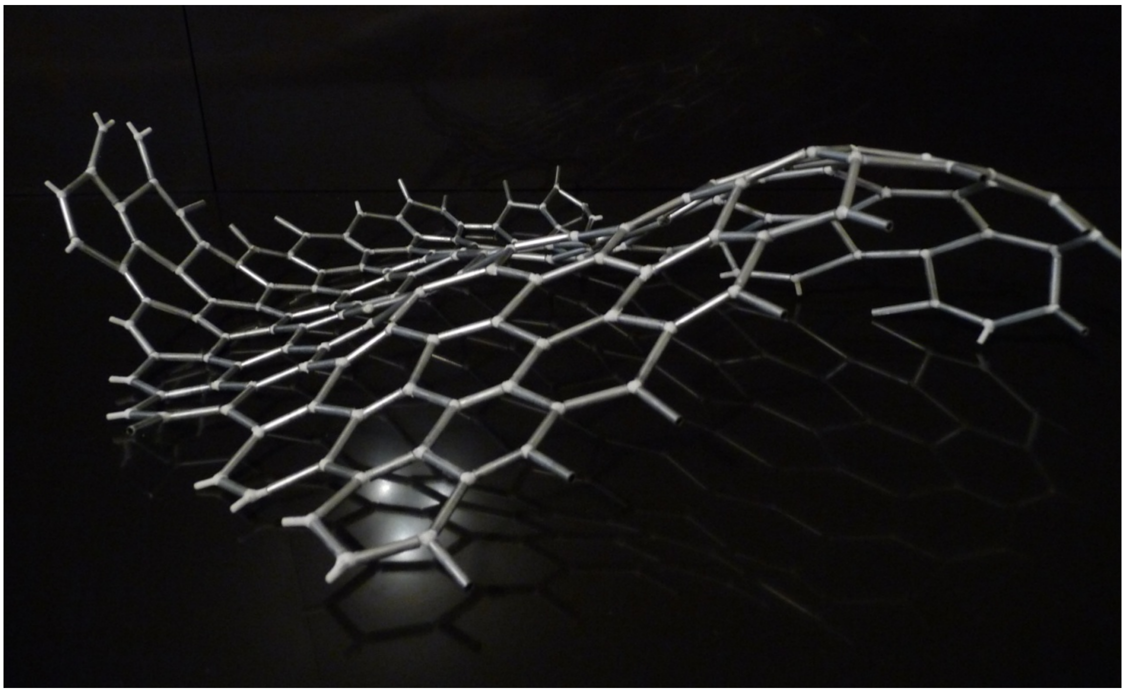
Honeycomb Topologies;
Design Rationalization of a free-form space frame structure

Sahar Darvishani-Fikouhi



This dissertation is submitted in partial fulfilment of the requirements for the degree of Master of Science in Adaptive Architecture & Computation from the University of London.

Bartlett School of Graduate Studies
University College London

September 2009

Abstract

An increased use of free form surfaces in design often results to mass customised components and structural systems which are complex to assemble and expensive to produce. Often to resolve these forms, a tessellation algorithm is used to break the geometry down to manageable components which can be fabricated individually. The issue that can arise is the production of highly differentiated components which can be inefficient to produce. Therefore this investigation aims to resolve the issues regarding build-ability and tessellation of a free-form surface through a design rationalisation of a honeycomb space frame structure. The process used entails the development of a generative algorithm combining dynamic relaxation and particle spring systems, which aims to minimise the number of structural variations that occur in an arbitrarily curved surface. The resultant algorithm is evaluated in light of a cost effective solution and an optimum result is formulated through the results of the compiled experiments. The outcome of the final tool demonstrates a capacity to improve the constructional parameters of a multitude of geometric forms, providing a methodology that can be easily incorporated by architects and designers for further advancement and use.

Word count: 10020

Acknowledgements

I would like to thank my supervisors **Sean Hanna** and

Alasdair Turner for all their patience and advice.

I would also like to thank my family and Arta

for their constant support and encouragement.

Table of contents

1.0 Introduction

The research presented in this thesis is a study into the particular geometrical and fabrication problems concerning the construction of a free-form space frame structure.

1.1 Optimisation of reticulated structures

In recent years creating free form geometry with the use of 3d Modelling software has become a relatively easy task to achieve, and although the use of non-uniform rational B-splines (NURBS) surfaces can help obtain almost any imaginable form, resolving these surfaces for construction is not so easy to achieve via conventional modelling packages. Unlike algebraic surfaces such as cylinders spheres and paraboloids, which are defined by fixed equations, NURBS-surfaces "require a complex construct of mathematical objects, like lines, curves and planes, formula and procedures, which interact to specify the form in an iterative way." (Stephan 2009) Therefore free-form surfaces in architecture generally use a form-finding process in order to resolve and construct them into buildable components.

Form-finding methods help find the optimal surface of material systems which are under the influence of forces. They are non-geometric mode of generating surfaces which and can be implemented by experimental or numerical techniques. In membrane structures the displacement of the points making up the surface are related to the specified material, and the state of equilibrium between the internal resistance and the external forces defines the final form of the surface. Traditional form finding methods employed by architects such as Gaudi and Frei otto include hanging of physical nets or fabrics, and soap bubbles, to help understand compression grid shells and shape pneumatics. The Numerical counterparts of the physical experiment include force density and dynamic relaxation algorithms which help produce digital simulations of the mentioned experiments. In digital simulations of this process, a mesh is used to denote a specified material with defined elasticity and material properties along with the inclusion of the relevant forces and fixed boundary points. The final resting position of the mesh is achieved via iterative calculations to obtain an equilibrium state. Generally form-finding methods are an optimisation technique which can help analyse and resolve the structural behaviour of free-form reticulated structures; however they can also be used to perform geometrical optimisation to resolve issues regarding fabrication parameters such as limiting the variations of component sizes or ensuring that quadrangular facets lie on the same plane between structural members.

Reticulated structures of free form surfaces can be obtained in various ways, such as extracting the intrinsic curves from the surface, or by mapping a planar network onto the surface. Although various methodologies are used to break up a surface into a series of buildable components, the most common method is the mapping of reticulated grid onto the surface. In this way the geometry is decomposed into smaller tiles that can be used as a guide for the structural system and cladding components. In order to resolve these panels into buildable components and reduce the discrepancies between them, top down optimisation techniques such as dynamic relaxation are generally used which allow the tessellated surface to be manipulated and refined globally. In this way, the surface is explored as a whole and the series of points making up the digital mesh are moved on the surface to reach the optimum result.

1.2 Cellular structures

Cellular structures are forms that appear repeatedly in nature including crystals of snowflakes, hexagonal lattice work of carbon molecules, and honeycombs. The geometry of such patterns arise as a result of efficiency and strength which have been explored by scientist including D'arcy Thomson, and proved that the self organisation principles of these forms arise from the same mathematical, geometrical and physical properties of soap bubbles.

In his book, on growth and Form, Thomson explains how the form of a close packed cluster of circular cells, where each cell is in contact with six others surrounding it, creates a hexagonal shape due to surface tension. In relation to a Honeycomb, the initial form of each cell, before crowding and mutual pressure, starts as hemispherical cups, as demonstrated by the close packing of three or four soap bubbles. These cups are then packed as close as possible which results in "symmetrical tensions in the semi-fluid films" (Thomson 1961: 89) and thus results in an equilibrium state of a hexagonal shape. Due to this formation, the honeycomb holds the most honey for the least wax, which in terms means the "minimum extent of boundary in a plane" (Thomson 1961: 90). In a similar way to the honeycomb, the hexagonal patterns of diatoms (a form of minute algae) are also as a result of the closest possible packing of vesicles which are surrounded by six others. On the whole, this geometry lends itself to efficiency in nature, with three sides, rather than two, for each half of the hexagon, as well as more strength due to the fact that the "radius of a circle which circumscribes the hexagon is the same length as one side of the hexagon." (Thomson 1961: 90).

By using biological models of the processes which are inherent in natural materials and forms, it is possible to establish new strategies for design and construction. Cellular biological materials are self-organised in hierarchies from simple components into complex structures which have characteristics, including their efficient use of material and their distribution of forces, which make them a significant point of reference for architectural design.

1.3 Space Frames

Space frame structures are a common architectural technique used predominantly for large span roofs with minimal vertical supports. They consist of a three dimensional network of nodes and struts which use an omni-directional spreading of loads. The linear elements known as struts are linked using a series of node connectors which distribute the applied forces in an axial direction.

There are many advantages to using a space frame construction, mainly because they are an economical system which can be mass produced into a series of small components that are easily transported, handled and assembled without highly skilled labour. Their lightweight nature also reduces susceptibility to seismic forces, although the most interesting aspect of a space frame system is their potential for versatility of shape and form. The integration of computers in both the design and manufacturing stage of the system are significant aspect of their versatility. Cad systems are used to help explore complex configurations and free form geometries whilst the cam systems used help cut and drill the elements with great precision and flexibility providing interesting geometric patterns.

Space frames can be arranged in multiple layers of intersecting members, which are usually built from repetitive modules with minimal differentiations in nodes and strut lengths. However recent advancement of single layer space frames, with the use of rigid connections, provide a system that is able to construct any conceivable geometry through the integration of differing node connectors that provide the necessary angles. The use of a single layer structure provides a more economical use of material as well as allowing for transparent building envelopes. Nonetheless, the structural issues of these systems can result in a rotation of nodes and lack of fit of members due to axial loads and residual stresses within the system. Therefore the underlining criteria that needs to be addressed in the design of such structures includes the geometry of the nodes as well as the connection of the struts and the polyhedral units possible for each system.

Due to the mentioned advantages, the use of a space frame construction is a logical method for the resolution of a complex free form surface. The ability to utilise a standard module and to integrate geometry with structure provides an easy methodology for the development of new architectural forms.

1.4 Problem Definitions and Thesis Aims

Due to the increase of free form surfaces, and the necessity for mass customisation, it is important to consider the construction logic and cost effectiveness of these structures. In general, free-forms are complicated to construct and expensive to build due to the variety of different components making up the geometry. Often, to resolve these forms it is important to use some type of a tessellation algorithm to break the geometry down to manageable components which can be fabricated individually. This approach requires a top down methodology, where the geometry is resolved in regards to its global topology.

The following problems are often points that need to be addressed in the construction of such structures:

- In a single layer structure, the structural behaviour is generally not predictable as the stress in the structural members can vary from solely tension, compression or bending stress.
- The local geometric parameters can have vast variations such as the angular separation and size of each structural member and connecting nodes.

This investigation aims to answer issues relating to the latter. Through the development of a programmatic tool which is able to resolve a free form surface into buildable components, the investigation will aim to answer the following question;

How to tessellate and construct a double curved surface using standard construction techniques in a cost effective manner?
How to minimise the level of variation between structural members in an arbitrary surface with minimal deformation of the defined geometry?

Given that the above questions are mainly focused on the issue of cost, the steering objective is to optimise a tessellated surface with minimum variations in structural members and the least amount of material use. In doing the following points are issues which need to be addressed in the search for an optimum solution;

- Multiple variations of nodes connecting each spring will require mass-customised fabrication to cater for the varying geometrical parameters of the structural members.
- Increased density of the tessellated grid means increased material use and greater cost
- Increased variation in lengths means greater cost for construction.

Although the cost of cutting different space frame members is almost negligible due to CNC production, the time required to assemble highly differentiated tessellations can produce an increased cost in the construction phase, and is therefore taken as the deciding factor. On the other hand, increasing the density of the tessellation can help reduce the variation of lengths, even though this means more material usage and increased cost. Therefore in order to satisfy an acceptable level of material use and variations in structural sizes, an optimum solution is sought for both criteria.

Given an arbitrary curve, it is not an easy task to create a tessellated pattern from equal sized members; therefore particle spring system optimisation can be used to define a series of optimum lengths as opposed to more common form-finding methods. Although dynamic relaxation algorithms are expected to minimise the variations in the member lengths, an optimisation procedure, through the use of particle spring systems, is the possible solution for achieving the exact required ideal lengths. By changing the lengths of each structural member it is expected that the initial form of the given geometry will be altered to accommodate the updated lengths, however the goal will be to test the efficiency of dynamic relaxation method and particle spring optimisation in order to achieve a solution which satisfies the mentioned objectives.

The intention of the investigation is to resolve issues regarding build-ability and tessellation of a free-form surface, and therefore is not intended as a design application, but simply a tool for the resolution of double curved building envelopes or structural systems. Therefore, given a free-form surface, the initial phase of this thesis is to tessellate the surface into small cells of hexagons, making up a structural system, which can be easily resolved into buildable components. The final aim is to create a single layer space frame construction using the results of the compiled experiment with minimal variations in strut members. In doing so, certain issues need to be considered as addressed below;

- The topology should consist of varying number of nodes which can be altered to create different density tessellations.
- All cells need to remain hexagonal in order to maintain a consistent pattern throughout.
- Nodes will have to be restricted to the given surface during dynamic relaxation so that the springs can adjust in relation to the overall topology.
- The spring lengths will need to be defined in relation to the overall range during the particle spring optimisation.
- Construction logic will need to be defined in the algorithm.
- Tolerance of the final structure needs to be considered before optimisation.

1.5 Structure of the thesis

In the following section some computational precedents in architectural design regarding the mentioned geometrical and fabrication issues will be presented, which will then be followed in section 3 by the methodological approach undertaken for the development of the final algorithm. The implemented code will be analysed in more detail in section 4, where experimentation of the various parameters will help define an optimum solution for the given geometry. These results will be evaluated in further detail in section 5 which will help assess the viability of this approach for future use and development. The final section will aim to present an overall review of this investigation and a final conclusion will be drawn based on the compiled investigations.

2.0 Review of related work

In this section a background of some methodological approaches regarding form-finding and construction of free-form cellular structures, will be presented through the analysis of various architectural and computational precedents.

2.1 Dynamic Relaxation

The Bergen National Academy of arts, Snohetta Architects
Dynamic relaxation is a method which is generally used to digitally simulate form-finding of membrane and cable net structures, however in recent years several architectural structures have also used a relaxation process as a means to optimise the structural geometry to fulfil various criteria.

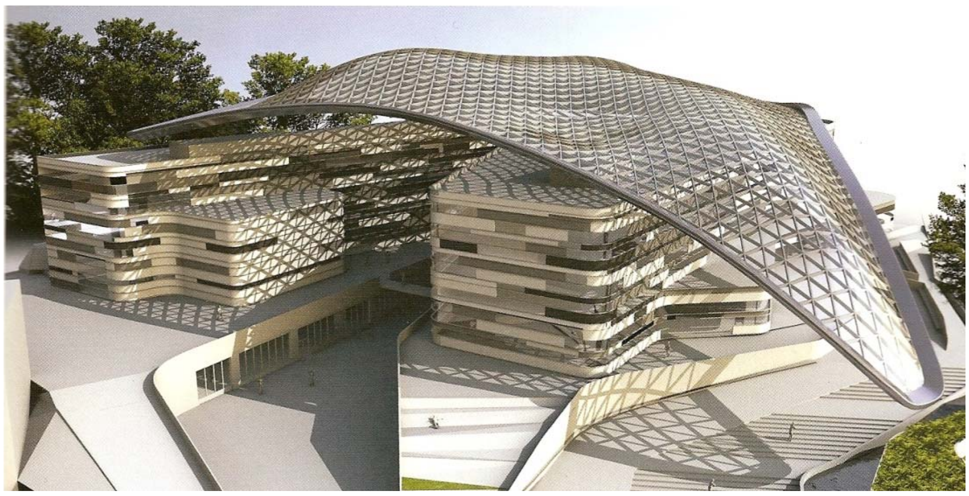


Figure 1. Computer model of the Bergen National Academy of arts (Littlefield 2008: 7)

The Bergen National Academy of arts by Snohetta architects and Buro Happold Engineers used a relaxation process for the optimisation of the steel and glass roof. The optimisation aimed at reducing the cost of two constructional parameters including the weight of steel and the area of glass. Primarily the optimum weight of steel was determined by a relaxation procedure which simulated the behaviour of the roof using a digital version of the hanging chain model. This simulation provided a direct relationship between the stress and thickness of each structural member, and iteratively changed the beam sizes until an optimum shape was determined with minimal deflection and stress. This procedure resulted in a “40 percent lighter and 3.2 metre higher” (Littlefield 2008: 7) outcome than the initial form of the roof.

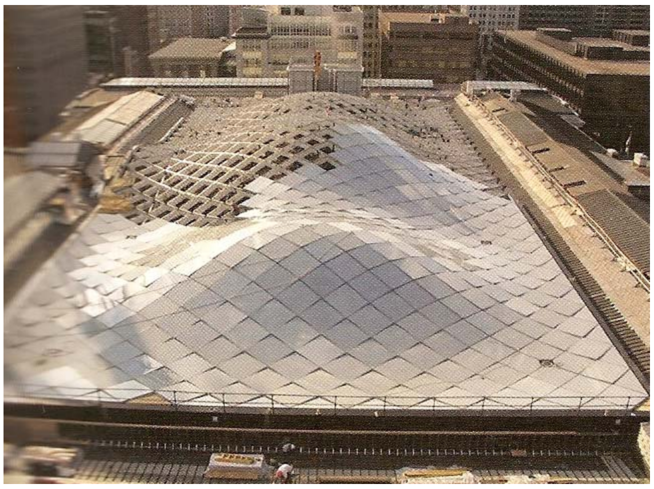
The final solution included a parallel iteration routine with other fabrication parameters such as glass sizes, properties, thickness and warping factors. The results of the structurally optimised form were then assessed against the various fabrication criteria with the use of an evaluation matrix which helped arrange the parameters in a hierarchical order of importance.

The significance of this project lies in its ability to resolve a complex geometry through a combined analysis of fabrication criteria and structural performance. The use of a the optimisation procedure not only created an optimum result in terms of performance, but it also helps determine a cost-effective solution for fabrication, where multiple variations of the proposal were analysed against one another to achieve the best outcome.

Great Court Roof, Foster and Partners

Another distinguished example of dynamic relaxation algorithms in architectural design is the Great Court Roof of the British Museum developed by Chris William in collaboration with Foster and Partners and Buro Happold. The shape of the roof is a double curved surface which was calculated using complex equations in order to locate the points on the surface forming the steel grid structure.

The roof is supported around the circular Reading Room inside the museum and on the external rectangular boundary. The construction of the roof comprised of a steel grid of triangular members which were glazed with flat glass panels covering an area of 70x100 meters. The process undertaken to create the final tessellation of the roof consisted of two stages; first the tessellation algorithm was produced by dividing the area of the roof with equally spaced points between the internal circular reading room and the external rectangular boundary. These points are then joined, and the created radial lines are again divided into varying number of equal segments. Although this methodology created the initial structural grid, the second stage of the algorithm aimed to remove the discontinuities between the lines through dynamic relaxation. In the Relaxation process a fictitious force is applied to each node based on the position of the neighbouring nodes. This process is repeated iteratively so that the nodes move around on the surface until the overall force is minimised and a state of equilibrium is reached.



In this project the procedure uses a top down optimisation, which requires an understanding of the mathematical equations that specify the given surface. Unlike the Bergen national Academy the overall geometry of the Great court roof was limited to the initial form, due to the tight restrictions of the surrounding context. Therefore the relaxation process was used to modify the arrangement of components making up this geometry and did not affect the final shape of the roof.

Figure 2. Construction of the great court roof (Littlefield 2008: 19)

Although the algorithmic process undertaken for the great court roof was significant for the analysis and design of the proposed geometry, the final result consisted of approximately 6000 individual members (Littlefield 2008: 19) that were welded, with a limited number of repetitions of the node types and panel sizes. Therefore a lengthy construction sequence was necessary to assemble each frame member to the correct specification using skilled labour, and in turn increasing the cost of fabrication.

2.2 Particle-Spring Systems

Although Particle systems are often used in computer graphics to simulate complex physical phenomena such as smoke and water, they have also recently been incorporated in architectural design as a tool for form finding using digital simulation.

Axel Kilian and Ochsendorf (2005) have used particle spring systems in the development of "*CADenary*", a three dimensional design and analysis tool which allows users to find structural forms in real time. In this research the authors use four examples including a catenary, a square mesh, a 3D cathedral structure, and a free-form grid shell in order to illustrate the potential for structural form finding with particle spring systems.

The CADenary tool allows the user to create geometry consisting of nodes and spring connections. The particle nodes are assigned a mass and can be fixed in space to create various support conditions. This geometry is then subjected to gravitational force which deforms the structure based on the applied gravity and the self-weight of the members. Once the initial parameters have been defined, the particles are free to move around until an equilibrium position is found using "Runge-Kutta solver" and the resultant particles are held in place by the deformed springs.

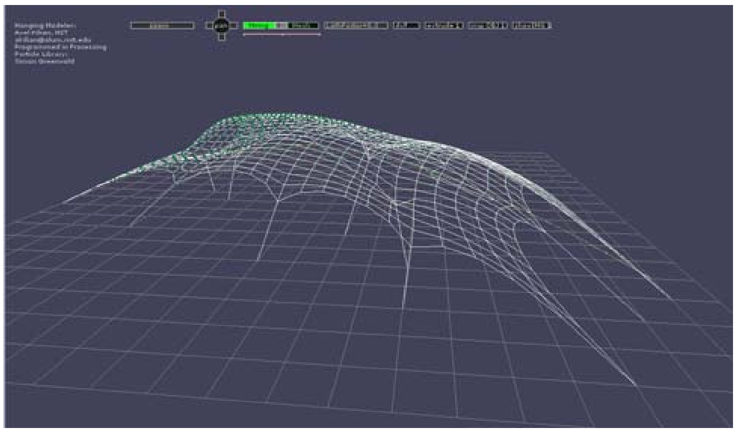


Figure 3. CADenary environment (Kilian 2005)

Another use of the system is that it can be used to explore hanging chain models as implemented by Antonio Gaudi by simply reversing the model in the horizontal plane so that the inverse geometry is evaluated as compression elements under gravity.

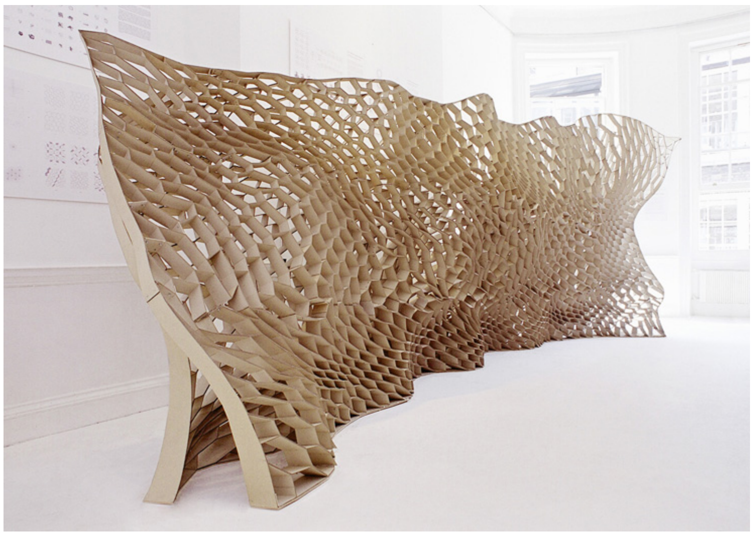
In general there are many analysis tools in design, which are often used for refining forms; however these analysis tools are rarely capable of exploring and creating new structural forms. The advantage of a spring system is that it allows the user to interact with the program as it is running thus allowing for the invention and creation of new forms rather than just analysis. This interactive environment provides a structural evaluation setup into the design process and thus heightens the "intuitive understanding for structural behaviour of complex forms at the early design stage." (Kilian 2005) The significance of this methodology is that it provides flexibility to any 3d model, and a capability to change various criteria such as member length, mass and strength so to analyse and modify designs accordingly.

2.3 Honeycomb structures

Honeycomb Morphologies, Andrew Kudless: 2004

Honeycomb Morphologies is a research project carried out by Andrew Kudless as part of a MA dissertation in Emergent Technologies and Design at the Architectural Association, which focuses on a development of a "polymorphous cellular structure" (Hensel 2006: 84) able to generate various geometric envelopes that can be easily fabricated for architectural applications.
In an aim to develop an "integrated and generative design strategy"(Hensel 2006: 84), the key objective was to develop a material system that would demonstrate the same level of integration found in natural material systems. The final algorithm produced a honeycomb system which could be incorporated into architecture by combining aspects concerning design, performance and fabrication. This entailed the development of a programmatic tool which generated a honeycomb grid on any given NURBS surface, whilst maintaining a construction logic that could adhere to the changing modulations of the geometric and material properties.

In the case of the honeycomb morphologies project three particular aspects needed to be addressed for the construction of a large scale prototype; these included a topological continuity to maintain a hexagonal configuration throughout, the generation of planar elements that could be easy laser cut with specific size constrains and material properties, and thirdly the assembly logic which needed to organise and label components to verify the correct construction sequence.



The constructed prototype and program was able to create a double curved surface comprising of hexagonal cells, with various shapes, sizes and depth. The resultant differentiation in the honeycomb structure provides the necessary capacity for adaptation to specific structural and environmental conditions whilst providing a technique that is buildable within the constraints of available fabrication technologies.

Figure 4. Constructed prototype of honeycomb system (Hensel 2006: 84)

The virtue of this project lies in its ability to provide a mechanism for the construction of free form surfaces of various cell densities and performance criteria. The integration of the construction logic is a fundamental quality of this research, which purposely provides versatility in geometry for greater flexibility, eliminating the traditional post-rationalisation procedure in the construction of complex forms. Therefore the algorithm does not seek to evaluate the form in any way, rather provides the necessary parameters which can be altered after future analysis.

Water Cube, PTW architects 2006
The design of the Water Cube swimming hall built for the 2008 Beijing Olympics by PTW architects and Arup provides another point of reference for cellular structures, which was resolved through a combination of form finding and optimisation techniques. The geometry of the space filling polyhedra that compose the three-dimensional Vierendeel superstructure was derived from the Weaire-Phelan structure, which denotes the most efficient way of space filling with equal sized cells and minimal surface area. This system is closely linked to the geometry that can be found in natural systems such as cells, crystals and soap bubbles, and is the driving concept for the organic looking array of hexagonal and pentagonal cells making up the structural system.

In order to achieve an optimum result in terms of structure and material, flexibility in the corresponding program allowed for control of the cell sizes, as well as the angles and number of the connections, so that the these variables could be easily adjusted to create the necessary strength and permeability. The optimization process for the structure was developed through "Strand7 structural FEA software," (Arkinstall 2008) which helped minimise the overall steel usage, by iteratively changing the steel sizes and performing an analysis until an optimum solution was achieved. Form-finding was also used for the ETFE pillows of the external cladding to determine the initial geometry of the foils, using a dynamic relaxation algorithm. This process helped find the minimal surface area between space frame boundaries combined with surface pressure so that the optimal final position could be obtained. The ultimate structure consisting of 22,000 steel tubes and 12,000 nodes (Arkinstall 2008) provides an open three-directional space frame system that is robust and ductile.

The design and construction of the water cube is a significant point of reference for this investigation. The use of space filling polyhedra constructed out of space frame technology provides the required level of structural integrity, whilst maintaining an economic use of material, and providing a visually and technically interesting structure. Furthermore, the significance of the cellular configuration aids in the provision of a lightweight structure capable of an omni-spreading of load that is inherent of many natural systems.

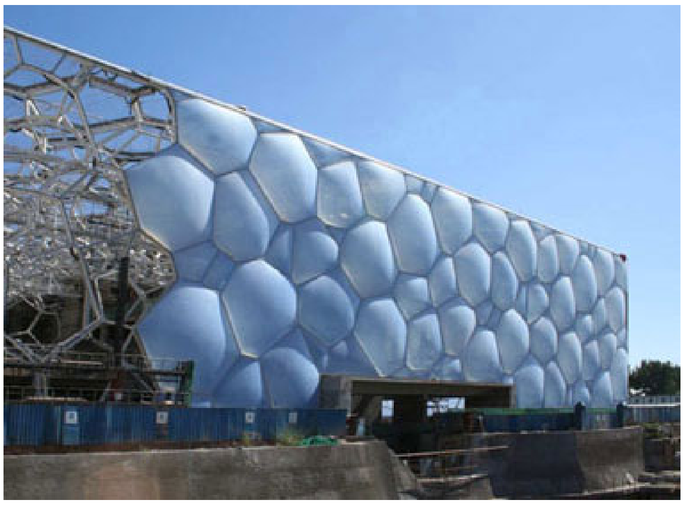


Figure 5. Watercube (Arkinstall 2008)

3.0 Method

The methodology outlined in this section consists of three steps that needed to be addressed in the construction and optimisation of an arbitrary curved surface. The initial phase of the algorithm deals with the subdivision of a NURBS surface into a series of hexagons forming a honeycomb mesh. This configuration utilises a particle spring system and a top down methodology to obtain the surface points of any given topology.

In an attempt to minimise the level of variation between the spring lengths, dynamic relaxation was used in the first stage of the optimisation to update the position of each node, making up the honeycomb grid until an equilibrium state was reached for the entire network. The resultant particles were then released from the surface and the final deviation was calculated against the number of specified spring lengths. In an attempt to provide a cost effective space frame construction whilst maintaining least deviance from the specified surface, it was important to find a solution that provided a minimum variation of spring lengths throughout.

The code was developed using Processing Programming language which provided an object orientated methodology that would allow easy manipulation of the NURB surface and the comprising structural members.

3.1 Description of NURBS surface and honeycomb grid

The initial setup of this project was concerned with the tessellation of an arbitrarily curved surface, where creation and manipulation of the overall geometry is defined under the specification of non-uniform rational b-spline surface (NURBS). Due to their ability to create endless possibilities of geometric forms, NURBS are often used in design for the creation of free form structures and products.

The form of these surfaces is governed by the position of the control points, which provide an easy procedure for manipulation of the overall geometry. This is due to the fact that each control point "has an associated weight, which determines the extent of its influence over the curve." (Kolarevic 2003 :18) So therefore the displacement of each control points has an effect on the weight of that point which in turn affects the corresponding curve and the overall shape of the surface. The control points are governed by 'basis' functions which are associated with the ratio of two polynomial equations. Each time a control point becomes active the basis function affects the curves which are associated with the particular control point and these curve sections are "delimited by knots"(Kolarevic 2003 :18). In short the knot vectors are used to divide the parametric space into intervals, and these knot spacings can be altered to produce non-uniform NURBS by providing unequal knot intervals. In this investigation a uniform NURBS has been used to minimise the variation of initial spacings for the tessellation algorithm. As well as Knots and control points another factor in a NURBS function is the inclusion of the 'degree' parameter which is associated with the proximity of the control points to the curve. A high degree function creates smooth curve segments whilst a low degree function would create a surface with straight line segments and control points which are situated close to the curve.

Another important aspect of NURBS is the definition of a U V coordinates system which in the words of Kolarevic is explained as a "local parametric space, situated in the 3d Cartesian geometric space". This space is two dimensional and denoted by U and V in order to distinguish them from the three dimensional Cartesian coordinate system made up of X, Y and Z. In relation to this project, the given NURBS surface was defined as a set of 3d points which could be adjusted by utilising the UV system in order to calculate new positions on that surface. The use of the UV coordinates was an important step for the dynamic relaxation process, where the nodes become free to move to any point on the surface *(refer to 3.2 for a description of dynamic relaxation algorithm)*.

Consequently given the mentioned parameters, calculating the points on a NURB surface require the inclusion of the following values;

a.) An array of knot vectors in the u and v direction with equal spacing, where the number of knots is associated with the number of control points;
u_Knots (x_i) = { 0.0, 0.125, 0.25, ... 0.875, 1.0 };
v_Knots (y_j) = { 0.0, 0.125, 0.25, ... 0.875, 1.0 };

b.) Two rows of control points m and n where;
0<= i >= number of u_control_points (m)
0<=j >= number of v_control_points (n)

c.) The degree of the curve in both u and *v* direction which is equal to;
u_degree (p) = u_knots.length - u_control_points - 1 ;
v_degree (q) = v_knots.length - v_control_points - 1 ;

d.) Two polynomial equations i.e. basis-U ($N_{i,p}$) and basis-V ($N_{j,q}$), where the shapes of the basis functions are determined by the knots vectors x_i, and defined by the following formula for the u-direction and alike for the v-direction.

$$N_{i,1}(u) = \begin{cases} 1 & if x_i \leq u < x_{i+1} \\ 0 & otherwise \end{cases}$$

$$N_{i,p}(u) = \frac{(u - x_i)N_{i,p-1}(u)}{x_{i+p-1} - x_i} + \frac{(x_{i+p} - u)N_{i+1,p-1}(u)}{x_{i+p} - x_{i+1}}$$

Subsequently the final calculation of the NURBS curve is determined by a parametric equation which calculates the points on the curve for u and v respectively.

$$P(u) = \sum_{i=1}^{m} N_{i,p}(u)P_i \quad and \quad P(v) = \sum_{j=1}^{n} N_{j,q}(v)P_j$$

Given m is the number of control points vertically and n is the number of control points horizontally, $N_{i,p}(u)$ and $N_{j,q}(v)$ are the B-spline basis functions with degree p and q, *and* P_i and P_j are the array of $m \times$ n control points, the resultant $P(u)$ and $P(v)$ define the points on the surface for a specific u,v location.

The surface points are therefore defined based on a set topology which calculates the position of each point in regards to the knot spacing, UV coordinates and the height of each control point. The code uses a double loop that calculates the NURBS equation for all the control points and returns a 3d vector containing the XYZ position of the points on the surface. These positions are dependent on the number of nodes required for the density of the tessellation. The density is altered through an additional double loop which determines the number of nodes to draw. These nodes are then calculated in relation to the NURBS function which draws them in accordance to their respective u and v location *(refer to section 3.3 for a description of the particle spring algorithm).*

The node class is fundamentally used as a guide for the connecting springs and overall mesh, the resultant honeycomb tessellation is derived through the transformation a regular 2d grid using a double array, where instead of equal positions in the horizontal *(a)* and vertical (b) rows, *b* is denoted by; *b =(a%2)*2* so that every other row is offset creating a triangular mesh. Two sets of nodes $\boldsymbol{n_{a,b}}$ and $\boldsymbol{n_{a,b+1}}$ are drawn using the mentioned triangular pattern to create the resulting hexagonal mesh.

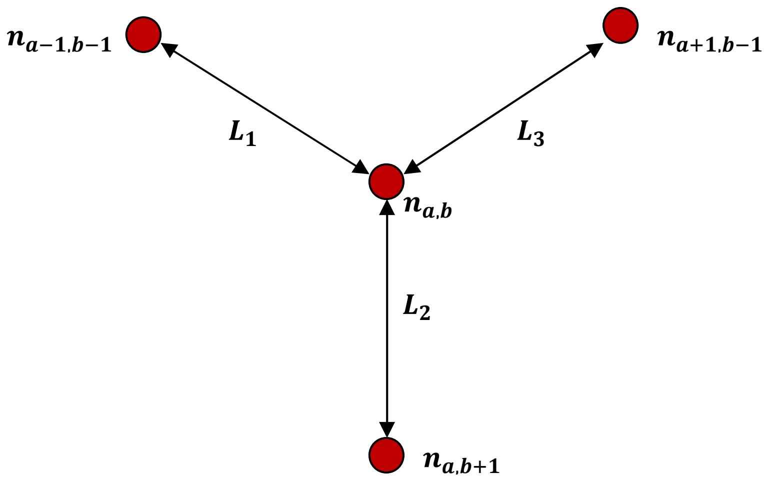


Figure 6. Node and spring arrangement

Once this grid has been defined the connecting springs, three of which are connected to each node, can be specified starting from each node $\boldsymbol{n_{a,b}}$ to the corresponding nodes, $\boldsymbol{n_{a-1,b-1}}$, $\boldsymbol{n_{a,b+1}}$, and $\boldsymbol{n_{a+1,b-1}}$ (*figure 6).* This way a regular hexagonal network of springs is achieved with flexibility in the total density by varying the number of nodes in the initial for-loop. *(More psuedocode of the honeycomb network and NURBS algorithm are presented in Appendix II.)*

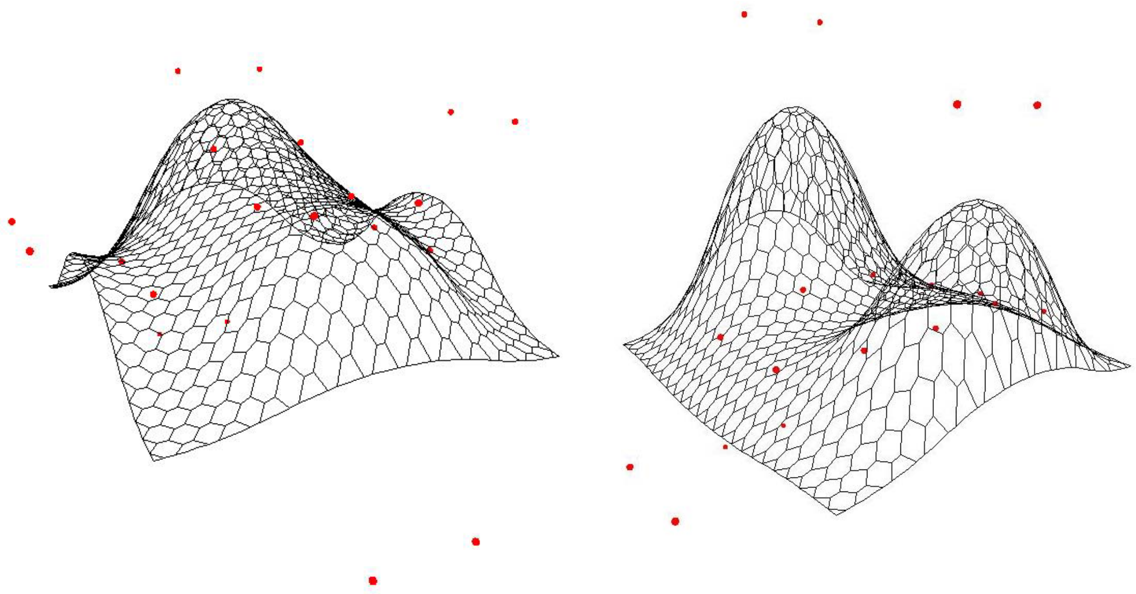


Figure7. Manipulation of the surface with control points

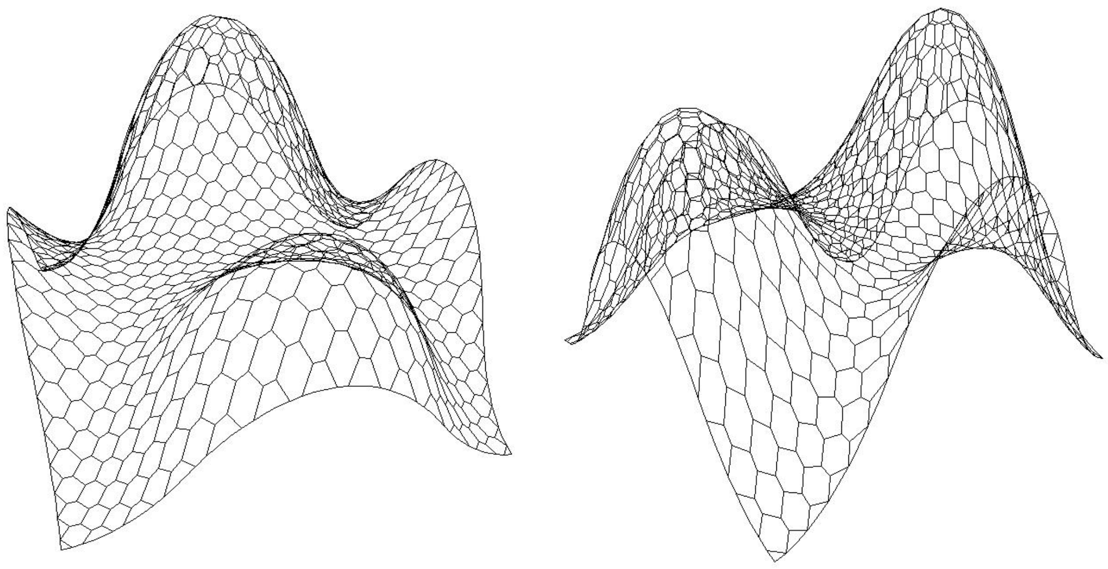


Figure8. View of the tessellated surface with honeycomb mesh

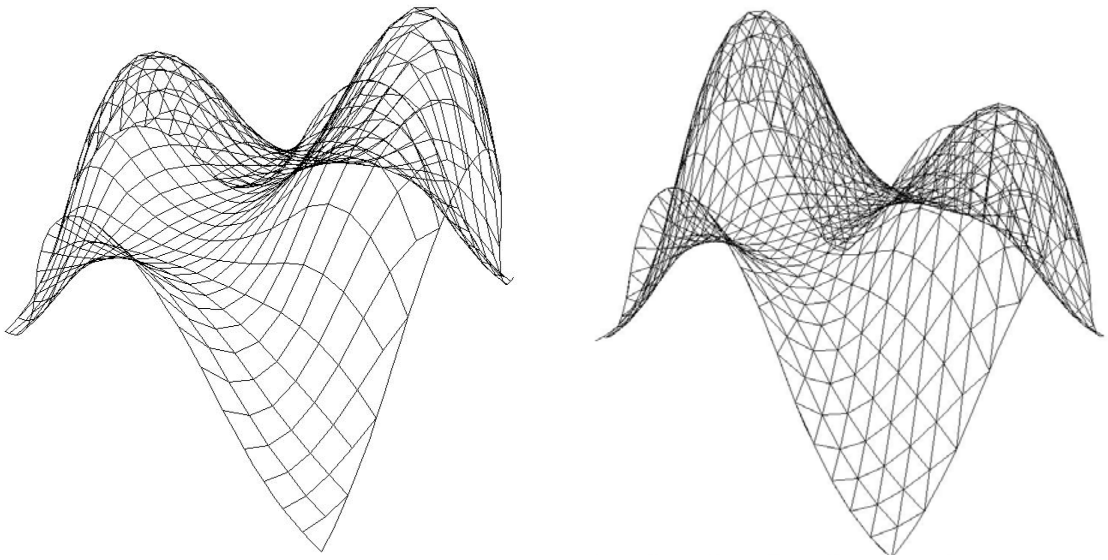


Figure 9. Initial tests with different tessellations

3.2 Description of dynamic relaxation

One of the main issues of a free form curve, derived through manipulation of the control points, is the creation of unequal panels in areas which are under greater transformation. For this reason dynamic relaxation was used to achieve a better distribution of nodes throughout the surface. The process consists of iteratively updating the geometry by altering the position of each node on the surface, thus affecting the interconnected links which represent various elements of the structure. In this way the code is able to optimise a defined mesh to reach a final minimum energy and an equilibrium state for the overall structure. The system is considerably simpler than what is considered to be a typical dynamic relaxation method, for example, there are no material properties such as elasticity or boundary specifications such as cables and beam components.

The relaxation process only affects the position of the nodes in the parametric space; therefore the nodes are free to move around on the surface through manipulation of their respective U and V coordinates. This 2-d coordinate system is derived through a double loop of $\boldsymbol{a}$ and $\boldsymbol{b}$, which defines the values of $\boldsymbol{n_{a,b}}$ (u) and $\boldsymbol{n_{a,b}}$ (v) in the first iteration. In order to change the spring lengths of each hexagon, it is important to clarify which way the corresponding nodes need to move. The issue with the mentioned tessellation is that each node is connected to three springs and therefore the movement of that node will need to be calculated as the average position of the three surrounding nodes. The UV coordinates of node $\boldsymbol{n_{a,b}}$ are therefore iteratively updated based on the position of the nodes $\boldsymbol{n_{a-1,b-1}}$, $\boldsymbol{n_{a,b+1}}$, and $\boldsymbol{n_{a+1,b-1}}$, by using the below formula.

$$n_{a,b}(u) = \frac{(n_{a-1,b-1}(u) \times L_1) + (n_{a,b+1}(u) \times \mathrm{L}_2) + (n_{a+1,b-1}(u) \times \mathrm{L}_3)}{L_1 + L_2 + L_3}$$

This formula is repeated for the calculation of the V-coordinates of node $\boldsymbol{n_{a,b}}$, and these new UV-coordinates are fed into the NURBS equation and re-calculated in relation to the defined surface. The lengths of the springs are used as weights in the equation and determine which direction gets the majority in the optimisation. This is repeated iteratively until an equilibrium position for all of the nodes is achieved. During this process the boundary nodes are fixed in place so that all other nodes move around within the confined region of the surface. The resultant process provides a dynamic computer model which manipulates the hexagonal mesh as if it were a membrane surface.

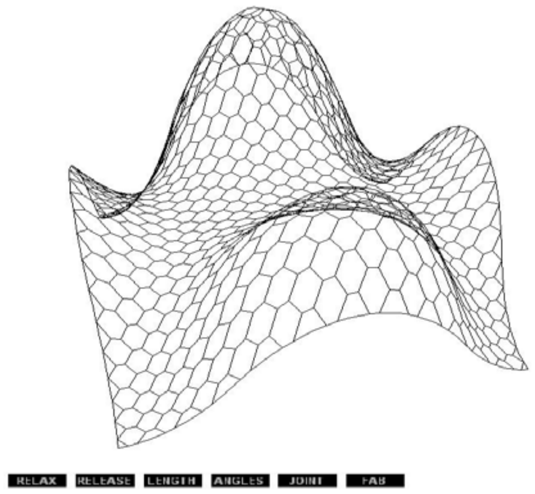


Figure 10. Tessellation before Relaxation

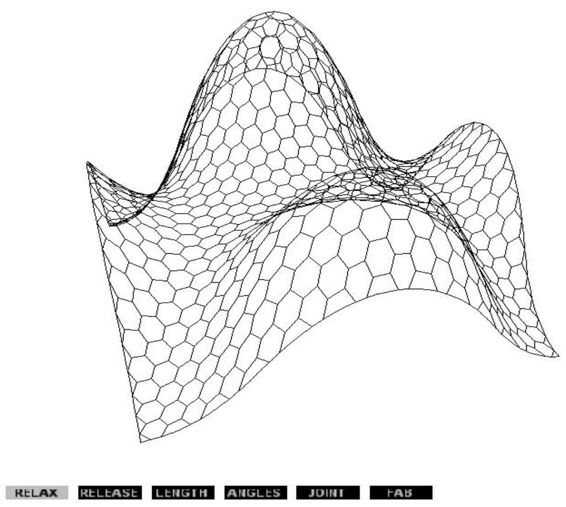


Figure 11. Tessellation after Relaxation

3.3 Description of the particle-spring system

The inclusion of a particle spring systems algorithm provides a means for the adjustments of the structural frame members comprising the honeycomb mesh. The system consists of a series of particles, which act as the nodal points of the space frame topology, and a set of springs which connect the nodes via the specified tessellation pattern. Each particle has three parameters, the XYZ position in the Cartesian space, a UV position in the parametric space, and a direction vector, which determines the movement of the particles. Due to the nature of the thesis proposal, parameters such as gravity, mass and damping which are often used in spring system algorithms were not included in the programmatic setup.

The preliminary positions of the nodes are derived from the mentioned NURBS algorithm, and their movement is determined as the sum of the current position ($\boldsymbol{n_{a,b}.pos}$) and direction vector ($\boldsymbol{n_{a,b}.dir}$), which is initially set as zero. At each iteration, the movement of the nodes are established depending on the ratio ($\boldsymbol{g}$) of the actual ($\boldsymbol{L}$) to ideal spring length ($\boldsymbol{r}$), which is used to scale a new direction vector($\boldsymbol{dir_1}$). This vector is the direction between the two corresponding nodes $\boldsymbol{n_{a,b}}$ and $\boldsymbol{n_{a,b+1}}$, and is either added or subtracted from the initial direction vector ($\boldsymbol{n_{a,b}.dir}$) depending on the initial length of the spring and the target value. This process is repeated iteratively until the direction vectors are close to zero and therefore reaching a state of equilibrium. *(More details about the movement of the nodes and the direction vectors are presented in Appendix III).*

The spring class holds three parameters, the start and end locations (determined by the position of $\boldsymbol{n_{a,b}}$ and $\boldsymbol{n_{a,b+1}}$) and the distance between $\boldsymbol{n_{a,b}}$ and $\boldsymbol{n_{a,b+1}}$, which determines the length of the spring ($\boldsymbol{L}$). At each iteration, the spring lengths *($\boldsymbol{L}$)* are compared against a series of ideal lengths ($\boldsymbol{r}$), which are determined prior to optimisation. The determination of these ideal lengths is discussed further in the results. Each pair of particles, corresponding to one spring, is examined in relation to their distance or spring length *($\boldsymbol{L}$)*, and this value is the guiding principle which drives the optimisation. The ideal length (**r**) determines if the state of the spring is either in compression, tension or equilibrium. So therefore as an example, if L<r then the state of the spring is in compression which pushes $\boldsymbol{n_{a,b}}$ and $\boldsymbol{n_{a,b+1}}$ away from each other, else if L>r then the spring exerts a tensional force which draws $\boldsymbol{n_{a,b}}$ and $\boldsymbol{n_{a,b+1}}$ closer together.

Given that each node $\boldsymbol{n_{a,b}}$ is connected to three springs, in each iteration the lengths and relevant directions are calculated in order of the three spring types $\boldsymbol{L_1}$, $\boldsymbol{L_2}$,and $\boldsymbol{L_3}$ shown in *figure 6.* These springs are released from the surface one at a time and if their length is within a defined range then they are re-sized based on the ideal length of that range. Although the particles are free to move to any position in the Cartesian space, the nodes surrounding the edges of the surface are fixed in place to prevent excess deformation of the boundary springs.

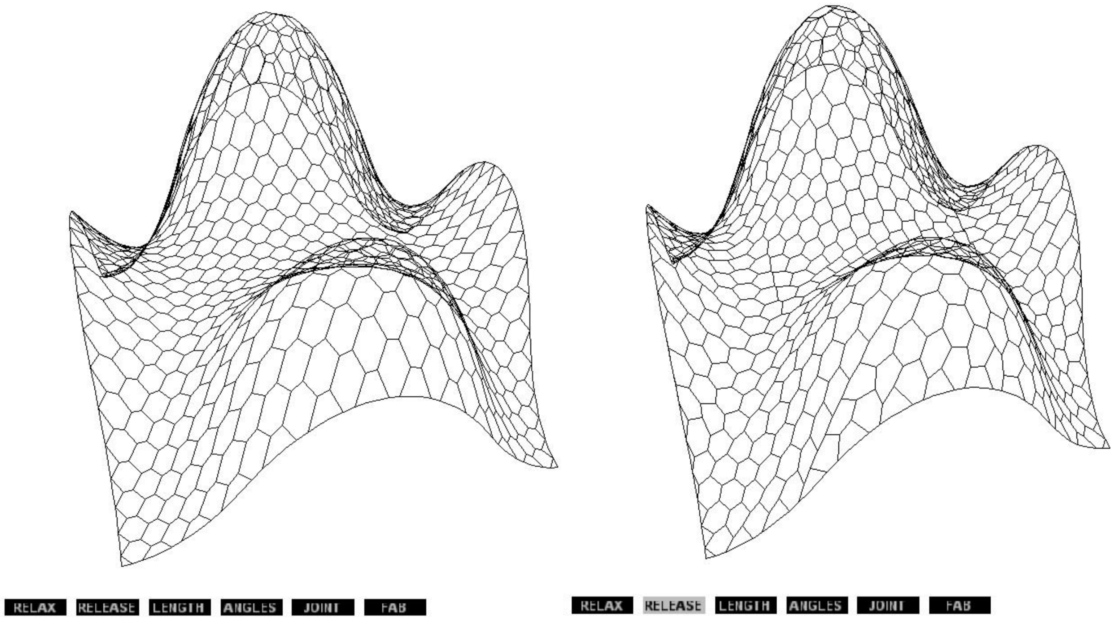


Figure 12 view of the springs before and after optimisation with 3 variations

In order to imbed the construction logic into the algorithm, the nodes are replaced with an appropriate geometry similar to that used for space frame connectors, so that the joints can be easily fabricated to the provide the exact angles for the assembly of the surface. A prototype of the joints is produced shown in figure13 which have been arranged in numerical order through an additional loop which translates the final nodes into a 2d grid.

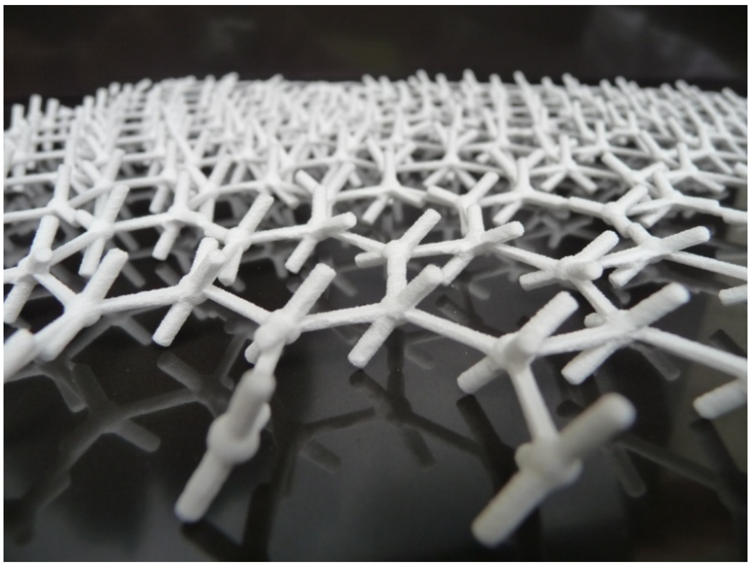


Figure 13. Prototype of the node connectors

3.4 Specifying the parameters for experimentation

The above methodology covers the basic rules necessary for the creation and manipulation of the honeycomb topology. Although informal observations of the algorithm show that dynamic relaxation is able to create a better distribution of points throughout the surface and thus reduce the variations of the spring lengths, it is still uncertain to what degree this has an effect on the overall range. Likewise the springs are also successful at reaching the defined lengths; however it is difficult to identify accurate length specifications which provide minimal deviation from the surface, without a more coherent understanding of the total length distribution. Therefore In order to obtain an optimum solution for the construction of such structures, it is necessary to define specific testable parameters that can help achieve a more consistent investigation of the algorithm.

The initial factor which needed to be addressed was the definition of a single NURBS topology which can be used to test the performance of the algorithm in more depth. The decision was made to define a single surface as opposed to testing multiple topologies due to the fact that the experimentation aims to pick up from the constructional phase of a design project and thus assumes a defined topology which needs to be resolved for fabrications. Given this topology, the density of the tessellation was the first parameter which had to be tested, this includes defining an optimum number of nodes which provide minimum variations of lengths whilst giving a cost effective solution. The results of this experiment coupled with the defined surface would inform the basis for the following investigations which aim to minimise the overall variation of lengths in the system.

Testing the affects of the particle spring algorithm requires an analysis of the deviations of each spring from the surface, therefore the XYZ Cartesian position of each node is another important parameter which has been evaluated for both dynamic relaxation and particle spring technique. As well as the positions of each node, the length of the springs also needed to be evaluated in terms of the total range before and after optimisation. This parameter is a fundamental criterion which informed the series of defined ideal lengths and ultimately the success of the entire algorithm.

4.0 Testing and Results

The testing of the algorithm was carried out in three stages; these include analysis of the tessellation, the effects of the dynamic relaxation and the reduction of the spring lengths. These results are finally evaluated in the concluding section to obtain an optimum outcome for the construction of a cost-effective solution. In order to define the precision of the experiments, it was necessary to set a tolerance for the spring lengths. For the purpose of this investigation, the system unit is defined as mm with a maximum tolerance of 0.1 mm. This value is used as the maximum error for the rest-lengths during the final optimisation as well as for distinguishing the number of variations between springs.

4.1 Analysis of the tessellation

The initial experiments were carried out using node samples ranging between 171 to 4851 nodes, which were tested against the resulting number of spring lengths. Nine categories of node samples were calculated in total, and the standard deviation of the spring lengths for each node category helped determine the most cost-effective density of the tessellation to use for further experimentation of the algorithm.

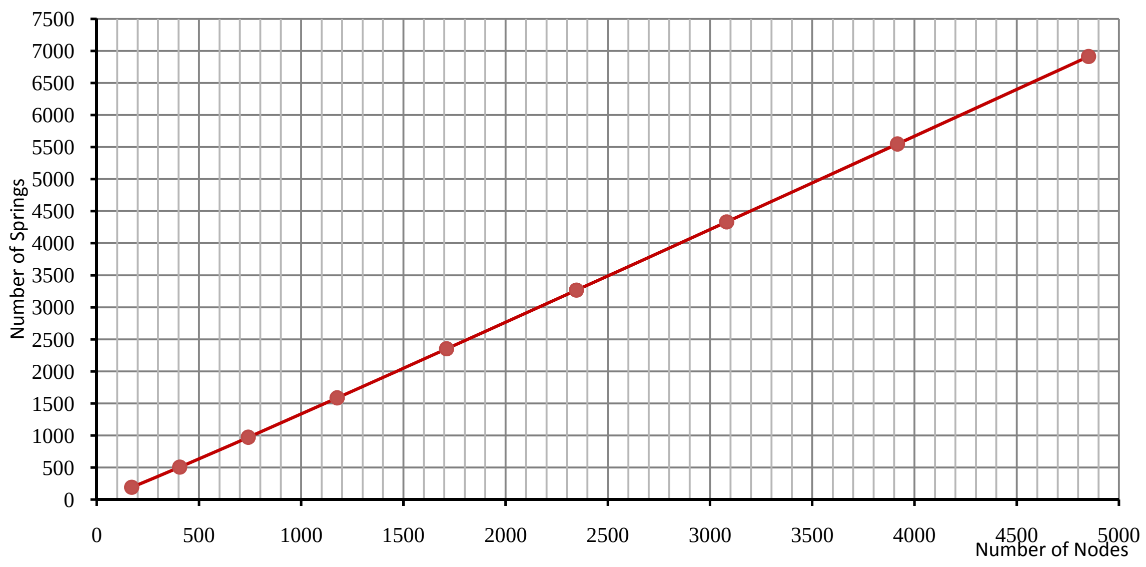


Figure 14. Number of springs depending on the density of the honeycomb mesh

As illustrated in *figure14* there is a linear increase in the number of springs to the number of nodes. Given that higher density results to increased number of springs and thus more material usage, we can assume that the cost of the tessellation is dependent on the number of nodes. An ideal density is determined by two variables, namely the cost and the level of variations between springs. Therefore in order to find an optimum density with minimum variations in size, the standard deviation of spring lengths has been mapped against the number of nodes (figure 15) to examine the behaviour of the varying spring lengths in relation to the density of the tessellation.

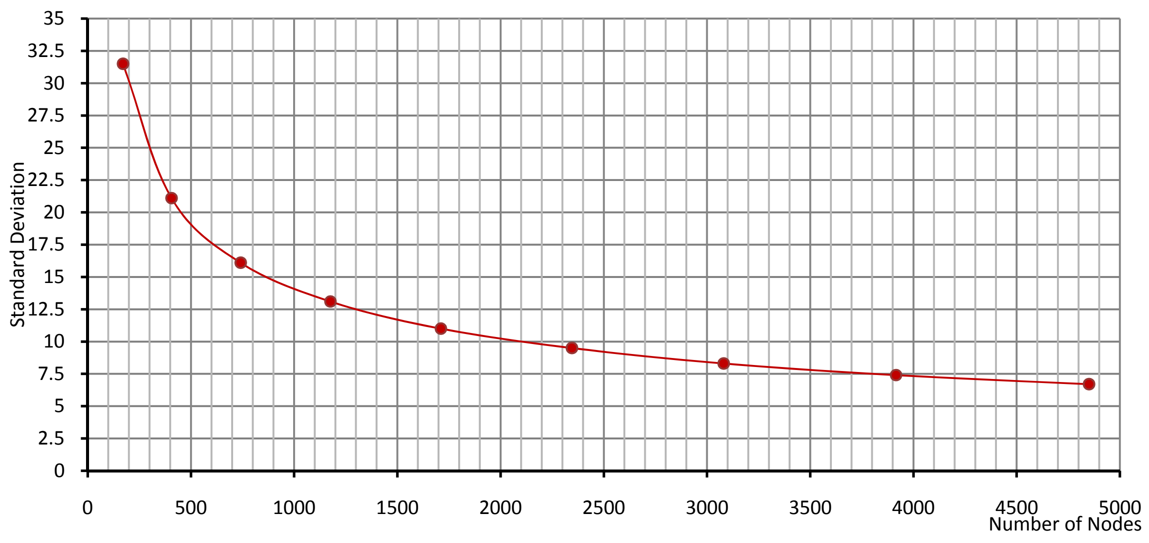


Figure 15. Standard deviation of spring lengths based on density of honeycomb mesh

As shown in figure15 the standard deviation of the spring lengths reduces with increased density. Although this is advantageous for achieving minimal variations in lengths, simply using a large density as the optimum solution does not take into consideration the issue of cost. Therefore in order to obtain the best resolution in terms of density and variation of lengths, figure 16 has been plotted to determine the intersecting point between these two variables.

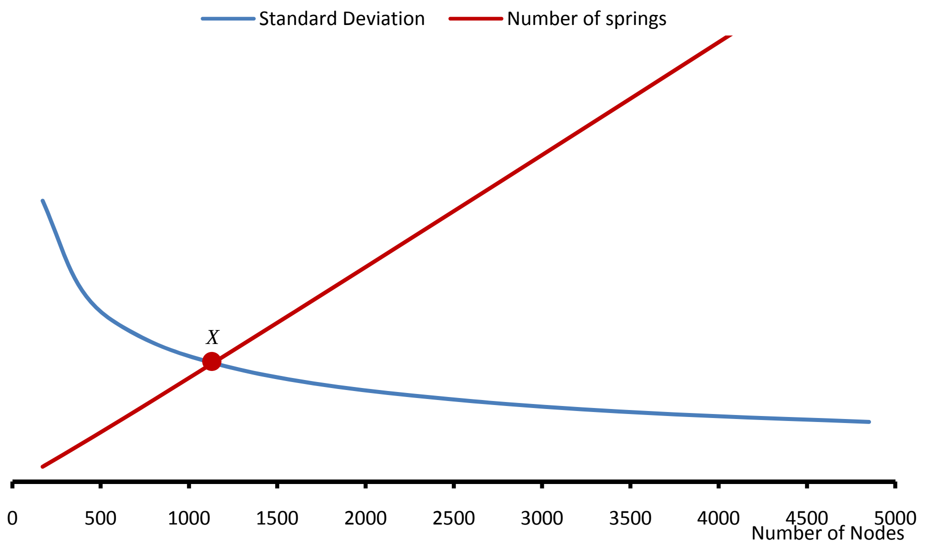


Figure16. Optimum cost based on density and variations of lengths.

The number of springs in figure 14 represents the linear increase of cost based to the number of nodes. This is compared against the standard deviation of spring lengths, and the resultant intersecting point “X”, at approximately 1150 nodes, denotes a possible optimum solution which satisfies both criteria for the construction of a cost effective topology.

4.2 Dynamic Relaxation

Having established a solution for the initial density of the honeycomb tessellation, the remaining tests have been conducted using this figure as the basis for experimentation.

Initial analysis of the spring lengths shows that out of 1587 springs there are 437 different variations of lengths. The aim of the dynamic relaxation algorithm was to reduce this value and create a more even distribution of points throughout the surface. The performance of the algorithm was tested by calculating the movement of each node in X,Y space, as well as the reduction in variations of lengths and the differences in the spring range.

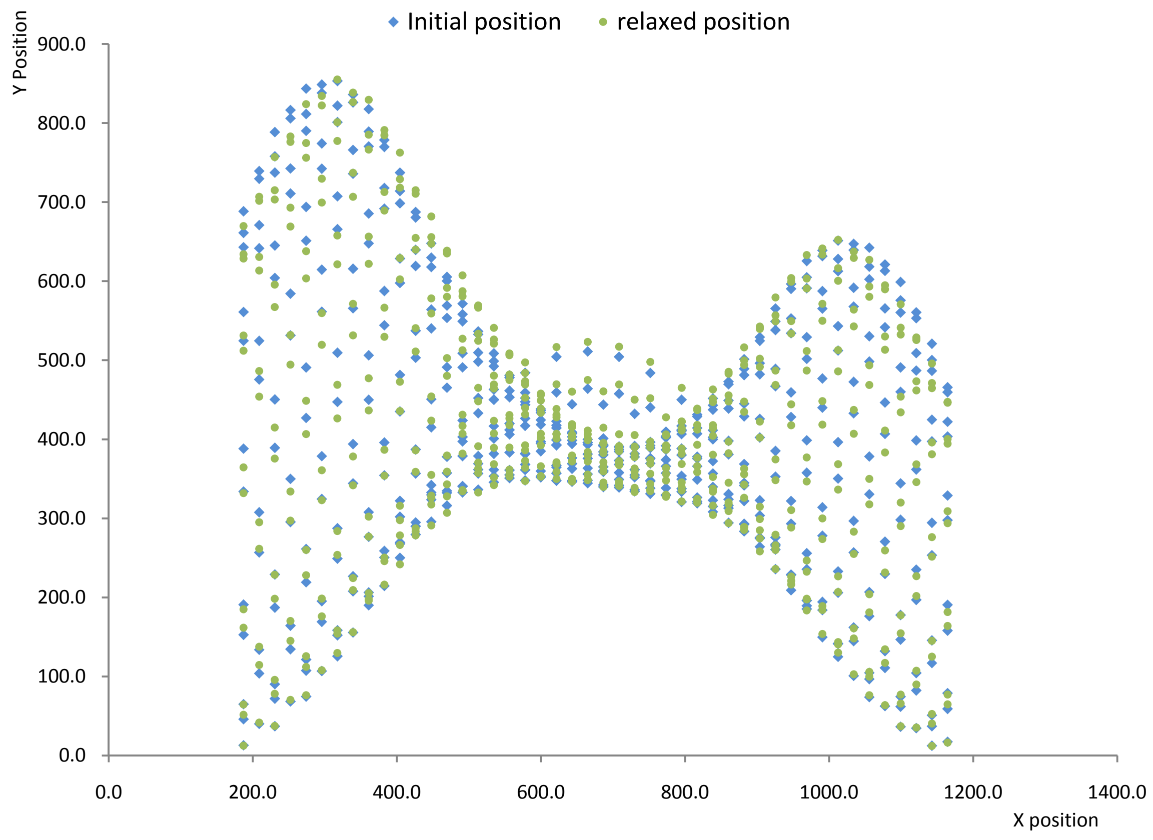


Figure 17. Movement of Nodes after relaxation

The results of the algorithm, as illustrated above, show that the node movement is higher in areas which are under greater transformation and which have longer initial spring lengths. Analysis of the lengths post relaxation shows that the number of variations was reduced to 316 as opposed to the initial 437, which is approximately a 28% reduction in the number of total variations, shown in figure 19.

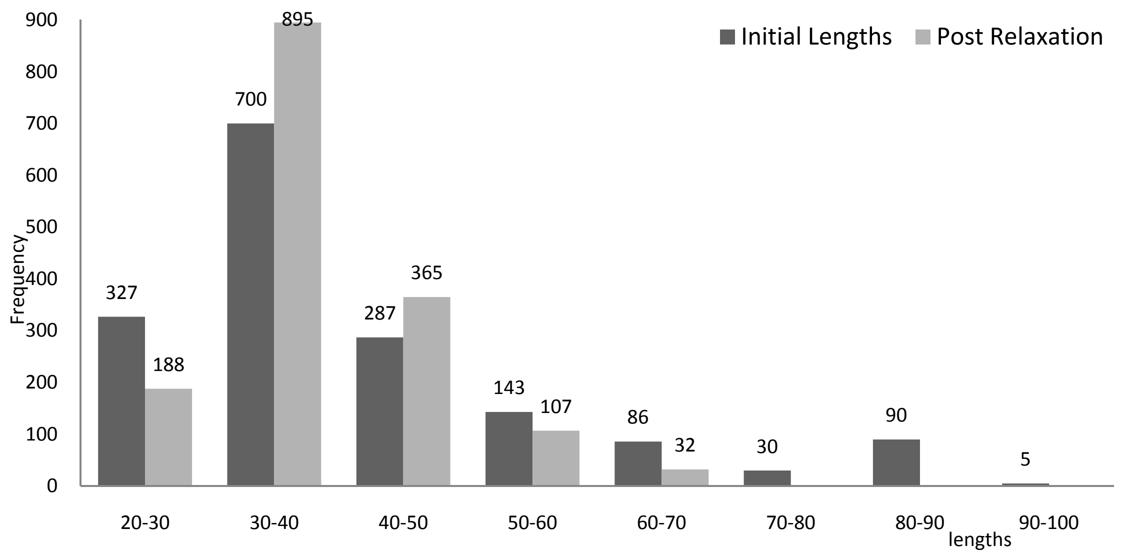


Figure 18. Frequency of spring lengths before and after relaxation

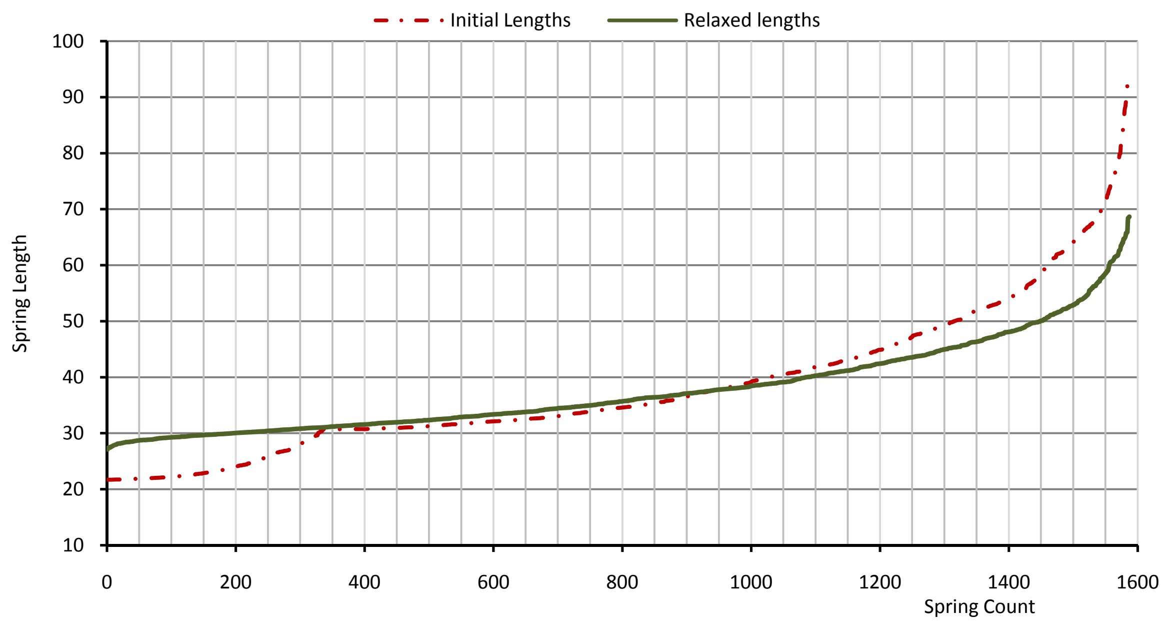


Figure 19. Range of spring lengths before and after relaxation

By comparing the new spring lengths with the initial values, we can see in figure 19 that the range of the springs is also decreased by approximately 30 units post-relaxation. The behaviour of the new spring lengths roughly exhibits a linear increase in the first 1200 springs with a much faster growth in the remaining 400. This distribution of lengths is important for the determination of new rest lengths required for the next optimisation procedure, as not only does it indicate the range of the total spring lengths, but it is also useful to determine the mean values for differing classes of lengths.

4.3 Optimisation of spring lengths

Having reduced the level of variations and total range through the previous experimentation, the following investigations were initialised from the relaxed positions of the nodes in order to work with a lower range of spring lengths. The Remaining tests were conducted to find an optimum number of ideal spring lengths which would produce minimal deviation from the given topology. The ideal rest-lengths were achieved through two different methodologies as explained below;

i.) Sort method; dividing the spring lengths;

The first tests were conducted to reduce the 316 variations by experimenting with 3, 5, 7 and 9 variations of ideal spring rest-lengths. These lengths were derived through sorting the actual spring lengths into numerical order, and then based on the number of variations required, the spring lengths were divided into groups, and the mean length of that group was then taken as the ideal length. For example, to obtain 3 variations of lengths, the mean of the smallest 529 lengths was calculated as the first ideal length, the mean of the middle 529 lengths were calculated as the second ideal length and the mean of the largest 529 lengths was used as the final ideal length.

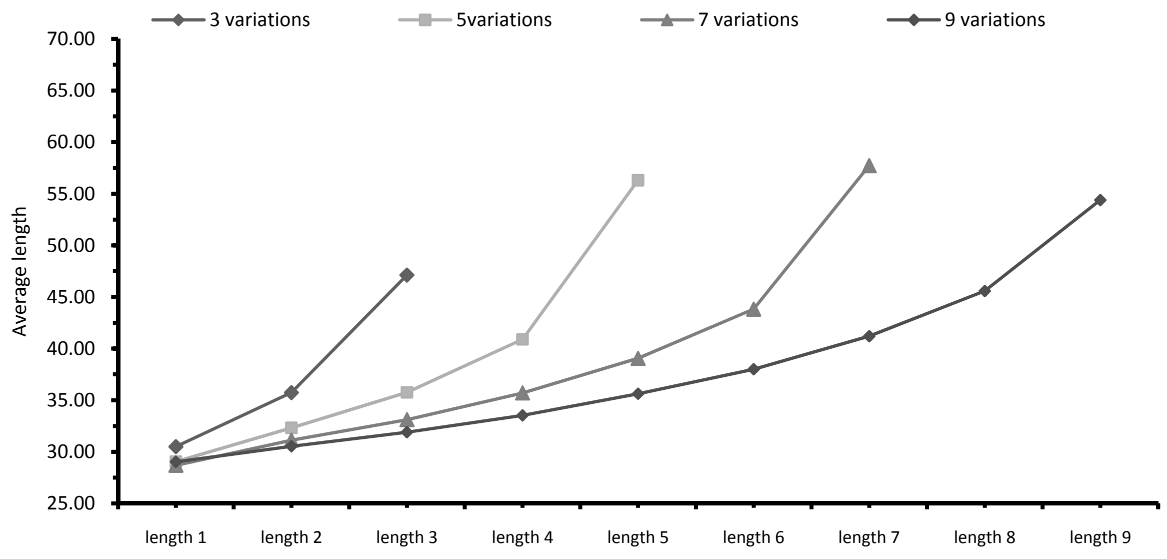


Figure 20. mean spring lengths for the various tests.

The performance of the algorithm was tested by measuring the deviation of each node in X Y and Z Cartesian space, and the final figure is taken as the average value of all three coordinates. By defining +1 or -1 as an acceptable level of deviation for each node, it is possible to see the intensity of deformation of the springs under greater distortion.

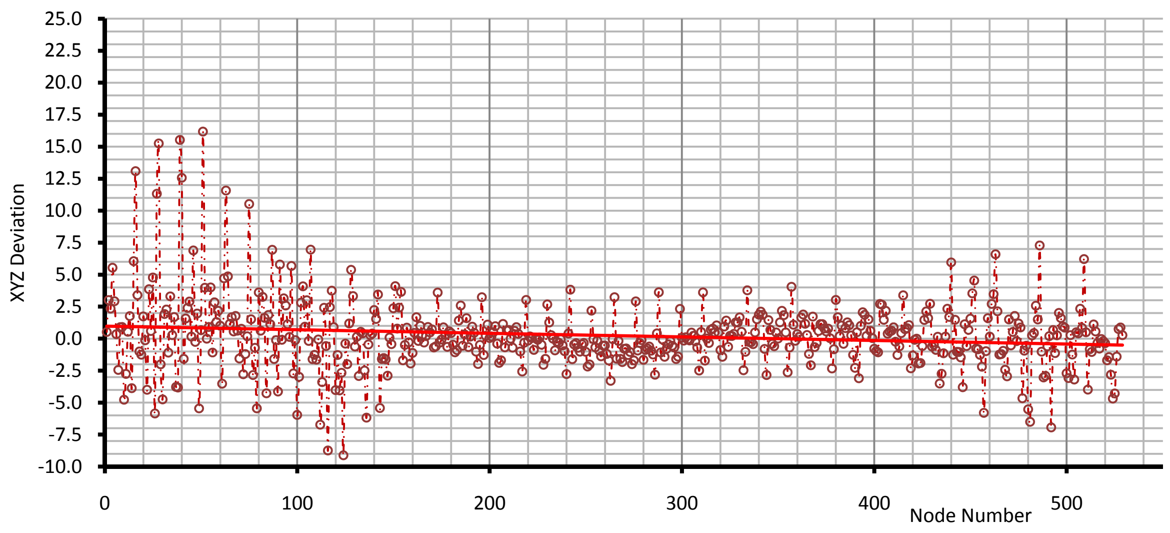


Figure 21. Deviation of springs with 5 set length

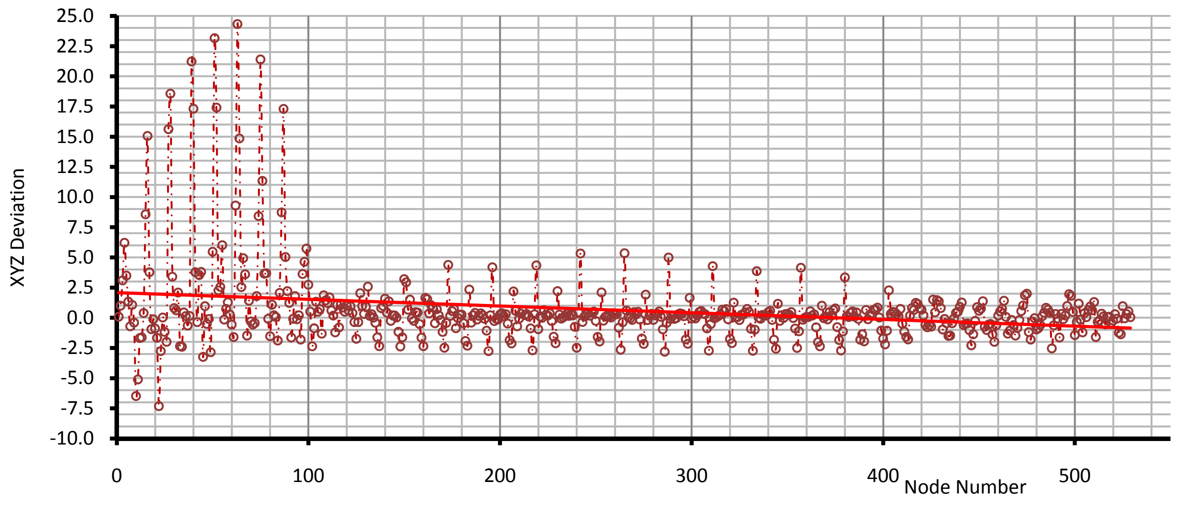


Figure 22. Deviation of springs with 7 set lengths

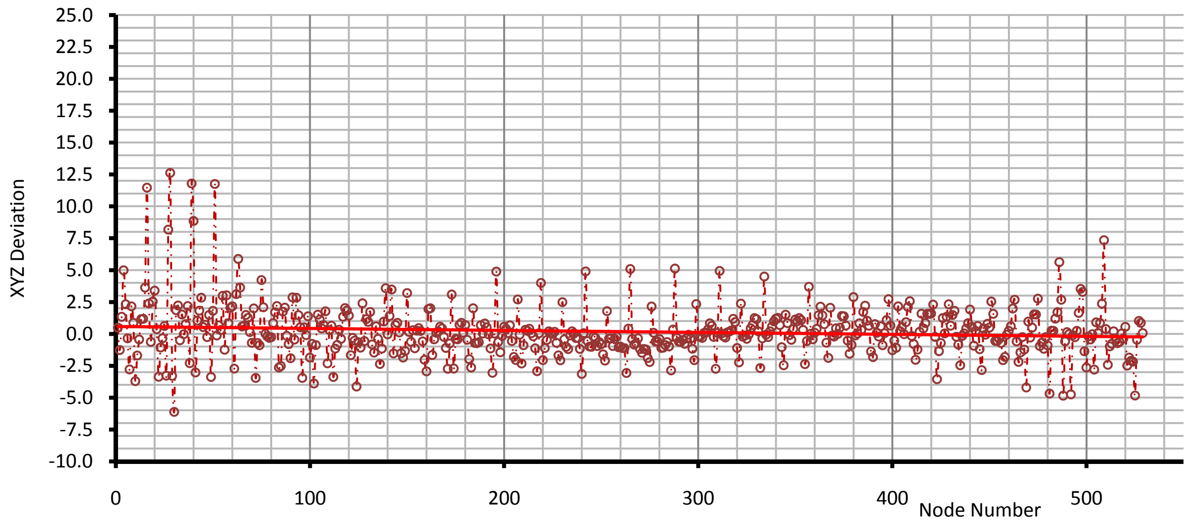


Figure 23. Deviation of springs with 9 set lengths

The results from the graphs presented above show that the level of deviation for each node decreases with a higher number of defined spring lengths. However, even with the maximum number of defined lengths shown in figure 23, a noticeable level of deformation is still present in areas with the largest initial spring lengths, which are evident between the first 100 nodes. Nonetheless the level of deviation appears to be relatively consistent throughout the remaining areas with little deformation of the tessellated grid.

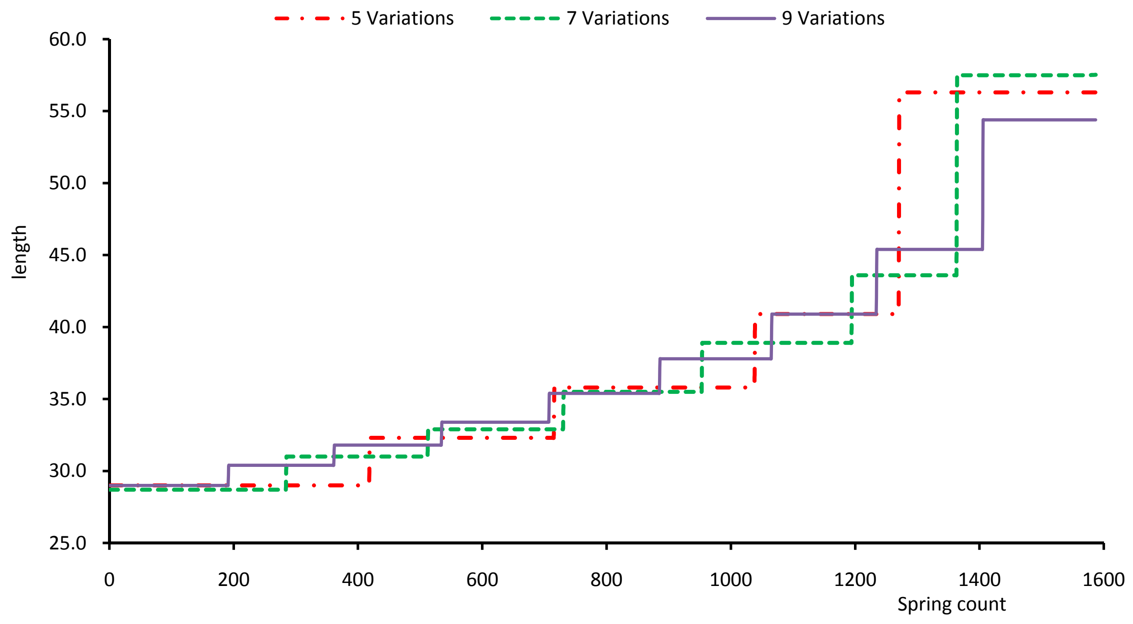


Figure 24. Final resting lengths of springs for all 4 variations using 'sorted' method.

In order to test the accuracy of the optimisation technique, the final rest lengths of the springs are presented in figure 24 which shows that during all 4 tests, the springs were successful at achieving the target rest-lengths.

ii.) Range method; dividing the spring range

In order to improve the level of deviations obtained from the previous experimentations, a second methodology for defining ideal rest-lengths was conducted. This entailed calculating the actual range of the spring lengths, and then dividing this figure by the number of variations required. The mean spring length of each category is then used as the final rest-length.

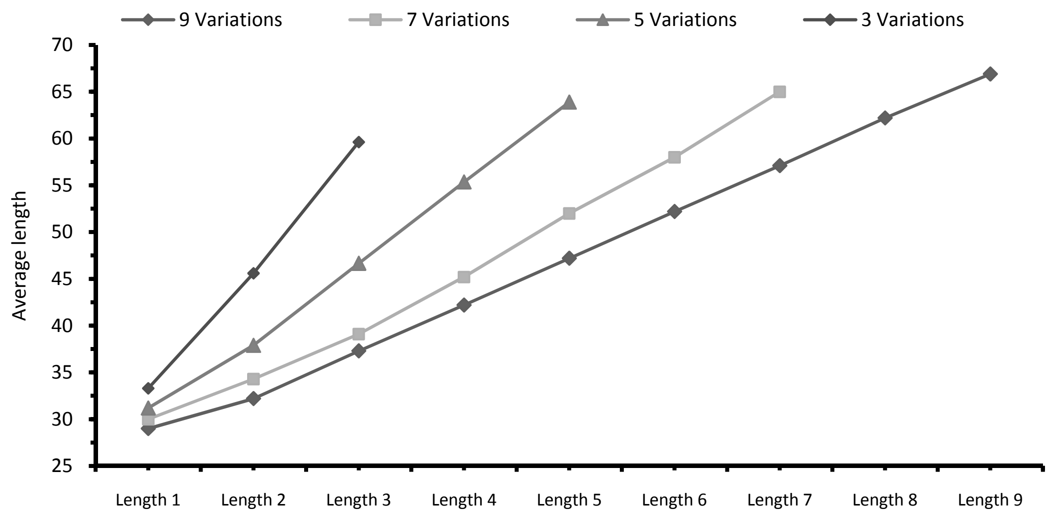


Figure 25. Average lengths based on the number of variations

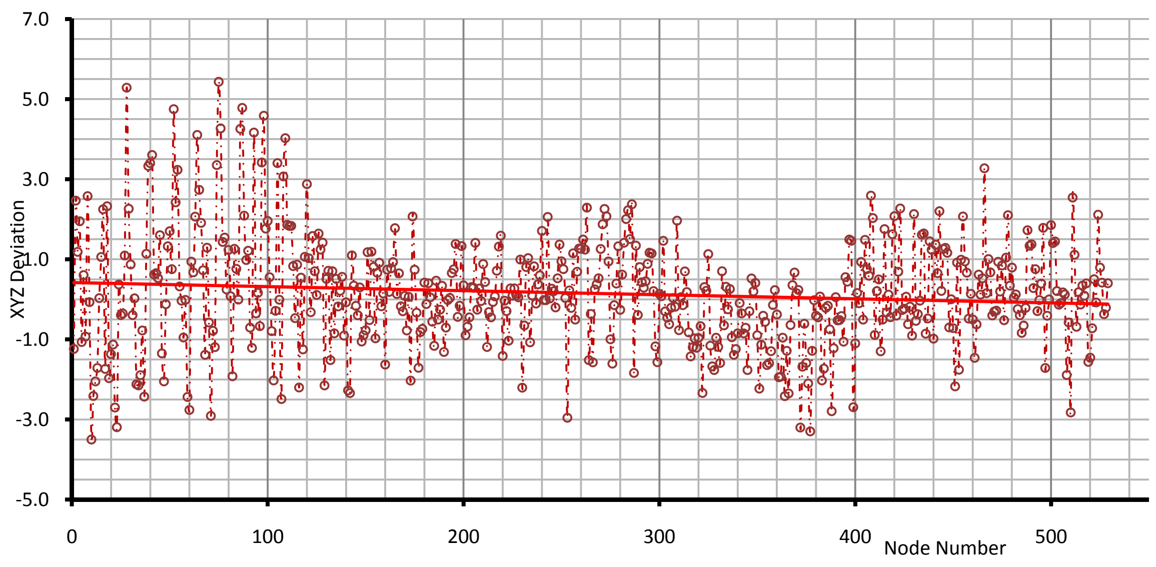


Figure 26. Deviation of springs with 5 set lengths

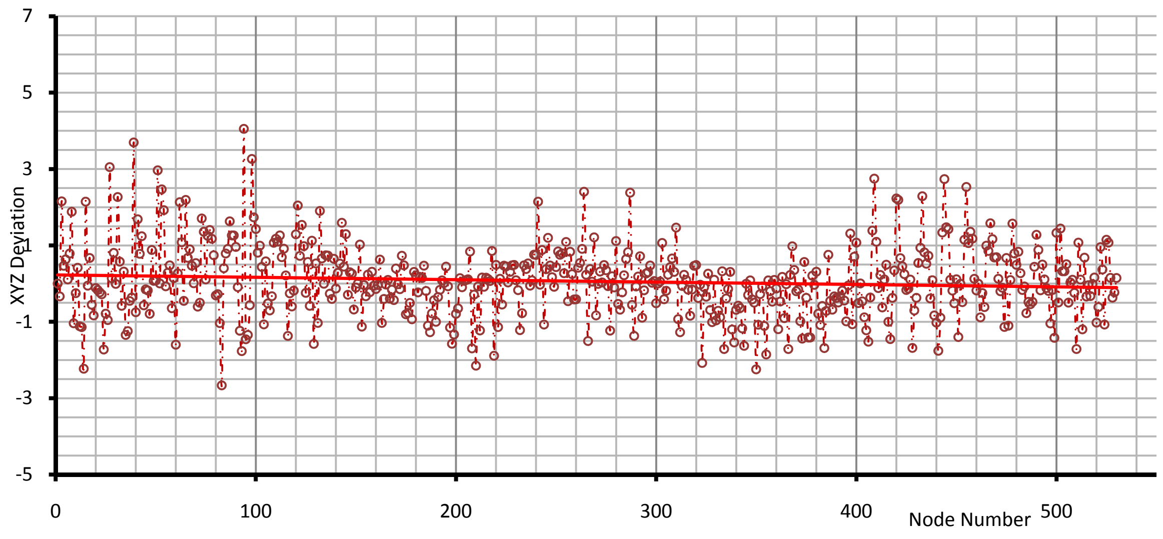


Figure 27. Deviation of springs with 7 set lengths

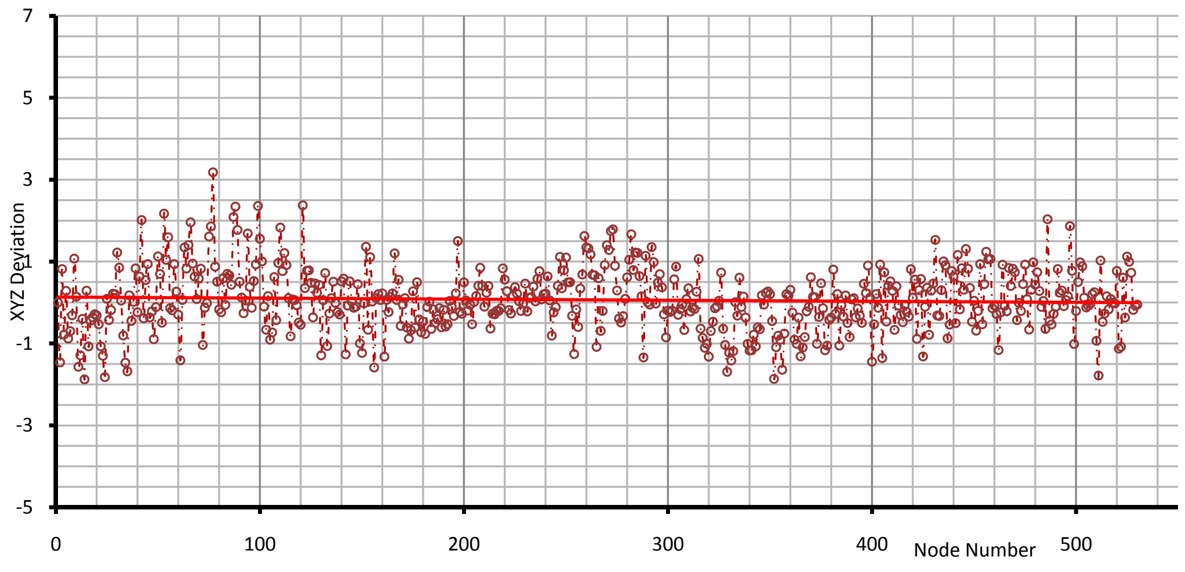


Figure 28. Deviation of springs with 9 set lengths.

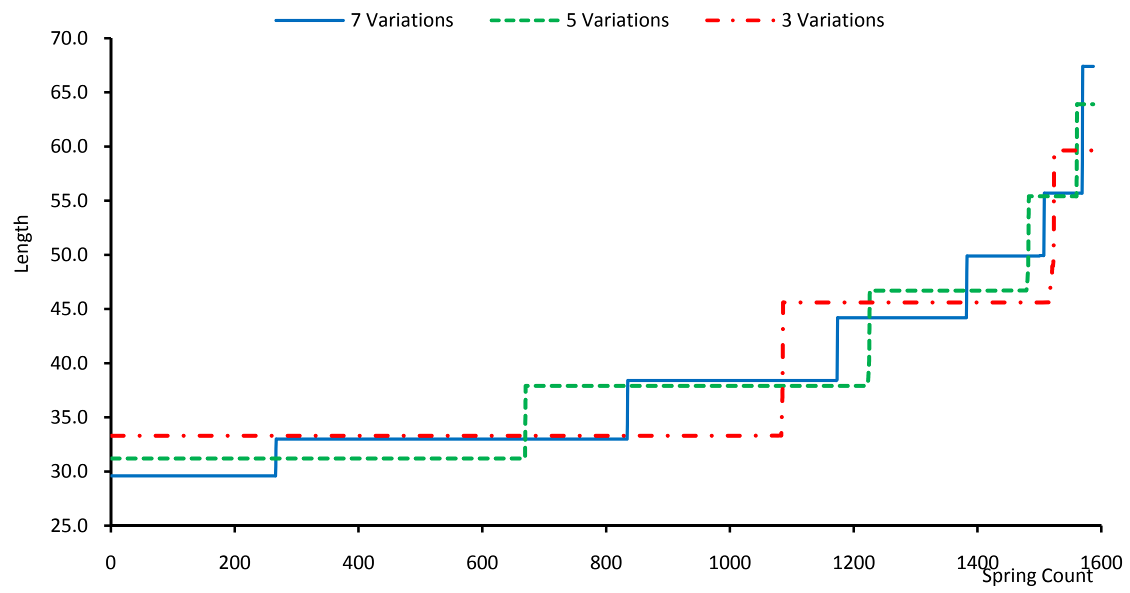


Figure 29. Final resting lengths of springs for all 4 variations using 'range' method

4.4 Comparison of the two methods

By analysing the average lengths used for both methodologies (figure 30) it seems that because the lengths of the sort methodology are closer to the actual spring lengths this method should produce better results. In practice the range methodology seemingly produced less deviation from the surface with a maximum error of 7 as opposed to the 'sort' method which was as high as 25 in certain areas. Consequently the average deviation seems to be more consistent throughout with minimal deformation of the tessellated pattern from 7 variations onwards.

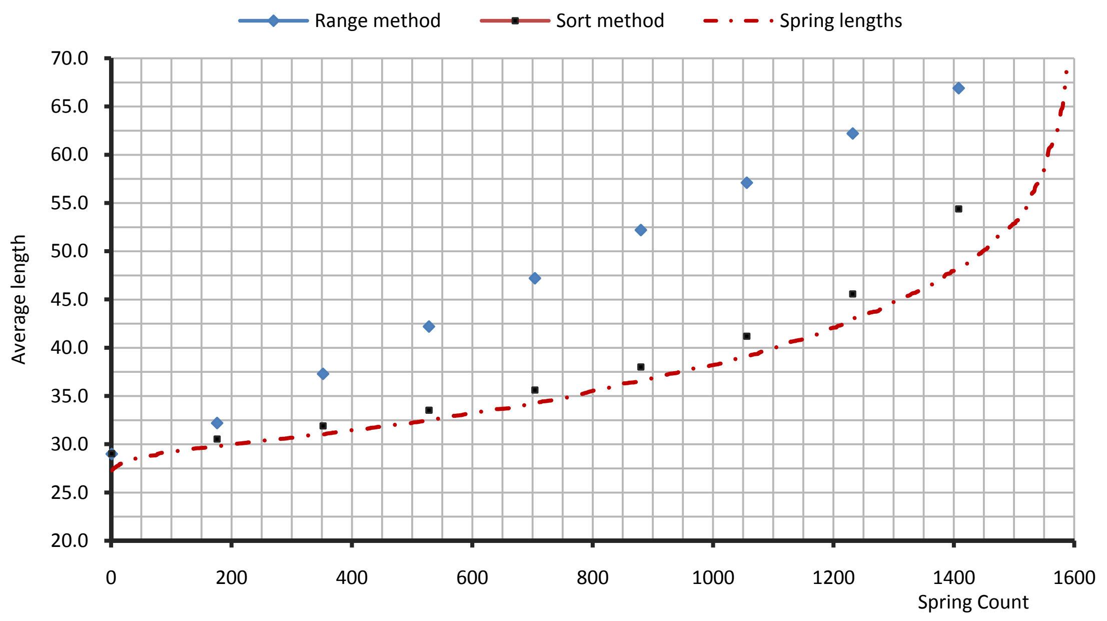


Figure 30. Average lengths for testing with 9 variations, showing values used for both methods.

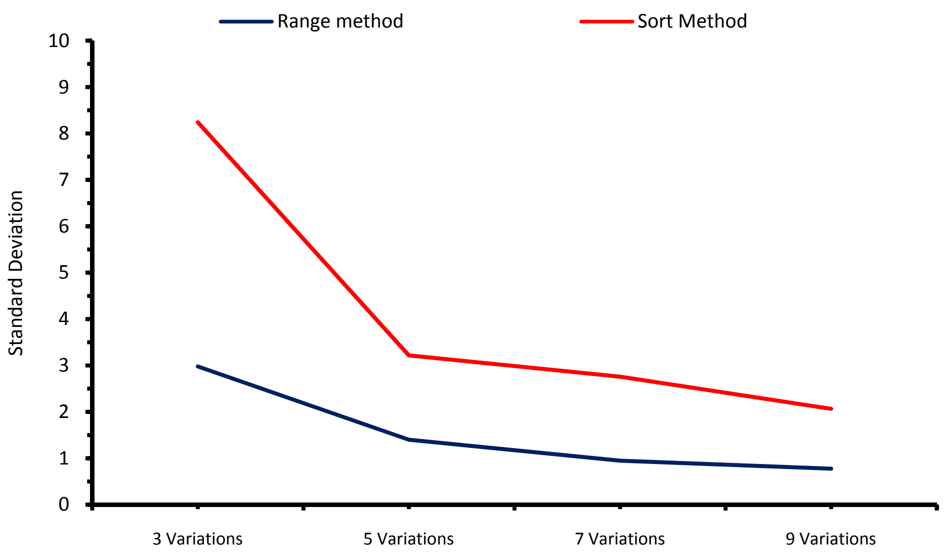


Figure 31. Standard deviation of node position based on the number of variations of springs

Figure 31 exhibits the main differences between the results of the two methodologies, which show that the range method was able to reduce the deviation of spring lengths to an average value of 0.7, which is below the accepted error margin of 1.

4.5 Cost analysis

Given that increasing the number of spring variations improves the level of deviation from the surface shown in figure 31, it is important to clarify a cost effective solution which takes into consideration the time necessary to construct multiple variations of hexagonal cells.

Assuming that the cost of cutting different length members is negligible, due to CNC production, the main issue needed to be considered is the ease of construction when dealing with various structural sizes. The following plans illustrate the differences between using 3, 5, 7 and 9 variations of lengths, where each hexagonal cell is colour coordinated to demonstrate the various lengths.

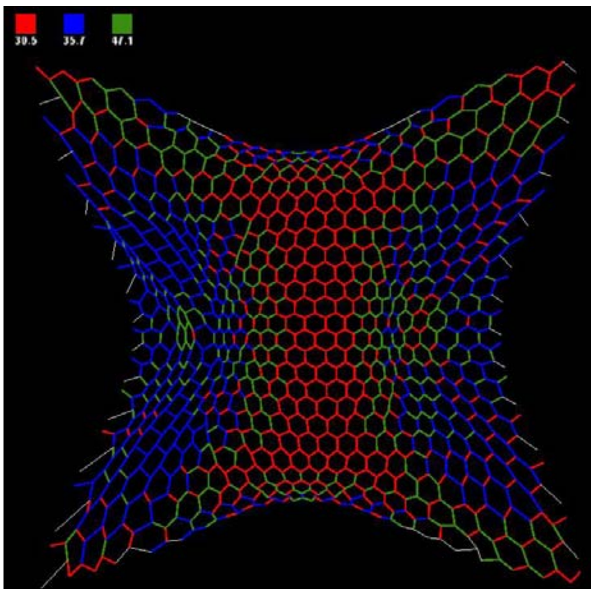


Figure 32. plan showing 3 variation

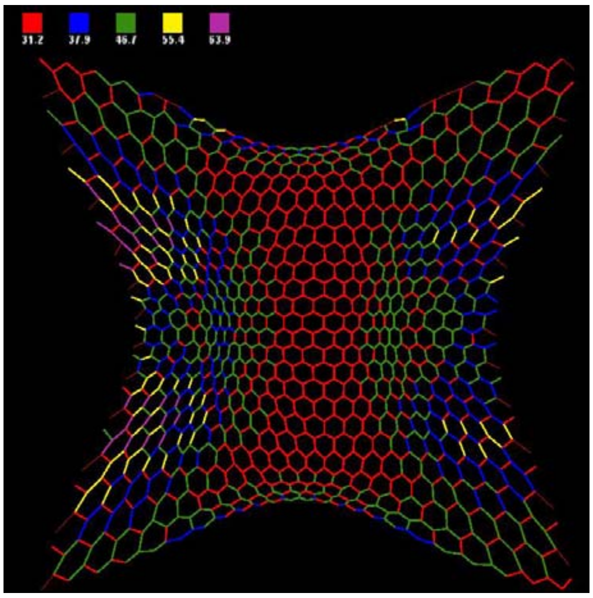


Figure 33. plan showing 5 variation

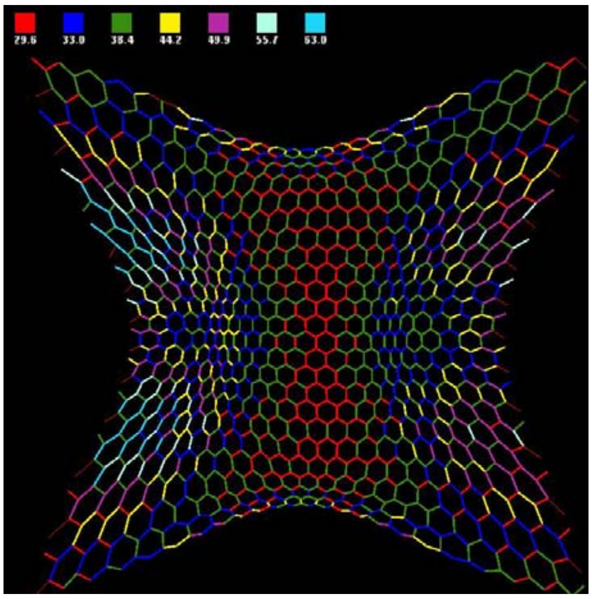


Figure34. plan showing 7 variations

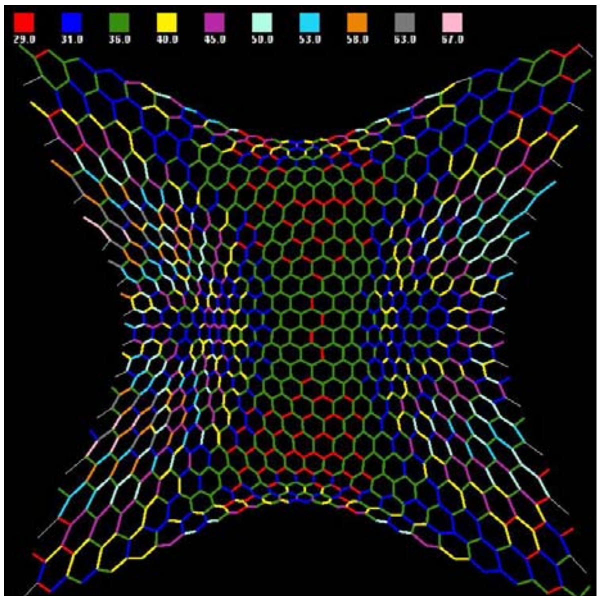


Figure 35. plan showing 9 variation

The use of colour coordinated plans gives us a better understanding of the process required for the construction of this specific geometry. Although it is difficult to specify an exact value for the time required to read and assemble these plans, we can set a hypothetical estimation of cost based on the time it takes to assemble one cell from different lengths; i.e. assuming that a cell made from equal sized members takes the least time to assemble, then providing 50 variation of lengths in total is going to increase the time of construction based on the number of different sized hexagons produced in the plan. Therefore if we assume that the cost of construction is exactly proportional to the number of initial variations, then the optimum number of variations to construct can be obtained through the intersecting point of both variables shown in figure 36 at point X.

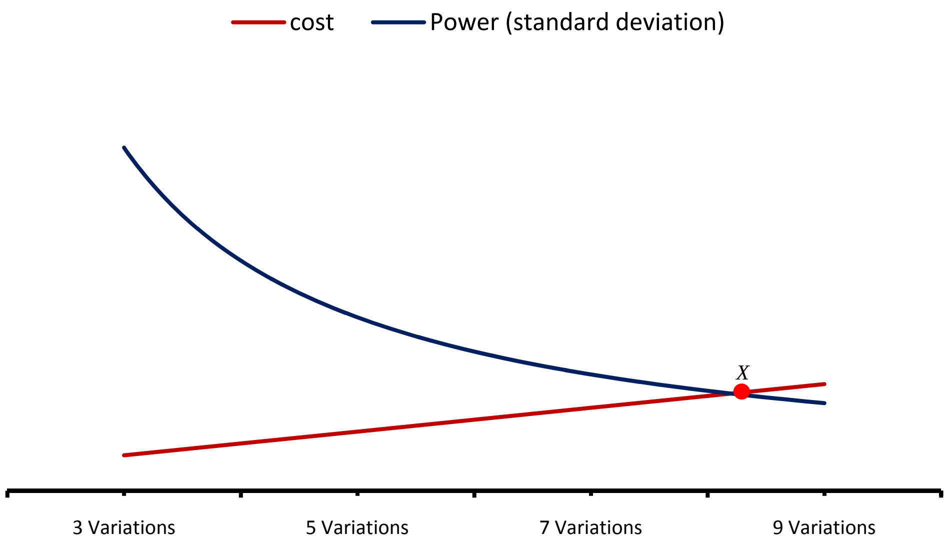


Figure 36. Standard deviation of the different variations and the cost of construction

5.0 Discussion

5.1 Overview of Findings

In regards to the aim of this thesis, the results show that out of the initial 437 variation of lengths in the given tessellation, the applied algorithm was able to reduce this number to a minimum of 8 different lengths whilst satisfying the issue of cost.

The analysis and testing of the dynamic relaxation method showed that by restricting the nodes on the surface, the system was able to only reduce the variation of lengths by a limited amount, and thus the optimisation of the particle springs, with defined rest lengths, was an imperative step in the final outcome. The results of the particle spring optimisation were entirely based on the number of the defined variations and the method used to obtain these ideal lengths. The resultant deviation of the final test using 9 variations derived through the range method proved to amount to a total average deviation of 0.7 as opposed to the sort methodology which was much higher at 2. The comparison of the two methodologies showed that the sort method produced more distortion towards the peaks of the surface. This is presumed to have been due to the inability of this methodology to cater for the rapidly increasing number of lengths in the last 400 springs. Subsequently more distortion is created in the areas with the largest initial spring lengths, producing a negative effect on the tessellation.

The final analysis into the cost implications of the system provided a means of answering the first question of the hypothesis concerned with a cost effective solution to the problems encountered by arbitrarily curved surfaces. Therefore the number of variations was weighted against the ease of construction which provided an optimum solution of 8 variations. This outcome provides an indicative solution to the difficulties in constructing highly differentiated structures and is a concluding remark based on the results of the compiled experiments.

5.2 Critical Assessment

In order to assess the viability of the implemented code it is necessary to take into considerations some of the concerns which could affect the results of conducted experiments. In respect to the issue of cost, it is difficult to provide a final solution without more accurate understanding into the cost of construction. Therefore the final result of the cost analysis is a hypothetical discussion as opposed to a concrete answer. The cost of such structures is based on many other criteria which have not been considered, and therefore approximating the ease of construction is a rough guideline for the cost evaluation of complex geometries. Likewise in the initial testing of the density, the cost of increasing the number of springs did not take into consideration all the other various criteria which can affect the density of the tessellated structure such as structural and environmental issues, and is therefore a rough estimate of a final result.

In the dynamic relaxation procedure although there is a significant decrease in the number of variations and the range of the spring lengths, the nature of the curve is a fundamental criteria in the success of this procedure and in this occasion was a limiting factor preventing all the springs from reaching similar lengths. The success rate of this algorithm is thus widely dependant on the initial form of the surface.

The methodology used to define ideal lengths can also be examined in much greater depth using other techniques to better obtain lengths from the given population. This was a fundamental issue with the sort method which could have provided better results if the average lengths had not been derived through equal groupings of the total number of lengths, and instead were better distributed along the number of springs.

Nonetheless, It can be argued that one of the most important features of the proposed method is the use of the generative procedure which provides an easy process for adjusting the specified lengths as well as the density of the tessellation and initial geometry, so that future manipulation of the system could provide a tool that can be tailored to meet different requirements. The use of the particle spring systems not only provided a means of optimising the spring lengths but it also provides the necessary specifications for the construction of the space frame joint system, which are easily exported for fabrication without any additional production drawings.

5.3 Future Developments

Although the algorithm succeeded in addressing the research questions presented in the hypothesis, the study presented serves only as a preliminary investigation of a cost effective solution to the resolution of an arbitrary surface. Further advancements of the algorithm can address multiple criteria which can aid in the standardisation of a complex surface, and some of which are addressed below;

- Although the variation of spring lengths have been reduced, the resultant node and connections between these structural members still remain highly differentiated, therefore requiring a mass-customised production to cater for the multiple variations. This issue is something which can be further explored to achieve a limited number of nodal types and angles between springs as opposed to the current state which presents over 1000 variations.
- Another important aspect which can affect the cost such structures is the cladding of each hexagonal cell, which depending on the material, will generally be constructed from planar elements that connect to each side of the hexagon. At present the current tessellation algorithm does not take this issue into consideration which makes the use of sheet materials an impossible task. Therefore further optimisation of the spring system could incorporate this issue and make each node comprising the hexagons lie on the same plane.
- The cost of space frames are also highly related to their structural integrity and material used. This issue is something which has not been taken into consideration at all and therefore can be further explored by combining structural analysis using Finite elements methods to provide a more accurate analysis of the tessellated pattern and resultant topology.

6.0 Conclusion

The project presented in this thesis set out to resolve some of the fabrication issues concerned with the construction of differentiated space frame components on an arbitrarily curved surface. A method for tackling the number of variations of structural sizes in such surfaces was proposed that entailed a generative algorithm using dynamic relaxation and particle spring systems. The study was initiated through an exploration of certain architectural precedents which have tackled the issues of constructability and geometrical optimisation related to arbitrarily curved surfaces as well as the use of cellular configurations in structural design. As a result, a honeycomb space frame construction was selected as a desirable tessellation and construction logic, and the suitability of this methodology was tested through formal experimentation of the various criteria that affect the cost-effectiveness of such structures. The testing was conducted in four stages which included the density of the tessellation, the effectiveness of the dynamic relaxation, exploration of length variations and finally a cost analysis into the ease of construction. The evaluation of the results confirmed that the algorithm was capable of reducing the number of variations in structural sizes with minimal deformation of the initial form, although further studies into the cost of construction were necessary to obtain a more accurate conclusion into the cost-effectiveness of the result.

The outcome of the final tool demonstrates a capacity to improve the constructional parameters of a multitude of geometric forms, something which can be easily incorporated by architects and designers for further advancement and use. The significance of the relaxation procedure is not limited to the use of reticulated structures but also provides a mechanism for material simulation of membranes, whilst the honeycomb tessellation also serves as a preliminary investigation which can be adapted to produce a variety of different networks. Moreover the use of the particle spring systems provides the adequate setup to proceed onto a structural evaluation of the system which can help find efficient structural forms in real time. Ultimately this dissertation was not only in search of a cost effective solution, but the creation of a tool that could embed the geometric behaviour, manufacturing constraints and assembly logic into a single system to provide an integrated strategy that is capable of responding to various criteria. In this way the resultant generative code provides the necessary parameters that can be further explored to assist in the production and construction of complex forms.

7 Appendices

Appendix I: Creation of Nodes based on Nurbs Equation

Sections of this function are referenced from 'NURBS fitting' by Alasdair Turner (2009)

```
float [] u_knots = {
  0.0, 0.125, 0.25, 0.375,0.5, 0.625, 0.75, 0.875, 1.0 };
float [] v_knots = {
  0.0, 0.125, 0.25, 0.375,0.5, 0.625, 0.75, 0.875, 1.0 };

class Node
{
  PVector pos; //x,y,z positions
  PVector n_dir;
  float u;
  float v;
  Node(float _u, float _v)
  {
    u = _u;
    v = _v;
    n_dir = new PVector(0,0,0);
  }
// calculate xyz position of node from NURBS Equation
  void calcpos()
  {
    pos = new PVector();

    for (int i = 0; i < u_ctrl_pts; i++) {
      for (int j = 0; j < v_ctrl_pts; j++) {

        float basisv = basisn(v,j,v_deg,v_knots);
        float basisu = basisn(u,i,u_deg,u_knots);
        PVector pk = PVector.mult( ctrl_pts[i][j], basisu * basisv);
        pos.add( pk );
      }
    }
  }
// change node position
  void move()
  {

//add the direction to the position vector
    pos = PVector.add(pos,n_dir);
// reset direction vector
    n_dir = new PVector(0,0,0);
  }

  void draw(int col)
  {
    pushMatrix();
    translate(pos.x,pos.y,pos.z);
    noStroke();
    fill(255,0,0);
    box(6,6,6);
    popMatrix();
  }
}
//basis functions for NURBS Equation
float basisn(float u, int k, int d, float [] knots)
{
  if (d == 0) {
    return basis0(u,k,knots);
  }
  else {
    float b1 = basisn(u,k,d-1,knots) * (u - knots[k]) / (knots[k+d] -
knots[k]);
    float b2 = basisn(u,k+1,d-1,knots) * (knots[k+d+1] - u) / (knots[k+d+1]
- knots[k+1]);
    return b1 + b2;
  }
}

float basis0(float u, int k, float [] knots)
{
  if (u >= knots[k] && u < knots[k+1]) {
    return 1;
  }
  else {
    return 0;
  }
}
```

Appendix II: Creation of honeycomb tessellation

```
 class Nurbs{
  PVector dir1, dir2, dir3,dir4, dir5,dir6,dir7, dir8 ;
  float L1, L2, L3,L4,L5,L6;

  Nurbs(){

    ctrl_pts = new PVector[u_ctrl_pts][v_ctrl_pts];
    u_deg = u_knots.length - u_ctrl_pts-1 ;
    v_deg = v_knots.length - v_ctrl_pts-1 ;

    u_spacing = (width / 3);
    v_spacing = (width / 3);

    nodes = new Node [num][num];
    spring1 = new Spring [num][num];
    spring2 = new Spring[num][num];
    spring3 = new Spring [num][num];
    randSurface();
  }

  void display()
  {
    for (int p = 0; p<num-1; p+=1)//draw honeycomb grid
    {
      for (int q =(p%2)*2+1; q<num-1; q+=4)
      {
        if (p>0&&q>0){
//create three springs to starting from node[p][q]
          spring1[p][q]=  new Spring(nodes[p][q],nodes[p-1][q-1]);//blue
          spring2[p][q]=  new Spring(nodes[p][q],nodes[p][q+1]); //red
          spring3[p][q]=  new Spring(nodes[p][q],nodes[p+1][q-1]);//green
           if(visible){
            if(p>1&&q<num-3&&p<num-2){
              spring2[p][q].draw();//red
            }
            if(p>1){
              spring1[p][q].draw();//blue
            }
            if(p<num-2){
              spring3[p][q].draw();//green
            }

            if(calcpos==true){
//calculate their position based on NURBS equation
              nodes[p][q].calcpos();
              nodes[p][q+1].calcpos();
            }
            }
          }
        }
      }

  void Draw_ctrlPoint(int i, int j){
    pushMatrix();
    translate(ctrl_pts[i][j].x, ctrl_pts[i][j].y, ctrl_pts[i][j].z);
    fill(140);
    stroke(140);
    sphere(7);
    popMatrix();
  }

// set up control points in a regular grid on the xz plane with defined
heights:

  void setHieght(int i, int j, int ct_pt1,int ct_pt2,int ct_pt3,int ct_pt4,
int Height){

    if (i==ct_pt1 && j==ct_pt2 || i==ct_pt3 && j==ct_pt4){
      ctrl_pts[i][j] = new PVector( u_spacing * i, Height, -v_spacing * j);
//MoveValue allows easy adjustment of the control points for further
manipulation
      ctrl_pts[inum][jnum].y+=MoveValue;
      Draw_ctrlPoint (i,j);
    }
  }

  void randSurface()
  {

    for (int i = 0; i < u_ctrl_pts; i++) {
      for (int j = 0; j < v_ctrl_pts; j++) {

        ctrl_pts[i][j] = new PVector( u_spacing * i,0, -v_spacing * j);
        setHieght( i, j,  2, 5, 3, 5, -1000 );
        setHieght( i, j,  2, 0, 3, 0, -800);
        setHieght( i, j,  2, 4, 3, 4, -300 );
        setHieght( i, j,  2, 1, 3, 1, -300);
        setHieght( i, j,  0, 3, 0, 2, 50);
        setHieght( i, j,  5, 3, 5, 2, 50);
        setHieght( i, j,  1, 3, 1, 2, -1050);
        setHieght( i, j,  4, 3, 4, 2, 800 );
        setHieght( i, j,  3, 3, 3, 2, 350);
        setHieght( i, j,  2, 3, 2, 2, -450);
      }
    }
//determine UV position of the nodes
    for (int p = 0; p < num; p++) {
      for (int q = 0; q < num; q++) {
nodes[p][q] = new Node( 0.25 + (0.75 - 0.25) * ((p + 0.5) /  float(num-1)),
0.25 + (0.75 - 0.25) * (q /  float(num-1))  );
        nodes[p][q].calcpos();
      }
    }
  }
```

Appendix III: Function for changing spring lengths

Sections of this function are referenced from code by Kanellos (2007)

```
class Spring

{
  Node a;
  Node b;
  float len;
  float g;
  float actualLength;
  float testlength;
  PVector  dir_1 = new PVector(0,0,0);
  PVector  dir_2 = new PVector(0,0,0);
  float ID_springLength;

  Spring(Node n1, Node n2)
  {
    a = n1;
    b = n2;
    len = PVector.dist(a.pos,b.pos);
  }
//change the length of springs by defining an ideal length ‘t’
  void changeLength(float t)
  {
    ID_springLength = t;
    actualLength = PVector.dist (a.pos, b.pos);

    dir_1 = PVector.sub(b.pos, a.pos);
    dir_2 = PVector.sub(a.pos, b.pos);

    g = ((abs(actualLength- ID_springLength))/actualLength)/2;
    dir_1.mult(g);  //scale the direction vector
    dir_2.mult(g);
//if actual length is smaller then move the nodes apart
    if (actualLength< ID_springLength)
    {
      a.n_dir = PVector.sub(a.n_dir, dir_1);
      b.n_dir = PVector.sub(b.n_dir, dir_2);
    }
//if actual length is bigger then move the nodes together
    if (actualLength>ID_springLength)
    {
      a.n_dir = PVector.add(a.n_dir, dir_1);
      b.n_dir = PVector.add(b.n_dir, dir_2);
    }
    if (actualLength==ID_springLength)
    {
      a.n_dir = new PVector(0,0,0);
      b.n_dir = new PVector(0,0,0);
    }
  }

  void draw()
  {
    float tol= 0.05;
    actualLength = PVector.dist (a.pos, b.pos);  //actual length
    ID_springLength=r;

//change colour  of the springs to illustrate the different lengths
    if(colour==true){
      if (actualLength>av-tol&&actualLength<av+tol)
      {
        stroke(Red);
        strokeWeight(st);
      }
      else if(actualLength>av1-tol&&actualLength<av1+tol)
      {
        stroke(Green);
        strokeWeight(st);
      }
      else if(actualLength>av2-tol&&actualLength<av2+tol)
      {
        stroke(Blue);
```

```
          strokeWeight(st);
        }
        else{
             stroke(255,255,255);
          strokeWeight(1.2);
        }
      }
      line(a.pos.x,a.pos.y,a.pos.z, b.pos.x,b.pos.y,b.pos.z);
    }
  }
  //determine if the spring is in between a certain range and change the
  length to the new rest lengths
  void calcVal(Spring [][] s1, float start_range, float end_range, float
  average_length, int p, int q ){

    if(s1[p][q].len>=start_range&&s1[p][q].len<end_range){
      r=average_length;
    }
  }
  //release the nodes off the surface by changing all the spring lengths
  void release(){
    for (int p = 1; p <49; p+=1)
    {
      for (int q =(p%2)*2+1; q<48; q+=4)
      {
        if(p>1 &&p<num-2 &&q<num-2&&q>0){

          calcVal(spring1, range1, range2, av, p, q );
          calcVal(spring1, range2, range3, av1, p, q );
          calcVal(spring1, range3, range4, av2, p, q );
          nodes[p][q].move();
          spring1[p][q].changeLength(r);//blue

          calcVal(spring3, range1, range2, av, p, q );
          calcVal(spring3, range2, range3, av1, p, q );
          calcVal(spring3, range3, range4, av2, p, q );
          nodes[p][q].move();
          spring3[p][q].changeLength(r); //green

          calcVal(spring2, range1, range2, av, p, q );
          calcVal(spring2, range2, range3, av1, p, q );
          calcVal(spring2, range3, range4, av2, p, q );
          nodes[p][q].move();
          spring2[p][q].changeLength(r);//red
          nodes[p][q+1].move();
        }
      }
    }
  }
```

Appendix IV: Dynamic Relaxation

```
void D_R (){
  for (int p = 1; p <num-1; p+=1)
  {
    for (int q =(p%2)*2+1; q<num-1; q+=4)
    {
      if(p>1 &&p<num-2 &&q<num-3&&q>0){
        surface.calcLength(p,q);
// determine new UV coordinated for each node through average position of
the three surrounding nodes
float averageU =(((nodes[p-1][q-1].u*surface.L1)+
(nodes[p][q+1].u* surface.L2)+(nodes[p+1][q-1].u*surface.L3))/
(surface.L1+surface.L2+surface.L3));

float averageV= (((nodes[p-1][q-1].v*surface.L1)+
(nodes[p][q+1].v* surface.L2)+(nodes[p+1][q-1].v*surface.L3))/
(surface.L1+surface.L2+surface.L3));

nodes[p][q].u=averageU;
nodes[p][q].v=averageV;
      }
    }
  }
}
```

Appendix V: Fabrication of nodes

```
class Cylinder{
  Node a;
  Node b;
  Cylinder(Node n1){
    a = n1;
  }
//create a cylinder
  void drawCylinder(){
    vertices = new PVector[2][pts+1];
    stroke(255);
    for (int i = 0; i < 2; i++){
      angle = 0;
      for(int j = 0; j <= pts; j++){

        vertices[i][j] = new PVector();
        vertices[i][j].x = cos(radians(angle)) * radius;
        vertices[i][j].y = sin(radians(angle)) * radius;
        vertices[i][j].z = cylinderLength;
        angle += 360.0/pts;
      }
      cylinderLength *= -1;
    }

    // draw cylinder tube
    beginShape(QUAD_STRIP);
    pushMatrix();
    rotateY(PI/2);
    translate(0, 0,cylinderLength);
    for(int j = 0; j <= pts; j++){
      vertex(vertices[0][j].x, vertices[0][j].y, vertices[0][j].z);
      vertex(vertices[1][j].x, vertices[1][j].y, vertices[1][j].z);
    }
    endShape();

    //draw cylinder ends
    pushMatrix();
    for (int i = 0; i < 2; i++){
      beginShape();

      for(int j = 0; j < pts; j++){

        vertex(vertices[i][j].x, vertices[i][j].y, vertices[i][j].z);
      }
      endShape(CLOSE);
    }
    popMatrix();
    popMatrix();

  }
//create connection detail by drawing a sphere and three cylinders which
are rotated to the direction of the springs
void connection( PVector dirctn1, PVector dirctn2, PVector dirctn3, int p,
int q){
    pushMatrix();
    if(fabricate){
//for fabrication translate all the connections to the same plane in y
      float xx= scl*a.pos.x;
      float yy=0;
      float zz= scl*a.pos.z;
      stroke(255,0,0);
      translate(xx,yy,zz);
    }
//otherwise draw the connections at the position of each node
    else{
      translate(a.pos.x,a.pos.y,a.pos.z);
    }
    noStroke();
    fill(255);
    sphere(2);
    sphereDetail(10);
    Tube(dirctn1);
    Tube(dirctn2);
    Tube(dirctn3);

    popMatrix();
  }

  void Tube(PVector direction){ //create tube
```

```
    pushMatrix();
    PVector polar = cartesianToPolar(direction);
    rotateY(polar.y);
    rotateZ(polar.z);
    drawCylinder();
    popMatrix();
  }
//this function is used for 3d rotation
  PVector cartesianToPolar(PVector theVector) {
    PVector res = new PVector();
    res.x = theVector.mag();
    if (res.x > 0) {
      res.y = -atan2(theVector.z, theVector.x);
      res.z = asin(theVector.y / res.x);
    }
    else {
      res.y = 0;
      res.z = 0;
    }
    return res;
  }
}


void display_Joints(){

  for (int p = 1; p <num-1; p+=1) //main section
  {
    for (int q =(p%2)*2+1; q<num-1; q+=4)
    {
      if(p>1 &&p<num-2 &&q<&&q>0){
        surface.calcLength(p,q);
  tube[p][q] = new Cylinder(nodes[p][q]);
  tube[p][q].connection(surface.dir1, surface.dir2, surface.dir3, p,q);
      }
    }
  }
}
```

8.0 Reference

Littlefield, D, (2008) Space craft: Developments in Architectural Computing, London, RIBA Publishing

Pearce, P (1978) Structure in Nature Is a Strategy for Design, Massachusetts, MIT Press

Hensel,M. and Menges,A. and Weinstock,M.(2006) Techniques and technologies in Morphogentic Design. Architectural Design 76(2) Chichester, Wiley Academy

Hensel,M. and Menges,A. and Weinstock,M.(2004) Emergence: Morphogenetic design strategies. Architectural Design 74(3) Chichester, Wiley Academy

Reas,C and Fry,B. (2007) Processing: a programming handbook for visual designers. Massachusetts, The MIT press.

Thomson, D'Arcy, (1961) On Growth and Form. Cambridge: Cambridge University Press

Barnes,M and Dickson,M, (2000) *Widespan roof structures*, London , Thomas Telford

Williams, Chris J K. THE ANALYTIC AND NUMERICAL DEFINITION OF THE GEOMETRY OF THE BRITISH MUSEUM GREAT COURT ROOF, University of Bath, UK accessed August 2009, <http://staff.bath.ac.uk/abscjkw/BritishMuseum/ChrisDeakin2001.pdf>

Kilian A. and Ochsendorf J, (2005), 'Particle-Spring Systems for structural form-finding', Journal of the international association for shell and spatial structures: IASS, no. 148, pp. 77-84, accessed August 2009,<http://designexplorer.net/newscreens/cadenarytool/KilianOchsendorfIASS.pdf>

Kanellos A.(2007), 'Topological Self-Organisation: Using a particle-spring system simulation to generate structural space-filling lattices, MSc thesis, Bartlett School of Graduate Studies, UCL, accessed August 2009, < http://eprints.ucl.ac.uk/archive/00002882/>

Schlueter,A and Bonwetsch,T. (2008) Design Rationalization of Irregular Cellular Structures. international journal of architectural computing. issue 02, volume 06, accessed August 2009, http://www.gramaziokohler.com/data/publikationen/592.pdf

Arkinstall, M and Carfrae,T, Structural Design and Optimisation of the Beijing National Aquatics Center, accessed August 2009, http://www.engineersaustralia.org.au/shadomx/apps/fms/fmsdownload.cfm?file_uuid=5E6A0D1A-AD5C-629D-8A46-457845CBC345&siteName=ieaust

Stephan, s and sánchez-alvarez,j and knebel, k, Reticulated structures on free-form surfaces, accessed August 2009, http://www.mero.de/fileadmin/downloads/bausysteme/publikationen/free_ret_stru_e.pdf

Kolarevic,B.(2003) Architecture in the digital age: design and manufacturing, accessed August 09, http://books.google.co.uk/books?id=qA7nT5J4WIC&printsec=frontcover&dq=architecture+in+the+digital+age&ei=Q1qlSqaiNYbEM-n26Y8I#v=onepage&q=&f=false