Graduation Project Report

Computation design method for Statics-Responsive Grid Shells

AUTHOR:
Gautam Tanwar
gautamtanwar@gmail.com
4718712

FIRST MENTOR:
Dr MSc Arch. Michela Turrin

SECOND MENTOR:
Ir. Peter Eigenraam

28/06/2019
Computation design method for Statics-Responsive Grid Shells

AUTHOR:
Gautam Tanwar
g.tanwar@student.tudelft.com
gautamtanwar@gmail.com
4718712

FIRST MENTOR:
Dr MSc Arch. Michela Turrin
M.Turrin@tudelft.nl

SECOND MENTOR:
Ir. Peter Eigenraam
p.eigenraam@tudelft.nl

EXTERNAL SUPERVISOR:
Dr Sylvia J.T.Jansen
S.J.T.Jansen@tudelft.nl

28/06/2019

TU Delft
Delft University of Technology
Acknowledgments

This document is the graduation thesis of Gautam Tanwar for MSc. Building Technology at the Faculty of Architecture and the Built Environment at TU Delft. I would like to acknowledge the following people for their help and support provided during the studio which made this thesis possible.

I would like to thank my mentor, Dr MSc Arch. Michela Turrin for her guidance and support, for understanding interests and encouraging me to pursue them, for sharing her knowledge about computational optimization and exploration. I would like to thank my mentor, Peter Eigenraam for his guidance and insights in parametric design of shell structures.

I would like to thank my family for their help and support. Without their stability and encouragement, this master would not have been possible.

I would like to thank all my peers. I learnt a great deal from them and their friendship in tough situations was invaluable. I would like to thank all teachers at the faculty of Architecture and the built environment for giving and arranging inspiration lectures and imparting knowledge and skills without which this thesis would not have been possible.

Abstract

Quadrangular Grid shells have generated interest in recent years for their application in rationalizing free form geometry in the built environment. Shell structure are efficient because their form is governed by flow of internal forces. But while discretizing shells into grid shells, instead of using flow of forces, current method follows patterns and tessellation techniques. Quadrangular grids are easier to manufacture but they are not stiff inherently compared to triangulated meshed grid, which doesn’t allow them to be used as frequently. There is a scope to improve stiffness by discretization informed by flow of forces.

A workflow was developed for designing quadrangular static-responsive grid shells which are structurally efficient, homogenous and has near planar cladding, including preferences of the designer. The workflow is set up in a parametric environment in grasshopper, a plugin for Rhino 3d modelling software. It uses particle spring method for form finding a shell which has membrane like load bearing behaviour. The solid shell is discretized into a grid shell by a custom stress line generator which uses principal stress vector field derived by Finite element analysis of the shell. The grid shell is homogenised and optimized for planarity by dynamic relaxation. Multiple design alternatives are generated and stored. Design space is explored by using data analytics and visualization techniques which helps user to make informed design decisions. The workflow is applied to create a grid shell over delft bus station as a case study to protect travellers from varying weather conditions.

The results are quite satisfactory in terms of structural performance when compared to methods used in state of the art in practice. Stiffness of a structure can be measure by total strain energy. The grid shell for delft bus station generated using this workflow was 32% and 49% lower in terms of total strain energy (compliance) than regular quad grid shell and diagrid quad shell respectively. The results are promising for real life application. Meaning that the workflow can be used to find a homogenised quadrangular grid shell which are stiffer than their predecessors. Grid shells are used for approximating free form geometry for various projects around the globe. Using this method can save time, money and material which was required to make a grid shells stiff by thicker beams or extra stiffening members.

Key words: Grid Shells, planar quadrilateral, Principal stress lines, surface discretization, structural optimization, parametric design, statics, stiffness.
Table of Contents

1. Introduction
   1.1 Problem Statement
   1.2 Research Questions
   1.3 Aim/Objectives
   1.4 Relevance
   1.5 Methodology
   1.6 Time Planning

2. Literature Review
   2.1 Background Shell Structures
      2.1.1 Collaboration of structure engineers and architects
      2.1.2 What is a shell?
      2.1.3 Membrane Theory
      2.1.4 Buckling
      2.1.5 Types of Shells
      2.1.6 Funicular arches
   2.2 Grid Shells
      2.2.1 Shift from continuous concrete shells to grid shells
      2.2.2 What is a grid shell?
      2.2.3 Strained/kinematic grid shells
      2.2.4 Unstrained grid shells
   2.3 Form Finding
      2.3.1 Particle Spring system
   2.4 Structural optimization
   2.5 Surface discretization
   2.6 State of the art in practice
      2.6.1 British Court Museum Roof
      2.6.2 YAS Hotel
   2.7 Statics of grid shells
   2.8 Precedent work
   2.9 Design exploration

3. Prerequisites for workflow
   3.1 Parametric modelling
   3.2 Tools Definition
   3.3 Workflow Scope
   3.4 Workflow strategy

4. Workflow Definition
   4.1 Form finding
   4.2 Finite element analysis for vector field
   4.3 Custom stress line generator
   4.4 Varying density
   4.5 Converting stress lines to a grid
   4.6 Subdividing polygons to quadrilaterals
   4.7 Homogenisation
   4.8 Alignment with principal stress lines
   4.9 Planarity
   4.10 Structural analysis (FEA)
   4.11 Generation of design alternatives and storing data
   4.12 Discussion
   4.13 Proposed workflow

5. Case Study
   5.1 Design Constraints
   5.2 Initial Steps
   5.3 Sequential strategy
   5.4 Integrated Strategy
   5.5 Comparison between both strategies
   5.6 Validation

6. Results and Discussions

7. Conclusions

8. References

9. Appendix A - Grasshopper Scripts

10 Appendix B - Custom Tools, Python Code
Introduction

Built environment has been influenced by structure, aesthetic, cost and constructability and sustainability. The process of design considering these factors is usually fragmented. Depending on different stages of design process, some factors hold more importance than others. Usually, building’s massing, geometry and form are decided during the conceptual stages of design by architectural practices. Engineering plays a role in later stages and may or may not be able to influence the form of the building. The industry is moving towards integrated design approach where all factors affecting the built environment are considered at early design stages. It has been demonstrated by iconic projects in history that integrated designs which are influenced structurally at the early design stage do not adversely affect architecture or aesthetic goals. Sydney Opera House, Sagrada Familia in Barcelona and Eiffel Tower are all examples of form influence by structural reasons or flow of forces.

Surface structures or shells have received considerable attention by researchers after the advances in the field of Computer Aided Design and introduction of advanced design tools. Advances in Digital fabrication techniques has enabled designers to generate structurally inspired forms that are very complex. Shell structures are used in architecture for efficiently spanning large lengths or when using materials which can take compressive forces much more than tension. These structures are perfect example of form derived by flow of forces. More often shells are discretised into lattice or grid rather than using a solid continuous surface as they are lighter, easier, faster and cheaper to construct. Grid shells have been used to realise free form structures in architecture quite often because of their ability to resist loads efficiently and freedom of design exploration.

In comparison to form finding, little research has been done and less tools are available to generate high performing topologies. The discretization is done on the basis of patterns and tessellations and can be done in various different topologies like triangular, quadrangular, hexagonal or hybrids. Since topology of any structure influences its structural performance, appearance and constructability, there is a need for more research and methods to improve topology generation. All topologies have their advantages and disadvantages. Triangular meshing is inherently stiff than other topologies. Quadrangular meshing for approximating free form shells have generated interest in the industry because they can be lighter, their nodes are easier to manufacture with less or no torsion compared to triangulated grid shells.
1.1 Problem statement

Shell structures are efficient because their form is governed by flow of internal forces. But while discretizing shells into grid shells, instead of using flow of forces, current method follows patterns and tessellation techniques. Quadrangular grids are lighter, cheaper and their nodes are easier to manufacture but they are not stiff inherently compared to triangulated meshed grid, which doesn’t allow them to be used as frequently. There is a scope to improve stiffness by discretization informed by flow of forces. Some recent studies have shown promising results in improving performance of grid shells by focusing on statics and using principal stress direction for discretization.

But these methods have the following limitations:
- do not align beams with direction of stresses,
- do not achieve homogenised grid for feasibility of construction,
- do not produce planar cladding, and
- do not take preferences of the designer into account.

This thesis aims to overcome these limitations in current methods for layout of quadrangular grid shells.

1.2 Research questions

What can be the method(workflow) to design statics-responsive grid shells with structural efficiency, homogenous grid and near planar cladding including preferences of the designer?
- How can a solid shell be discretized into a grid shell with direction of beams aligned with direction of principal stresses?
- How can the grid of obtained shell be homogenised?
- How can obtained grid shell be optimized for planarity of faces?
- How can the workflow facilitate designer’s informed decision-making process?

1.3 Aim/objectives

To define a method(workflow) to design statics-responsive grid shells with structural efficiency, homogenous grid and near planar cladding including preferences of the designer?
- Define a method for form finding a shell structure with membrane-like load bearing behaviour.
- Develop a method to discretize solid shell into a grid shell with direction of beams aligned with direction of principal stresses.
- Develop a method to homogenize the grid pattern for better aesthetics and ease of construction.
- Define a method to optimize grid shell for planar faces.
- Define a workflow which enables designer to visualise the design alternatives and facilitate decision making with readable results.

1.4 Relevance

Shell structures have some unique features, which has developed interest in the design and engineering community. Shells can be very elegant and light if designed correctly. Large spans can be achieved with minimum material. Grid shells are sustainable in the built environment in following ways:
- They allow adaptive reuse of historic buildings and museums. They have been used to cover open courts or spaces without overcoming the architectural expression of the building.
- If designed efficiently, they can save up to 50% of structural material compared to conventional framed structures and can also be constructed using wood or cardboard tubes with lower embodied energy than conventional materials.
- They allow more natural light to enter the enclosure and facilitate cooling or heating of the space to create more comfortable environment (Malek,2012).

Grid shells are used to for approximating free form geometry. While grid shell structures are already efficient in spanning large spans, they can still be optimised by surface discretization informed by flow of forces. If the beams are aligned in the direction of principal stresses, more forces act axially on them which results in reduced bending moments in beams and reduced shear stress in nodes. Hence, resulting in smaller cross sections, lighter nodes and making the grid shell stiffer.

Carbon emission of steel is roughly 2 tonnes of Co2 per tonne of steel. By reducing the amount of steel or in fact any other material required to make large span structures, carbon footprint of the structures can be reduced. Developing a method to make quadrangular grid shells lighter and stiffer can make these structures cheaper and more environment friendly.

1.5 Methodology

Literature Research
- Review theory of shells, form finding, surface discretization, state of the art grid shells in practice and precedent work.
- Review the existing computational methods for design exploration, data management and visualization.

Workflow Definition
- Set up a parametric model for form finding surface structures membrane-like load bearing behaviour.
- Define an algorithm for making quadrangular dominant grid aligned with principal stress directions approximating the found form surface.
- Define an algorithm for homogenizing the grid pattern for better aesthetics and ease of construction.
- Set up parametric model for generating design alternative on the basis of density of grid, alignment with principal stress direction, planarity and size of cross-section of beams.
- Define a method to visualise, analyse design alternatives and facilitate informed design decisions.
- Propose a workflow

Case Study
- Apply the workflow on a case study, validate the method and discuss results.
Figure 1.1: Overview, Source: author
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Topic Selection</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1 Selection of Graduation topic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2 Research and project overview</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Literature Research</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1 Background and methodology research</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.2 Shells and membrane theory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3 Grid shells and types</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4 Form finding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5 Structural optimisation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.6 Statics of grid shells</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.7 Precendent Work</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.8 Computational Optimization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9 Design Exploration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.10 Data Analytics and Visualisation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.11 Report and Presentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Practical Research (Project Development)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.1 Prerequisites for workflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.2 Form Finding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3 Finite element analysis for vector field</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4 Custom Stress line generator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.5 Converting Stress line to grid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.6 Homogenize grid pattern</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.7 Alignment with principal stress lines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.8 Optimise for planarity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.9 Structural Analysis (FEA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.11 Generating Design Alternatives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.12 Case Study and Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.13 Results Discussions and conclusion</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Final Documentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.1 Final Report</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.2 Final Presentation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.2: Time Planning
2.1 Background Shell Structures

2.1.1 Collaboration of structure engineers and architects

In ancient and renaissance times, there was no separation between roles of an architect and structure engineer, then only existed the master builder who had knowledge of every aspect of constructing a building. This resulted in truly informed designs where structure was not separated from the building. With advancement in technology and increase in complexity of building design, it is no more possible for one professional to know everything, which gave rise to multiple disciplines where professionals specialize in their field of work and deal with their limited field only. Architects job is to form a collaboration between every discipline.

The usual practice is that the architect develops the concept and design and structure engineers are responsible for the stability of the structure. However, this depends on the type of project, for example a small family house does not need an engineer necessarily and a bridge if it only serves a singular structure purpose might not need an architect. However, a high-rise building need both. The more the form of the building or structure depends on the flow of forces, the more it is under the responsibility of engineer (Schlaich, 2014).

For shell structures, shape is derived from their flow of forces for which engineers have developed form finding methods. Generally, the collaboration works in such a way that the architects comes up with an idea, concept, proposal and structural solution is derived by engineers.

2.1.2 What is a Shell?

A shell is a structure defined by a curved surface. It is a thin surface, where thickness is referred in the direction perpendicular to the surface. However, it might be curved in one direction like a cylinder or two directions like a dome or a cooling tower. In addition, shell is also identified by its rigidity to distinguish from tension structures like cable nets or membranes. The relationship between a beam and an arch is similar to the relationship between a plate and a shell. Similar to arches, shells resist load perpendicular to surface by tension or compression parallel to the surface (Williams, 2014).

Just as a beam can be assembled by using straight element like a truss, a shell can me created by assembling straight elements for example grid or lattice or reticulated shells. Elements can be in a single layer or multiple layers like a space frame.

2.1.3 Membrane theory

Shells use all available modes of structural action used by beams, struts, cables, arches and plates, and another mode called as ‘membrane action’. An arch or a cable can carry vertical loads by axial compression or tension. Membrane stresses in a shell are comparable to these axial compression or tension in an arch or a cable. Membrane stresses are caused by forces acting tangentially to the surface of the shell.

Membrane action shows relationship between stress and curvature. At every point on a surface, radius of curvature can be calculated in any direction tangent to the surface at that point. Two of these are principal curvatures, just like principal stresses. Gaussian curvature is the dot product of principal curvatures.
• If both principal curvatures are positive or both are negative, the dot product and gaussian curvature is positive. These surfaces are called synclastic.
• If the principal curvatures are of opposite signs, then the dot product and gaussian curvature is negative. These surfaces are called anti-clastic.
• If one of the principal curvatures is zero, then gaussian curvature is also zero. These surfaces are cylindrical.
• If both principal curvatures are zero, then also gaussian curvature is zero. These a plane surface.

![Types of gaussian curvature](https://discourse.mcneel.com/t/curvature-analysis-unit/37027/8)

Gauss's theorem (Theorema Egregium) states that the gaussian curvature of a surface can be calculated by measuring lengths on the surface. Membrane stress try to stop the change in lengths on the surface. When a surface deforms without change in lengths on the surface that means when gaussian surface does not change, it is called inextensible deformation. Developable surfaces like cones and cylinders can made by folding a plane surface without changing the lengths of a surface. Thus, a surface which cannot be bent inextensibly can work by membrane action. As stated in the Cohen-Vossen theorem from differential geometry, a closed surface with positive gaussian curvature like an egg, cannot be deformed without changing lengths on the surface. But half an egg can be deformed inextensionally. This stiffness can be regained by fixing it to a support to form a dome. Thus, membrane action does not only depend on the gaussian curvature but boundary conditions as well (Williams, 2014).

Shells structure are also susceptible to twisting and bending moments and conforming shear forces perpendicular to the surface. Shells should still be designed to work through membrane action rather than bending energy as they are stronger and stiffer that way. (Williams, 2014).

2.1.4 Buckling

Columns carry load via axial forces, but bending stiffness is required to stop buckling. Shells carry load in the same way, although they resist buckling and inextensional deformation by a combination of bending and membrane action. The membrane stiffness in shells is much higher than the bending stiffness. A shell structure can absorb a lot of membrane strain before deforming a lot. The buckling phenomenon is when the shell structure is suddenly failing dramatically in order to exchange the membrane energy into bending energy (D. Bushnell, 1981).

Buckling in shell structures is particularly critical because they suddenly collapse without showing much deflection. The analysis of buckling in shell structures by hand calculation is next to impossible. Even eigenvalue analysis of spherical shells can give wildly optimistic results. This is the reason why computer analysis is required to analyse buckling in shells, keeping in mind that these results should be carefully examined by principles from classical theory of shells. There is still room for physical testing with models for qualitative insight which cannot be obtained from numerical analysis (Williams, 2014).
2.1.5 Types of Shells

Shells can be categorised into 3 types based on how their geometry was created. (Adriaenssens et al., 2014):

1. Free form shells – They are developed using influence from organic shapes or sculptural design development. They are generated without considering structural requirements.
2. Mathematical or geometrical shells – these are based on analytical functions. For ease of construction and calculations, mathematic functions are used to define their geometry. Typical shapes are represented by quadratic surfaces, such as hyperboloids, ellipsoids and hyperbolic or elliptic paraboloids.
3. Funicular or found form shells – The shape of these shells is derived by hanging models or computer aided form finding methods. These structures can carry funicular forces without any bending moment, which means that the line of thrust follows the geometry.

2.1.6 Funicular arches

If a rope carries its own weight, when hung it takes the shape which is referred as ‘catenary’. If the rope carries a uniform vertical load per unit horizontal length, then it takes the shape of a parabola. The shape rope takes under the corresponding funicular load is called a ‘funicular shape’.

Figure 2.4: Catenary (solid curve) and parabola (dashed curve) (Williams, 2014).

Figure 2.5: Replicated hanging chain model of church of Colònia Güell by Gaudi. Source: author. The structure of the church of Colònia Güell by Gaudi was based on funicular shapes.
2.2 Grid shell

2.2.1 Shift from continuous concrete shells to grid shells

Thin concrete shells during 50s-60s were very popular because of their elegance and structural and material efficiency. But these structures did not spread widely. One of the most obvious reasons could be that the cost of human work was increasing in comparison with cost of materials. The doubly curved formwork which can get very complex required to build these concrete shells would have been specifically made for each project and could not be used again resulting in repeated labour. In current times, material optimization with shell structures is not economical if a lot of manual labour is required. Grid Shells presented the opportunity to reduce the manual labour considerably to bring down costs. One of the first reputed examples of rigid grid shell was Mannheim Multihalle (1975) by Frei Otto which used elastic deformation to convert simple geometry of flat grid to complex doubly curved geometry. Secondly the development of computer aided manufacturing techniques enabled unique components to be produced at acceptable costs.

2.2.2 What is a grid shell?

A grid shell is a structure with the form and mechanical properties of a continuous shell but is made of grid or lattice instead of a solid surface. They can span large areas with very less material. They can be made using wood, steel aluminium, cardboard tubes and even fibre reinforced polymers. One can say that shell action in grid shells is imitated by discretized shell elements or edges. The elements can only transmit forces in the direction of the edges and can resist out of plane bending. There are broadly two types of grid shells, kinematic (strained) and unstrained grid shells.

2.2.3 Strained/Kinematic grid shells

Strained grid shells are made by bending a flat quadrangular lattice of beams/bars hinged connected at their intersection into a shell. These are usually made with wooden beams to be able to bend. After the desired shapes is achieved, the required stiffness is achieved by adding a cross bracing on each quadrilateral. This cross bracing could be a prestress cable like in the Mannheim Multihalle (figure 5a) or diagonal beam. In some cases, even cladding adds addition stiffness to the structure like in the Savill Garden grid shell (figure5b). The main advantage of these types of grid shells is their ability to transform from a regular size grid into the desired shape which is cost effective compared to unstrained grid shells where each member and joint has to be manufactured to fit in their specific position. Disadvantages include bending stress in the members and large amount of time required on site to erect the structure.

2.2.4 Unstrained grid shells

Unstrained grid shells are erected in the required shape with factory made parts which are sometimes custom made. These are usually made with steel because they don’t need to be bent on site. Standardization of elements like all members having equal length or planarization of surface for cladding with quadrangular meshes for saving in cost and ease of construction at site may be required for these types of shells. But quadrangular lattice may not be suitable for these shells from the point of view of stiffness and achieving planar surfaces. Cross bracing or use of triangular meshes has been done in the past for unstrained grid shells as well. Advancement in digital design and fabrication techniques have made these shells economical compared to 20th century. Examples given in chapter 2.6.
2.3 Form finding

As part of conceptual design of shell structures, it is important to have structurally adequate shape for
the load bearing behaviour and architectural expression of design. Due to intrinsic interaction between
form and forces, the shape for shells should not be freely chosen. Form finding is needed to create
form which follows the forces to create force equilibrium structures. In this thesis form finding will be
used to generate shapes for optimising grid of the shell as well as optimising free form shapes towards
equilibrium structures keeping the boundary conditions same. There are many established form finding
methods for shells (with year of first publication in context of shells), to name a few, Force Density
Method (1974), Dynamic Relaxation (1977), Particle Spring System (2005) and Thrust Network Analysis
(2007). Many algorithms and digital tools for form finding in architecture use a particle-spring framework
to simulate hanging or pretensioned chains and grids.

2.3.1 Particle Spring system

The objective of particle spring system is to find static equilibrium for structures. They are based on
lumped masses, called particles. These particles are connected with linear elastic springs, together they
form a network. Each particle has its own mass and external force acting on it. Each spring is having an
initial length (rest length), a constant axial stiffness and may or may not have a damping coefficient.

If an external force (for example gravitational pull) is applied to a particle/node, it causes displacement
and elongation of the springs attached to it. Due to this elongation, the spring exerts a counter force
based on spring’s offset from its rest length. The elongation continues till the external force applied
on the node matched the counter internal force exerted by the spring to create a balance of forces on
each node and overall to an equilibrium shape. ‘The motion of the particle is governed by Newton’s
second law of motion, and the force in the spring by Hooke’s law of elasticity.’ (Bhooshan, Veenendaal
& Block, 2014). Each particle is free to move in any direction. Displacement vectors of each particle can
be restrained by adding supports.

Usually when the simulation is started, the particle spring system is not in equilibrium. As the particles
and spring seek their equilibrium position, the movement persists throughout the system. The
particles continue to move, and springs continue to stretch till the system reaches equilibrium or the
simulation is terminated. It is necessary to apply damping to the system to prevent the particles from
oscillating about their equilibrium positions. Damping can be applied as a coefficient to each spring or
particles can be subjected to viscosity in the surrounding environment. Even when the system appears
to be not moving and equilibrium is achieved if even exists, the solver doesn’t stop and keeps on revising
the position of particles. A limit has to defend for the change of position of every particle or velocity of
the particles as cut of threshold to stop the simulation (Kilian & Ochsendorf, 2005).
2.4 Structural Optimization

The objective of the thesis is to develop a tool for optimizing gridshell structures which requires to understand various types of optimization and what are the objective functions, constraints and design variables used for solving structural optimization problems. Usually any of the finite element responses or geometric characteristics of the model can be used as objective function for minimization or maximization. Often mass, strain energy, displacement and natural frequencies are used as objective functions. Similarly, for constraints often mass, displacements, velocities, accelerations, stresses, strains, natural frequencies, bucking load factors, and/or temperatures are used. Design variables are parameters that can change directly or indirectly the dimension of elements, grid locations and/or material properties. These variables depend on the type of structural optimization performed (Leiva & Watson, 2016).

Types of structural optimization

There are majorly three types of structural optimization, Sizing, shape, and topology optimization. They address different aspects of the structural design problem and has a different set of boundary conditions (Christensen & Klarbring, 2008).

Sizing optimization

In a typical sizing problem, the objective is to find the optimal thickness of structural members such as a grid shell members’ cross section. By changing the thickness of each member, the objectives would be to minimize strain energy (compliance), peak stress, deflection, etc. of the whole structure while equilibrium constraint and other design variables are satisfied (Christensen & Klarbring, 2008). One approach could be to find optimal size for each member but aesthetic and feasibility constraint for manufacturing and assembly comes into play. Other approach could be to find the maximum size for one of the components and apply it to all. This approach can lead to wastage of material, but manufacturing and assembly could be faster depending on the production process. A compromise between the two approaches may also be used, where members are divided into groups and cross sections are assigned per group. There also exist case like the Great British Court grid shell where members were tapered (Williams, 2001).

Shape optimization

In a shape optimizing problem, the objective is to find out the optimal shape by changing the boundary conditions. By changing the shape, the objective would be to minimize a certain cost function or objective functions like strain energy (compliance), peak stress, deflection, etc. while satisfying given constraints. The topology of the structure that is the connectivity of the elements remains unchanged, but the nodes can move to alter the geometry (Christensen & Klarbring, 2008). In case of shells, the change in shape can result in better load transfer via bending or membrane action, hence achieving the objective functions.

Topology optimization

In a topology optimization problem, the objective is to optimizes material layout within a given design space under boundary conditions. Neither the topology nor the geometry is a constraint. The objective is again to minimize or maximize) certain objective functions. The optimization method computes the most efficient material distribution in the design space (Christensen & Klarbring, 2008).

Figure 2.7: Types of structural optimization. a) Sizing optimization of a truss structure, b) shape optimization and c) topology optimization (Bendsoe & Sigmund, 2003).
2.5 Surface discretization

For form founding and finite element analysis, continuous surfaces are discretized into meshes. This facilitates fast intersections and distance computations for rendering and dynamic simulation. They also constitute the geometric foundation of discrete methods for solving differential equations. For grid shells discretization is not limited to analysis but application also. It is important to study the implications of discretization of free form surfaces into different topologies. Most common topologies adopted are triangular, quadrangular and seldom hexagonal.

<table>
<thead>
<tr>
<th>Surface Approximation</th>
<th>Face Planarization Complexity</th>
<th>Valence of Regular Nodes</th>
<th>Torsion of Nodes</th>
<th>Overall Stiffness</th>
<th>Sensitivity to Imperfections</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangular</td>
<td>Optimal</td>
<td>Intrinsically flat</td>
<td>6</td>
<td>Yes</td>
<td>High</td>
</tr>
<tr>
<td>Quadrangular</td>
<td>Good</td>
<td>Fair</td>
<td>4</td>
<td>No</td>
<td>Low</td>
</tr>
<tr>
<td>Hexagonal</td>
<td>Quite good</td>
<td>Difficult</td>
<td>3</td>
<td>No</td>
<td>Very Low</td>
</tr>
</tbody>
</table>

Figure 2.8: Comparing grid shells’ topologies main properties (Andreussi, 2016)

- A node (vertex) is a point where edges converge,
- The valence of a node is the number of edges incident to the node,
- The torsion of nodes consists in the twisting of sides of the meshes adopted due to an applied torque: the tendency of a force to rotate an object about an axis.

Triangular meshes

Triangular meshes are most accurate in approximating a freeform surface than any other mesh. They are also inherently stiff. But they produce a high complexity of nodes due to valence of six. This means that every node of a triangle mesh has six edges merging into it. Which does not leave enough degree of freedom available for nodes to become torsion free. Torsion in nodes means when central plane of beams are not co-axial. Torsion free nodes in a triangular mesh can be realised only if the shape is optimized or is close to a sphere. Manufacturing of nodes is complex task, which is caused by the phenomenon that the symmetry planes of beams are connected to one node do not intersect nicely in a common node axis. Quadrangular meshes are more often used for free form shapes these days to avoid complexity of nodes as seen in YAS hotel (Wallner & Pottmann, 2011).

Quadrangular Meshes

Quadrangular meshes are not as stiff as compared to triangular meshes and need optimisation for planarization if required. Flat panels are easy and cheaper to produce, and single curvature cold bent panels are also feasible option. They are easier to produce, lighter and possible to optimise for minimum torsion in the nodes. (Wallner & Pottmann, 2011).

Hexagonal Meshes

Hexagonal meshes exhibit a low overall stiffness compared to an equivalent triangular grid. They have only three members connected to a node which makes their production easier (Andreussi, 2016). They are often related to organic forms occurring in nature and in some cases preferred for aesthetic qualities.
2.6 State of the art in practice

2.6.1 British museum great court roof (2001)

The steel and glass British Museum great court roof designed by the architects, Foster and Partners and the engineers, Buro Happold and was fabricated and erected by Waagner Biro. It covers a rectangular area of 73 by 97 metres containing a reading room of 44 metres diameter. A triangulated grid is used for structural stiffness and to have flat glazing panels to fit over each triangle of the grid. The roof had to supported by a ring of 20 columns around the Reading room and the rectangular boundary of the courtyard but without putting lateral thrust on the existing museum building. It sits on sliding bearings on the boundary and can only push outwards at the corners where the edge beams support it. Because of this reason, the higher curvature can be seen in the corners. I would like to direct the readers towards (Williams, 2001) for mathematical functions defining the surface geometry of the grid shell. Over 3000 line of computer code was specially written for geometry definition and structural analysis of this project. These constraints show that every project has different boundary conditions and there is need to develop a robust tool which can provide a feasible solution considering all variables input by the user.

The structural grid was started with a simple geometric drawing in which equally spaced points on the on the circular reading room circumference are connected to equally spaced points on the rectangular courtyard boundary by radial lines. These radial lines are divided into equal segments to form a grid. This mathematical grid was used to produce the structural grid by connecting ‘joining the dots’. However, this process did not produce continuous grid specially at the diagonals as seen in right hand half of figure 7(Williams, 2001).

To resolve this issue, force was applied on each point on the grid which was calculated by adding 4 force vectors derived from 4 neighbouring grid points. The point was allowed to slide over the surface defining the shape until the component of the force tangential to the surface applied on the point becomes zero. However, a variable was introduced in the equation to control the, maximum size of the glass triangles. Dynamic relaxation was used to converge the final structural grid. For the functions and algorithms used refer (Williams, 2001).

---

Figure 2.10: Final surface Source: (Williams, 2001)

Figure 2.11: (Left) Starting grid ( ) Relaxed grid

Figure 2.12: The Great Court of the British Museum, Source: http://www.galeriemagazine.com/ultimate-art-insiders-guide-london/
Construction

5162 steel box beams that intersect at 1862 structural six valence nodes, each one different and manufactured to precise tolerance were used to construct the shell. The roof contains 3312 unique triangle double glazing panels. Both of these combined weighted 800 tonnes which was carried by temporary props. As complex and susceptible to misalignment the roof was, there was no room for manual errors. All production was automated and carried out by computer aided manufacturing. The node is cut out of a thick plate as a star with 6 legs perpendicular to the plate surface by flame cutting using computer generated drawings. A robot arm was used to cut all steel box sections as each and every end was different (Sischka, 2000).

As the roof was building, the increase in the weight of the steelwork caused deflection in the structure carrying it and position of every node had to be examined regularly and pushed back into place. The tolerance for each structural node was 3mm and single mistake placing the cross member could cause the roof to fail due to misalignment as explained by (Brown, 2000) a partner in structural engineering Buro Happold. The roof was deliberately constructed slightly higher to allow for this deflection which happened after planned sequential removal of props supporting the roof (Sischka, 2000).

2.6.2 Yas hotel

The grid shell of Yas Marina Hotel in Abu Dhabi was designed by Asymptote Architects, New York and was completed in November 2009. Wagner Biro fabricated the grid shell and Schlaich Bergermann was the structural engineers and grid shell consultants. Tessellation optimization was done by Evolute consultants, Front Inc were the façade consultants and Gehry Technologies Inc managed the integrated delivery of project.

The geometry of the grid shell is a doubly curved surface covering two hotel blocks, 217-meter expanse. It is constructed out of steel sections and about 5800 mobile diamond shapes glass panels. The surface was generated by the architects, and the wireframe model of the grid was generated and optimised by Evolute consultants in collaboration with the architects. The process included subdivision then switching to diagonal grid followed by hand tuning for homogenisation, torsion free nodes, etc. (Figure 11). The hand tuning includes avoiding or hiding singularities on top of the surface, making grid line co planar, closeness to reference surface, etc. The tool used for optimizing the grid is now publicly available as a plugin for rhino which contains other optimization tools like fairness springs, fairness curvature, equality of edges or panels, planarity of panels, etc.

Figure 2.13: (Left) Computer rendering of extreme node with smallest angle between arms, 26.66°, (Right) Cutting of nodes out of steel plate 180 mm with autogenous cutting, Waagner Biro AG 1999 (Sischka, 2000).

Figure 2.14: Yas Hotel, Abu Dhabi, Source: https://www.asymptote.net/yas-slide-show
The grid shell unlike the British museum exhibits a quadrangular mesh with non-planar faces which are not covered by glass in water-tight way but fixed to the structural grid by pivoting hinges as seen in figure 2.15.

2.7 Statics of Grid shells

State of stress can be described at each point in a structural body by two normal forces and a shear force component considering infinitesimal cubical element at one plane. If the element is rotated or the plane is changed, the state of stress still remains same, but the stress components adjust themselves. There is one plane where normal stresses are maximum and shear stress component becomes zero, the direction of the normal forces at this plane are called principal stress directions. These two principal stress directions are always perpendicular to each other. A common graphical method to calculate principal stresses is Mohr’s Circle.

When principal stress directions are calculated at sufficient number of points in a structural body, a conjugate vector field can be created. Principal stress line can be generated by tracing a particle which moves through this vector field. These lines indicate the natural load path or flow of forces through the structural body of an applied force.

Principal stress directions are not affected by change in material stiffness or change in magnitude of applied forces. They are dependent on the direction of forces acting on the structure, boundary condition such as supports and geometry. Usually these conditions are fixed except the case of very high wind loads or lateral loads in case of earthquakes. Due to their consistency and capacity to suggest flow of forces, principal stress lines can be used for finding optimal topology for grid shells.

Strength of a grid shell is dependent on how the load is distributed among it beams. Instinctively, one can say that the structure would be stronger if the load is distributed uniformly. Moreover, if the beams are aligned in the direction of principal stresses, more forces will act axially on them. This results in reduced bending moments in beams. Thereby reducing strain energy on the structure and shear in each node. However, loads vary for each point on the structure and these variation increases with complexity of the shape. Following the direction of principal stress at each point can result in a non-uniform mesh with singularities. Hence there is also a need to homogenise the grid for better aesthetics and easier manufacturing and assembly.
2.8 Precedent work

Schiftner & Balzer (2010) proposed a method for statics-aware-initialization procedure for the layout of Planar quadrangular meshes approximating a given free form surface. Focussing on surface structure which show membrane like load bearing- behaviour, they generated a quad mesh where one set of lines are aligned to integral lines of vector field derived from maximal principal strain in every point. Figure 2.17 shows how this technique helps in stiffening the grid with alignment. Based on compliance (strain energy) analysis, they concluded that this method has favourable impact on the mechanical properties of the final PQ mesh.

In figure 2.18 (left), it can be seen that overlaid level set curves (black) have emerged from a trade-off between optimization for alignment with integral of guidance vector field as well as equal spacing. In figure (right) Transverse level set (curves of constant function value) curves have been optimised to become conjugate to the aligned level set curves as well as equally spaced.

A major drawback in this approach is that integral of guidance vector field they derived is not tangential to the vector field at every point. The function they used to derive the integral line moves towards singularity when the vector field changes direction. If they had used the vector field to create curves which are always tangential to the vector field, the resulting curves would depict the flow of forces better and it would be possible to follow alignment and keep the curves equally spaced. This is shown in figure 2.19.

Pietroni, Tonelli, Puppo, Froli, Scopigno & Cignoni (2015) introduce a framework for generation of polygonal grid shell whose topology is designed to excel in static performances. They achieve this by analyzing the input surface and inducing a non-Euclidean metric over it using the resulting tensor field. Varying the density and anisotropy of existing polygon meshes approximating the input surface gives a better distribution of strain energy over their elements. Meshes are further optimized for symmetry and regularity of cells to improve aesthetics. As this method relies on already exiting topology of mesh, it cannot perfectly align the grid with principle stress direction. Hence, the results don’t guarantee stiff and torsion free quadrilateral frame. In figure 2.20 we can see this method applied to various existing meshes. As we can see in first two meshes from left (quad meshes), the direction of members is highly dependent on the base mesh and does not guarantee alignment with the principal stress directions.
Tam developed a method in his thesis (2014) for generation of quadrangular topology based on generating streamlines across shell surface in 3D influenced by conjugate vector field derived from principal stress directions. He experimented with different methods to select streamlines based on angle conformity or strain energy to optimise spatial and structural objectives. These methods showed 20-40% improvement in performance on the basis of strain energy for early course of selection iterations but went on to become worse after more iterations. The results also required some manual editing for cleanup.

Cobb (2018) also attempted to develop quadrangular gridshells using similar methods. In case of method used in Cobb (2018), the mesh was flattened, a grid was generated and then mapped back to the original surface. This resulted in exaggerated spacing between curves where the surface is near vertical, it also required some manual editing for cleanup, also the mesh was not homogenous enough to be suitable for construction. Both of these methods did not claim to have achieved a desired result for parametric process to create grid shell suitable for construction but did lay ground for future research.

Looking at precedent work was highly motivating to pursue this research. Learnings from each research helped to frame the objective functions and methodology for this study. Schiftner & Balzer (2010) showed that a trade-off between objective functions of alignment with guidance vector field and equal curve spacing can give results which are aesthetically better and feasible for construction. Pietroni, et al. (2015) showed the importance of user control over the strength of objective function and having a parameter to do so can generate various option for user to choose from. Research by Tam (2014) and Cobb (2018) gave a promising method for extracting curves which are always tangential to the guidance vector field.
2.9 Design exploration

The role of optimization in design is to find the optimal solution or a front of best solutions according to the design criteria. This is one of the major potentials of optimization techniques, but exploration of sub optimal design solution has potential for supporting learning process of the designer and dealing with ill-defined criteria like aesthetics, constructability which are significant part of architectural design process (Turrin, et al., 2016). Moreover, search for optimality is a complicated matter considering the fact that uncertainties are often present in the real-world system, as explained by Yang & Koziel (2011). For example, material properties like young’s modulus and strength show inhomogeneous variations. The chosen solution will be affected by the non-availability of standard materials. Therefore, we look beyond optimal solution and seek robust or suboptimal solutions which are practical in reality. Many current researches tend to pay more attention to obtaining the pareto front and neglect post processing and interpretation of optimization results.

According to Yang, Sun, Di Stefano & Turrin (2017), “design exploration refers to the process of extracting information or knowledge and applying it to formulate or re-formulate a design concept”. They explore information or knowledge hidden behind obtained data sets and how this can be used to support human decision making on ideation. Some of the methods propose solutions by focusing on evolving design objectives, variables and constraints and giving importance to sub-optimal solutions.

Von Buelow (2012) proposed a tool Paragen which explores parametric geometry based on performance as well as visual criteria. Within its cyclic structure it incorporates “parametric geometry generation, simulation for performance evaluation, and the ability to sort and compare a wide range of solutions based on single or multiple objectives”. It facilitates human interaction by providing a graphic user interface to compare geometric alternative visually and giving option of parent selection and breeding. Therefore, the designer is able to analyse both the numeric data from the performance evaluations and visualize the 3D geometry being able to focus the generation towards other sub-optimal solutions that better meet other criteria such as personal preferences.

The paragen cycle follows five basic steps(figure):
1. Select - either with human interaction or by GA
2. Breed - GA program on a web server
3. Generate - the geometry using parametric software
4. Evaluate - with simulation software, e.g. FEA
5. Store - store the results in a database for selection – on web server

Van Kastel (2018) proposes a visual analytical system that visualizes geometries and performances of large data set of design alternatives in a highly interactive data environment, in a single viewport. The data environment integrates data analytics charts like self-organizing maps, hierarchical clustering, stacked bar graph, a decision tree and pictograms charts and presents it in a game-like 3-d landscape. User can navigate through the landscape to analyse hard (quantifiable performance) as well as soft criteria (aesthetics, looking at geometry). The SOM is the base of the landscape where design alternative geometries are assigned to corresponding cell. High performing solutions are higher in the landscape indicated by hilltops and low performing solutions are indicated by valleys like a stacked bar graph. Overall it is an effective system to find desirable design solution but interrelationships between design variable is not always clear.
3 Prerequisites for Workflow

3.1 Parametric modelling

Parametric modelling is the process of making a geometric representation of a design with editable attributes and geometric entities associated with each other. Attributes can be values input in the model, variation of which can give different solution of the model while respecting the pre-established relationships among geometric entities (Turrin, Yang, D'Aquilio, Sileryte, & Sun, 2016). It is rather important to select the variables according to the exploration goal targeted. The identification and analysis of the exploration goal is crucial for the whole process and should be prioritized before the parameterization process because the entire solution space is dependent on it. Logically generating design alternatives facilitates quick visualization of various options and sometimes new unthought of design directions. This helps the designer to evaluate aesthetic as well as performance criteria based on simulations software or finite element analysis in case of this research. Design alternatives can then be processed for performance in respect to certain design requirements. Performance indicators are required to assess how the design alternative is performing. While certain design requirements can be expressed based on indicators, for example maximum deflection of the shell. Other design requirements are subjective or based on personal preferences which needs to be inserted in the workflow by other means. For every architectural project some performance indicator might be important than other. Hence it is important to set the right combination of indicators before simulation.

Simulation programs are usually used to assess the building energy demand, costs, daylight, CFD or structural analysis. In this research, finite element analysis will be used for calculating structural performance. While structural performance is the most important factor, some other performance criteria are required to judge the feasibilty and cost of the project. These criteria include planarity of panels and amount of material and number of elements. This does not require any simulation engines.

3.1.1 Objectives

The objective of the workflow is to help user to derive a quadrangular grid shell based on principal stress directions. There are also sub objectives like structural stability, homogeneity and planarity. While some criteria like structural stability and planarity can be measure with performance indicators like buckling load factor, utilization, deflection and planarity deviation. Homogeneity has to be measured from visualisation and human input. The decisions are dependent on various factors like personal preferences, cost, material used, time availability, location, etc. In this thesis, an extensive cost analysis has not been performed. But some performance indicators are given which can be used calculate cost like amount of material used, total number of elements and nodes, number of unique elements and nodes, size and planarity deviation. These performance indicators have been explained in section 3.1.3

3.1.2 Variables

Density of grid

Panel size affects the cost as the denser it will be, the no of nodes and members increase which increases the construction cost. Also, if the grid shell has to be cladded with panels, depending on the material, the size of panel will affect the cost as well. The transparency of the grid shell will also depend on the density of grid.

While the exact size of panels cannot be controlled, but density, the factor which controls the panels size can be controlled.
Alignment with flow of forces

While alignment of member with principal stress direction is the core of the workflow, it should be a choice of user that how much alignment is suitable. Theoretically, the more members are aligned with the principal stress direction, less stresses will be induced in the nodes and stiffer the grid shell will be.

Planarity Deviation

Grid shells can be covered with panels for protection against natural elements. These panels are required to be in single plane or curved in on direction in order to be manufactured easily or to reduce costs. The deviation from planarity is calculated by generating a plane from three points in the quadrilateral and calculation the normal distance from the fourth point to the plane. In most cases making faces planar will affect the structural performance of the shell negatively, we have conflicting objectives. The decision depends on the user, if curving the cladding panels is costlier than increasing member cross section to make the structure stable or if curved glass panels is causing unwanted optical distortion, then planarity would be an important factor.

Beam cross section size

While beam cross section should be optimised to be minimum provided structural stability. This has to be optimised for each design alternative to compare amount of material required. Even if a common cross section size is used to compare every design alternative, after choosing a grid design, the cross section has to be optimised.

3.1.3 Performance indicators

• **Buckling load factor** - it is the factor by which the current load has to increase to cause buckling. For safety in grid shells, this factor is taken as more than 1.

• **Maximum Global Deflection** - the maximum displacement of elements or nodes caused due to loads.

• **Maximum panel size** - The maximum panel size for cladding is important as some materials are either expensive or not available in larger sizes or difficult to handle.

• **Maximum planarity deviation** - Bending cladding material can be expensive after a certain limit. So, the maximum deviation from planarity becomes important.

• **Utilization of beams** - Utilization means ratio of stress induced in the element to yield stress.

• **Strain Energy/Compliance** - Total Strain energy is used to check the stiffness of a structure. It is also called as deformation energy. This can be used to compare the stiffness of grid shells if they have same mass.

• **Number of nodes** – Nodes are referred to the connection of beams in grid shells. Increase in number of nodes will increase manufacturing costs as well as labour cost.

• **Number of beams and number of unique beams** – Number of beams is important to calculate costs and if there are more unique beams, the costs will go even higher.

• **Total mass of structure** – Mass of the structure provide the amount of material used for making grid shell.

3.1.4 Constraints

Constraints will vary for every project. The user will have to choose solutions which are within the limit of defined constraints manually in sequential strategy or it can be automatically done by the algorithm in integrated strategy. The following constraints can be input by the user:

• Maximum Buckling Load Factor
• Maximum Global Deflection
• Maximum Beam length
• Maximum planarity deviation
• Utilization of beams
3.2 Tools Definition

The tool selection affects the entire workflow definition, in terms of time consumption, computational power and compatibility with other available tools. This particular workflow required parametric modelling. Rhino-Grasshopper environment or Revit-Dynamo are two common platforms used for parametric modelling. In academic institutions, Grasshopper is widely used and there is always support from peers in case needed. The other important aspect required in this research was Finite Element Analysis for which there are multiple commercial tools available. For example, Diana FEA, ANSYS FEM as external software and Karamba (Preisinger & Heimrath, 2014) in the Grasshopper environment. Though external software is said to be more robust and reliable by other users, there are advantages of keeping the workflow in the Grasshopper environment to make the workflow automated and fast. Kangaroo is a physics engine used for form finding surfaces with in-plane stresses and dynamic simulation for various purposes multiple times throughout the workflow. Weaverbird is a plugin for Grasshopper which was used for dealing with meshes. Wherever required custom tools were made using GHPython component in Grasshopper several times in the workflow. Tools developed by TT toolbox were used for data managing and design exploration. Colibri was used for data recording and exporting into spreadsheets and images. Google drive was used to store data and finally Design Explorer was used for exploring different design alternatives as it is easy to use, freely available and has a web interface which can be used by anyone in the team or clients.

3.3 Workflow scope

Stress lines can be used to generate topology of any structural element. In this thesis, the focus lies on form found 2.5D membrane structure which are subjected to mostly in-plane stresses. To define the workflow, certain types of shells were considered which defines the scope of structures the workflow could be applied to.

The workflow was generated keeping in mind that it has to work with all types of 2.5D shells and the implementation chapter shows how the workflow was developed through few different types of examples. Multiple examples are taken because the definitions change according to different shapes, there are many conditions and exception introduced in the workflow to accommodate as many types of shells as possible.

3.4 Workflow strategy

Before starting to define a workflow, it is required to understand how to cater to soft criteria from the perspective of architect and hard criteria from the perspective of engineer. Optimisation of hard criteria is important in the concept stage of every design as it has the most effect on the life cycle cost of a project. But it is always possible to miss out on many possible sub optimal solutions. Sometimes clients and architects prefer solution according to their aesthetic experience even if it costs more.

As the design space for this particular problem is not very big, optimisation engines will not be used. Instead all design alternative will be stored. This means the user can browse through all alternative options. Visualisation of data set might provide more insight and might lead to redefinition of design objectives and variables. It is important to limit the number design alternatives to save time in generation and selection of design options.

While the workflow can be used in any way suitable for the user, there are two proposed strategies in this thesis and other is the integrated strategy. In sequential strategy, the workflow will be divided into several steps where design alternatives are evaluated, and a decision is made. The solution chosen at one step will serve as the input of next. In every complex problem there can be multiple objectives and variables. Some variable effects a particular objective and some don’t. At each step, change in one variable will generate multiple design options. The user can judge which option to choose by exploring the design space and evaluating performance indicators.

In integrated strategy, the generation of all design alternatives can be automated, and no human interaction is required while the simulation runs. All possible design alternatives are evaluated at the end of the simulation. Due to large number of options, it gets difficult to browse through each solution. For making informed decisions, design exploration tools are required as discussed before.
Figure 3.2: Sequential strategy (left), Integrated strategy (right)
4. Workflow definition

Once the objectives, variables, constraints, tools, scope and strategy were established on a conceptual level, next step was to develop algorithms to achieve these objectives. To develop and test algorithm, 2.5 d surfaces were required. Developing algorithms were treated as a design problem from the prospective of an architect and engineer. A typical shell with 4 supports were taken initially and stress lines were extracted using Karamba. Then a grid was hand drawn on the shell as an architect would design considering alignment and homogeneity as seen in figure below. This was an attempt to visualize what the algorithm has to do. This step helps formulating logic for algorithms.

![Figure 4.1 Hand drawn grid following principal stress lines.](image)

It is not necessary that if an algorithm works on one shell, it will work on all shells. This is because these are very complex problems and requires a lot of rules and exceptions. Some rules and exceptions can be foreseen but many are discovered while developing the algorithm. To make these algorithms robust, they were tried on various shells and modified accordingly. Two shells, Shell A and Shell B, with very different boundary conditions have been used to demonstrate why certain rules and exception were included in the process. Each algorithm has been described in the following sections, more or less in a chronological order they are used in the workflow. Each grasshopper script has been shown with explanation of each step. Some steps required to do things which were not available as tool or plugin in grasshopper. Then custom tools were developed using the coding language Python. The code for each of the custom tools can be found in the Appendix1.
4.1 Form finding

A mesh is required for form finding using Kangaroo. There can be several ways to define a mesh for a given closed polyline. The form finding method will work on any mesh, but due to characteristics of the method used, it gives better results in terms of membrane action if the mesh is fairly uniform. This is done by using Karamba’s MeshBrep component demonstrated on Shell A. This method produced triangular mesh. Another method is demonstrated on Shell B to develop a quadrangular mesh by dividing the boundary into 4 parts and then using weaverbird’s (grasshopper plugin) constant quad subdivision tool while keeping the support corners fixed and smoothing the edges. This method always produces quadrangular meshes and can only be used with line-like curve.

Particle spring method was used to form find the shell. A physics engine known as Kangaroo was used to implement the particle spring method. This method is explained in the section 2.3.7 In this method a uniform force is applied on the mesh to mimic a shell of constant thickness under self-weight. In kangaroo force is applied on points. As every face of the mesh is not equal in area, the density of points varies across the mesh. If equal force is applied on every vertex of mesh, it will result in uneven force distribution. To apply a uniform force over the entire surface, a different value of force is calculated for every vertex. This is done by taking the average area of four faces shared by a vertex and then applying a unit force in positive z axis multiplied by the calculated area. Each line in the mesh is considered a spring and the strength given to the spring is decided on the basis of minimum height required for the shell for buses to pass. If the spring strength is higher, they will stretch/expand less and therefore the effective height of the shell will be less. The spring strength was increased until a desired height was reached.

A disadvantage in this method of form finding is that after relaxation the springs which should have been smaller in length than original resist the change in length due to their stiffness. So, they start to change their orientation in out of plane direction so they can retain their length which results in an uneven surface. To smoothen the mesh Laplacian smoothing algorithm was used. In this method each vertex is moved to a new position chosen based on the mean value calculated by the position of neighbouring vertex. A tool in Weaverbird plugin was used to apply this algorithm.
4.2 Finite element analysis for vector field

Finite element analysis is performed in karamba3d on the shell obtained from kangaroo simulation. A gravity load is applied on the shell which will give similar results for flow of force as applying a uniform force which was used for form finding. The points which were set as supports are also set as supports for form finding process. Principal stresses direction and magnitude are calculated on two points on each face of the mesh obtained from parallel shell analysis done in Karamba3d. The vectors at these points becomes the guidance vector field for generating stress lines.

Karamba (Preisinger & Heimrath, 2014) has an inbuilt stress line generator component. But this component is designed to visualize the flow of forces and is not useful for generating a topology suitable for grid shell. There are methods like Euler’s method & Runge-Kutta algorithm available in grasshopper to trace a particle in a vector field. In every method a starting point or seed point is required to start tracing or generating a curve. Most tracers are based on a simple logic that seed point has to move a small distance (step size) in a certain direction which is influenced by the vector field until it reaches the boundary condition. All methods vary in calculating the direction in which the point moves, and some are more accurate than the others. The methods would work in the absence of singularities in the vector field and no ring forces in shell structures. In a vector field, singularities are the positions on which a particle tracer can go in multiple directions. This would result in different stress lines intersecting near this point which is not desirable output. In case of ring forces in shells, stress lines will make a loop and intersect with itself at a certain point and still continue its path which is also undesirable. Hence, there is a requirement to control the outcome of generation of streamlines according to the criteria which can lead to suitable topology generation. For this reason, a custom stress line generator was developed.

Figure 4.5: Flow Chart for finite element analysis and input for Custom Stress Line generated. For GH Script refer Appendix 9.3-9.7.

Figure 4.6: Conjugate Vector Field derived from principal stress 1 (blue) and principal stress 2 (Red) for Shell A (left) and Shell B (right)

Figure 4.7: Visualisation of Stress lines from Karamba for Shell A (left) and Shell B (right)
4.3 Custom Stress line generator

For the custom stress line generator, methodologies from Tam (2015) and Cobb (2018) were referred. Tam (2015) takes an average of all vector inside the step radius to calculate the next step direction. Cobb (2018) uses gaussian function to give weightage to all vectors that define the new step direction. These tools had certain exceptions like looping, enforce offsets between stress lines, etc.

These methods were not used because they did not produce the desired result required to achieve the goals of this research. Also, not enough information was provided in the papers to be able to use these methods. A mixture of these methods was used to calculate the step direction with certain exceptions and changes. The result can only be controlled by adding more rules to the algorithm. The new custom stress line generator was developed with more rules and parameters which can be controlled by the user, depending on the type and size of the mesh and also depending on the desired typology. The code for this tool was written in python from scratch. The code can be found in Appendix 10.3.

In the beginning of algorithm each vector in the vector field is unitized so the magnitude of stress at a position doesn’t affect the tracer. In order to determine the step direction, the last point in the stress line searches for vectors which are within a given radius. These vectors will influence the step direction in a way that the vector which is closest to the last point on the point trail gets more weightage. This radius size should be chosen by the user or can be calculated according to the density of the vector field because if the radius is too small, there will be no vectors inside it. If the radius is too big it will just increase the computational time required to calculate the step direction with negligible influence on the final direction. Each vector in the radius is multiplied with the calculated weight and sum of all these vectors is unitized to get the final direction of next step. The weight given to each vector is calculated based on the following function:

\[
\text{Weight} = \left( \frac{1}{\text{distance}} \right)^{\sigma}
\]

Where:

- **Weight**: weightage given to each vector within the given radius
- **distance**: distance of seed point or last point on the point trail to each point in the vector field
- **\(\sigma\)**: variable

\(\sigma\) is a variable which the user can change or by default it is taken as 1. If \(\sigma\) is zero then weight is 1, that means equal weightage is given to every vector. As you increase the value of \(\sigma\), the vector which is closer (distance is smaller) gets higher weightage in influencing the next step direction.
4.3.1 Direction correction

Before calculating the step direction, it is required to make correction in the direction of vectors inside the radius. Also stress line needs to go in two directions from the seeding point, in one of the two trails vectors will always be in opposite direction. In order to correct the direction, the angle of the vector is check with direction the trail is moving in, if the angle is greater than 90 degree, its direction is reversed. When the direction is determined the last point moves certain distance in that direction. This distance can be input by the user as step size and in this case, it was taken as 0.6m. to make sure the point trail stays on the mesh, after each step the new point is replaced by the closest point on the mesh.

4.3.2 Boundary detection

Once the algorithm is running and stress line is generated step by step, it needs to stop when the it reaches the boundary of the mesh or when the stress lines completes a loop and intersect with itself. In the case when streamlines crosses the naked edge of the mesh, the tracer stops. Now to place the last point exactly on the naked edge, intersection point is checked with high tolerance. The high tolerance is required because the streamline and naked edge never intersects as they or not on the same plane.

4.3.3 Rule based corrections

Step maximum angle
Every time a new direction is calculated for next step in the tracer, it checks the angle if with the existing direction of tracer. If this angle is more than input maximum angle set by the user, the direction is automatically dialed back to the maximum angle allowed.

Looping
In case of ring forces in shell as seen in Shell A, the stress lines intersect with itself or comes too close to itself, the tracer stops. But the tracer runs in two directions form the seed point, so there are four possibilities of intersection. When first tracer intersects with itself, when first tracer intersects with second tracer and vice-versa. In the case of one of the tracers intersecting with itself, second tracer points are removed completely and the points of the first tracer before the intersection points are removed and the curve is closed. In the case of one tracer intersecting with other, the second tracers point after the intersection points are removed and the curve is closed. There is also an option for user to not apply this rule according to the desired topology.
Intersection
There is a high possibility of stress lines converging at singularities. At every step of tracer, there is check for intersection with another stress line. If intersection happens, a new direction is searched for that step. It checks for every direction it can go into, in an interval of 5 degrees in a range of 180 degrees, unless there is no intersection. In the case where there is no way to avoid intersection, for example when the current stress line is between two already exiting converging stress lines, the original path is continued. All the stress lines which intersect with another stress lines are removed. There is an option to keep the intersecting lines, which is controlled by the user. It is advantageous to avoid intersection because when these curves become structure members of a grid shell, the point of intersection will become a joint. These joints will have a valence of more than 4 and high torsion in nodes. Also, the acute angle of intersection will create joints more difficult to design and manufacture. There is also no way of predicting the topology in the case of intersection as there will be many triangles created instead of quadrilaterals and increase the post processing burden.

Non-Continuous load paths
Due to the logic used for this tool and other exception introduced for rule based corrections, it is possible that some stress lines are left without ending at a boundary or without looping. These stress lines are removed automatically.

Opposite boundary ends
This is an optional correction. In some case like Shell B, it might not be desired that principal stress line 1 (blue) ends at support edges as it creates triangles in the grid. Same goes for principal stress lines 2 (red) ending at free edges. In cases like shell A, this is not avoidable.

Seeding
It is essential for further steps in the method to work properly that the stress lines cover the entire surface more or less uniformly. To ensure that, the seed points are selected uniformly all over the mesh surface. There are instances that a seed point is very close to existing stress line, in that case this seed point is skipped to avoid stress lines which are very close to each other. The algorithm keeps generating stress lines until the maximum density is reached according to the set parameter for allowed min distance between lines.

Figure 4.13: Intersection correction

Figure 4.14: Seeding correction

Single boundary ends
This is an optional correction. In some case like Shell B, it might not be desired that the stress lines end at a single support edge. This can happen if a stress line changes direction at singularities when trying to avoid intersection with other stress lines. In cases like shell A, this is allowed to happen.

Figure 4.15: Non-Continuous load path correction

Figure 4.16: Opposite boundary ends correction

Figure 4.17: Single boundary ends correction
4.4 Varying Density

Another algorithm is used to reducing the density of curves on the mesh. This is done by removing the curves whose mid-point is closer than the distance (set by the user). If the required density is not known to user, for exploration of various design options, multiple options can be generated automatically. GH Script can be found in Appendix 9.9.

4.5 Converting stress lines to a grid

The output from the stress line generator before it goes into post processing has to be converted into polylines with line segment instead of polycurves. Also, in practice, most grid shells are made out of straight elements. This is done by finding the intersection points between all generated curves and boundary curves as well and then joining the intersection points along existing polycurves. GH script can be found in Appendix 9.10.

4.6 Subdividing polygons to quadrilaterals

In the streamline generation process, the topology for further steps is fixed. If there are polygons which have more than 4 sides, then these polygons are subdivided into quadrilaterals. This is done because, after homogenisation these polygons look odd and are bigger than other quadrilaterals. In case of hexagons, this can be done automatically by the script below, which connects two opposite nodes with a line segment to create two quadrilaterals. Shells which are supported with 2, 3 or 4 supports mostly have hexagons and quadrilaterals. For other types of shells like Shell A, either the designer can manually edit it lines or leave it as such until a final design is reached and then edit manually or do nothing. It is the designer’s preference. Using the script below, polygons with more than 4 sides can easily be identified as shown in the figure below. For Shell A, some manual editing was done at this step and for Shell B, the script below was used to automatically subdivide. GH Script can be found in Appendix 9.11.
4.7 Homogenizing grid

There are many advantages for homogenizing the existing grid. The initial form finding of the shell structure was based on equal distribution of loads so if the loads are still more or less equally distributed, the stress are more likely to stay in plane. Considering all members are designed to be of same cross section. Making members equally spaced can reduce the cross section of beams as the largest span is reduced. Also overdesigning of very small beams can be avoided. Construction can become easier and more cost effective. It can also improve aesthetics of the grid shell.

This was done by dynamic relaxation using kangaroo2 plugin. The inspiration for using this method came from the design of British museum court roof but same technique would not work because the mesh has too much variation in density. There were three objective functions used after trying out a lot of options. Every line segment is treated as spring and its rest length is calculated by averaging out the length of lines close to it which are which are parallel or within a range of 30 degrees of angle with the given line segment. Figure 4.21 demonstrates the same.

The springs exert a force on other springs and the points move tangentially on the mesh surface until rest length of springs is reached or forces balances out each other. To make the points move tangentially on the mesh, an inbuilt objective function in kangaroo2 is added which keeps all points on the mesh. As the rest length of various springs will be more than their original length, there is a risk of wrinkling affect in the mesh. To avoid this, the rest length of every line was reduced by a factor. The points on the edges or mesh boundary were allowed to move along their respective edges using an objective function which keeps point on a given curve or in this case the given polycurve. An objective is added for connecting springs with mesh boundary to maintain their original direction to avoid parallel polycurves from merging with edge polycurves. The maximum and minimum length a spring could take was also determined and set.
4.8 Alignment with principal stress lines

In the process of homogenising the mesh, the alignment with principal stress direction is not an objective. So, some lines may not be aligned any more. This objective was not applied in the previous step because with each iteration the position of members changes and with that the target angle for alignment should also change. But with the previous step we already have a topology which can manipulated to achieve maximum alignment with guiding vector field. Now to align the members, another dynamic relaxation was setup with kangaroo. The main objective for this dynamic relaxation was to align every member of the existing homogenised grid to principal stress direction. The vector field derived by FEA was not used to calculate the target angle every line segment had to reach because, it would give the undesirable results at singularities. Instead the lines which were outcome of streamline generator was converted into vectors. Each line takes the average of group of vectors within a given radius as the target direction.

Along with the main objective some constraint objective were also used which keep all points on mesh and keep the original length of lines. The inbuilt component in kangaroo2 for alignment of springs with vectors was used. Like every other goal objects or components in kangaroo2, this alignment component also has a strength variable. Different values for this variable can give different results. In most case higher alignment gives better results structurally as expected. It is also true for some case that more alignment means less homogenous grid, which may or may not be desirable for some cases. So, the strength of the alignment factor will be taken as variable parameter which user can decide according to their preference.

4.9 Planarity

Planar or near planar faces of meshes are required for cladding most materials at minimum cost including glass. At low curvature glass can be cold bended at the site. But there is limit for deviation from planarity after which cold bended glass fluctuates with variable forces like wind or live load. After identifying this limit, a simulation needs to be run for making all quadrilaterals planar, under the maximum deviation allowed. There is an inbuilt feature in kangaroo2 plugin that tries to make each face in a mesh planar. To use this feature, we need to convert are output of lines to mesh. This is done by using weaverbird plugin. It was observed that the objective does not work efficiently if the constraint of keeping points on the mesh is stroner. This constraint is used but with lower strength. As we know that the structural efficiency of the grid shell depends on the form, after this step to achieve planar faces, the form deviates from the original form found shape. Due to change in this shape, some out of plane streses are introduced. Stronger the objective for planarity is, the structural performace of the grid shell in terms of buckling load factor and strain energy becomes worse. This objective may produce results which may or may not be desirable for each case. Hence the strength of objective of planarity is kept has a variable parameter adjustable by the user.

Figure 4.24: Flow Chart for Alignment and Planarity.
GH script can be found in Appendix 9.13 and 9.14.

Figure 4.25: Aligned grid

Figure 4.26: Planarity Visualisation
4.10 Structural analysis (FEA)

Karamba is used for structural analysis. Material is taken as steel from the material library of Karamba. The circular cross section is used with diameter of 35cm and thickness of 5 mm. Loads due to self-weight can be calculated by Karamba as gravity loads. Loads in case of cladding with glass panels is taken as 0.5kN/m2. Wind load is taken as 0.5 kN/m2 on projected geometry in negative x-direction. Snow load is taken as 0.7kN/m2. Safety factor on dead load is taken as 1.3 and live load is taken as 1.5. Supports are taken to be fixed in translation and rotations in all axes. Karamba has three algorithms for calculating deflection in a model. For grid shells AnalyzeTHll was chosen as it uses second order theory for small deflections. Karamba’s non-linear analysis component was not used as it is still a work in progress and might give errors while analysing. The output extracted from this analysis are utilization of beams (considering stresses induced axial forces, shear, bending moment and torsion), Bucking load factor, Mass of structure, strain energy and maximum global displacement.

Figure 4.27: Final Shell A after all steps

Figure 4.28: Final Shell B after all steps

Figure 4.29: Visualisation of Utilization (left), Visualisation of Deflection (right)
4.11 Generation of design alternatives and storing data

Colibri plugin from core studio is used for generation of design alternative and storing data. In Colibri the variables (iteration Genome) and parameters or results (iteration Phenome) are input. These inputs are mentioned in the section 3.2. All Data for each iteration is stored in a spreadsheet. Colibri is also able to capture images of viewports for every iteration it runs. To evaluate each design certain visualization are helpful like the utilization of beams displacement and planarity deviation. Each visualization is set in different viewports so it can be recorded. By using spectacle 3d geometry can also be recorded in Colibri. Spectacles is also part of TT toolbox developed CORE studio. All this recorded data can be viewed in Design Explorer.

Figure 4.30: Flow Chart for finite element analysis and generating design alternatives and storing data. Gh Script can be found in Appendix 9.15 - 9.18

4.12 Discussion

Unique number of elements

Grid shells are difficult to assemble because each beam can be of different lengths and each node can be unique. Construction can be made easier if it is possible to limit the number of unique elements. An algorithm was attempted which tried to limit the number of unique beams. The logic for this algorithm was to calculate the minimum and maximum length of beams in the existing grid. Then calculating 10 different values of length between the minimum and maximum. All lines in the mesh will have to adjust their length to get to one of the given 10 values. All lines or beams are considered as springs and given the rest length of one of the 10 values. Dynamic relaxation simulation was run using kangaroo. But some constraints were added like, the points on naked edges of the mesh can only move along them. All points have to stay on the mesh.

This simulation reduced the number of unique lines, but the lines did not remain in a smooth curve. It was tried to reduce the strength of constraint which keeps all points on the mesh, but this resulted in undulation on the surface of the mesh. Due to time constraints, this algorithm was not debugged but it is possible to make it work with different constraints. For example, a constraint which doesn’t allow the angle between intersecting lines to change by a few degrees. The constraint on boundary edges have to change in order to reduce the number of unique lines. This will affect the shape of the shell as well. If the designer allows the change in boundary conditions, this algorithm can be quite useful. Along with this, it can also be attempted to reduce the number of unique nodes.
4.13 Proposed workflow

The following workflow is proposed after combing all the algorithms developed. There can be several ways of using these algorithms but two strategies are proposed, Sequential and Integrated.

![Proposed Workflow Chart](image-url)

Figure 4.31: Proposed Workflow chart
5. Case study

A design application is required to study the application of workflow developed during the research. Delft Bus station was chosen as the site to make a steel grid shell covered with glass panels to protect domestic as well as international travellers from varying weather conditions of Netherlands.

Figure 5.1: Proposed shell at Delft Bus Station
5.1 Design Constraints

To begin using the defined method there are certain design constraints required to make decisions at different steps throughout the workflow.

Minimum buckling load factor - 4
Shells are very efficient structures because they are optimised to be as thin as possible. Shells takes most load by membrane action rather than bending which is dependent on their shape. In shells buckling is very critical because they collapse suddenly without showing much deformation. This happens because when it starts deforming, the natural load path goes out of plane and causes bending. For this reason, the buckling load factor is taken as 4 to be safe.

Maximum Global Deflection - Thickness of the shell or Span/250
As long as the shell doesn’t buckle, slight deformation is allowed. For initial stages of design as a rule of thumb, maximum deflection allowed can be taken as much as the thickness of the shell or span by 250, whichever is lower.

Maximum panel size - 2.4m
The maximum panel size depends on the availability, transportation, cost and workability and strength. Laminated glass is available up to 9m in length in Europe but as the glass panels are horizontally placed and will deform after exceeding a certain span. The thickness of the glass will have to be increased and it will become heavier. As the optimal glass panel size is in itself a long process to decide. For sake of demonstrating the workflow, maximum panel size of 2.4 m and thickness of 12mm (two layers of 6mm) laminated structural glass is taken.

Maximum planarity deviation - 101mm
In case of twisted glass panels, three points are in a plane and fourth point is moved to cold bend glass. In this type of bending, after certain deviation from planarity, glass panel starts to buckle, and the sides begin to curve (Staaks, 2003). As we are limited to only straight beams, this is not possible to do. According to Staaks, the limit to deviation from planarity for a square rectangular panel is 16.8 times the thickness of the panel, before it buckles. This is only dependent on thickness, and not on size of the panel or material. There is no comment on laminated glass panels, so we consider the thickness of single layer of glass which is 6mm in this case. The maximum deviation is 16.8 times 6mm which is 101 mm.

Maximum Utilization of beams - 100%
The beams are allowed to take as much stress until it reaches its yield strength. For this example, steel is taken as the material for beams and yield strength of steel is taken as 235 MPa.
Figure 5.2: Overview of Grasshopper definition of the workflow for the Case Study using integrated Approach
5.2 Initial steps

Figure 5.3: Workflow initial steps until homogenisation process
5.2.1 Defining boundary edges and mesh

The first step of the workflow is to define boundary edges and then a mesh which will be used for form finding. The boundary conditions mean defining the shape of the shell, the supports and free edges. The position of supports is influenced by constraints on site. The entry and exit points, vehicular and pedestrian movement and visual access to the station entry. The anchor edges were curved outwards to provide double curvature at the legs. Double curvature increases the stiffness of the shell. The free edges were curved inwards to reduce the extra material and increase the height of the openings to provide clearance for double decker buses. The maximum span of the structure is 65m. A mesh resolution of 1m was taken for smooth surfaces and dense enough vector field for accurately tracing principal stress lines.

5.2.2 Form Finding

Kangaroo was used for finding. This component requires two inputs, a flat mesh and support edges. According to mesh resolution and span of the shell, the strength of the springs is decided. The height of the shell will decrease with increase in strength. In this case it was important to check the height of openings for clear passage for double decker buses. The spring strength was taken as 350 after exploring some options. The minimum height of pedestrian entry came out to be 5.5m, vehicular entry as 7m and maximum height of the shell was 10m.

5.2.3 Finite element analysis

Finite element analysis was done using karamba. This is majorly done to extract principal stress vector field. The resolution for vector field was also set as 1m. Karamba’s stress line generator is used for visualising stress lines.
5.2.4 Custom stress line generator

The inputs for stress line generator as shown in figure. There were 300 seeding points taken. The step size and radius depends on the size and resolution of the mesh. After observing the stress line input from karamba, it was decided to not allow looping as it is not required. Also in this case, each edge is properly defined and no stress line has to return to same edge, SingleBoundaryEnds was set to false. It was seen that principal stress line 1 were ending at support edges, which leads to triangles in the mesh. As this was only happening near the supports and will not affect the shell structurally, oppositeBoundaryEnds was also set to False.

![Figure 5.7: Setting for Stress line generator](image)

The final result was quite without the unwanted intersections karamba produces and with well distributed quadrangular faces. The tool to create stress lines to a grid does not require any input and did the job correctly. The tool developed for subdividing polygon to quadrilaterals was used and no manual editing was required in this case.

![Figure 5.8: Principal stress 1 lines ending at support edges](image)

5.2.5 Homogenisation

The result of homogenisation was also satisfactory with the default setting of the algorithm. After this step user can choose how to approach the given problem.

![Figure 5.9: Output of Custom Stress line generator tool(left), Homogenised mesh(right)](image)
5.3 Sequential Approach

Figure 5.10: Workflow for sequential strategy
5.3.1 Density

Using colibri multiple option of densities were created and data was stored with performance indicators in the form of spreadsheet, images and 3d lines and mesh in .json format. This data was uploaded on design explorer through google drive.

The main points to note while deciding the density are maximum member length, number of nodes and beam members. Our goal is to keep no of nodes and members minimum because they lead to increase in costs. We had also set the limit for maximum beam length as 2.4 metres. The option with density 3 has maximum member length of 2 metres which is below the required limit with least number of nodes. Option with density 4 has a maximum member length of 2.5 metres. Considering the flexibility of tool to affect maximum member length, option with density 4 was chosen because the number of nodes and beams increase drastically with density 3.

The factor which governs the maximum length of any beam was by default set to 1.7 multiplied by the average length of all beams, this is changed to 1.5 and then homogenised again to get a maximum length of 2.36 metres.

The main points to note while deciding the density are maximum member length, number of nodes and beam members. Our goal is to keep no of nodes and members minimum because they lead to increase in costs. We had also set the limit for maximum beam length as 2.4 metres. The option with density 3 has maximum member length of 2 metres which is below the required limit with least number of nodes. Option with density 4 has a maximum member length of 2.5 metres. Considering the flexibility of tool to affect maximum member length, option with density 4 was chosen because the number of nodes and beams increase drastically with density 3.

The factor which governs the maximum length of any beam was by default set to 1.7 multiplied by the average length of all beams, this is changed to 1.5 and then homogenised again to get a maximum length of 2.36 metres.

### Figure 5.11: Design option with varying density

### Figure 5.12: Design explorer interface screenshot

### Figure 5.12: Performance indicators

The main points to note while deciding the density are maximum member length, number of nodes and beam members. Our goal is to keep no of nodes and members minimum because they lead to increase in costs. We had also set the limit for maximum beam length as 2.4 metres. The option with density 3 has maximum member length of 2 metres which is below the required limit with least number of nodes. Option with density 4 has a maximum member length of 2.5 metres. Considering the flexibility of tool to affect maximum member length, option with density 4 was chosen because the number of nodes and beams increase drastically with density 3.

The factor which governs the maximum length of any beam was by default set to 1.7 multiplied by the average length of all beams, this is changed to 1.5 and then homogenised again to get a maximum length of 2.36 metres.

<table>
<thead>
<tr>
<th>Density</th>
<th>max beam length[mm]</th>
<th>No. of nodes</th>
<th>No. of beams</th>
<th>No. of unique beams</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.70613</td>
<td>1978</td>
<td>599</td>
<td>131</td>
</tr>
<tr>
<td>2</td>
<td>1.907621</td>
<td>2302</td>
<td>4460</td>
<td>143</td>
</tr>
<tr>
<td>3</td>
<td>2.04667</td>
<td>1802</td>
<td>3477</td>
<td>161</td>
</tr>
<tr>
<td>4</td>
<td>2.506874</td>
<td>1180</td>
<td>2258</td>
<td>182</td>
</tr>
<tr>
<td>5</td>
<td>2.765148</td>
<td>1009</td>
<td>1925</td>
<td>189</td>
</tr>
<tr>
<td>6</td>
<td>3.118975</td>
<td>734</td>
<td>1388</td>
<td>207</td>
</tr>
<tr>
<td>7</td>
<td>3.466783</td>
<td>621</td>
<td>1360</td>
<td>205</td>
</tr>
<tr>
<td>8</td>
<td>3.773268</td>
<td>498</td>
<td>931</td>
<td>231</td>
</tr>
<tr>
<td>9</td>
<td>4.234506</td>
<td>413</td>
<td>766</td>
<td>213</td>
</tr>
<tr>
<td>10</td>
<td>4.909841</td>
<td>326</td>
<td>599</td>
<td>232</td>
</tr>
</tbody>
</table>

**Discussion**

If an optimization algorithm was used with a fitness function which excluded all design options with maximum beam length of more than 2.4 metres, it wouldn’t have been possible to consider option with density 4. But exploring the design space gives more insight and lets user go back and manipulate the design variables to create.
5.3.2 Alignment

After the decision on density, the next step is to decide the degree of alignment with principal stress lines. This step is required because in homogenisation processes, the direction of elements might get changed. From this step, the structural analysis is required for performance indication. For structural analysis, certain inputs are required. The rotations of supports are fixed in all axes. The material is chosen as steel, circular cross section with diameter as 15 cm and thickness as 1 cm. The beams which are at the edge and beams which are connected to the supports are bigger than the others because stresses are highest in these beams. They are taken as 30 cm in diameter and 1.5 cm thickness. This is done so that the high stresses can be read easily in the rest of the shell.

The decision for degree of alignment is based on structural and visual factors. It is noticeably visible that as you increase the alignment factor, the stresses in the middle area of the shell start to reduce as they get more aligned to principal stress directions. This is for the same reason that as they are aligned with the direction of force, there will be less stresses caused by bending moment.

But by looking at the numbers, the maximum utilization does not change significantly. This is because the maximum stresses are still occurring in beams which are close to supports. In compression structures, the supports are usually made thicker because they take the maximum load. If the members near the supports are made even thicker, then we will be able to see the reduction in total stresses which are caused by in-plane forces acting perpendicular to the direction of beam. Theoretically, it can be assumed that the option with the highest degree of alignment will have least stresses. We can also see that with increase in alignment some parts of the shell become less homogenous which affects the visual appearance. As a balance between structural performance and homogeneity, the option with an alignment factor of 255 was chosen.

<table>
<thead>
<tr>
<th>Alignment</th>
<th>Max beam length</th>
<th>Buckling load factor</th>
<th>Displacement (mm)</th>
<th>Max utilization</th>
<th>Mass (kg)</th>
<th>Strain energy kNm</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.36</td>
<td>5.31</td>
<td>58.3</td>
<td>56.1</td>
<td>122836</td>
<td>11.78</td>
</tr>
<tr>
<td>3</td>
<td>2.36</td>
<td>5.31</td>
<td>58.3</td>
<td>56.1</td>
<td>122836</td>
<td>11.78</td>
</tr>
<tr>
<td>15</td>
<td>2.36</td>
<td>5.31</td>
<td>58.3</td>
<td>56.0</td>
<td>122835</td>
<td>11.78</td>
</tr>
<tr>
<td>63</td>
<td>2.37</td>
<td>5.35</td>
<td>56.8</td>
<td>55.5</td>
<td>122764</td>
<td>11.22</td>
</tr>
<tr>
<td>255</td>
<td>2.38</td>
<td>5.39</td>
<td>56.8</td>
<td>59.4</td>
<td>122735</td>
<td>10.93</td>
</tr>
<tr>
<td>1023</td>
<td>2.41</td>
<td>5.44</td>
<td>56.8</td>
<td>64.6</td>
<td>122675</td>
<td>10.68</td>
</tr>
</tbody>
</table>

The decision for degree of alignment is based on structural and visual factors. It is noticeably visible that as you increase the alignment factor, the stresses in the middle area of the shell start to reduce as they get more aligned to principal stress directions. This is for the same reason that as they are aligned with the direction of force, there will be less stresses caused by bending moment.

But by looking at the numbers, the maximum utilization does not change significantly. This is because the maximum stresses are still occurring in beams which are close to supports. In compression structures, the supports are usually made thicker because they take the maximum load. If the members near the supports are made even thicker, then we will be able to see the reduction in total stresses which are caused by in-plane forces acting perpendicular to the direction of beam. Theoretically, it can be assumed that the option with the highest degree of alignment will have least stresses. We can also see that with increase in alignment some parts of the shell become less homogenous which affects the visual appearance. As a balance between structural performance and homogeneity, the option with an alignment factor of 255 was chosen.
5.3.3 Planarity

After alignment step, the next step is for making the faces as planar as possible and within the limit of maximum deviation from planarity mentioned before. The decision for planarity is based on deviation value and structural performance. With increase in the strength of planarity function, the deviation reduces, and structural performance becomes worse. This is because in attempt to make planar faces, the shape of the shell changes. With change in shape, the forces go out of plane and cause bending. In this case, no option goes above 101mm which is the maximum deviation allowed for 6mm glass. But its still easier to work with plane glass. As all other factors do not exceed their respective limits, the option with least deviation should be chosen. In case of planarity, it is important to have a look at the 3d model as it might affect the shape. Some undulations developed in the surface with the least deviation as shown in figure 5.19, so the second-best option was chosen.
5.3.4 Member diameter

For member diameter until now was taken as 15cm. It could not be assumed as the lowest possible size. To check if a smaller size is possible, different sizes were analysed. Going lower than 14 cm was possible in terms of displacement and utilization but it did not satisfy the condition of minimum buckling factor of 4. So, 14 cm was chosen as the beam cross-section diameter.

<table>
<thead>
<tr>
<th>Beam diameter (cm)</th>
<th>Flaw detection (cm²)</th>
<th>Max beam length (m)</th>
<th>Buckling load factor</th>
<th>Displacement (mm)</th>
<th>Max utilisation (%)</th>
<th>Mass (kg)</th>
<th>Strain energy (kJ/m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0.6</td>
<td>4.39</td>
<td>2.96</td>
<td>123.55</td>
<td>110.7</td>
<td>96480</td>
<td>17.30</td>
</tr>
<tr>
<td>13</td>
<td>0.6</td>
<td>4.39</td>
<td>3.00</td>
<td>91.62</td>
<td>95.7</td>
<td>105362</td>
<td>16.38</td>
</tr>
<tr>
<td>14</td>
<td>0.6</td>
<td>4.39</td>
<td>4.45</td>
<td>73.23</td>
<td>68.2</td>
<td>113991</td>
<td>12.46</td>
</tr>
<tr>
<td>15</td>
<td>0.6</td>
<td>4.39</td>
<td>5.36</td>
<td>57.28</td>
<td>57.4</td>
<td>122777</td>
<td>11.17</td>
</tr>
<tr>
<td>16</td>
<td>0.6</td>
<td>4.39</td>
<td>6.38</td>
<td>47.25</td>
<td>48.7</td>
<td>154459</td>
<td>10.24</td>
</tr>
<tr>
<td>17</td>
<td>0.6</td>
<td>4.39</td>
<td>7.00</td>
<td>38.77</td>
<td>41.9</td>
<td>140391</td>
<td>9.56</td>
</tr>
</tbody>
</table>
5.4 Integrated approach

Figure 5.22: Workflow with integrated approach
Structural performance, planarity, constructability, cost and visual appearance were the criteria to choose the best design amongst all generated options structural. With such number of options, it was not possible to look at all of them. Design explorer let you fix a range on any variables or performance criteria to filter the results. First step was to make a filter of constraints from the point of view of structural stability. Option with buckling load factor below 4, utilization above 100% were discarded. The max member length could be modified by some amount by changing the setting in the algorithm, so the option with max member length which exceeded the set limit of 2.4 metre by more than 5%, were discarded.

Variables set
Beam cross section diameter = 20, 21, 22, 23 cm
Planarity - 0, 99, 999, 9999
Alignment - 0, 3, 15, 63, 255, 1023
Grid Density - 3, 4, 5

A total of 270 options were created and recorded with the new set variable range in a span. The algorithm ran for 7 hours.

Structural performance, planarity, constructability, cost and visual appearance were the criteria to choose the best design amongst all generated options structural. With such number of options, it was not possible to look at all of them. Design explorer let you fix a range on any variables or performance criteria to filter the results. First step was to make a filter of constraints from the point of view of structural stability. Option with buckling load factor below 4, utilization above 100% were discarded. The max member length could be modified by some amount by changing the setting in the algorithm, so the option with max member length which exceeded the set limit of 2.4 metre by more than 5%, were discarded.
These remaining options were all similar in structural performance. Now the option with least deviation from planarity were considered. With only five options remaining, all five had only alignment varying, and other variables were same. Each option was now observed visually, but not much difference was found. This was not the case in the sequential strategy. This is because we had already changed the variable that govern the maximum length of beam. Changing that variable disturbs the direction of beams. As the maximum length of 2.4 metre was set as a constraint. This variable was changed and then these 6 options were compared again. The option with alignment 255 was chosen for same reason mentioned in the sequential strategy when comparing different alignment for principal stress direction, which was for balance between structural performance and homogeneity.

When all the remaining members were within the given constraints, the remaining options could be filtered down with performance. After choosing the options which were lightest, only 31 options remained.
Figure 5.27: Design explorer interface screenshot showing selected design
5.5 Comparison between both strategies

Time
The sequential strategy took 30 mins of computational time and 30 mins of design exploration and decision making. Considering the low computational time required, more experiment could be done with other variables in the workflow. For example, the variable which governs the maximum beam length in homogenisation process. The integrated strategy took 7 hours of computational time and 30 mins of design exploration and decision making. Breaking down the process in parts saved a lot of computational time as the design space gets narrows down with each step.

Learning Process
Sequential strategy helped more in learning process as at each step we were able to understand why a decision was being made whether it was for structural reasons or personal preference or a combination of both. As the design space was small at each step, it naturally lets one go deeper and get more insight from visualisation. For example in case of deciding alignment factor, we could notice how stresses in the middle part of the shell decreased with increase in alignment though performance indicators were not very helpful.

Decision making
While both strategies led to same result, the approach taken to get the result was quite different. In the sequential strategy one is forced to follow and order of considering variables for decision making. In fact, it was easier to make decisions when the design space was smaller. In Integrated strategy, one has multiple paths and each path might give different insight. For example, it reveals relationship between variables and performance criteria which was not noticed in sequential strategy.

Figure 5.28: Final grid shell
5.6 Validation

The resulting grid shell from the workflow was compared to similar method which is used in state of the art in practice YAS grid Shell. Two grid shell of same mass or amount of material, same cross section of beams and similar beam count were created two compare the structural performance. The static responsive grid shell outperforms the grid shells derived from standard methods. It had much lower displacement and maximum utilization or stress values. Total Strain energy or compliance is considered to check the degree of stiffness for structure. Structures are considered to be stiffer if their total strain energy/compliance is lower. Static responsive grid shell is 45% lower than regular quad grid shell and diagrid shell in terms of strain energy. Utilization is also very high in traditional shells. It can be seen in the utilization visualization that areas where beams are not aligned with principal stress direction are high in stress (red beams).

![Figure 5.29: Parallel coordinate chart](image)

![Figure 5.30: Performance indicators for different methods](image)

![Figure 5.31: Performance visualisation for different methods](image)

<table>
<thead>
<tr>
<th>Shell</th>
<th>Minority deviation (mm)</th>
<th>Max minor length (mm)</th>
<th>Ratting load factor</th>
<th>Displacement (mm)</th>
<th>Max utilization</th>
<th>Min (log) Strain energy (MEd)</th>
<th>No. of nodes</th>
<th>No. of beams</th>
<th>No. of unique beams</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>46</td>
<td>2.50</td>
<td>6.35</td>
<td>65</td>
<td>75.5</td>
<td>113901.2</td>
<td>22.04</td>
<td>1216</td>
<td>2240</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>5.86</td>
<td>4.90</td>
<td>360</td>
<td>75.4</td>
<td>112808.1</td>
<td>22.04</td>
<td>1205</td>
<td>2139</td>
</tr>
</tbody>
</table>

![Shell 0 (Static Responsive Quad Grid Shell)](image)

![Shell 1 (Regular Quad Grid Shell)](image)

![Shell 2 (Diagrid Quad Shell)](image)
The stress lines were checked again with wind load, but they did not change as the ratio of force applied by wind and forces acting due to gravity is very small as seen in figure 5.34(a). To check what happens if the shell was covered with a fabric instead of glass which has very light weight. This will increase the ratio of force acting on the shell by wind. Even then the combined load of the structure and snow load were high enough to not let the stress lines change(figure 5.34,b).

But snow is not always there even if the structure is designed for snow load. Then snow load was removed and finally some change in stress lines were noticed as seen in the figure 5.34,(c). These changes were not drastic. The structure still perform because the cross section was designed to take snow load and stresses in beams still stay under the limit.

In the case when cross section of the structure was not designed to take snow load, the self-weight of the structure will decrease. Now the ratio between self-weight of the structure to wind load is much higher to affect stress lines drastically as seen in figure 5.34(d) and theoretically this method may not be useful in that case. In case where the structure has holes or its not covered with any type of cladding, the wind load does is very minimum, and this method can be used. This leaves us with extremely few cases where using stress lines is not beneficial for discretization. This is the case where, the grid shell is covered with very light material (like a fabric or ETFE), it does not have holes, it is in the region where there is no chance of snow fall and wind load is high. It can be concluded that this method is beneficial for most grid shells without wind load being a problem.

There is another aspect which needs to be addressed to validate the research, that is the effect wind pressure. This method is based on using the flow of forces or direction of principal stress lines. The principal stress lines are calculated only be the force induced by gravity on the shell. Because any other load is variable, and it does not make sense to consider it while using this method of discretization. While analysing the structure, a wind load of 0.5 kN/m² in negative x-axis was applied on all shells in the case mentioned above, which should change the stress lines and the grid may not be aligned with them. It has already been discussed that the shell designed by this method performed better than other regular quad grid shell and diagrid shells. This is because the force acting on the shell by gravity is much higher than the force acting by wind. The self-weight, and snow load contribute to the forces acting in negative z-axis due to gravity.

Earlier the visualisation of stress lines were done without wind load. The shell taken for this case study requires CFD simulations for calculating wind pressure on the surface. Due to limitations of the thesis, these simulations was not performed. As this shell is very close to a dome but with openings, the standard wind pressure values given by ASCE7-05 (American Society of Civil Engineers) for a dome were taken to calculate wind pressure. The script for calculating wind load on each point on the grid shell can be found in Appendix 9.17.

Figure 5.32: Showing positive and negative pressure on dome like structures

Figure 5.33: Wind Pressure coefficient for domes, Source: Cheng, Fu & Lin(2008)

The stress lines were checked again with wind load, but they did not change as the ratio of force applied by wind and forces acting due to gravity is very small as seen in figure 5.34(a).

To check what happens if the shell was covered with a fabric instead of glass which has very light weight. This will increase the ratio of force acting on the shell by wind. Even then the combined load of the structure and snow load were high enough to not let the stress lines change(figure 5.34,b).

But snow is not always there even if the structure is designed for snow load. Then snow load was removed and finally some change in stress lines were noticed as seen in the figure 5.34,(c). These changes were not drastic. The structure still perform because the cross section was designed to take snow load and stresses in beams still stay under the limit.

In the case when cross section of the structure was not designed to take snow load, the self-weight of the structure will decrease. Now the ratio between self-weight of the structure to wind load is much higher to affect stress lines drastically as seen in figure 5.34(d) and theoretically this method may not be useful in that case. In case where the structure has holes or its not covered with any type of cladding, the wind load does is very minimum, and this method can be used. This leaves us with extremely few cases where using stress lines is not beneficial for discretization. This is the case where, the grid shell is covered with very light material (like a fabric or ETFE), it does not have holes, it is in the region where there is no chance of snow fall and wind load is high. It can be concluded that this method is beneficial for most grid shells without wind load being a problem.
6. Results and Discussions

Using the workflow for creating a shell structure for Delft bus station gave more than satisfactory results in terms of structural performance, homogeneity, planarity and informed decision making. The grid designed using this method structurally outperforms grid made from standard methods. While there are many other ways the algorithms were used, two different strategies were shown through a case study. Both strategies more or less lead to similar results but sequential strategy takes much less time than integrated strategy. The users can use the given strategies or alter the algorithm according to their needs.

In the case study it was assumed that a greater number of beams and nodes meant more costs in manufacturing and assembly. But in real case scenario it is much more complicated than that. For example, increasing the grid density does not necessarily mean that more beams will make the structure heavier because with more beams, the cross section of beams also decrease. It might make the structure lighter as the glass panels are also smaller, thinner, cheaper and lighter. Making the glass panel thinner will also affect the constraint for planarity. If a proper cost analysis was done considering all these factors, both strategies might have given different results. For example, when deciding density, we would need more insights in costs, structural performance and planarity deviation to make decisions. Sequential strategy might not have let us have a look at the larger picture and how changing one variable affect other aspects.

It was also noticed that with this particular boundary conditions, the factors of alignment and planarity didn’t have very large structural implications that was because the homogenisation process did not change the direction of beams by much. This is not always true with other shells as observed while developing and testing the algorithm.
7. Conclusions

What can be the method(workflow) to design statics-responsive grid shells with structural stability, homogenous grid and near planar cladding, including preferences of the designer?

A workflow was developed for designing statics-responsive grid shells which are structurally efficient, homogenous and near planar cladding, including preferences of the designer. The workflow is set up in a parametric environment in grasshopper, a plugin for Rhino 3d modelling software. It uses particle spring method for form finding a shell which has membrane like load bearing behaviour. The solid shell is discretized into a grid shell by a custom stress line generator which uses principal stress vector field derived by Finite element analysis of the shell. The grid shell is homogenised and optimized for planarity by dynamic relaxation. Multiple design alternatives are generated and stored. Design space is explored by using data analytics and visualization techniques which helps user to make informed design decisions. The workflow is applied to create a grid shell over delft bus station to protect travellers from varying weather conditions. The results are quite satisfactory in terms of structural performance when compared to methods used in state of the art in practice. The grid shell for delft bus station generated using this workflow was 45% lower in terms of total strain energy(compliance) than regular quad grid shell and diagrid quad shell.

• How can a solid shell be discretized into a grid shell with direction of beams aligned with direction of principal stresses?

This is done by a custom stress line generating tool which traces a point travelling through a vector field of principal stresses. Certain rules are applied to these lines which the user can alter. For example, if the principal lines intersect each other, the distance they maintain with each other and the way they culminate. This tool was written in Python and performs without any errors on different types of 2.5D surfaces tested.

• How can the grid of obtained shell be homogenised?

This is done by dynamic relaxation using physics engine, Kangaroo. The lines obtained from the custom stress lines tracer are treated as springs. These springs are given a rest length which is average length of lines close to it. These springs apply force on each other and are allowed to move until they balance out the forces or given number of iterations are over or movement is lower than given threshold. Points which are on the boundary are allowed to only move along the boundary. All points are allowed to move along the surface of mesh. The maximum and minimum length a line can be is set. Default values for constraints have been set but they might require some adjustment to get desired result for different types and sizes of shell. This simulation results in a homogenised grid.

• How can the obtained grid shell be optimized for planarity of faces?

This is done by dynamic relaxation using physics engine Kangaroo. The lines obtained from homogenised grid are treated as springs with their rest length as their original length. Points at each face of the mesh moves until each face becomes planar or close to planar. The points along the boundary are only allowed to move along the boundary. The translation of points which are the at the supports is fixed in z-axis with very high strength so that they remain grounded. This simulation performs well with any mesh and does not require alteration of constraint values.

• How can the workflow facilitate designer’s informed decision-making process?

Running performance simulations, storing data and visualisation of performance indicators helps designer to choose between different design alternatives. In this case, performance indicator like utilization, displacement and planarity were visualized in the form of images. Along with these all other
data like planarity deviation, buckling load factor, strain energy, number of nodes and beam, maximum beam length and mass were stored in a spreadsheet along with these images and 3d model. This data was then uploaded on Design Explorer which uses scatter plots and parallel coordinated charts for visualising. It also allows to browse through images and 3d model of each design alternative. This helps user to narrow down design space and choose the best solution based on performance indicators and personal preferences.

Future Research and Recommendations

Standardisation of elements
Standardisation of element help in easier and faster assembly process. During the research, it was attempted to reduce the unique number of beams and nodes. This objective could not be achieved because the boundary conditions were fixed for the given structure. Having a few standard elements is only possible if the boundary is flexible and the grid is dense. Scope of the research was to discretization of shell surfaces without changing the boundary conditions. An algorithm could be added in the workflow which can reduce the number of beams and nodes provided that the designer is flexible with the shape of the shell.

Automation of workflow
The workflow requires some manual editing to produce desired results for some types of shells where ring forces are present. An algorithm could be developed which can do this automatically by rule-based corrections and create design alternative for user to choose from.

3d Geometry
The scope of the research was only limited to 2.5d shapes, it can be expanded to work for 3d surfaces.

Possibility to manipulate density according to magnitude of stresses
While this happens to some extent by following the direction of principal stresses, but an algorithm could be developed which uses the result of finite element analysis and add or remove continuous beams path according to the magnitude of stress in the grid shell.


9 Appendix A
Grasshopper Scripts
9.1 Script for defining Mesh and anchor points.

9.2 Script for form finding.
9.3 Script for Finite Element Analysis

9.4 Script to extract principal stress vector field
9.5 Script to get mesh naked support edges and free edges

9.6 Script to extract seed points

9.7 Script to visualize stress line from Karamba
9.8 Component for Stress line generation and script for dividing naked curves at corners

9.9 Script for varying density of grid
9.10 Script to convert stress lines to a grid

9.11 Script to subdivide polygons into quadrilaterals
9.12 Script for homogenisation

9.13 Script for Alignment
9.14 Script for Planarity

9.15 Script for Visualizing Planarity Deviation
9.16 Script for Finite element analysis for grid shell
9.17 Script to calculate wind load

9.18 Script to create design alternative and store data
10 Appendix B

Custom Tools

Python Code

10.1 Corner point of support curve (CT1)

#Works only if the support curve is on ground(z-axis=0)
import rhinoscriptsyntax as rs
import ghpythonlib.treehelpers as gt
anchor=[]
other=[]
alldis=[]
i=0
breakPts=[]
for i in range(len(x)):
    if x[i][2]<0.05:
        if y[i][2]>0.05:
            breakPts.append(x[i])
        if y[i][2]<0.05:
            if x[i][2]>0.05:
                breakPts.append(y[i])

10.2 Sort points by reference point (CT2)

import rhinoscriptsyntax as rs
dist=[]
a=[]
for i in range(len(x)):
    dis=rs.Distance(y,x[i])
    dist.append(dis)
newdist=dist[:]
newdist.sort()
for i in range(len(x)):
    a.append(x[dist.index(newdist[i])])

10.3 Custom stress line generator(CT3)

import rhinoscriptsyntax as rs
import math
import Rhino.Geometry as rg
import ghpythonlib.components as gc
import ghpythonlib.treehelpers as gt

def streamGen(FieldVectors,boundary,boundaryopp,StartPts):
    #***********************************************************************
    #unitizing vector field
    #***********************************************************************
    m=0
    UniVectors=[]
    mne = rs.AddPolyline(MeshNakedPts)
    while m<len(FieldVectors):
        UVector= rs.VectorUnitize(FieldVectors[m])
        UniVectors.append(UVector)
        m+=1
    io=0
    z=0
    streamline=[]
    pointTrailall=[]
if streamlinesExisting != None:
  streamlines.extend(streamlinesExisting)

while z < len(StartPts):

  # To remove start/seed points which are closer than the lineMinDist
  ptrail1 = StartPts[z]
  if z>0 and len(streamlines)>0:
    k=0
    actD=[]
    actT=[]
    while k<len(pointTrailall):
      t = rs.PointArrayClosestPoint(pointTrailall[k],ptrail1)
      d = rs.Distance(ptrail1,pointTrailall[k][t])
      actD.append(d)
      minD = actD.index(min(actD))
      k+=1
      if actD[minD]<linesMinDist:
        io+=1
        z+=1
        continue

  i = 0
  j=0
  D=[]
  pointTrail1=[]
  pointTrail2=[]
  ptrail1 = StartPts[z]
  ptrail2 = StartPts[z]
  pointTrail1.append(ptrail1)
  pointTrail2.append(ptrail2)
  countx1a=0
  countx1b=0
  countx2a=0
  countx2b=0

  # Principal stress line 1 (start)
  while i<c:
    d1=rs.PointArrayClosestPoint(FieldPts,ptrail1)
    if i<1:
      ptrail1 = rs.CopyObject(ptrail1,(UniVectors[d1]*StepSize))
    a= rs.VectorCreate(ptrail1,pointTrail1[i])
    else:
      a= rs.VectorCreate(pointTrail1[i],pointTrail1[i-1])
    Ptsln=[]
    Vecin=[]
    weight=[]
    f=0
    while f<len(FieldPts):
      disR=rs.Distance(ptrail1,FieldPts[f])
      if disR<radius:
        Ptsln.append(FieldPts[f])
        Vecin.append(UniVectors[f])
        weight.append(math.exp(-(disR/sig)))
      f+=1
    h=0
    ResVec1n=[]
    while h<len(Vecin):
      anglein=rs.VectorAngle(a,Vecin[h])
      if anglein<70:
        ResVec1n.append(Vecin[h]*weight[h])
      if anglein>110:
        Vecin[h]=rs.VectorReverse(Vecin[h])
        ResVec1n.append(Vecin[h]*weight[h])
      h+=1
    if len(ResVec1n)>1:
      ResVec1n.append(a)
      h=1
    FinVec1=ResVec1n[0]
    while h<len(ResVec1n):
      FinVec1=rs.VectorAdd(FinVec1,ResVec1n[h])
      h+=1
    FinVec1=rs.VectorUnitize(FinVec1)
    UV1=FinVec1
  x=pointTrail1[i-1]
  angle1=rs.VectorAngle(a,FinVec1)
  if angle1> 80:  # correcting the streamline direction
    UV1 = rs.VectorReverse(FinVec1)
    angle1a = rs.VectorAngle(a,UV1)
    if angle1a>stepMaxAngle:
      UV1 = rs.VectorRotate(UV1,-(angle1a-stepMaxAngle),[0,0,1])
    angle1b=rs.VectorAngle(a,UV1)
    if angle1b>angle1a:
      UV1 = rs.VectorRotate(UV1,2*(angle1a-stepMaxAngle),[0,0,1])
    ptrail1 = rs.CopyObject(ptrail1,(UV1*StepSize))
  else:
    for s in range(len(streamlines)):
      if rs.CurveCurveIntersection(latestline,streamlines[s],0.1) !=None:
        counter+=1
    if counter>0:
      for f in range(13):
        counter=0
        UV1new=rs.VectorRotate(UV1,5*f,[0,0,1])
        ptrail1 = rs.CopyObject(pointTrail1[len(pointTrail1)-1],UV1new*StepSize)
        latestline=rs.AddLine(pointTrail1[len(pointTrail1)-1],ptrail1)
        for t in range(len(streamlines)):
          if rs.CurveCurveIntersection(latestline,streamlines[t],0.1) !=None:
            counter+=1
    if counter>0:
      f=1
      while f<13:
        counter=0
        UV1new=rs.VectorRotate(UV1,5*f,[0,0,1])
        ptrail1 = rs.CopyObject(pointTrail1[len(pointTrail1)-1],UV1new*StepSize)
        latestline=rs.AddLine(pointTrail1[len(pointTrail1)-1],rs.MeshClosestPoint(Mesh1[0],ptrail1)[0])
        for t in range(len(streamlines)):
          if rs.CurveCurveIntersection(latestline,streamlines[t],0.1) !=None:
            counter+=1
        if counter<1:
          UV1=UV1new
          break
      if 0>f>-13:
f=1
if f<-12:
f=0
if f>1:
f=1
if counter>0:
    ptrail1=rs.CopyObject(pointTrail1[len(pointTrail1)-1],(UV1*StepSize))
projectpoint1=rs.ProjectPointToMesh(ptrail1,Mesh1,(0,0,1))

# To avoid intersection of stress lines (end) #

# looping (start) #
if len(projectpoint1)>0:
    finalpoint1=rs.MeshClosestPoint(Mesh1[0],ptrail1)
    ptrail1=finalpoint1[0]
    for w in range(len(pointTrail1)):
        dis= rs.Distance(ptrail1,pointTrail1[w])
        if dis<((linesMinDist/2)+.02):
            countx1a+=1
    for w in range(len(pointTrail2)):
        dis= rs.Distance(ptrail1,pointTrail2[w])
        if dis<((linesMinDist/2)+.02):
            countx1b+=1
    if (countx1a>0 or countx1b>0) and allowLooping==True:
        break
    pointTrail1.append(ptrail1)
else:
i+=1000
i+=1
# looping (end) #

# Last point on the pointTrail1 on mesh boundary (start) #
if len(pointTrail1)>1 and len(projectpoint1)<1:
    lastline=rs.AddLine(pointTrail1[len(pointTrail1)-1],ptrail1)
    for q in range(len(meshNakedEdge)):
        interlast=rs.CurveCurveIntersection(meshNakedEdge[q],lastline,.5)
        if interlast!= None:
            lastPt=interlast[1][1]
            del pointTrail1[len(pointTrail1)-1]
            lastDis=rs.Distance(pointTrail1[len(pointTrail1)-1],lastPt)
            if lastDis<(StepSize/2)-2):
                del pointTrail1[len(pointTrail1)-1]
pontTrail1.append(lastPt)
lastDis=rs.Distance(pointTrail1[len(pointTrail1)-1],pointTrail1[len(pointTrail1)-2])

# Last point on the pointTrail1 on mesh boundary (end) #

# principal stress line 1 (end) #

# principal stress line 2 (start) #
while j<c:
    d2=rs.PointArrayClosestPoint(FieldPts,ptrail2)
    if j==0:
        ptrail2= rs.CopyObject(ptrail2,(UniVectors[d2]*StepSize*(-1)))
        projectpoint2=rs.ProjectPointToMesh(ptrail2,Mesh1,(0,0,1))
    if len(projectpoint2)>0:
        ptrail2=projectpoint2[0]
        b= rs.VectorCreate(ptrail2,pointTrail2[j-1])
    PtsIn=[]
    VecIn=[]
    weight=[]
    f=0
    while f<len(FieldPts):
        disRs=rs.Distance(ptrail2,FieldPts[f])
        if disRs<radius:
            PtsIn.append(FieldPts[f])
            VecIn.append(UniVectors[f])
            weight.append(math.exp(-(disRs/sig)))
        f+=1
    h=0
    ResVecIn=[]
    while h<len(VecIn):
        angleIn=rs.VectorAngle(b,VecIn[h])
        if angleIn<70:
            ResVecIn.append(VecIn[h]*weight[h])
        if angleIn>110:
            VecIn[h]=rs.VectorReverse(VecIn[h])
            ResVecIn.append(VecIn[h]*weight[h])
        h+=1
    h=1
    if len(ResVecIn)<1:
        ResVecIn.append(b)
        FinVec=rs.VectorUnitize(FinVec)
    while h<len(ResVecIn):
        FinVec=rs.VectorAdd(FinVec,ResVecIn[h])
        h+=1
    FinVec=rs.VectorUnitize(FinVec)
    UV2=FinVec
    y=pointTrail2[j-1]
    angle2=rs.VectorAngle(b,FinVec)
    if angle2>80:
        UV2 = rs.VectorReverse(FinVec)
        angle2 = rs.VectorAngle(b,UV2)
    if angle2>stepMaxAngle:
        UV2 = rs.VectorRotate(UV2, angle2-stepMaxAngle),[0,0,1])
        angle2 = rs.VectorAngle(b,UV2)
    if angle2>angle2a:
        UV2 = rs.VectorRotate(UV2, stepMaxAngle),[0,0,1])
        angle2 = rs.VectorAngle(b,UV2)
ptrail2 = rs.CopyObject(ptrail2, (UV2 * StepSize))

###############To avoid intersection of stress lines(start)###############
latestline = rs.AddLine(pointTrail2[len(pointTrail2) - 1], ptrail2)
counter = 0
for s in range(len(streamlines)):
    if rs.CurveCurveIntersection(latestline, streamlines[s], 0.1) != None:
        counter += 1
if counter > 0:
    f = -1
    while f < 13:
        counter = 0
        UV2new = rs.VectorRotate(UV2, 5 * f, [0, 0, 1])
        ptrail2 = rs.CopyObject(pointTrail2[len(pointTrail2) - 1], UV2new * StepSize)
        latestline = rs.AddLine(pointTrail2[len(pointTrail2) - 1], rs.MeshClosestPoint(Mesh1[0], ptrail2)[0])
        for t in range(len(streamlines)):
            if rs.CurveCurveIntersection(latestline, streamlines[t], 0.1) != None:
                counter += 1
        if counter < 1:
            UV2 = UV2new
            break
        if 0 <= f < 13:
            f = 1
        if f >= 12:
            f = 0
        if f > 1:
            f += 1
        if counter > 0:
            ptrail2 = rs.CopyObject(pointTrail2[len(pointTrail2) - 1], (UV2 * StepSize))
projectpoint2 = rs.ProjectPointToMesh(ptrail2, Mesh1, [0, 0, 1])

###############To avoid intersection of stress lines(end)###############

riba2 = UV2 = UV2new
if len(projectpoint2) > 0:
    finalpoint2 = rs.MeshClosestPoint(Mesh1[0], ptrail2)
    pointTrail2 = pointTrail2[:]
    del pointTrail2[len(pointTrail2) - 1]
    for w in range(len(pointTrail2)):
        dis = rs.Distance(ptrail2, pointTrail2[w])
        if dis <= (linesMinDist / 2) + .02:
            countx2a += 1
for w in range(len(pointTrail1)):
    dis = rs.Distance(ptrail2, pointTrail1[w])
    if dis <= (linesMinDist / 2) + .02:
        countx2b += 1
if (countx2a > 0 or countx2b > 0) and allowLooping == True:
    break
pointTrail2.append(ptrail2)
else:
    j += 1000
    j += 1

if len(pointTrail1) > 1 and len(projectpoint2) < 1:
    lastline = rs.AddLine(pointTrail2[len(pointTrail2) - 1], ptrail2)
    for q in range(len(meshNakedEdge)):
        interlast = rs.CurveCurveIntersection(meshNakedEdge[q], lastline, .5)
        if interlast != None:
            lastPt = interlast[0][1]
            lastDist = rs.Distance(pointTrail2[len(pointTrail2) - 1], lastPt)
            if lastDist <= (StepSize / 2) - .2:
                del pointTrail2[0:last2]
                del pointTrail1[0:len(pointTrail1)]

###Last point on the pointtrail2 on mesh boundary(start)###

###Last point on the pointtrail2 on mesh boundary(end)###

###principal stress line 2 (end)###

###looping corrections(start)###

###looping corrections(end)###

###looping corrections(start)###

###looping corrections(end)###

pointTrail = pointTrail2[:]
list.reverse(pointTrail)
del pointTrail[len(pointTrail)-1]
if len(pointTrail)>0:
    pointTrail.extend(pointTrail1)
streamline = rs.AddPolyline(pointTrail)
if rs.Distance(pointTrail[0],pointTrail[len(pointTrail)-1])<(StepSize*2) and (countx1a>0 or countx1b>0 or countx2a or countx2b):
    streamline=rs.CloseCurve(streamline,(StepSize*2))
if streamline!=None:
    count=0
    badloop=0
    countboundcorrect=0
    countbounddouble=0
    looping=0
    freecurve=0
if countx1a>0 or countx1b>0 or countx2a or countx2b:
    explodelines=rs.ExplodeCurves(streamline)
    rogueangle=rs.Angle2(explodelines[0],explodelines[len(explodelines)-2])
    print rogueangle[0]
    if rogueangle[0]>120:
        badloop+=1
    if len(streamlines)>0:
        for r in range(len(streamlines)):
            inter=rs.CurveCurveIntersection(streamline,streamlines[r],0.1)
            if inter!=None:
                count+=1
        for e in range(len(boundary)):
            boundarycorrect=rs.CurveCurveIntersection(streamline,boundary[e],0.02)
            if boundarycorrect!=None:
                countboundcorrect+=1
        for e in range(len(boundaryopp)):
            doubleboundary=rs.CurveCurveIntersection(streamline,boundaryopp[e],0.02)
            if doubleboundary!=None:
                countbounddouble+=1
        forfreecurve=0
        for n in range(len(meshNakedEdge)):
            if rs.CurveCurveIntersection(streamline,meshNakedEdge[n],0.02)==None:
                forfreecurve+=1
                paramsnew=params[:]
        if (count<1 or allowIntersection==True) and (countboundcorrect<1 or allowOppositeBoundaryEnds==True) and forfreecurve<1 and badloop<1 and (countbounddouble<1 or allowSingleBoundaryEnds==True):
            pointTrailall.append(pointTrail)
            streamlines.append(streamline)
    z+=1
    return (streamlines)

10.4 Corner point CT4)
import rhinoscriptsyntax as rs
import ghpythonlib.treehelpers as gt
sP=[]
for i in range(1):#(len(x)):
    points=[]
    lines=rs.ExplodeCurves(x[i])
    for j in range(len(lines)-1):
        if rs.Angle2(lines[j],lines[j+1])>20:
            points.append(rs.CurveCurveIntersection(lines[j],lines[j+1],.5)[0][1])
            print rs.Angle2(lines[0],lines[len(lines)-1])
        if rs.Angle2(lines[j],lines[j+1])>20:
            points.append(rs.CurveCurveIntersection(lines[j],lines[j+1],.5)[0][1])
sP .append(points)
shatterPoints=gt.list_to_tree(sP)

10.5 If x is Null return y(CT5)
import rhinoscriptsyntax as rs
a=x
if len(x)<1:
    a=y

10.6 Intersection points(CT6)
import rhinoscriptsyntax as rs
import ghpythonlib.treehelpers as gt
i=0
allPts=[]
allPoly=[]
while i <len(x):
    temp=x[i]
    del temp[i]
    b=[]
    j=0
    while j < len(tempx):
        a=rs.CurveCurveIntersection(tempx[j],tempx[j+1],0.03)
        if a!= None:
            for q in range(len(a)):
                b.append(a[q][1])
        j+=1
    allPts.append(b)
    del x[i]
    allPtstree=gt.list_to_tree(allPts)

10.7 Making Polyline along Curves with point Cloud (CT7)
import rhinoscriptsyntax as rs
allPoly=[]
for i in range(len(curves)):
    a=[]
    params=[]
    c=[]
    for j in range(len(points)):
        closestparam= rs.CurveClosestPoint(curves[j],points[j])
        closest=rs.EvaluateCurve(curves[j],closestparam)
        disrs.Distance(closest,closestpoint)
        if dis<0.05:
            a.append(points[j])
            params.append(closestparam)
            paramsnew=params[:]
list.sort(paramsnew)
sortedindex=[]
for t in range(len(params)):  
    index=params.index(paramsnew[t])  
    sortedindex.append(index)
for w in range(len(paramsnew)):  
    c.append(a[sortedindex[w]])
first=c[0]
poly=rs.AddPolyline(c)
count=0
if rs.IsCurveClosed(curves[i])==True:
c.append(c[0])
poly=rs.AddPolyline(c)
allPoly.append(poly)

10.8 Calculating Rest Length (CT8)

import rhinoscriptsyntax as rs
import math as math
linesbrk=[]
lineslen=[]
linesvec=[]
restlen=[]
for i in range(len(lines)):  
    linebrk=rs.DivideCurve(lines[i],2)  
    linesbrk.append(linebrk)
    linevec=rs.VectorCreate(linebrk[0],linebrk[2])  
    linesvec.append(linevec)
    linelen=rs.CurveLength(lines[i])
    lineslen.append(linelen)

for i in range(len(lines)):  
    # sph1=rs.AddSphere(linesbrk[i][1],radius)  
    linesin=[]
    ptsin=[]
    a=lines[i]
    for j in range(len(lines)):  
        dist=rs.Distance(linesbrk[i][1],linesbrk[j][1])  
        if dist<=radius:
            ptsin.append(linesbrk[j][1])
            angle1=rs.VectorAngle(linesvec[i],linesvec[j])
            if angle1<35 or angle1>145:
                linesin.append(lineslen[j])
    r=sum(linesin)/len(linesin)
    restlen.append(r)

10.9 Calculating alignment Vector(CT9)

import rhinoscriptsyntax as rs
midy=[]
midx=[]
vecy=[]
vecx=[]
finVecs=[]
for i in range (len(y)):  
    pointsy=rs.DivideCurve(y[i],2)  
    pointsx=rs.DivideCurve(x[i],2)  
    midy.append(pointsy[1])  
    midx.append(pointsx[1])
    vecy=rs.VectorCreate(pointsy[0],pointsy[2])
    vecx=rs.VectorCreate(pointsx[0],pointsx[2])
    vecy=rs.VectorUnitize(vecy)
    vecx=rs.VectorUnitize(vecx)
    vecy.append(vecy)
    vecx.append(vecx)
    vecy.append(vecy)
    vecx.append(vecx)

for i in range(len(y)):  
    distance=[]
    vecin=[]
    vecinFin=[]
    for j in range(len(y)):  
        distance=rs.Distance(midy[i],midx[j])  
        if distance < radius:
            vecin.append(vecy[j])
        if rs.VectorAngle(vecy[i],vecin[j])>90:
            vecin[j]=rs.VectorReverse(vecin[j])
        if rs.VectorAngle(vecy[i],vecin[j])<40:
            vecinFin.append(vecin[j])
    if len(vecinFin)>0:
        finVec=vecy[i]
    k=1
    while k<len(vecinFin):
        finVec=rs.VectorAdd(finVec,vecinFin[k])
        k+=1
    finVec=rs.VectorUnitize(finVec)
    finVecs.append(finVec)

10.10 Intersecting curves(CT10)

import rhinoscriptsyntax as rs
import ghpythonlib.treehelpers as gt
a=[]
line=[]
i=0
while i < len(z):
    count=0
    bri=[]
    for j in range(len(z[i])):  
        inter=rs.CurveCurveIntersection(z[i][j],0.02)
        if inter!=None:
            a.append(z[i][j])
            count+=1
        if count<1:
            line.append(z[i][j])
            i+=1
    a=gt.list_to_tree(a)