded structure to be of actual use.

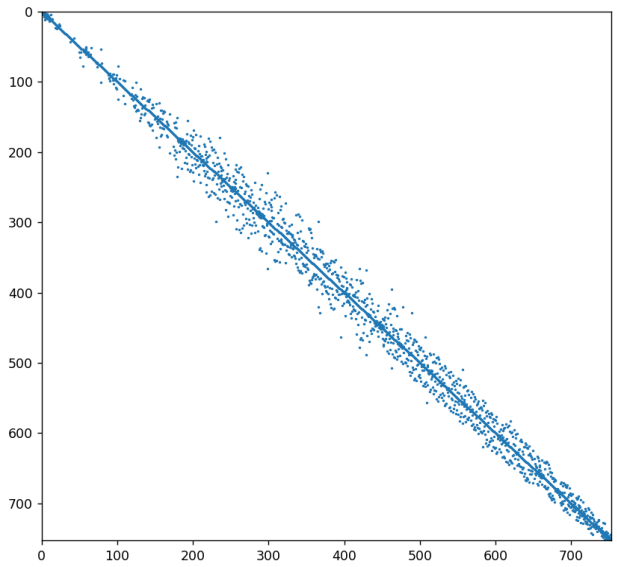




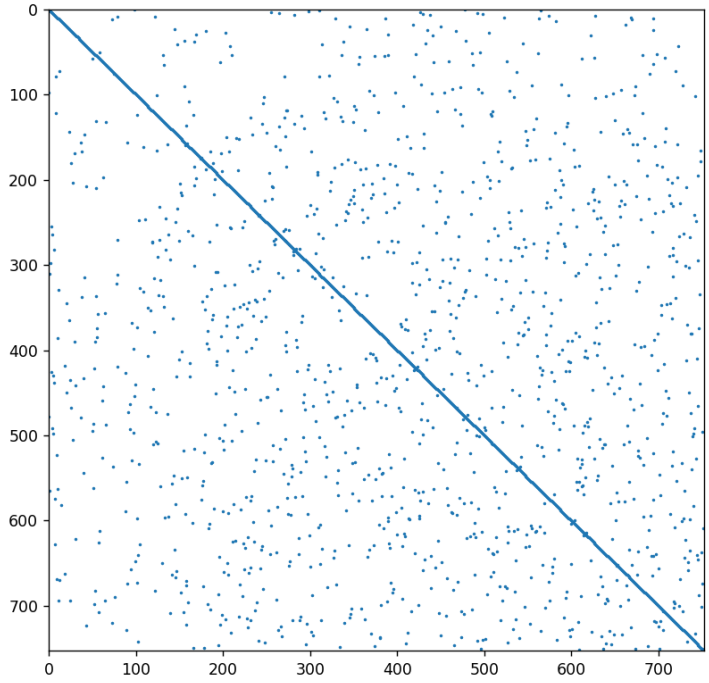
Stiffness matrix sparsity plot for different node-numbering schemes (using matplotlib scatter plots)



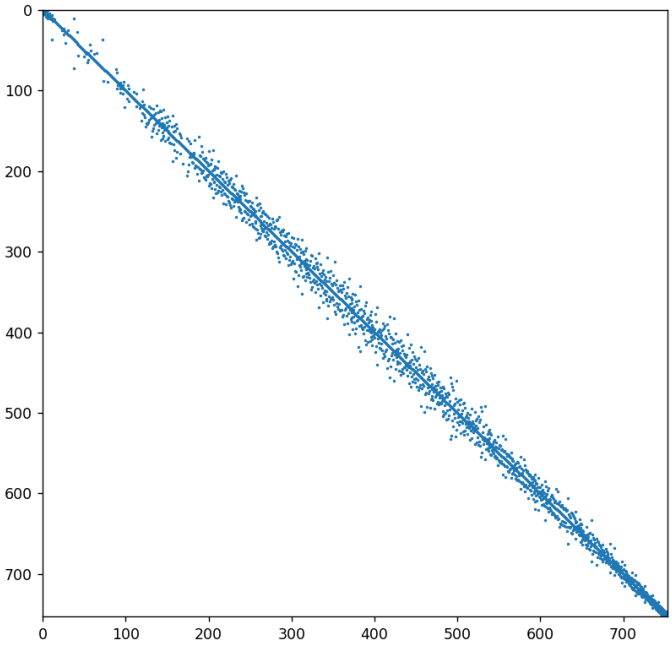
ALONG GLOBAL Y AXIS (**MOST OPTIMAL**)



ABOUT A FIXED POINT (CURSOR DISTANCE)



RANDOM ORDER (LEAST OPTIMAL)



ALONG GLOBAL Z AXIS

Lesser bandwidth meant faster computation time. I soon learnt that a sequential node-numbering scheme provided minimised bandwidth. Out of many such schemes, 3 were selected and their performance was plotted for various meshes. I learnt that selecting the most optimal node-numbering scheme depends on the mesh orientation, and nodal interconnectivity mainly.

Hence, the solver now finds the scheme providing the least bandwidth for each mesh **autonomously,** ultimately optimizing solution time. **RANDOM ORDER** method is ignored by default as it gave the worst results by far.

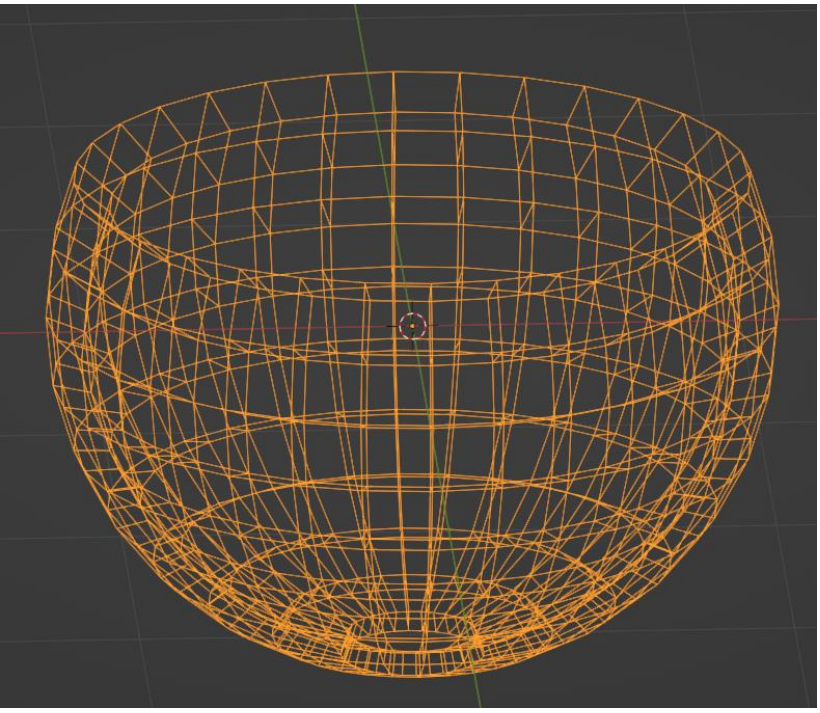
```
Node numbering Methods:  ['VIEW_ZAXIS', 'VIEW_XAXIS', 'CURSOR_DISTANCE']
Possible Bandwidths:  [306, 192, 450]
Method for Minimum Bandwidth:  VIEW_XAXIS :192

Solution time only Thomas: 0.059387922286987305
Solution time only LU: 0.7787473201751709
```

FrameFEA sorting through the optimal node numbering method

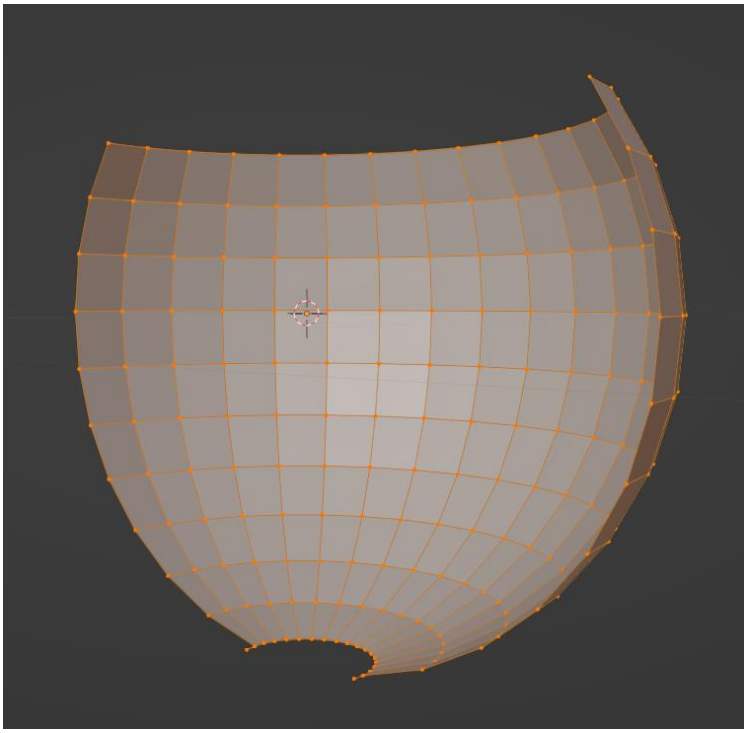
Future Scope

.

- Extension to support 4-noded isoparametric shell elements (currently being worked upon).
- Extension to support for hexahedral solid elements.
- Blender has an extremely limited scope for solid meshing. However limited support using hexahedral elements for shell structures is also currently being studied.
- Blender can be a blazingly fast preprocessor for these types of elements and can be a good alternative to commercial ones.



Shell mesh made of 8 node hexahedral elements



Shell mesh made of 4 node shell element

Access

- FrameFEA is available as an open-source Blender add-on and can be accessed through my Github repository: https://github.com/Karth1kn/GIT_FILES/tree/main.

- For technical capability demonstration: https://youtu.be/0kan9xtWoqY