Harmonic form-finding for the design of curvature-stiffened shells

by

Cecilie Søs Brandt-Olsen

Supervised by

Dr. Paul Shepherd

Prof. Paul Richens

A thesis submitted for the degree of Master of Philosophy
University of Bath

Department of Architecture and Civil Engineering

September 2015

This page is intentionally left blank.

Abstract

The last decade of technological advancements have set architects free to explore a vast variety of shapes, which in effect has given rise to a trend of complex free-form buildings in contemporary architecture. These shapes are generally based on pure aesthetics which often results in awkward and over-dimensioned structures or very costly construction. As a consequence, engineers have developed methods to inform the shape design such that form follows force. While these structures are highly effective, they lack considerations of practical constraints and reduce the need for interaction between the architect and engineer.

This research offers a novel free-form modelling technique with inherent shapes suitable for stiffening of shells through curvature. The methodology has been implemented in a software tool to help guide the design at the conceptual stage by providing upfront feedback to changes of form, thus integrating architectural vision with structural logic. The modelling approach is based on harmonics, which makes it possible to parametrise a given mesh by a few variables and simultaneously perform advantageous analysis of the geometry. The generated shapes are subsequently evaluated in terms of their buckling capacity, where it is evident that the inherent double curvature provides geometrical stiffness to better resist sudden failure due to high compressive forces.

Case studies of the British Museum Great Court Roof and other smaller examples, combined with a continuous dialogue with people from the industry, have been used to assess and enhance the applicability of the design tool in practice.

Acknowledgement

This thesis is dedicated to my grandfather whose many adventures to follow his passion inspired this one of mine.

Thank you to my two supervisors Dr. Paul Shepherd and Prof. Paul Richens for their excellent guidance and inspirational discussions throughout this year.

Thank you to Duncan Horswill and Andreas Bak for their support and expertice to prepare me for this MPhil in the first place.

Thank you to Stephen Melville, John Harding, Will Pearson and Kristjan Plagborg Nielsen from Ramboll Computational Design for their help and inspiration throughout this year.

Thank you to Stefan Brandt, Bjarke Koch-Ørvad and Jacob Drachmann for laying the stepping stones for this computational design journey a few years ago.

Finally, thank you to my family and Josef for their love and encouragement.

Contents

List of Figures

Chapter 1

Introduction

This chapter highlights tendencies and describes current approaches within contemporary shell design. It motivates the development of a tool, which balances visual expression with structural rationality rather than focusing on one or the other. A number of objectives for this research are identified and an overall structure of the thesis is outlined.

1.1 Free-form surface design

Architectural forms have changed remarkably over the last 20 years as a consequence of technological advancements. Previously, designs were sketched by hand using pencil and paper and different tools such as rulers and compasses helped to guide this process. Today the computer, with its computer aided design (CAD) packages, is an integrated part of the design process, providing the architects with more freedom. Almost every imaginable shape can be modelled by means of Non-Uniform Rational B-Splines (NURBS) or subdivision surfaces. As a result, organic free-form shapes are more commonly seen in contemporary architecture. They have the advantage of being visually expressive, creating interesting spaces and spanning long distances without the need for supports. A good example of these aesthetically pleasing structures is the Heydar Aliyev Center by Zaha Hadid Architects as shown in Figure 1.1.1.

The design flexibility that the modelling tools provide is of great importance

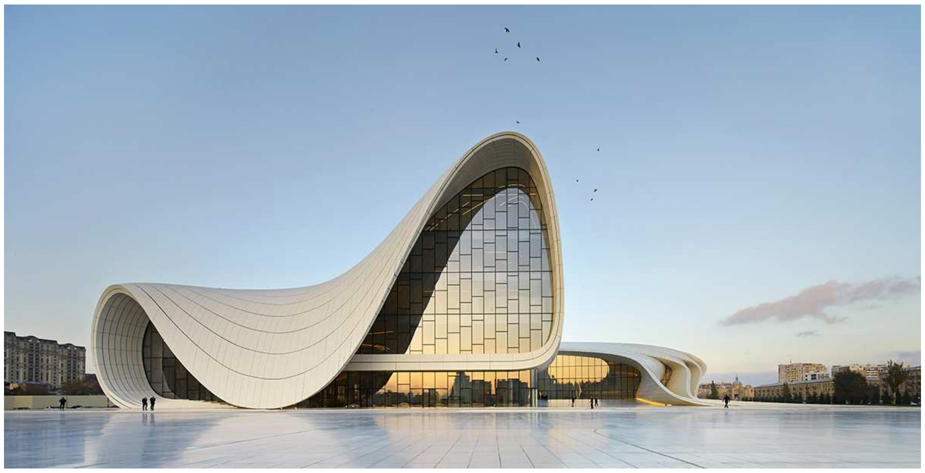


Figure 1.1.1 – Heydar Aliyev Center by Zaha Hadid Architects (Photo by Hufton+Crow Photographers)

to maintain the creative and innovative role of the architect who thereby can keep on pushing the limits of what is possible. However, CAD packages make it easy for a designer to create complex doubly curved surfaces with no understanding of how the shape is generated geometrically, performs structurally or its constructability, which can be crucial factors for a successful realisation of a project.

Some architects have succeeded in integrating some of those factors in the design process and used it to develop their architectural language. As an example, Frank O. Gehry has developed his signature by only using developable surfaces (zero Gaussian curvature) for his designs. They express a high level of complexity but the surfaces can be created from folded sheet materials, which significantly increase the constructability and reduce the cost in relation to fabrication.

1.2 Structural rationalisation

Every free-form surface design contains three aspects: shape, topology and sizing. The shape is the spatial configuration of a surface (geometry), the topology refers to the connectivity of structural members within that shape domain and the sizing is concerned with the dimensioning of those members. Usually, the architect is responsible for the shape and perhaps the topology and the engineer for the sizing. If the free-form surface design is complex, it is sometimes the case that the engineer is responsible for the topology as well i.e. the architect provides the engineer with a given surface design and the task for the engineer then becomes to develop a scheme of structural members to approximate that shape in the best way and subsequently dimension the members. While this is generally possible, it often leads to very awkward structures where the size of the elements completely remove the focus from the visual expression of the surface (if not externally then internally).

Due to the interesting nature of shell structures, engineers have developed different methods to determine efficient shapes under given loadings. The hanging chain model is one of the oldest methods published by Robert Hooke in 1676 and described as: *"As hangs the flexible line, so but inverted will stand the rigid arch"* (Hooke, 1676). This method takes advantage of the hanging chain being in pure tension and free of bending under its own weight and thus when inverted will obtain a pure compression state under the same load. This principle can be transferred to shells in order to find a compression-only shape, which is beneficial as the forces only act in the tangential plane of the shell and therefore only need a small cross section. Antoni Gaudi (1852-1926) is well-known for having applied this physical form-finding technique using weighted strings to mimic the load e.g. for the design of Colónia Güell in Barcelona. The architectural engineers Frei Otto (1925-2015) and Heinz Isler (1926-2009) are also famous for having implemented this form-finding principle to experiment with soap-film and fabric respectively, which helped to guide the design of projects like the German Pavilion at the Montreal Expo in 1967 and the Sicli SA Factory in Geneva in 1970. These projects share the property that each individual shape is derived from structural logic and thus expresses a high level of elegance as form follows force. However, physical form-finding methods can be very time consuming to set up and the manual measurements that are necessary to describe the new shape can be cumbersome to obtain and hence cause inaccuracy.

Advances in computer technology allows these physical form-finding methods to be translated into numerical form-finding methods instead, led by the invention of dynamic relaxation (DR) by Alistair Day in 1965 (Day, 1965). The method can be summarised as a system of springs and particles with masses representing the surface, where the applied load causes the particles to move according to Newton's second law of motion through time (Adriaenssens et al., 2014). The oscillating motion of the particles eventually stabilises around the equilibrium solution. As the springs only work by tension or compression forces, the equilibrium shape mimics a shell that only works by membrane action. Another recent digital form-finding technique is *"Thrust Network Analysis"* (TNA), developed by Philippe Block in 2009. It can be characterised as a more geometrical approach to finding three-dimensional equilibrium shapes. The method differs from DR by exploiting the indeterminacy of the shell structure thus allowing a broader variety of possible equilibrium solutions to be explored. It is achieved by separating the horizontal and vertical equilibrium, which allows the designer to modify the force flow (taking advantage of the dual relationship between the form and force diagram) and this in effect changes the geometry (Block, 2009).

The transition to digital form-finding techniques has many advantages such as allowing multiple configurations to be explored in a short time and directly exporting the new shape into other CAD programs. The described methods are the engineers response to the shape aspect of surface design in contemporary architecture. However, they completely exclude the architect from the design process, and as the resulting shapes are purely guided by the applied load, they are very restricted and thereby cannot take other practical design constraints such as internal height at the boundaries or views into account. In practice it is very rare that a structure is only influenced by one load case (for these methods the self-weight) and it means that for example point loads and construction tolerances push the design away from the "optimal".

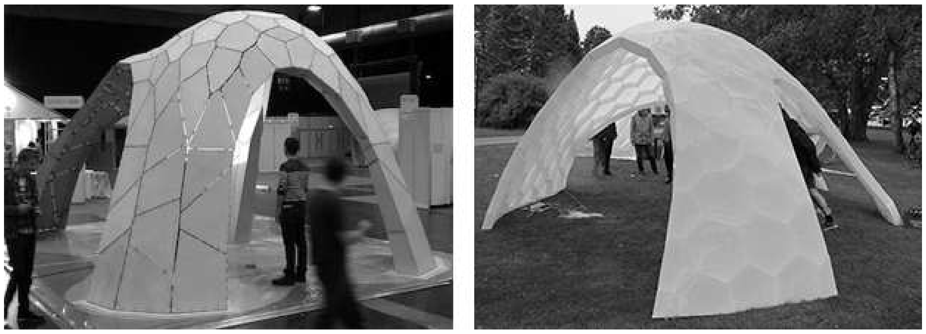


Figure 1.2.1 – Pavilions shaped by dynamic relaxation. Left: Trada Pavilion designed by Ramboll Computational Design. Right: SJ DHL tent designed by Soren Jensen Consultant Engineering

Compression-only shells are vulnerable to failure by buckling, especially in areas where the curvature is low. Figure 1.2.1 shows two different pavilions that have been form found by DR and it can be seen that flat areas arise at the legs thus risking buckling in these zones as is evident from the right of Figure 1.2.1. To prevent this behaviour, the edges along the openings in the Trada pavilion (left of Figure 1.2.1) were strengthened with extra panels.

It is characteristic for these form found shells to occupy a large footprint in order to achieve a desirable height for the internal volume, and as a consequence a lot of space along the boundary is wasted. For the Trada pavilion this problem was tackled by introducing a funnel in the middle.

In summary, the restricted design space dictated by the relationship between form and force, the practical constraints, which push the solution away from the "optimal", the vulnerability against buckling and the large footprint to height ratio suggest that the best solution in practice is not to only design for a pure compression state.

1.3 Balancing form and force

Besides giving rise to non-elegant structures, it can be argued that it is no longer sufficient to design from a pure aesthetic point of view, as the requirements for low-energy buildings, as well as a focus on recycling, material economy and structural efficiency, are ever more stringent. On the other hand, it is not desirable to let the engineer control the entire design process either for the reasons highlighted above. The best solution in practice therefore merges the creative and visually pleasing free-form surface design of the architect with the structural logic of the engineer. This is achieved by changing the work flow such that the shape, topology and size aspects all become part of an iterative process in the conceptual design stage. Figure 1.3.1 shows two different pavilions that emerged from this process. The shapes are geometrically more stringent but it is evident from both that structural innovation is still possible within this restricted design space and with remarkable results.

One way to achieve this enriched process is by enabling upfront feedback in the modelling environment such that any changes to a shape can be immediately quantified in terms of some kind of structural logic. If the shape design is in focus (rather than the topology), the problem firstly becomes to parametrically define the geometry with a desired flexibility whilst at the same time keeping the number of parameters low. This is to avoid controlling the position of each point on the surface individually and hence better create smooth shapes that

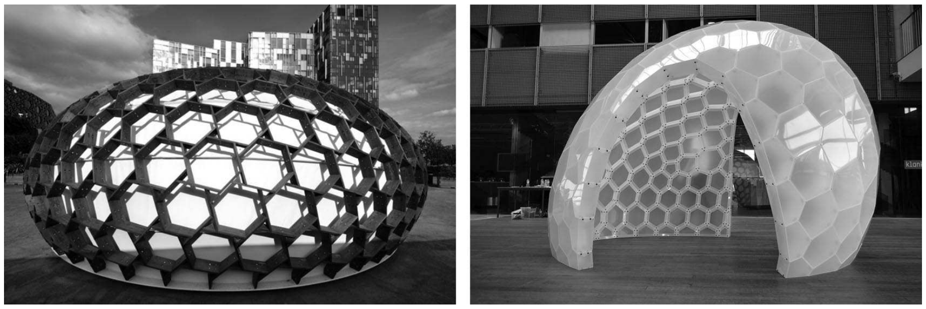


Figure 1.3.1 – Pavilions shaped by a balance between form and force. Left: Kreod Pavilion designed by Chun Qing Li and Ramboll Computational Design. Right: SJ IASS Polyshell designed by Soren Jensen Consultant Engineering

are easily modified. Secondly, the structural behaviour needs to be quantified by some measure that is relevant to the specific project.

1.4 Aim and objectives

This research aims to bridge the gap between free-form surface design based only on aesthetics and the pure engineering approach with a very restricted relationship between form and force. In order to achieve this, the development of a digital tool with the following objectives is necessary:

- Use a free-form modelling strategy with a low number of design parameters
- Be intuitive and flexible
- Improve the current cumbersome work flow between the geometric surface model and finite element model
- Provide real-time structural feedback in terms of stresses, deflection or buckling capacity
- Help guide the shape in the concept design stage

By using several smaller examples and a large case study, combined with a continuous dialogue with people from the industry, a software tool will be developed that is applicable to both the architectural and engineering setting.

1.5 Thesis structure

Chapter 2: reviews relevant literature in relation to the objectives defined above in order to identify current approaches and their limitations and thereby concretise the path of this research.

Chapter 3: contains a description of the theoretical framework chosen to obtain a low parametrisation, along with its implementation and validation. Further refinements are introduced to specifically target the development for an architectural context.

Chapter 4: focuses on the structural feedback, which is linked to specific properties of the shapes resulting from the implemented modelling strategy. It presents a novel approach to quantify this structural behaviour where integration with the existing modelling environment and computational speed are key priorities.

Chapter 5: conducts a case study on the British Museum Great Court Roof, which demonstrates the applicability of the developed design tool on a real world project and highlights the value it creates.

Chapter 6: concludes this thesis with a summery of what this research has delivered and contains proposals for future work.

Chapter 2

Literature review

This chapter provides an overview of relevant methods and examples of applications according to the objectives described in Section 1.4. A full review of each technique is not within the scope of this thesis, but it is intended to build an understanding of previous work to help to identify advantages, limitations and gaps in order to concretise the path of this research. An interview with Zaha Hadid Architects, famous worldwide for their organic free-form shapes, serves to broaden the understanding of surface design in practice. Literature more specific to the implementation is reviewed in subsequent chapters.

2.1 NURBS surfaces

A NURBS (Non-Uniform Rational B-Spline) surface is one method of achieving a low parametrisation of a surface. In other words, it allows the creation of a smooth surface by only defining a few parameters and an algorithm interpolates the rest. As a NURBS surface and a NURBS curve is generated from the same principles, the latter is briefly explained to get an understanding of these parameters and how they affect the underlying algorithm.

The most common way to describe a curve is by a parameter t, which can be thought of as distance. For each value of t there is an associated point on the curve, which can be described as $P(t) = (x(t), y(t))$. The curve is therefore a result of a moving point as described by a given mathematical expression.

However, is not very intuitive or easy to think in terms of mathematical expressions when modelling free-form curves. Thus, the idea is to maintain the t-parametrisation but exchange the mathematical expressions with a few intuitive parameters to create any desired curve. These parameters are: *control points, degree, weights and knot vector.*

The control points form, when connected, a so called control polygon, which the generated curve intuitively follows as seen in Figure 2.1.1. Furthermore, the control polygon provides a very tangible handle for modification of the curve.

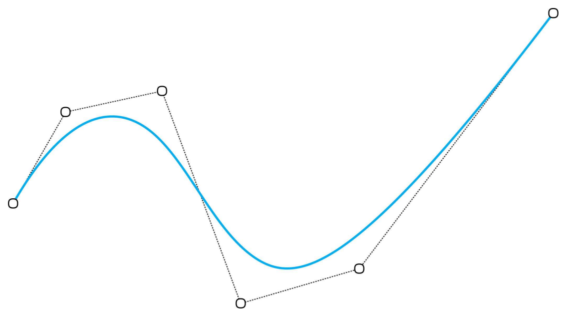


Figure 2.1.1 – A NURBS curve with its control polygon

By moving a control point, local control of the curve is obtained. This local control is achieved by the three other parameters in the following way. For every t, the associated point is a weighted average of the control points. In order to describe in which time interval and how strongly the moving point is influenced by each control point, a basis function per control point is introduced (the "B" in B-Spline stands for basis). The knot vector makes it possible for some control points to influence larger time intervals and of different intensities than others. It essentially divides the curve into time intervals and the degree specifies the polynomial degree of the basis functions and for how long in relation to the knot vector that each function has a non-zero value. Hence, the "Non-Uniform" property. The knot vector can furthermore be used to ensure that the curve passes through the end points in the control polygon and allow kinks to be created. Lastly, each control point has a weight parameter associated with it, which intuitively attracts the curve more for higher values. It allows rational curves to be generated (the "R" in NURBS), which is necessary to e.g. create a circle (Schneider, 2015). Since a surface is most commonly represented by a (u, v) parametrisation, it is possible to translate the described principle to this setting by replacing the t parameter with u and v.

NURBS successfully implement a low-parametrisation strategy of a given surface and the described parameters are very intuitive. Especially the control polygon provides a very spatial handle of the geometry, which makes it easy to create and modify surfaces. Only the knot vector requires a bit more technical understanding and experience to use as a modification tool. However, in many cases the control polygon suffices to create a desirable surface geometry, leaving the rest of the parameters as non-utilised features. This flexibility makes NURBS a very straightforward tool that is widely used in practice.

As a NURBS surface definition is based on a (u, v) parametrisation, it has certain limitations related to this representation. The (u, v) parameters span a rectangular two-dimensional region and a certain mapping is responsible for the translation of this into a three-dimensional surface. Thus, any (u, v) point inside this rectangle is mapped to a spatial point, which in effect makes it impossible to define holes in the surface and trim operations become difficult. This representation furthermore requires an attentive definition of the (u, v) intervals to e.g. avoid that two different (u, v) points map to the same spatial point (Pressley, 2012). As a result, NURBS surfaces are defined as patches that have to be “glued together” to create more complex surfaces. It is generally hard to ensure the same tangent along those seams and that can cause undesirable visual disruptions. In the context of building design, a smooth surface representation also has the disadvantage of requiring additional steps to translate the shape from computer model to a realisable project in the real world, where only discrete elements exist.

Sasaki (2014) has developed a method called “the sensitivity analysis”, which utilises the NURBS representation to embed some structural knowledge into the shape with the aim of preserving the architectural vision. This method was successfully used to modify the original shape of the Teshima Art Museum in Japan (Figure 2.1.2), which spans an area of 42.7 x 60.2 m in plan and has a maximum height of 5.12 m.

The methodology is illustrated in Figure 2.1.3. The control points of the original NURBS surface provided by the architect are used as variables in an optimisation process, where the objective is to minimise the total strain energy of the shell under self-weight. A measure for the strain energy for each configuration is obtained by an additional step, where the surface is discretised into a mesh, which then forms the input for a finite element analysis. The optimisation problem is solved by a gradient descent strategy.

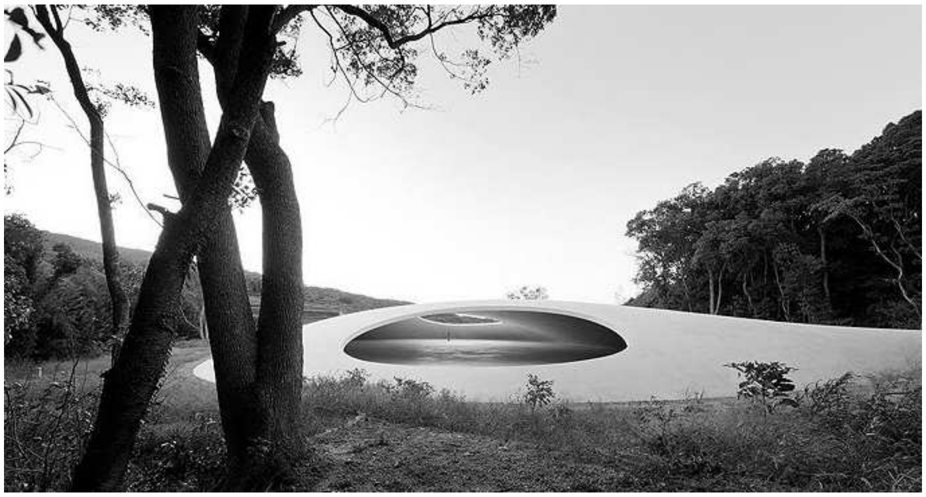


Figure 2.1.2 – Teshima Art Museum, Japan 2010. Copyright Iwan Baan.

Figure 2.1.3 shows how the strain energy and maximum vertical displacement are reduced compared to the initial shape during the optimisation process thus injecting more structural logic into the shell shape. As the method tries to minimise the modifications in relation to the original shape, in this case resulting in a maximum deviation of 400 mm, the solution is heavily dependant on the initial input. The optimised shape is subsequently used as a basis for thorough structural analysis including stress validation and non-linear stability check.

Sensitivity analysis is a good example of how the reduced number of variables of a NURBS representation can be used to embed structural logic into a shape. The large number of projects this method has been applied to, including the Kitagata Community Centre and the Kakamigahara Crematorium in Japan, furthermore demonstrate its practical integrity obtained by respecting the architectural vision. The only noticeable methodological disadvantage is the additional step of converting the smooth surface to a mesh each time the shape is updated in order to extract structural feedback.

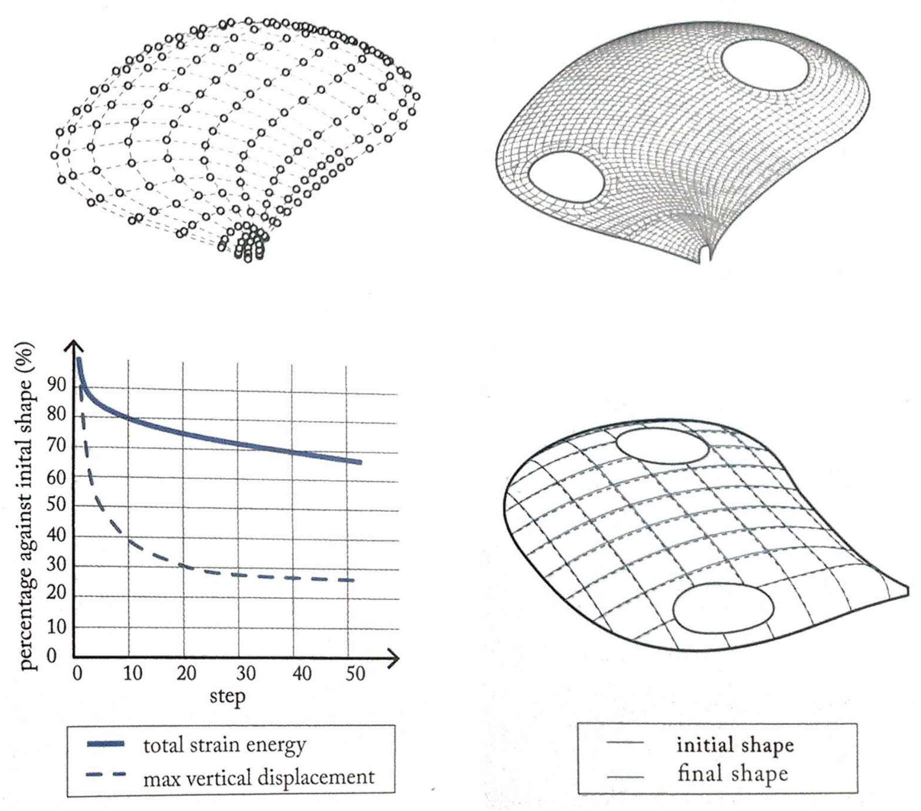


Figure 2.1.3 – Sensitivity analysis of Teshima Art Museum (Sasaki, 2014)

2.2 Subdivision surfaces

Another way of achieving a low parametrisation is by means of subdivision surfaces, which are based on a mesh representation. Usually, the smoothness of the surface is lost when it is converted into a mesh, and thus there is a compromise between the level of refinement to achieve a certain accuracy and how much information has to be stored. Subdivision surfaces solve this issue by being defined from a coarse base mesh, which is the only necessary information to store, and from this any desirable level of refinement can be achieved from a scheme description. A scheme is essentially an algorithm, which specifies this refinement process. It consists of two parts: firstly, a description of how to subdivide each face into several faces topologically by introducing new child vertices, and secondly a procedure of how to move these child vertices (and maybe the original vertices) based on a weighted average of their original neighbour parent vertices. Specific rules apply to the boundaries to obtain desirable results. The scheme is called an interpolation scheme if only the child vertices are moved, and an approximation scheme if both child vertices and the parent vertices are moved. Within these two categories, several schemes exist, which operate on different base meshes. The so-called Loop scheme is an example of an approximation scheme for triangular base meshes. The general idea is that the base mesh converges towards a limit surface by repeating the subdivision scheme at each level of refinement as shown in Figure 2.2.1. This process has the effect of smoothing the vertices, which in general ensures G2-continuity (Shepherd, 2014).

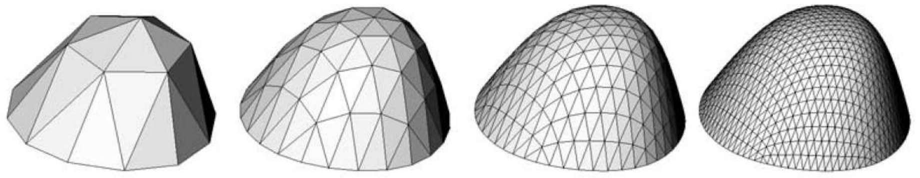


Figure 2.2.1 – Subdivision surface at different levels of refinement (Shepherd, 2009)

The base mesh for a subdivision surface therefore functions similarly to a control polygon of a NURBS surface in the way that the number of variables to control the surface can be reduced to the number of vertices in the base mesh. Thus, it exhibits the same advantage of providing a very intuitive and spatial handle to control the underlying surface shape. The different refinement strategy from control polygon to smooth surface for subdivision surfaces and NURBS surfaces makes it possible to create shapes with holes and in general avoid the issue of surface patches. The mesh representation also offers a direct link to performance analysis software and the necessary level of refinement for each analysis type can easily be generated without any additional efforts (Shepherd, 2014).

While it is easy to define a coarse control mesh and study the resulting limit surface it converges towards, it is much harder to go the other way i.e. model a specific spatial shape by defining a coarse control mesh. However, the latter is often the case in an architectural context when a sketched idea has to be translated into a digital format. Another disadvantage is that boolean operations for subdivision meshes are not obviously defined, hence making mesh modelling a challenging task.

Zaha Hadid Architects are famous worldwide for the use of organic shapes, which has become a signature of their projects including the London Aquatics Centre, Heyday Aliyev Centre and Galaxy Soho. In an interview on 5 June 2015 with Shajay Bhooshan, Associate and founding member of the Computation and Design (co|de) group at Zaha Hadid Architects, the author gained insight into how these complex shapes are modelled. During various projects (Bhooshan and Sayed, 2012), the team has acquired so much experience in working with subdivision surfaces that currently this is their primary modelling strategy. Instead of seeing the difficulties of creating a coarse control polygon from a desirable shape as a limitation, the work flow within the team is rather to explore the broad variety of "sketches" that subdivision surfaces offer. In other words, the implementation of the subdivision surface approach becomes the sketch tool and not the other way around. It was explained that the spatial control mesh encourages an "edit and observe" process, which combined with an aesthetic evaluation at each step and interim performance evaluations, guides the final design. Due to this modelling approach, the most commonly used platform within the team is Autodesk Maya (Autodesk, 2015a), which offers advanced modelling with meshes and subdivision surfaces and is customisable via bespoke C++ scripts.

The design of a building envelope for the tropical hothouse in Aarhus by C.F. Moeller Architects and Soren Jensen Consultant Engineers serves as a good example of how subdivision surfaces can be used in an optimisation process to increase the environmental performance. The architectural concept was a dome structure, which was initially created as the limit surface of a 7 vertex control mesh as seen from Figure 2.2.2 and later on refined by one subdivision step to gain further control. This way of representing the smooth dome structure significantly reduced the number of variables needed to define the shape.

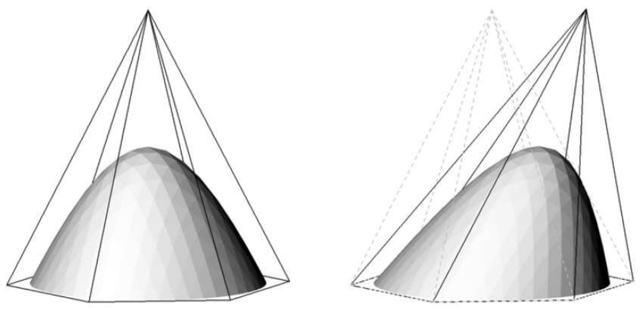


Figure 2.2.2 – Optimisation with subdivision surfaces exploiting the coarse control mesh to modify the shape (Shepherd, 2009)

Since the dome-like structure was designed to function as a hothouse, several criteria had to be met which included an increase of the internal enclosing volume with minimal effect on the total surface area and a reduction of the heating requirements in winter and cooling requirements in summer. From the geographical location, the amount of solar radiation that reached the surface for specific weather conditions and time of the year was calculated and used to evaluate changes to the form. A total score based on a weighted average of the different requirements was used to arrive at the final optimised design (Shepherd, 2009). The result is shown in Figure 2.2.3.

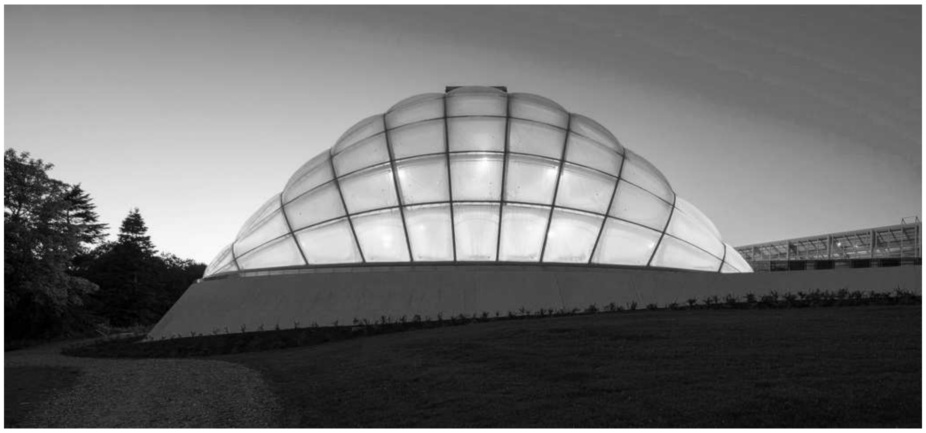


Figure 2.2.3 – Tropical hothouse in the botanical garden of Aarhus, Denmark 2013. Copyright Quintin Lake

The project highlights how subdivision surfaces offer an efficient way of modelling smooth complex geometries with only a few variables that can be used in an optimisation process to inform the design. It demonstrates the integration of the architectural concept in the optimisation process and furthermore how the final design can be enriched by the embedded logic.

2.3 Eigenshells

Michalatos and Kaijima (2014) propose a novel way to obtain a low parametrisation of a surface by a linear combination of the eigenfunctions of a Laplacian matrix. While the eigenfunctions of a Laplacian matrix are widely explored in computer graphics, the application in architecture is to date very limited. Like subdivision surfaces, this method is based on a mesh representation as initial input and consists of three steps; the construction of a Laplacian matrix based on the topology (and possibly geometry) of the mesh, an eigendecomposition of this matrix to extract a range of eigenvectors and eventually a combination of these by a weighted sum, which is interpreted as a displacement field describing the movement of each vertex. In general there exists as many eigenfunctions as there are vertices in the mesh. However, it is possible to obtain a low parametrisation by introducing a filter. As a result, the variables are reduced to the weights associated with each eigenfunction in the linear combination.

Michalatos and Kaijima (2014) explain that these eigenfunctions can be understood as the vibrational modes of a membrane with a given boundary sorted according to their frequency from lowest to highest. This is useful as it is generally only desirable to work with shapes of low frequency (more smooth) in an architectural context, which implies that it is sufficient only to extract the lower range of the eigenvectors. The method therefore has many similarities with Fourier analysis but operates on three-dimensional meshes instead.

The described method has been implemented in the Grasshopper plug-in called "Millipede" (Michalatos and Kaijima, 2015). The author has tested this design tool in order to obtain a better understanding of the methodology in the context of shape generation and modification. Millipede contains seven components related to this topic, which can be described as follows:

- **EigenSystem:** calculates the spectrum of the Laplacian matrix, which is constructed from the input mesh. The eigenvalues and eigenmodes are sorted in ascending order. The eigensystem is used as input for the rest of the components.
- **Extract EigenVector:** extracts a specific eigenvector from a defined index value. The length of the eigenvector is 1.
- **Extract spectrum:** extracts the coordinate spectrum of the mesh for x, y, and z respectively. E.g. for the x coordinate, the spectrum contains a list of numbers specifying one value per mode in the eigensystem.
- **Displace by Spectrum:** deforms the mesh in the normal direction based on a defined spectrum, which corresponds to the reduced variables (weights) from the methodology description. It appears that the length of the spectrum does not have to match the size of the eigensystem. In other words, spectrum value number 1 gets assigned to mode 1, spectrum value

number 2 gets assigned to mode 2 etc. When the spectrum is empty, the spare eigenmodes get a zero value assigned.

- **Reconstruct:** similar functionality as the “Displace by Spectrum” but the spectrum in this case refers to multipliers of the XYZ coordinates (obtained from the “Extract spectrum” component) rather than normal displacements.
- **Spectral Filter:** reconstructs the mesh from its x, y and z spectrum using only a specified range of the eigenmodes. This functionality links to the noise removal aspect of the methodology description.
- **Mesh Visualization:** visualises a specific mode from a defined index value as normal displacements and colouring of the mesh. Multiple indexes can be specified at once but the created meshes need to be separated manually afterwards.

While the tool enables the generation of smooth free-form surfaces from only a few weights by using the “Displace by Spectrum” component, it is not easily understood by the uninitiated user. The so-called spectrum is a rather confusing concept that is not well explained and is furthermore blurred by its relation to both normal displacements and coordinate reconstruction. The author never succeeded in using the “Extract spectrum” and “Reconstruct” component in any meaningful way (unclear how the X, Y and Z spectrum are provided simultaneously) and suspects that the “Spectral Filter” can replace this functionality. Additionally, the spectrum definition for the “Displace by Spectrum” component makes it easy to reduce the number of variables if only the lower range of the eigenmodes is considered but needs to be increased if control over the higher frequency modes is desirable. In other words, it is not possible to select the eigenmodes individually. In a larger scale, the most obvious disadvantage of this tool is the lack of boundary control, which is a very important factor for the tool to be applicable in an architectural context. The paper describes how boundaries can be enforced by introducing a scalar field over the vertices of the mesh with a value of one if the vertex is fixed and a zero value otherwise. This scalar field is translated into a diagonal matrix, which is added to the Laplacian matrix. In effect, the eigenvectors vanish in desirable regions but there is no clear way of providing this information with the existing software tool. Since the sorting of the vertex indexes is arbitrary, the most practical way to implement the described functionality is to select a number of nodes that are intended to be fixed, search for those amongst the vertices in the mesh and automatically generate the diagonal matrix from this information. Colouring the mesh with a gradient, which is mapped back to a value per vertex in the range between 0-1 helps to acquire more control of the boundary transition. However, the latter is not easily defined if the mesh is not symmetric.

In general, the method differs significantly from NURBS and subdivision surfaces by having weights as parameters instead of the more visual control polygon. It makes the method more abstract to understand and less intuitive to use, but still offers a way of obtaining a low parametrisation of a surface. The link to Fourier analysis is an interesting property, which provides this method with a different shape language and possibly enables advantageous analysis besides noise removal to be an integrated part of the modelling process. These unexplored properties and necessary refinements to make it applicable within an architectural context suggest several research opportunities.

Michalatos and Kaijima (2014) also demonstrate how the low parametrisation of the eigenshells can be used in an optimisation process, where the objective is to minimise the maximum deflection. The mesh representation is, similarly to subdivision surfaces, useful to provide a direct link to finite element analysis software. The method has not yet been implemented in the design process of a real project but its potential is visualised through a case study by the authors of the paper as shown in Figure 2.3.1 (the surface pattern is part of another topic, which has been ignored for this context).

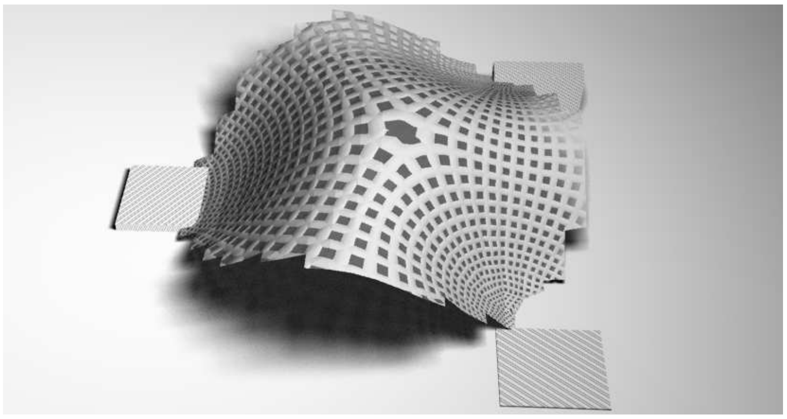


Figure 2.3.1 – Resulting eigenshell from structural optimisation (Michalatos and Kaijima, 2015)

2.4 Curvature stiffened shells

In terms of structural performance, most shells are evaluated by means of a maximum stress or deflection measure in the conceptual design stage including the previously described examples and traditional form-finding techniques. These measures are often used because they are simple and only require first order analysis. However, buckling is more often the governing failure mode for shells, as was experienced for the Trada and Soren Jensen DHL pavilions. The flat areas were especially vulnerable to failure by buckling and this observation suggests that curvature is an important design feature in stiffening a shell.

Malek (2012) conducts a thorough parametric study on the impact of corrugations in relation to the buckling load factor for a barrel vault structure. The parameters include the location of the corrugations (cosine waves), the aspect ratio (frequency and amplitude) and the span-to-height ratio of the shell leading to one-hundred combinations to analyse for a continuous shell and three-hundred for a grid shell. The study clearly proves the gain in buckling capacity by introducing corrugations at either the edge, the crown or both locations at the same time. In the continuous case, it is possible to increase the capacity up to 80 times with corrugations at both the edges and the crown for a 1 % increase in volume. The gain in buckling capacity is less significant for grid shells but a similar behaviour is observed (up to a factor of 8 for less than 3 % increase in volume). The buckling capacity is obtained from a linear buckling analysis (eigenvalue problem), which for the barrel vault is shown to be a conservative measure compared to a non-linear collapse analysis.

The study shows how double-curvature can be used as means of stiffening shell structures (increasing their buckling capacity) and it successfully provides rules of thumb to include these considerations in the conceptual design stage, rather than postponing buckling analysis until the final structural validation. Buckling measurements require a second order analysis to take the deformed shape (or initial stress state) into account, which in general reduces the computational speed. In the study by Malek (2012), the work flow to obtain such a measure utilises a bespoke Matlab script, which from the investigated parameters generates a text file with the necessary information about geometry, support conditions and loads to use as input for the finite element software ADINA. While this work flow is sufficient to set up a number of design rules based on a parametric study, it is too slow to provide upfront feedback in a design process or integrated as part of an optimisation problem. The work flow is furthermore limited to specific types of shells e.g. the barrel vault but should ideally be able to introduce corrugations to any type of shell.

The concept of using double curvature to stiffen shells is not new. In fact, this technique has been used by famous architectural engineers such as Felix Candela (1910-1997) and Eladio Dieste (1917-2000) as seen in Figure 2.4.1. The introduced double curvature adds quality to both the aesthetic character and the structural performance of the shell. Even though this is an appealing property, little research or digital tools exist to explore and evaluate shells stiffened by their curvature.

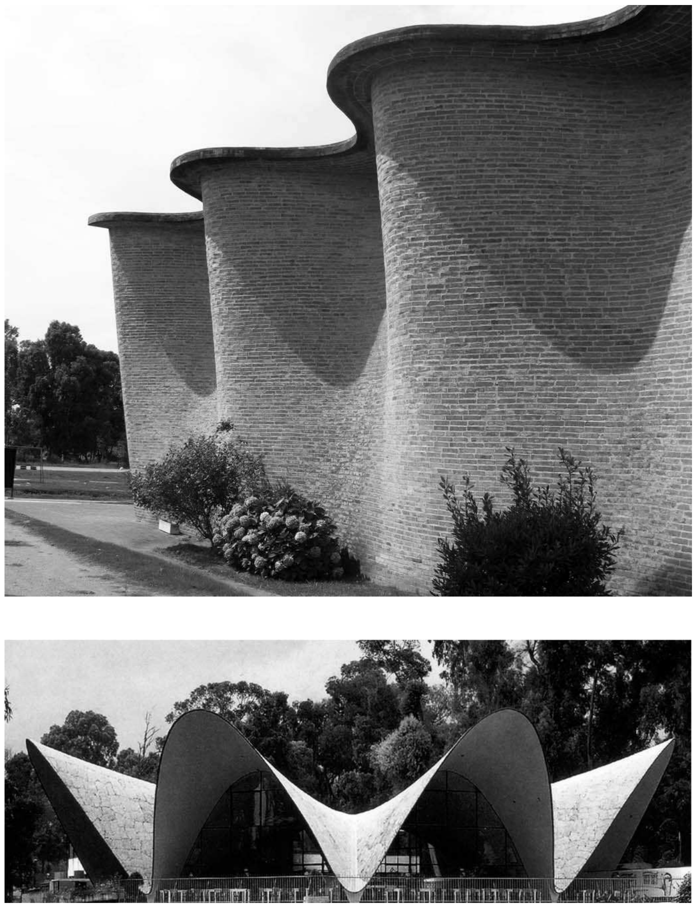


Figure 2.4.1 – Shell design by Eladio Dieste and Felix Candela. Top: Iglesia cristo obrero by E. Dieste, Uruguay 1952 (Wazeone, 2010). Bottom: Los Manantiales by F. Candela, Mexico 1958 (Miller, 2014)

2.5 Thesis direction

The literature review identifies little research exists into shell shapes in relation to their buckling capacity, even though this is considered the dominant failure mode for these structures (Malek, 2012). Experience of traditional form-finding techniques, observations of historic buildings and a newly conducted parametric study by Malek (2012) all point towards the influence of curvature as an essential design parameter to increase the shell stiffness and hence the buckling capacity. The difficulties of obtaining a buckling measure have created opportunities for work flow improvements and in general there exists a need to expand the conclusions outlined in the parametric study of Malek to shell types other than the barrel vault.

The wave-like shapes of the historic curvature stiffened structures (Figure 2.4.1) interestingly links to the eigenshell method, which exactly generates free-form shapes by combining waves of different amplitude and frequency. As discussed, this strategy to obtain a low parametrisation in an architectural context has many interesting unexplored properties and the mesh representation is useful to create a direct link to a buckling analysis. NURBS and subdivision surfaces have already been developed thoroughly and investigated in a structural context.

The specific direction for this research is therefore firstly to explore and further develop the eigenshell methodology to obtain a low parametrisation of a surface with the same flexibility and level of intuition as the NURBS and subdivision surface approaches. Secondly, to exploit the resulting wave-like shape with its inherent doubly curved nature to stiffen the shell against failure by buckling.

2.6 Software development

In an attempt to reach a wider audience (not limited to engineers and programmers only), and for easy integration in practice, the author has chosen to develop the software to interface with McNeel's 3D modelling program named Rhinoceros (McNeel, 2015b). Rhinoceros is a widely used program within architectural practices due to its ease of use, flexibility and high accuracy associated with the modelling of spatial shapes by means of NURBS curves and surfaces (McNeel, 2014a). Triangle and quad meshes are also an integral part of the software, even though the mesh tools are more limited compared with other modelling programs such as Autodesk Maya. Rhinoceros supports customisation through the creation of plug-ins, which are based on the open source RhinoCommon Software Development Kit (SDK). That way it is possible, via scripting, to access predefined geometry classes, take advantage of the existing graphical system to display and navigate around the geometry and use the general functionality implemented in Rhinoceros to avoid writing every method from scratch.

Grasshopper3D is one such free plug-in for Rhinoceros developed by David Rutten (McNeel, 2015a), which has gained much popularity within the last ten years. Due to the wide applicability of Grasshopper, the Rhinoceros modelling program has started to grow into the engineering/contractor practices as well. This makes Rhinoceros/Grasshopper the ideal platform for this software project since it aims at encouraging the dialogue between the architect and engineer in the conceptual design stage, which is enhanced by using the same software. Bespoke components are written by the author in the object-oriented C# programming language using Microsoft Visual Studio developer environment for compiling the programs. For numerical calculations a Matlab COM interface is used in order to call Matlab functions from within Grasshopper (MathWorks, 2015).

Since the harmonic modelling approach uses a mesh representation, a data structure to store the mesh information in an efficient manner is essential. Several different structures exist and they consist of the same general classes (vertex, edge, face and mesh class) but differ in the way they store connectivity information. For this thesis the following observations are made:

- Only 2-manifold meshes are considered i.e. each edge can only have two adjacent faces. This excludes t-junctions and internal polygons (McGuire, 2000)
- The mesh can be of arbitrary topology
- Efficient adjacency queries are a high priority to improve computational speed

A halfedge data structure supports these topological and algorithmic requirements (Botsch et al., 2010) and has recently been implemented under the project name “Plankton” in the chosen software platform (Piker and Pearson, 2015). For these reasons a halfedge mesh structure forms the basis for the functionality implementation as part of this thesis.

Chapter 3

Harmonic modelling

In this chapter the mathematical framework behind modelling with harmonics is explained. Initial tests serve to verify a correct implementation and to develop an intuitive understanding of the design tool. From the basic framework further refinements are introduced, guided by several smaller examples to make the tool more suitable for architectural and engineering applications.

3.1 Framework

Harmonic modelling is essentially a method of performing Fourier analysis on meshes (Lévy and Zhang, 2009). The reader is referred to Appendix A for a more detailed description of this topic. It was originally developed by Taubin (1995) in the context of surface fairing but has recently found several other applications including mesh quadrangulation, mesh segmentation and geometry compression (Zhang et al., 2010). In this thesis the method is used to achieve a low parametrisation of a given mesh.

This low parametrisation is the key to developing a free-form modelling tool that holds the same level of intuition and ease of modification as NURBS and subdivision surfaces. The main function of the low parametrisation is therefore to create a relation between the vertices in the mesh to guarantee smoothness and at the same time provide sufficient design flexibility, ideally such that any shape can be modelled. The harmonic modelling approach is fundamentally different from NURBS or subdivision surfaces. For the latter, the control polygon and thus the modification of surfaces becomes tangible due to the spatial configuration it operates on, whereas the variables associated with the harmonic modelling are numerical values that control the summation of waves with different frequency. This concept is more abstract but it has other advantages: It is known from Fourier analysis that representing a function in the frequency domain provides information that it is not possible to extract from the spatial domain, which enhances the understanding of what the function is composed of and thereby allows data manipulation. Furthermore, the design language is very different due to the summation of waves and in some situations this may be beneficial to broaden the architectural expression of shells and provide efficient structural solutions.

Botsch et al. (2010) identify the translation of the theory behind Fourier analysis to arbitrary meshes is not directly possible without the missing link observed by Taubin (1995); *"the classical Fourier transform of a signal can be seen as the decomposition of the signal into a linear combination of the eigenvectors of the Laplacian operator"*. Mathematically, this can be expressed as

$$\underline{\mathrm{L}} \cdot \vec{v} = \lambda \cdot \vec{v} \tag{3.1.1}$$

Where $\underline{\mathrm{L}}$ is the Laplacian matrix, λ are the eigenvalues and $\vec{v}$ are the eigenvector. This eigenvalue problem has many similarities with a finite element modal analysis, which performs an eigendecomposition of the stiffness matrix of a structure in order to describe the vibrational modes (Cook, 1995). Various information such as choice of material and degrees of freedom are necessary to construct this stiffness matrix, which is undesirable in the context of free-form modelling in the conceptual design stage. The advantage of the Laplacian operator is that it is purely related to geometry but nevertheless it can be thought of as a simplified stiffness matrix. This coherence gives a physical interpretation of the eigenvectors as the vibrational modes of a mesh and the eigenvalues as the squared frequencies $\sqrt{\lambda} = f$ (Dong et al., 2006).

The following sections contain a more detailed description of the Laplacian operator, its implementation and how to perform an eigendecomposition in order to obtain a more intuitive understanding of the framework. Subsequently, these parts are combined to form the harmonic modelling set-up.

3.1.1 The discrete Laplacian

The Laplacian is a second order differential operator in n-dimensional space defined by

$$\triangle = \frac{\partial^2}{\partial x_1^2} + \ldots + \frac{\partial^2}{\partial x_n^2} \tag{3.1.2}$$

It describes the divergence of the gradient, which for non-mathematicians can be thought of in 3D as how much gradient comes into a point (infinitesimal cube). To get a better general understanding of the operator the rewriting of the second derivative (Herholz, 2012) is helpful

$$\frac{\partial^2 f}{\partial x^2} = \lim_{h \to \infty} \left(\frac{f(x+h) + f(x-h) - 2f(x)}{2h^2} \right) \tag{3.1.3}$$

From this equation it is seen that the Laplacian describes the difference between a function value in x and the average of a small neighbourhood. For meshes this is equivalent to the difference between a function value in a specific vertex and the (weighted) average of the 1-ring neighbourhood. The discrete Laplacian operator is defined (Reuter et al., 2009) as

$$\triangle f(\mathbf{v_i}) = \frac{1}{d_i} \cdot \sum_{j \in N(i)} w_{ij} \cdot (f(\mathbf{v_i}) - f(\mathbf{v_j})) \tag{3.1.4}$$

Here $w_{ij} = w_{ji}$ is a symmetric edge weight, $N(i)$ is the set of vertices included in the 1-ring neighbourhood of vertex v_i and d_i is the mass associated with vertex v_i. Even though the Laplacian only incorporates local information it is still capable of acting globally and reveals properties that are unique to the given mesh (Zhang et al., 2007).

Equation 3.1.4 can be rewritten in matrix form as an n x n matrix where n is the number of vertices in the mesh. Vallet and Lévy (2008) conclude that a discrete Laplacian that meets all the properties of the continuous operator cannot exist on general meshes. Therefore several different discretisations of the Laplacian exist, where each one tries to capture specific properties. Rewriting Equation 3.1.4 in matrix form generally does not result in a symmetric matrix due to the possibly non-uniform mass. However, for the scope of this thesis the symmetry of the Laplacian matrix is essential to guarantee real eigenvectors that create an orthogonal basis. This matrix structure can be written as

$$L_{ij} = \begin{cases} -w_{ij} & if\ i \neq j\ and\ v_i\ is\ adjacent\ to\ v_j \\ \sum\limits_{j \in N(i)} w_{ij} & if\ i = j \\ 0 & otherwise \end{cases} \tag{3.1.5}$$

Here L_{ij} is a symmetric matrix where each row and column sums up to zero. The simplest discretisation of the operator is the graph Laplacian, which is defined by an edge weight equal to 1 and with a uniform mass distribution. It means that the matrix has -1 in off-diagonal cells where two vertices are connected by an edge, and the sum of the edge weights on the diagonals thus represents the valence (Michalatos and Kaijima, 2014). Geometrically this is interpreted as a vector pointing from a given vertex towards the barycentre of its 1-ring neighbourhood.

This definition of the graph Laplacian was the one initially used by Taubin (1995) to set up the framework. The definition of the graph Laplacian is very simple and easy to compute and is applicable to arbitrary mesh topologies but the drawback is that it solely depends on that topology. In other words, it does not adapt to any non-uniform or spatial distribution of the vertices but it only changes if the topology is modified, which makes the vertex valence a sensitive issue. Another consequence of the definition is that the Laplacian vector can be non-zero even for a flat mesh. However, in such configuration the Laplacian is expected to be zero since it represents the divergence of the gradient and the gradient for a flat mesh is zero. This motivates the definition of a Laplacian, which is both topology and geometry aware with zero vectors for flat meshes.

As a result, Pinkall and Polthier (1993) have derived the widely used geometric mesh Laplacian with cotangent weights based on mesh energy considerations. A more intuitive derivation of the same formula by Desbrun et al. (1999) uses the gradient of the area of all the triangles in the 1-ring neighbourhood of a vertex to achieve the described property. This derivation is based on the observation that the area does not change if the centre vertex moves in-plane whereas a movement out of plane increases the area. In other words, the area function has a local minimum for a flat mesh configuration and therefore a zero gradient value. Hence, the desired property of a zero Laplacian vector for locally flat regions is obtained no matter the vertex valence, edge lengths or aspect ratio of faces. The cotangent edge weights are defined as

$$w_{ij} = \cot(\alpha_{ij}) + \cot(\beta_{ij}) \tag{3.1.6}$$

Where α_{ij} and β_{ij} are the angles opposite the the edge connecting the vertices v_i and v_j as illustrated in Figure 3.1.1. In order to talk about an *opposite* angle this implies that the mesh is triangulated. Due to the missing triangle for an edge located at the boundary α_{ij} or β_{ij} is set to zero corresponding to Neumann boundary conditions (Vallet and Lévy, 2008).

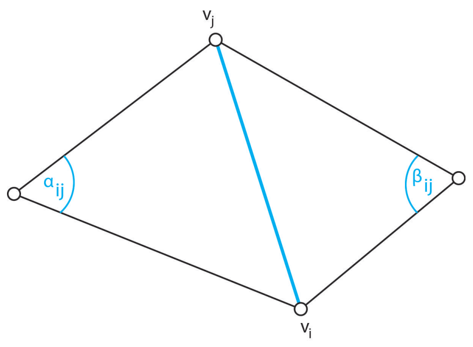


Figure 3.1.1 – The angles associated with calculation of the cotangent weights

A geometrical comparison between the graph Laplacian and the Laplacian with cotangent weights is shown in Figure 3.1.2, which confirms the desired zero vector property for flat meshes. From this it is not surprising that the cotangent weights are used in the calculation of the mean curvature of meshes as well.

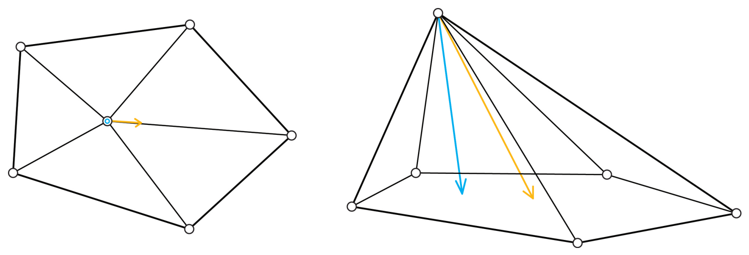


Figure 3.1.2 – A geometrical comparison between the graph Laplacian (orange) and the Laplacian with cotangent weights (blue)

The disadvantage of the cotangent weights is that for obtuse triangles the value becomes negative, which have the potential to cause problems depending on the application (Botsch et al., 2010). Another disadvantage of this discretisation of the Laplacian is that it lacks a proper mass weighting, which means that the weights are dependent on the mesh density (Reuter et al., 2009). Desbrun et al. (1999) solves this problem by introducing mass weightings based on the area of a local neighbourhood associated with each vertex. The boundary of this neighbourhood is created by straight lines connecting the midpoints of the radiating edges from a given vertex and the barycentre of the adjacent faces associated with the same vertex thereby forming so-called *barycells* as seen in Figure 3.1.3. By definition this construction partitions one triangle into three regions of equal area making it simple to compute the area of each barycell as seen in Figure 3.1.4a. The area of one triangle is calculated as half the magnitude of the cross product of two adjacent edge vectors in a face. Meyer et al. (2002) modifies this approach by using the area created by straight lines connecting the midpoints of the radiating edges from a given vertex and the circumcenters of the adjacent faces associated with the same vertex thereby forming so-called *voronoi cells* as seen in Figure 3.1.3b-c.

In both cases a perfect tiling of the mesh is obtained meaning that no overlapping or non-covered areas exist but Meyer et al. (2002) argues that the voronoi cells give a better approximation of the values obtained from the continuous Laplacian operator since the voronoi cells exactly contain the closest points to each vertex. This means that the voronoi area only depends on vertex positions and not the connectivity (Jacobson, 2010). Working with voronoi areas require more attention and computation time because if obtuse triangles exist in the mesh then the circumcenters lie outside the boundaries causing problems for the area calculation. Meyer et al. (2002) have developed a hybrid approach to take obtuse triangles into account by fixing the circumcentre to the midpoint of the edge opposite to an obtuse angle whereby one of the edges in the voronoi cell collapses to a point. The voronoi area calculation around each vertex therefore includes a test of each adjacent triangle specifying whether it contains an obtuse angle or not. Based on that evaluation the area contribution from one triangle is calculated from (referring to Figure 3.1.4b-c)

$$A_v = \begin{cases} \frac{1}{8} \cdot \left(\parallel PR \parallel^2 \cdot \cot(\angle Q) + \parallel PQ \parallel^2 \cdot \cot(\angle R) \right) & non-obtuse \\ \frac{A_T}{2} & obtuse\, and\, P \geq \frac{\pi}{2} \\ \frac{A_T}{4} & otherwise \end{cases} \tag{3.1.7}$$

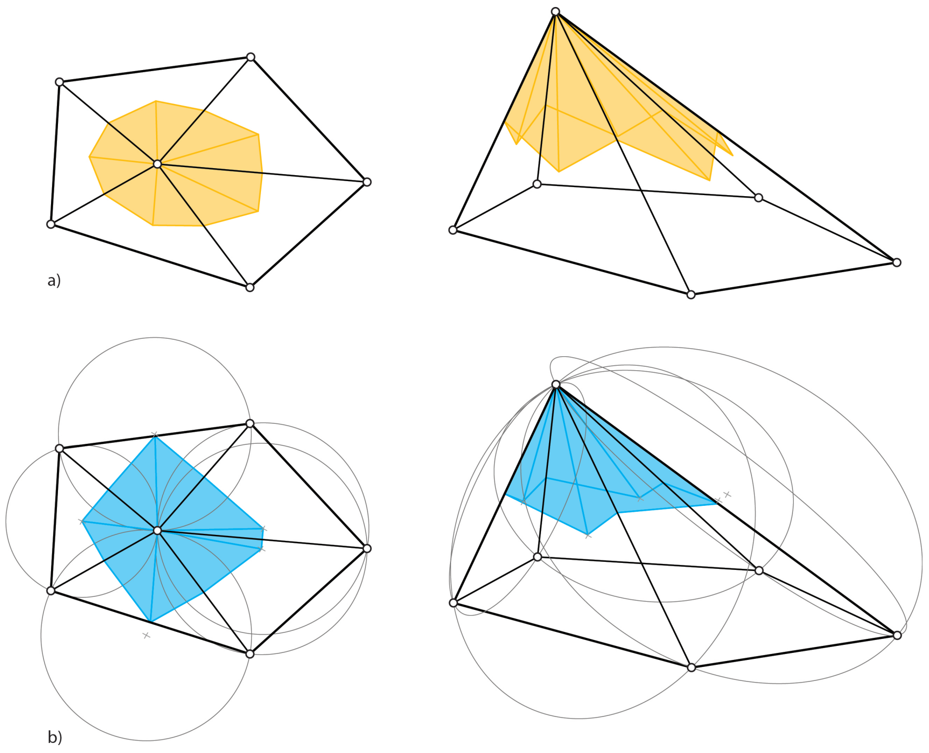


Figure 3.1.3 – a) barycell and b) voronoi cell for a flat and spatial configuration

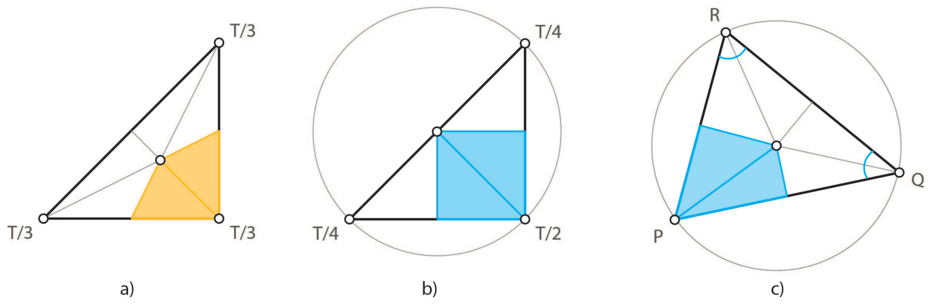


Figure 3.1.4 – Area calculation of a) barycell, b) voronoi cell (obtuse/right angle) and c) voronoi cell (non-obtuse)

The vertex areas can be included in Equation 3.1.6 in several ways to account for the mass however it often leads to an asymmetric matrix because the area associated with vertex v_i is not necessarily the same as the area associated with the vertex v_j. Vallet and Lévy (2008) propose a symmetrisation of the problem and it is shown that this weighting meets the desired property of making the Laplacian operator mesh independent but compromises the property of having a zero Laplacian vector for flat mesh regions (only small deviations). The edge weight is defined as

$$w_{ij} = \frac{\cot(\alpha_{ij}) + \cot(\beta_{ij})}{\sqrt{A_i \cdot A_j}} \tag{3.1.8}$$

The choice between the graph Laplacian and the cotangent Laplacian and the different area weighting options for the latter depends on the application but it is summarised to the following observations. The graph Laplacian can be constructed for any mesh topology, it is computationally very fast and only needs to be calculated once (under the assumption that the mesh topology remains unchanged). It performs well if the mesh is very regular with similar edge lengths. The disadvantage is its sensitivity to vertex valence meaning that vertices with low valence have less “stiffness”. The cotangent Laplacian works for triangulated meshes only and is advantageous if it is desirable to let the spatial configuration of the mesh influence the result. The area weighting option is useful if the density of the vertices varies within the mesh, otherwise the cotangent Laplacian without any area weighting suffices. The area weighting using barycells is less computationally heavy but the voronoi cells are the better choice if it is desirable to be independent of the topology of the triangulated mesh.

Implementation and validation

The different discretisations of the Laplacian as discussed above are implemented in two Grasshopper components (graph Laplacian and cotangent Laplacian) where the output in both cases is characterised by a real symmetric n x n matrix. Due to the more complex definition of the cotangent Laplacian a small test case is created to validate the results, as shown in Figure 3.1.5 . The output from the Grasshopper component with the three different area weighting options is shown in Figure 3.1.6.

This output is compared with results obtained by hand-calculations using Equation 3.1.6 and Equation 3.1.8. Initially, the areas associated with each vertex are constructed manually and measured with Rhino’s built-in area function with the results shown in Table 3.1.1.

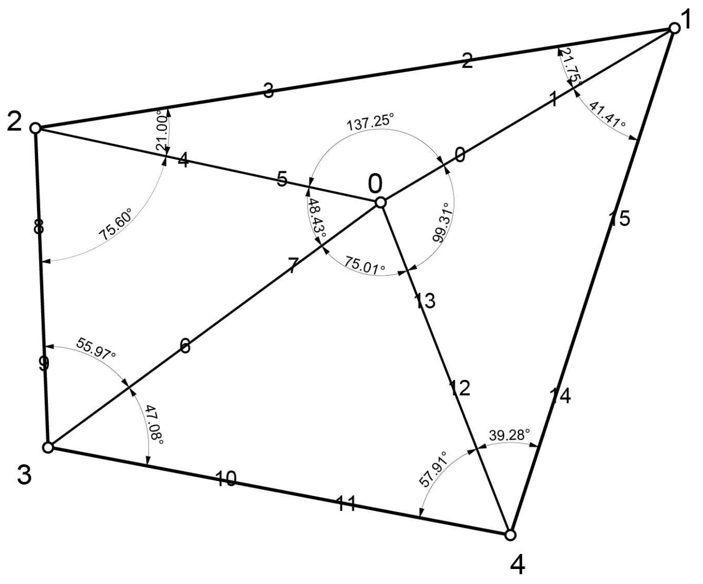
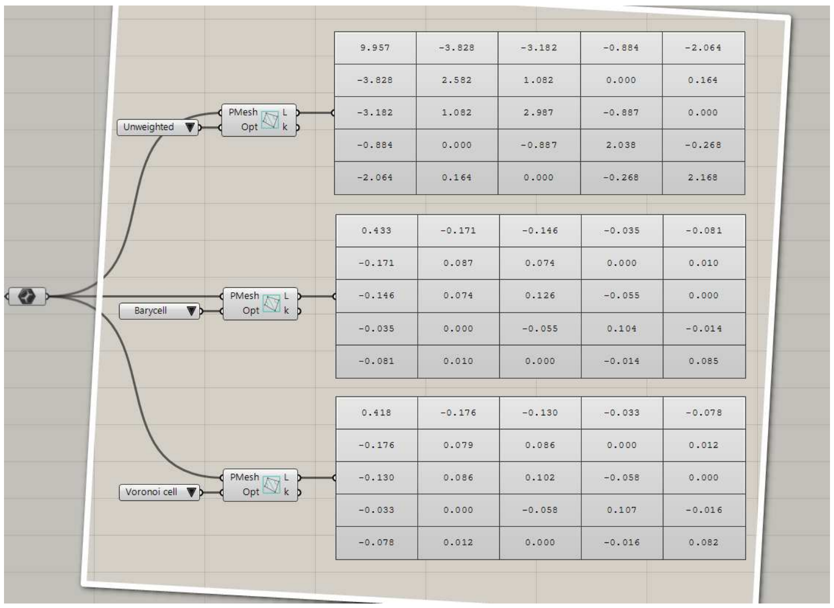


Figure 3.1.5 – Test case to validate the implementation of the discrete Laplacian



9.957	-3.828	-3.182	-0.884	-2.064
-3.828	2.582	1.082	0.000	0.164
-3.182	1.082	2.987	-0.887	0.000
-0.884	0.000	-0.887	2.038	-0.268
-2.064	0.164	0.000	-0.268	2.168

0.433	-0.171	-0.146	-0.035	-0.081
-0.171	0.087	0.074	0.000	0.010
-0.146	0.074	0.126	-0.055	0.000
-0.035	0.000	-0.055	0.104	-0.014
-0.081	0.010	0.000	-0.014	0.085

0.418	-0.176	-0.130	-0.033	-0.078
-0.176	0.079	0.086	0.000	0.012
-0.130	0.086	0.102	-0.058	0.000
-0.033	0.000	-0.058	0.107	-0.016
-0.078	0.012	0.000	-0.016	0.082

Figure 3.1.6 – Grasshopper implementation of the cotangent Laplacian with different area weighting options

v #	$\mathbf{A}_{ii}$ (Barycell)	$\mathbf{A}_{ii}$ (Voronoi cell)
0	33.58	42.08
1	14.98	11.23
2	14.14	14.23
3	18.61	16.66
4	19.44	16.55

Table 3.1.1 – Vertex areas from manually constructed voronoi- and barycells

The angles are measured with Rhino's angle dimension tool and used in the calculation of the cotangent edge weights as seen in Table 3.1.2.

e #	v #	$\mathrm{w_{ij}}$ (unweighted)	$\mathrm{w_{ij}}$ (Barycell)	$\mathrm{w_{ij}}$ (Voronoi cell)
0, 1	0, 1	3.825	0.171	0.176
2, 3	1, 2	-1.082	-0.074	-0.086
4, 5	0, 2	3.183	0.146	0.130
6, 7	0, 3	0.884	0.035	0.033
8, 9	2, 3	0.887	0.055	0.058
10, 11	3, 4	0.268	0.014	0.016
12, 13	0, 4	2.065	0.081	0.078
14, 15	1, 4	-0.164	-0.010	-0.012

Table 3.1.2 – Cotangent edge weights from different area options

The vertex weights are subsequently calculated as the sum of the adjacent edge weights as seen in Table 3.1.3.

v #	e #	$\mathrm{w_{ii}}$ (unweighted)	$\mathrm{w_{ii}}$ (Barycell)	$\mathrm{w_{ii}}$ (Voronoi cell)
0	0, 5, 7, 13	9.960	0.433	0.418
1	2, 1, 15	2.580	0.087	0.078
2	3, 4, 8	2.990	0.127	0.102
3	9, 6, 10	2.040	0.104	0.107
4	11, 12, 14	2.170	0.085	0.082

Table 3.1.3 – Cotangent vertex weights from different area options

A comparison between the off-diagonal matrix values in Figure 3.1.6 and the edge weights from Table 3.1.2 (note the opposite sign according to Equation 3.1.5) as well as the diagonal matrix values in Figure 3.1.6 and the vertex weights from Table 3.1.3 shows compliance and thus validates the implementation. With this confidence, the implementation is refined to map the calculated areas into a predefined range from 0-10 in order to make the output independent on the scale of the mesh. The upper limit ensures that rounding error problems in Grasshopper are avoided (referring to Equation 3.1.8).

3.1.2 Eigendecomposition

As the discrete Laplacian operator is defined as a real symmetric matrix it has real eigenvalues and a set of real and orthogonal eigenvectors (Lévy and Zhang, 2009), which form the desired basis equivalent to the complex exponential function used for the Fourier Transform (see Appendix A). However, one noteworthy distinction is that the basis functions are fixed for the Fourier Transform but change for the extension to meshes depending on mesh connectivity, geometry and type of Laplacian (Zhang et al., 2007).

As mentioned, these eigenvectors are physically interpreted as the vibrational modes of the mesh, whereas the eigenvalues are associated with the frequencies. The first eigenvalue is always zero, which corresponds to an eigenvector of constant values implying a rigid body motion. These physical properties are important because they enable a sorting of the eigenvectors according to the eigenvalues, such that the mode which requires the least energy for the original shape to deform into (or geometrically speaking the most smooth shape) is listed first and it is exactly what makes this particular basis interesting compared to any other orthogonal basis.

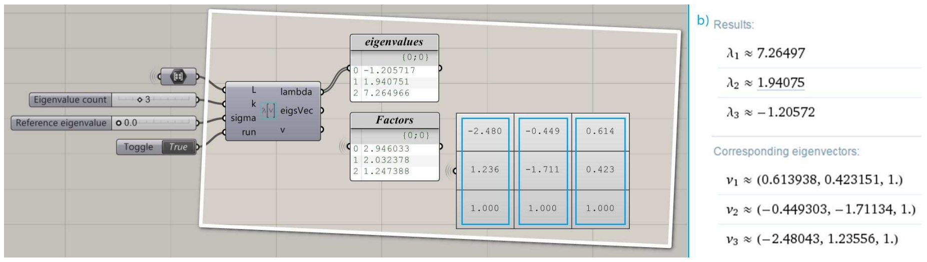
Implementation and validation

To calculate the eigenvalues and eigenvectors of a matrix in Grasshopper it is sensible to use an external library. The author has chosen Matlab for this purpose, based on the experience gained by Gebreiter (2012) in his work on obtaining a quadrilateral mesh from a similar spectral approach. Matlab provides efficient iterative algorithms to perform an eigendecomposition of a large sparse matrix and it allows the specification of a desired number (k) of eigenvalues/vectors to be computed given a reference eigenvalue. By default, the eigenvectors are sorted in ascending order according to the eigenvalues and normalised to unit length.

To verify the implementation, a simple 3 x 3 real symmetric matrix is defined as

$$\underline{M} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 2 & 1 \\ 3 & 1 & 5 \end{bmatrix}$$

The eigenvalues/vectors are calculated using WolframAlpha (WolframAlpha, 2015), which is an online mathematical calculation engine, and subsequently compared with the values obtained from the implemented Grasshopper component (linking to Matlab). Since WolframAlpha scales the eigenvectors in such a way that the last value of each vector equals one, it is necessary to find the scale factors between the two outputs for comparison.



```
eigenvalues
{0;0}
0 -1.205717
1 1.940751
2 7.264966
```

```
Factors
{0;0}
0 2.946033
1 2.032378
2 1.247388
```

```
-2.480  -0.449  0.614
 1.236  -1.711  0.423
 1.000   1.000  1.000
```

Results:

$\lambda_1 \approx 7.26497$

$\lambda_2 \approx 1.94075$

$\lambda_3 \approx -1.20572$

Corresponding eigenvectors:

$v_1 \approx (0.613938, 0.423151, 1.)$

$v_2 \approx (-0.449303, -1.71134, 1.)$

$v_3 \approx (-2.48043, 1.23556, 1.)$

Figure 3.1.7 – Comparison of eigendecomposition results between the implemented Grasshopper component (a) and WolframAlpha (b)

The result is shown in Figure 3.1.7. For the Grasshopper component the eigenvalues are shown in the upper list and the corresponding eigenvectors (columns of the matrix) are shown in the matrix below. The eigenvectors are scaled similarly to WolframAlpha with the displayed factors. The pairs of eigenvalues/vectors are sorted in ascending order for the Grasshopper component and in descending order for the WolframAlpha results. With that in mind for the comparison, it is clear that the results are identical.

The following example serves to build a better visual understanding of the described harmonic behaviour from a decomposition of a Laplacian matrix associated with a string.

Example - eigenvectors of a string

A graph representing a string is constructed from 8 equally spaced vertices with edges connecting them. The Graph Laplacian is defined as

$$
\underline{L} = \begin{bmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1
\end{bmatrix}
$$

The result of computing the eigendecomposition of this matrix and plotting the first 6 eigenvectors separately as the vertical displacement is shown in Figure 3.1.8. The displaced vertices are in each case connected by a polyline and compared to a sinusoid with the same wave length and amplitude if possible.

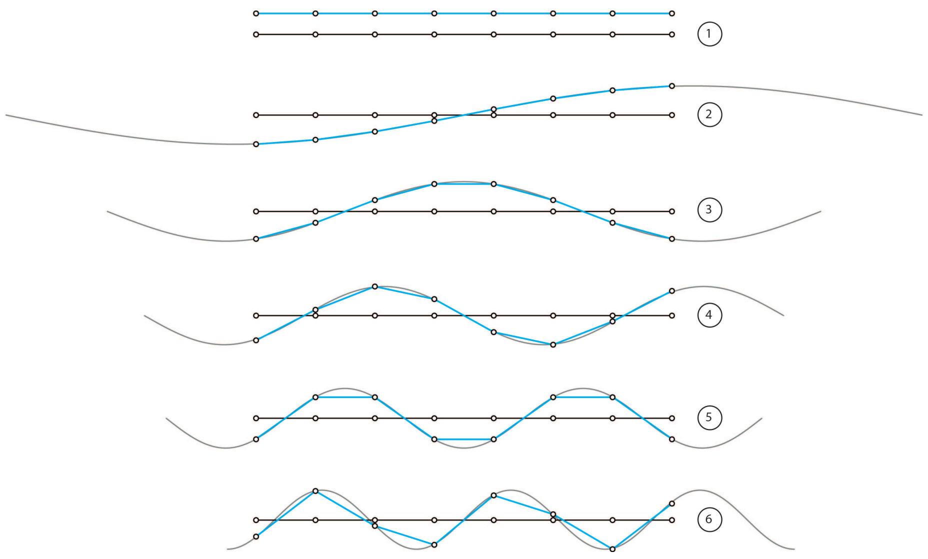


Figure 3.1.8 – The first six modes of a 8-vertex string (blue) compared with sinusoids of similar amplitude and frequency (grey)

The figure clearly illustrates the harmonic behaviour of the eigenvectors and thus the coherence with the Fourier transform. The string nicely approximates a continuous sinusoid of the same wave length and amplitude for the first 4 modes but as the frequency increases, the spacing between the sample points becomes too big to capture the details. Hence, the location of the vertices and overall refinement of the string are important to capture the peaks and troughs and ensure symmetry of the solution.

3.1.3 Harmonic shape generation

When working in 3D with meshes, it is possible to move each vertex in the x, y and z directions in order to control its position. An important design decision therefore is to choose the number of degrees of freedom to be controlled by the harmonic modelling tool. If all three translational degrees of freedom are to be controllable, it means that each eigenvector has to be replicated three times in order to assign one eigenvector to the movement in the x, y and z directions respectively, which in effect triples the number of variables. Unfortunately, the eigenvector replication does not take the stiffness of the structure in each direction into account and therefore introduces undesirable in-plane vibrational modes as part of the lower frequency domain. In comparison, the stiffness matrix in a finite element analysis takes three translational and three rotational degrees of freedom into account and therefore holds information about the axial and bending stiffness of each element in relation to global directions. As a result, the structure becomes much stiffer in-plane than out-of-plane, which in effect makes the in-plane vibrational modes associated with higher frequencies only. From these considerations, and to make the design tool more intuitive by having fewer variables, the author has chosen to implement a single degree of freedom approach. To be as versatile as possible within this restriction, the vertex normals are chosen as the direction of movement. Each normal is calculated as the weighted average of the adjacent face normals, where each face normal is computed as the average of the cross products of all edge pair vectors in that face. The face normals are not normalised since they contain information about the face areas from their lengths. Thus, for each vertex the adjacent face normals are summed up and eventually normalised. Some of the limitations due to this single degree of freedom approach are discussed later in this chapter.

The key concept behind the modelling with harmonics is that any n-dimensional vector (representing the normal displacements) can be constructed as a linear combination of the computed eigenvector basis. The coefficient (referred to as a weight) related to each eigenvector determines the amplitude of that wave in the total shape. A low parametrisation of the mesh is obtained when a certain amount of the higher order frequencies are cut off in order to only generate smooth shapes. Thereby the number of variables is reduced from one value per node to one weight per included mode!

Mathematically, the construction of a new n-dimensional vector $\vec{v}$ from a collection of eigenmodes is expressed by

$$\vec{v} = w_a \cdot \begin{bmatrix} e_{0,a} \\ e_{1,a} \\ \vdots \\ e_{n-2,a} \\ e_{n-1,a} \end{bmatrix} + w_b \cdot \begin{bmatrix} e_{0,b} \\ e_{1,b} \\ \vdots \\ e_{n-2,b} \\ e_{n-1,b} \end{bmatrix} + \ldots + w_c \cdot \begin{bmatrix} e_{0,c} \\ e_{1,c} \\ \vdots \\ e_{n-2,c} \\ e_{n-1,c} \end{bmatrix} \tag{3.1.9}$$

Where w_a is the weight associated with eigenvector $\vec{e}_a$ and $a, b, c \in \mathbb{Z} \mid 0 \leq a, b, c < n$. This vector describes the magnitude of movement of the vertices along their normals. The weights are limited to a range between -1 and 1 (a minus value reverses the wave) and an overall scale factor is introduced to resize the shape as desired.

Implementation and validation

This functionality of translating the computed eigenvectors into normal displacements by specifying the weights for a linear combination is implemented in a new Grasshopper component as seen in Figure 3.1.9. In order to reduce the number of variables, an additional component is implemented to extract the desired k eigenvectors (columns) from the entire n x n matrix. The initial mesh, reduced eigenvector matrix, vertex normals, weights and scale factor are given as input to the component and the mesh with adjusted vertex positions is output. The mesh is coloured to further visualise how the new shape is generated by mapping each value from the vector resulting from the linear combination into the integer interval from 0 - 255 used for grey scale. If the range of the values in the vector is zero (constant vector values) then all vertices are assigned a black colour.

By only extracting one eigenvector each mode can be visualised separately. Alternatively, this can also be achieved by extracting the first k eigenvectors to keep the number of variables fixed and subsequently assign one weight with a value of 1 (or -1) and set the rest to zero. This is useful to verify that the results correspond to the ones found in the literature.

A planar square mesh consisting of 10x10 quadrilateral faces is used as basis for such a comparison The first nine eigenvectors are calculated (out of the

121 possible) from the graph Laplacian matrix to illustrate the first eight non-constant modes (the first mode is a pure translation). The result is seen in Figure 3.1.9 and it corresponds well with the results obtained by Dong et al. (2006) as shown in Figure 3.1.10. The only noticeable difference is that mode 4 and 5 have switched places but a further investigation shows that the eigenvalues/frequencies are the same for these two modes and the sorting thus becomes arbitrary.

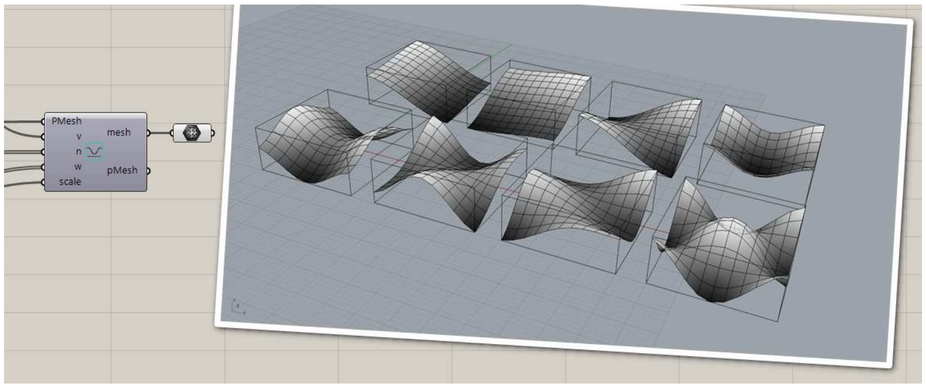


Figure 3.1.9 – The first 8 non-constant mode shapes of a flat square mesh resulting from the Grasshopper implementation

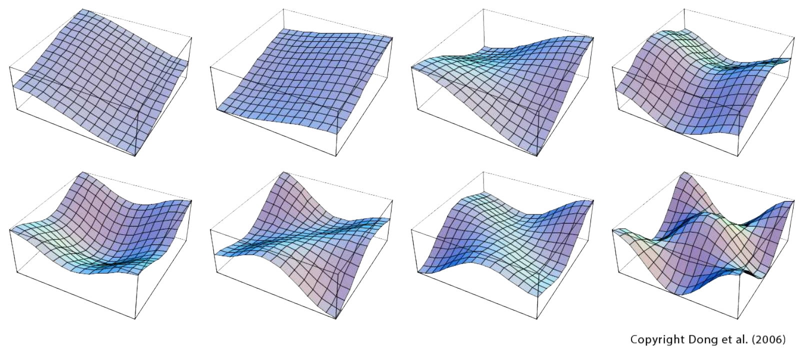


Figure 3.1.10 – The first 8 non-constant mode shapes calculated by Dong et al. (2006)

Visualisation

As most shapes will be a combination of modes rather than an individual mode it is important to enhance the intuition associated with the modelling such that it becomes more instinctive to choose and combine modes as well as adjust the corresponding weights. For this purpose two visualisation Grasshopper components are developed. One facilitates the display of a specified range of the modes arranged in a grid structure in the Rhino viewport as seen in Figure 3.1.11. This mode catalogue is useful for a general overview to better find the desirable modes to include. The second component facilitates the display of selected modes on the Grasshopper canvas by providing the necessary input for the "Squid" plug-in (Zwierzycki, 2015). This visualisation component is mostly useful when the desirable modes have already been chosen (see Figure 3.1.12). The modes are displayed next to the slider weights such that the user knows what is being controlled as shown in Figure 3.1.13.

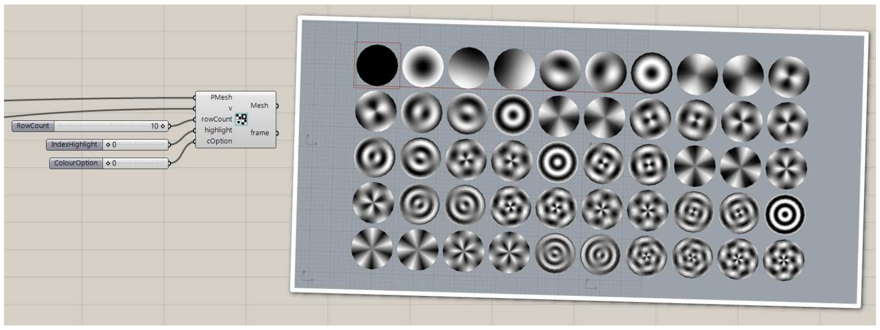


Figure 3.1.11 – The implemented Grasshopper component to visualise a selected range of the mode catalogue in the Rhino viewport

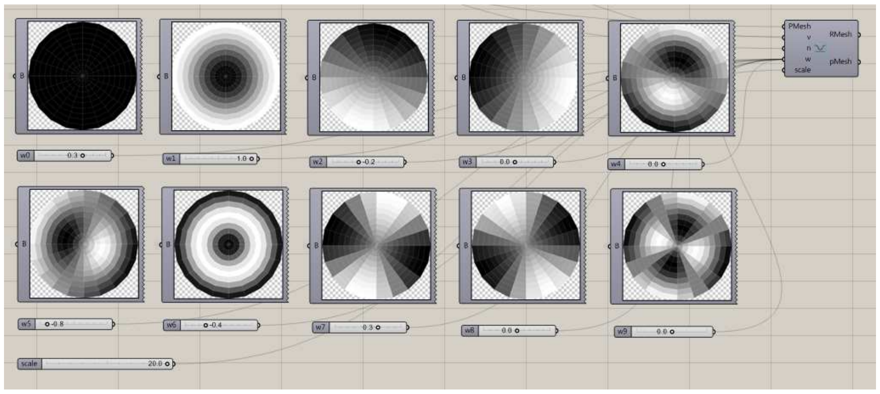


Figure 3.1.12 – The implemented Grasshopper component to visualise some selected modes on the canvas to accompany the corresponding weights

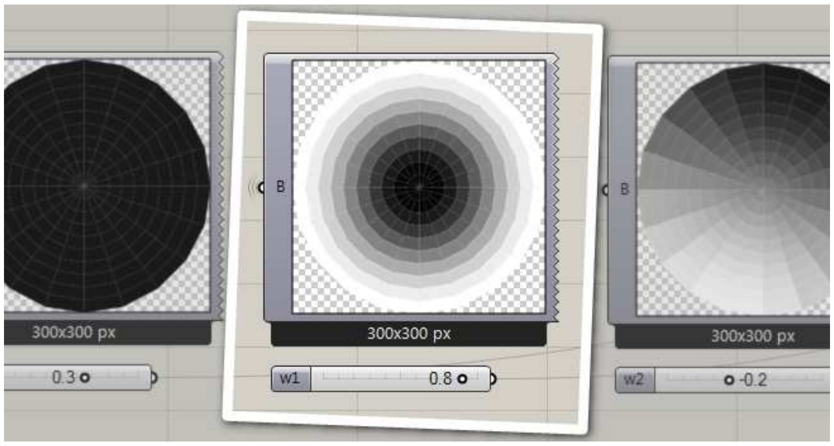


Figure 3.1.13 – Mode with corresponding weight slider so the user knows what is being controlled

The meshes shown in the Rhino viewport are coloured per vertex and a gradient scheme ensures a smooth colour perception whereas the more rough colouring of the meshes on the Grasshopper canvas is due to a face based scheme (average of face vertex colours).

3.2 Refinements

The framework described in the previous section makes it possible to generate harmonic shapes from a starting mesh, only by controlling a few weight parameters and an overall scale factor. However, it is essential to be able to control the boundary conditions in order for this method to be applicable in an architectural or engineering context. This is therefore the main focus of the refinements. Additionally, the coherence between the harmonic modelling approach and the Fourier Transform is further explored with the intention to approximate, analyse and remove noise from existing surfaces.

3.2.1 Boundary conditions

Due to the single degree of freedom design approach it is only possible to control whether a vertex is fixed or not, hence no sliding boundary supports can be applied. The problem becomes to manipulate the Laplacian matrix in such a way that this control is acquired. In other words, how to artificially manipulate the values in the matrix such that the values in the computed eigenvectors corresponding to the fixed vertices equal zero and the neighbouring vertices are adjusted accordingly. This property is achieved by constructing the Laplacian matrix according to its general definition (Equation 3.1.5) for the entire mesh and then subsequently increase the diagonal values related to the fixed vertices to a large number e.g. 100,000. That way the connectivity information is kept intact and the large ratio between the artificial vertex stiffness and the rest of the values in the matrix ensures that the modes associated with a movement of the vertices intended to be fixed appear last in the eigenvector list because they require the most energy. It means that these modes still exist but are excluded by only asking for the first k modes, where k is defined as

$$k \leq n - c \tag{3.2.1}$$

Here n is the number of vertices and c is the number of imposed constraints. The result of this action is visualised for a 15-vertex string in Figure 3.2.1 and it demonstrates how the ends are successfully fixed while the mode shapes at the same time closely approximate sinusoids of similar frequency and amplitude.

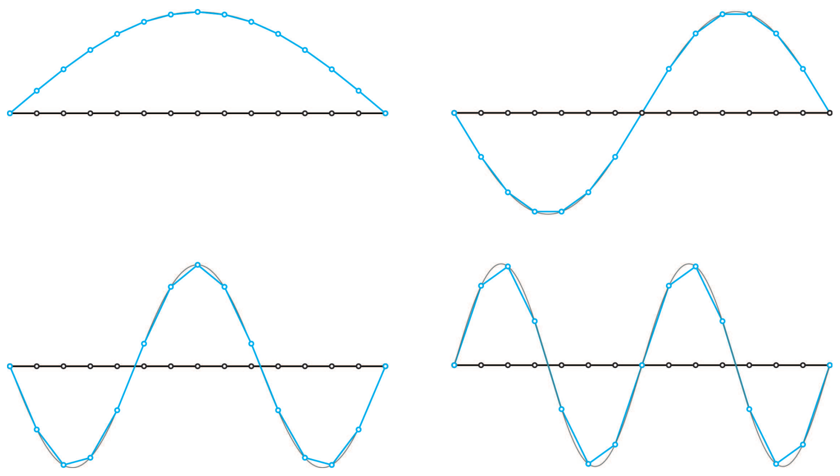


Figure 3.2.1 – The first four displacement eigenfunctions of a 15-vertex string with fixed ends (blue) in comparison with continuous sinusoids of similar wave lengths and amplitudes (grey)

This methodology is similar to the one described by Michalatos and Kaijima

(2014) as outlined in Chapter 2. However, a more detailed study of the boundary conditions shows that adding a value of 1 to the diagonals of the Laplacian matrix corresponding to the fixed vertices is insufficient for the purpose of this thesis (see Appendix B).

Implementation and validation

The described methodology is implemented in a Grasshopper component and the first eight modes of a flat square mesh fixed along its boundary using the graph Laplacian is shown in Figure 3.2.2. The component receives a list of points which are intended to be fixed, searches for them amongst the vertices of the mesh to find their indexes and eventually replaces the current diagonal values with a high stiffness value at these indexes in the Laplacian matrix.

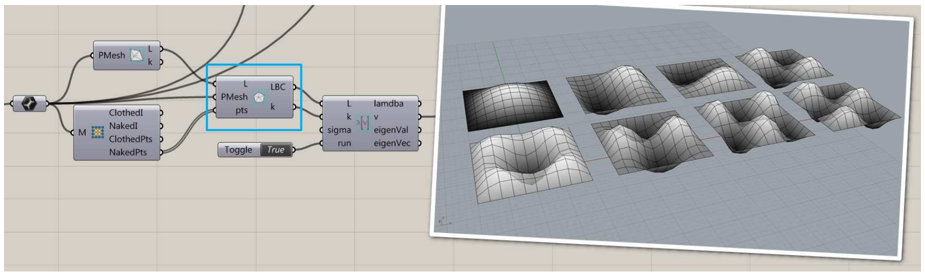


Figure 3.2.2 – The first eight modes of a flat square mesh with fixed boundaries. The Grasshopper component, which imposes the constraints is highlighted in blue

To validate the results, the modes are compared with the results obtained from a modal analysis in Autodesk Robot (Autodesk, 2015b). A 10x10 m plate meshed into a 10x10 grid forms the foundation for the analysis. In order to perform a modal analysis in Robot the plate is assigned a concrete material of the type C25/30, a thickness of 200 mm and pinned supports along its boundary. Even though the calculation in Robot is based on a different matrix with 6 times more degrees of freedom, a coherence with the results obtained from Grasshopper is expected because the same mesh is used where the uniform edge lengths for the graph Laplacian approach correspond to the isotropic material properties and uniform thickness in the finite element approach. This is also seen from Figure 3.2.3 (the scale is arbitrary).

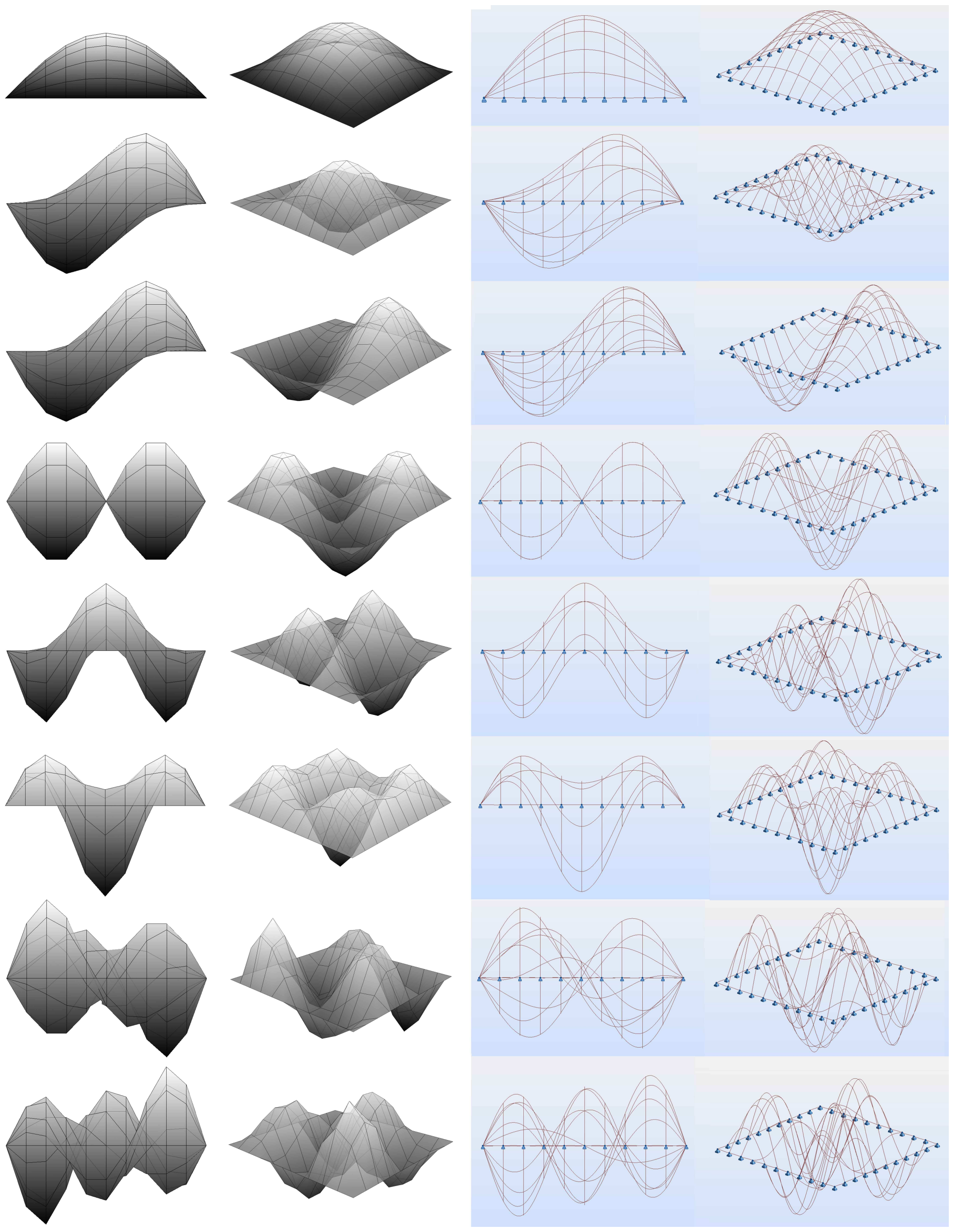


Figure 3.2.3 – Comparison between the first eight mode shapes of a flat square mesh calculated from an eigendecomposition of the graph Laplacian (left) and a modal analysis in Autodesk Robot (right)

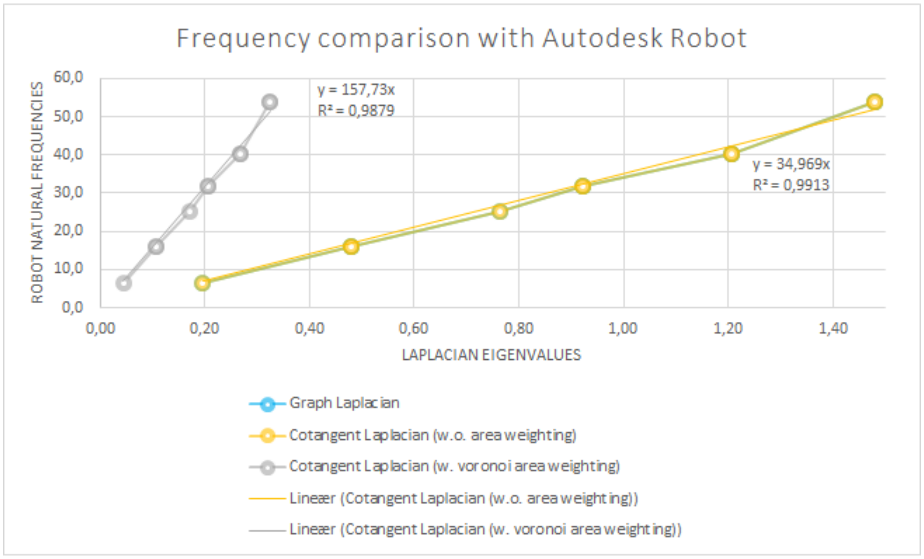


Figure 3.2.4 – Frequency comparison between the results obtained from a modal analysis in Autodesk Robot and the eigenvalues from an eigendecomposition of the different discretisations of the Laplacian matrix for the ten first modes of a flat square mesh. A linear relation is observed

The frequencies are also compared and a linear relation is observed as shown in Figure 3.2.4. This is a surprising result, as Dong et al. (2006) identify that the eigenvalues represent the squared frequencies, which is a reasonable expectation since the natural frequencies of a structure in a finite element analysis are calculated by (Cook, 1995)

$$\left[\underline{\mathbf{K}} - \omega^2 \underline{\mathbf{M}}\right] \vec{d} = 0 \tag{3.2.2}$$

where $\underline{\mathbf{K}}$ is the stiffness matrix, $\underline{\mathbf{M}}$ is the mass matrix, ω is the natural frequency and $\vec{d}$ is the vibration mode. Since both methods are eigenvalue problems based on a stiffness matrix it is expected that ω and λ are directly comparable. However, the unexpected linear relation is most likely a consequence of the different strategies to include the mass, which again relates to the degrees of freedom of the entire system. While the area weighting of the cotangent Laplacian mimics the mass matrix in the finite element method, it is unlikely that the chosen symmetrisation method (Equation 3.1.8) is equivalent to the mass matrix constructed from shape functions and thus also includes rotary inertia. Another effect of this methodological variation is evident from studying the mode shapes close to the boundaries. In these regions non-smooth transitions are observed even though the purpose of the area weighting of the cotangent Laplacian is to make the results independent of the mesh topology (see Figure 3.2.5). This behaviour only arises when boundary constraints are imposed and it is a consequence of an increased stiffness of the edges connecting the boundary vertices with the internal vertices. As the area surrounding a boundary vertex is smaller than the area surrounding an internal vertex, the area product is smaller which in turn increases the edge weight. In effect, the vertices connected to the boundaries are pulled too far down compared to the internal vertices. This behaviour does not occur with the finite element method, which emphasises the need for further refinements in relation to the boundary conditions.

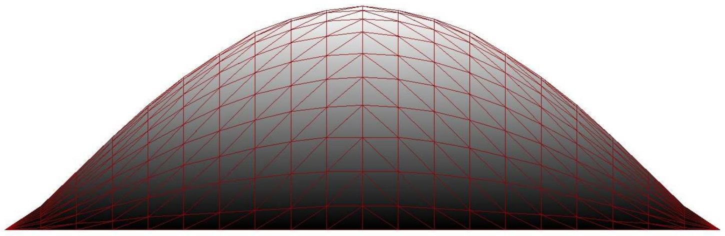
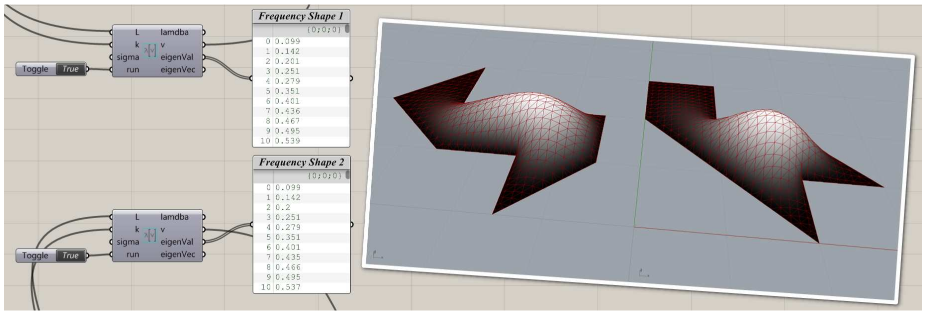


Figure 3.2.5 – Non-smooth boundary transition for area-weighted cotangent Laplacian.

While the eigenvalues therefore have to be translated into natural frequencies with caution, they are still useful as a means of sorting the modes according to quantify how noisy they are. Regardless of the frequency relation, the implementation provides sufficient accuracy to answer the question *"Can one hear the shape of a drum?"* as the following example shows.

Example - Can one hear the shape of a drum?

Kac (1966) raised this question based on the observation that the tones from a drum depend on the frequency at which the membrane vibrates. The question can be reformulated as *"Are the frequencies and thus the tones of a drum unique to its shape?"*. Only in 1992 were two different 2D shapes with identical frequencies successfully constructed (Wikipedia, 2015a), which proved that it is *not possible* to hear the shape of a drum. The tool developed for this thesis verifies this result by using the planer geometries from 1992, creating a mesh from them, fixing the meshes along their boundaries, defining the Laplacian matrix (cotangent without area weighting) in each case and calculating the eigendecomposition. The list of (very close to) identical frequencies, as well as the first mode shape of each drum, are shown in Figure 3.2.6.



```
Frequency Shape 1
         {0;0;0}
 0 0.099
 1 0.142
 2 0.201
 3 0.251
 4 0.279
 5 0.351
 6 0.401
 7 0.436
 8 0.467
 9 0.495
10 0.539
```

```
Frequency Shape 2
         {0;0;0}
 0 0.099
 1 0.142
 2 0.2
 3 0.251
 4 0.279
 5 0.351
 6 0.401
 7 0.435
 8 0.466
 9 0.495
10 0.537
```

Figure 3.2.6 – Two different shapes with identical frequencies

Convergence

With imposed boundary conditions it is observed that some modes vary as the mesh is refined as shown for a square mesh with fixed boundaries in Figure 3.2.7. For a coarse mesh it appears that a diagonal direction is favoured while a straight orientation becomes more significant with the mesh refinement (evident from mode 2, 3, 9, 10). The sorting of mode 5 and 6 is arbitrary since they have the same frequency.

The reason for this behaviour can be explained from Figure 3.2.8, where the blue circles indicate the "freedom" of each vertex measured as its distance to the boundary. For the coarse mesh (left) the paths with the most freedom are through the diagonals where the circles are largest. The mode shapes therefore favour this direction in an attempt to minimise the energy to deform from its original state. As the mesh is refined (right), more "freedom" is pushed towards the boundaries, which allows more flexibility in a straight direction.

Mode shape variations are similarly observed when performing modal analyses in Autodesk Robot and it is therefore evaluated to be a natural phenomenon that is not a side-effect of the artificial boundary conditions.

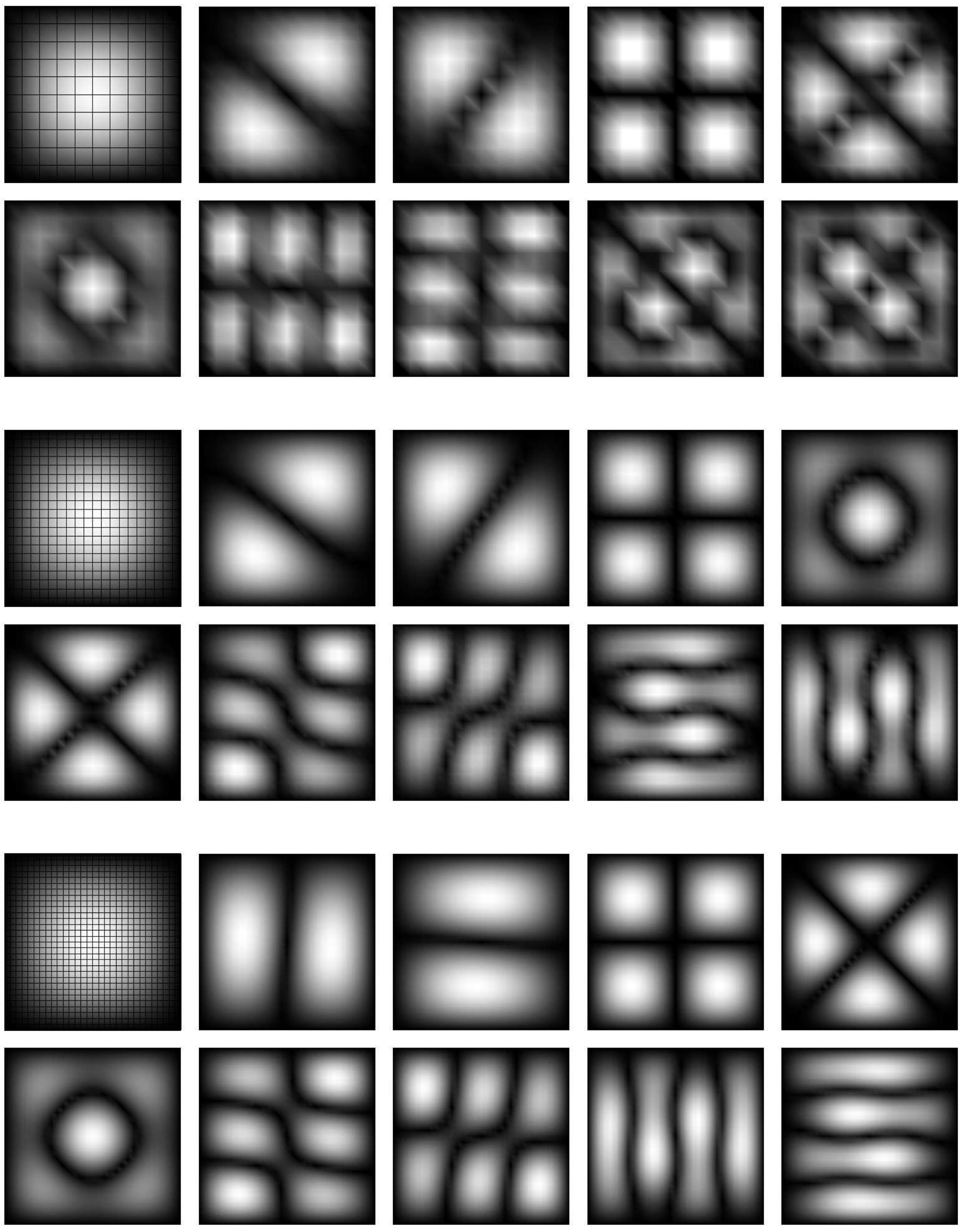


Figure 3.2.7 – Mode convergence from mesh refinement (10x10, 20x20, 30x30). The colouring is based on absolute values of the vertex displacements.

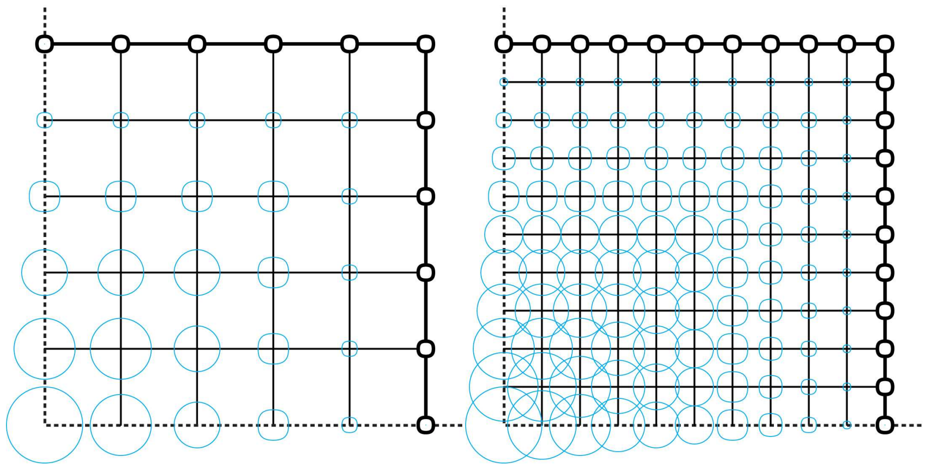


Figure 3.2.8 – Diagonal versus straight mode behaviour from mesh refinement. The dotted lines are symmetry axes and the blue circles indicate the "freedom" of each vertex measured as its distance to the boundary

Limitations

Due to the single degree of freedom design approach it is not possible to define sliding supports or control the rotation at these locations. The latter is useful to specify the tangency at the boundaries, which is a desirable design handle. An attempt to acquire this control can be thought of as coupling vertices in the mesh together such that they move the same amount. One immediate limitation hereof is that it will only work for a plane, cylinder or sphere (constant curvature) because the values in the eigenvectors are translated into normal displacements. Theoretically, identical values can be achieved by increasing the vertex stiffness for diagonal values in $\underline{\mathbf{L}}$ corresponding to the coupled vertex pair and replacing the off-diagonal edge weights (e.g. -1) with a similarly high stiffness value of opposite sign such that the rows and columns still sum up to zero. In Figure 3.2.9 the first four modes of a 15-vertex string are illustrated, which tries to mimic reflective symmetry boundary conditions of a simply supported beam. While the horizontal tangent is maintained, the transition to the internal vertices is not taken into account and as a result just postpones the problems to the vertex neighbours. In contrast, the 6 times larger stiffness matrix in finite element analysis, which includes rotational degrees of freedom makes it possible to adjust the vertex positions such that a smooth transition occurs.

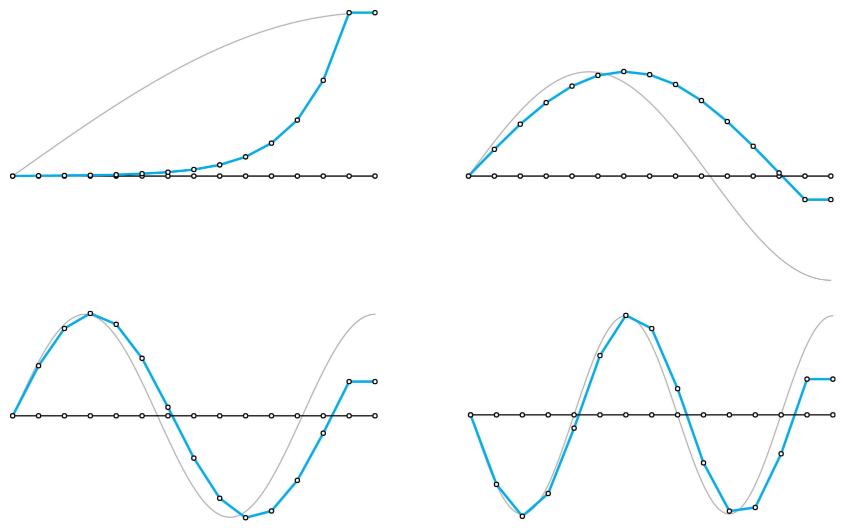


Figure 3.2.9 – Rotational d.o.f.'s by vertex coupling. Showing the first four modes of a 15 vertex string (blue) compared with the expected result (grey) for a simply supported beam with reflective symmetry boundary conditions

3.2.2 Target approximation and shape analysis

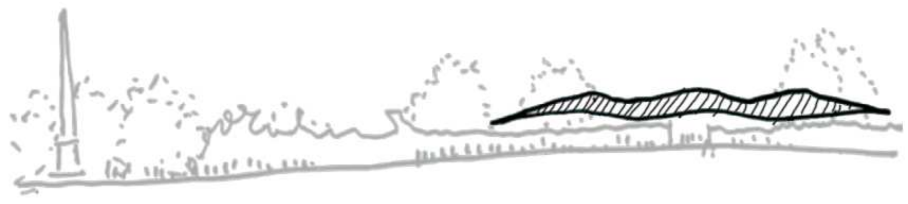


Figure 3.2.10 – Saville Garden conceptual sketch (copyright Glenn Howel)

With the given set-up, it is generally hard to find the desirable modes in the mode catalogue and to mix them with proper weights to match e.g. a sketched shape as shown in Figure 3.2.10 for the Saville Garden Gridshell. Thus, the following question arises: *"Given a starting mesh, is it possible to back-calculate which modes to use and their corresponding weights in order to reach a target surface?"*

An initial approach may be to define the problem as an optimisation exercise and use a certain distance measurement from the generated mesh to the target surface as the object function to be minimised. In order to obtain reasonable results, it is necessary to limit the number of variables significantly as the design space to search is too big with as many variables as there are vertices in the mesh. However, there is no indication of which modes to look for in the mode catalogue and the method is therefore not very suitable.

Fortunately, an analytical solution to this question is surprisingly simple - the beauty of mathematics. In this case, the target can be expressed as a signal which relates to the initial mesh. Since the vertices of the mesh are displaced in the normal direction, the signal is exactly given as the vector that contains n normal distance values measured from the n vertices in the initial mesh to the target surface. Projecting this signal onto each eigenvector using the dot product gives the weights (one weight per mode) and can be interpreted as the amplitude of a specific wave that exists in the given signal. The theoretical background for this observation is further explained in Appendix A. By sorting the modes according to their corresponding weights in descending order such that the modes with the highest weights (absolute value) are listed first, it is possible to extract useful information about the primary ingredients of the target surface and to significantly reduce the number of variables. The reduction of variables is possible because many of the modes do not exist in the defined signal (or only with very small weights) and thus their weights can be rounded off to zero. This enables a sorting of the modes according to their significance in relation to the target surface, which is useful to obtain a low parametrisation with a good approximation but also helps to understand what the target surface is built up from and gives the designer control to remove undesirable noisy ingredients.

While this method successfully solves the initial goal of approximating a target surface, two questions may remain unclear in the larger picture of free-form modelling and are thus outlined in the following.

1. *If a target surface already exists, why not just use that for the design?* As this tool is intended for the early conceptual design stage, where no final shape yet exists, this method opens up for new inspirational forms emanating from the initial target. Thereby new shapes can be explored within a more restricted design space.

2. *Why is it useful to approximate the target surface from a different starting mesh instead of just converting the surface into a mesh and using that directly with its right spatial configuration?* The approximation approach has the advantage of making shape analysis possible as part of the modelling

process itself. That way noise can be removed to create cleaner geometries and the decomposition of the target into its harmonic components in relation to e.g. a flat structure gives an enriched geometrical understanding of the shape. It is also possible to use the structure as starting mesh and then approximate the target surface representing the façade from this configuration to e.g. determine the lengths of the sticks to attach the two parts.

Implementation

The functionality of back-calculating the significant modes with their weights to approximate a target is implemented in a Grasshopper component as shown in Figure 3.2.11. The initial mesh, target surface, vertex normals, eigenvectors, number of variables, sorting option and pre-set slider value option are given as input. The target is represented as a NURBS surface as the author considered this to be the most common case in practice.

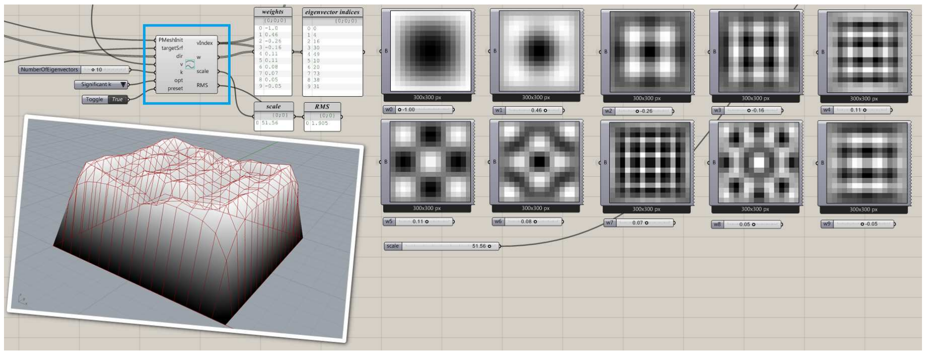


Figure 3.2.11 – The implemented Grasshopper component to back-calculate the necessary weights and their corresponding modes to approximate a cube

As described, a distance signal is essential to the back-calculation process. The construction of this signal involves more steps to take several scenarios into account; firstly the closest point on the target surface from a given vertex is found and if the distance if less than a threshold it is assumed to already lie on the surface and the distance is therefore zero. If it does not already lie on the surface a “RayShoot” method from the RhinoCommon SDK (McNeel, 2014b) is used to find the first intersection with the target surface by shooting a ray pointing from a given vertex in the normal direction and afterwards calculating the distance between the vertex and the intersection point (Figure 3.2.12 a).

The shooting direction is reversed if no intersection point is found from the first attempt (Figure 3.2.12 b). An error occurs if still no intersection point is detected and it indicates that the initial mesh has to have a proper relation to the target surface (Figure 3.2.12 c). One limitation associated with this method is therefore that only the first intersection between the target surface and a ray is detected, which means that it is not possible to approximate a double layered surface. Overhangs can be handled to a certain degree by adjusting the initial mesh to follow the target in a better way. Thus, it is evaluated that this limitation does not impose any significant restrictions for free-form surface design seen from an architectural perspective.

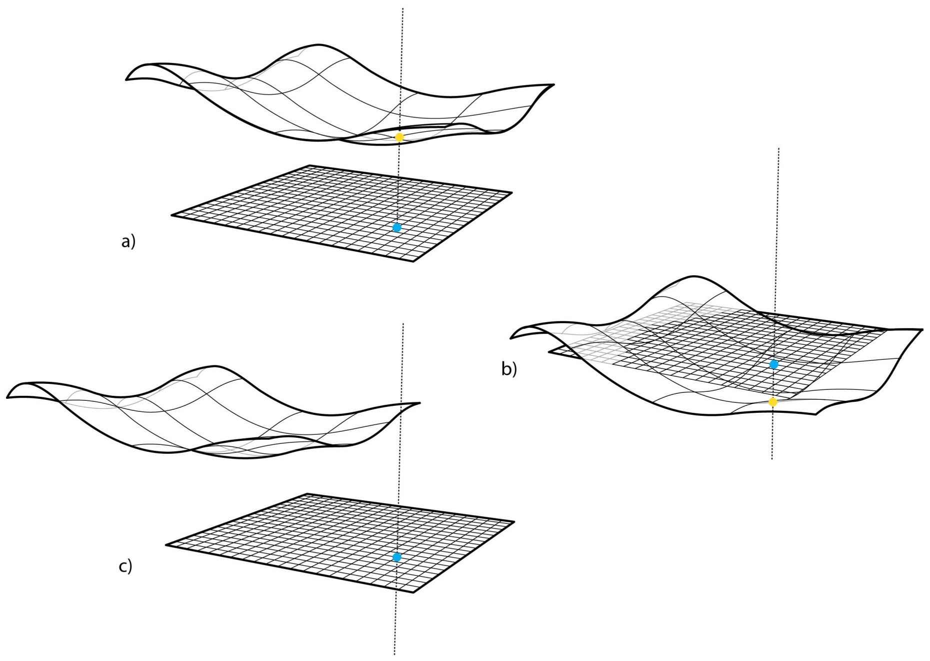


Figure 3.2.12 – Rayshoot method to create a distance to target signal. a) The first intersection with the surface from a ray in the normal direction. b) The first intersection with the surface in the reversed normal direction since the first attempt failed. c) No intersection between the surface and the ray in both directions is detected, which causes an error to be raised

The component outputs the necessary information (mode indexes, weights and scale factor) for the eigenfunction component to translate it into a normal displacement of the vertices (to be consistent with the existing work flow). The approximation result is evaluated by means of a root mean square (RMS) value of the distances from each vertex in the approximation mesh to the target surface measured in the normal direction from the initial mesh (i.e. in the direction of the movement).

Accuracy

From Fourier analysis it is well-known that a better global approximation of the target function is achieved by increasing the number of modes that are included in the summation. Similarly, it is expected that an increasing number of modes with corresponding weights results in a better approximation of the target surface evident from a decreasing RMS value. In the discrete setting only a finite number of modes are available (equal to the number of vertices) and as shown from the square wave example in Appendix A an accurate result, which coincides with the sample points (distance signal) is obtained when all the available waves are included. In other words, a RMS value of zero is expected if all the modes with their calculated weights are combined. This property is mostly useful to validate the correctness of the theory and implementation. In practice the aim is rather to obtain a low parametrisation of the mesh in which case a plot of the RMS value as a function of the number of included modes helps to understand this trade-off and decide a cut-off limit. In this context, boundary conditions are useful because they help to achieve a better approximation of the target with fewer variables.

Simple approximation cases of a cone, cylinder and cube are used to verify that the method exhibits the expected behaviour and thereby demonstrates the Fourier transform extended to three-dimensional meshes.

Example - Approximation of a cone, cylinder and cube

A flat circular mesh divided in radial and tangential directions is used as the initial mesh for the approximation of a cone and a cylinder respectively. The cone surface is modelled with the same radius and an arbitrary height. The cylinder is modelled as a translated disc since a cylinder cannot be represented as a single NURBS surface patch. The boundary vertices are fixed and the graph Laplacian is used to create the necessary matrix since the mesh consists of non-triangular faces. The harmonic shapes generated from an increasing number of the most significant modes are illustrated in Figure 3.2.13 and Figure 3.2.14.

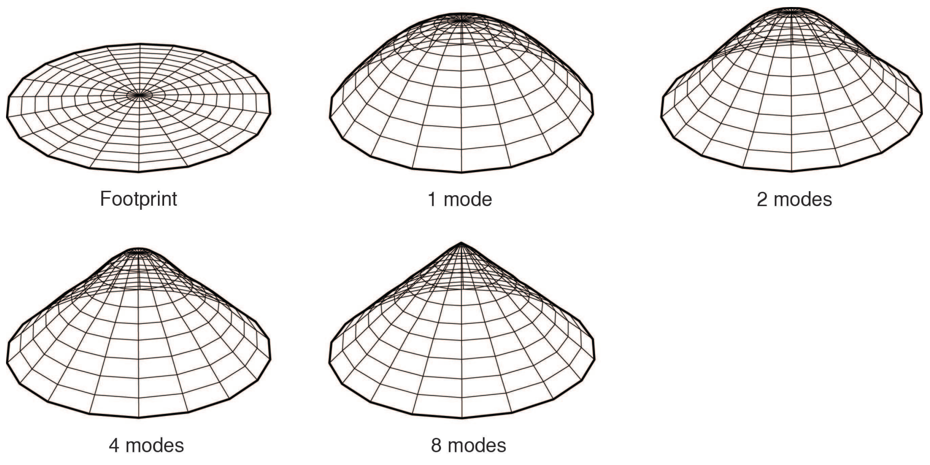


Figure 3.2.13 – Approximation result for a cone

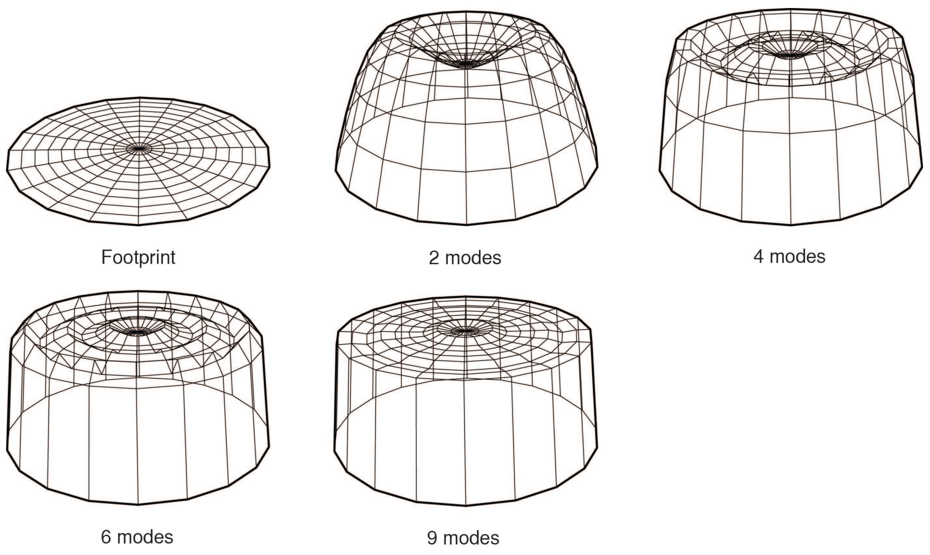


Figure 3.2.14 – Approximation result for a cylinder

A flat quadrangular mesh is used as the initial mesh for the approximation of a cube. Similar to the cylinder, the cube is modelled as a translated square. The boundary vertices are fixed and the graph Laplacian is used for the same reason. The harmonic shapes resulting from an increasing number of the most significant modes are illustrated in Figure 3.2.15.

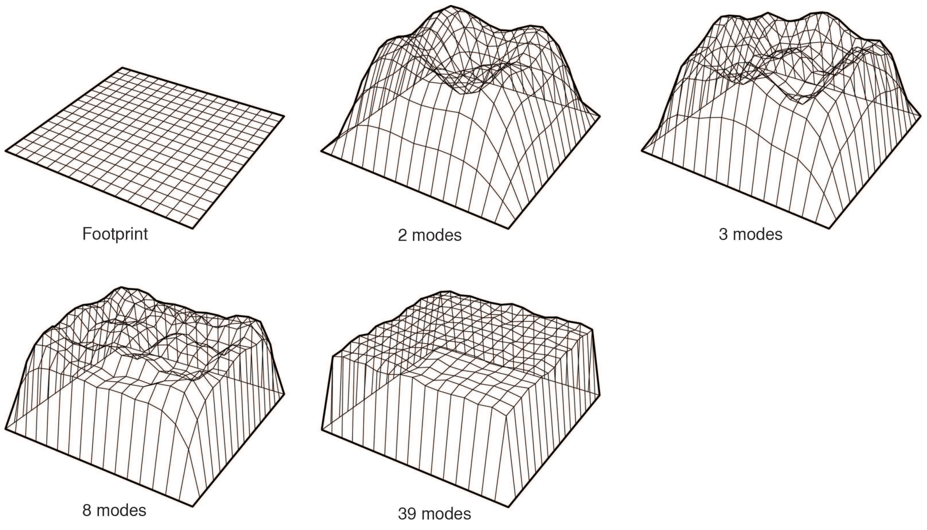


Figure 3.2.15 – Approximation result for a cube

A plot of the RMS value as a function of the number of included significant modes to approximate each target surface is shown in Figure 3.2.16. The RMS value decreases as the number of modes increases, which is expected based on the knowledge from Fourier analysis. The reason why the RMS value for the cylinder and the cube does not converge to zero is because the boundary vertices (fixed) and the target surface do not coincide at this location so a certain deviation is expected. The speed of convergence towards zero (or a constant) depends on the complexity of the target. In this case 50 modes for the cube and 10 modes for the cone and cylinder were necessary to achieve the minimum RMS value. The example shows promising results in terms of the accuracy that is obtained with 10 variables or less.

This analytical approach is superior in comparison with other optimisation strategies. While the back-calculation method provides a solution in less than 25 ms for a cube with the first 10 modes, it takes the built-in Grasshopper component "Galapagos" 1 minute and 44 seconds with a simulated annealing search strategy and 17 minutes for an evolutionary strategy to achieve similar results.

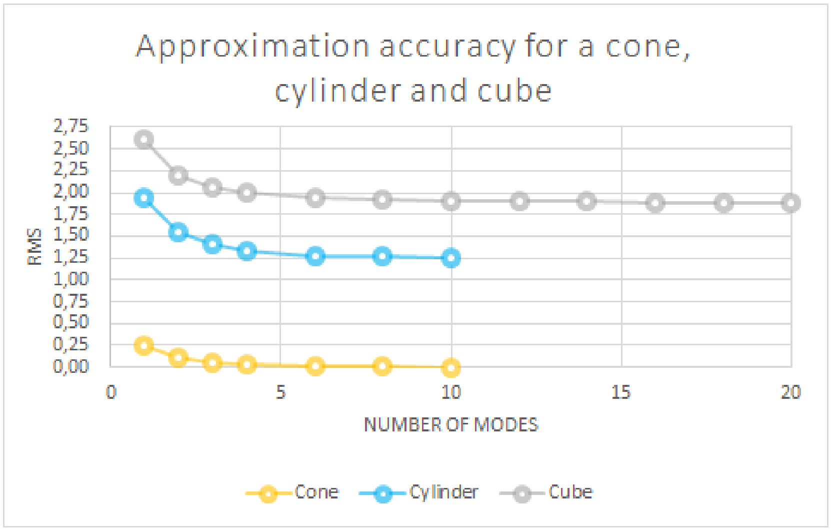


Figure 3.2.16 – Approximation accuracy for a cone, cylinder and cube

Application in practice

In practice it is most likely a mixed approach between the arbitrary harmonic modelling and the target approximation that will be employed. This work flow is characterised by the following steps:

1. Model a NURBS surface to imply the spatial intention
2. Create a simple mesh of desirable refinement to be used as basis for the surface approximation
3. Impose boundary conditions (optional), choose the type of the Laplacian matrix, compute the eigendecomposition, and back-calculate for example the 10 most significant modes and their weights to approximate the target surface
4. Display these modes to achieve a better understanding of what the surface is built up from. Optionally remove noisy modes. Adjust the weights from this starting point to explore alternative design possibilities related to the design space that is dictated by the original target surface
5. Look for inspiration in the mode catalogue and extract modes of interest to blend with the approximation result

3.2.3 Non-linear morphing

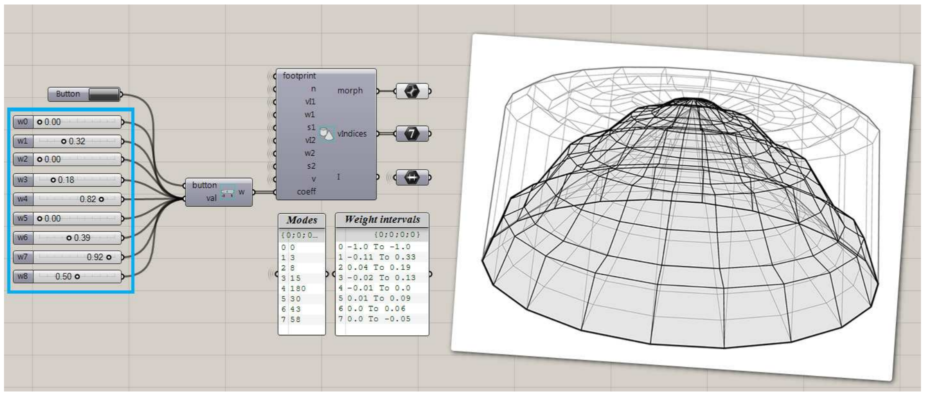
Morphing is the concept of changing one shape into another shape, which enables the exploration of the design space in between. There has to be a certain relation between the two shapes such that each point on one surface always can be mapped to a point on the other surface. For a mesh representation this means that the topology has to be consistent. The straight lines between corresponding vertices in each mesh form the displacement paths and a percentage parameter specifies the distance amount along each path from which a new "in-between-shape" emerges. This is referred to as linear morphing because the paths are traversed by linear interpolation.

The harmonic modelling tool however, enables a non-linear morphing process by taking advantage of the additional information that is available about the mode shapes. The same mesh as footprint must be used in order to have a consistent mode catalogue. The two desirable limit shapes to morph between are modelled as NURBS surfaces and a number of significant modes and their weights are back-calculated for each shape based on the common footprint (note that the defined number of significant modes for each shape can vary). From this information a shared list of modes that characterise both shapes is generated (duplicates are removed) and a percentage parameter per mode controls the non-linear morphing process. Thus, the number of parameters depends on the number of significant modes that is chosen to approximate each NURBS surface and how many modes they share. If all parameters are set to the lower limit (0%) the emerging surface coincides with the first NURBS surface and likewise if all the parameters are set to the upper limit (100%) the emerging surface coincides with the second NURBS surface.

More technically the morphed shape is calculated as the sum of displacements related to each mode shape from Equation 3.2.3

$$\vec{\mathbf{D}} = \sum_{m \in modes} \lambda_m \vec{\mathbf{A_m}} + (1 - \lambda_m)\,\vec{\mathbf{B_m}} \tag{3.2.3}$$

Where $\vec{\mathbf{D}}$ is a n-dimensional vector specifying the vertex normal displacements of the common footprint, λ_m is a parameter between 0.0 and 1.0 that controls the amount of mode m that exists in the morphed shape in relation to the limit shapes, $\vec{A_m}$ is the mode m scaled by the back-calculated weight to approximate the first limit shape and $\vec{B_m}$ is the mode m scaled by the back-calculated weight to approximate the second limit shape. The methodology is implemented in a Grasshopper component, which requires the footprint and mode catalogue (shared properties) as well as the significant mode indexes, corresponding weights and scale factor (unique to each limit shape). The morphed shape, shared mode list and corresponding weight intervals are output to better understand the process. The result of a non-linear morphing between a cone and a cylinder is shown in Figure 3.2.17, where the percentage parameters towards either of the limit shapes are highlighted in blue.



```
w0  0.00
w1  0.32
w2  0.00
w3  0.18
w4  0.82
w5  0.00
w6  0.39
w7  0.92
w8  0.50
```

```
Modes
{0;0;0…
0 0
1 3
2 8
3 15
4 180
5 30
6 43
7 58
```

```
Weight intervals
        {0;0;0;0}
0 -1.0 To -1.0
1 -0.11 To 0.33
2 0.04 To 0.19
3 -0.02 To 0.13
4 -0.01 To 0.0
5 0.01 To 0.09
6 0.0 To 0.06
7 0.0 To -0.05
```

Figure 3.2.17 – Non-linear morphing between a cone and a cylinder based on a flat circular footprint

In an architectural context morphing is useful to explore different solutions in between two predefined limits that may result from pure aesthetics or actual design constraints.

Chapter 4

Curvature-stiffened shells

Double curvature is a key design parameter for the shape of a shell to stiffen it against buckling failure. To evaluate the response of different shell shapes it is necessary to develop a tool that is capable of calculating this buckling capacity. Since it is desirable that this measure can be used to inform the shell shape in the conceptual design stage it is essential that this tool is an integrated part of the modelling environment to enable real-time interaction. This chapter briefly describes buckling as a structural failure mode to create the foundation for developing such a tool. Different options are investigated and a barrel vault study serves to evaluate the reliability of the tool.

4.1 Buckling behaviour

Buckling is an instability problem of a structure in compression, which leads to a sudden failure before the ultimate compressive stress of the material is reached. The buckling behaviour is evident from a plot of the force versus displacement as shown in Figure 4.1.1. Initially, the displacements follow the applied load with an approximate linear relation, but at a certain point (bifurcation) a critical load is reached, beyond which an infinitesimally small load increment causes a significant change in the displacement. Mathematically, this means that two different equilibrium states exist for the same load. This new buckled configuration can either be adjacent to the original configuration such that the structure quickly finds new stability and the load capacity thereby continues to increase although with a slower rate (a) or it can be a so-called snap-through where there is no adjacent equilibrium configuration resulting in large displacements before stability is reached again (b) (Cook, 1995).

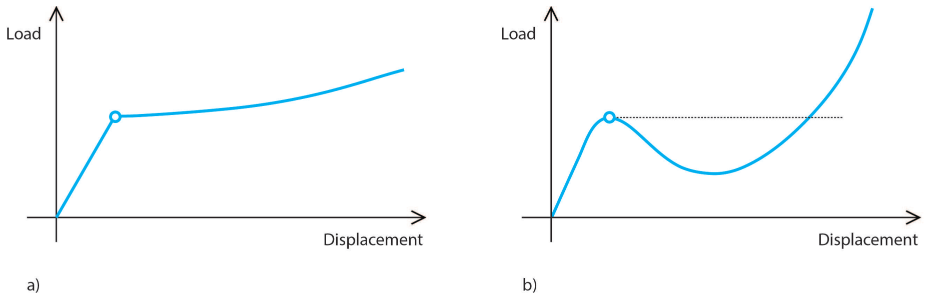


Figure 4.1.1 – Buckling behaviour from load versus displacement plot. a) Adjacent equilibrium configuration. b) "Snap-through"

The buckling capacity is usually quantified by means of a buckling load factor (BLF), which specifies how much the applied load can be scaled before buckling occurs. Generally, a buckling problem can be divided into two solution strategies; a linear and a non-linear approach. The linear approach is based on the original undeformed configuration from which the stress state resulting from the applied load is computed and used to construct a geometric stiffness matrix K_σ. The purpose of this matrix is to either increase the conventional stiffness matrix K if the structure is in tension or decrease it if the structure is in compression. The buckling problem is subsequently solved as an eigenvalue problem given by

$$[\mathbf{K} + \lambda \cdot \mathbf{K}_\sigma]\,\delta = \mathbf{0} \tag{4.1.1}$$

Where λ is the buckling load factor and δ is the displacement associated with the buckling shape (Cook, 1995). As mentioned, this approach does not take the deformed shape into account, which might alter the force distribution differently than a pure scaling of the initial stress state. As a consequence, linear buckling analysis often overestimates the capacity and provides a result on the unsafe side. However, the simplicity of the method makes it widely used.

The non-linear approach aims to account for the change in response due to the large deformations. This complicates the problem because the solution has to incorporate information about the actual configuration, which is not fully known until the solution is known. The procedure is therefore to obtain the solution through multiple linear load steps and trace the displacements. Different numerical techniques exists to make sure that this traced curve stays as close as possible to the correct (in reality unknown) solution curve, the simplest one being Euler's method (Cook, 1995). Other more advanced methods include bisection (Weisstein, 2015a), Newton-Raphson (Cook, 1995) and dynamic relaxation (Day, 1965). The bifurcation point is characterised by the gradient of the curve being equal to zero, which means that the structure has zero stiffness for this mode shape. For snap-through problems it is often more stable to incrementally prescribe the displacements and trace the reaction forces instead in order to obtain the entire post-buckling behaviour as shown in Figure 4.1.1 (b). Otherwise it is not possible to trace the part of the curve below the dotted line.

4.2 Buckling measure

Different options to obtain a measure for the buckling capacity were investigated. Integration with the developed harmonic modelling tool and low computational time (possibly at the cost of less accuracy) were the main priorities.

The initial idea was to use the first eigenvalue (different from zero), calculated from the already implemented framework with the Laplacian matrix, as a measure to compare the buckling capacity for different shapes due to the similarities between this approach and the linear buckling analysis. Even though the cotangent Laplacian takes the geometry into account, it was observed that the first eigenvalue remained almost constant for different shapes and hence unsuitable as a measure.

Karamba (Karamba3D, 2015) is a finite element analysis software embedded in Grasshopper and with the most recent version 1.1.0 (released 14 March 2015) it is possible to perform a linear buckling analysis. While this is a fast way to calculate the buckling load factor, it failed to work when integrated in an optimisation process with more than 5 variables (the weights to control the shape) as the program consistently crashed. For this reason it was decided not to use Karamba.

The option of manually constructing the conventional and geometric stiffness matrix from finite element formulations was also considered. This matrix formulation fits well with the already implemented framework. However, for three-dimensional shell elements this formulation becomes rather complicated. Therefore it was concluded that there was no point in reinventing the wheel and it was furthermore unknown what could be gained from it in terms of computational speed.

Piker (2015) recently released a completely rewritten version of Kangaroo, which is a plug-in for Grasshopper useful for simulating physics. With this new version it is possible to employ the dynamic relaxation method to solve a set of non-linear equations with improved stability and remarkable convergence speed. Based on these promising improvements, it was chosen to investigate this option further.

4.2.1 Non-linear buckling with Kangaroo

The Kangaroo engine uses Newton's second law, which relates the residual force acting on an object with its mass and acceleration ($F_{res} = m{\cdot}a$). Different physical behaviours are translated into so-called "Goals", which specify the directions and magnitudes of forces acting on the predefined geometry e.g. a mesh. These forces are summed for each vertex and the residual force dictates where this vertex will move to and how fast it will get there. The new positions are subsequently obtained in an iterative process and if the goals are conflicting, the solution becomes a compromise between them. This framework enables a dynamic relaxation process (as described in Chapter 1) by converting the edges of the mesh into springs with a rest length and stiffness, assigning masses to the vertices, converting selected vertices into anchor points and applying a load. The engine calculates the new positions that represent the equilibrium state where the residual force at each vertex is zero.

As the edges in the original mesh are converted into springs, only axial forces can be transferred, which mimics a truss structure rather than a plate-shell structure. As a consequence, if the faces in the mesh are n-gons with $n > 3$ they loose their in-plane stiffness and hence the ability to transfer shear forces. A simple way to account for that is to brace the structure with diagonal members to transfer this shear force as shown in Figure 4.2.1. For a mesh this can be formulated into a triangulation requirement.

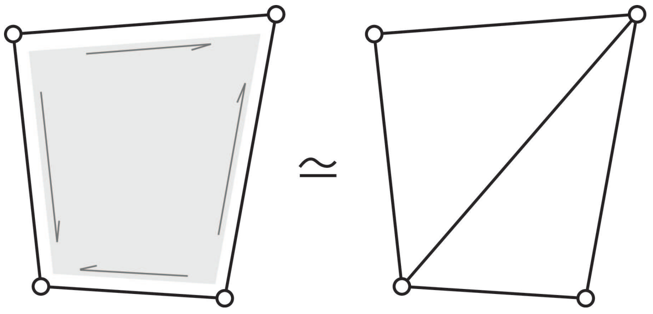


Figure 4.2.1 – Structural systems for shell and truss equivalence to transfer the shear force

Concept

The general idea behind the buckling simulation is to define the anchor points and edge springs as fixed goals referring to the undeformed mesh, initiate a load increment loop and in each step scale and update the applied load goal and calculate the equilibrium state. That way the response of the deformed structure is taken into account and evaluated against a defined criteria in order to detect when buckling occurs. To better mimic shell behaviour it is possible to add a hinge goal (again referring to the undeformed mesh), which simulates rigid connections between two adjacent triangular faces to transfer bending moments. The flow diagram for the implementation, which takes advantage of the new scripting opportunities with Kangaroo is shown in Figure 4.2.2.

Force goals

Each edge in the original mesh is converted into a spring goal by specifying the start and end vertex it acts between, its rest length equal to its current length and a stiffness value. The latter is divided by the rest length to mimic material behaviour where the stress is proportional to the strain. Selected vertices are converted into anchor goals by assigning a very high stiffness value of a zero-length spring to each of them, which connects the vertex with a target particle of infinite mass. This corresponds to a pinned support (all translational d.o.f.'s fixed) in a finite element analysis. To simulate bending behaviour, each edge with two adjacent triangular faces is converted into a hinge goal, which works by applying out-of-plane forces to the four vertices in an attempt to maintain the angle between the triangular faces in the undeformed mesh. The angle is calculated from the two face normals and the hinge strength is defined as 1/10 of the average spring strength (stiffness divided by rest length) to ensure that

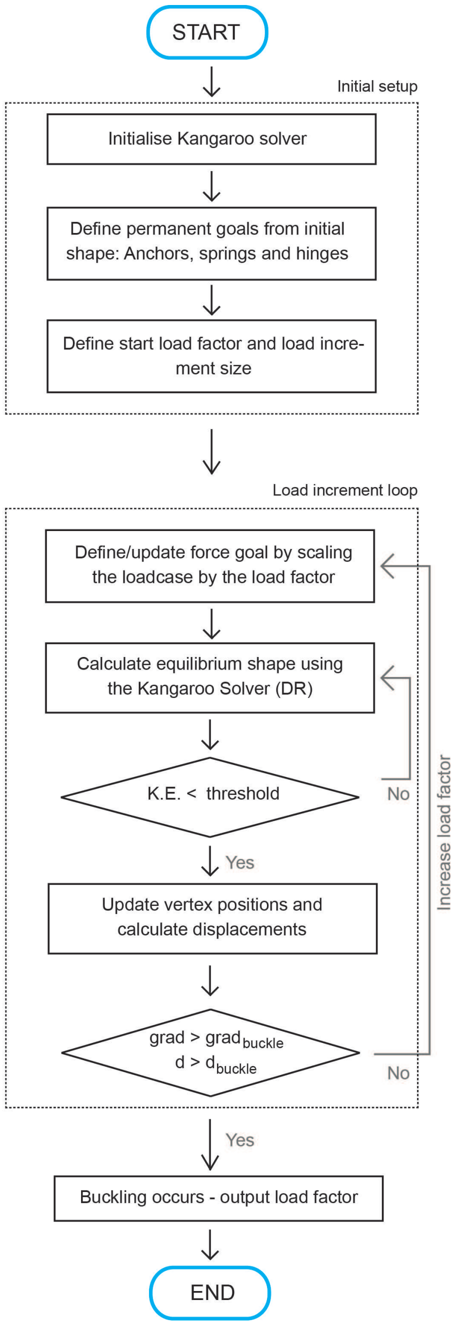

START
Initial setup
Initialise Kangaroo solver
Define permanent goals from initial shape: Anchors, springs and hinges
Define start load factor and load increment size
Load increment loop
Define/update force goal by scaling the loadcase by the load factor
Calculate equilibrium shape using the Kangaroo Solver (DR)
K.E. < threshold
No
Yes
Update vertex positions and calculate displacements
grad > grad$_{buckle}$
d > d$_{buckle}$
No
Increase load factor
Yes
Buckling occurs - output load factor
END

Figure 4.2.2 – Flow diagram for non-linear buckling with Kangaroo. K.E. is an abbreviation for kinetic energy and it is measured as the average squared magnitude of the velocity of the particles. The buckling criteria is a combination of the gradient (grad) of the load-displacement graph and the displacement (d) itself.

the structure mainly resists the load by axial forces. The applied load mimics the self-weight of the structure and it is calculated by a separate Grasshopper component as a lumped force on each vertex represented by a vertical vector pointing in the negative direction with a magnitude corresponding to the area of its associated voronoi cell (see Chapter 4). This load case is only converted to force goals during the next phase.

Load increments

A loop is initialised where the original load case is multiplied by a load factor and converted to force goals. The load factor incrementally increases in each iteration based on a start value and a step size. The process is visualised in Figure 4.2.3.

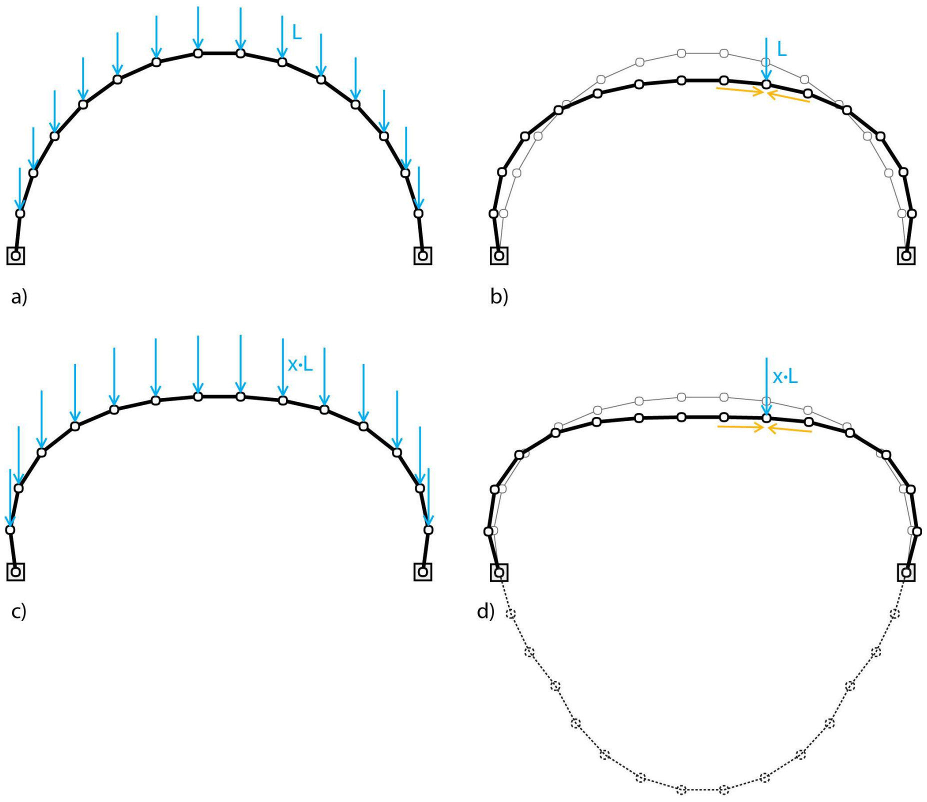


Figure 4.2.3 – Non-linear buckling procedure with Kangaroo. a) Initial shape. b) Equilibrium shape 1. c) Equilibrium shape 1 with increased load. d) Equilibrium shape 2.

The anchor, spring and optionally hinge force goals are all defined from the undeformed structure with rest lengths and rest angles inherent from this configuration (Figure 4.2.3 a). Thus, the structure only starts to move when the first load step is applied and settles when the spring and hinge forces equilibrate the load (Figure 4.2.3 b). The displacements between the vertices in the initial configuration and the new configuration are calculated and evaluated against a defined buckling criteria. The loop is repeated if buckling does not occur i.e. the load case is scaled, the force goals are updated (Figure 4.2.3 c), and equilibrium is calculated from the deformed structure determined from the previous equilibrium step (Figure 4.2.3 d).

Buckling criteria

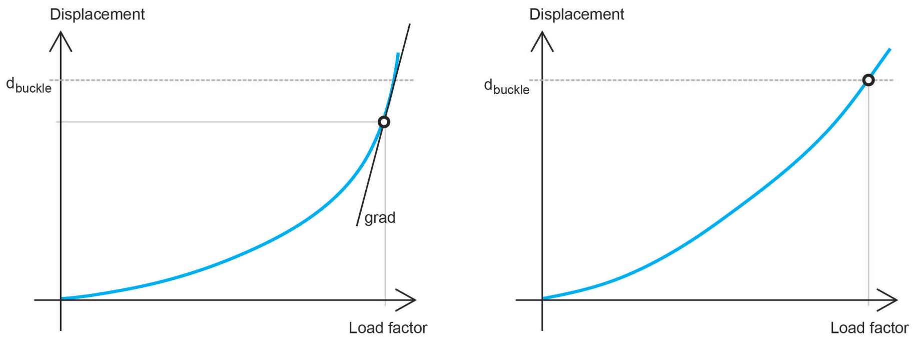


Figure 4.2.4 – Combined gradient (left) and displacement (right) buckling criteria

The buckling criteria aims to detect a sudden change in vertex displacements during the load increments. The gradient of the load-displacement curve is a useful measure in this regard. The values from the previous and current load step are used to calculate this rate of change. When buckling occurs, the gradient increases significantly as shown in Figure 4.2.4 (left). Due to the reversed axes, the behaviour is opposite to Figure 4.1.1 where the gradient vanishes at the bifurcation point and as a result it is only possible to trace the curve above the dotted line. The vertex displacements can be quantified in several different ways for example as individuals, a RMS value or a maximum value. The first method ensures that local buckling of an element is detected while the other methods act more globally. The maximum displacement is used for the implementation as part of this thesis. The gradient is evaluated against a buckling criteria, which in this case is predefined as the gradient corresponding to the maximum displacement in the current equilibrium configuration being 0.5 times larger than in the previous configuration (the maximum displacement is calculated according to the original mesh). In addition to the gradient criteria, a maximum displacement criteria is introduced (default value is 1m), which is useful if the structure exhibits ductile behaviour as shown in Figure 4.2.4 (right). However, it is possible to adjust both the gradient and displacement buckling criteria for the specific application.

Accuracy

The accuracy of the buckling analysis based on this methodology is mainly influenced by three factors; the start load factor (which the load initially is scaled by), the step size and the threshold specified for equilibrium to be reached. The accuracy improves if all of these values are decreased. However, this comes with the cost of increased computational time. The influence of the threshold is evaluated by calculating the ratio between the sum of the applied load and the reaction forces. As the threshold is lowered this ratio gets closer to 1.0. Prioritising computational speed over accuracy for the conceptual design stage, no other accuracy improving algorithms such as bisection were implemented. Additionally, the strength ratio between the membrane and bending action is of importance and was set to a default value of 1/10 based on small tests. It is essential to be able to model bending, as many structures would otherwise collapse under their own weight. However, if the load is primarily resisted by bending then the structural behaviour becomes more ductile and no sudden change in displacements can be detected. In a finite element analysis, the thickness of the shell has a similar effect (influencing the moment of inertia and thus the ability to transfer forces via bending).

A buckling analysis of a half sphere shell structure (span of 20 m) performed with the developed tool is shown in Figure 4.2.5. It shows promising results in terms of computational speed (only 2.1 seconds to calculate this example) and the shape just before buckling occurs is consistent with structural intuition.

To the knowledge of the author, no such buckling analysis tool exists in a parametric modelling environment like Grasshopper and with the observed computational speed.

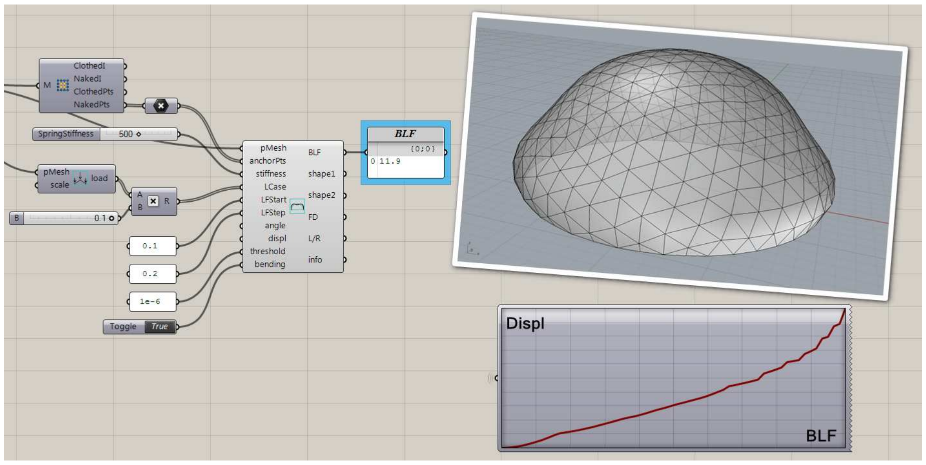
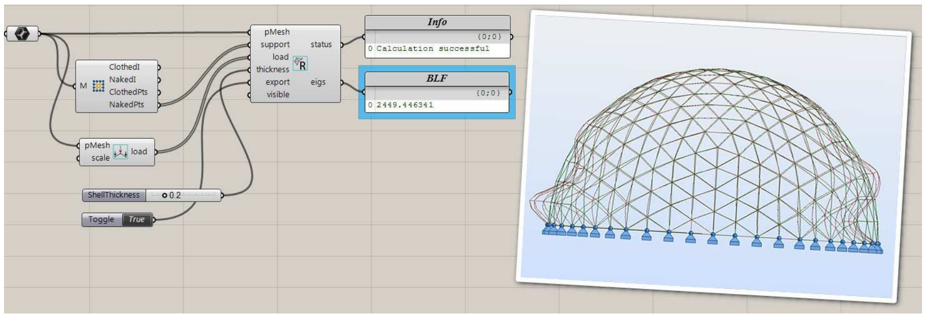


Figure 4.2.5 – Buckling analysis of a half sphere using Kangaroo

4.2.2 Linear buckling with Autodesk Robot

A work flow with Autodesk Robot Structural Analysis is established to validate the results obtained from the above described buckling analysis tool. It is similarly integrated as a parametric tool in Grasshopper and utilises the Robot API to export the geometry to this platform, perform a linear buckling analysis and retrieve the first buckling load factor. The linear buckling analysis is employed due to its simplicity. Since Autodesk Robot is a widely used finite element software in the industry it is evaluated as a trustworthy source for validation. The work flow enables a fast way to generate the complex shell geometries in the finite element software and repeated manual work with the risk of human errors is avoided by specifying all the necessary properties such as support conditions, material properties, shell thickness, load and analysis type as generic values.

The work flow integrating Autodesk Robot in Grasshopper and the results retrieved from a linear buckling analysis of the same half sphere geometry is shown in Figure 4.2.6. The solution is calculated within 42.5 seconds, which is a significant performance overhead compared with the developed Kangaroo buckling analysis tool.



```
{0;0}
0 Calculation successful
```

```
{0;0}
0 2449.446341
```

Figure 4.2.6 – Linear buckling analysis of a half sphere with Autodesk Robot integrated in Grasshopper. The first modal shape is shown

4.3 Barrel vault study

A barrel vault structure represented by a mesh with 567 vertices and 1040 faces is illustrated in Figure 4.3.1 and used as test case to evaluate the reliability of the developed buckling analysis tool. The evaluation is based on a comparison between the results obtained from the Kangaroo buckling analysis and Autodesk Robot. A number of different harmonic shapes with areas restricted to a 0.5%, 1% and 3% increase from the barrel vault reference are used as input for the analysis. The area restriction is useful to compare the shells like for like as it is otherwise expected that a shape with a larger area has more material and therefore more stiffness to withstand the applied load compared to a shape with a smaller area. However, given the same area it is possible to draw conclusions about efficient geometries to stiffen the structure and hence study the best locations to add curvature.

The harmonic shapes used for this study follow the same corrugation principles as described by Malek (2012) and are thus divided into the same three categories; edge-, crown and in-phase corrugations. Each corrugated shape is generated by an approximation of a parametrically constructed surface with an integrated cosine function to define the corrugations. The approximation is based on the reference cylinder and uses the 10 most significant weights (cotangent Laplacian with voronoi area weighting). This process is necessary because a single mode shape that only corrugates e.g. the edges and leaves the crown flat does not exist and also better mimics the intended optimisation process (shown in the case study), where a number of modes are combined with different weights to form a curvature-stiffened shell. The different shapes are shown in Figure 4.3.2

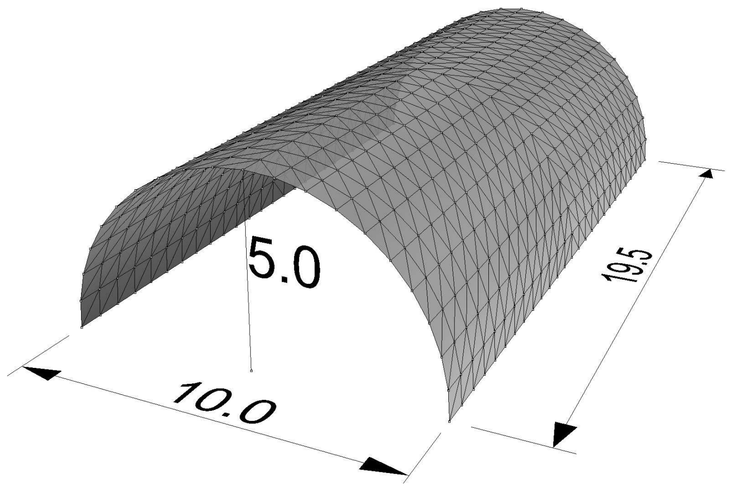


Figure 4.3.1 – The barrel vault reference geometry [m]

where the colours represent the area category they belong to (orange=0.5%, blue=1% and grey=3%). Pinned supports at the ground-level vertices and a load mimicking self-weight are applied to each structure.

For each shape the buckling load factor is calculated using the developed Kangaroo approach and Autodesk Robot. For the Kangaroo approach it is important to include bending action as the arch cross-section otherwise collapses under its own weight. The computed buckling load factor is normalised according to the performance of the barrel vault reference such that a value larger than 1.0 indicates a better shaped shell to avoid buckling failure. Specific to the non-linear Kangaroo approach is a start load factor of 0.5, a load step size of 0.05, a stiffness of 500 and a threshold of $1 \cdot 10^{-6}$. The results from the two different analysis methodologies are shown in Figure 4.3.3 and 4.3.4.

General observations

The order of the curves (orange, blue, grey) is consistent throughout the study, which confirms that a larger surface area increases the stiffness of the shell. Additionally, all the shapes have a buckling load factor larger than 1.0 indicating that the corrugations have a positive effect on stiffening the shell. The three plots for each corrugation location approximately exhibit the same tendency where the spacing in-between the curves reflects the difference in area increase.

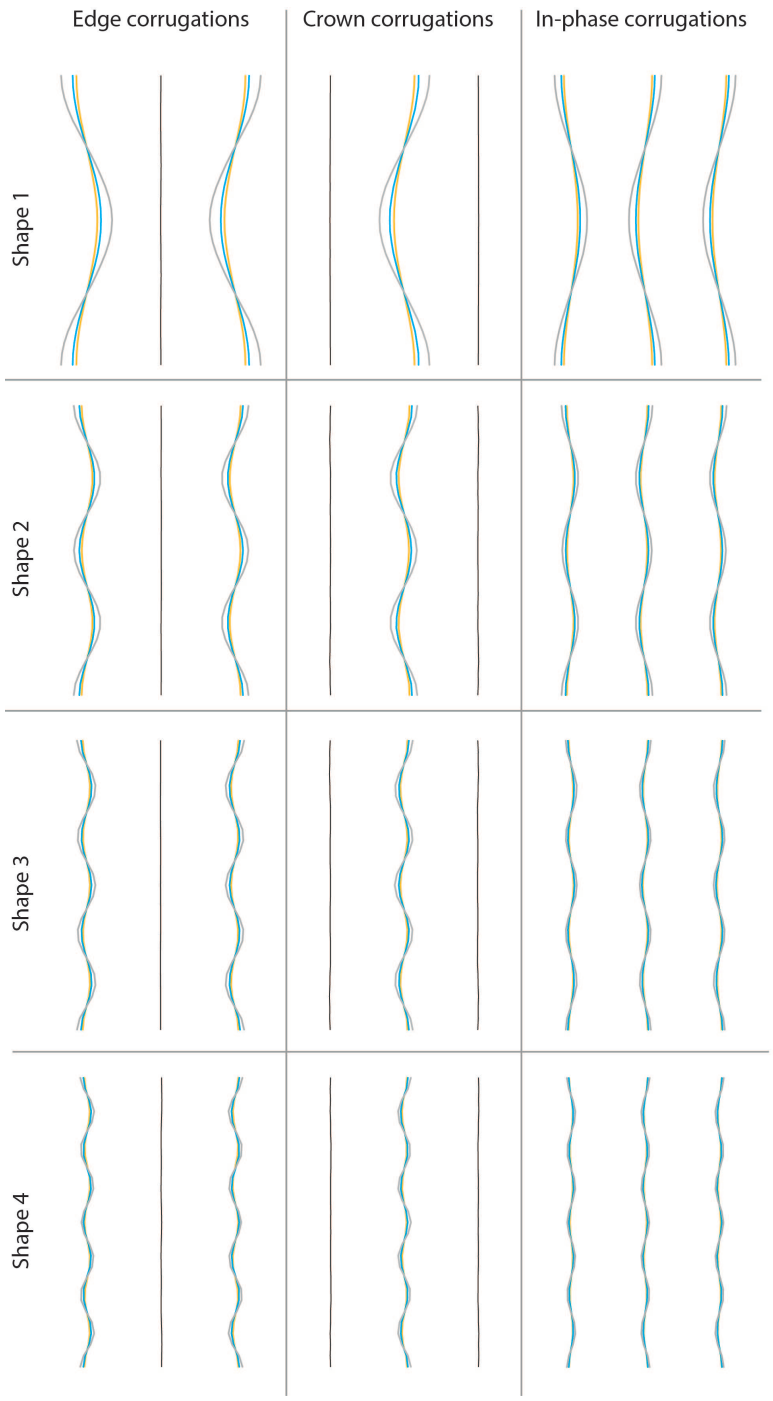
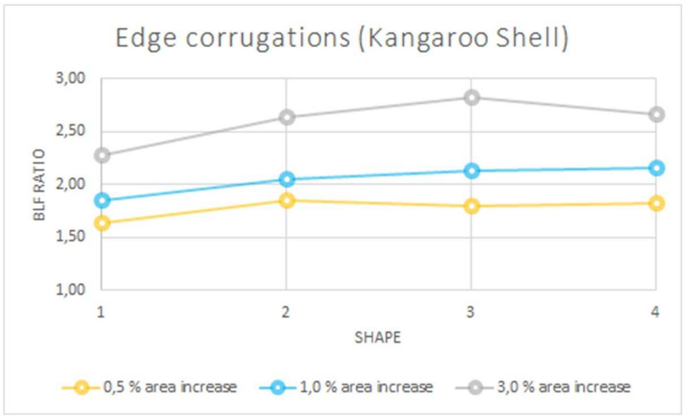
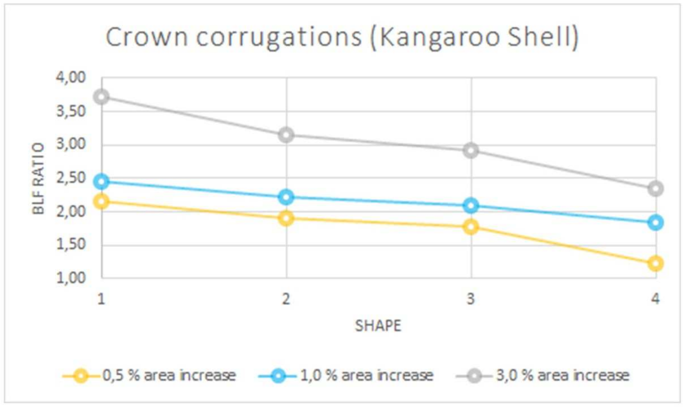


Figure 4.3.2 – Shape variations from the barrel vault reference geometry with area increase restrictions (orange=0.5%, blue=1% and grey=3%)





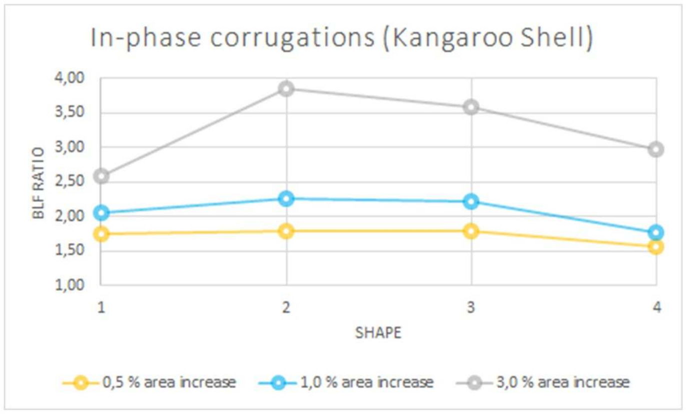


Figure 4.3.3 – The buckling load factor for edge-, crown- and in-phase corrugations of a barrel vault using Kangaroo

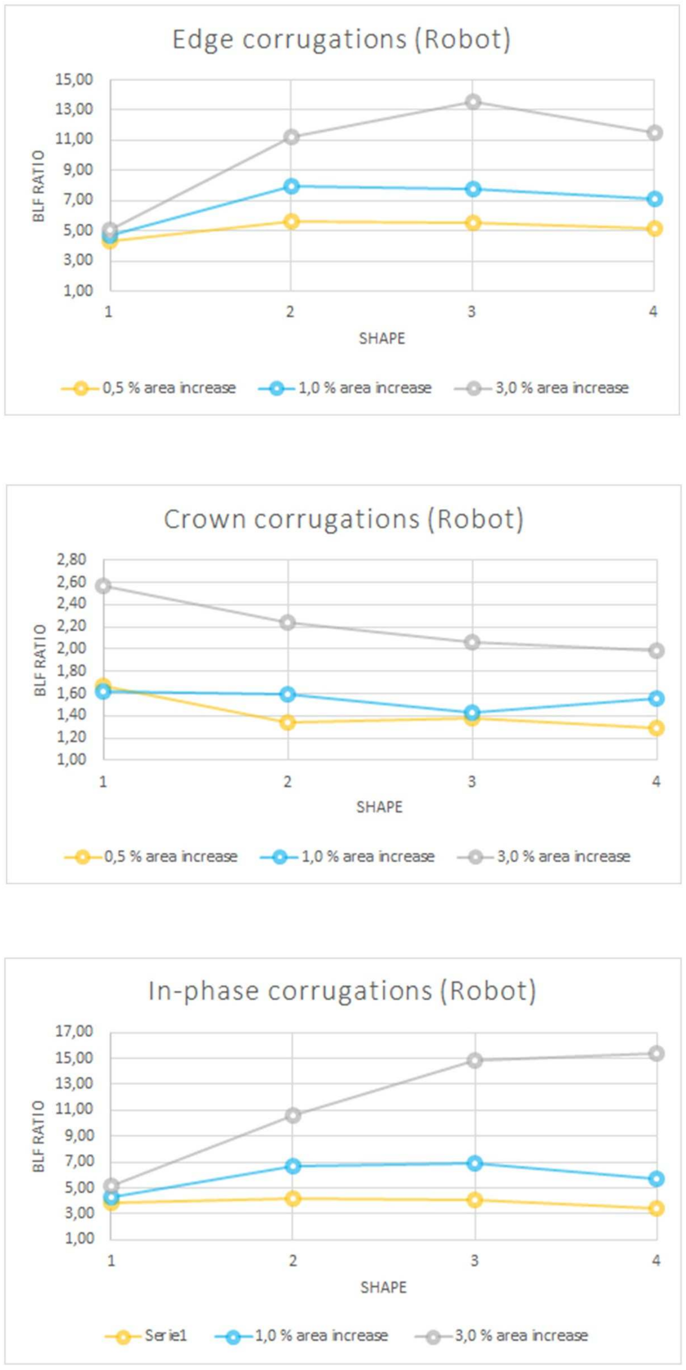


Figure 4.3.4 – The buckling load factor for edge-, crown- and in-phase corrugations of a barrel vault using Autodesk Robot

Edge corrugations

Both analyses show that an increasing number of edge corrugations initially has a positive effect on the BLF but seems to have a peak value for shape 3 after which the BLF decreases. A further investigation of the buckling modes from the Robot analysis shows that the first buckling mode changes from swaying in the longitudinal direction to the transverse direction between shape 2 and 3, which explains the increasing value for shape 3. However, the decreasing value for shape 4 indicates that there is a compromise between more waves and smaller amplitude or fewer waves and larger amplitude in order to satisfy the area requirement. For shape 4 the amplitude is so small that the edges almost become flat and the shell therefore gains less stiffness from its curvature.

Crown corrugations

Increasing the number of crown corrugations has a negative effect on the BLF according to both analyses. The tendency is less clear for the 0.5% and 1% area increase from the Robot analysis, but in general no additional capacity is gained from shape 1 to 4. As a corrugation of the crown has a similar effect as adding a stiffened longitudinal beam to the barrel vault (Malek, 2012), this behaviour can be explained by the decreasing moment of inertia due to the smaller amplitude as the number of waves in the crown is increased.

In-phase corrugations

For a 0.5% and 1% area increase the in-phase corrugations have a positive effect on the BLF with peak values for shape 2 and 3. For a 3% area increase both analyses show significant gains in BLF but the behaviour between them differs. While the Kangaroo analysis amplify the behaviour from the 0.5% and 1% cases, the results from the Robot analysis continuously grow although with a much slower rate between shape 3 and 4. Corrugating the edge and the crown simultaneously helps to increase the bending stiffness of the entire shell due to cross section variations (Malek, 2012). The decreasing BLF for shape 4 therefore implies that the amplitude becomes too small to provide enough cross-section variation thereby reducing the stiffness in the transverse direction. Shape 1 suffers from too few edge corrugations to make the shell sufficiently stiff in the longitudinal direction, which results in large deformations of the free edges. Shape 2 and 3 are therefore the best compromise. A further investigation of the different tendency for the Robot analysis of the 3% case shows that the transverse stiffness is reduced at a later stage due to the increased area, which means that the best compromise rather exists between shape 3 and 4. An additional test case with more in-phase corrugations verifies that the BLF starts to decrease after this point. In general, the BLF level for both analyses shows that the in-phase corrugations are more efficient than the edge- and crown corrugations separately.

Evaluation

This barrel vault study shows promising results in terms of observing similar behaviour between the Kangaroo and Robot buckling analysis, which helps to increase confidence in the implementation. The linear versus non-linear analysis type, combined with the effect of the strength ratio defined for the Kangaroo analysis and its relation to the chosen 200 mm thickness in the finite element analysis, are the most likely explanations for the observed deviations. As the Kangaroo buckling analysis takes less than 2 seconds to perform and the average computational time for the Robot buckling analysis is 1.1 minutes, the similarity in results makes the Kangaroo analysis tool attractive for the conceptual design stage. Malek (2012) experiences an increase in buckling capacity for an increasing number of waves at each corrugation location and observes that the in-phase corrugations are the most effective means of stiffening the barrel vault. However, several aspects make it difficult to compare the results from this study with those conclusions, including the boundary conditions, a different cross-section, the span-to-height ratio and wave amplitudes. If the same structural behaviour is expected, it is crucial that the boundary conditions match. The boundary conditions in this study are limited to pinned supports due to the stage of development at the time of writing, and for comparability reasons prioritised to be similar in the Robot model. The boundary conditions in the study by Malek are different but questionable, as they include vertical supports at the ends of the barrel vault. Equally important is the fact that the amplitude of the corrugations in Malek's study are defined as a ratio of the height or the length rather than controlled by the area increase and her conclusions in this regard are therefore not surprising. An area weighting is introduced at a later stage but with the purpose of determining the most efficient location to corrugate the vault. In general this discussion highlights the difficulties of comparing the buckling capacity between different shapes.

Chapter 5

Case study

This chapter demonstrates the applicability of the developed software tool in an architectural and engineering context using the British Museum Great Court Roof as a case study.

5.1 The British Museum Great Court Roof

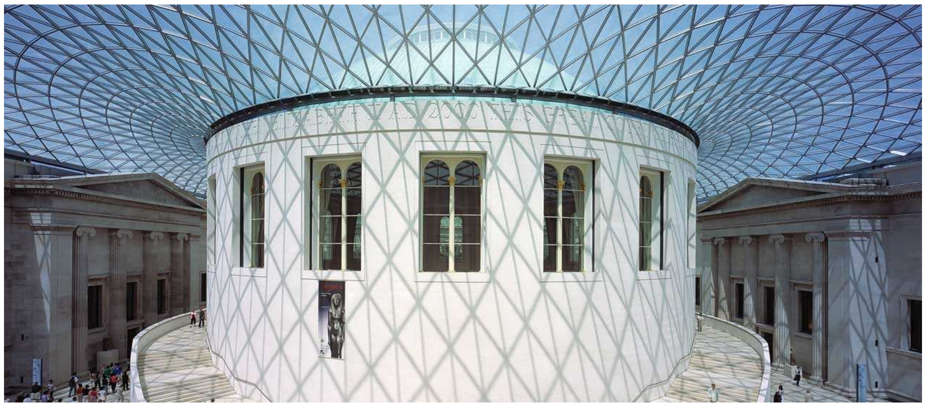


Figure 5.1.1 – The British Museum Great Court Roof

The British Museum Great Court Roof in London as seen in Figure 5.1.1 was completed in 2000 and designed by Foster + Partners in collaboration with Buro Happold and Chris Williams. Since then it has attracted much attention due to its architectural quality and unique geometric definition and the project has in general encouraged the design of gridshell structures even though it was not the first of its kind.

The roof surface is defined as a sum of three mathematical functions in order to fulfil specific site constraints and only transfer horizontal thrust to the corners of the building that supports it (Williams and Shepherd, 2010). The three functions are illustrated in Figure 5.1.2.

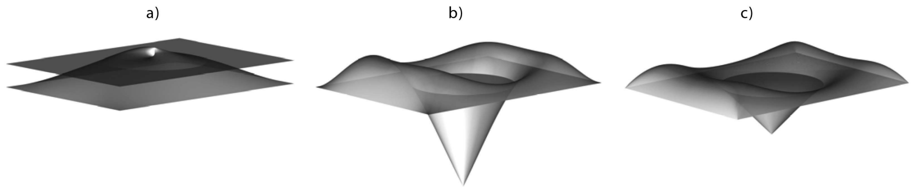


Figure 5.1.2 – The mathematical functions defining the British Museum roof surface. a) level change function, b) function without curvature singularity and c) function with curvature singularity (Williams and Shepherd, 2010)

The aim of this case study is threefold: Firstly to decompose the surface into its harmonic components to gain a better understanding of what it is built up from. Secondly to investigate the trade-off between the approximation accuracy and the number of included modes, whilst simultaneously evaluating the effect of each mode with regard to the buckling capacity. And thirdly to explore different variations of the shape by a modification of the modal components based on pure aesthetics and a buckling optimisation.

5.1.1 Shape analysis

A DXF file of the original geometry of the British Museum roof structure is used as basis for this study. A point cloud is created from the intersections between all the curves and approximated by a NURBS surface patch as shown in Figure 5.1.3.

The harmonic modelling framework requires a mesh as input. A mesh similar to the one used for the grid of the British Museum before it was relaxed over the mathematically defined surface is downloaded from GeometryGym (Mirtschin, 2009). It consists of 1806 vertices and 3372 faces and is symmetric about the x-axis (see Figure 5.1.4). The vertices of the mesh are projected vertically onto a ruled surface spanning between the boundary curves (located in two different levels) to be able to fix the mesh at these locations. Fixing the boundaries helps to ensure a good approximation with a limited number of variables.

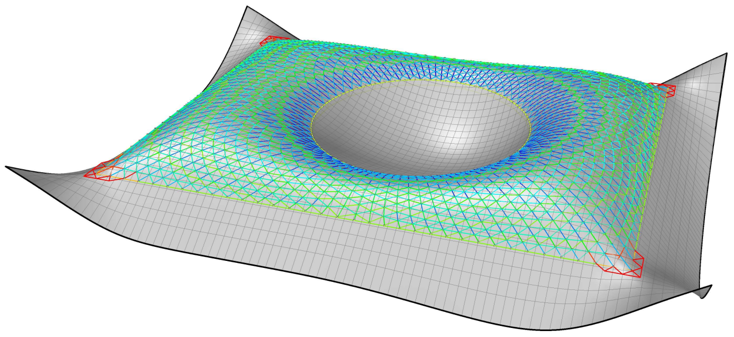


Figure 5.1.3 – Surface patch approximation of the British Museum roof structure

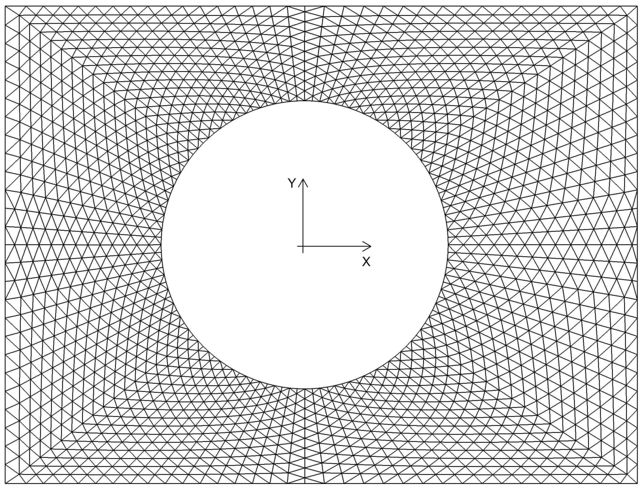


Figure 5.1.4 – The mesh used for the harmonic modelling framework

From a construction of the graph Laplacian followed by an eigendecomposition, the 10 most significant modes to approximate the NURBS surface patch are back-calculated, as visualised in Figure 5.1.5. The approximation result is smooth and has a RMS value of 0.14 m, which is acceptable when compared to the size of the structure (72 x 96 m) and the fact that only 10 variables are used. The modal components are both fascinating and revealing at the same time. Especially the first 3 modes, which have very recognisable shape characteristics.

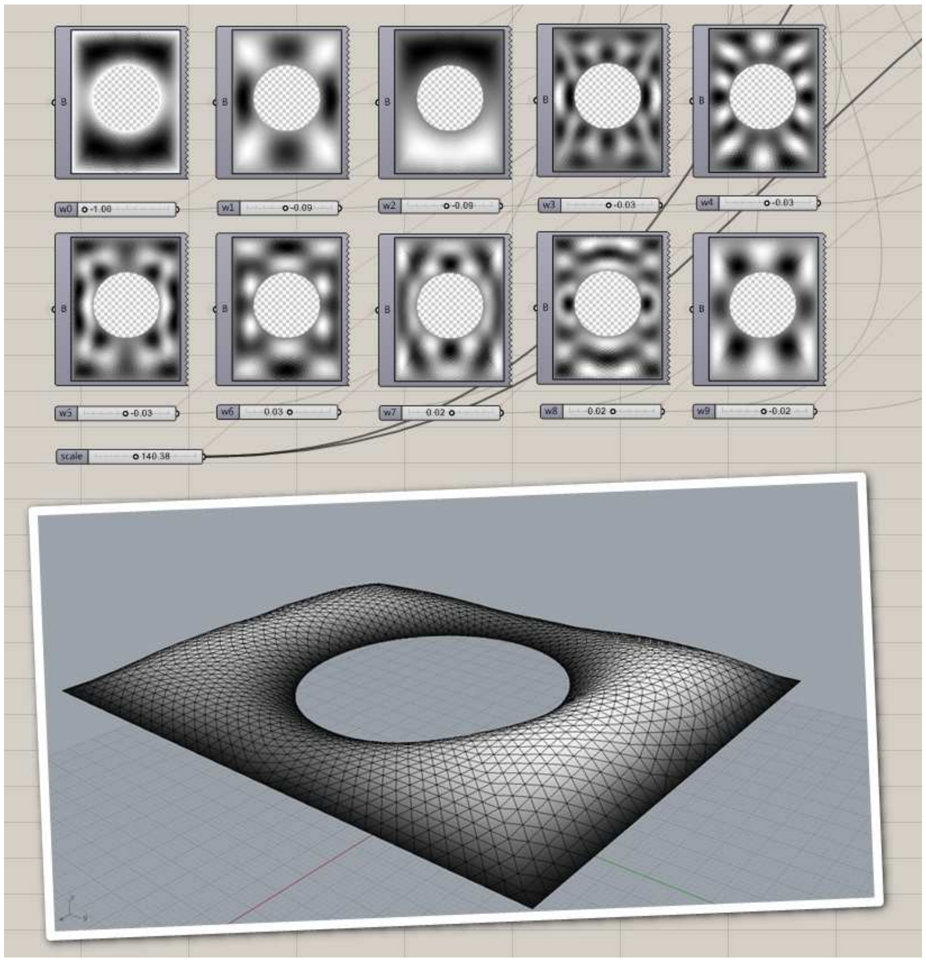

w0 -1.00
w1 -0.09
w2 -0.09
w3 -0.03
w4 -0.03
w5 -0.03
w6 0.03
w7 0.02
w8 0.02
w9 -0.02
scale 140.38

Figure 5.1.5 – The 10 most significant harmonic modes of the British Museum and the approximation result

5.1.2 Approximation accuracy and buckling capacity

The plot in Figure 5.1.6 serves to gain a better understanding of the trade-off between the number of variables (mode shapes with weights) and the approximation accuracy (quantified by means of a RMS value) and the influence of the choice of Laplacian. For less than 25 variables the graph Laplacian gives the best accuracy after which the cotangent Laplacian without area weighting becomes more efficient. The area weighted cotangent Laplacian consistently results in lower accuracy for any number of variables. This behaviour is due to the very regular distribution of the vertices in the mesh. The plot also shows how the RMS value rapidly decreases towards zero for the first 20 modes and afterwards has a much slower rate. This is promising as it means that a good approximation of the British Museum Great Court Roof can be achieved with less than 20 variables for a mesh with 1806 vertices.

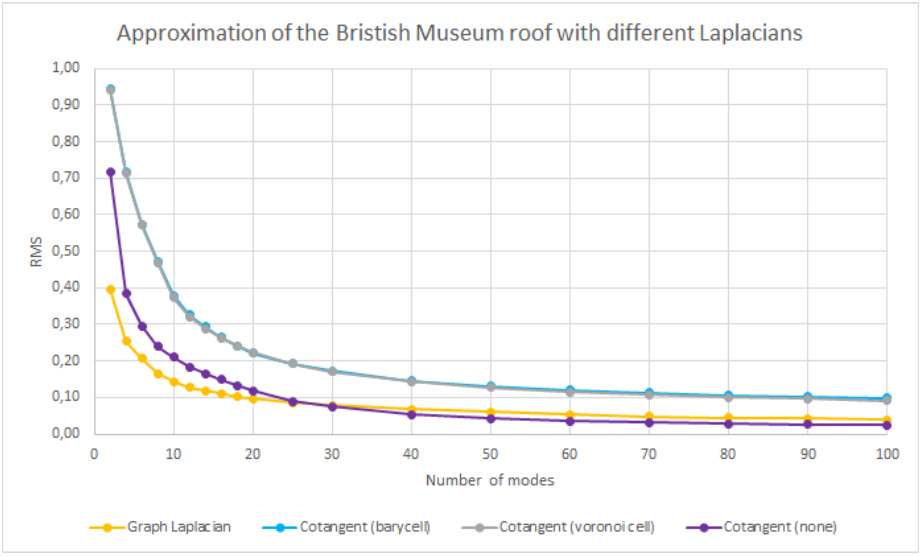


Figure 5.1.6 – Approximation accuracy of the British Museum with different discretisations of the Laplacian

It is often the case that the structural performance of a shell can be noticeably affected by only small changes to the geometry. The buckling capacity is one such measure that is very sensitive to its shell form and it is therefore interesting to study how the addition of extra modes influences the performance. Some modes may only be of aesthetic character while others improve the buckling capacity. A plot of the buckling load factor as a function of the number of modes included to approximate the target shape therefore helps to guide the final design towards a good compromise.

For this analysis the boundary conditions are simplified to pinned supports along the two boundary curves instead of the sliding supports, which were used for the real project to only allow horizontal thrust to be transferred at the corners. This is due to the limitations of the software implementation related to buckling at the time of writing. It is therefore important to emphasise that the conclusions and results outlined in this case study do not take the structural constraints into account, which were critical for the original shell. However, it still demonstrates the capabilities of the developed design tool.

The buckling load factor for each shape is calculated with both the non-linear Kangaroo approach (including bending action) and a linear buckling analysis in Autodesk Robot. As expected, the failure is observed to be a collapse of the dome-like part with the longest span. The computational time for the Kangaroo component is on average 10 seconds while the Robot component uses at least 3 minutes to calculate the solution. The Robot analysis is therefore only feasible because 10 shapes are studied but is not suitable for an optimisation work flow. The result is shown in Figure 5.1.7.

A similarity between the results is noticeable up until the addition of 7 modes. After that the BLF significantly decreases according to the Robot analysis while the Kangaroo analysis predicts a continuous increase. The different calculation strategies (linear versus non-linear) and the arbitrarily defined strength ratio between membrane and bending action of the shell for the Kangaroo simulation are concluded to be the most significant cause for the deviation. Due to the latter, two different shell thickness are examined (t=200 mm and t=400 mm) for the Robot analysis. It is evident how this alters the force distribution and therefore affects the BLF results even within the same analysis software. This highlights the difficulties of obtaining a buckling measure from a generic set-up.

From Figure 5.1.7 it is concluded that adding the six most significant modes has a positive influence on the buckling capacity while at the same time approximating the target better. The different analyses agree that mode three (see Figure 5.1.5) has a negative effect on the structural performance and therefore is of pure aesthetic value. The design tool makes it possible to remove this mode and study the shape consisting of mode 1+2+4+5+6 instead. The buckling load factor increases from this action (Kangaroo: 4.75, Robot (t=0.2): 58.9 and Robot (t=0.4): 145.2) compared to the values from Figure 5.1.7 (number of modes = 6). This method is therefore an effective means of improving the structural performance with little deviation from the target.

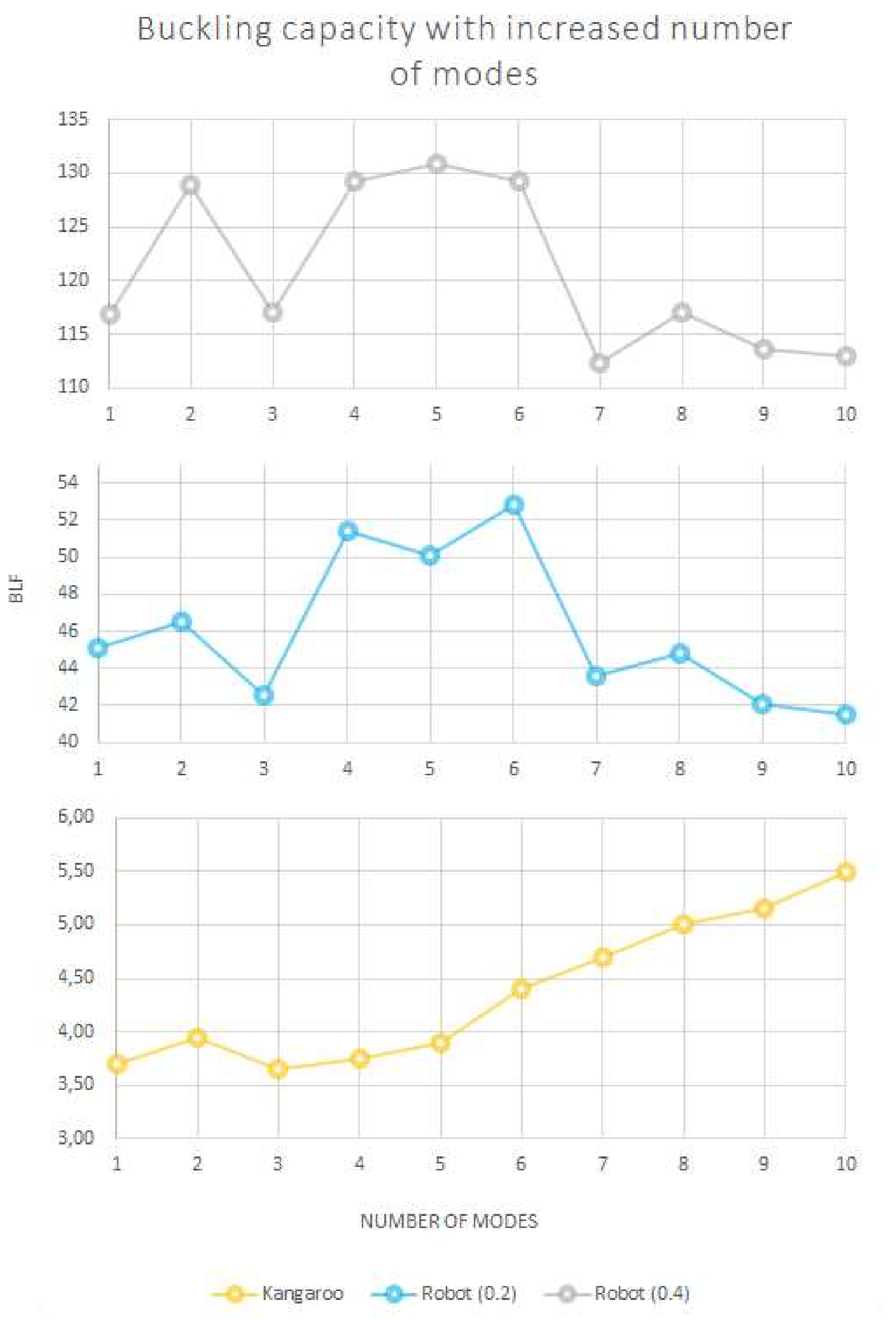


Figure 5.1.7 – The buckling load factor as a function of the number of included modes. The measurement is obtained from Kangaroo (orange), Autodesk Robot with a thickness of 200 mm (blue) and 400 mm (grey)

5.1.3 Shape variations

With the 10 short-listed modal components for the British Museum Great Court Roof it is easy to explore a variety of other design possibilities emanating from the original shape by modifying the quantities of these components and adding new interesting ones as well.

Aesthetics

A few examples of shapes created from a pure aesthetic point of view are shown in Figure 5.1.8. They highlight the wide range of designs that can be achieved from the target surface, which can be used as inspiration for the architect in the conceptual stage.

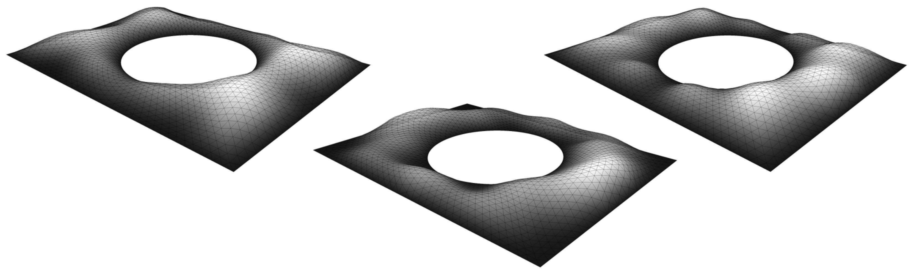


Figure 5.1.8 – Shape variations of the British Museum Great Court Roof from a modification of the harmonic ingredients and quantities from an aesthetic perspective

Optimisation

The modification of the mode shapes can also be based on an evaluation of the buckling capacity in an optimisation process. This work flow is supported by the low-parametrisation of the mesh and the computational speed of the non-linear Kangaroo buckling component. Thus, the variables become the weights of the 10 most significant modes plus the scale factor and the objective function for the optimisation is to maximise the buckling load factor. The built-in “Galapagos” component in Grasshopper with a simulated annealing search strategy is used for this purpose.

In order to compare the results like for like it is essential that the generated shapes share the same area increase from the target surface such that the best shape reflects geometric stiffness rather than stiffness from more material. As the vast majority of the generated shapes will have an area which does not correspond to the desirable increase, an algorithm is developed to avoid discarding all these potential solutions. The algorithm calculates the area of the target mesh and the area of the newly generated mesh and evaluates whether the weights need to be increased or decreased in order to reach a defined percentage (the desirable area increase). Based on this evaluation the weights are changed in small steps defined as a factor of the deviation between the weights of the target and the weights of the generated shape. After each step, the area of the mesh from the adjusted weights determines whether the desirable percentage is hit or the process shall continue. A bisection strategy is embedded to adjust the step size if the target percentage is passed. This way it is avoided to use an inefficient area filter and all potential solutions get a chance to be evaluated in terms of their buckling capacity by being rescaled. In this case, the weight corresponding to mode number 1 is limited to negative values only (referring to Figure 5.1.5) to avoid hanging structures.

Figure 5.1.9 shows the optimised results for a 1%, 3% and 5% area increase from the approximated target surface using 10 weights. For a 1% area increase there is not much freedom to significantly alter the geometry from the original roof shape but it is noted that the corners (where the biggest spans occur) are inflated while the middle parts with the shortest spans are deflated and by doing so the BLF is increased from 5.5 to 6.25. A similar tendency is observed for the 3% area increase where it furthermore becomes clear how the harmonics help to stiffen the inflated parts of the shell through curvature in effect doubling the buckling load factor! The form-found shape for the 5% area increase does not follow the same material distribution pattern, which results in a disappointing buckling load factor. This may be a consequence of the large search space and computational speed to calculate one solution (10 seconds) as the best fit is influenced by the run time. Another option is that it reflects a compromise with the self-weight of the structure but it needs further investigation. In general it is noted that the results obtained from the Kangaroo and Robot analysis agree in terms of ranking the shapes according to their buckling capacity, which make the results more reliable.

It is important to highlight that the shells shown in Figure 5.1.9 are optimised for buckling only, whereas a real structure has other structural performance requirements that needs to be fulfilled as well.

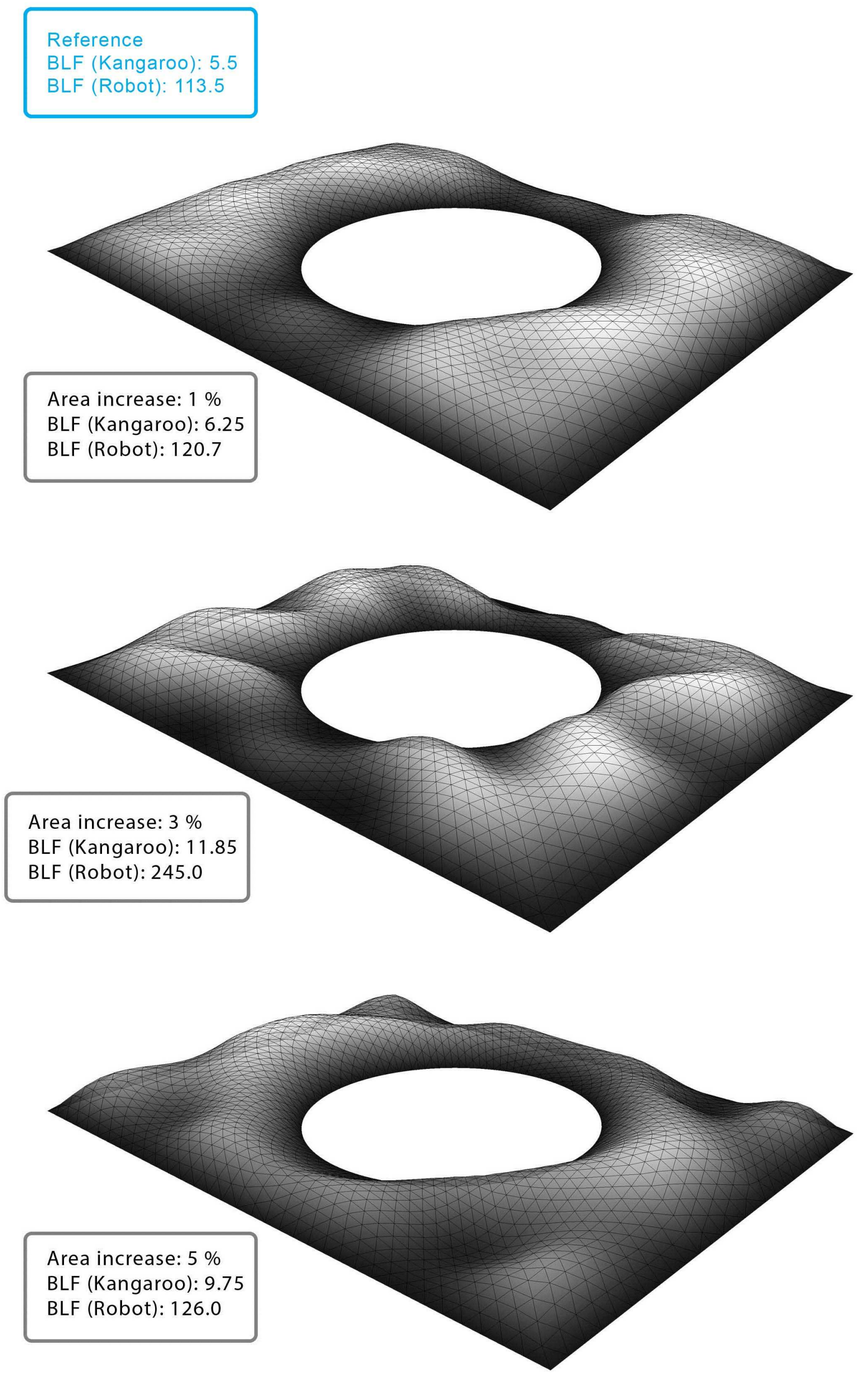


Figure 5.1.9 – The results from a buckling optimisation with different requirements to the area increase from the original roof

Chapter 6

Conclusions

6.1 Summary

The aim of this research was to develop a software tool to assist in the design of shells in the conceptual design stage to encourage the interaction between the architect and the engineer rather than either/or. This has been accomplished by an implementation of a low parametrisation modelling strategy with an inherent shape language suitable for the stiffening of shells through their curvature and quantified by a real-time buckling measure. The software tool was developed as a plug-in to Grasshopper (see Appendix C) to make it accessible for a wide range of users and integrate it with current work flows.

The modelling strategy was based on harmonics, with a direct link to Fourier analysis to achieve the low parametrisation. A literature review highlighted that the framework behind this was well-known and widely used in computer graphics but had not yet been satisfactorily adapted to an architectural context. A number of initiatives were introduced, which included a single degree of freedom design approach for simplicity, the ability to impose boundary conditions, aids for visualisation and guidelines towards specific spatial configurations. The result was a flexible free-form modelling tool that not only enabled the creation of arbitrary doubly-curved surfaces, but also allowed simultaneous shape analysis to achieve a better understanding of the spatial components, remove noise and inspire new shapes emanating from the original shape.

The inherently different shapes compared to NURBS and subdivision surfaces (given the same amount of parameters) made buckling an interesting structural performance criteria to asses the efficiency of the shape. Furthermore the literature review identified that buckling was considered the dominant failure mode for shell structures, yet a capacity check was only calculated in the late structural verification phase, whilst other more common quantities such as stress or deflection were used to inject logic into the shape design. Only one study that related shell geometry with the buckling capacity was found, however the work flow was too cumbersome for integration in the conceptual design stage and thus rather provided rules of thumb to use at this stage. Kangaroo, a force based algorithm that integrates Newton's second law of motion, was scripted for bespoke application in relation to buckling to simulate shell behaviour under an increasing load. The implementation showed very promising results in terms of computational speed and from comparisons with results obtained from Autodesk Robot it was concluded to be of sufficient accuracy to provide a quantitative buckling measure for the conceptual design stage.

The software was applied to the roof structure of the British Museum Great Court, which demonstrated how the most significant mode shapes could be used to gain a better understanding of the original geometry and help improve it by evaluating each mode in terms of its contribution to the overall approximation accuracy and buckling capacity. It also showed how different shape variations of the original geometry could be explored, either from a pure aesthetic point of view or based on an optimisation process if more design freedom was allowed. The optimised shell geometries all had their harmonic components amplified and they exhibited up to a doubling in buckling capacity for a 3% increase in roof surface area.

6.2 Discussion and future work

One of the main disadvantages of the harmonic modelling tool is the lack of tangible spatial control like the control polygon for NURBS and subdivision surfaces. The numerical parameters are more abstract and it is generally hard to predict the result of adding multiple mode shapes together. However, this is counterbalanced by the ability to analyse the shape whilst modelling. Therefore the biggest advantage of this tool is realised through a combination of modelling techniques such that, for example NURBS surfaces are used to imply the spatial design intent and subsequently the harmonic modelling tool is used to analyse this shape, make it cleaner and possibly inspire new shapes. The Specialist Modelling Group at Foster + Partners expressed positive feedback about this approach and imagined it would be useful when sculpting surfaces.

The ability to impose boundary constraints was one of the main initiatives to adapt the mathematical framework behind the harmonics into an architectural setting. Pinned supports were successfully assigned to specifically chosen vertices, but it is associated with a number of limitations. First of all, the chosen single degree of freedom design approach only made it possible to define pinned supports, which from an architectural perspective limited the control of the tangency at these locations. Furthermore, it was observed that the area weighting of the cotangent Laplacian caused non-smooth boundary transitions even though its purpose was to make the harmonic behaviour independent of the mesh. This suggests a further investigation into rotational degrees of freedom to improve these shortcomings while at the same time prioritising a simplistic design tool would be beneficial.

In general the area weighting of the cotangent Laplacian did not show its potential, as it performed worse than the simple graph Laplacian or unweighted cotangent Laplacian in most cases for target approximation purposes. Whether this was due to the implementation, or the uniform vertex distributions masked its effects, is unknown but needs further investigation. Since other linear operators can be used for the harmonic framework as well, another interesting area of study would be to investigate the Biharmonic operator, which is the second order Laplacian (Botsch et al., 2010) and observe if any behavioural differences occur. While the Laplacian operator tries to average the gradient (pushing each vertex towards the barycenter of its 1-ring neighbours to make it flat), the Biharmonic operator tries to average the curvature by taking the 2-ring neighbours into account. This behaviour is interesting because it mimics bending and the inclusion of the 2-ring neighbours might allow more control of the tangency at the boundaries.

Currently the slowest part of the software workflow is the calculation of the eigenvalues/vectors by linking to Matlab. Matlab was chosen for its reliability, option to sort the eigenvectors in ascending order according to the frequencies and capability to only extract a desired range of eigenvectors based on a reference eigenvalue. However, to make the plug-in independent of proprietary software and possibly improve computational speed it is recommended to investigate other libraries for this task. Matlab can then serve as a good platform to validate the new implementation.

In contrast to other structural performance measurements such as stress or deflection, buckling of shell structures is more difficult to quantify because the associated large deflections alter the way the applied load is resisted by the structure and results in a non-linear relation. The implemented physical simulation of this buckling behaviour using Kangaroo highlighted a sensitivity to several issues, including the load step size, equilibrium tolerance, the ratio between membrane and bending action and a generic criteria to determine when buckling occurred. The ratio between membrane and bending action was mostly based on trial and error and needs further investigation to improve the accuracy of the simulation. Ideally this would lead to some guidelines that relates the ratio to the intended thickness of the structure. This factor is crucial, because without it, a barrel vault fails under its own weight, whereas a ratio that makes bending action too dominant results in more ductile behaviour of the structure and causes the buckling load factor to be determined from a maximum deflection criteria rather than a sudden change in displacements. This research only included the self-weight of the structure, however the buckling component is such that any load case can be applied without further complications. The next step is to include sliding supports as the shell behaviour is very dependent on its boundary conditions. This is straight forward since a Kangaroo force goal (“AnchorXYZ”) that only restrains a vertex along defined global directions already exists.

On a final note, it is essential to emphasise that the harmonic modelling tool and the buckling analysis can be seen as separate parts, which means that it is possible to use subdivision surfaces for example as the low parametrisation strategy instead and let the control points be modified according to a buckling evaluation. Another option is to link the harmonic modelling tool to another structural quantity such as maximum stress level or even exclude any kind of structural logic. However, what binds the two parts together is the inherent doubly curved nature of the harmonic shapes, which has proven very useful to stiffen shells against buckling failure as the barrel vault and the British Museum Great Court Roof demonstrated. When used together in an optimisation process, care has to be exercised concerning the area in order to compare like for like and thereby obtain results that reflect an increased stiffness through geometry instead of stiffness through extra material.

Overall a harmonic form-finding tool for the design of curvature-stiffened shells has been developed and tested through multiple case studies with promising results for the application in the conceptual design stage.

Bibliography

Adriaenssens, S., M. Barnes, R. Harris, and C. Williams: 2014, *Shell structures for architecture: Dynamic relaxation*. Routledge.

Autodesk: 2015a, 'Autodesk Maya: Overview [Online]'. Available from: www.autodesk.com/products/maya/overview [Accessed 22 July 2015].

Autodesk: 2015b, 'Robot Structural Analysis Professional: Overview [Online]'. Available from: www.autodesk.com/products/robot-structural-analysis/overview [Accessed 9 June 2015].

Azad, K.: 2010, 'Intuitive Understanding Of Eulers Formula [Online]'. Available from: www.betterexplained.com/articles/intuitive-understanding-of-eulers-formula [Accessed 13 July 2015].

Azad, K.: 2012, 'An Interactive Guide To The Fourier Transform [Online]'. Available from: www.betterexplained.com/articles/an-interactive-guide-to-the-fourier-transform [Accessed 24 June 2015].

Bhooshan, S. and M. Sayed: 2012, 'Sub-division surfaces in architectural form finding and fabric forming'. *Second International Conference on Flexible Formwork* pp. 64–74.

Block, P.: 2009, 'Thrust Network Analysis - Exploring three-dimensional equilibrium'. Doctor of philosophy, Massachusetts Institute of Technology.

Botsch, M., L. Kobbelt, M. Pauly, P. Alliez, and B. Lévy: 2010, *Polygon Mesh Processing*. Natick, Massachusetts: A K Peters, Ltd.

Cook, R. D.: 1995, *Finite element modeling for stress analysis*. John Wiley & Sons, Inc.

Day, A.: 1965, 'An introduction to dynamic relaxation'. *The Engineer 219* pp. 219–221.

Desbrun, M., M. Meyer, P. Schröder, and A. H. Barr: 1999, 'Implicit fairing of irregular meshes using diffusion and curvature flow'. *Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99* **33**, 317–324.

Dong, S., P. T. Bremer, M. Garland, V. Pascucci, and J. C. Hart: 2006, 'Spectral surface quadrangulation'. *ACM Transactions on Graphics* **25**(3), 1057.

Gebreiter, D.: 2012, 'Structuring Free Form Building Envelopes'. Mphil thesis, University of Bath.

Herholz, P.: 2012, 'General discrete Laplace operators on polygonal meshes'. Diploma thesis, Humholdt Universität zu Berlin.

Hooke, R.: 1676, *A description of helioscopes, and some other instruments.* Royal Society, London.

Jacobson, A.: 2010, 'Barycentric versus voronoi regional area in mass matrices [Online]'. Available from: www.alecjacobson.com/weblog/?tag=circumcenter [Accessed 17 June 2015].

Kac, M.: 1966, 'Can One Hear the Shape of a Drum?'. *American Mathematical Monthly, Volume 73, Issue 4* pp. 1–23.

Karamba3D: 2015, 'Karamba: About [Online]'. Available from: www.karamba3d.com/about [Accessed 29 July 2015].

Lévy, B. and H. Zhang: 2009, 'Spectral Mesh Processing [Online]'. Available from: www.emu.edu/cjp/spi/courses/624 [Accessed 28 April 2015].

Malek, S. R.: 2012, 'The Effect of Geometry and Topology on the Mechanics of Grid Shells'. Doctor of philosophy in the field of structures and materials, Massachusetts Institute of Technology.

MathWorks: 2015, 'Documentation: Pass Complex Data to MATLAB from C sharp Client [Online]'. Available from: www.uk.mathworks.com/help/matlab/matlab_external/call-matlab-function-from-a-c-client [Accessed 29 May 2015].

McGuire, M.: 2000, 'The Half-Edge Data Structure [Online]'. Available from: www.flipcode.com/archives/The_Half-Edge_Data_Structure [Accessed 28 May 2015].

McNeel: 2014a, 'Rhinoceros 5 Features [Online]'. Available from: www.rhino3d.com/features [Accessed 29 May 2015].

McNeel: 2014b, 'RhinoCommon SDK [Online]'. Available from: www.4.rhino3d.com/5/rhinocommon/ [Accessed 17 June 2015].

McNeel: 2015a, 'Grasshopper - Algorithmic Modeling for Rhino [Online]'. Available from: www.grasshopper3d.com [Accessed 29 May 2015].

McNeel: 2015b, 'Rhinoceros [Online]'. Available from: www.rhino3d.com [Accessed 14 September 2015].

Meyer, M., M. Desbrun, P. Schr, and A. H. Barr: 2002, 'Discrete Differential Geometry Operators for Triangulated 2 Manifolds'. *Visualization and Mathematics III* pp. 113–134.

Michalatos, P. and S. Kaijima: 2014, *Shell structures for architecture: Eigenshells*. Routledge.

Michalatos, P. and S. Kaijima: 2015, 'Millipede [Online]'. Available from: www.sawapan.eu [Accessed 22 July 2015].

Miller, M.: 2014, 'AD Classics: Los Manantiales / Felix Candela [Online]'. Available from: www.archdaily.com/496202/ad-classics-los-manantiales-felix-candela [Accessed 27 July 2015].

Mirtschin, J.: 2009, 'Geometry Gym: British Museum Great Court Roof using StructDrawRhino [Online]'. Available from: www.geometrygym.blogspot.co.uk/2009/11/british-museum-great-court-roof-using [Accessed 08 August 2015].

Piker, D.: 2015, 'Kangaroo 2.0 - Now out of the pouch and available for testing! [Online]'. Available from: www.grasshopper3d.com/profiles/blogs/kangaroo-2-0-now-out-of-the-pouch-and-available-for-testing [Accessed 29 July 2015].

Piker, D. and W. Pearson: 2015, 'Plankton [Online]'. Available from: www.github.com/meshmash/Plankton [Accessed 17 June 2015].

Pinkall, U. and K. Polthier: 1993, 'Computing Discrete Minimal Surfaces and Their Conjugates'. *Plateau* pp. 1–28.

Pressley, A.: 2012, *Elementary Differential Geometry*, Vol. 79. Springer, second edition.

Reuter, M., S. Biasotti, D. Giorgi, G. Patane, and M. Spagnuolo: 2009, 'Discrete Laplace Beltrami operators for shape analysis and segmentation'. *Computers & Graphics* **33**, 381–390.

Rowland, T.: 2015, 'Vector Space Basis [Online]'. Available from: www.mathworld.wolfram.com/VectorSpaceBasis [Accessed 14 July 2015].

Sasaki, M.: 2014, *Shell structures for architecture: Structural design of free-curved RC shells*. Routledge.

Schneider, P. J.: 2015, 'NURB Curves: A Guide for the Uninitiated [Online]'. Available from: www.mactech.com/articles/develop/issue_25/schneider [Accessed 21 July 2015].

Shepherd, P.: 2009, 'Digital Architectonics in Practice - Aarhus Botanical Garden Hothouse Competition'. pp. 1–8.

Shepherd, P.: 2014, *Shell structures for architecture: Subdivision Surfaces*. Routledge.

Taubin, G.: 1995, 'A signal processing approach to fair surface design'. *SIGGRAPH 95 Conference Proceedings* pp. 351–358.

Vallet, B. and B. Lévy: 2008, 'Spectral geometry processing with manifold harmonics'. *Computer Graphics Forum* **27**(2), 251–260.

Wazeone: 2010, 'Eladio Dieste [Online]'. Available from: www.wazeone.wordpress.com/2010/02/15/day-32-eladio-dieste [Accessed 27 July 2015].

Weisstein, E.: 2015a, 'Bisection [Online]'. Available from: www.mathworld.wolfram.com/Bisection [Accessed 28 September 2015].

Weisstein, E. W.: 2015b, 'Discrete Fourier Transform [Online]'. Available from: www.mathworld.wolfram.com/DiscreteFourierTransform [Accessed 02 April 2015].

Wikipedia: 2015a, 'Hearing the shape of a drum [Online]'. Available from: www.en.wikipedia.org/wiki/Hearing_the_shape_of_a_drum [Accessed 12 June 2015].

Wikipedia: 2015b, 'JPEG [Online]'. Available from: www.en.wikipedia.org/wiki/JPEG [Accessed 17 June 2015].

Williams, C. J. K. and P. Shepherd: 2010, 'Definition of the geometry of the British Museum Great Court roof'. In: *The New Mathematics of Architecture*. Thames & Hudson, pp. 122–125.

WolframAlpha: 2015, 'About Wolfram Alpha [Online]'. Available from: www.wolframalpha.com/about [Accessed 17 June 2015].

Zhang, H., O. Van Kaick, and R. Dyer: 2007, 'Spectral Methods for Mesh Processing and Analysis'. *Proceedings of Eurographics* **92**(RC-20404), 1–22.

Zhang, H., O. Van Kaick, and R. Dyer: 2010, 'Spectral mesh processing'. *Computer Graphics Forum* **29**(6), 1865–1894.

Zwierzycki, M.: 2015, 'Squid [Online]'. Available from: www.food4rhino.com/project/squid [Accessed 17 June 2015].

Appendix A

Fourier analysis

This appendix contains a description of the relevant aspects of Fourier analysis in relation to the harmonic modelling framework and it serves to form the theoretical foundation for extending these principles to meshes in three dimensions.

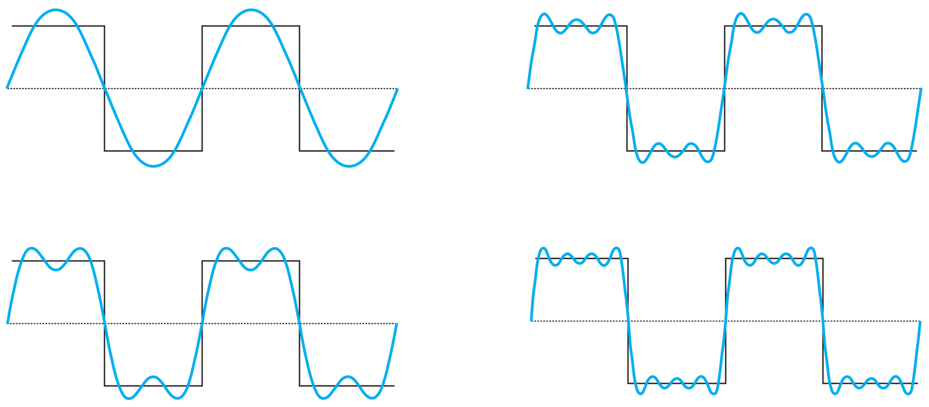


Figure A.0.1 – Continuous square wave function approximated by a Fourier Series with an increasing number of sinusoids

Fourier analysis enables the global approximation of a function by a sum of sinusoids as shown in Figure A.0.1 for a continuous square wave function. The more sinusoids that are included in the summation, the better approximation. This approximation is advantageous because it enables a possibly complex expression to be reduced to terms of trigonometric functions thus simplifying various problems.

In order to do so, the function defined in the spatial domain is transformed into the frequency domain by means of the Fourier Transform and then back to the spatial domain as a sum of sinusoids via the Inverse Fourier Transform. At first, this transformation from the spatial domain to the frequency domain may seem mysterious but the purpose is to change the perspective from *"what can I see?"* to *"what is it made of?"* (Azad, 2012). Imagine the function as a cake, then the Fourier Transform finds the recipe for that cake e.g 3.5 dl flour, 1.0 dl sugar, 2.0 dl water. This decomposition into ingredients is very useful because it describes the original input much better such that it can be analysed, compared and modified. Continuing with the cake analogy, the inverse Fourier Transform is then responsible for mixing the ingredients back together to the original cake.

The Fourier Transform is able to find the recipe by running the function through various filters that catch the different ingredients. Each filter has the property of only catching one specific ingredient and in total there must be as many filters as there are ingredients in order to catch them all. The idea is that any function (signal) can be filtered into various different circular paths and be rebuild again from them (Azad, 2012).

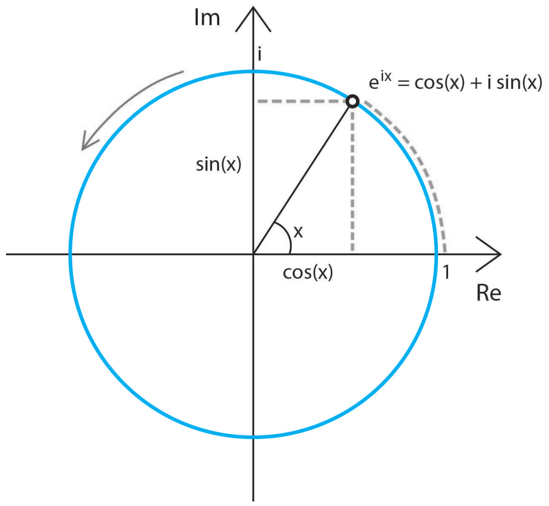


Figure A.0.2 – Traversing a circle in two different ways described by Euler's formula

A circular path (meaning how to move in circles) can be described in two different ways according to Euler's formula (Azad, 2010) and is visualised in Figure

A.0.2

$$e^{ix} = \cos(x) + i \cdot \sin(x) \tag{A.0.1}$$

Due to its simplicity and compactness the complex exponential function is used in the formula for the Fourier Transform. Euler's formula describes the movement along a unit circle (implicit a factor of 1 in front of the expression) but to make it more general any factor multiplied with the expression determines the amplitude (radius). The idea behind the Fourier Transform is therefore to filter any signal into cycles of different frequencies (as many as needed to catch all the ingredients), where each cycle is uniquely defined from the polar form of Euler's Equation ($r \cdot e^{ix}$) specifying the amplitude (radius) and phase (angle of starting point). The inverse Fourier Transform combines these cycles again to rebuild the signal. When different cycles are combined it causes constructive or destructive interference between them and it it this behaviour that is exploited to reach a specific target value.

Only the discrete version of the Fourier Transform (DFT) is described in the following as it is the most relevant version to practical applications where only a finite number of sample points exist. It also better relates to the application in this thesis since the generated 3-dimensional shape is represented by a mesh, which is the discrete version of a smooth surface.

A given signal X consisting of N real or complex numbers $X = [x_0, x_1, \ldots, x_{N-1}]^T$ can be transformed into a same-sized N-periodic signal of complex numbers by the following definition also known as the Discrete Fourier Transform (DFT)

$$\tilde{X_k} = \frac{1}{N} \cdot \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi ikn/N}, \ n \in \mathbb{Z}, \ k = 0, 1, \ldots, N-1 \tag{A.0.2}$$

In total N new values are obtained (one per frequency). It is generally the case that each new value $\tilde{X_k}$ is a complex number even if the signal only consists of real values. As mentioned, this complex number represents both the amplitude and phase of a circular path of frequency k cycles per N samples (Weisstein, 2015b), which gives an overall measure for the amount of a certain frequency that exists in the original signal. Thus, the signal has been transformed from the spatial domain into circular path ingredients in the frequency domain. From each circular path corresponding to a frequency k in the complex plane, the familiar sinusoidal curve (as seen from Figure A.0.1) can be obtained by tracing the real value of the moving point along the circle and it can be visualised by the expression (derived as the real part of the right-hand-side of Euler's equation)

$$f(x) = r \cdot \cos\left((2\pi k/N) \cdot x + \varphi\right) \tag{A.0.3}$$

The original signal can subsequently be reconstructed by the Inverse Discrete Fourier Transform (IDFT) defined by

$$x_n = \sum_{k=0}^{N-1} \tilde{X}_k \cdot e^{2\pi ikn/N}, \; k \in \mathbb{Z}, \; n = 0, 1, \ldots, N-1 \tag{A.0.4}$$

The reconstructed signal can be visualised by adding all the sinusoids together and the desirable behaviour is achieved as the combined wave passes through all the sample points. What happens in between is irrelevant because it is unknown how the signal travels there.

The discrete version of the square wave from Figure A.0.1 is used as an example to better understand the concepts and formulas described above.

Example - Discrete square wave

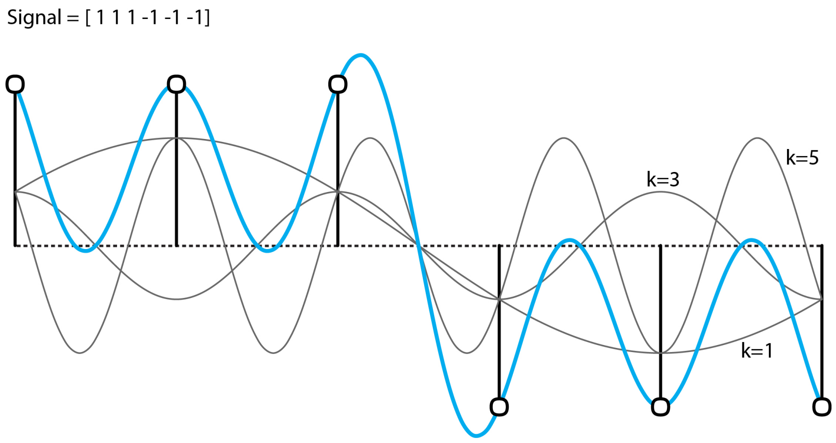


Figure A.0.3 – Discrete square wave with its frequency components

A discrete square wave signal consisting of six values (N=6) is given by

$X = [1, 1, 1, -1, -1, -1]^T$

The signal is visualised in Figure A.0.3 as vertical lines. The Discrete Fourier Transform (DFT) filters the signal into its frequency components as follows by inserting in Equation A.0.2

$$\tilde{\mathbf{x_0}} = \frac{1}{6} \cdot (1+1+1-1-1-1) = \mathbf{0}$$
$$\tilde{\mathbf{x_1}} = \frac{1}{6} \cdot \left(1 + e^{-\frac{1\cdot\pi}{3}\cdot i} + e^{-\frac{2\pi}{3}\cdot i} - e^{-\frac{3\pi}{3}\cdot i} - e^{-\frac{4\pi}{3}\cdot i} - e^{-\frac{5\pi}{3}\cdot i}\right) = \mathbf{\frac{2}{3} \cdot e^{-\frac{\pi}{3}\cdot i}}$$
$$\tilde{\mathbf{x_2}} = \frac{1}{6} \cdot \left(1 + e^{-\frac{2\pi}{3}\cdot i} + e^{-\frac{4\pi}{3}\cdot i} - e^{-\frac{6\pi}{3}\cdot i} - e^{-\frac{8\pi}{3}\cdot i} - e^{-\frac{10\pi}{3}\cdot i}\right) = \mathbf{0}$$
$$\tilde{\mathbf{x_3}} = \frac{1}{6} \cdot \left(1 + e^{-1\cdot\pi\cdot i} + e^{-2\cdot\pi\cdot i} - e^{-3\cdot\pi\cdot i} - e^{-4\cdot\pi\cdot i} - e^{-5\cdot\pi\cdot i}\right) = \mathbf{\frac{1}{3}}$$
$$\tilde{\mathbf{x_4}} = \frac{1}{6} \cdot \left(1 + e^{-\frac{4\pi}{3}\cdot i} + e^{-\frac{8\pi}{3}\cdot i} - e^{-\frac{12\pi}{3}\cdot i} - e^{-\frac{16\pi}{3}\cdot i} - e^{-\frac{20\pi}{3}\cdot i}\right) = \mathbf{0}$$
$$\tilde{\mathbf{x_5}} = \frac{1}{6} \cdot \left(1 + e^{-\frac{5\pi}{3}\cdot i} + e^{-\frac{10\pi}{3}\cdot i} - e^{-\frac{15\pi}{3}\cdot i} - e^{-\frac{20\pi}{3}\cdot i} - e^{-\frac{25\pi}{3}\cdot i}\right) = \mathbf{\frac{2}{3} \cdot e^{\frac{\pi}{3}\cdot i}}$$

It is observed that all of the even frequency components do not exist in the signal as their amounts are zero. The amplitude and phase of each frequency component that exist in the signal can be directly extracted from the complex numbers given on polar form as follows

$$\mathbf{a_1} = \mathbf{\frac{2}{3}}, \quad \varphi_\mathbf{1} = -\mathbf{\frac{\pi}{3}}$$
$$\mathbf{a_3} = \mathbf{\frac{1}{3}}, \quad \varphi_\mathbf{3} = \mathbf{0}$$
$$\mathbf{a_5} = \mathbf{\frac{2}{3}}, \quad \varphi_\mathbf{5} = \mathbf{\frac{\pi}{3}}$$

The sinusoids with frequency k=1, k=2 and k=3 and with their corresponding amplitude and phase are also plotted in Figure A.0.3 using Equation A.0.3. The sum of these sinusoids is highlighted in blue and it is observed how this curve exactly passes through the sample points of the original signal. It is furthermore noted how the three waves of different frequency reach the target value when they are summed up by either constructive or destructive interference. The figure proves that the values at the sample point locations exactly match the original signal when approximated by a sum of sinusoids. It can also be calculated and verified by the inverse discrete Fourier Transform (see Equation A.0.4) in a similar way.

The Discrete Fourier Transform has many practical applications which all take advantage of the ability to extract the ingredients of a given signal. A vibrational signal can be recorded from an earthquake and its ingredients (waves of different frequency and amplitude) are useful to building designers who can use the information to design structures with vibrational modes that do not interact with the strongest waves. Decomposing a sound wave into its ingredients makes it possible to remove certain noisy frequencies or to better compare the sound recipe with other sound waves, which is what music recognition software utilises. An image can also be interpreted as a two-dimensional signal and by finding its ingredients the less important ones can be ignored, which helps to compress the file and hence reduce the size (Azad, 2012). The following example describes the JPEG compression process in more detail.

Example - jpeg compression

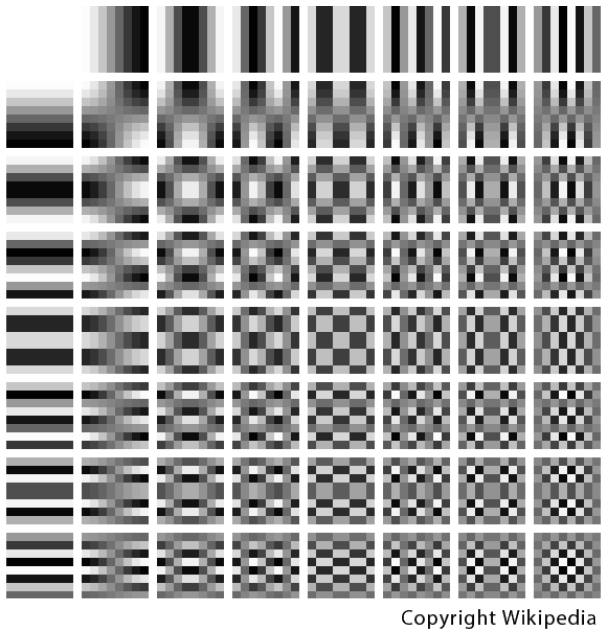


Figure A.0.4 – Cosine waves used for JPEG compression

The Discrete Fourier Transform (more specifically the Discrete Cosine Transform, DCT) is used for compression of JPEG images, thereby enabling a trade-off between image quality and file size. An image consists of a 2D spatial domain of pixels ordered in rows and columns, where each pixel has a value between 0-255 specifying a colour. For the human eye to perceive an image, sharp transitions in intensity are ignored. These sharp transitions in intensity correspond to values calculated by the DCT in the higher frequency domain. Since these are all small, the general idea is to discard that information. This is why it is useful to extract the ingredients of the image, which is exactly what the Fourier Transform is responsible for.

The procedure is to subdivide the image into smaller domains of 8 x 8 pixel and construct a corresponding 8 x 8 matrix representing the colour value in each pixel. Since the matrix is a 2D domain, a two-dimensional DCT is applied and the result is a new 8 x 8 matrix with values that each represents the amount of a wave with specific frequency that exists in the image signal. The 64 2D spanning cosine waves of increasing frequency are illustrated in Figure A.0.4. Any image signal can be constructed by a linear combination of these waves.

Since the new matrix contains as many values as the image signal itself this is a lossless process and nothing has been achieved. The value arises when a certain amount (specified by compression ratio) of the smaller coefficients are rounded off to zero. By this operation the 64 new values can be reduced to the number of non-zero values and this is the only information the computer needs to store thus reducing the file size. This process is possible because the 64 waves illustrated in the figure are constant due to the fixed structure of the 8 x 8 row/column pixel blocks and the image can therefore be reconstructed from the knowledge of the non-zero coefficients and which waves they belong to (Wikipedia, 2015b) .

The transformation from the spatial domain to the frequency domain is essentially a change of basis. A basis is a set of linearly independent vectors (Rowland, 2015), which implies that one vector cannot be represented as a scalar multiple of another vector in this set. Thus, it is possible to represent any vector as a linear combination of all the basis vectors. The concept of two different bases ($\{v_i\}$ and $\{u_i\}$) is shown in Figure A.0.5. Any vector a_v can be described from each of those bases. A change of basis from $\{v_i\}$ to $\{u_i\}$ can be achieved by an orthogonal projection of the vector a_v via the vector dot product. This concept is not restricted to 2D or 3D but it is obviously easier to illustrate it in these dimensions. However, the following example serves to increase the understanding of how this concept can be used to reformulate the equations related to the Fourier Transform by expanding it to higher order dimensions.

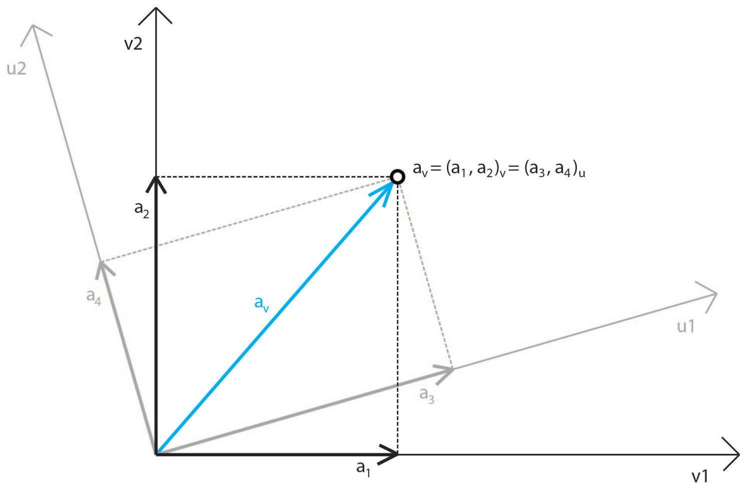


Figure A.0.5 – Change of basis of an arbitrary vector a_v

Example - Change of basis

A standard orthonormal basis (denoted *"st"*) is defined by the set of normalised vectors

$$\vec{v}_1 = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \vec{v}_2 = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \vec{v}_3 = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \vec{v}_4 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

The dot product of any vector with another vector in this set is equal to zero. A given signal X in this basis can therefore be represented by a linear combination of the set

$$X = \begin{pmatrix}3\\2\\5\\8\end{pmatrix}_{st} = 3 \cdot \begin{pmatrix}1\\0\\0\\0\end{pmatrix} + 2 \cdot \begin{pmatrix}0\\1\\0\\0\end{pmatrix} + 5 \cdot \begin{pmatrix}0\\0\\1\\0\end{pmatrix} + 8 \cdot \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Another orthonormal basis (denoted *"new"*) is defined by the set of normalised vectors

$$\vec{u}_1 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \ \vec{u}_2 = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}, \ \vec{u}_3 = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix}, \ \vec{u}_4 = \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix}$$

Again, the dot product of any vector with another vector in this set is equal to zero. The change of basis (projection of the signal X from the standard basis onto the new basis) gives the following coefficients

$$a_1 = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 8 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} = 9, \ a_2 = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 8 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} = -4$$

$$a_3 = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 8 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix} = 2, \ a_4 = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 8 \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} = -1$$

The signal in each basis is therefore

$$\begin{pmatrix} 3 \\ 2 \\ 5 \\ 8 \end{pmatrix}_{st} = \begin{pmatrix} 9 \\ -4 \\ 2 \\ -1 \end{pmatrix}_{new}$$

Multiplying the coefficients with the corresponding vectors in the new basis and summing those, reconstructs the original signal in the standard basis

$$\sum a_i \cdot \vec{u}_i = \mathbf{9} \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \mathbf{-4} \cdot \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} + \mathbf{2} \cdot \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \end{pmatrix} \mathbf{-1} \cdot \begin{pmatrix} \frac{1}{2} \\ -\frac{1}{2} \\ \frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} = \begin{pmatrix} 3 \\ 2 \\ 5 \\ 8 \end{pmatrix}_{st}$$

The Fourier Transform can therefore be explained as a change of basis where the desirable signal (function) to approximate is defined in space by the standard basis ($\vec{i}$, $\vec{j}$, $\vec{k}$ unit vectors following the x, y and z axes respectively) and changed during the transformation into a frequency basis build from the complex exponential function $e_k = e^{2\pi ikn/N}$. In both cases the set of vectors defining the basis are orthogonal to each other (the dot product of a pair of vectors equals zero). According to the previous example, this change of basis can be achieved by a projection of the spatial vector signal X onto the frequency basis vectors e_k. As a result, the DFT as defined in Equation A.0.2 can be reformulated as

$$\tilde{X_k} =< X,\ e_k > \tag{A.0.5}$$

Here $<,>$ symbolises the dot product. As the dot product geometrically describes how much of one vector is contained in another vector, the coefficient $\tilde{X_k}$ is simply a measure of the amplitude of the sinusoid with frequency k that is contained in the spatial signal X. The IDFT as defined in Equation A.0.4 can similarly be reformulated as (Botsch et al., 2010)

$$x_n = \sum_{k=0}^{N-1} < X,\ e_k > \cdot e_k \tag{A.0.6}$$

Appendix B

Boundary conditions study

This appendix contains a detailed study on how to artificially impose boundary conditions by a manipulation of the Laplacian matrix. The study is based on a simple 2D example of a string with the reference shown in Figure B.0.1 (top).

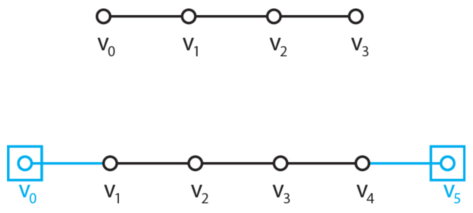


Figure B.0.1 – The reference string set-up where the boxes symbolise fixities

The graph Laplacian for this 4-vertex string is defined as

$$\underline{L} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

The corresponding eigenvalues and eigenvectors are

$$\lambda_0 = 0.0,\ \lambda_1 = 0.586,\ \lambda_2 = 2.0,\ \lambda_3 = 3.414$$

$$\vec{v}_0 = \begin{pmatrix} -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \end{pmatrix}, \vec{v}_1 = \begin{pmatrix} 0.653 \\ 0.271 \\ -0.271 \\ -0.653 \end{pmatrix}, \vec{v}_2 = \begin{pmatrix} 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \end{pmatrix}, \vec{v}_3 = \begin{pmatrix} -0.271 \\ 0.653 \\ -0.653 \\ 0.271 \end{pmatrix}$$

The eigenvectors interpreted as vertex displacements of the reference string are illustrated in Figure B.0.2.

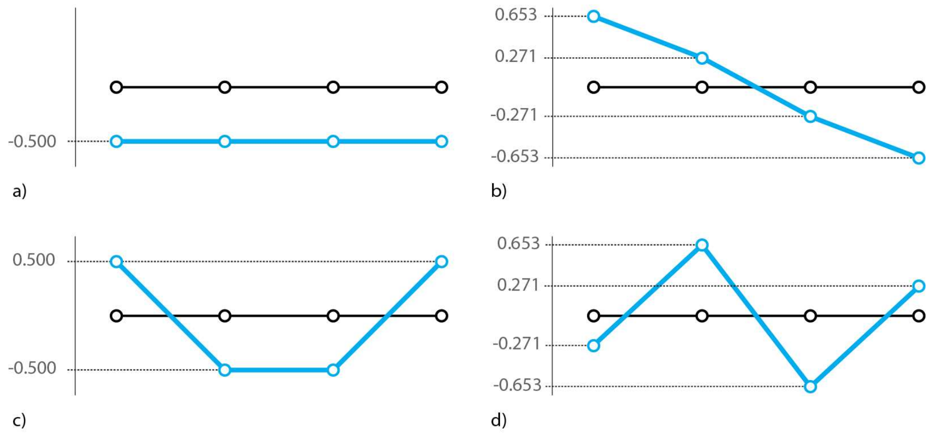


Figure B.0.2 – The displacement eigenfunctions of a 4-vertex string where (a) represents v_0, (b) v_1, (c) v_2 and (d) v_3

To evaluate the influence of the different matrix manipulation options, the 4-vertex reference string is expanded with one additional vertex in both ends as shown in Figure B.0.1 (bottom) where the goal is to fix those vertices.

Row/column elimination or equivalent stiffness method

One option is to ignore the vertices that are specified as fixed and thus only set up the Laplacian matrix for internal vertices or in other words eliminate rows/columns corresponding to the fixed vertices. The outcome with this strategy is in this case a 4 x 4 matrix with the same eigenvalues and eigenvectors as the reference. Since the string consists of six vertices, these four values are only mapped back to the internal vertices and the fixed vertices stay in place. In relation to Figure B.0.2 it means that the end points of the string are connected back to points located at the zero line. However, when the simple string is replaced with a complex mesh, where the vertices may be sorted in an arbitrary order, it becomes cumbersome to keep track of which value in the eigenvector corresponds to which vertex.

To avoid this complexity it is desirable to maintain the size of the Laplacian matrix such that each value in the eigenvectors is mapped to one vertex. As the Laplacian matrix functions similarly to the stiffness matrix in a finite element program, it can be thought of in terms of stiffness as well. The stiffness matrix in a finite element program is part of another equation though $\left(\vec{F} = \underline{K} \cdot \vec{d}\right)$ and in this case a zero displacement is obtained by setting the force element to zero, replace row/columns values with zero and set one as the diagonal value in $\underline{K}$ for the relevant nodes. An equivalent to this action, which can be used for this framework instead, is to increase the stiffness of a vertex i.e. set the diagonal value to a large number and the row/column values to zero. Thereby, the modified graph Laplacian for the 6-vertex string becomes

$$\underline{L} = \begin{bmatrix} \mathbf{1000} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & 1 & -1 & 0 & 0 & \mathbf{0} \\ \mathbf{0} & -1 & 2 & -1 & 0 & \mathbf{0} \\ \mathbf{0} & 0 & -1 & 2 & -1 & \mathbf{0} \\ \mathbf{0} & 0 & 0 & -1 & 1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1000} \end{bmatrix}$$

By doing so, the same result as the row/column elimination is obtained but the eigenvector now consists of six values instead of four with values equal to zero for the fixed vertices. Since the Laplacian matrix now is a 6 x 6 matrix it also means that six eigenvalues and eigenvectors are obtained instead of only four. These are listed below and visualised in Figure B.0.3.

$$\lambda_0 = 0.0,\ \lambda_1 = 0.586,\ \lambda_2 = 2.0,\ \lambda_3 = 3.414,\ \lambda_4 = 1000.0,\ \lambda_5 = 1000.0$$

$$\vec{v}_0 = \begin{pmatrix} 0.0 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ 0.0 \end{pmatrix},\ \vec{v}_1 = \begin{pmatrix} 0.0 \\ 0.653 \\ 0.271 \\ -0.271 \\ -0.653 \\ 0.0 \end{pmatrix},\ \vec{v}_2 = \begin{pmatrix} 0.0 \\ 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \\ 0.0 \end{pmatrix}$$

$$\vec{v}_3 = \begin{pmatrix} 0.0 \\ -0.271 \\ 0.653 \\ -0.653 \\ 0.271 \\ 0.0 \end{pmatrix}, \vec{v}_4 = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix}, \vec{v}_5 = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{pmatrix}$$

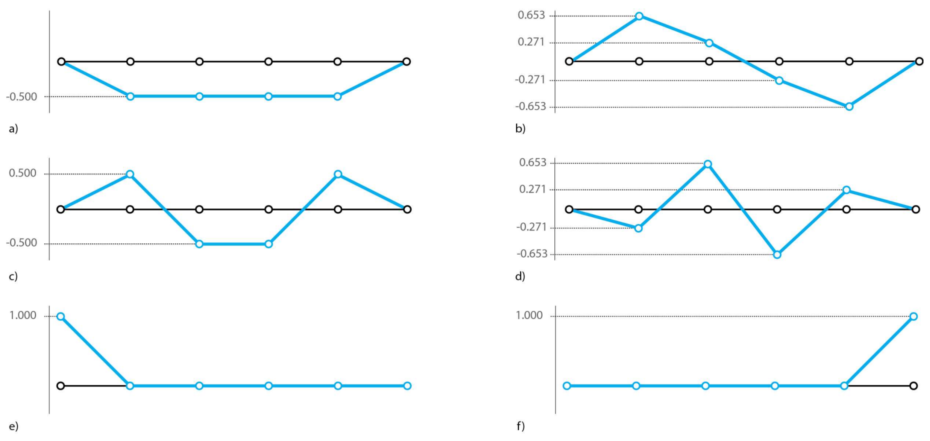


Figure B.0.3 – The initial displacement eigenfunctions of a 6-vertex string with fixed ends where (a) represents v_0, (b) v_1, (c) v_2, (d) v_3, (e) v_4 and (f) v_5

It is observed that the last two eigenvectors do not have zero values corresponding to the fixed vertices and they are therefore excluded. Thus, the number of desirable eigenvectors k can more generally be expressed as

$$k = n - c$$

Where n is the number of vertices and c is the number of imposed constraints. It is useful that the excluded eigenvectors are located last in the list because then it is safe to only ask for k eigenvalues and eigenvectors as part of the eigendecomposition. It is furthermore noticeable that the last two eigenvectors have significantly larger eigenvalues which mirror the manipulated value for the vertex stiffness. It means that it takes much more energy to displace the artificially "fixed" vertices than it does for the internal vertices. As the eigenvectors are sorted in ascending order according to the eigenvalues the only requirement to the vertex stiffness is that it has to be larger than the highest eigenvalue obtained for the 4-vertex string reference. Otherwise the desirable excluded eigenvectors will no longer be located last in the list. The downside of this method is the non-smooth boundary transitions due to the lack of connectivity information in the Laplacian matrix.

Inclusion of edge connectivity information

To address this problem the edge connectivity information for the fixed vertices is included to avoid the isolated behaviour. Hence, zero gets replaced with minus one in the row/column element of a fixed vertex if it is connected to another vertex by an edge. Following this rule the modified graph Laplacian is written as

$$\underline{L} = \begin{bmatrix} 1000 & \mathbf{-1} & 0 & 0 & 0 & 0 \\ \mathbf{-1} & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 1 & \mathbf{-1} \\ 0 & 0 & 0 & 0 & \mathbf{-1} & 1000 \end{bmatrix}$$

The corresponding eigenvalues and eigenvectors are listed below

$$\lambda_0 = -0.001,\ \lambda_1 = 0.585,\ \lambda_2 = 1.999,\ \lambda_3 = 3.414,\ \lambda_4 = 1000.001,\ \lambda_5 = 1000.001$$

$$\vec{v}_0 = \begin{pmatrix} 0.001 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.001 \end{pmatrix},\ \vec{v}_1 = \begin{pmatrix} -0.001 \\ -0.653 \\ -0.271 \\ 0.271 \\ 0.653 \\ 0.001 \end{pmatrix},\ \vec{v}_2 = \begin{pmatrix} -0.001 \\ -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \\ -0.001 \end{pmatrix}$$

$$\vec{v}_3 = \begin{pmatrix} 0.0 \\ 0.271 \\ -0.653 \\ 0.653 \\ -0.271 \\ 0.0 \end{pmatrix},\ \vec{v}_4 = \begin{pmatrix} -0.718 \\ 0.001 \\ 0.0 \\ 0.0 \\ 0.001 \\ -0.696 \end{pmatrix},\ \vec{v}_5 = \begin{pmatrix} -0.696 \\ 0.001 \\ 0.0 \\ 0.0 \\ -0.001 \\ 0.718 \end{pmatrix}$$

The fixities are weakened by the inclusion of the edge connecticity information as evident from the eigenvector values and the plot in Figure B.0.4. In other

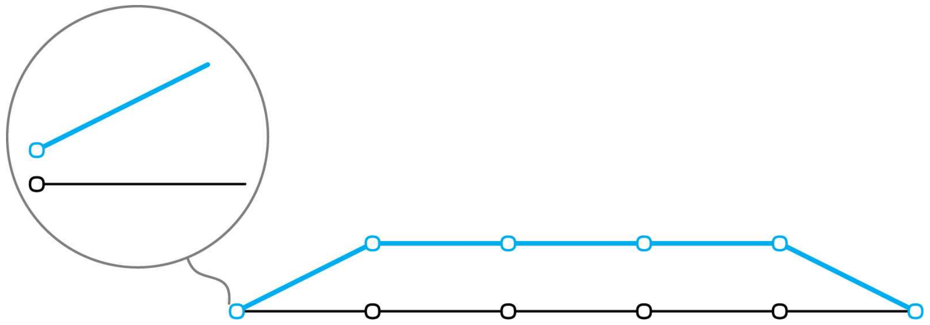


Figure B.0.4 – Weakened fixities due to the inclusion of edge connectivity information. The first eigenvector is plotted

words, the vertex next to the fixed vertex pulls it upwards by its edge because it now knows it is connected to it. The ratio between the artificial stiffness to fix a vertex and the edge weight determines the amount of movement. Hence, if the vertex stiffness is decreased the movement becomes larger. Since the ratio in this case is $1/1000$, the solution is very close to the previous example only with an irrelevant change of sign, which is arbitrary. However, this weakening of the fixities is in general undesirable. As a consequence, the artificial vertex stiffness for the fixities is further increased to avoid these effects and the edge connectivity information is kept in the matrix as it captures changes in topology, which the stiffness for internal vertices may not detect on its own as shown in Figure B.0.5.

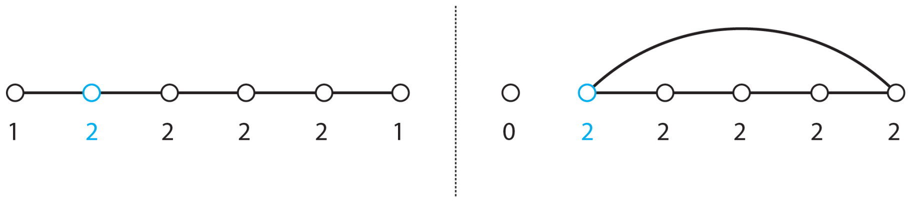


Figure B.0.5 – Example of how the edge connectivity information captures changes in the topology of a string. The highlighted vertex have the same vertex stiffness in both cases but is connected by different edges and therefore a different behaviour is expected

Inclusion of vertex connectivity information

The diagonal values in the matrix associated with the vertices have to include the connectivity information as well. It means that a vertex connected to a fixity has to update its stiffness to its new valence, which changed when the additional fixed end vertices were attached to the string. The modified Graph

Laplacian with the increased stiffness for fixed vertices and updated stiffness for vertices connected to a fixity is written as

$$\underline{L} = \begin{bmatrix} 100000 & -1 & 0 & 0 & 0 & 0 \\ -1 & \mathbf{2} & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & \mathbf{2} & -1 \\ 0 & 0 & 0 & 0 & -1 & 100000 \end{bmatrix}$$

The result from the eigendecomposition is listed below and visualised in Figure B.0.6. The desirable effect can mainly be recognised from the first two modes due to the coarseness of the string. It is observed how the increased stiffness of a vertex connected to a fixity results in a larger resistance to move and the edge connectivity information helps to tie down the adjacent vertex whereby it nicely approximates a sinusoid with a smooth boundary transition. It is furthermore seen that the first eigenvalue is no longer zero and the corresponding pure translation mode has disappeared.

$$\lambda_0 = 0.382,\ \lambda_1 = 1.382,\ \lambda_2 = 2.618,\ \lambda_3 = 3.618,\ \lambda_4 = 100000.0,\ \lambda_5 = 100000.0$$

$$\vec{v}_0 = \begin{pmatrix} 0.0 \\ 0.372 \\ 0.602 \\ 0.602 \\ 0.372 \\ 0.0 \end{pmatrix},\ \vec{v}_1 = \begin{pmatrix} 0.0 \\ -0.602 \\ -0.372 \\ 0.372 \\ 0.602 \\ 0.0 \end{pmatrix},\ \vec{v}_2 = \begin{pmatrix} 0.0 \\ -0.602 \\ 0.372 \\ 0.372 \\ -0.602 \\ 0.0 \end{pmatrix}$$

$$\vec{v}_3 = \begin{pmatrix} 0.0 \\ -0.372 \\ 0.602 \\ -0.602 \\ 0.372 \\ 0.0 \end{pmatrix},\ \vec{v}_4 = \begin{pmatrix} 1.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{pmatrix},\ \vec{v}_5 = \begin{pmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{pmatrix}$$

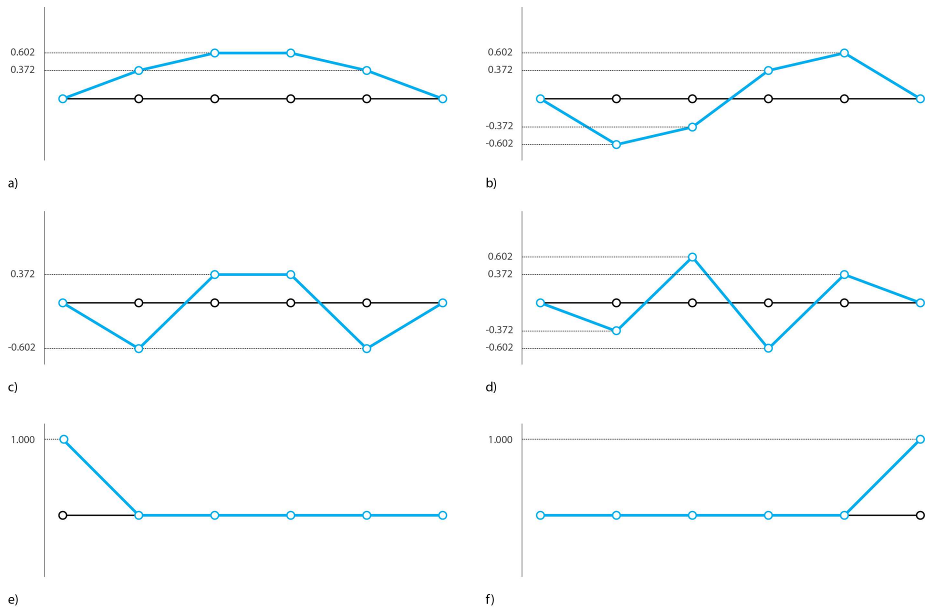


Figure B.0.6 – The displacement eigenfunctions of a 6-vertex string with fixed ends (final) where (a) represents v_0, (b) v_1, (c) v_2, (d) v_3, (e) v_4 and (f) v_5

Summary

In summary, vertex fixities can be imposed by constructing the Laplacian matrix from its general definition (see Equation 3.1.4) considering the entire mesh to begin with and subsequently replace the stiffness of fixed vertices with a much larger number e.g. 100,000. That way both edge and vertex connectivity is included. The above described process merely serves to highlight the influence of the different parameters involved in the artificial manipulation of the Laplacian matrix.

A further refined string as shown in Figure B.0.7 is used as a more convincing example to visualise the resulting modes from fixing the end vertices (blue) and a comparison with continuous sinusoids of similar amplitude and frequency (grey). The importance of the embedded connectivity info is clearly seen by a comparison with the modes resulting from a matrix decomposition without that information included (orange).

The smoothness of the result heavily relies on the equally spaced vertices in the string beacuse the matrix and therefore also the computed eigenvectors remain unchanged when the geometry is modified. It is therefore obvious that the plotted modes will deviate from the shape of sinusoids when the displacement values stay intact but the vertices are repositioned.

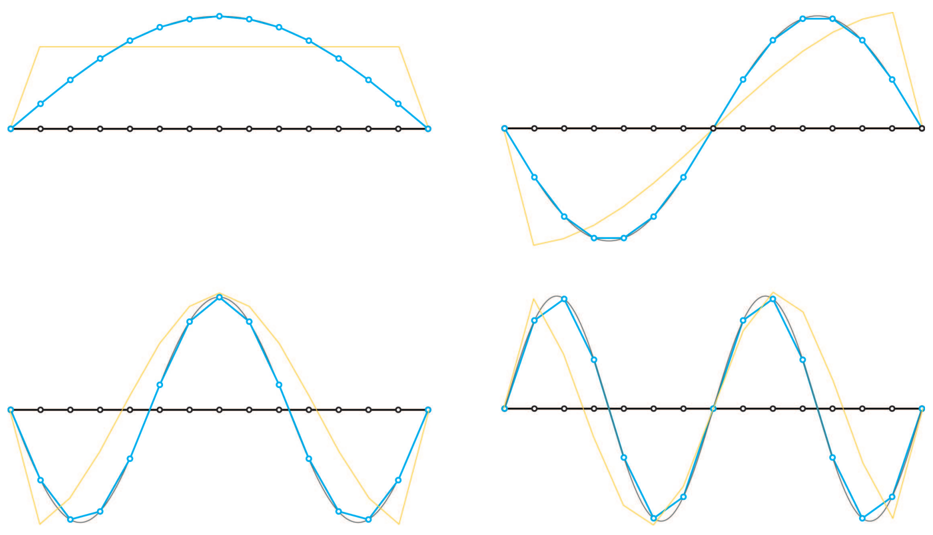


Figure B.0.7 – The first four displacement eigenfunctions of a refined string with fixed ends (blue) in comparison with continuous sinusoids of similar wave lengths and amplitudes (grey). The eigenfunctions with non-smooth boundary transitions as a result of omitting connectivity information are also shown (orange)

Appendix C

Plug-in manual

A harmonic form-finding plug-in for Grasshopper is the outcome of this research. It consists of a number of components as seen in Figure C.0.1, which are grouped into the following six categories: Mesh, Matrix, Harmonics, Buckling, Display and Utility. A small manual with a brief overview of the components in each category is provided in the following.

Mesh

The components in this category aim to support the modelling with Plankton meshes. See Table C.0.1.

Matrix

The components in this category build the necessary matrices and perform linear algebra operations to establish the framework for harmonic modelling. See Table C.0.2.

Harmonics

These components utilise the framework to model with harmonics and extend the advantageous properties of Fourier analysis to meshes. See Table C.0.3.

Buckling

The components in this category aid to evaluate the buckling capacity of the harmonic shapes in a pursuit to exploit their doubly curved nature. The tools

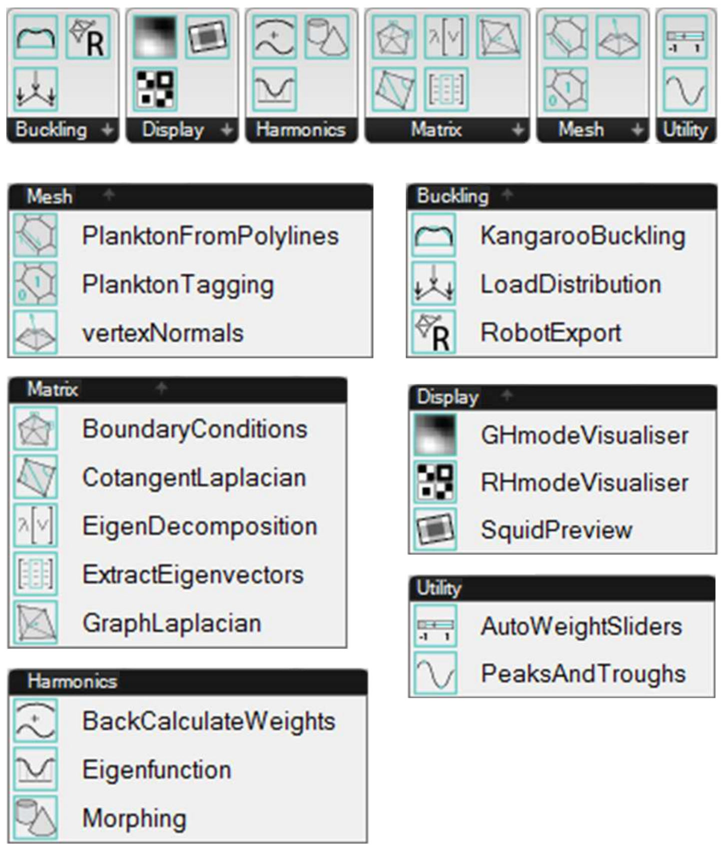


Figure C.0.1 – Harmonic form-finding plug-in for Grasshopper

are ideal for an optimisation work flow due to their integrity and computational speed. See Table C.0.4.

Display

These components help to visualise the mode shapes in an attempt to increase the level of intuition associated with this tool. See Table C.0.5.

Utility

Lastly, the components in this category help to support the work flow and provide additional useful information. See Table C.0.6.

Name	Functionality
PlanktonFromPolylines	Creates a Plankton mesh from face polylines (CCW direction) and vertex points
PlanktonTagging	Tags vertex, halfedge and face indexes to display the mesh connectivity
vertexNormals	Computes the vertex normals as a weighted average of the neighbouring face normals (eventually normalised)

Table C.0.1 – Mesh components

Name	Functionality
GraphLaplacian	Constructs the Laplacian matrix based on the topology of the Plankton mesh
CotangentLaplacian	Construct the Laplacian matrix based on the geometry and topology of the Plankton mesh
BoundaryConditions	Manipulates the Laplacian matrix to impose boundary conditions
EigenDecomposition	Computes the eigendecomposition of the Laplacian matrix to obtain the desired orthonormal basis (eigenvectors) with harmonic behaviour
ExtractEigenvectors	Extracts specific eigenvectors from the eigenvector matrix according to an index list

Table C.0.2 – Matrix components

Name	Functionality
Eigenfunction	Generates harmonic shapes by a linear combination of eigenvectors with predefined arbitrary weights
BackCalculateWeights	Back-calculates the most significant eigenvectors (modes) and their corresponding weights to approximate a target surface from a base mesh. The output is suitable for the "Eigenfunction" component
Morphing	Explores the design space in-between two limit surfaces in a non-linear way

Table C.0.3 – Harmonics components

Name	Functionality
LoadDistribution	Calculates the lumped forces in each vertex based on the voronoi areas to mimic self-weight
KangarooBuckling	Performs a non-linear buckling analysis of a shell using a dynamic relaxation approach with Kangaroo
RobotExport	Performs a linear buckling analysis of a shell with Autodesk Robot

Table C.0.4 – Buckling components

Name	Functionality
GHmodeVisualiser	Generates the necessary input for the “SquidPreview” component to display the mode shapes as bitmaps on the Grasshopper canvas
SquidPreview	Creates 10 Squid bitmaps as default
RHmodeVisualiser	Visualises the mode shape catalogue in the Rhino viewport to obtain a better overview

Table C.0.5 – Display components

Name	Functionality
AutoWeightSliders	Auto generates sliders to specify weights in a range between -1 and 1 (two decimals) with useful labelling
PeaksAndTroughs	Calculates the number of peaks and troughs for each mode shape from the mesh geometry and topology

Table C.0.6 – Utility components