Linear Elasticity Finite Element Simulation Web Application

This project is a web application for running finite element method (FEM) simulations. It consists of a React frontend and a Python backend. The frontend allows users to upload CAD data, monitor simulation progress, and view the results, while the backend handles the simulation process.

Author:

Pavan Kulkarni
10/09/2024
Bengaluru, India

Preface

Dear Srinivas,

I want to extend my heartfelt thanks for the opportunity to work on this project and for your trust in my abilities. It has been a truly rewarding experience to contribute to such an exciting FEM simulation onto a Web platform. Your support, guidance, and insights have been invaluable throughout the process.

I am eager to see how this project continues to grow, and I look forward to any future collaboration. Thank you once again for allowing me to be a part of this incredible initiative.

Warm regards,
Pavan Kulkarni

Table of Contents

Project Structure

```
root
├── server/
│   ├── app.py                 # Python Flask backend
│   ├── <simulation_logic>.py   # Simulation logic (replace <simulation_logic>
│   ├── results/               # Directory for storing results
│   ├── static/                # Static files for backend
│   └── uploads/               # CAD file uploads
├── client/
│   ├── public/                # Public directory for React
│   ├── src/
│   │   ├── components/        # React components
│   │   ├── App.js             # Main React app
│   │   └── index.js           # React entry point
│   └── package.json           # Frontend dependencies
├── README.md                  # Project documentation
└── requirements.txt           # Backend dependencies
```

Requirements

- React
- Python
- Linux/Mac

Installation

- **Frontend(client) (React)**

Create project in the root by

    npx create-react-app client
    cd client

Install dependencies using npm:

    npm install

- **Backend(server) (Python)**

Navigate to the server/ directory:

```
cd server/
```

Create a virtual environment or Conda environment:

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

or

conda create -n env1 python=3.9
conda activate env
```

Install backend dependencies:

```
pip install -r requirements.txt
```

Running the Application

Step 1: Run the Backend (Python Flask)

Navigate to the backend/ directory:

```
bash
cd server/
```

Start the Flask server:

```
Bash
conda activate env
flask run

or
conda activate env
python app.pyc
```

By default, the backend will be available at http://127.0.0.1:5000/.

Step 2: Run the Frontend (React)

Open a new terminal and navigate to the frontend/ directory:

```
bash
cd client/
```

Start the React development server:

```
bash
npm start
```

The frontend will run on http://localhost:3000/.

Step 3: Access the Application

Once both servers are running, you can access the application by visiting http://localhost:3000/ in your browser.

Project Complete Implementation Description

1. **Open-Source FEA Library**

- **Libraries:** FEniCS FEA library is used. This is powerful and have a solid open-source community. They support linear elastic simulations, and can be integrated into a larger workflow.

2. **Backend for Computation**

- **Server Setup:** I have setup a backend server where the FEA calculations are performed. This is done using a Python-based web framework **Flask**. The server will handle importing CAD files, setting up the simulation, and running the FEA calculations.

3. **Import the 3D CAD Model**

- **File Import:** The user is able to upload CAD files in **STEP** format.

4. **Material Parameters**

- **Material:** Steel is considered as default with Young's modulus 210 GPa and Poisson's Ratio 0.3

5. **Mesh Generation**

- **Mesh Generation:** The CAD model is meshed using **Gmsh** before analysis.100 elements on x direction and 10 elements along y direction. Linear tetrahedral (solid) element is a three-dimensional finite element are chosen.

6. **Load and Boundary Condition Specification**

- **Dirichlet BC :** Fixed for Rotation and Translation on all directions (Encastre) condition is used on the left boundary.
- **Force :** Taken as user input which is applied on the right side of the Beam in Newtons. **1000N**(default)

7. **Run the FEA Simulation**

- Once the model is prepared (imported, meshed, and boundary conditions are set), the linear elastic simulation runs using the FEniCS FEA library.
- The backend server will execute the simulation and collect the results.

8. **Display the Results**

- **Post-Processing:** Displacement and Stress tensors are computed internally and saved in the output files **".xdmf"** file in the app directory
- **Result Visualization:** The results are sent to the frontend and displayed in the 3D viewer using Plotly library.

9. **Web Integration**

- **File Upload:** The React frontend sends the CAD file to the backend via the endpoint using a POST request with form data collected
- **Simulation Start:** After the file is uploaded, the form data (including the CAD file path) is sent to the backend via the endpoint Simulation start on button click event.
- **Simulation Progress:** The frontend periodically checks the simulation progress by polling the endpoint.
- **Fetching Results:** When the simulation is complete, the frontend fetches the results from the endpoint.

  This pattern of communication follows the client-server model, where the React frontend (client) sends requests, and the backend (server) processes these requests and responds with the necessary data or actions.

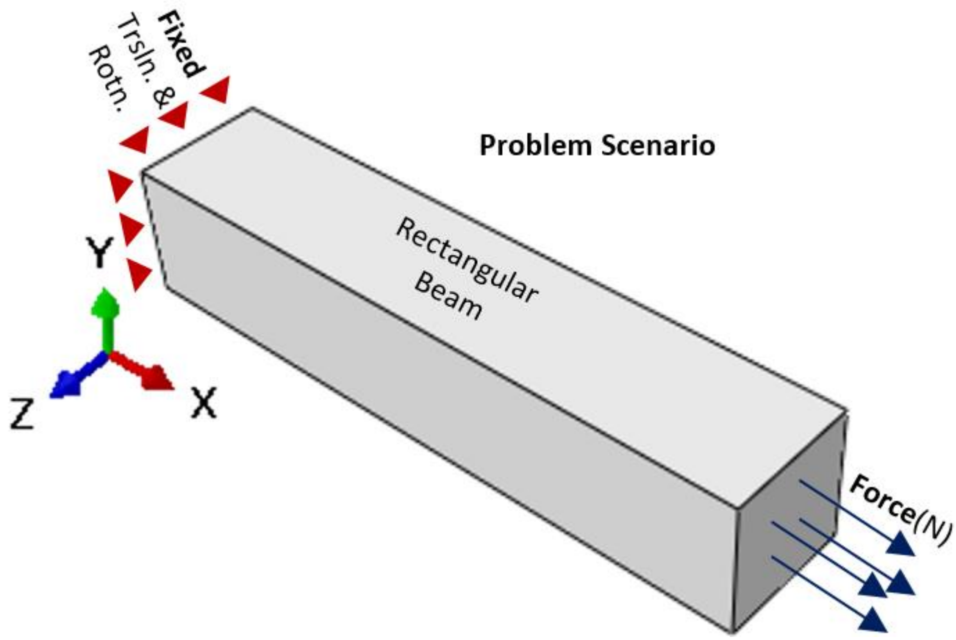
Workflow

1. User uploads a CAD file (STEP format).
2. User inputs the force and material parameters.
3. User clicks the "Run Simulation" button.
4. The simulation runs on the backend.
5. The progress of simulation run is seen on the progress bar
6. The results are computed and visualized on the frontend.
7. Results are viewed separate tabs of the browser.

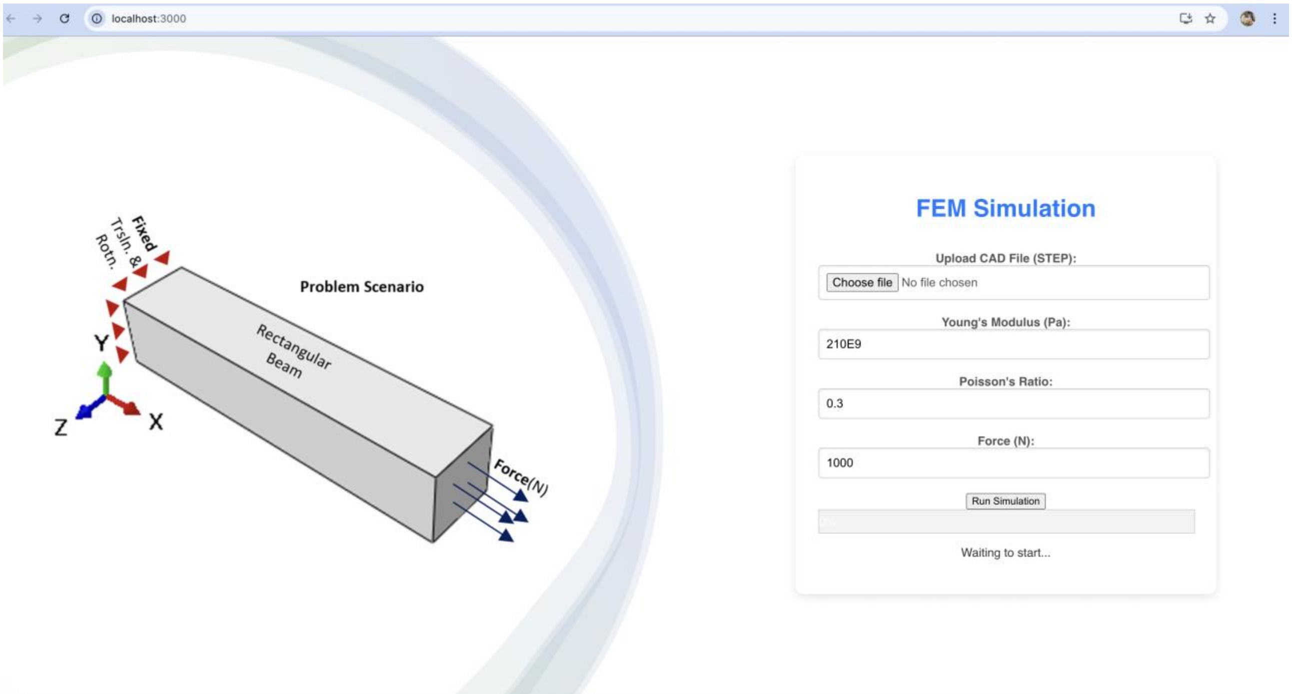
Tech Stack

- **Backend:** Python (/Flask), FEniCs, gmsh
- **Frontend:** React.js, REST API
- **Visualization:** Poltly

Problem Scenario Considered

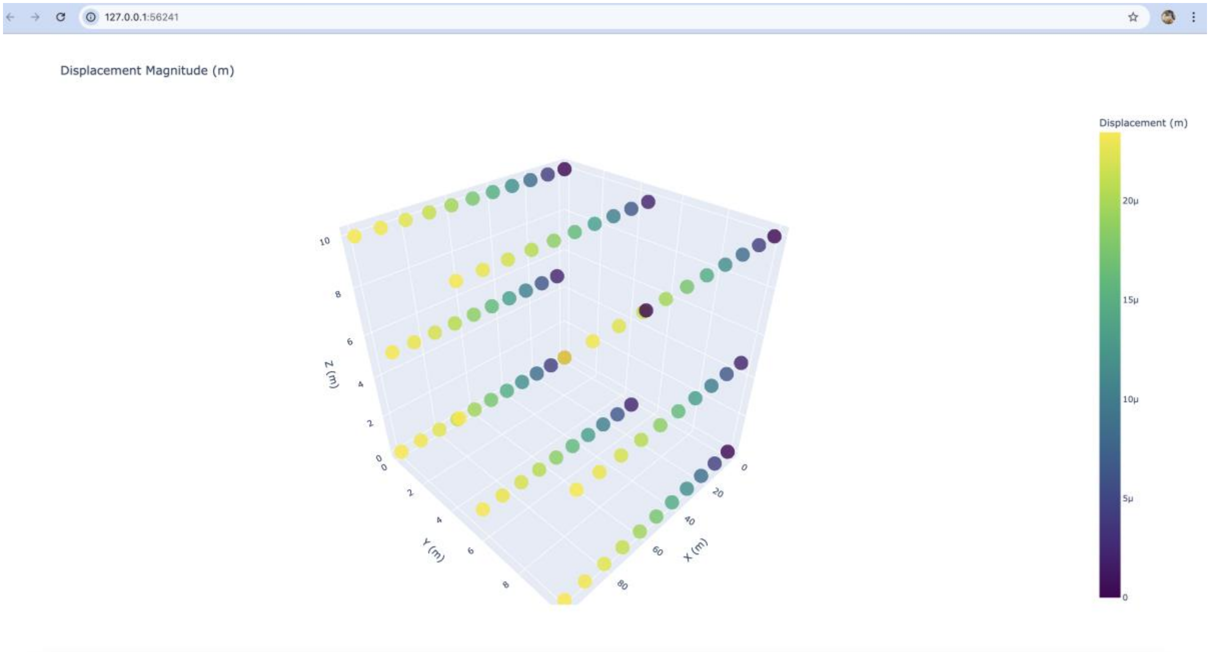


Frontend Interface

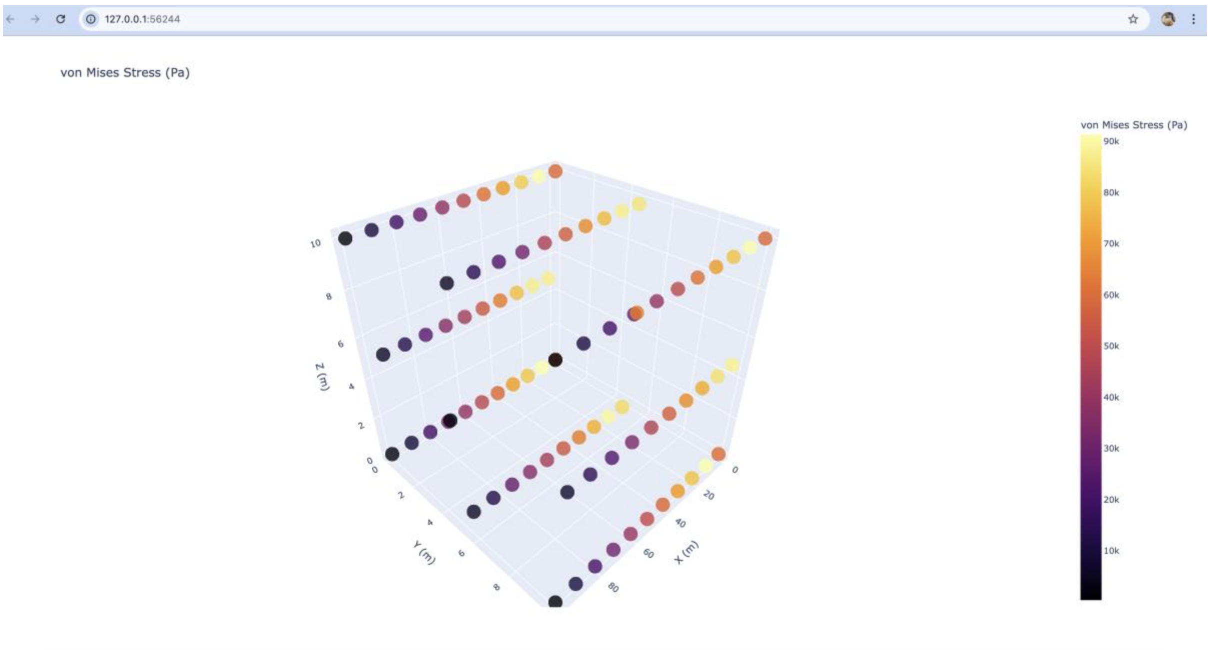


Results

Displacement Plot



Stress Distribution Plot



Future Scope

There is too much to talk about this ☺

1. Enhanced User Interface (UI) and User Experience (UX):

- **Advanced CAD Viewer**: Integrate a 3D viewer for CAD files in the frontend, allowing users to rotate, zoom, and inspect the uploaded model before simulation.
- **Real-time Updates**: Provide real-time visualization of simulation progress, such as intermediate results (e.g., stress distribution updates while the simulation is running).
- **Interactive Simulation Input**: Allow users to define forces, constraints, or boundary conditions directly on the CAD model through the UI.
- **Customizable Mesh Generation**: Allow users to modify meshing parameters (e.g., mesh density) in the UI for more refined simulations.

2. Support for Different Simulation Types:

- **Non-linear Material Models**: Extend the solver to handle non-linear materials such as plasticity or hyperplastic materials, enabling more realistic simulations.
- **Dynamic Analysis**: Add support for dynamic simulations such as vibration analysis, time-dependent loading, or fatigue analysis.
- **Thermal and Fluid-Structure Interactions**: Expand to multi-physics simulations that couple thermal, fluid, and structural analyses to simulate heat transfer, fluid flows, and their effects on the structure.
- **Non-Linear Geometries**: Add support for large deformations and non-linear geometries for more advanced structural analysis.

3. Integration with Cloud and Distributed Computing:

- **Cloud-Based Simulations**: Enable cloud-based FEM simulations by offloading computation to cloud services (e.g., AWS, Azure) to scale large simulations and reduce dependency on local hardware.
- **Parallel Computing**: Implement distributed or parallel FEM solvers using libraries like mpi4py or GPUs for significantly faster computation in large-scale or real-time applications.
- **Simulation as a Service (SaaS)**: Offer a web-based service where users can run FEM simulations without needing local computational resources.

4. Post-Processing and Result Analysis:

- **Enhanced Visualization Tools**: Incorporate more advanced 3D visualization tools using libraries such as PyVista, ParaView, or VTK for deeper analysis of simulation results.
- **Data Export Options**: Allow users to export simulation results in various formats (e.g., VTK, HDF5, Excel, etc.) for post-processing in other tools.
- **Automated Reporting**: Automatically generate detailed reports with visualizations, stress analysis, and deformation statistics once the simulation is complete.

5. Machine Learning and AI Integration:

- **Simulation Optimization**: Use machine learning techniques (e.g., neural networks, genetic algorithms) to optimize FEM simulations by adjusting parameters like meshing, boundary conditions, or material properties.
- **Predictive Simulations**: Train AI models to predict simulation results based on historical data or previous simulation outcomes to reduce computational effort.
- **Surrogate Modelling**: Create surrogate models (reduced-order models) using AI/ML techniques for real-time simulation predictions, particularly useful in iterative design processes.

6. Multi-User Collaboration:

- **Collaboration Platform**: Enable multiple users to work on the same simulation project in real-time, offering shared CAD models, simulations, and results.
- **Version Control**: Implement version control for CAD files and simulations, allowing users to track changes, revert to previous versions, and compare different simulations.

7. Extended File Format Support:

- **Additional CAD Formats**: Expand support for a wider range of CAD file formats (e.g., STL, OBJ, DWG) to allow more flexibility in input.
- **Mesh Import/Export**: Enable importing and exporting of mesh files for users who already have pre-processed meshes from other software.

8. Mobile and Cross-Platform Support:

- **Mobile-Friendly Interface**: Develop a mobile-friendly UI for running, monitoring, and reviewing FEM simulations on tablets or smartphones.

- **Cross-Platform Deployment**: Build the frontend and backend to be deployable on various platforms, including desktops, cloud servers, and mobile apps.

9. Integration with CAD Software and PLM Systems:

- **Direct Integration with CAD Tools**: Offer plugins for popular CAD software like SolidWorks, AutoCAD, or Fusion 360, so users can run simulations directly from their CAD software.
- **Product Lifecycle Management (PLM) Integration**: Integrate with PLM systems for traceability, version control, and simulation data management throughout a product's lifecycle.

10. Educational and Training Tools:

- **Tutorial and Learning Modules**: Integrate built-in tutorials and guided simulations for educational purposes, helping new users or students learn FEM concepts.
- **Simulation Templates**: Provide predefined simulation templates for common analysis types (e.g., beam deflection, stress-strain analysis) to reduce setup time.

11. Commercial and Industry Applications:

- **Industry-Specific Applications**: Customize the simulation platform for specific industries such as aerospace, automotive, civil engineering, and biomechanics.
- **Regulatory Compliance**: Add features for compliance with industry standards (e.g., ISO, ASTM) for structural simulations and material testing.