Lattice Generation Program

Antony Hwang
Timothy Lu

Supervisor

Mark Lee

Abstract

This technical report presents the development of a computer software application, providing new methods for generating lattice meshes. We introduce various lattice generation methods, applying the algorithms and mathematical calculations onto the mesh data set produced by ANSYS. The project aims at developing a software that is able to process existing mesh data files and output the results as a single lattice structure data file. Also, to investigate which of the methods implemented is more desirable for lattice generation. Finally, we make comparisons between each of the methods, evaluate the performance and the set of results given by each method.

Contents

1 Introduction

1.1 Outline

3D printing is a new and powerful tool that has opened up avenues for putting new ideas into practice. With this in mind, lattice structures have the potential to substitute for solid structures as a lighter weight, but still structurally sound substitute. In order to investigate further, there should be some way to easily and automatically create 3D models of lattice structures whether for the purpose of printing or for analysis.

1.2 Aims and Goals

Aims for this project:

- To be able to construct a mesh in ANSYS.
- To learn efficient ways to multiply shapes in three dimensions.
- To learn about programming design patterns.
- To be able to use the gained knowledge to develop a script that can multiply unit meshed shapes into large lattice volumes.

Goals for this project:

- Create 3D unit shapes as meshes in order to multiply them into a lattice structure.
- Develop an algorithm to multiply shapes into a lattice structure.
- Develop a program that writes lattice structures into useful file formats.
- Remove duplicate nodes created in lattice generation.
- Differentiate beams of lattice shapes generated by the program.
- Develop a user-friendly GUI using a web-platform.

1.3 Overview of the Project

The project was carried out with an iterative approach. Weekly meetings were scheduled with supervisor. The meetings allowed project progress to be reviewed and receive feedback from the supervisor. Also, weekly tasks were set, to ensure everything was on track. The goals and tasks set for each week were recorded in a document. During the meetings, demonstrations of the application were given, to monitor the progress of the development.

1.4 Structure of the Report

The structure of the report is divided into 5 chapters and appendix sections. Following the

Chapter 1 Introduction covers the aims and goals of this project.

Chapter 2 Context covers the background research and contains a short description for each tools, software, frameworks and libraries that have been used in the project.

Chapter 3 Requirements and Analysis, includes a detailed list of all the functional and non-functional requirements defined for this project. Furthermore, covers the entity relationships between the program and the mesh data.

Chapter 4 Design and Algorithm Implementation covers the reasoning of the GUI design and contains description of the algorithm implemented for lattice generation process.

Chapter 5 Evaluation and Conclusion summarises the project. The section gives a summary of what has been accomplished at the end and includes detailed evaluation of the results computed by different algorithms. Finally, concludes the whole project with final thoughts.

2 Context

2.1 Sources of Information

2.1.1 NumPy and SciPy Documentation

NumPy documentations provides developers with support for the NumPy library. This document was a critical resource for the project, as it has the complete API/Class listings. Also provides the descriptions of capabilities of each class and supplies example codes.

2.1.2 The New Boston Tutorials

The New Boston is a website and a YouTube channel that provides online learning services. It includes a series of tutorials on HTML and PHP web application development framework. The tutorials are simple screencasts with voiceover explanation. These tutorials are useful for beginners, as it walked through the fundamental steps that were required to implement a web application using HTML and PHP.

2.1.3 Stack Overflow

Stack Overflow is an online community, provides an online platform for developers to discuss and share computing knowledge across the internet.

2.2 Supporting Tools, Software, Libraries and Frameworks

2.2.1 ANSYS

ANSYS is a software tool for structural analysis. It's Mechanical Ansys Product Launcher (APDL) Workbench acts as a CAD software with many features including meshing and CAD file creation.

2.2.2 Gmsh

Gmsh is a 3D mesh generator software with built-in CAD engine and post-processor. It aims to provide a fast meshing tool with parametric input and visualisation capabilities. Also, supports various 3D model file formats, including .msh, .stl and .step etc.

2.2.3 MAMP

MAMP is a software server tool for web development, able to create a local server environment for testing web applications locally.

2.2.4 NumPy

NumPy library provides powerful N-dimensional array objects and sophisticated functions for scientific computing with Python.

2.2.5 Pymesh

Pymesh is a geometry processing library that runs on Python 2 and 3. Library has built-in functions that allows user to manipulate mesh files using Boolean operations with simple function calls. It is able to do basic data handling with the NumPy module for inner calculation.

2.2.6 ProgressBar (ProgressBar2)

ProgressBar is a Python library that allows users to visualize progress in applications runtime. The visual display is in text format and will appear in the console.

3 Requirements and Analysis

3.1 Requirements

Each requirement in the table is given a unique ID for convenience, since requirement reference might be needed as project progresses. Furthermore, the requirements were prioritised individually using the MoSCoW prioritisation technique. The MoSCoW system consists of four categories, Must Have, Should Have, Could Have and Won't Have:

- **Must Have** requirements – Features that are required and must be included in the application. The application will fail without satisfying these requirements.
- **Should Have** requirements – Features that are important and can add extra functionality to the application. However, these requirements are not critical, the application can still be useful without it.
- **Could Have** requirements – Features that are not essential to the application and can be left out. These requirements are removed, if timescales are later at risk.
- **Won't Have** requirements – Features that are not to be implemented at this stage of development and may be included in future development.

3.1.1 Functional

ID	Description	MoSCoW
Category: Retrieving and Storing Data		
FR1.1	The application should be able to read ASCII mesh files	Must Have
FR1.2	The application should be able to extract data from the ASCII mesh files and store it into variables.	Should Have
FR1.3	The application should be able to store data into files for further computation.	Should Have

FR1.4	The application should be able to save the lattice mesh into files.	Must Have
Category: Processing Data		
FR2.1	The application should be able to process 90 and 45 degrees angle meshes	Must Have
FR2.2	The application should be able to duplicate the input mesh data	Must Have
FR2.3	The application should be able to calculate the lattice structure in any dimensions	Must Have
Category: Graphical User Interface		
FR3.1	The application should allow user to construct lattice of any dimensions that inputted by the user	Must Have
FR3.2	The application should be able to display the lattice structure	Must Have

3.1.2 Non-Functional

ID	Description	MoSCoW
Category: Compatibility		
NFR1.1	The application should be compatible with Windows devices.	Must Have
Category: Graphical User Interface		
NFR2.1	The application should have a simple and user-friendly graphical user interface.	Should Have
Category: Performance		
NFR3.1	The application should not take more than 5 seconds to start.	Should Have

3.2 Entity Relationship (ER) Diagram

The ER diagram represents the connections between the data and the program itself. The entities illustrate the data that is used or stored by the program. The attributes in each entity describes the data that are stored within the entity. To show the relationship between entities, the cardinalities between entities are defined in the diagram.

Figure 3.1 details the data that is read and stored by the application. As shown in the diagram, the application can read ASCII file. The ASCII files contains elements and nodes data. Each mesh element contains 3 nodes and a node can be mapped to multiple mesh elements. The

data set is then analysed to generate a lattice data set and output it as a .stl file. Finally, the Gmsh application is called to convert .stl file into .msh format to remove duplications.

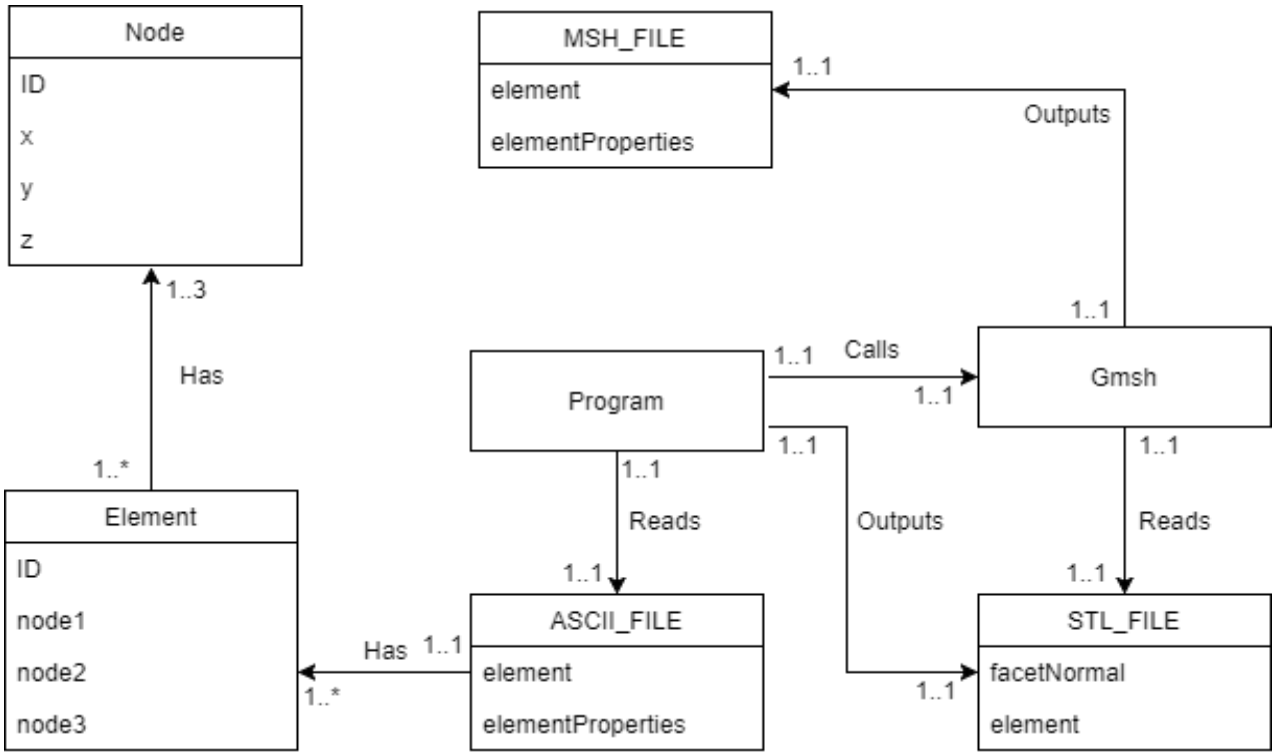


Figure 3.1: Application Entity Relationship Diagram

4 Design and Algorithm Implementation

4.1 Application Flowchart

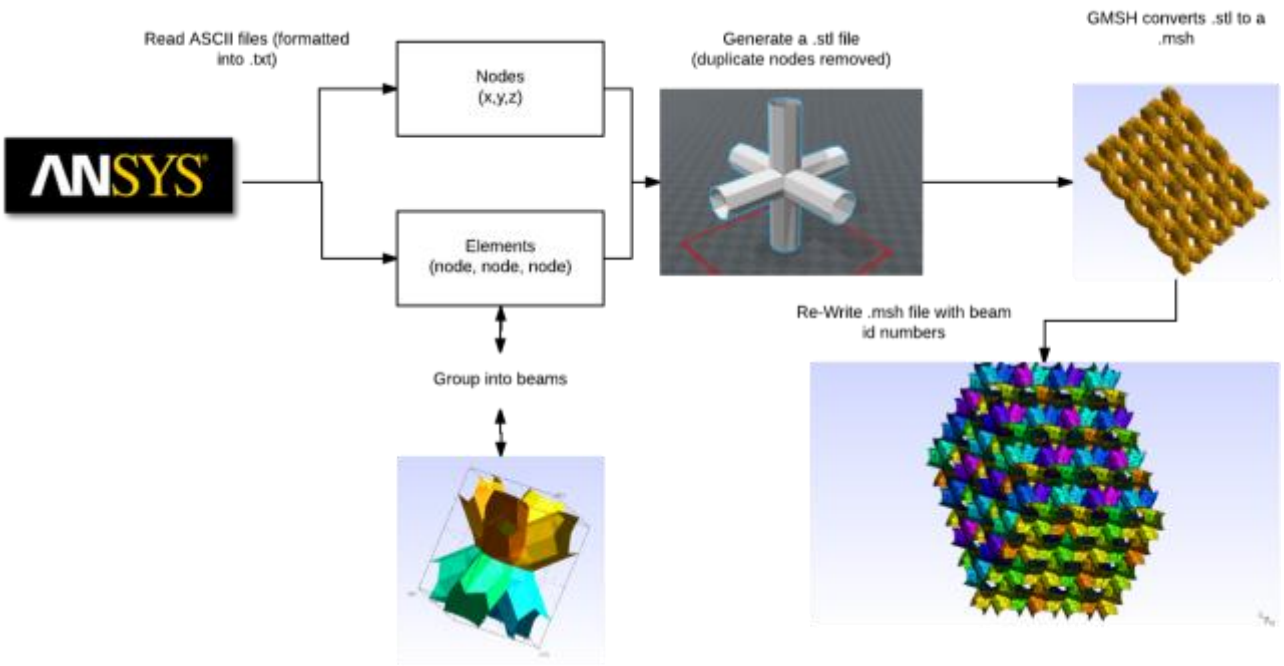


Figure 4.1: Application Flowchart

Figure 4.1 shows a flowchart of the developed application. The flowchart describes the sequence of steps that are taken by the program, based on the information given to each part of the program, starting with the user input.

The application first proceeds with the data retrieval process and reads the unit mesh data into Node and Element objects. Node objects store x, y and z coordinate values while Element objects point to Nodes and material attributes. The application then divides these Element objects into beams and assigns them identification numbers (which are the last number of the material attributes). The application proceeds to multiply the data to create a lattice, and

simultaneously write it to .stl file format. Then the application calls a bash script which invokes GMSH to convert the output from .stl to .msh file format. Then, finally the application rewrites the .msh with each element's material values (for beam identification).

4.2 Algorithm Implementation

The following algorithms are intended to identify beams and group nodes and elements into beams. Because there are two different unit shapes or 3D models we are using to create lattices, there are two algorithms to account for that.

4.2.1 90 Degrees Model

When looking at the 90 degrees unit model in Figure 4.2, the set of vertical lines on each of the beams have similar direction vectors and there are no irregular meshes. Hence, the beams on the 90 degrees model can be easily identified.

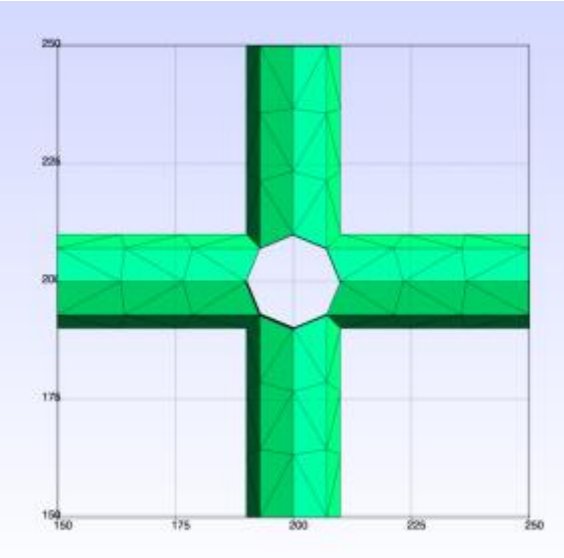


Figure 4.2: 90 Degree Unit Model

In Figure 4.3, the grey dots indicate the boundary nodes and the highlighted red triangle represents one of the elements within the mesh. The algorithm implemented for the 90 degrees model, filters out all the boundary nodes and group the boundary nodes that belongs to the same beam together by using the nearest neighbour algorithm. Next, identifies other nodes that are lies on the dotted lines by finding the next nearest node with the same direction

vector. By this method, all the nodes can be identified up to the mid-point of the mesh unit and group the nodes into separate beam objects.

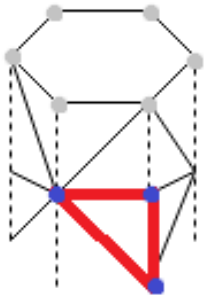


Figure 4.3: Beam Mesh

In Figure 4.4, the code stores nodes in the order from the closest to the furthest to the node variable inside an array. This is done, by comparing and computing the distance between node and all the nodes in the nodes list and perform indirect sort to that array. Here is the code for the nearest neighbour algorithm:

```
def nearest_neighbour(node, nodes):
    d = [((n.xyz - node) ** 2).sum() for n in nodes]
    ndx = np.argsort(d)
```

Figure 4.4: Nearest Neighbour Code

Here is the pseudocode for the algorithm:

```
For each Boundary Node
       Find nearest point in Nodes
       Calculate direction vector
       get_nodes(point, vector)

get_nodes(point, vector)
       Find nearest point in Nodes
       If point is on the line and < mid point
              Add node
              Remove node from nodes
              Repeat with new node
       If point not on line
              Remove node from nodes
              Repeat with previous node
       Return
```

Figure 4.5: Algorithm Pseudocode

4.2.2 45 Degrees Model

This model proved to be a little more difficult. When taking a look at the direction vectors created between nodes, the vectors that were on the same beam were not necessarily scalars of each other. Therefore, the algorithm used for the 90 degrees model would not apply the same to this model. Noticing that the unit shape for a 45 degree model is made up of eight beams, and that each beam lies in a different octant, the approach to detecting beams for this model was to divide elements up by which octants in 3D space they lay in. The result of this algorithm, shown in Figure 4.6, is a unit shape with each different beam holding a different identification number (material attribute).

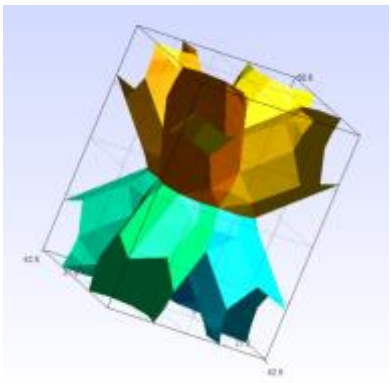
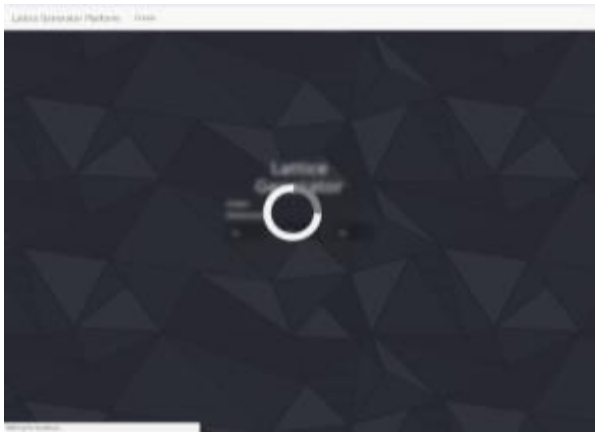


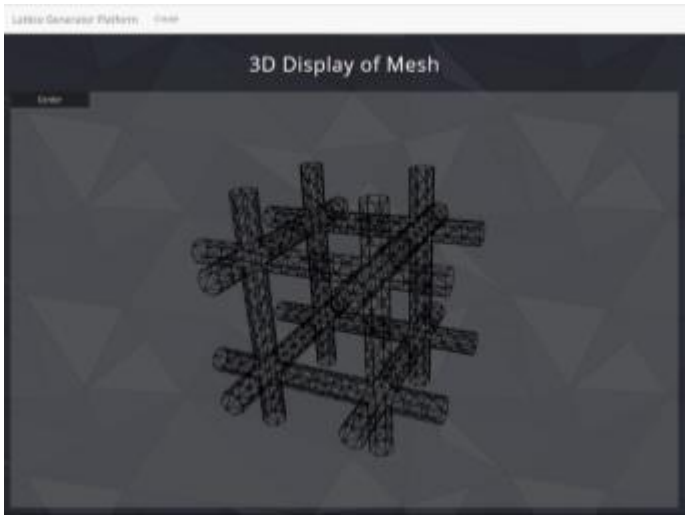
Figure 4.6: 45 Degree Unit Model Divided by Octant

4.3 Graphical User Interface Design

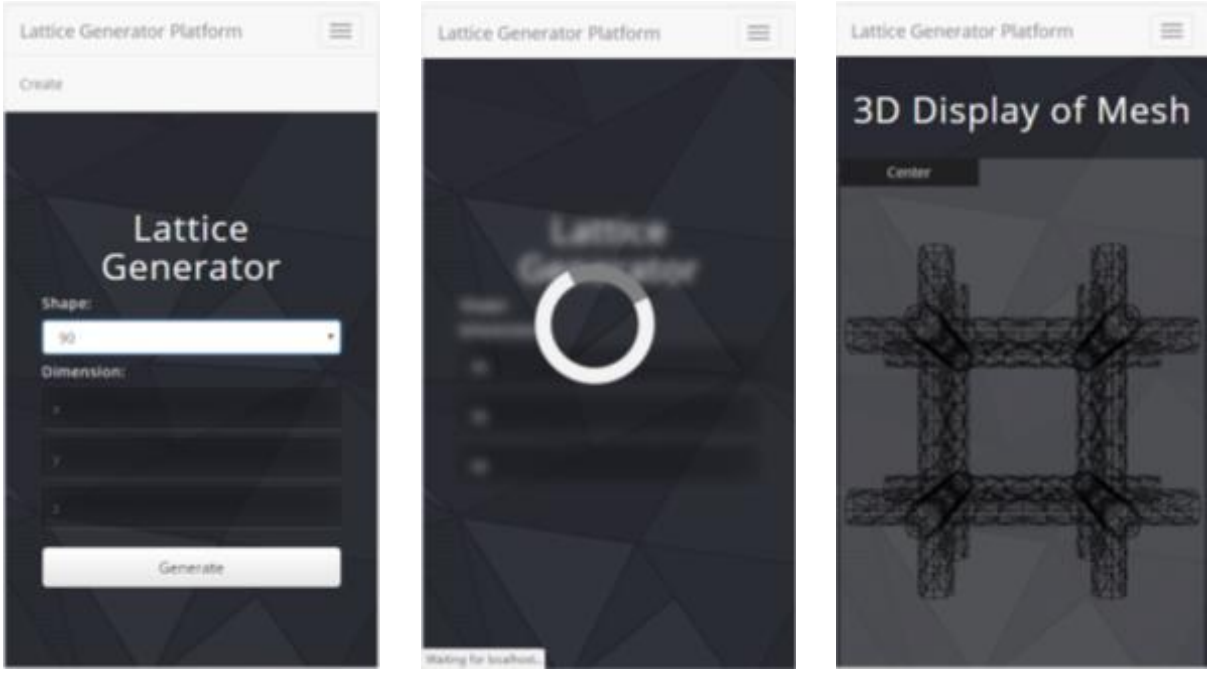
Homepage



Loading Page



Mesh Viewer Page



Mobile Version

5 Evaluation and Conclusion

5.1 Evaluation

For the runtime of the application, adjustments were made to the lattice multiplication\writing algorithm. In Figure 5.1 are the runtimes and file sizes chronicling the change of different optimization adjustments made. When all the multiplication was changed to addition we saw a 20+% decrease in runtime. When newlines were changed to spaces (when possible) there was about a 4% decrease in file-size. With so many calculations and operations to be ran in multiplication, longer runtimes and file sizes may be inescapable. However, any small optimizations made can be felt by the user in a big way.

	Multiplication (sec)	Addition (sec)	\n (bytes)	' ' (bytes)
0x0x0	0.0005599570274	0.0007019281387	88	88
1x1x1	0.002931296825	0.002108049393	13288	12154
5x5x5	0.2205212712	0.1813923359	1944710	1876260
50x50x1	4.407399631	3.57706244	43999026	42211326
20x20x20	13.90983202	11.51052659	144160240	140549840
100x100x1	18.25467761	14.43250287	184637240	177501040

Figure 5.1: Table of Optimization Analysis

5.2 Conclusion

Lattice volume generation, although a longer process than most, is possible. The application developed in this project is capable of writing large lattices to .stl file format and .msh file format. Although at this stage, the algorithms used are not a silver bullet solution, they work to create lattice volumes from two unit shapes: a model divided by 45 degrees and a model divided by 90 degrees in all directions. Our application comes in two forms: a web-app based

GUI and a console version. The web-app allows for in browser viewing of the basic lattice structure (not separating material types) with a X3D display of the nodes and elements in all the same color. This viewer will struggle with bigger models. The console version allows users to accurately track the progress of the application, but the output must be viewed with other CAD software such as ANSYS or GMSH.

While the runtime of the application depends on the size of the volume and file sizes are large (also depending on the volume), this application has been developed in order to accomplish the functional and non-functional requirements. The inputs can be read, the program starts and the outputs are stored. On top of that, the optimizations made to the application have improved both runtime and output file size greatly.

Appendices

A Source Code Repository

The source-code for this project is available on a private GitHub repository. To request an access to the repository please send an email to Antony.Hwang.14@ucl.ac.uk.

GitHub Repo: https://github.com/AntonyHwang/Lattice-Structure-Computation-Program.git

B Application User Manual – Console Version

Lattice Structure Computation Program – Console Version

Getting Started

These instructions will get you a copy of the project up and running on your local machine for development and testing purposes. Download the project files from the **master** branch on GitHub here.

Prerequisites

Operating System: Windows

Software Requirements:

- Python 3.6
- NumPy
- Progressbar2
- GMSH
    - GMSH can be installed from the link here. After you extract it, the direcctory must be added to the path. Make sure gmsh is a valid command in your console

Numpy and Progressbar can be installed by calling:

```
pip install numpy
pip install progressbar2
```

in the command-line.

Generating a Lattice

Run Lattice Generation by running in the project directory:

```
python app.py
```

Input the model you want to multiply 45 or 90, followed by the dimensions of the volume x, y and z.
Output files are placed in lattice\output in .stl and .msh format.

Running the tests

Go to the directory and run:

```
python unit_test.py
```

To test different values, change the *x,y,z* values in individual_test().
To run a general set of unit tests, replace `individual_test(...)` with `unitTest(nodes, elements, displacement_factor)`.
All test outputs are written in `output\test_output.txt`

C Application User Manual – Web App Version

Lattice Structure Computation Program – Web App Version

Getting Started

These instructions will get you a copy of the project up and running the web app version on your local machine for development and testing purposes. Download the project files from the **web-version** branch on GitHub here.

Prerequisites

Operating System: Windows

Software Requirements:

- Python 3.6
- MAMP
  - MAMP can be installed from the link here.
- NumPy
- Pythonocc
  - Install Pythonocc follow the instructions on here.
- GMSH

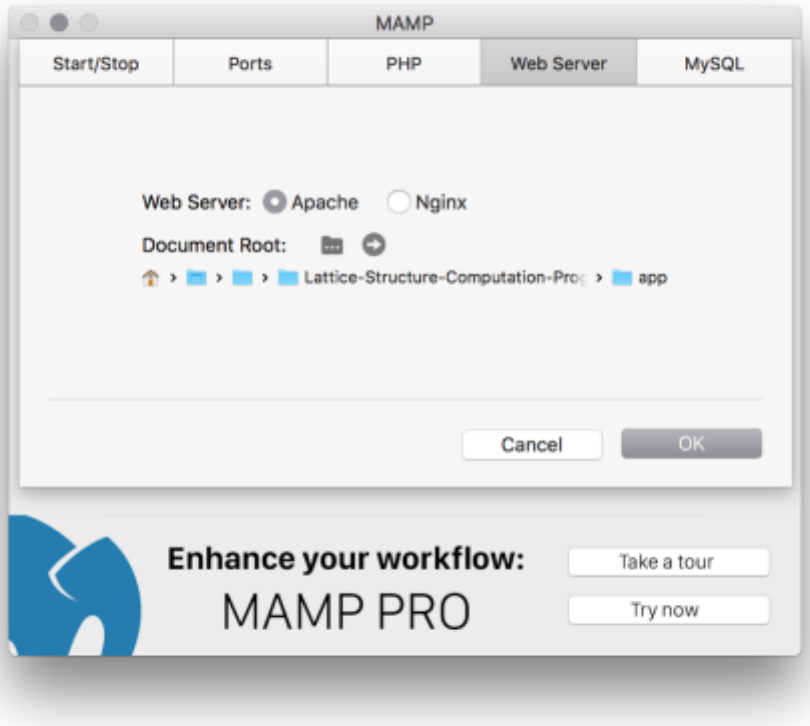
- GMSH can be installed from the link here. After you extract it, the direcctory must be added to the path. Make sure gmsh is a valid command in your console

Numpy can be installed by calling:

```
pip install numpy
```
in the command-line.

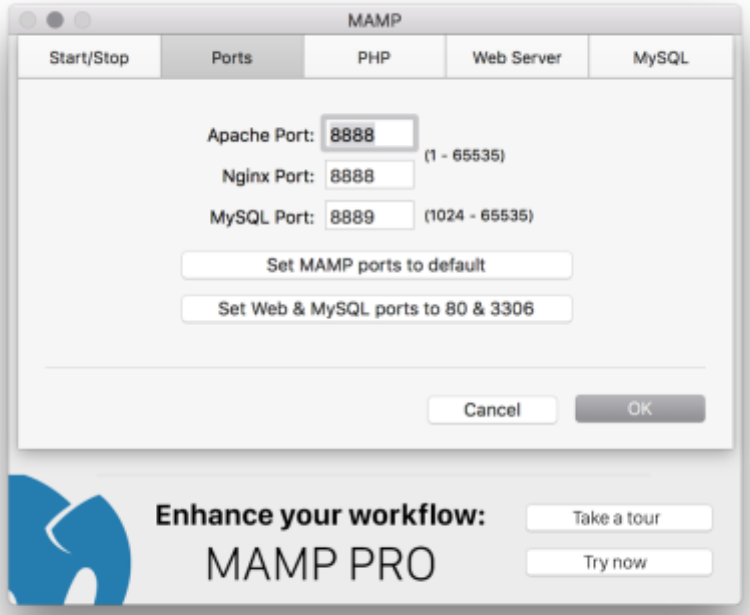
Setting Up Web Application

- Extract the source code downloaded from GitHub
- In ../app/includes/config.php file
  - Change the $py_path variable to the path of the python.exe on your machine
- Open MAMP
  - Go to **MAMP Preferences** page
  - Go to **Web Server** Page
    - **Select** the app folder in the root of the source code folder as the **document root**



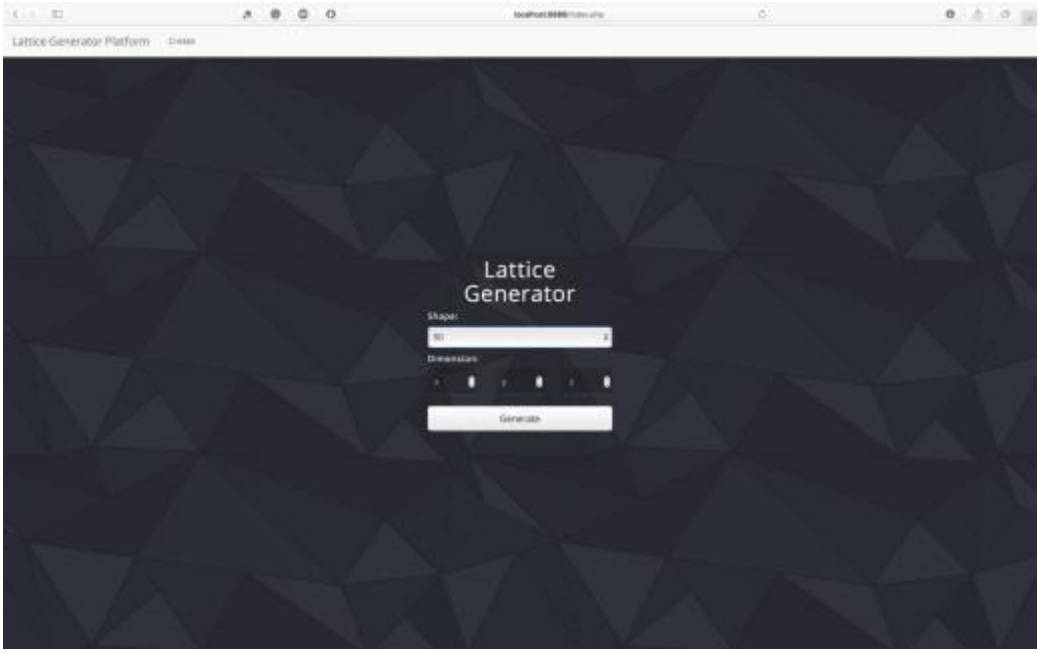
- Click on **OK**
- Click on **Start Servers**

- Go to **Ports** page in the **Preferences** page
  - Record your **Apache Port**
  - In this case, type http://localhost:8888 in browser to run the web app



Generating a Lattice

- Select the shape in the drop-down box
- Input the dimension of the lattice
- Click on the **Generate** button to construct the lattice mesh



D Video Demo

Lattice Generation Program Run Through Console Version:

https://www.youtube.com/watch?edit=vd&v=WxrpLdT6CLw
Lattice Generation Program Run Through Web App Version:

https://www.youtube.com/watch?v=ZSYUJ9MVYKg&feature=youtu.be

E Code Reference Guide

The code reference guide is available in a separate .pdf document. Please see the **Lattice_Generation_Reference_Library.pdf** file.