Chapter 6

Parametric Models (cell-like grid)

This chapter starts to implement parametric design strategy, to design and optimize the cell-like grid structure for Shanghai Natural History Museum (SNHM). The first step is building the three dimensional computer models (by configuration of parameters) in which all unique structural elements are described. These models will be analysis and evaluated to explore an ‘optimal’ structure.

This chapter will give description for several parametric models of the cellular wall.

Models can be summarized by categories:

- Structured grid models
- Modified structured grid models
- Unstructured grid models

For each model, description and discussion include:

- The architectural geometry theory
- Strategies and parameters (Variables)
- Implementation of the geometry generation tools
- Advantages and disadvantages

6.1 Surface generation

As described in Chapter 3.2.2, the surface of the cellular wall is a single curved surface (developable). It can be generated as Ruled / Lofted BSpindeSurface in GenerativeComponents. The design of the building shape and floor plan was provided in AutoCAD drawing. The file (.dwg) format is supported in Rhinoceros[6] and to get the construction information of the free form curves [from floor plans].

1. Analysis the boundary curves in Rhinoceros and write a point file (.txt) of the control points to construct the free form Curves; When write coordinates of the control points to the point file, all the points should be listed by sequence – in one direction along the curves, this is important for building up a correct BSpindeCurve.

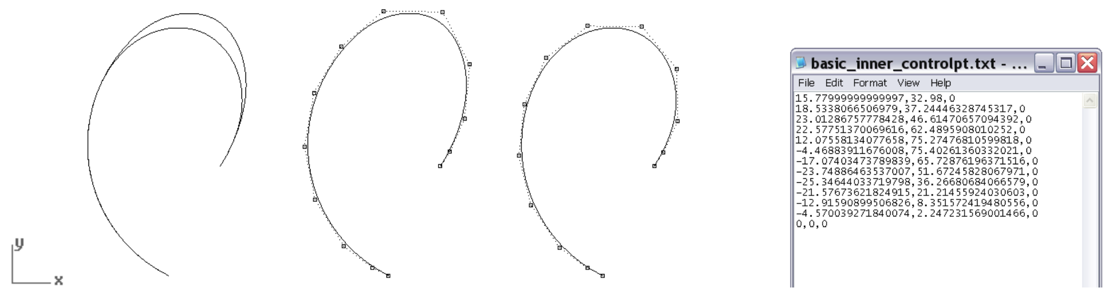


Fig. 6.1 the control points of the boundary curves (left) and point file (right)

2. Import the point file (.txt) to GenerativeComponents to rebuild the boundary curves [BSpindeCurve.bypoles] and generate the wall surface [BSpindeSurface.LoftCurve];

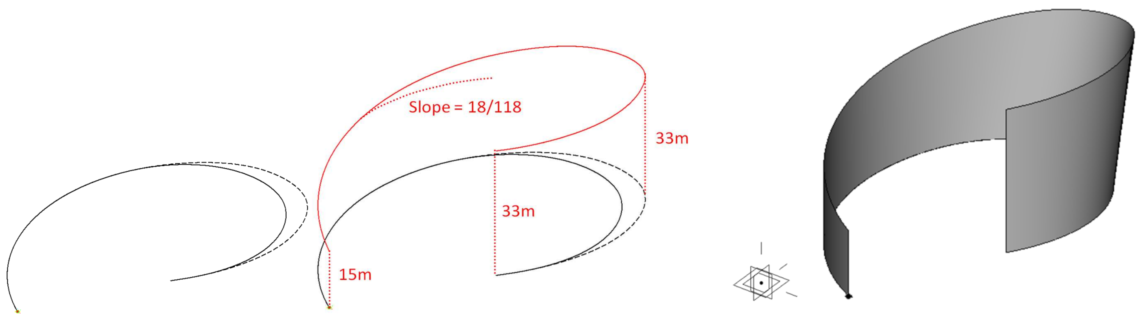
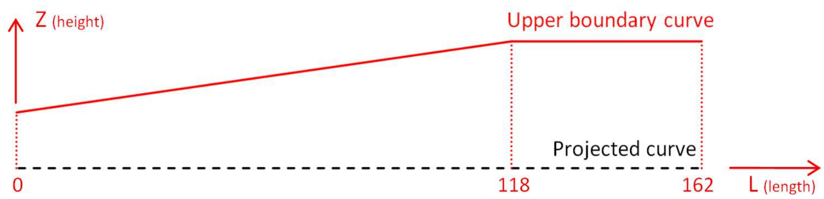


Fig. 6.2 geometry details of the boundary curves (left) and the wall surface (right)

Unfolded curve (2d) shows the calculation of the upper curve: Z = 15+L*(18/118) [L<=118m] or 33 [L>118m]



[6] 3D NURBS modeling program Rhinoceros [McNeel, 2005]
The possibilities for sculptural modeling with this software package are extensive. But for parametric associative structural modeling, Rhinoceros is no very suited. Rhinoceros is not equipped with user friendly interfaces that enable the structural engineer to model the structural design intend, instead of just drawing. A feature in Rhinoceros that might enable skilled programmers to develop a structural design in a parametric associative way is RhinoScript, which is based on Microsoft's VBScript language. For the design stage of modeling the structural geometry, the parametric associative software package GenerativeComponents [Aish, 2005] offers many advantages over scripting the structural geometry with Rhinoceros.

[Surface information for Assembling]

Technical Value of the structure is also an important criterion for evaluation of a structure, and it is mainly depended on the assembling/construction solution. The structural topology of an irregular cell-like pattern is complex, it requires carefully coding and coordinating every single elements for assembling.

A proposal solution is to build up frameworks according to the geometry of the all surface, as temporary props. Individual Elements (RHS steel tube and joins) will be supported on a network of temporary props and welded together first. After that, these large welded segments can be positioned on site and welded in-situ to finish the construction. By doing this, the assembly process – especially the welding work will be much easier.

- Divide the wall surface into pieces
- Generate geometric information for the pieces/segments of the surface

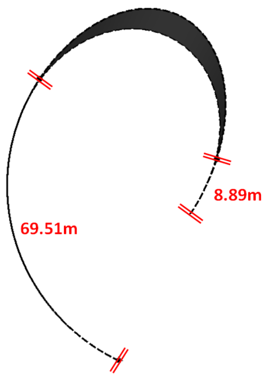
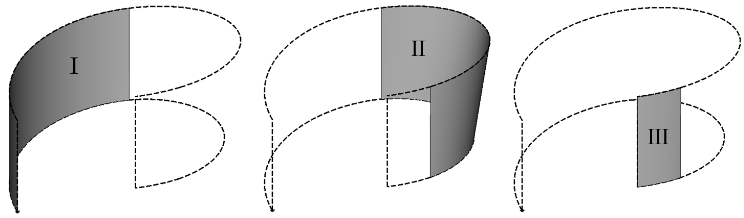


Fig. 6.3 According to the curvature and the local inclination (vertically) of this wall, the whole surface can be divided into 3 main parts: Part I & III have no inclination, thus they are surfaces on cylinder coordinate; and Part II has outward inclination. Surface subdivision can be executed separately in three three parts.



Discrete the subdivided surface into vertical/horizontal strips (u,v) – surface information can be arranged and write to output files [Excel/Text documents], repairing for frameworks (temporary props).

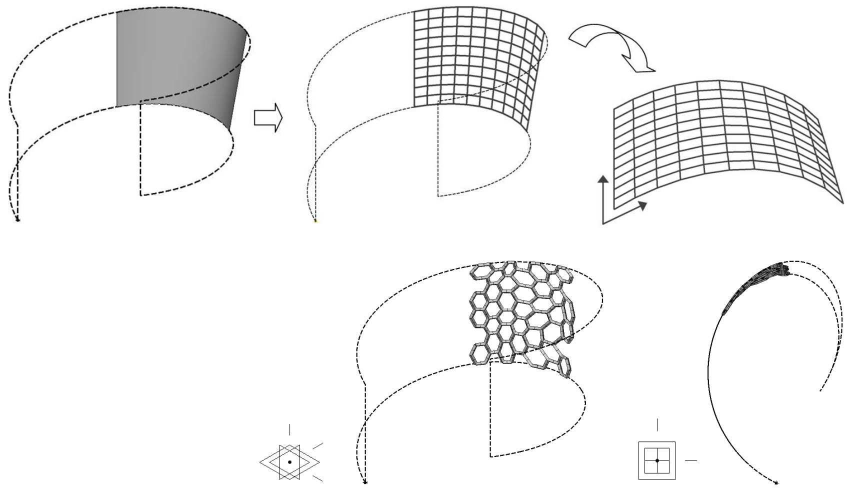


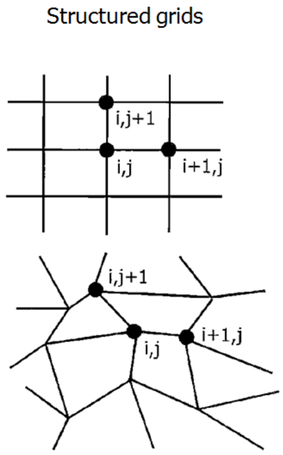
Fig. 6.4 Example of surface subdivision and segment generation

The previously described process have developed a free form surface which is almost ready to used for generating structural geometry in GenerativeComponents. What is missing is a (point) grid on which the structural elements can be allocated. The following parts will focus on generating a suited structural grid, and building up the structural elements for the grid structure.

Basic on the literature study for grid generation technology in Chapter3.1, the grid models for cellular wall can be described in 3 categories: structured grid model, modified structured grid model, and unstructured grid model. The grid generation algorithm, implementation and grid property will be described for each category.

6.2 Structured grid model

"A mesh is considered to be a structured grid if the local organization of the grid points and the form of the grid cells do not depend on their position but are defined by a general rule. The connectivity of the grid is implicitly taken into account. – Liseikin, 1999"



Introduction

The grid points were organized and listed in a 2D array by sequence to create a rule for structured grid. Changing the position of any grid point, the connectivity of the grid won't be changed. Only the grid cell will deform (cell-shape changed). Thus, the main design variables in the structure grid model are: the average grid size and the scale of randomness, which is defined by the range of the grid point's movement.

Purpose

Taking the advantages of structured grids - easy to implement and good efficiency (accuracy), the main purpose to create the structured grid model was to have a parametric model which can be easily handled and quickly implemented.

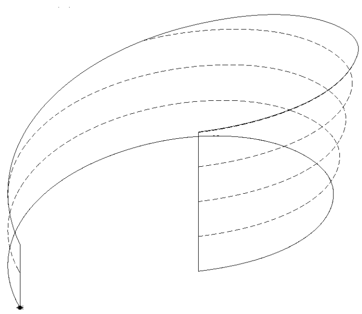
Generated by organized grid points and simple rules, the resulting structured grid models can be modified by changing size, and the margin of randomness - to get an irregular appearance. With this model, a series of grid sizes can be tested to provide recommended values for the cellular wall structure.

6.2.1 Generate grid points

The starting point of the structured grid model was to generate regular triangle grids; modeling steps:

1. Define graph variable L as grid-size

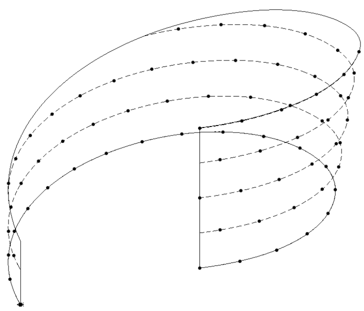
2. Create cutting curves along surface



The distances between different cutting curves were defined by L (grid-size); but in order to divide the surface equally into several horizontal regions, this pre-defined grid-size was used to round up the number of division, and a new size was recalculated:

```
int n = pt[pt.Count].Z/L;          // the surface height/pre-defined grid size
                                   // signing double value into int, to round it up
double newsize = pt[pt.Count].Z/n;        // calculate new grid size for cutting curve
```

3. Generate points on each cutting curve



The same calculation was implemented to round up the number of division; In general, there will be a offset (= size/2) of the grid points between the ever and odd cutting curves; However, due to the incline angle of the wall surface, the 'elongation' of the cutting curves should be taken into account; The implemented solution was to introduce a increment according to the height of the cutting curve:

```
double increment = (Outercurve.Length- Innercurve.Length)*cur.EndPoint.Z/33;
                                            // total increment of each cutting-curve (cur)
pt[j].ByDistanceAlongCurve(cur, j*size + increment*j/n));
                                           // pt[j] (j=0~n) will slightly move by 'increment*j/n'
```

Grid points generated along each cutting curve were organized in number, as preparation for structured grid generation. Extra grid points were also generated along the upper boundary curve, which were use to fill in the edge – continue the grids to the boundary. More details will be described in the following section – solution for the edge.

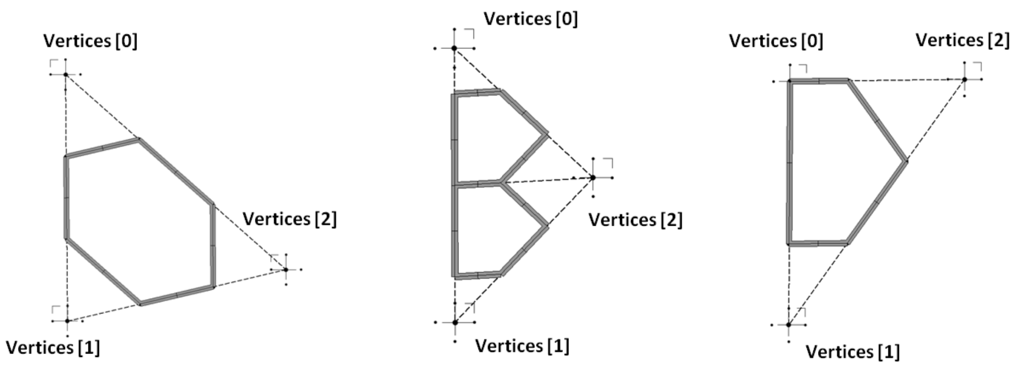
4. Create triangle grid (structured)

Triangular grid was generated as ‘intermediate’ grid to create hexagon patterns. This triangular grid was a structured grid, and the hexagonal grid was depended on it.

6.2.2 Define new feature

For structured grids, hexagonal pattern can also be generated by scripting. However, to directly define hexagonal grids requires for more organized grid points (6 vertices in each hexagon) and the topology is more complex. The implemented solution was to define new features, hexagonal in-pack to triangle grid, modify the grids from triangles to hexagons, to simplify the topology. Another advantage is that the hexagon elements are built on the triangle surface, they are all in-plane. This is an important point when the structural grid is designed to integrate glass façade.

3 features were defined to generate hexagon pattern: PtoH, Pto2H, and Ptohalf_H (from Points to Hexagon). In which, feature Pto2H, and Ptohalf_H were used to complete the grid near the boundaries. (Note: the vertices must be selected in sequence, except the feature ‘PtoH’ – there is no influence caused by random connect vertices because of the multi-symmetry.)



[Feature Name] 1. PtoH 2.Pto2H 3.Ptohalf_H

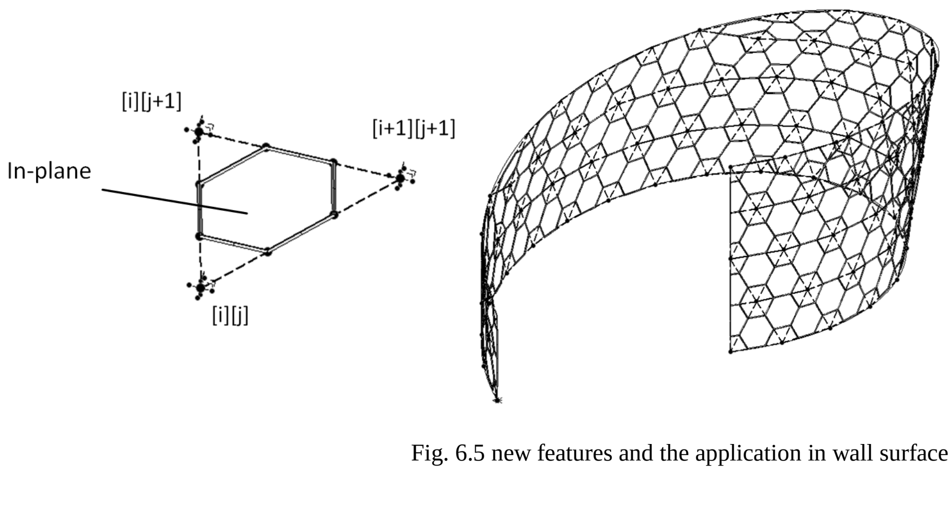


Fig. 6.5 new features and the application in wall surface

6.2.3 Solution for the edge

As described in Chapter3.1, when structured grids are applied, it is difficult to handle complex shapes, or to preserve the structured nature of grid when doing local grid refinement. In the specific case of the cellular wall, surface geometry is relatively simple, but the slope of the upper boundary also brings the problem of local grid refinement:

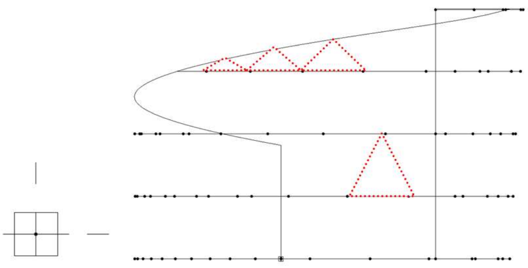


Fig. 6.6 changed grid sizes at the upper edge

The objective of this section is to generate almost equally distributed grids (with the same grid-size). Thus, solutions for the grid generation near the upper boundary should be found, to smoothly generate the whole grid pattern, but not to bring very notable/sudden change in local area.

Optional solution 1 is to give each cutting curve a slope, gradually incline to match the surface boundary shape; Disadvantage is the grid-sizes will be various along the length, smaller in the lower edge side and low density in the higher edge side.

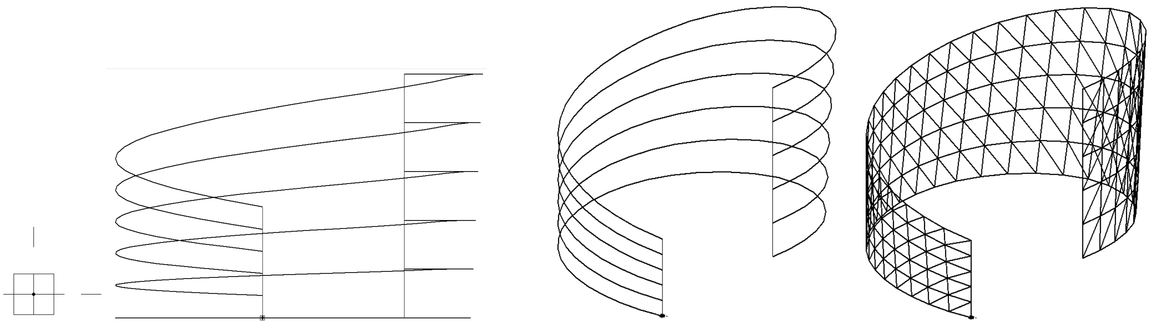


Fig. 6.7 structured grid by iso-curves (horizontal)

Optional solution 2 is to continue the structured grids, and trim them by upper boundary curve. Disadvantages of this solution is that some small and unstructured elements will occur, which are not favorable for structural and architectural functions.

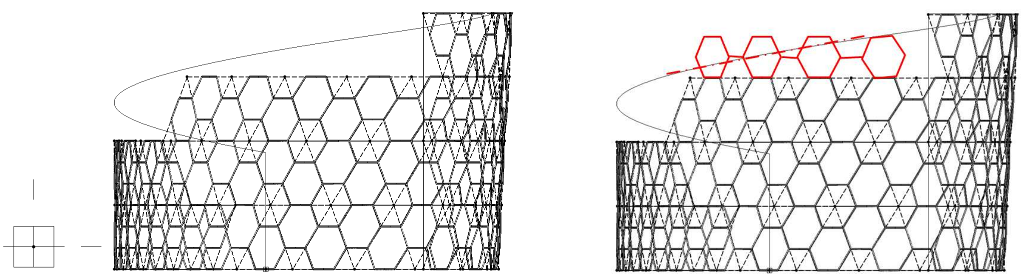


Fig. 6.8 trimming regular grids at upper edge

Implemented solution

The implemented solution is to arrange/move the grid points that are closed to the upper boundary, and then generate unstructured grid locally to fill-in the boundary gaps. Processed as:
Apply an extra checking for the grid points: If (d < grid_size/3) → project them to the upper edge
If (size/3 < d < grid_size/1.5) → move downward | grid_size /3 |
In which, d = the vertical distance of the grid point to the upper boundary
Then continue the grids to the boundary to get a better structural pattern (better cell-sizes and nodal angles).

When introduce some margin of randomness to the grid points, the difference between the grid that close to the edge and the other is even more invisible.

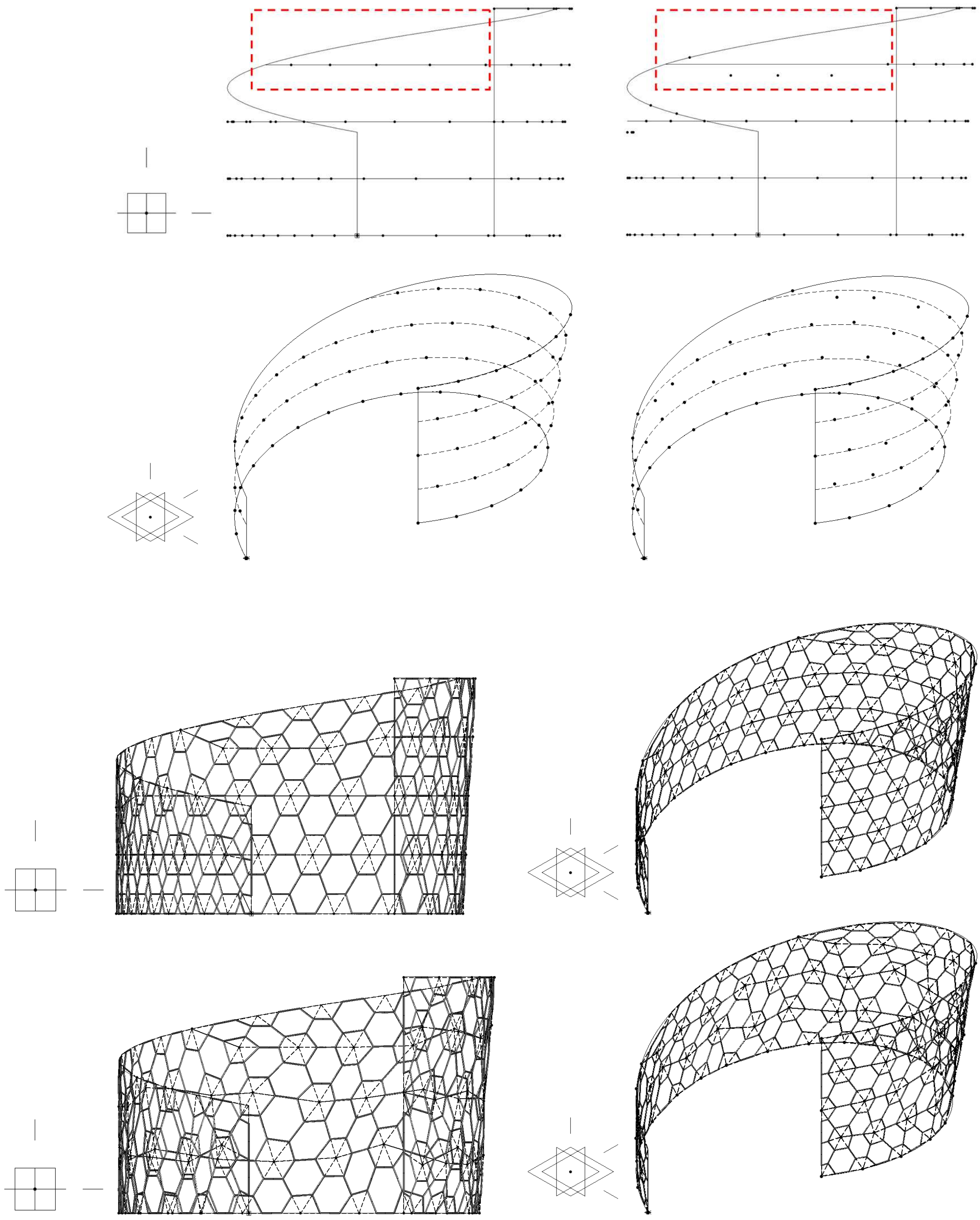
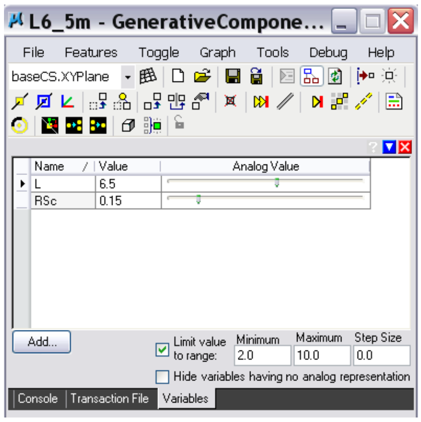


Fig. 6.9 implemented solution for the grid near upper edge

6.2.4 Graph Variables

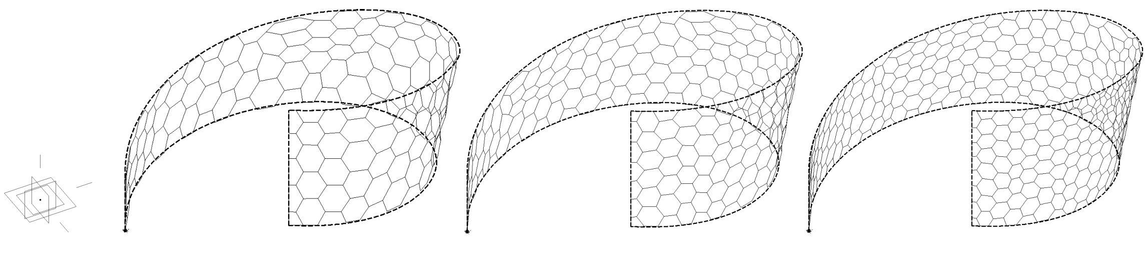
The parameters that can be taken into account in the grids generation are: grid size / grid organization / grid shape; These items can be translated as graph variables, which are defined and controlled in GenerativeComponents:



L - The average size of the triangular grids
Corresponding size of the hexagonal grids is L/3

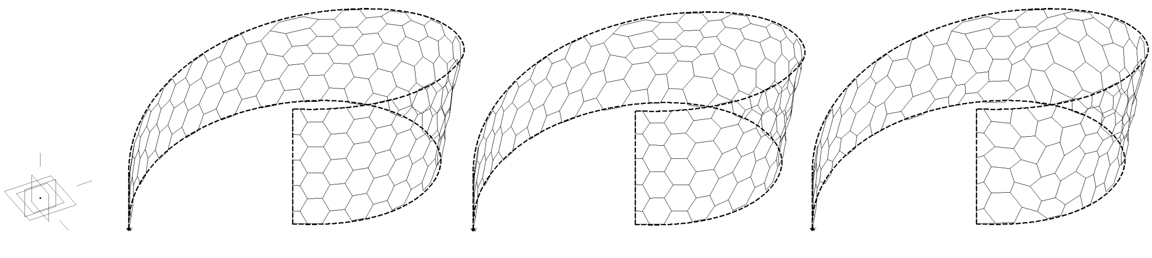
RSc – The scale of the randomness [in %]
This variable was used to control the range of randomness to the grid point movement. Each grid point was given a step size between 0 and RSc*L. [Random (0, RSc*L)]

Case1_ Change L (in which RSc=15%):



L=11m L=8m L=6.5m

Case2 _ Change RSc (in which L=11m):



RSc = 0 (regular) RSc = 15% RSc = 30%

Fig. 6.10 sample models generated by structured grid model

In the structured grid model, when apply graph variable RSc (the Scale of randomness), the grid points are unequally distributed over the surface, this results in:

1. Large differentiations in the lengths of the structure elements and surfaces for the façade elements will occur, which increase the complexity of construction technology and cost.

2. Deformed cell elements, in which some unfavorable nodal angles and cell shapes will occur. So it is not recommended to apply large randomness scale (RSc), in case the hexagonal-cells deform too much.
(The architectural concept design is an irregular pattern to better represent this idea of ‘cellular’, thus the randomness level should also be determined with a wide range of consideration.)

6.2.5 Summary for the structured grid model

Structured grids have advantages of easy to implement and good efficiency (accuracy). The main purpose to create the structured grid model was to have a parametric model which can be easily handled and quickly implemented. With this model, a series of grid sizes can be tested to provide recommended values for the cellular wall structure.

In addition, by create new feature to generate hexagon elements from triangulation, the hexagonal cells are in-plane. This is quite an important point if the structural grid is designed to integrate glass façade. A normal hexagon is not guaranteed to be planar, and the level of out-of-plane of the vertices (supporting points for the glass panels) will influence the application of glass panels.

Because the connectivity rules were simple and organized, the structured grid model was easy to be combined with some solutions for fabrication (Reduce the different elements, for example).

It was also used as the basic model in the member design experiment – assigning different profiles/cross-sections to the structural elements; instead of apply various cell densities in the grid structure.

One notable limitation of the model is that (locally) various grid sizes can't be easily introduced or the grid cells will deform too much. To implement different cell-densities, unstructured grid model will be created in the following sections.

6.3 Modified structured grid models

Based on the structured grid model described in the former section, two trivial strategies were implemented to modify the grid, in order to improve the model for structural functions. This section only described the geometry rules and model generation. Since the grid modification was based on the structural analysis result; more information about these models was integrated in Chapter 8 – Structural Analysis.

6.3.1 Inserting triangle elements

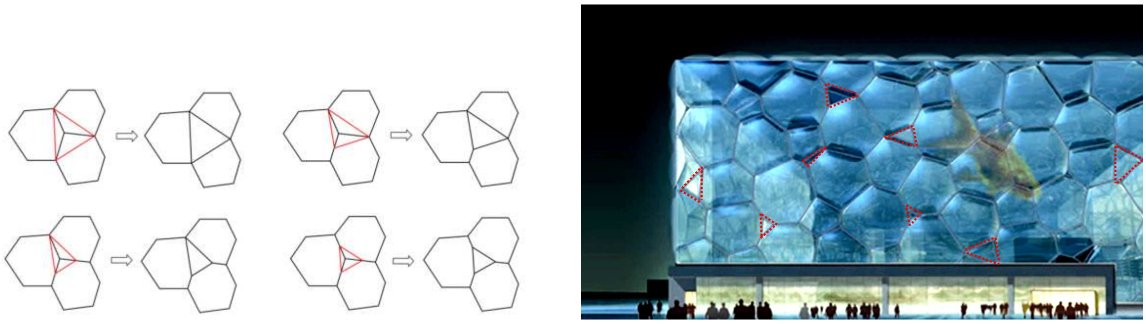
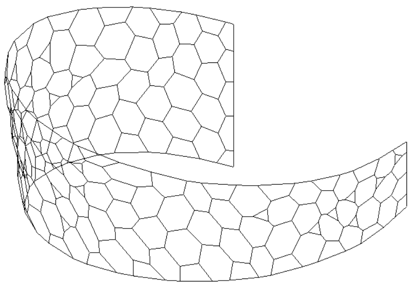


Fig.6.11 inserts triangles by cutting out vertices in hexagon grid (left)
Triangle elements in irregular pattern [Water Cube, Beijing] (right)



[Purpose]
In one hand, all the elements in the structured grid model were defined as hexagons; thus the resulted grid pattern won't have a very irregular shape. By inserting triangle elements, more irregular pattern can be got, to better represent the architectural concept. In another, hexagon grid is not a stiff structure; inserting triangle elements can increase the stiffness of the lattice structure. The inserting is based on local stiffness requirements in the lattice structure.

6.3.2 Locally doubled-up hexagon grid

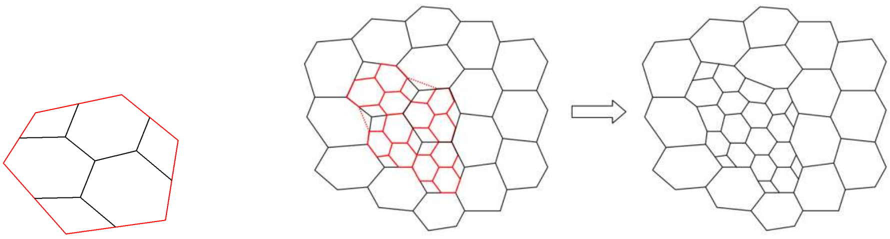
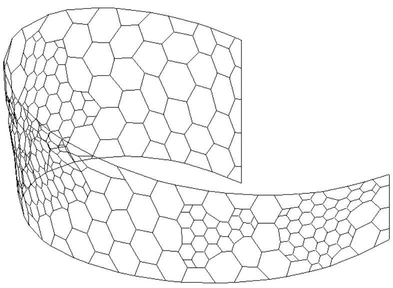


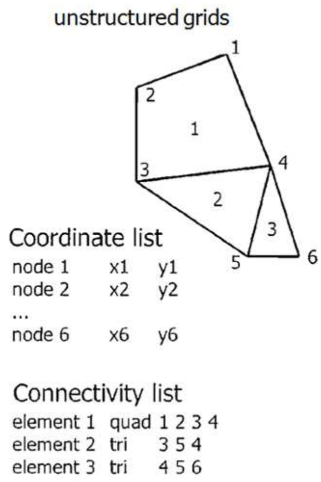
Fig.6.12 new feature type of fractal hexagon (left)
The geometric stragtegy of locally doubled hexagn grid (right)



[Purpose]
A simple application of fractal[7] geometry is to repeat the geometry in a smaller scale by rules. This can be easily achieved by create a new feature type (Fig.6.12, left) and apply it to selected grid elements (Fig.6.12, right).
The resulted grid has changing pattern: where stress was needed, where there was more intensity, the pattern was doubled-up so there was a double rhythm. The 'doubled-up' is based on local strength requirements in the lattice structure.

[7] The word "fractal" was coined by Benoit Mandelbrot in the late 1970's, but object now defined as fractal in form have been known to artists and mathematicians for centuries.

6.4 Unstructured grid models



"A mesh is to be considered an unstructured grid of the connection of the neighboring grid nodes varies from point to point. The connectivity of the grid must be explicitly described by an appropriate data structure procedure. – Liseikin, 1999"

Introduction

Structured grids lack of the flexibility and robustness for handling domains with complicated boundaries, or the grid cells may become to skewed and twist. These can be noted from the structured grid model for the cellular wall (chapter6.2).

With an objective of optimize the grid pattern of the cellular wall, one design alternative was to introduce different grid densities (locally) according to the structural requirements. Executable approach to achieve this in structured grid is to move the grid points. But when the positions of the grid points are changed significantly, the grids will deform too much, since the connectivity is based on the same rule. For this reason, unstructured grid model was introduced.

The implementation of unstructured grid model generation included two parts: **grid points generation** and **mesh generation**. Each part will be discussed and experimented by several trivial solutions in the following sections:

6.4.1 Grid points generation

In this section, several experiments were done in GenerativeComponents and processing[8], to find a suitable approach for adaptive point-set generation. As for unstructured grid, there was no requirement for the organization (listed in sequence) of the grid points. The grid connectivity was implemented separately to the point-set by specific algorithms, as the next step.

In the case of unstructured grid model of the cellular wall, expectation for the point-set includes:

1. Local distances of grid points can be pre-determined freely by designers, and the grid points distribute according to the pre-determined distances;
2. The change of the local distances (presented by cell sizes/densities in the model) in the point-set should be as smoothly as possible;
3. The resulted distribution of the grid points should be suitable for generating equilateral triangle mesh – which can provide good angle between each node and its neighbors to create standard hexagonal pattern (based on triangulation) or 'centralized' Voronoi Diagram (algorithm was described in Chapter 3.1.2), in order to void unfavorable grid elements.

Strategy 1: Circle packing

Apply circle packing (or sphere packing in 3D space) to control the point-point distances during the point-set generation process: every point in the point-set is located in the center of a circle (or sphere), and the point-point distances are defined by the radius of the circles (or spheres).

[8] Processing [Ben Fry and Casey Reas, 2005] is an open source programming language and environment for people who want to program images, animation, and interactions. More information: http://processing.org/

[An Application Example] A grid generation tool[9] by M.H. Toussaint
The purpose of this grid generation tool was to create a grid suitable for gridshell structure - a surface with elements of equal length. Steps of the Algorithm:

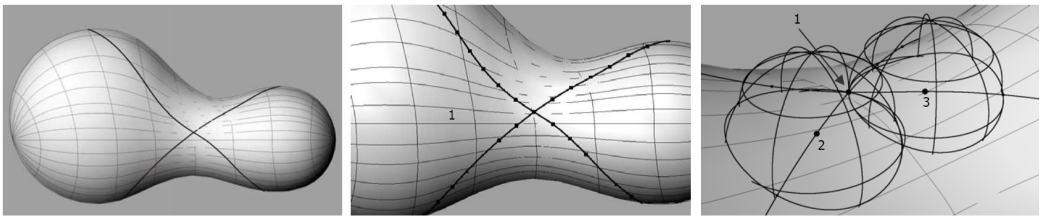


Fig.6.13 Creating sections (left) → sections are divided by length (middle) → Spheres and intersections created (right)

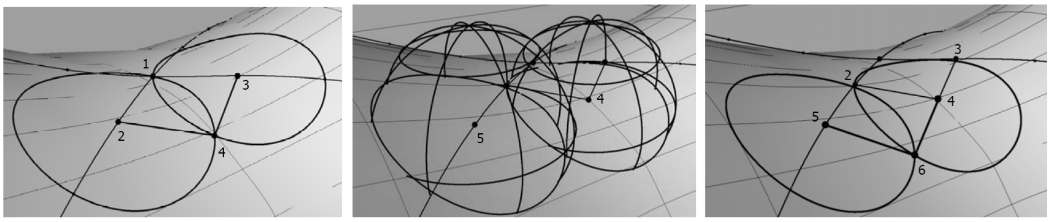


Fig.6.14 Intersection point located (left) → Sphere are created at point 5 and 4 (middle) → point 6 is created at the intersection of the intersection curves (right)

This grid generation tool was aiming to create elements of equal length, thus, each sphere had the same radius. In the following section, several circle packing (2D) experiments will be performed, in which the radius of each circle was determined by the position of its center point in the surface range, and each position was given a predetermined value according to the functional requirements.

The experiments are done in the programmable environment of Processing or GenerativeComponents. Strategies, algorithms and experimental results will be provided in detail, and the Pseudo-code will be provided in Appendix. To be notice, all of the 2D circle-packing experiments are for quickly test the conceptual strategies, no implementation in any actual design cases.

Circle packing 1

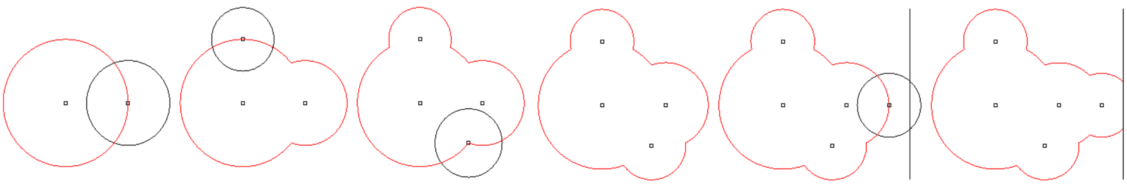


Fig.6.15 Generation procedure

Process of the algorithm:
Create a new point on the effective circumference [**Red**] → take the new generated point as center and its pre-defined value as radius to draw a new circle [**Black**] → combine the former effective circumference and the new circle to get a new effective circumference → create next point on the new effective circumference… The generation continued until the effective circumference was completely out of the boundaries.

9 A Design Tool for Timber Gridshells, MSc Thesis by M.H. Toussaint, 2007, Structural Design Lab, Citg, TUDelft

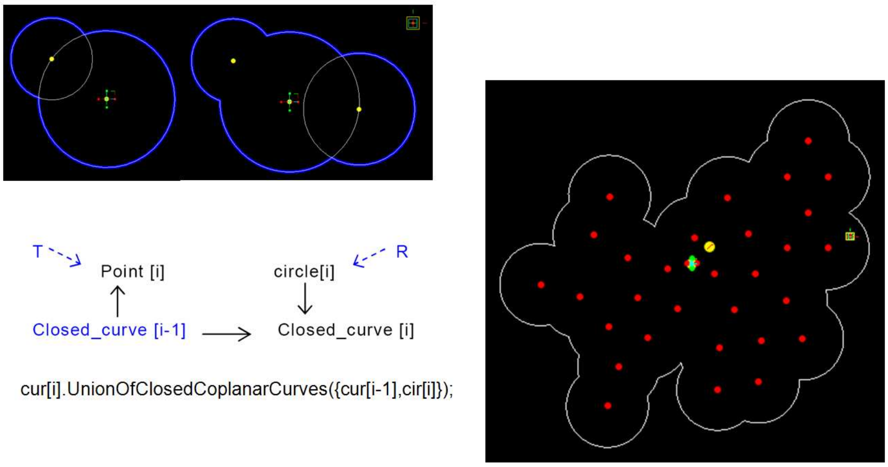


Fig.6.16 Circle packing1 - implementation in GenerativeComponents

A proper definition to the ‘effective circumference’ was very important in this method. One feature called ‘UnionOfClosedCoplanarCurves’ in GenerativeComponents was introduced: it can combine several closed curves and take the outer boundary of them to create a new closed curve [**Blue**]. With help of this feature, it was easy to build up the iteration loop. But it can’t be implemented to many iteration cycles to deal with large amounts of points/curves, because of the memory limitation. This backup feature predefined in GenerativeComponents is a ‘black box’; it can’t be modified by users to solve the problem.

Circle packing 2

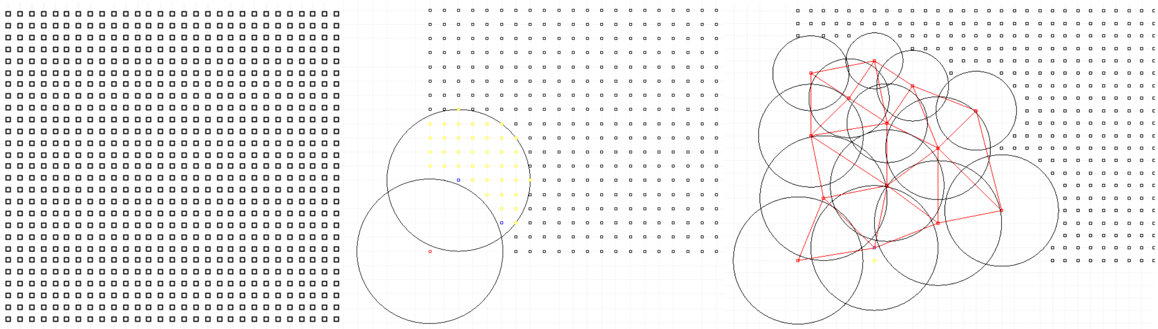


Fig.6.17 Selection and removal procedure

In this case, the rules were the same as the former method, but an inverse procedure was implemented. A large amount of grid points were generated as default grid, and then a series of selection and removal were implemented: select the first point [**Red**] and create a circle according to the same rule described in the previous experiment; default grid points located inside of the circle [**Yellow**] were deleted → select a new point from the closest point(s) [**Blue**] in the rest grid points and draw a new circle … The selection and removal continued until all the default grid points were deleted.

3 point-lists were defined in this method:

1. **default grid[]** – [**Black**] its items will be removed when they are located inside the generated circles, and it will be a null list after all the selection and removal;
2. **final list[]** – [**Red**] items that have been selected from default grid as effective points will be stored in this list, to provide the resulted point-set after iteration;

3. **closest point(s)[] –** [**Blue**] every time when a new point is selected and a corresponding circle is created, a group of closest points (closest to the new selected point) will be updated, and the next selection will be from this group

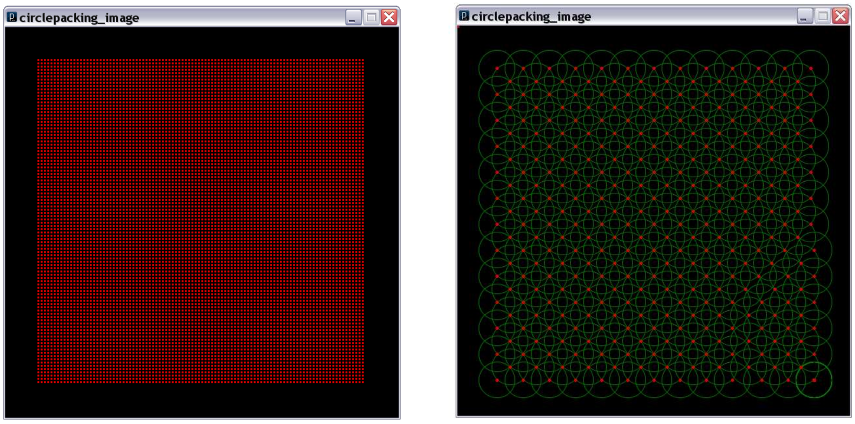


Fig.6.18 Circle packing2 – implementation 1 in processing

In this case, each position was given the same value for local radius and the resulted grid points should be regular/ equally distributed.
The default grid points before iteration (left), and the final points after all the selection and removal (right);

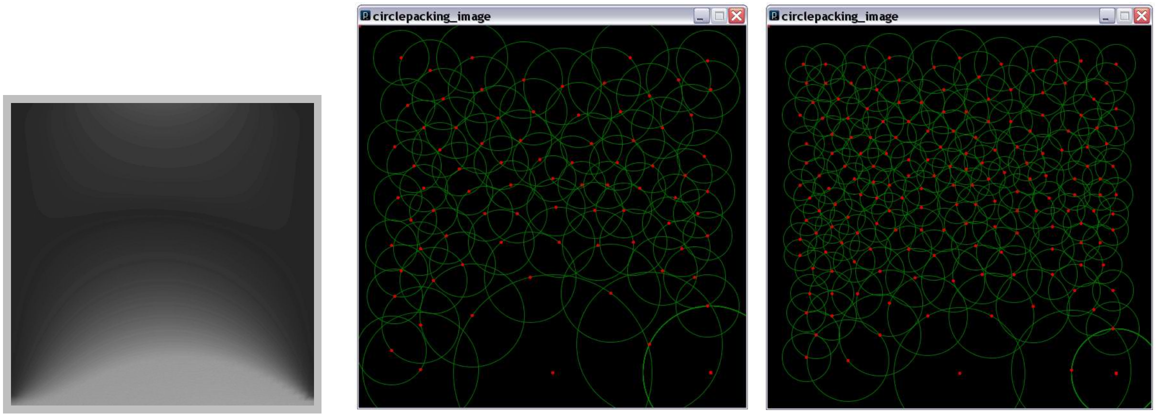


Fig.6.19 Circle packing2 - implementation 2 in processing

In this case, the local distances were determined by color values – the final point-set distribution was corresponding to the color gradient. A gray image (left) was used as background; color value of each position in this image was get and signed to the local distance (radius). The rule between color value and radius can be defined freely, to get different grid densities (right).

Summary of circle packing strategy

The circle packing approach provided a grid point distribution that represented the required density distribution quite well, but the algorithms was quite slow – although some recursions to organize the data structure have been introduced.

Another drawback was that the local distance can be only controlled in one side of the circles – follow the direction of the generation [Fig.6.20]. Thus, the generation should start from the places with large distances (radius) to get a better result.

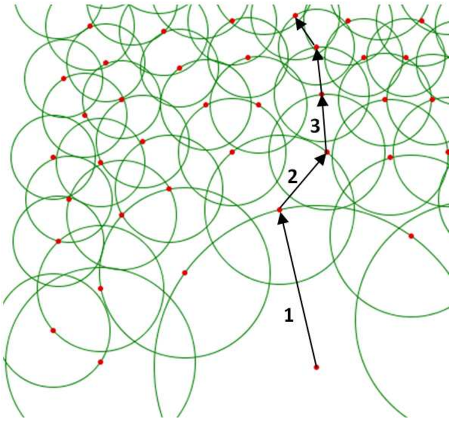


Fig.6.20 a sample direction of the generation and distance control

Implemented solution: Attract and repel

The implemented strategy was to apply a virtual spring system, by defining attract and repel actions between the grid points. The position of each grid point was settled down when all the grid points in the system was equilibrium.

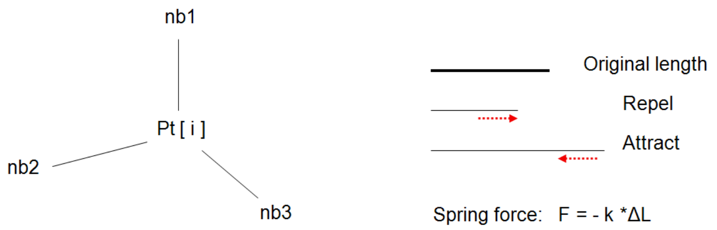
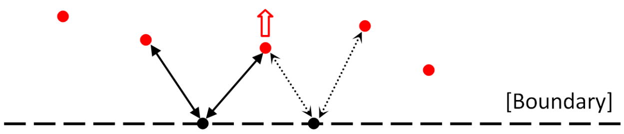


Fig.6.21 the rules in attract and repel strategy

Grid points were defined as 'vector agents'[10], and their behaviors were determined according to an established set of rules. Here all the rules were based on 3d vectors: Each point had actions and reactions with its 3 closest neighbors. It might repel them if their distance was smaller than a predetermined value or do the opposite. The sum of the 3 forces (vectors) to the agent determines its movement. Magnitudes of the forces were defined as springs $F = - k*\Delta L = - k*(d-d_0)$, in which, d_0 was the predetermined value for local distance, the same as the circle packing strategy.

[Note] Agents were defined by two categories: **points on the boundaries** and **points inside**
The points on the boundaries were stay still; they looked for 2 closest points inside to add forces on: the acted points inside the boundary then moved according to these forces .



The points inside the boundaries kept moving until equilibrium. There might be two kinds of forces acted on each of them: one from its 3 closest neighbors (inside points) and the other from the boundary points – if fits the description in category 1.

Boundary checking: during the iteration and movement, a boundary checking was implemented to make sure all the points stay within the boundaries. If one point went out of the boundaries, it would be given a random place inside, and then be involved in the system again.

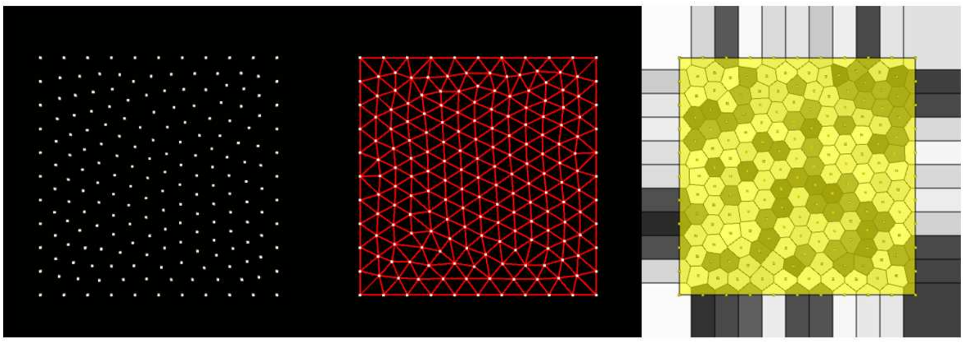


Fig.6.22 attract and repel - implementation in processing

[10] Agents can be thought as small algorithms based on a set of rules whereby different reactions can be simulated when encountering different situations. There are two main types of agents: dumb agent, which is the one used in this experiment and intelligent agent. The first one behaves according the some rules and cannot adjust them, the second one is able to learn from the environment in which is placed and infer decisions.

Disadvantages

In this strategy, the number of grid points within the boundary was an input value. Estimation and trivial-runs should be done to provide a suitable value. By doing this, situations that grid points were not enough to fully fill-in the area (Fig.6.23 left), or too much to represent the predetermined distances (Fig.6.23 right – points were pressed in tank) can be prevented.

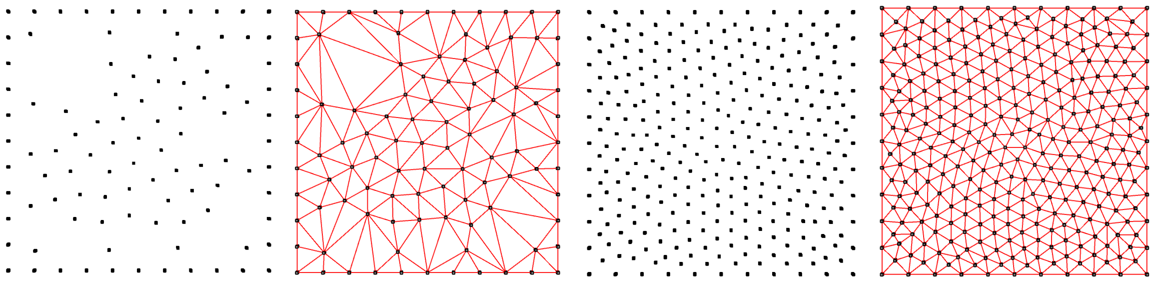


Fig.6.23 bad situations by attract and repel strategy

Advantages

1. No need to deal with complex topology – the mathematics of the algorithm was simple.
   The required inputs were: the number of grid points within the boundaries, the points distributed on the edges, and the local distances. The agents (grid points) will find the suitable places to stay by gaining equilibrium.

2. And the resulted distribution of the grid point was suitable for generate: a) Equilateral triangle mesh – which provided a good angle for the nodes in hexagonal pattern or b) centralized Voronoi Diagram

3. Easy to implement parametric tests;
   Adaptation of the grid is difficult in unstructured grids, since the representation is not parametric. Adaptation is done by hand, by removing points, inserting points, displacing points. Because the local organization in the grid is not bound, but can be redefined in the data structured, local refinement or coarsening is possible (as exampled in Fig.6.24). However, it is not easy to control the grid size and to get good grid shapes.

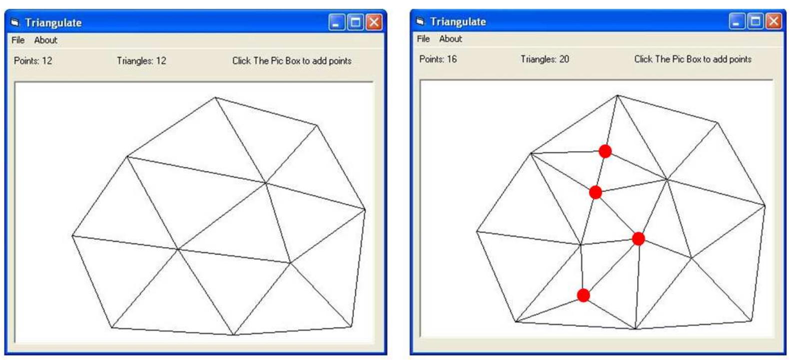
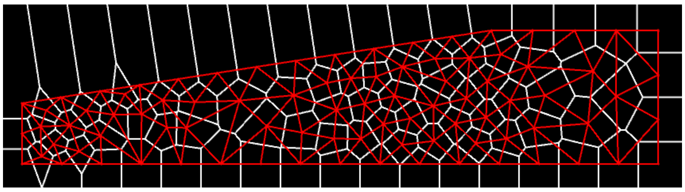


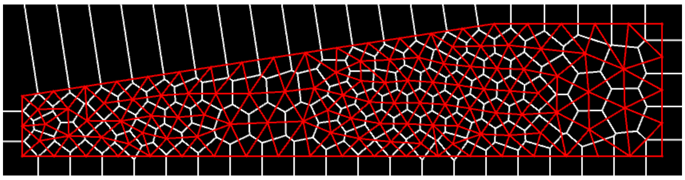
Fig.6.24 local refinement by inserting points in Triangulation Tool

By the strategy of attract and repel, parametric information can be introduced in the predetermined values, and regeneration can be quickly executed. It provides freedom to change grid densities, and the change from one density to another is relatively smooth.

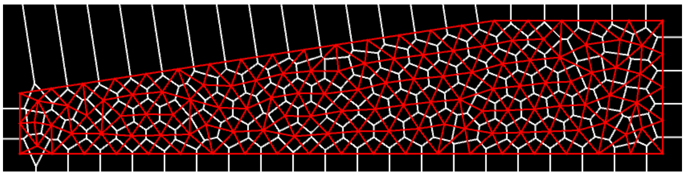
Input_1:



1_Various densities – un-efficient generation

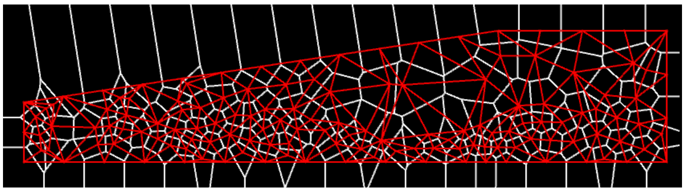


2_Various densities – fine tuned-up

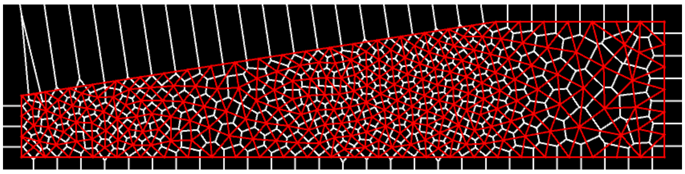


3_The same density - regular grid size

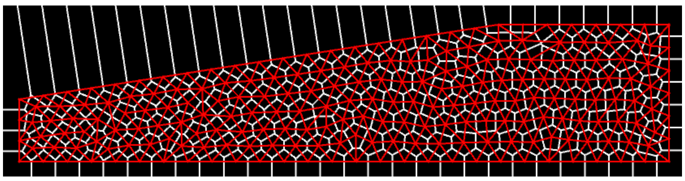
Input_2:



1_Various densities – un-efficient generation



2_Various densities – fine tuned-up



3_The same density - regular grid size

Two examples of the “attract & repel” strategy are shown in Fig.6.28.
Red: Delaunay triangulation
White: Voronoi Diagram

In this generation tool, user can pre-determine the local grid size – by grid point distances (as the original length of the virtual spring between every two grid points).

It should be noticed that when various (local) densities are applied, the amount of grid points must be properly defined in the input – by estimation or trivial generations, otherwise it will result in an “un-efficient generation”.

The iteration process is fast and visualized. By creating Voronoi Diagram for mesh, it looks like the cell-generation in nature.

“The purity of structures is generally compromised by the variety of different loads they must endure and the multiplicity of functions that are demanded of a building structure. The art in selecting structure is in recognizing how and when the designer can impose his wishes with a minimum of compromise. Yet, compared with nature, we humans are still in our infancy when it comes to mastering the combination of structural actions, a single orchid flower probably contains more variety and subtlety of structural actions than the most remarkable building; and natural structures have a factor of safety of very nearly one.

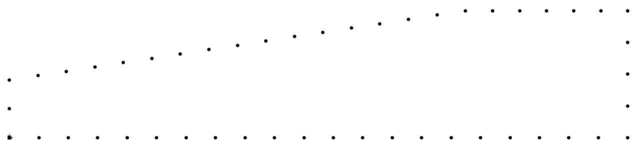
– The Art of the Structural Engineer, Bill Addis”

Fig.6.28 Pattern Generation [Tool: processing] - Animation Record during iteration process

Implementation in GenerativeComponents

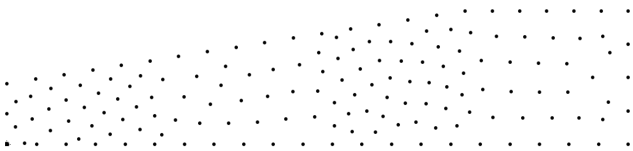
The grid point generation program by processing was translated into GCScript, to execute the same procedure in GenerativeComponents. An extra input - the number of iteration cycles was introduced. No graph update during the iteration process, in order to speed up the iteration process.

1. Generate points on boundaries
   The number of points and their distribution along the boundaries can be determined freely. In this example, boundary points were distributed uniformly:

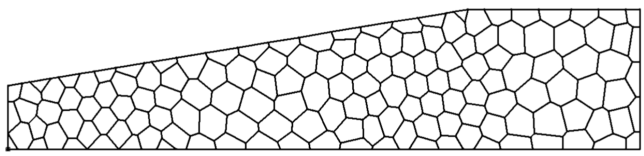


2. Generate random points within the boundaries
   The number of points should be estimate by the predetermined local distances.

3. Run the iterations – attract and repel
   Resulted grid points distribution was corresponding to the predetermined rules – in this example, the whole area was divided into 4 regions, and each region was give a specific distance value:



4. Generate mesh (in this example, Voronoi Diagram) based on the resulted grid points

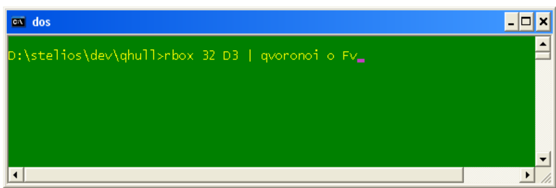


6.4.2 Mesh generation

Grid generation technology has been described in chapter 3.1.2. In this section, mesh was generated for unstructured grid points with non-uniformly distribution. To implement this, the algorithm of Delaunay triangulation and Voronoi Diagram were introduced.

Since Delaunay triangulation and Voronoi Diagram have popular application in various fields, a lot of open sources can be found. The source used here was Qhull[11].

Qhull works in the dos console, and runs very fast. The required input is a custom point coordinate file (.txt). Implementation of the algorithm is done by dos commands. An output file (.txt) with all the information to draw Delaunay triangulation or Voronoi Diagram will be created.



[11] Qhull is open source software for calculating convex hulls, Delaunay triangulations, Voronoi diagrams etc. http://www.qhull.org/

In order to linking grid generation (attract and repel model in GenerativeComponents) to mesh generation, an interface between Qhull and GenerativeComponents was developed. The interface was based on 'rcqhull'[12] plug-in by Robert Cervellione, 2008. The 'rcqhull' feature was written in C# and integrated into GenerativeComponents as a dynamic link library (.dll) file.

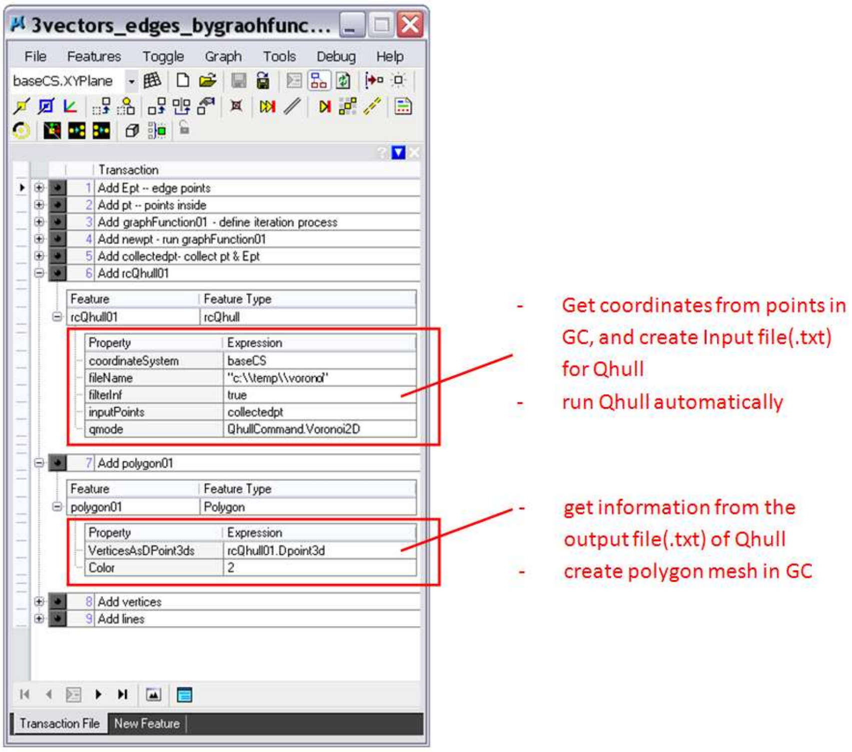


Fig.6.29 Implement rcqhull in the attract and repel model

[Note] The mesh directly generated by the 'rcqhull' plug-in was in a geometry format of polygon, which is not compatible in structural analysis software (by .dxf file). It was then regenerated by create lines by the vertices of each polygon.

6.4.3 UV Mapping

In previous description, the grid point and mesh (grid) generation has created a pattern on 2D plane. The next step is to the 2D pattern onto 3D surface.

Surface analysis

In rough representation, curves can be viewed as a connected one-dimensional series of points. Similarly, surfaces can be considered as type of two-dimensional skin in space. Surfaces are constructed by series of curves. Analogous to these curves we can introduce parametric, explicit, and implicit representations of surfaces for mathematically handling surfaces and studying their geometry analytically. A parametric representation of surfaces is helpful for mapping procedure – transform elements from 2D plane to surface in 3D space.

---

[12] The 'rcqhull' plug-in for GenerativeComponents was originally built by Robert Cervellione, 2008. More information can be found on his website: www.cerve.org. The author had the permission from Robert Cervellione to use and modify the plug-in and source codes in this thesis research.

Parametric representation

The coordinates of a surface point depend on two different parameters u and v. thus a parametric surface S can be represented by p(u,v) =(x(u,v),y(u,v),z(u,v))), where the parameters u and v assume all values in a two-dimensional region R. Then we can have a continuous mapping of a two-dimensional region into space. Fig.6.30 Describes a mapping from a region R of the (u,v) parameter plane to a surface patch S in three-dimensional space.

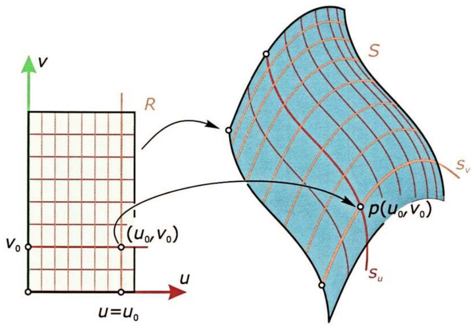


Fig.6.30 the parametric representations
Source: Architecture Geometry–Bentley institute press

Unused strategies - Follow along the surface

Follow along the surface was a strategy to include the UV mapping in the 'attract and repel' grid point generation. In a 3D space, the vectors were also defined in 3D. Thus, when the agents (points) were moving by unbalanced forces (vectors) from their neighbors, their movement can't be restricted on the surface. An extra checking was made, similar as the boundary checking in 2D, to make sure the agents stay on the surface:

Case 1: When a surface can be defined by equations, the boundary checking can be applied to make sure the coordinates of each grid point fulfill the equations of the surface.

Tested surface: Monkey Saddle
Equations of the surface is given by $x = u; y = v; z = u^3 - 3uv^2$ $(-1 <= u <= 1, -1 <= v <= 1)$

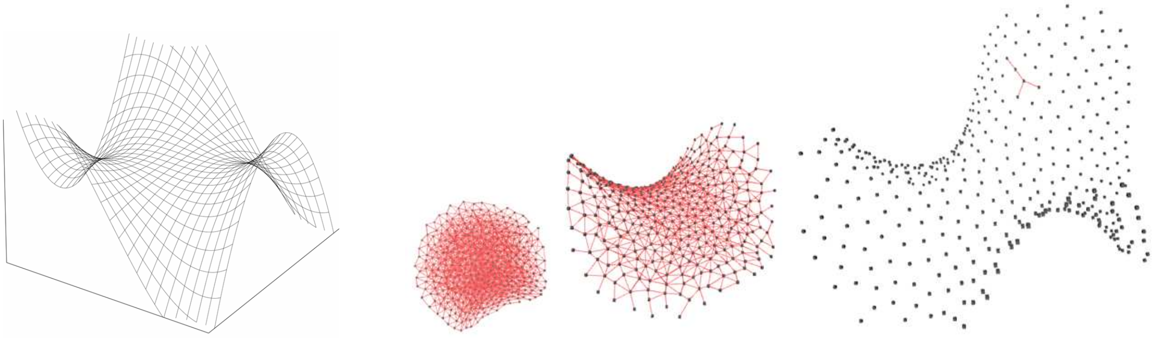
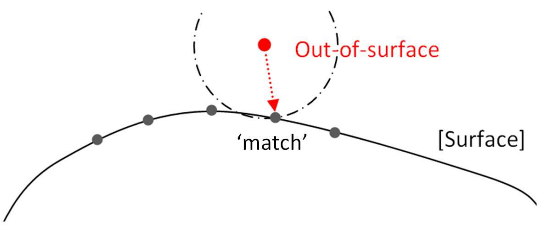


Fig.6.31 generation procedure in Processing
The iteration process: from left to right (in this experiment, the local distances are set to a same value). Each red line represents two points that have reaction with each other; when all the points are under equilibrium, no red lines is showed.

Case 2: When the surface cannot be formulated, the solution is to apply a search function to find a match for the out-of-surface point (a 'match' is a closest surface grid to this point), and then push it off to this match's place to remap it back onto the surface:



To create unstructured grid for a point-set along certain surface requires a 'surface guide'. As illustrated in the example (Fig.6.32): in the case of null guide, the point-set was projected to a basic plane to generate a normal 2D mesh(left); given a surface guidance, the algorithm of grid generation was implemented in the UV surface region, to create a mesh tilled on surface.

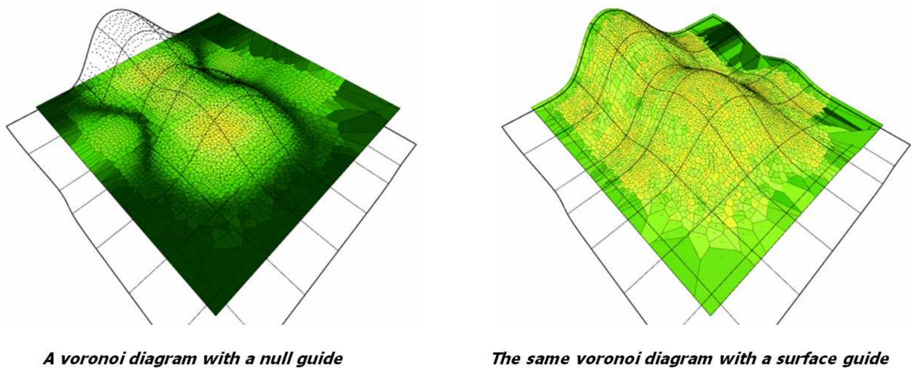


Fig.6.32 Voronoi diagram of a same point-set, without and with surface guide
[Source: PSR (PointSet Reconstruction)[13] plug-in in Rhinoceros]

The problems of this strategy are: 1. Define the density rules directly on surface is more complex than in two-dimension (a normal coordinate) and implementation of the point-set generation is very slow; 2. To combine the 'surface-guide' rules in the 'rcqhull' plug-in requires a new C# script for 'rcqhull' feature.

Implemented strategy - UVMapping Tool

In a 2D model, the grid distribution can be viewed clearly, to better define the grid densities (locally). Once the pattern is transformed onto the surface in 3D space, it is not easy to visually evaluate the pattern. Thus, the implementation strategy was to finish the grid generation on plane and then map the grid onto surface.

Algorithm of the surface mapping

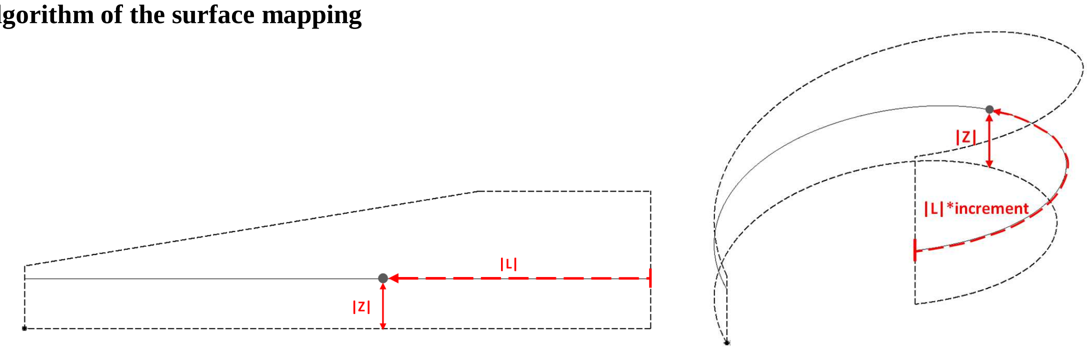


Fig.6.33 mapping of a two-dimensional point into space

Fig.6.33 shows the corresponding points in-plane(left) and on the wall surface(right). The implementation process in GenerativeComponents:
1. Get the information of original point (|z|, |L|);
2. Create a horizontal cutting-curve on wall surface according to |z|.
3. Generate the new point on cutting-curve according to the calculated length |L|*increment.

Introduction of the 'increment' was due to the incline slope of the wall surface. The surface was inclined outward, thus, the horizontal circumferences along the surface height were increased.

[13] PSR (PointSet Reconstruction) for Rhinoceros is the process of creating 'higher level' geometry from ordinary points. http://en.wiki.mcneel.com/default.aspx/McNeel/PointsetReconstruction.html

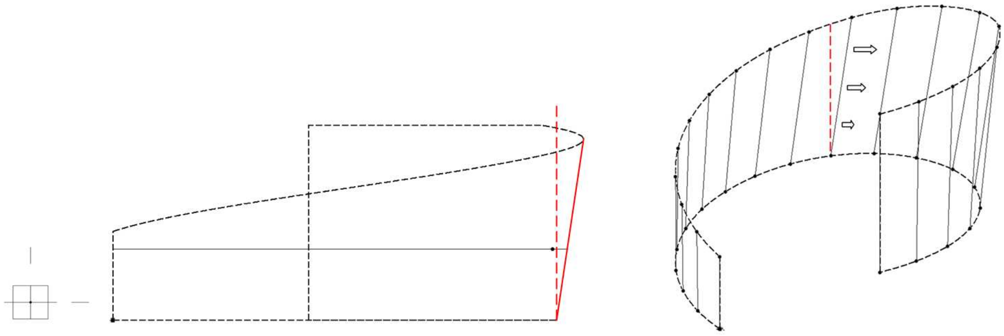


Fig.6.34 increment due to the incline slope

Solution for the incline slope is to discrete the surface into vertical strings, and stretch these strings along the circumference (Fig.6.34, right). The approach is suitable for Single curved surface (developable). Thus, each point has an increment value according to its height on the wall surface.

Calculation for the items showed in Fig.6.34:
At the height of |z|, the total increment of the circumference ('elongation') is:

$$L_{Scaled} = L_{inner}+(L_{Outer}- L_{Inner}) *(|z|/H_{max})$$

For each point on this cutting-curve (|z|), the transformation procliple is:

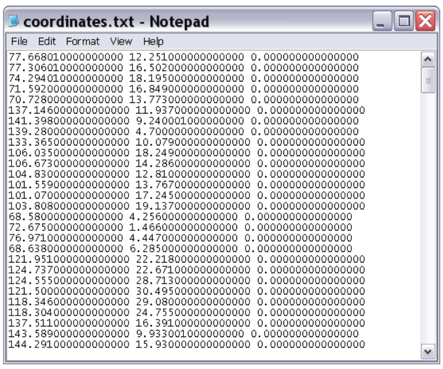
$$|L|*\mathbf{increment} = L_{scaled}*(|L|/ L_{Inner})$$

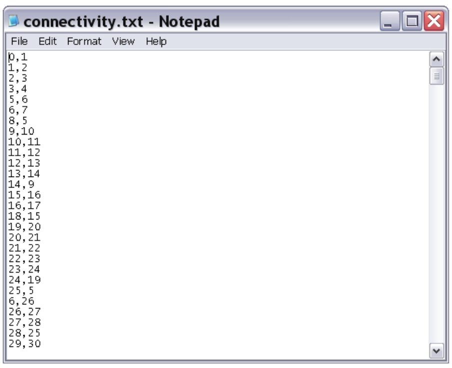
(L_{inner} and L_{Outer} are the lengths of the bottom and top lofted curves projected to horizontal plane.)

The implementation of Mapping Tool

Step 1. Regenerate the 2D grid in GenerativeComponents

As the theory of unstructured grid indicates, to describe an unstructured grid, two files for appropriate data structure have to be provided: Nodal coordinate list and Connectivity list.





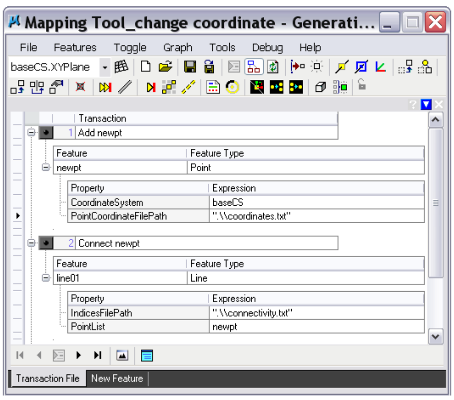


Fig.6.35 the format of the prepared .txt files and regeneration in GenerativeComponents

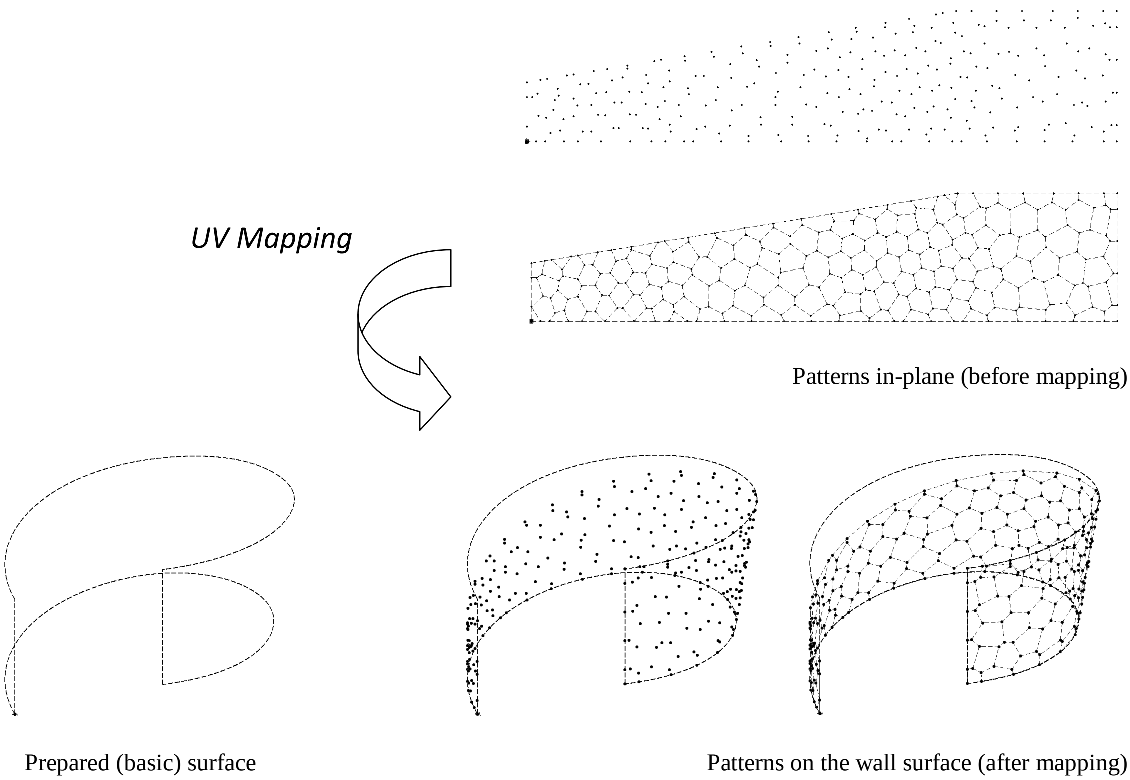


Fig.6.36 grid patterns before & after mapping

Step 2. Apply UVMapping to create new grid points on wall surface

The process and calculation has been described before: height of the original point → horizontal cutting-curve on wall surface → length along cutting-curve, including increment (stretch) → generate new point on the proper position; The resulted pattern was showed in Fig.6.36.

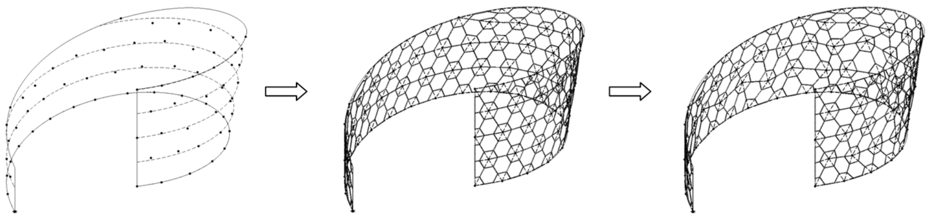
Step 3. Export coordinates

The output of the mapping tool is a coordinate file (.txt or .xls) of the new generated points on wall surface. Overwrite the Nodal coordinate list by this output file, rebuild the grid by original Connectivity list – by doing this, 3D grid can be get. (Note: there is no change to the sequence of grid points in the Nodal coordinate list. Thus, the same connectivity list can be applied.)

6.5 Summary

This chapter, several parametric models were described:

1. **Structured grid model**



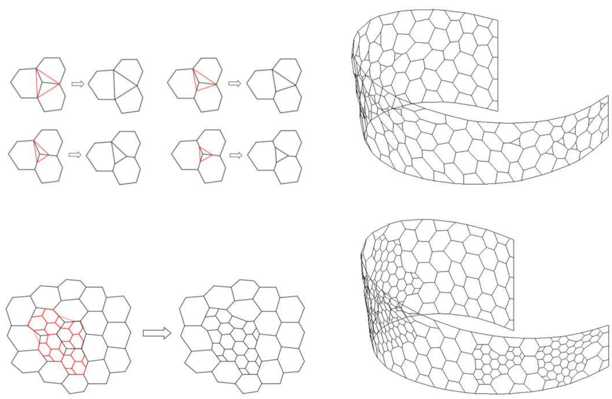
The structured grid model was based on regular grids, and a solution was implemented to solve the continuity at upper boundary. The main parametric variables were: L – grid size, represented by the average length of the triangle edges; RSc – randomness level, represented by the movement of the grid point.

Structured grids have advantages of easy to implement and by create new feature to generate hexagon elements from triangulation, the hexagon cells are in-plane. Limitation of the model is that (locally) various grid sizes can't be easily introduced.

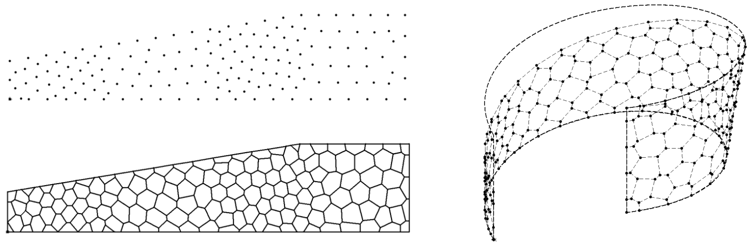
2. **Modified structured grid model**

- Insert triangles to get more irregular and stiffer pattern; The inserting is based on local stiffness requirements in the lattice structure.

- 'doubled-up' according to local strength requirements in the lattice structure: where stress was needed, where there was more intensity, the pattern was doubled-up so there was a double rhythm.



3. **Unstructured grid model**



Unstructured grid model has flexible distribution by pre-determined local distances;
Process: - generate points on the boundaries (arranged) and inside (randomly)
- determine the local distances (mainly by divide the region into several parts)
- apply the attract & repel iteration to redistribute the points inside
- after generation in 2D, mapping the pattern onto the wall surface

(Note: there is no change to the points on boundaries, only the points inside will redistribute.)

Chapter 7

Structural Analyses and Evaluation

Chapter 6 has give description for several generation strategies and the resulting parametric models. In this chapter, series of these parametric models will be analysis and evaluated. Analysis and Evaluation of the design alternatives should be multi-criteria, but in this section, the criteria will be limited within structural requirements, to explore structural efficiency.

As discussed in Chapter 4, building up a design in a parametric associative way, each design step might require a design tool. In this chapter, several computational tools have been built to aid the parametric analysis. Description of these tools will be included in the steps of analysis procedure.

With the parametric models described in the former chapter, this chapter will focus on the structural analysis and evaluation of these parametric models. The following design stages can be split up in two main parts, the structural analysis and the evaluation of the designed structure.

Tool Aided Design

Parametric design process, the model is defined parametrically and a series of models will be analyzed and evaluated. Thus, some tools are needed for aiding the parametric analysis and evaluation process.

The first stage - structural analysis will be executed by GSA[14]. For the parametric associative process, all data required for a finite element analysis needs to be generated an exported to GSA. Besides the structural geometry, information about connections, supports, sections, loads and load combinations need to be provided. This information is not easily generated by GenerativeComponents. The choice has been made to generate the required data for finite element analysis by spreadsheet software package, Microsoft's Excel. Several tools were created with three software packages [GenerativeComponents, Excel and GSA][15], to generate and deliver data during the structural analysis procedure. These tools will be described in different stages.

7.1 Geometry Export & Import

The first step of structural analysis is to interface modeling and structural analysis software packages.

With the parametric models and tools described in the previous section, the generated models can be exported as CAD drawing format. But in most of the cases, .dxf/.dwg file format is not favorable in structural analysis software package.

For analyzing the structure in GSA, 2 sets of geometrical data are needed, similar as the implementation of unstructured gird model construction. The first set is coordinates list of the nodes and the second is the connectivity of the structural elements. The implemented strategy is to construct the structure by points and lines in GenerativeComponents and all the geometry data is written to Excel worksheet. Restructure with Excel, the geometry can be neatly built in GSA.

Different geometry export & import methods were applied in different groups of parametric models. For the structured grid models, coordinates and connectivity files are easy to be built, due to the structured data.

[Main Problems]

- Remove the duplicated points/lines (nodes/elements)

In most of the cases, some components are overlap – with the same definition; In GenerativeComponents functions, an array/a list can be rearranged by removing duplicated items.

- Dynamic range in Excel when writing the data to spreadsheets

Parametric models with flexible data structure require for adjustable ranges. For writing data to excel, GenerativeComponents feature ExcelRange was use:

```
WorkbookFileName   = ".\\geometry.xlsx";
SheetName          = " nodes ";
RangeAddress       = "A2:A"+ToString(gridpoint.Count);     // dynamic range
Value              = Flatten(gridpoint.X);     // change 2D array into single array
```

---

[14] GSA is the in-house finite element software package of ARUP [Oasys,2006]

[15] GSA has the ability to make use of 'COM Export Functions' to communicate with Excel and vice versa. There functions are driven by VBA code; GenerativeComponents can also communicate with Excel via features 'WriteExcelRange' and 'ReadExcelRange'.

7.2 Section Property Definition

The choice has been made to apply rectangular hollow section steel beams in the lattice structure. Section profiles were defined as STD RHS (Standard Rectangular Hollow Sections) with isotropic properties.
User's definition: Depth/width (D)
[side/top and bottom] wall thickness (t)

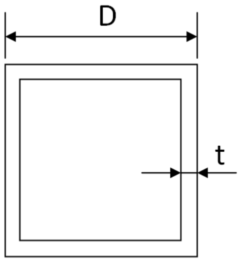


Fig.7.1. Definition of the section profile

In this stage, a uniform profile (with the same wall thickness) was assigned to all the beam elements; but different profile dimensions were analyzed by changing the section properties totally within the Lattice structure. In chapter 7.6 Member Design, parametric section definition will introduced for assigning different cross-sections (with different wall thickness) to the beam elements. As mentioned in chapter 5.4 Member/local buckling problem, too slender section should be avoided.

Material: In GSA, several materials are predefined. These standard materials can be called in the definition of the section properties and the characteristics of the material are used in the structural analysis. Standard steel was used for the cellular wall structure.

7.3 Boundary Condition Definition

Three different restrains were applies in the cellular wall structure:

1 _ Foundation
The Lattice structure was set to be fixed to foundation;
Fixed: All (Fx, Fy, Fz, Mx, My, Mz)

2 _ Side edges
Side edges were boundaries of the underground part, providing horizontal supporting for the cellular wall.
Fixed: Fx, Fy
Free: Fz, Mx, My, Mz

3 _ Green roof
Moment fixed joint will require for large amount of steel, combining with the reinforcement of the roof. Furthermore, if the roof is pre-tension, extra moments from the roof will add to the cellular wall, that is an unfavorable situation.
Therefore, a simply pin connection was chosen for the roof. This connection was assumed to be rigid enough, thus the roof can provide fully horizontal support. Reaction forces will be checked for the roof connection.
Fixed: Fx, Fy
Free: Fz ,Mx, My, Mz

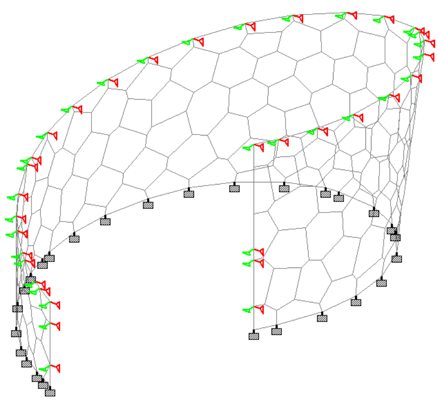


Fig.7.2. Definition of boundary conditions

Note: Intermediate supports/loads from the local floors were not taken into account in the structural analysis.

7.4 Load definition

Three categories of main loading were taken into account in the structural analysis: the dead load of the structure, the resulted load from the green roof and the resulted load from the façade. The three defined loads all depends on the geometry of the structure and will be driven by the geometrical data. The load definition was estimated according to Chinese Code [GB 50009—2001] - Load code for the design of building structures.

Table 7.1 Applied load cases

	Dead Load	**Live Load**
Wall	L1 [Self-weight]	
Roof	L2 [Roof-weight]	L3 [live load]
Façade	L4 [glass-weight]	L5 [wind load]

Combination:
SLS (Serviceability Limit State) L1+L2+L3+L4+L5
ULS (Ultimate Limit State) 1.2L1+1.2L2+1.35L3+1.2L4+1.35L5
(Safety factor for dead load and live load are 1.2 and 1.35.)

Load Case 1 - Self-weight

The dead weight of the lattice structure depends on the chosen sections and the geometry of the structure. To calculate the weight of a structural element, the following data is needed: the cross-section area, the length, and the density of the material. For each individual element, all of this data is available, the load as result of the own weight of the structure can be explicitly calculated. Since all the input data for the determination of the own weight of the structure is known in GSA, it is simply implemented by apply 'Gravity Load' to all the elements [COM Export Function: LOAD_GRAVITY | all | 1 | 0 | 0 | -1].

Load Case 2, 3 - Roof load

The weight of the green roof is calculated according to the concept design of the ramp and roof structure:

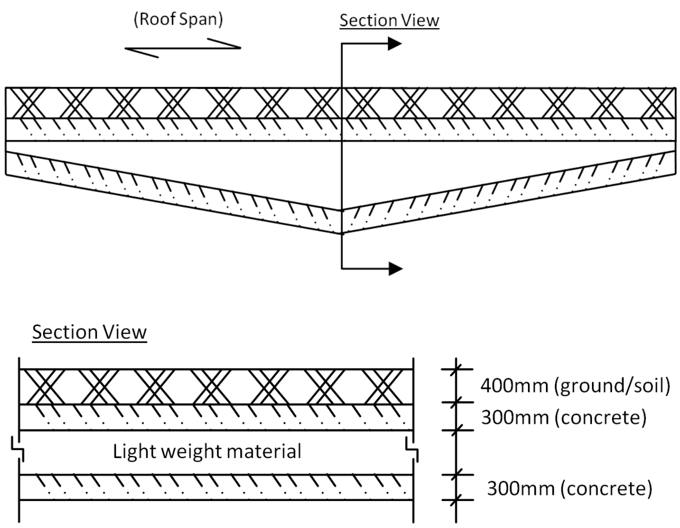


Fig.7.3. the concept design of the ramp and roof structure

The weight of the green roof $Q_d = 15 + 1 + 8 = 24kN/m^2$

In which, estimated load for Top ground/soil = $8kN/m^2$

Light weight material = $1kN/m^2$

Concrete layers = $15kN/m^2$

Live load for the green roof $Q_l = 3kN/m^2$

For the nodes on the upper edge:

Per node: $V = Q \times Area[kN]$

Area is the loaded area for each node on the upper edge $[m^2]$

LC2 $\quad V_d = \frac{1}{2} \times 24 \times Area = 12 \times Area[kN]$

LC3 $\quad V_l = \frac{1}{2} \times 3 \times Area = 1.5 \times Area[kN]$

The loaded area for each node was driven by the geometrical data, which includes the distance between the node and its neighbors, and the roof span of the nodal position on upper boundary. In a parametric analysis process, for each cellular wall model, the nodes on the upper boundary are different located; and for each location, the roof span is different. So does the distance of nodal distances. Thus the corresponding loaded area for each node needed to be recalculated. Because of the irregular shape of the green roof, calculation for the loaded area is inconvenient.

A Roof Load calculation tool was created to define the nodal loads resulted from the green roof. The strategy of this tool consists of two components: the first one uses GenerativeComponents to calculate the corresponding loaded area for each node on the upper edge and write the geometry information to Excel. The second component uses Excel to calculate the nodal loads according to the areas that have been calculated in GenerativeComponents.

Implement of the tool

Step1- preparation in GC

According to the roof shape and the setting of interior columns (concept design), the area of the green roof, which will put loads on the shear walls (including the cellular wall and the solid wall in the other side of the roof span) can be defined. The defined shape was then regenerated in GenerativeComponents:

- Cut out the defined area in floor plane [Rhinoceros]
- Get coordinates information of curves' control points [Rhinoceros – create point file]
- Generate the boundary/shape curves [GenerativeComponents.BSplindeCurve.byPoles]

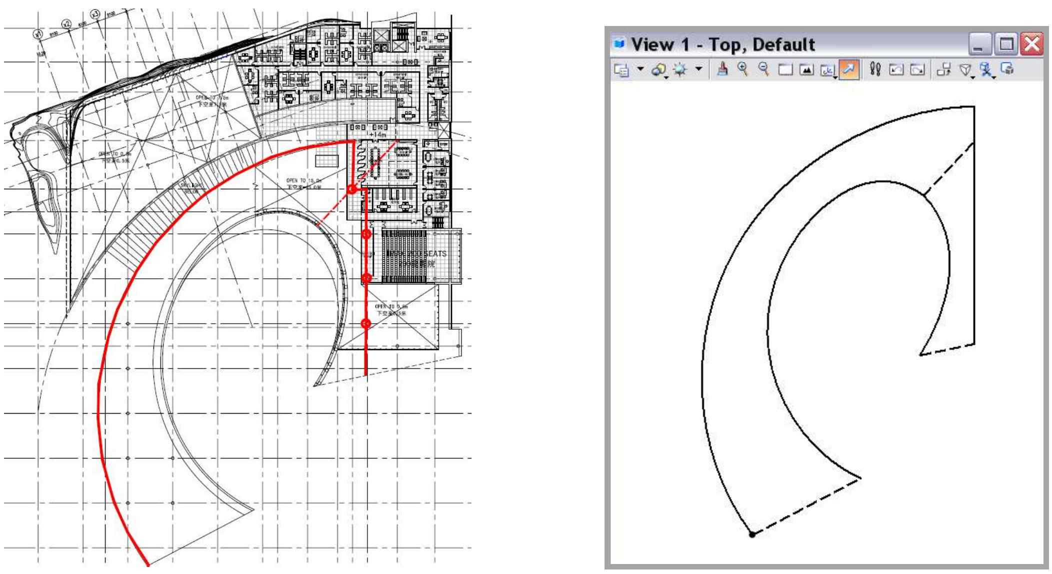


Fig.7.4 Roof loaded area from floor plans (left) and configuration in GenerativeComponents (right)

Such an area configuration was regenerated in GenerativeComponents and used to calculate the cross-span to each local position. For following calculation, the area configuration was represented in Cartesian coordinate system. The process was similar as unfolding the boundary curves to straight lines, and keeping the span information for each position on boundaries.

A law curve[16] based on the ‘Span-Length’ relationship was created in GenerativeComponents. [Note: the range of the length was scaled to 165m, considering the area configuration was generated by floor plans, in which the boundary curves were projected to the ground plane]

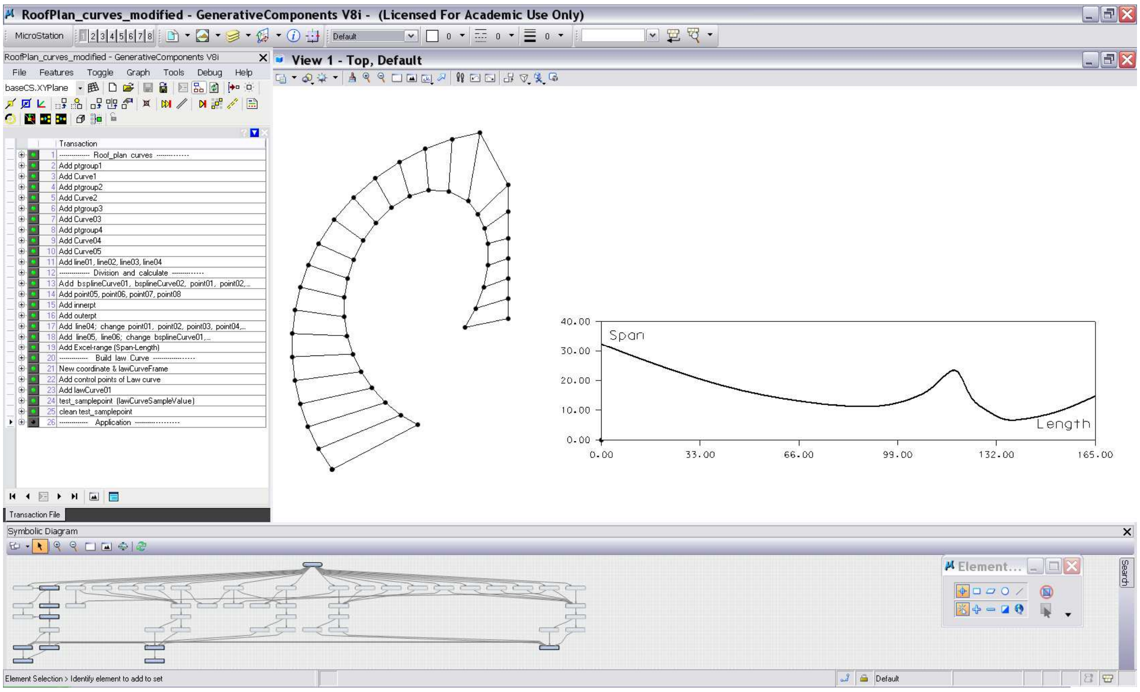


Fig.7.5 Represent the configuration by LawCurve

Step2: Apply to the cellular wall model

Data flow: GSA → Excel → GC → Excel → GSA

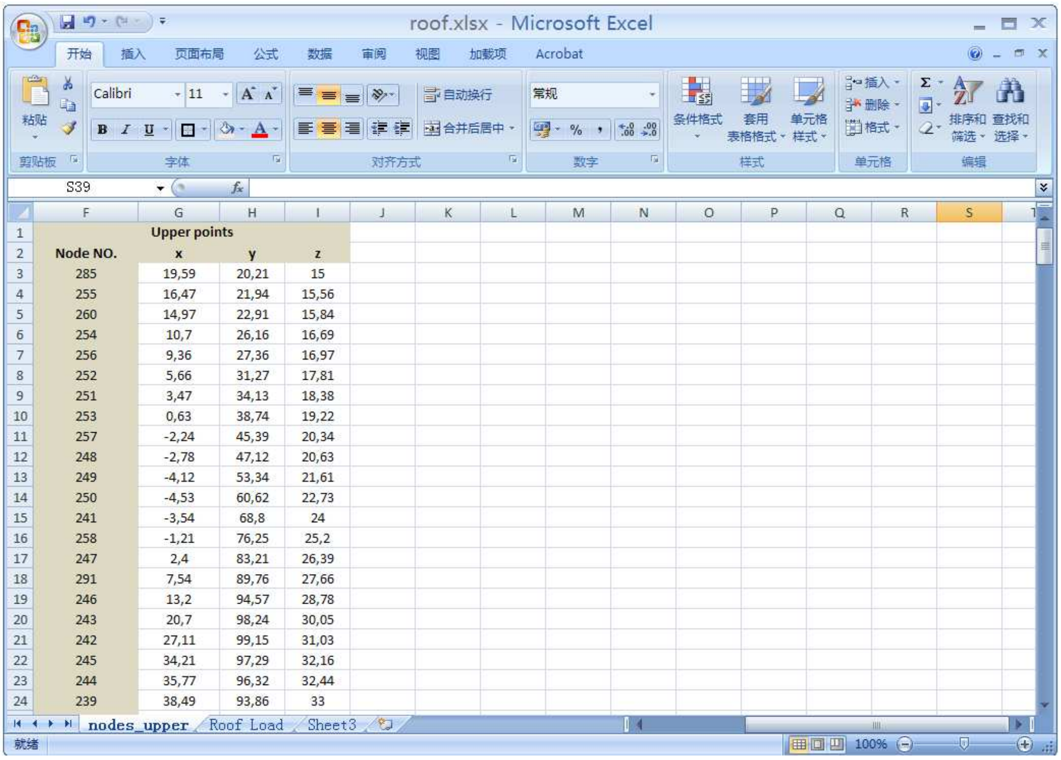
1. Get the coordinates of nodes on upper boundary from GSA to Excel (Note: the nodes must be listed by sequence – on one direction along the curve - for the following implementation in GenerativeComponents);

2. Generate the upper nodes in GenerativeComponents by the coordinates in Excel, and create connectivity to represent the upper boundary curve;

3. Create sample points on the LawCurve, according to the upper nodes (note: the sample points must be generated by the function of “Point.OnPlane”); and get the corresponding span value from the LawCurve for each node (function “LawCurveSampleValue.BySliderPoint”); Two components for each node were written to Excel: Length (along the curve) and span;

4. The nodal loaded areas were calculated in Excel according to the exported information from GenerativeComponents; Resulted values were nodal loads for LC2 and LC3;

[16] A Law Curve is a Feature in GenerativeComponents, which has a level of control over its properties, represented in a Law Curve Frame or graph. The X and Y values of the graph are both Properties of the Law Curve. The X direction is the Independent and the Y direction is the Dependant.



Node NO.	x	y	z
285	19,59	20,21	15
255	16,47	21,94	15,56
260	14,97	22,91	15,84
254	10,7	26,16	16,69
256	9,36	27,36	16,97
252	5,66	31,27	17,81
251	3,47	34,13	18,38
253	0,63	38,74	19,22
257	-2,24	45,39	20,34
248	-2,78	47,12	20,63
249	-4,12	53,34	21,61
250	-4,53	60,62	22,73
241	-3,54	68,8	24
258	-1,21	76,25	25,2
247	2,4	83,21	26,39
291	7,54	89,76	27,66
246	13,2	94,57	28,78
243	20,7	98,24	30,05
242	27,11	99,15	31,03
245	34,21	97,29	32,16
244	35,77	96,32	32,44
239	38,49	93,86	33

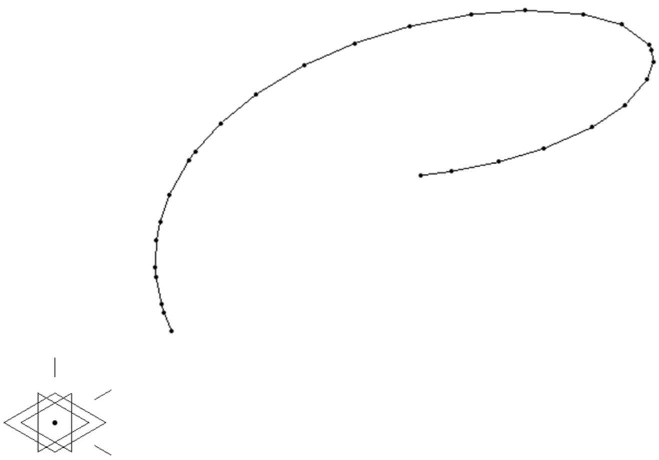
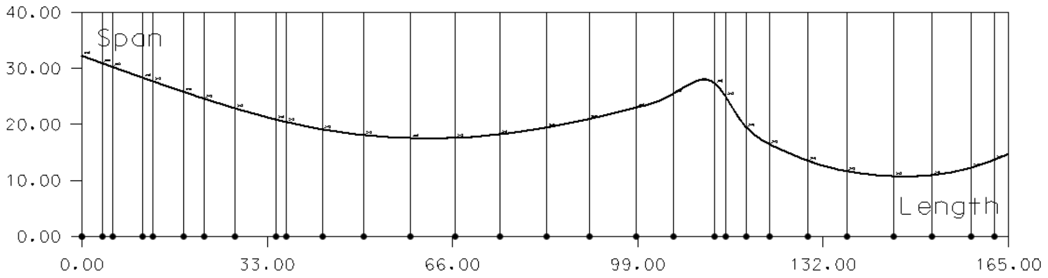
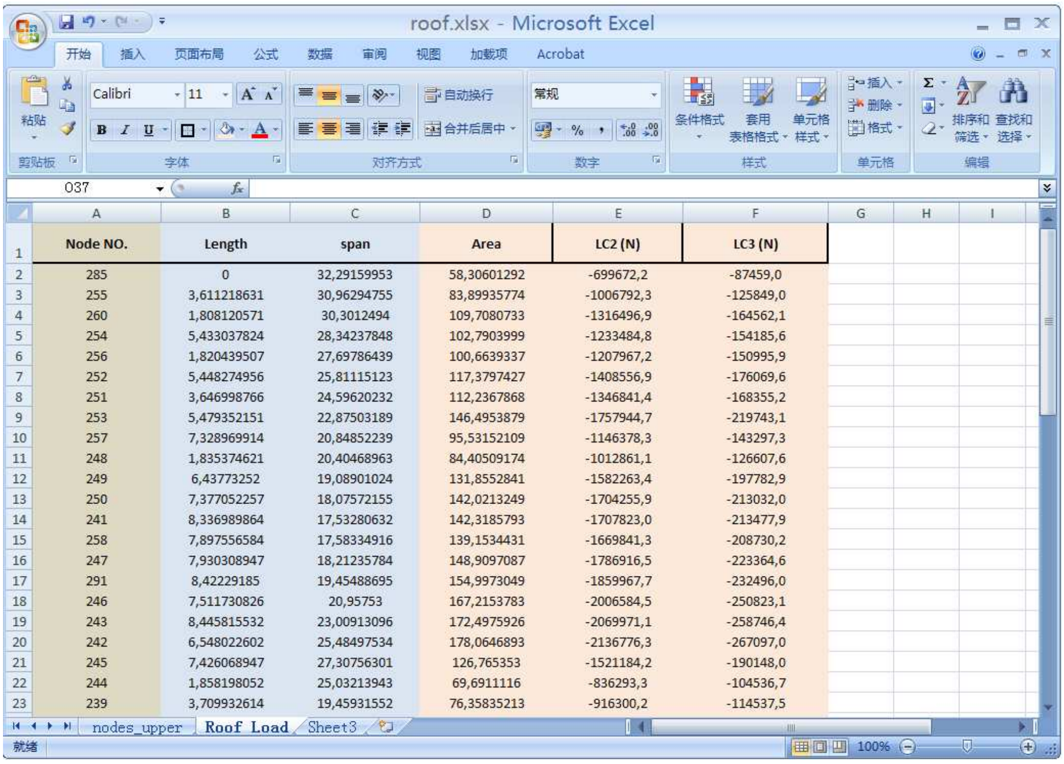


Fig.7.5 nodal coordinates from GSA to Excel, listed by sequence (top)

Regenerate the nodes on upper boundary and the connectivity (bottom)
Note: since the nodes were listed by sequence, the connectivity was simply defined as connecting nodes by sequence;





Node NO.	Length	span	Area	LC2 (N)	LC3 (N)
285	0	32,29159953	58,30601292	-699672,2	-87459,0
255	3,611218631	30,96294755	83,89935774	-1006792,3	-125849,0
260	1,808120571	30,3012494	109,7080733	-1316496,9	-164562,1
254	5,433037824	28,34237848	102,7903999	-1233484,8	-154185,6
256	1,820439507	27,69786439	100,6639337	-1207967,2	-150995,9
252	5,448274956	25,81115123	117,3797427	-1408556,9	-176069,6
251	3,646998766	24,59620232	112,2367868	-1346841,4	-168355,2
253	5,479352151	22,87503189	146,4953879	-1757944,7	-219743,1
257	7,328969914	20,84852239	95,53152109	-1146378,3	-143297,3
248	1,835374621	20,40468963	84,40509174	-1012861,1	-126607,6
249	6,43773252	19,08901024	131,8552841	-1582263,4	-197782,9
250	7,377052257	18,07572155	142,0213249	-1704255,9	-213032,0
241	8,336989864	17,53280632	142,3185793	-1707823,0	-213477,9
258	7,897556584	17,58334916	139,1534431	-1669841,3	-208730,2
247	7,930308947	18,21235784	148,9097087	-1786916,5	-223364,6
291	8,42229185	19,45488695	154,9973049	-1859967,7	-232496,0
246	7,511730826	20,95753	167,2153783	-2006584,5	-250823,1
243	8,445815532	23,00913096	172,4975926	-2069971,1	-258746,4
242	6,548022602	25,48497534	178,0646893	-2136776,3	-267097,0
245	7,426068947	27,30756301	126,765353	-1521184,2	-190148,0
244	1,858198052	25,03213943	69,6911116	-836293,3	-104536,7
239	3,709932614	19,45931552	76,35835213	-916300,2	-114537,5

Load Case calculation

For point[i]: Area[i] = span[i] * (Length[i] + Length[i+1]) /2

LC2[i] = Area[i]*1.5

LC3[i] = Area[i]*12

Fig.7.6 geometry information by sample points on LawCurve (top)

Geometry information from GenerativeComponents and load calculation by Excel (bottom)

Load Case 4, 5 - Façade load

The dead load of the glass façade

The starting point for the definition of the dead load of the façade is that the glass panels are point supported by spider fittings (As an example in Fig.8.7). More information for the integration of structure and façade, design consideration of glass structure were described in Chapter10 Detailing design.

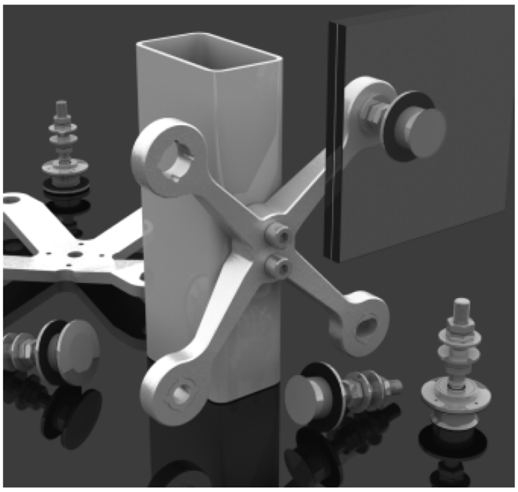


Fig.7.7 PSG (Point Supported Glass) system, Novum structures 2006

In a paper of frameless glazing, John Colvin [Hansen Glass Processing Ltd. 2005] gives an overview of glass types giving some rules of thumb for estimating the likely thickness of glass required to resist the applied loads for various applications where frames are eliminated. The paper is meant for designers to be able to quickly ascertain the feasibility of the glazing ideas where frames are eliminated. By using the rule of thumb, the designer ensures himself of the fact that initial estimates of cost and determination of glass thickness start off on basis which will be not too far from the final requirements.

The rule of thumb for toughened glass, sloping glazing is:

$$\text{Maximum unsupported span} \approx 150 \times \text{thickness}$$

For the calculation of the loads resulted from own weight of the glass panels, the following data is used:

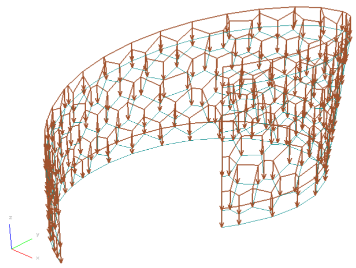
- Edge length (L)
  The length of the edge is needed to estimate the thickness of the glass, and determine the façade surface per node.
- The glass panel thickness
  The thickness of the glass is estimated by the given rule of thumb: cell span/150;
  In the fabrication point of view, thickness of the glass panels are set to be the same in one model, thus, estimation should be done by the largest cell span.
- The glass density
  The density of glass is set at 25 (kN/m^3)
- Façade surface per node (A)
- Before any calculation tool for geometry information of cell elements is created, a rough calculation was done to get the façade area loading on each node. Given the fact that the grid cells of the lattice structure are mainly distorted hexagons, the edge length of the glass can be approximated by regular hexagon: $A = 3\sqrt{3} \times L^2 / 4 \approx 1.3L^2$.
- Point load of the glass
  The point load on nodes of the structure is calculated by multiplying the façade surface per node, the thickness of the glass and the density of the glass.

Wind load

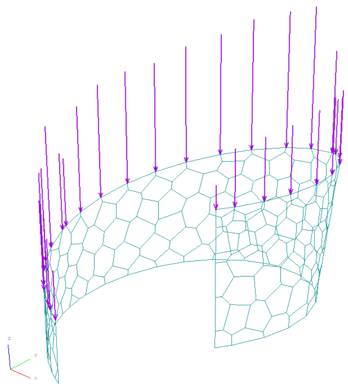
In the design process, the approximated value for the wind load was set at 1 kN/m^2; and the point load on nodes of the structure is calculated by multiplying the façade surface per node and 1 kN/m^2.

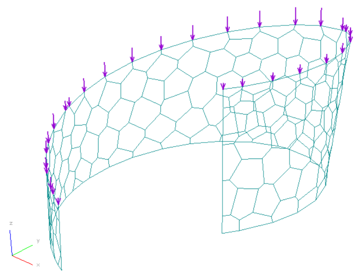
Note: Wind load for non-standard shaped building is a complex problem. There are no sensible statements to be made on how the wind load will act on the building. Therefore, in this thesis study, only one direction (orientating to the opening of the building) will be loaded by wind force. And no negative wind pressure was taken into account.

LC1: self-weight

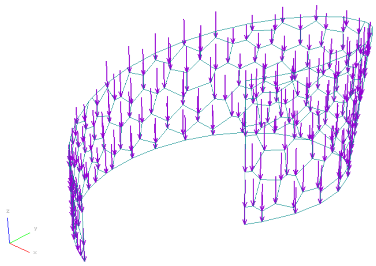


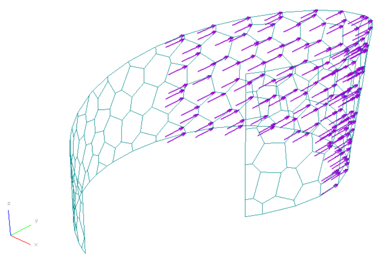
LC2: dead load of the green roof, LC3: live load of the green roof





LC4: weight of the glass façade, LC5: wind load from façade





Arrangement of load cases

Load cases:
L1, L2, L3, L4, L5

Combined load cases:
C1 (SLS) = L1+L2+L3+L4+L5
C2 (ULS) =1.2*(L1+L2+L4) + 1.35*(L4+L5)

7.5 Parametric Analysis Results

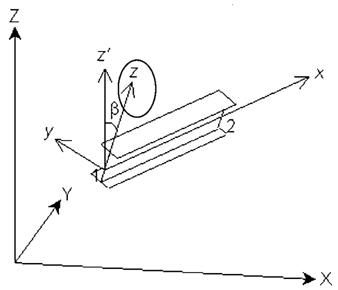
Model validity checking

Before analysis the structural model and evaluating the results of the structural analysis, the validity of the generated GSA model should be checked. With developing the generation process, the constructed GSA model was checked for many possible parametric variations. In a parametric design, it is necessary to check the structural 3D model.

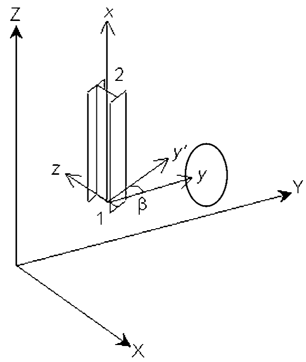
The geometry generation process generates irregular patterns (structured/unstructured grid) in different generation strategies, it is possible that one of the algorithms does not function properly under all possible parameter configurations or unwanted geometry is generated. When a structural model is constructed in GSA, there are two main points of the geometry should be checked first:

1) The pattern configuration should be checked, which includes: to avoid the unwanted elements – too deformed cells, too small/large beam elements; and to delete the nodes/elements that have duplicated definition; this is very necessary when the geometry is imported by .dxf format.

2) The local coordinates of the elements (especially the vertical elements) should be checked;
In GSA, local coordinates are defined as beam element axes: Beam elements including bars, ties and struts are defined by two nodes locating the ends of the element. The x axis of the element is along the axis of the element (taking account of any offsets) from the first topology item to the second. The definition of the element y and z axes then depends on the element's orientation, verticality, and orientation node and orientation angle. The element is considered vertical in GSA if the element is within the "vertical element tolerance".



Non-vertical elements
If an orientation node is not specified, the element z axis of a non-vertical element defaults to lying in the vertical plane through the element and is directed in the positive sense of the global Z direction. The element y axis is orthogonal to the element z and x axes. The element y and z axes may be rotated out of this default position by the orientation angle.



Vertical elements
If an orientation node is not specified, the element y axis of a vertical element defaults to being parallel to and is directed in the positive sense of the global Y axis. The element z axis is orthogonal to the element x and y axes. The element y and z axes may be rotated out of this default position by the orientation angle.

Orientation node
If an orientation node is specified, the element xy plane is defined by the element x axis and a vector from the first topology position to the orientation node, such that the node has a positive y coordinate. The element z axis is orthogonal to the element x and y axes. Specifying an orientation node overrides the "vertical element" and "non-vertical element" definitions described above. The element y and z axes may be rotated out of this default position by the orientation angle.

Orientation angle
The element y and z axes are rotated from their default positions about the element x axis by the orientation angle in the direction following the right hand screw rule. This occurs regardless of whether or not the element is vertical and of whether or not an orientation node is specified.

[Process] Check for the vertical elements, rotate them by changing the 'orientation angle' to fit the surface geometry – local y axis perpendicular to the surface.

Besides the consistent geometry, the structural 3D model of the cellular wall structure should also be checked for correct restrain definition (supports), correct section properties and correct loading. Once the validity of 3D model has been ascertained, the model is ready for analysis, and the results of the analysis can be evaluated.

Criteria for Structural Analysis and Evaluation

This section focuses on the evaluation of structural behaviors; Evaluation is based on the structural analysis results, which includes force flow/stress distribution/deformation/reaction forces etc. The failure modes of a structure can be summarized into two categories: material failure and stability problem. The influence to the other parts of the building or to the service functions should be also taken into account. Therefore, the main design criteria include:

1_Allowable Stresses (Static-linear analysis)

Steel hollow section (rectangular) was chosen for the lattice structure. High strength steel plates can be applied. Thus in the first stage of design, allowable stress of the steel was set around 400MPa – as a checking value in ULS analysis.

2_Deformation (Static-linear analysis)

The cellular wall structure has a dimension of 33×160 (33m high and circumference of 160m). "Deformation of the whole lattice structure < span/200" can be used in s simplified checking.

The starting point of the cellular wall structure was to integrate the steel primary structure and glass façade by apply point supported glass system. In that case, the main consideration for deformation was based on the glass panels. The design values have been set by the rule of thumb "Maximum unsupported span ≈ 150 × thickness". The span to thickness ratio is very large and therefore a deflection close to the thickness of the glass still results in a very small rotational angle. For more detailing, two aspects can be further checked for the glass panels:
- In plane compression and tension by beam element- axial strain or equivalent calculation
- Rotation of the glass panel by out-of-plane level of the cell elements after deformed

Note: Deformation in the vertical direction - |Uz| might cause the cracking problem on the other side of the roof or the mid-span columns (as shown in Fig.7.8, middle). It should also be evaluated.

3_Reaction forces (Static-linear analysis)

Besides the normal checking for foundation (reaction forces at bottom and side edges), the main concern is for the roof and the shear wall on other side (as shown in Fig.7.8, right). Verify if they can provide a fully horizontal support (or efficient K) to the cellular wall.

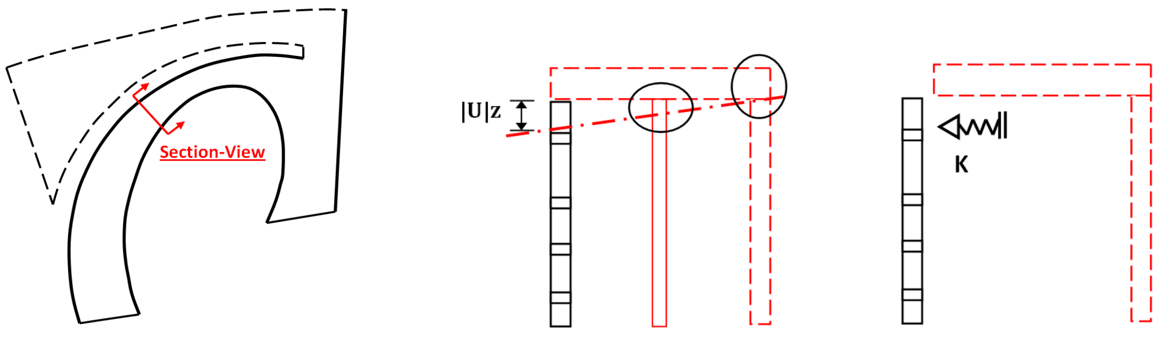


Fig. 7.8 deformation and reaction problem shown by section-views

4_Buckling problems (Buckling - Modal analysis)

Buckling problem includes: Modal buckling and member buckling. Member buckling problem is not included in this thesis study. Model buckling will be implemented and evaluated.

7.5.1 Structured grid models

The Parametric variables in the structure grid model are L & RSc, as described in the former chapter. The starting point (model 1) was based on the architectural concept design, in which the beam elements have an average length around 4 meters. More grid structures with increased densities (average length around 3m and 2.5m) were compared. And the same randomness level (RSc=20%) was applied to the models.

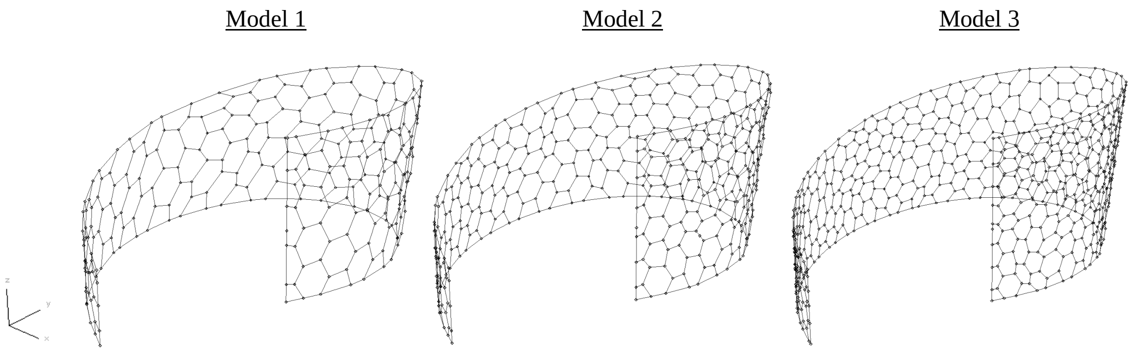


Table 7.2 structured grid models data:

Series 1 – apply the same cross-section to the three models:

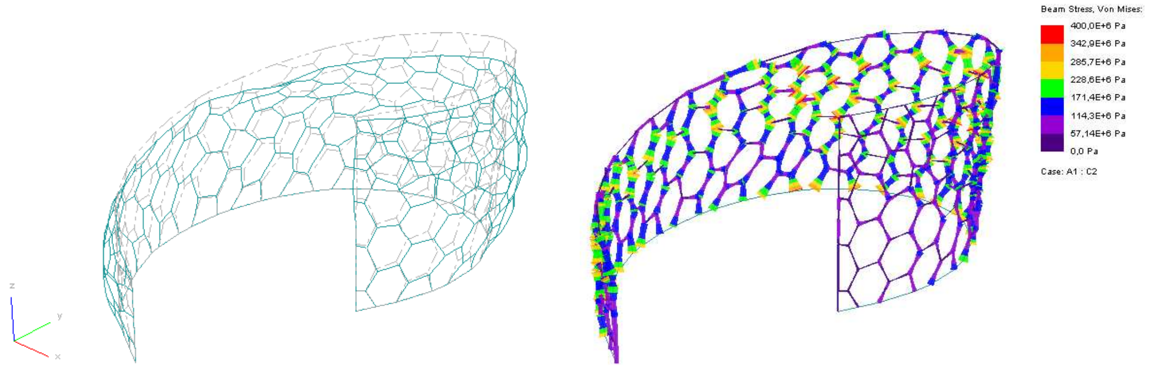
Model		1	2_a	3_a
Parametric variables		L=11.0m, RSc=20%	L=8.0m, RSc=20%	L=6.5m, RSc=20%
Cross-section		D=600mm, t=20mm	D=600mm, t=20mm	D=600mm, t=20mm
Average element length		4m	3m	2,5m
Glass thickness		26,7mm	20,0mm	16,7mm
Total Load [kN]	LC1 - own weight	5812,9	7023,5	8460,1
	LC2 - Roof DL	37653,8	37795,4	37851,5
	LC3 - Roof LL	4706,7	4724,6	4731,7
	LC4 - glass façade	3838,1	2507,7	2168,5
	LC5 - wind load	2974,3	2609,0	2673,1

Series 2 – apply the different cross-sections to the three models, to set them with similar own-weight
(Thus, the models were built using the same amount of material)

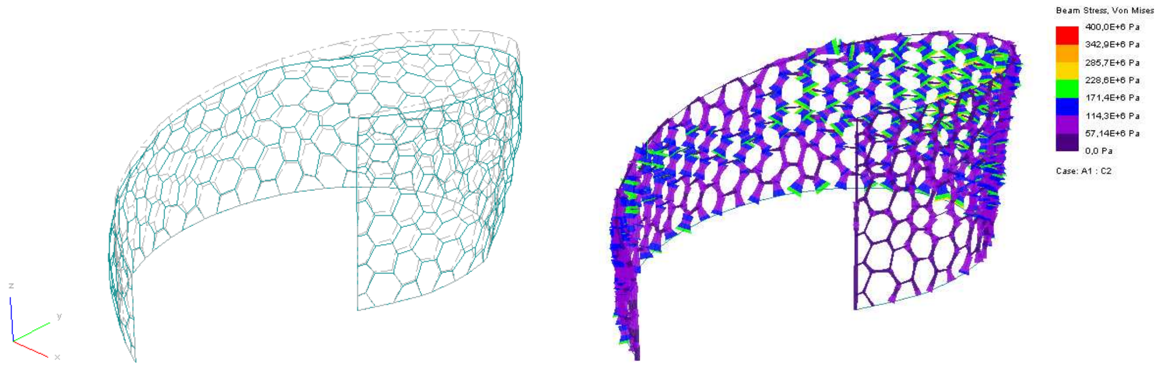
Model		1	2_b	3_b
Parametric variables		L=11.0m, RSc=20%	L=8.0m, RSc=20%	L=6.5m, RSc=20%
Cross-section		D=600mm, t=20mm	D=600mm, t=16,5mm	D=600mm, t=13,6mm
Average element length		4m	3m	2,5m
Glass thickness		26,7mm	20,0mm	16,7mm
Total Load [kN]	LC1 - own weight	5812,9	5829,6	5816,6
	LC2 - Roof DL	37653,8	37795,4	37851,5
	LC3 - Roof LL	4706,7	4724,6	4731,7
	LC4 - glass façade	3838,1	2507,7	2168,5
	LC5 - wind load	2974,3	2609,0	2673,1

Note: All the load cases (L1, L2, L3, L4, L5, C1, and C2) described in chapter8.4 were analysis.
The following analysis results were selected from ULS analysis (C2).

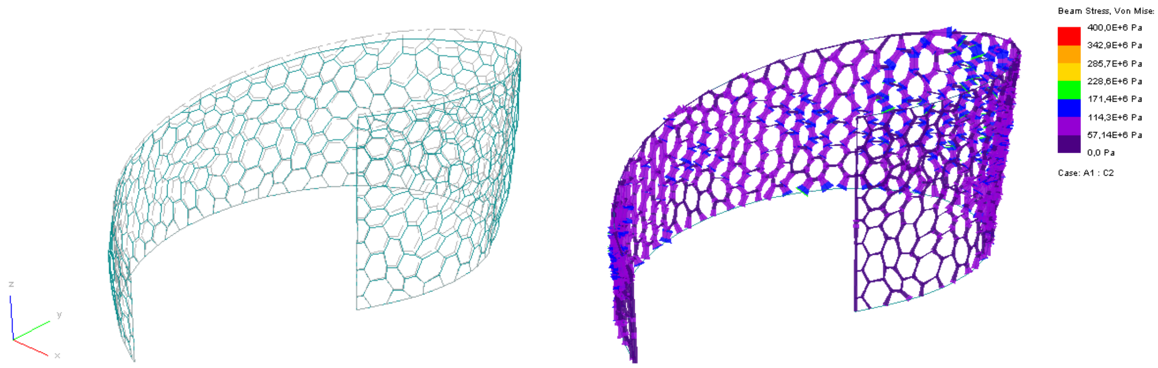
Series 1: apply the same cross-section



Model 1: Average element length=4m (t=20mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 67mm; Z-Translation $|Uz|_{max}$ = 49mm



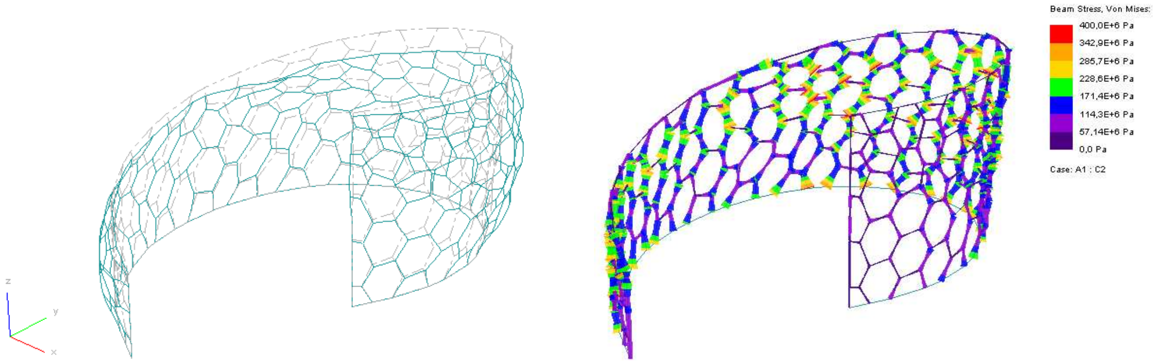
Model 2_a: Average element length=3m (t=20mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 41mm; Z-Translation $|Uz|_{max}$ = 33mm



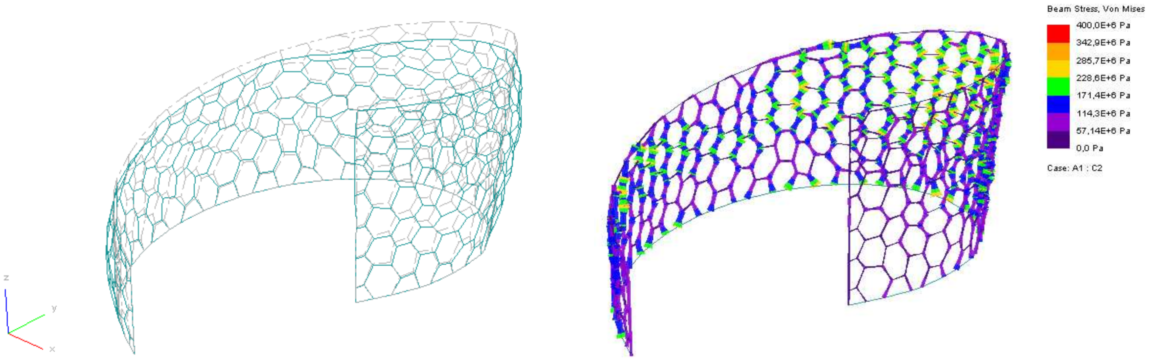
Model 3_a: Average element length=2,5m (t=20mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 31mm; Z-Translation $|Uz|_{max}$ = 25mm

Fig.7.8 structured grid models – series 1 analysis results – deformation and Von Mises stress
[Graph scale: Deformation magnification = 100, stress range 400~0 MPa]

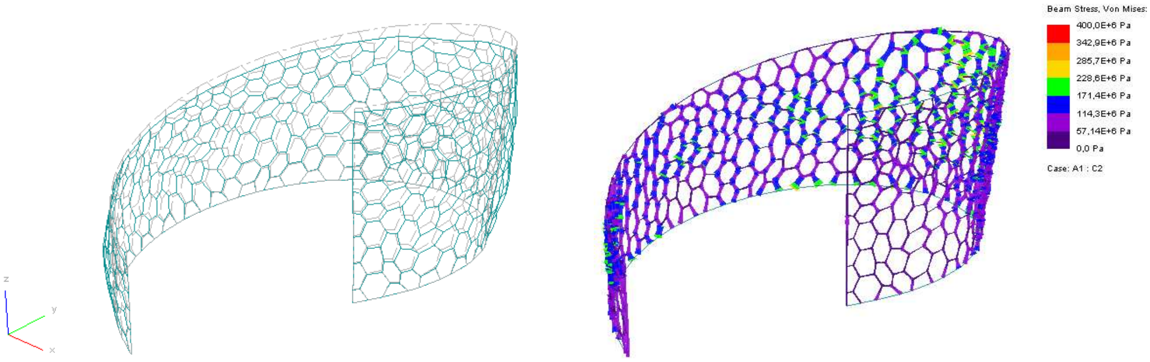
Series 2: apply different cross-sections, with the same total weight



Model 1: Average element length=4m (t=20mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 67mm; Z-Translation $|Uz|_{max}$ = 49mm



Model 2_b: Average element length=3m (t=16,5mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 48mm; Z-Translation $|Uz|_{max}$ = 39mm



Model 3_b: Average element length=2,5m (t=13,6mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 44mm; Z-Translation $|Uz|_{max}$ = 35mm

Fig.7.9 structured grid models – series 2 analysis results – deformation and Von Mises stress
[Graph scale: Deformation magnification = 100, stress range 400~0 MPa]

7.5.2 Modified Structured grid models

Modified structures grid models were based on the structured grid models in the former section. The original structured grid models used in this part is the one with L=11m and RSc=20%. From the previous analysis, the structural behavior of the lattice structure (represented by forces and stresses distribution) were known. The following two modified structure was built according to the stiffness and strength requirements in the lattice network. Cross-section profiles with reduced wall thickness (t=18,5mm and 16mm) were applied, and the total weight of each modified structure was the same as the original one. Thus, the imposed load cases were approximately the same in the compared models.

Modified-1 hexagonal pattern with inserting triangles

From the analysis results, the lattice network with inserting triangles was stiffer than the original model. More moment-free joints thus the material can be used in more efficient way. But in most of the cases, when triangles are interested to the network, large stresses might occur nearby (locally). Several iterations (modify grid – structural analysis – feedback cycle) are needed to optimize the interesting.

Modified-2 hexagonal pattern with local double rhythm

By locally double the hexagonal grid, the lattice structure also got stiffness to restrict the deformation.
The stress distribution was more even than the original structure, and with lower values. The large stresses occurred near to the boundaries, due to the moment fixed boundary condition. In those doubled-up regions, the stresses of the beam elements were relatively low. Thus, a trivial solution can be applying reduced cross-section profiles in the small elements (doubled-up regions).
Compare with the Modified-1 lattice network by inserting triangles, the implementation of double-up grid is easier, but the cell elements are relatively regular.

Original model — modified-1 — modified-2

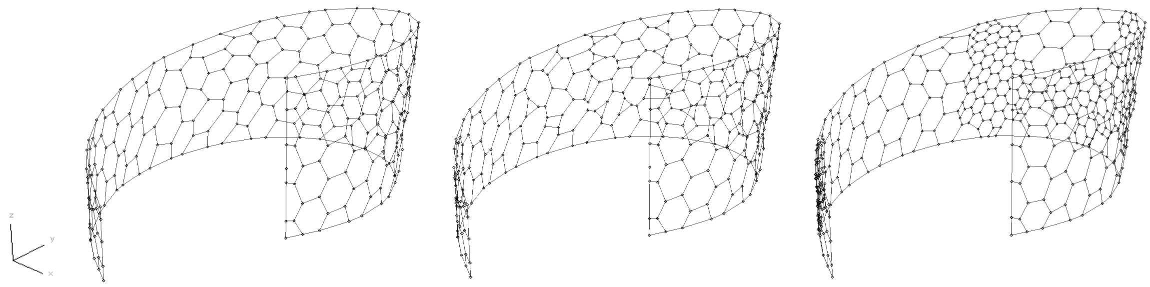
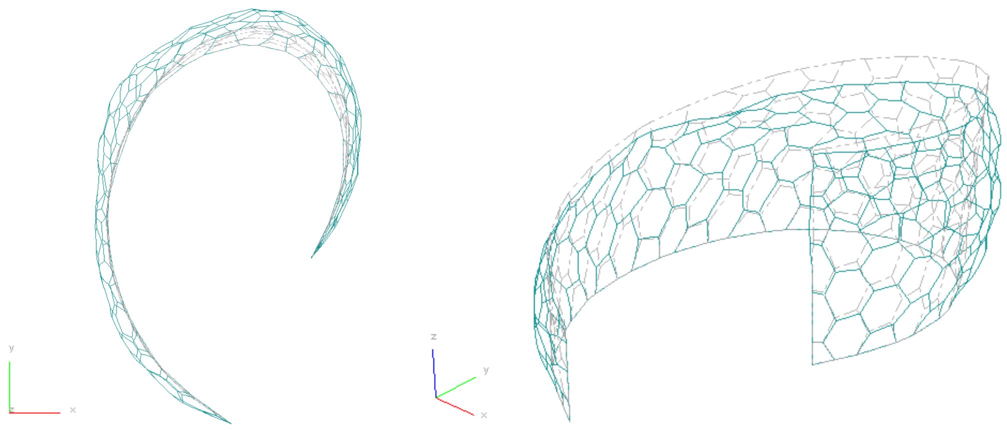


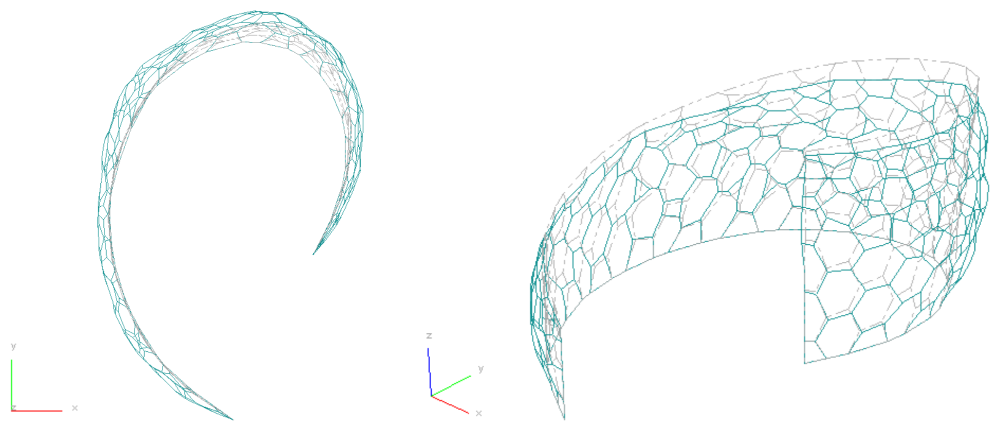
Table 8.3 Modified Structured grid models data:

Model		Original	modified-1	modified-2
Parametric strategy		-	Inserting triangles	Double hexagons
Cross-section		D=600mm, t=20mm	D=600mm, t=18,5mm	D=600mm, t=16mm
Glass thickness		26,7mm	26,7mm	26,7mm
Total Load [kN]	LC1 - own weight	5812,9	5831,3	5840,7
	LC2 - Roof DL	37653,8	37655,8	37905,3
	LC3 - Roof LL	4706,7	4706,8	4737,9
	LC4 - glass façade	3838,1	3796,2	3602,6
	LC5 - wind load	2974,3	2974,4	2683,2

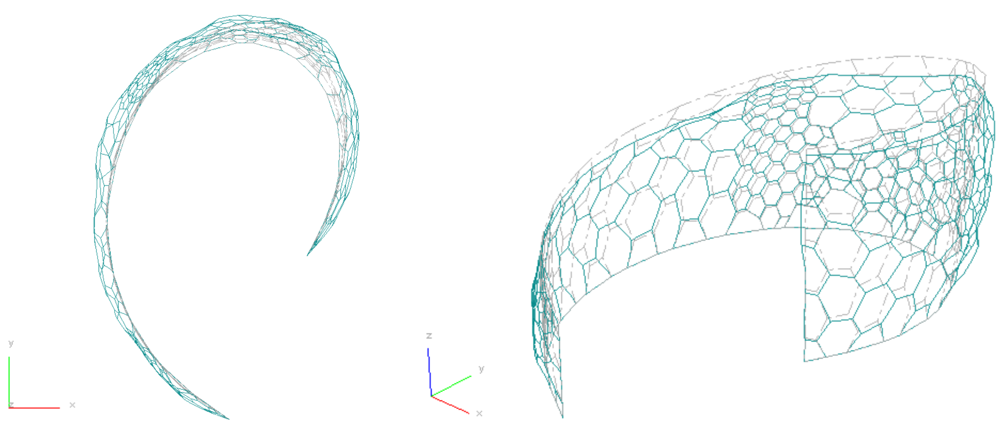
Note: All the load cases (L1, L2, L3, L4, L5, C1, and C2) described in chapter8.4 were analysis.
The following analysis results were selected from ULS analysis (C2).



<u>Original structured grid model: distorted hexagons</u>
Nodal displacements: Resolved Translation $|U|_{max}$ = 67mm; Z-Translation $|Uz|_{max}$ = 49mm



<u>Modified-1: distorted hexagons pattern with inserting triangles</u>
Nodal displacements: Resolved Translation $|U|_{max}$ = 53mm; Z-Translation $|Uz|_{max}$ = 40mm



<u>Modified-2: hexagon pattern with local double rhythm</u>
Nodal displacements: Resolved Translation $|U|_{max}$ = 55mm; Z-Translation $|Uz|_{max}$ = 41mm

Fig.7.10 modified structured grid models analysis result - deformation
[Graph scale: Deformation magnification = 100]

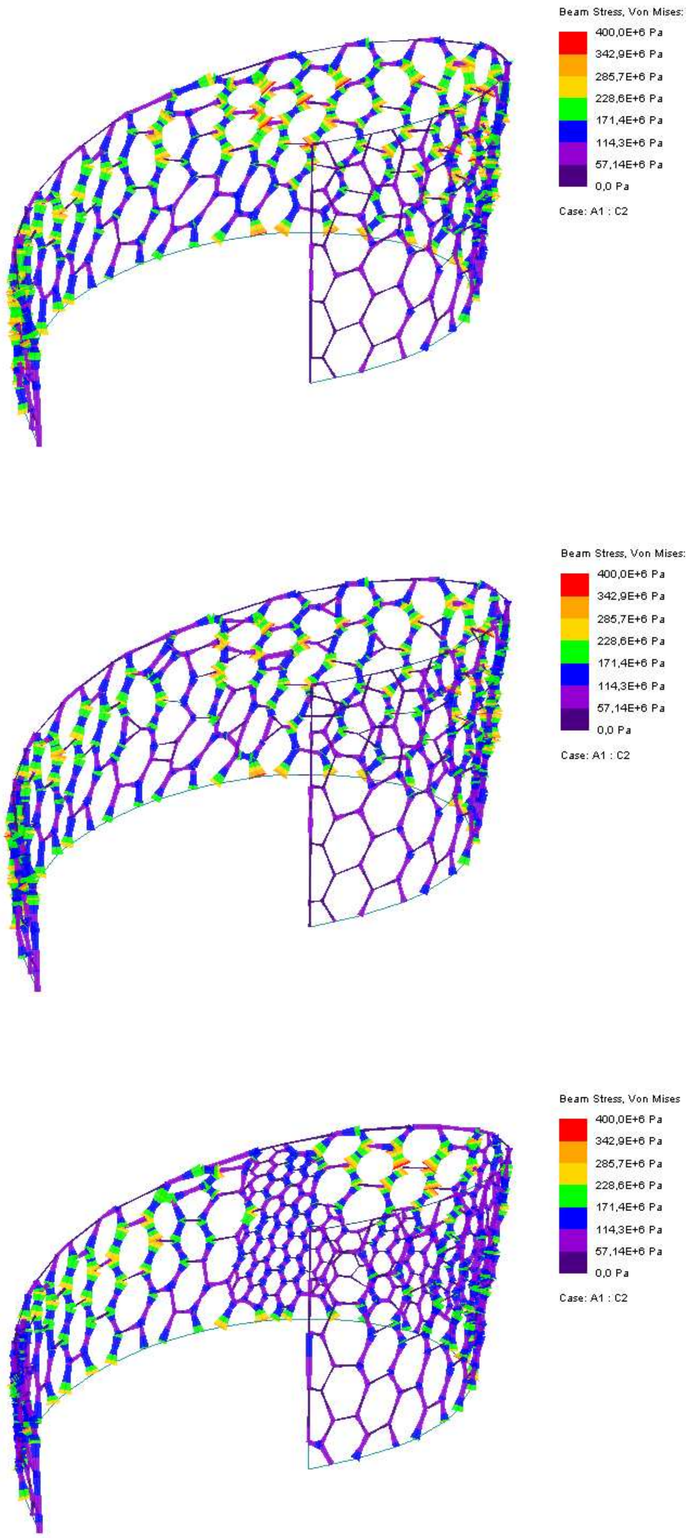


Fig.7.11 modified structured grid models analysis result – Von Mises Stress
[Graphs with the same scale: stress range 400~0 MPa]

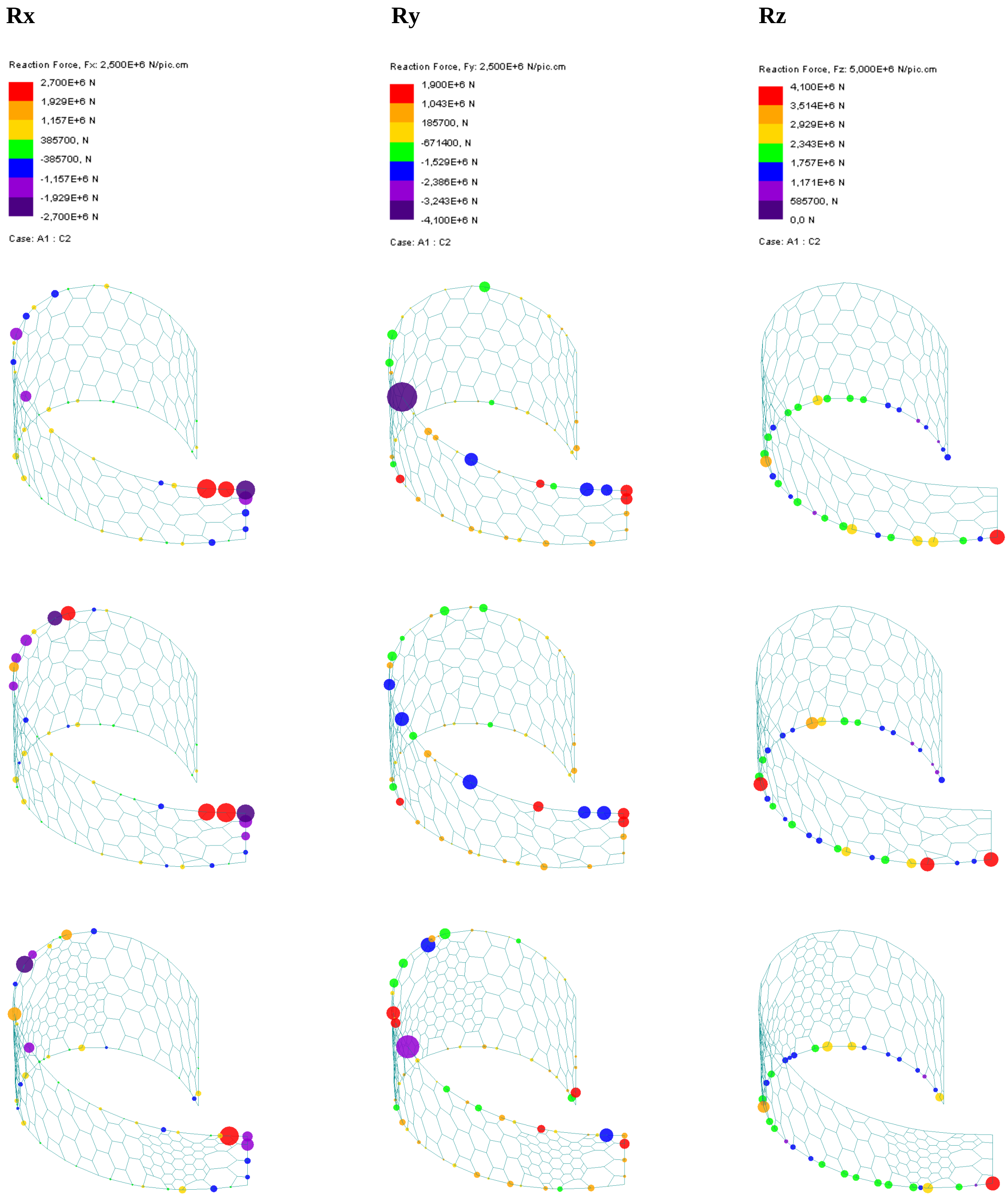


Fig.7.12 modified structured grid models analysis result - Reaction forces

Buckling – Modal Analysis

Buckling Mode 1 **Buckling Mode 2**

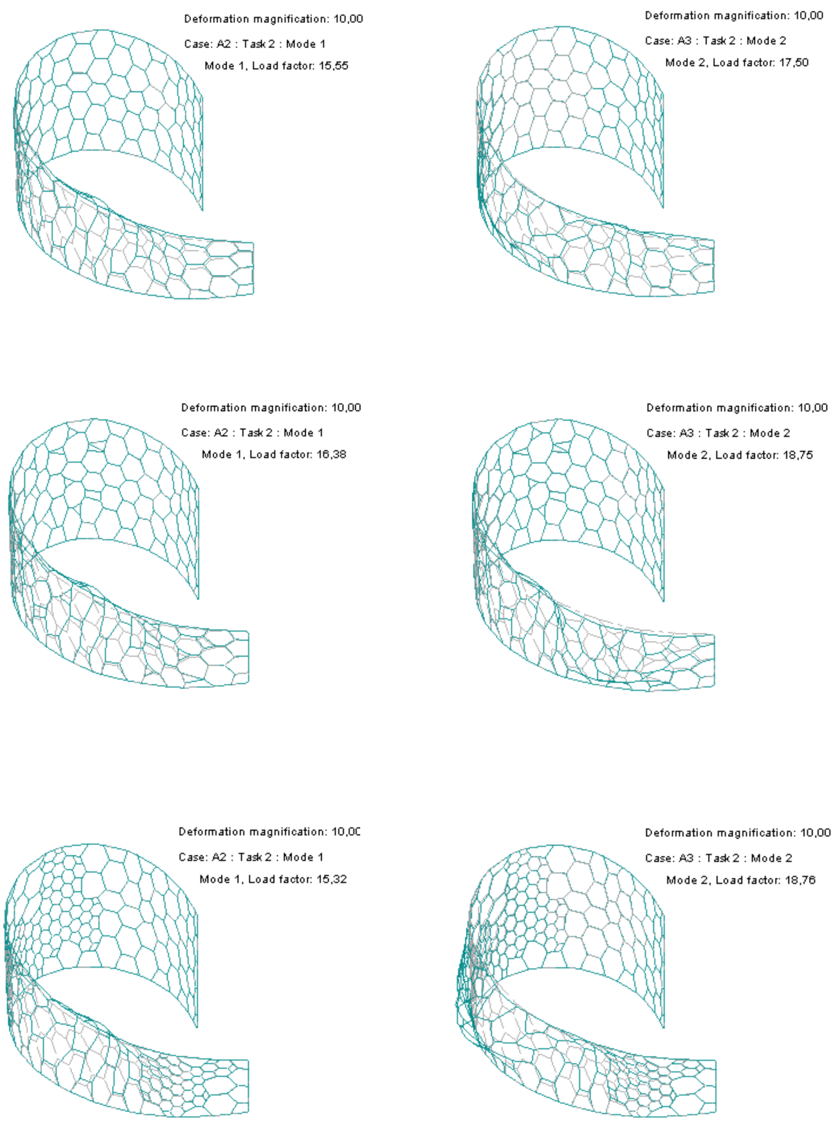


Fig.7.13 modified structured grid models – buckling analysis (Mode1&2)

Note: The results from static linear analysis and buckling modal analysis give clues that buckling is not a critical problem for these cell-like grid structures. The structural deformation and material strength (stresses) determine the dimensions of the structure elements.

7.5.3 Unstructured grid models

Cell densities can be freely predetermined in unstructured grid models, to get lower stress levels, and better forces distribution. The local cell densities are changed smoothly in the lattice network. Instead of distorted hexagons pattern, unstructured grid models apply ‘Voronoi Diagram’. Another advantage of this strategy is that the patterns can be viewed and modified in 2D plane, and then mapped to 3D surface to build up 3D structures.

Two trivial models were analysis and compared:
Model 1 has a relatively coarse pattern. From the analysis results, various cell-densities contribute to the better forces/stress distribution, but local densities did not tune up with the imposed load cases.

Based on Model 1, a finer pattern with modified cell densities was built. Model 2 fits the structural requirements quite well (due to the specific imposed load cases in ULS), thus the resulted stresses were even distribute and with a low level. Model 2_a has the same own-weight as Model 1. By comparison, in Model 2_a the same amount of material built up a stiffer and stronger structure. Model 2_b, reduced cross-section profile was applied.

2D view of the pattern　　　　3D lattice structure

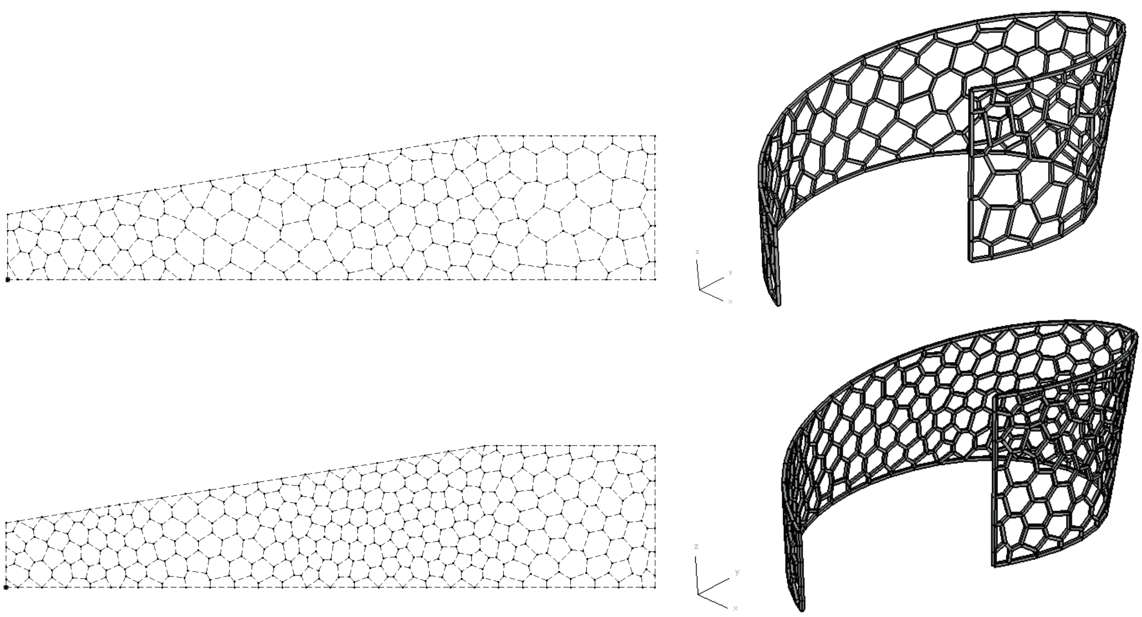
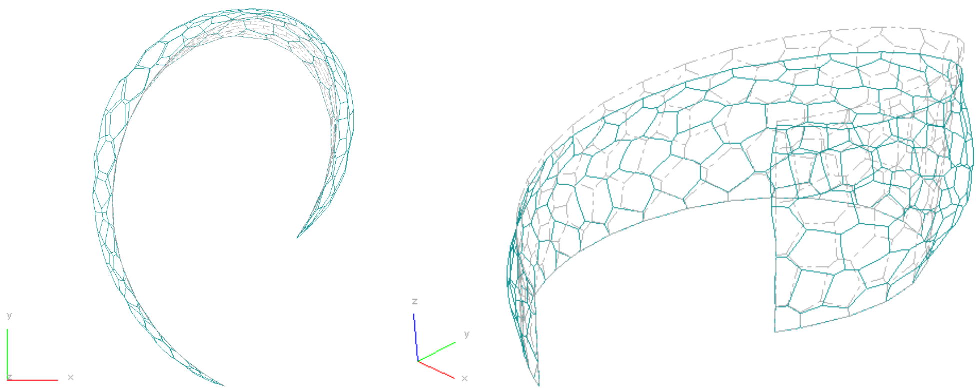


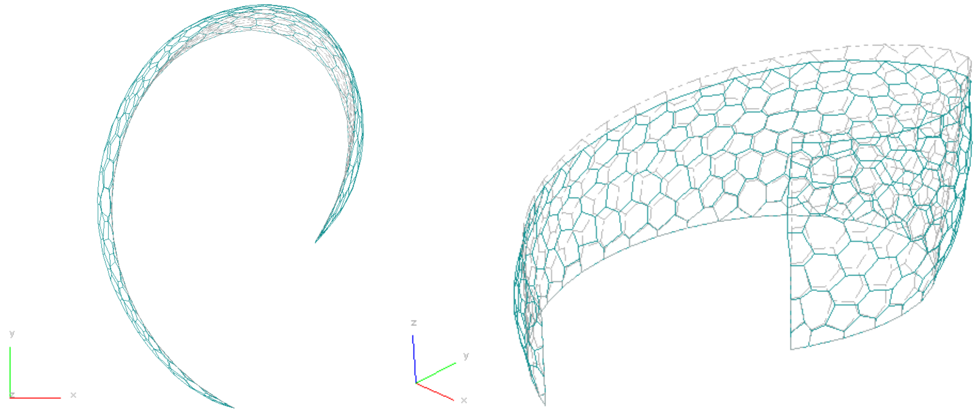
Table 8.4 Unstructured grid models data:

Model		1	2_a	2_b
Parameters (elements)		Voronoi D = 6~8m beams = 3.5~4.5m	Voronoi D = 5~7m beams = 2.8~4.2m	
Cross-section (D=600mm)		t=20mm	t=16mm	t=12,5mm
Glass thickness		30mm	26,7mm	
Total Load [kN]	LC1 - own weight	6003.7	6004,1	4718,5
	LC2 - Roof DL	38247.8	38043,3	
	LC3 - Roof LL	4780.4	4755,5	
	LC4 - glass façade	4096.1	4243,6	
	LC5 - wind load	3078.6	2913,3	

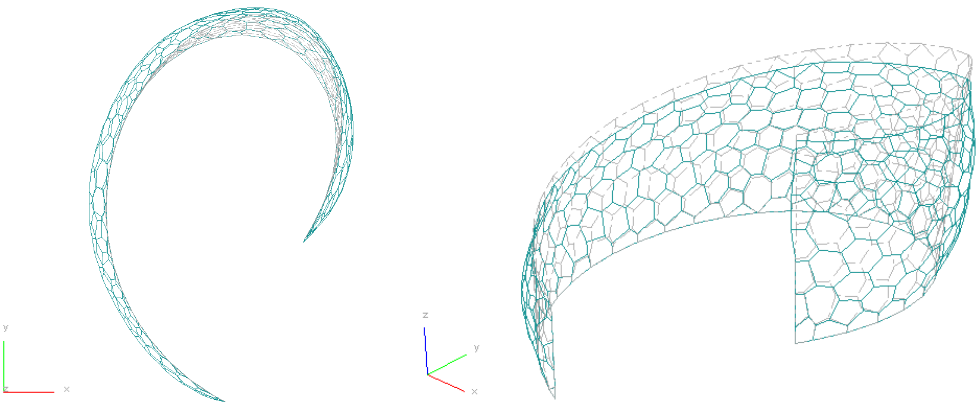
Note: All the load cases (L1, L2, L3, L4, L5, C1, and C2) described in chapter8.4 were analysis.
The following analysis results were selected from ULS analysis (C2).



Pattern1: Voronoi D = 6~8m (t=20mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 72mm; Z-Translation $|Uz|_{max}$ = 49mm

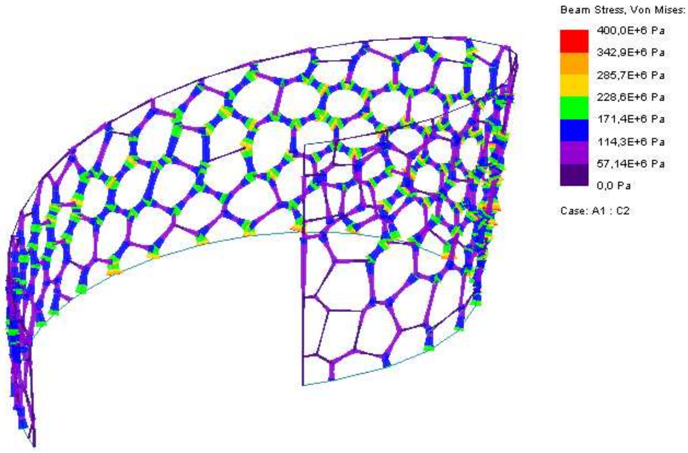


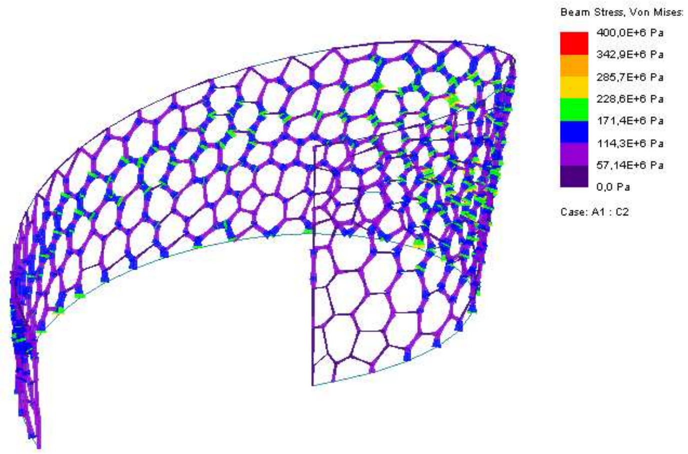
Pattern2_a: Voronoi D = 5~7m (t=16mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 39mm; Z-Translation $|Uz|_{max}$ = 34mm



Pattern2_a: Voronoi D = 5~7m (t=12,5mm)
Nodal displacements: Resolved Translation $|U|_{max}$ = 50mm; Z-Translation $|Uz|_{max}$ = 43mm

Fig.7.14 unstructured grid models analysis result - deformation
[Graph scale: Deformation magnification = 100]





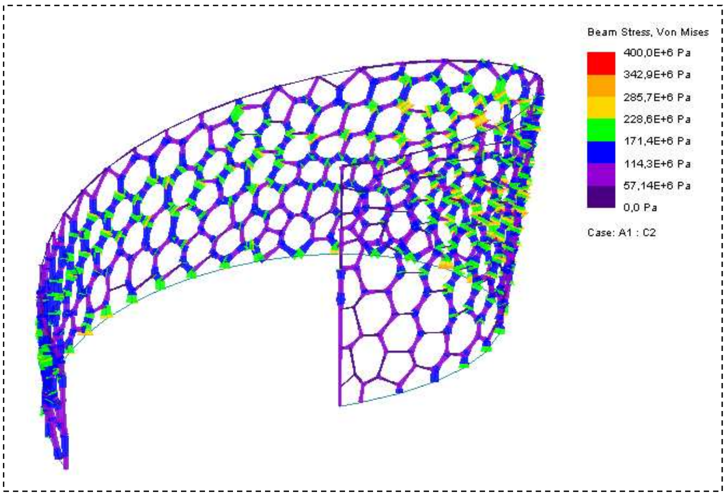


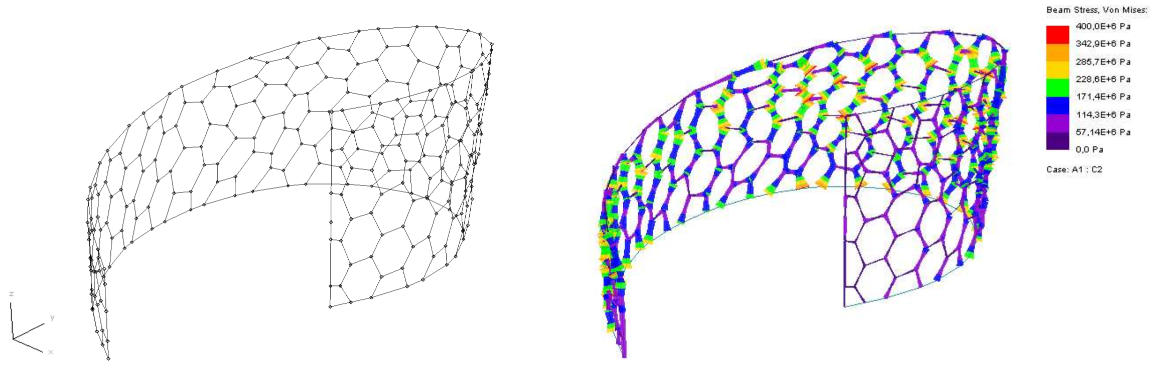
Fig.7.15 unstructured grid models analysis result - deformation
[Graphs with the same scale: stress range 400~0 MPa]

7.6 Member Design

7.6.1 Purpose

The layout of beams in the lattice structure is unconstrained. Hence, the lattice network could be optimized with respect to material efficiency, by scheming for a denser pattern in areas where the structural action is the most demanding (Implementation as the unstructured grid model – various cell densities can be applied). Alternatively, structural capacity could also be increased by apply other cross sections.

This step (Member Design) is the dimensioning of the structural elements. In the first round of structural analysis in GSA (in the previous steps), estimated values have been used for the side length and wall thickness for the RHS (Rectangular Hollow Sections) and all the elements were set to have the same profile.



In the case of the lattice structure that built with the same grid density, the stresses distribution was uneven in the whole lattice network, because of the non-uniformly imposing load cases. [Graph above: Von Mises stress of structured grid model (L=11m, RSc=20%)] Thus, the material was used with a low efficiency level. The purpose of Member Design is to efficiently use material, by assigning different cross-section profiles to the elements according to the strength requirements.

Von Mises stress

The Von Mises stress depends on the axial forces, the bending moment, the through thickness shear forces, and the torsional moment; Simplified calculation of Von Mises stress in GSA:

$$\sigma_{VM} = \sqrt{\sigma_{xx}^2 + 3\tau_{xy}^2 + 3\tau_{xz}^2}$$

Components	Depended forces	Section properties
σ_{xx}	Axial force F_x	A
	Bending M_{yy}, M_{zz}	I
τ	Torsion M_{xx}	J
	Shear F_y, F_z	A_{shear}

For define the dimensions of different profiles, the practical solution is to apply the same diameter but different wall thickness of the RHS (Rectangular Hollow Sections) by groups. For each group, the maximum occurring stresses (Von Mises) are known from the first iteration and the allowable stresses are defined as the strength of standard steel. For the second iteration, the sections are modified in such a way that the occurring stresses were set in several ranges, for each stresses range, a corresponding cross-section profile was signed to the elements.

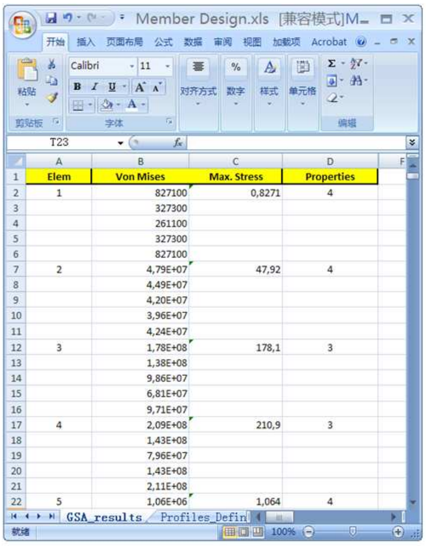
Three software packages were used in this Tool:

Excel - two worksheets are built; one is to parametrically define the section properties that can be used as data base for selecting cross-section profiles and prepare element list to export to GSA; another is to assign profiles according to the stresses results from GSA;

GenerativeComponents - linking to Excel, give a color representation of the different profiles that applied in the lattice structure; virtualizes the implementation;

GSA - interface with Excel, assigning different cross-section properties, repeat the structural analysis, to provide new stresses distribution. In some cases, extra iteration cycles or trivial runs can be applied to optimize the properties.

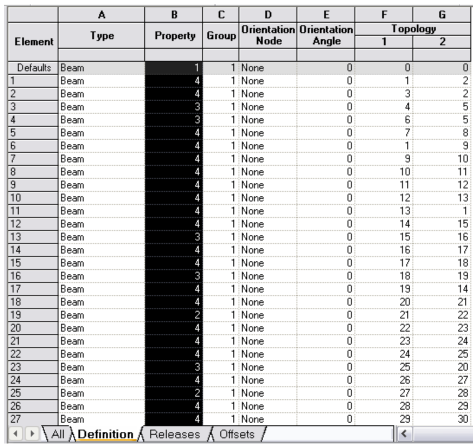
7.6.2 Strategy



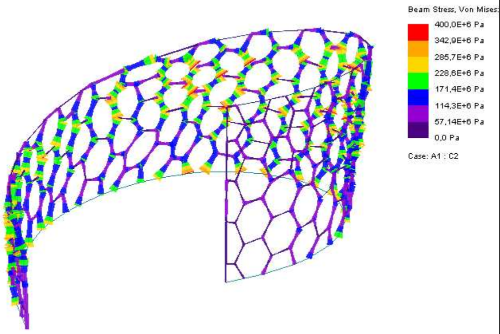
	Elem	Von Mises	Max. Stress	Properties
2	1	827100	0,8271	4
3		327300		
4		261100		
5		327300		
6		827100		
7	2	4,79E+07	47,92	4
8		4,49E+07		
9		4,20E+07		
10		3,96E+07		
11		4,24E+07		
12	3	1,78E+08	178,1	3
13		1,38E+08		
14		9,86E+07		
15		6,81E+07		
16		9,71E+07		
17	4	2,09E+08	210,9	3
18		1,43E+08		
19		7,96E+07		
20		1,43E+08		
21		2,11E+08		
22	5	1,06E+06	1,064	4

Excel

"GSA_results" – assign profiles according to beam-stresses
"Profiles Definition" – define properties of the profiles

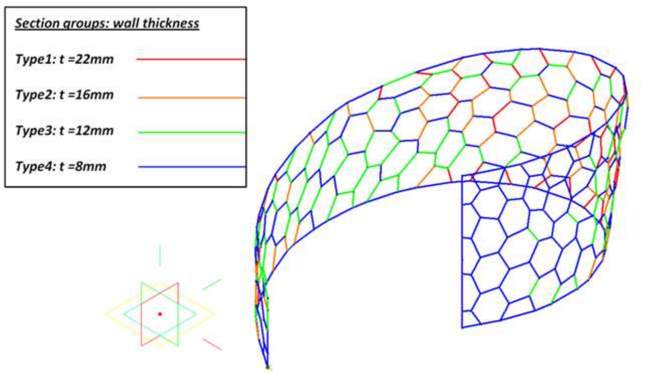


Element	Type	Property	Group	Orientation Node	Orientation Angle	Topology 1	Topology 2
Defaults	Beam	1	1	None	0	0	0
1	Beam	4	1	None	0	1	2
2	Beam	4	1	None	0	3	2
3	Beam	3	1	None	0	4	5
4	Beam	3	1	None	0	6	5
5	Beam	4	1	None	0	7	8
6	Beam	4	1	None	0	1	9
7	Beam	4	1	None	0	9	10
8	Beam	4	1	None	0	10	11
9	Beam	4	1	None	0	11	12
10	Beam	4	1	None	0	12	13
11	Beam	4	1	None	0	13	1
12	Beam	4	1	None	0	14	15
13	Beam	3	1	None	0	15	16
14	Beam	4	1	None	0	16	17
15	Beam	4	1	None	0	17	18
16	Beam	3	1	None	0	18	19
17	Beam	4	1	None	0	19	14
18	Beam	4	1	None	0	20	21
19	Beam	2	1	None	0	21	22
20	Beam	4	1	None	0	22	23
21	Beam	4	1	None	0	23	24
22	Beam	4	1	None	0	24	25
23	Beam	3	1	None	0	25	20
24	Beam	4	1	None	0	26	27
25	Beam	2	1	None	0	27	28
26	Beam	4	1	None	0	28	29
27	Beam	4	1	None	0	29	30



GSA

Modify profiles – overwrite element list
Structural analysis – new stress distribution



GenerativeComponents

Property of Lines - Color representation of the profiles arrangement

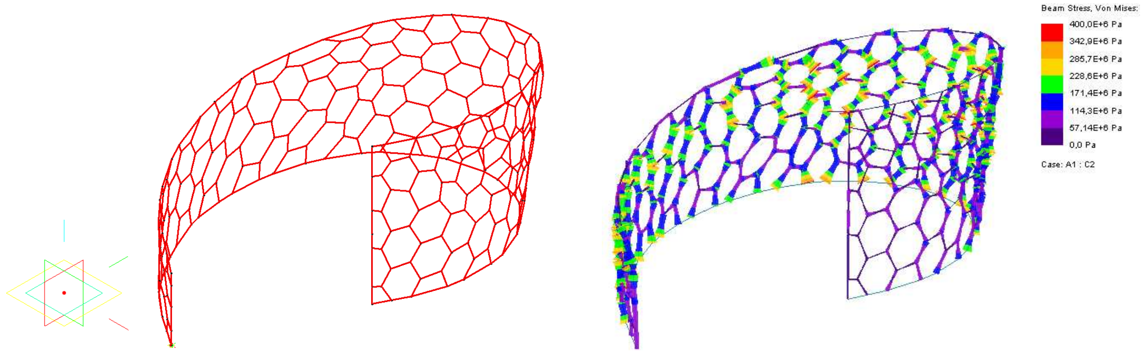
Fig7.16 Strategy of the Member Design

7.6.3 Applications and structural analysis Results

1 Original

Apply one standard profile D=600mm, t=20mm

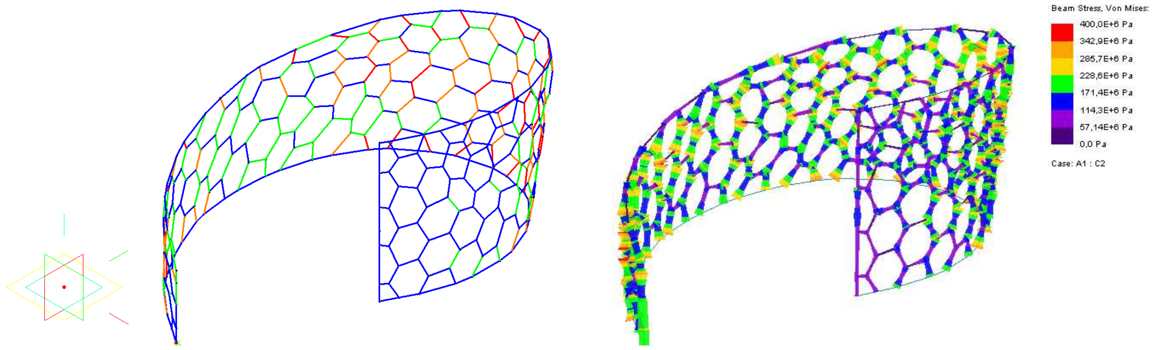
Total weight of the lattice structure: 5812,9 kN



2 By local beam-stresses

Apply 4 groups: profiles (D=600mm), t=24mm, t=18mm, t=14mm, t=8mm

Total weight of the lattice structure: 3663,6kN



2 By uniform regions

Apply 3 groups: profiles (D=600mm), t=22mm, t=14mm, t=8mm

Total weight of the lattice structure: 4764,2 kN

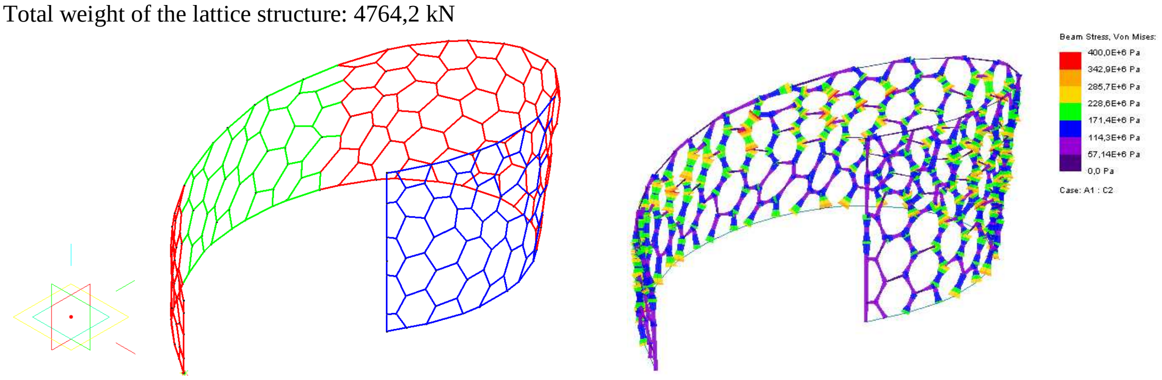


Fig7.17 Color representation of cross-section groups (left) and resulted stress distribution (right)

Note: Graphical representation of the beam stress-Von Mises was selected from ULS analysis (C2).
By apply different cross-section, the resulted stresses have relatively even distribution;

7.7 Compared design cases

As the general grid structure analysis and comparison in Chapter5: triangular grid is always rigid, both in-plane and out-of-plane stiffness are high; rectangular grid (orientating as beam-columns) has weak stiffness, but it shows great advantage under vertical loads; hexagonal grid has relatively good out-of-plane stiffness, but very low in-plane stiffness, thus when vertically loaded (in-plane), it shows the worst behaviors.

Under the specific load cases of the cellular wall – large proportion is vertical load from the green roof – rectangular grid network might be more suitable, comparing with cell-like (hexagonal) grid. This section will make comparison to several design cases for different grid types under the actual design condition of the cellular wall.

Compared cases:
Design case 1_ cell-like grid model (Voronoi Diagram, tuned up densities)
Design case 2_ triangle gird model (regular sizes)
Design case 3_ beam-column model (rectangular grid, regular sizes)
Design case 4_ brick-like model (discontinue rectangular grid, regular sizes)

All of the models have:
- The same D/t ratio of section profiles
- The same load cases (evaluate the analysis results from ULS)
- The same boundary conditions

Design principles:
Decrease the cross section until one of the failure modes is reached:
Buckling factor < 6
or Maximum stresses > 400Mpa

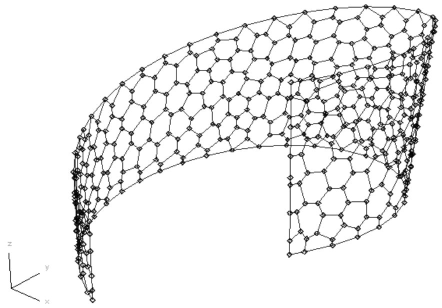
Evaluation Results

Information table of these design cases and the analysis results under C2 (Combine load case - ULS) are listed in the following pages.

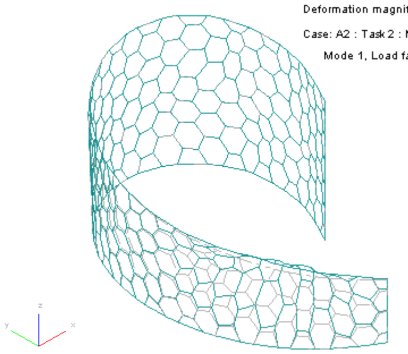
Basically, the failure modes of a structure could be summarized by two categories: one is material yield – strength problem; the other is un-recovered deformation – geometry/stability problem (Loosing stability brings large change to the geometry of the structure, which can be caused by different reasons.)
For the cell-like grid, the deformation and the strength determine the dimensions – as design criteria; while for rectangular grids (beam-column model and brick-like pattern) and triangular grid, the buckling problem is critical – buckling load will determine the dimensions.

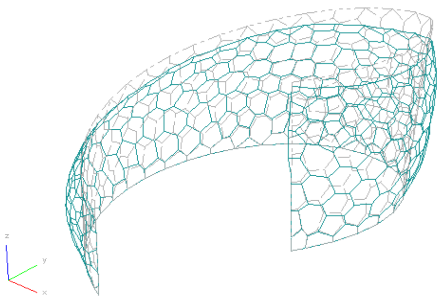
The weak out-of-plane stiffness of the beam-column model can be improved by offsetting the beams, breaking the continuity of the horizontal components. Such a brick-like grid has larger capacity for out-of-plane loads, which can be found from the out-of-plane deformation in the linear static analysis.

In the specific load cases of the cellular wall, the main load is the weight of the green roof (vertically loaded on the wall), while wind load has small proportion. From the amount of used material in different design cases, conclusion can be draw that cell-like grid structure is not an economic/efficient structure form, comparing to the other two grid types.

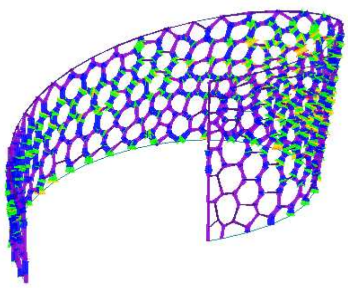


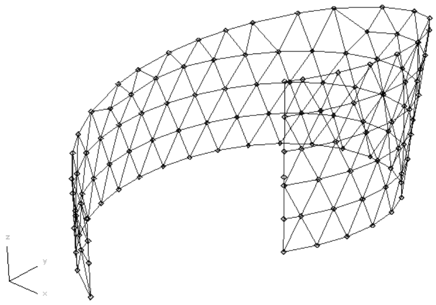
Design case		1
Description		Cell-like grid (tuned up densities)
Element length		4m
Cross-section		D=600mm, t=12mm (D/t = 50)
Glass thickness		26,7mm
Total Load [kN]	LC1 - own weight	4533,5
	LC2 - Roof DL	37795,4
	LC3 - Roof LL	4724,6
	LC4 - glass façade	3203,4
	LC5 - wind load	2609,0



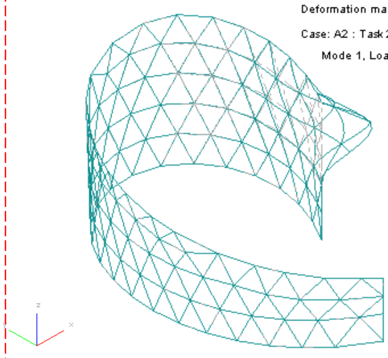


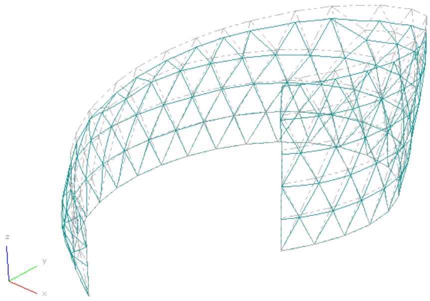
$|U|_{max}$ = 49mm, $|Uz|_{max}$ = 53mm



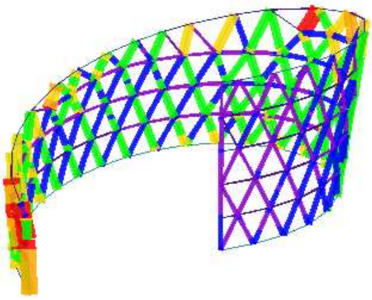


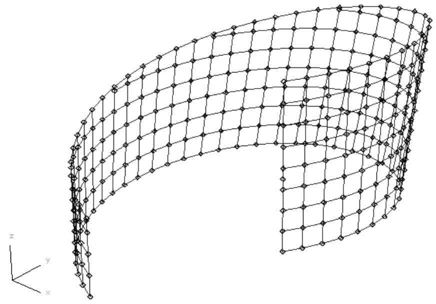
Design case		2
Description		Triangular grid (regular sizes)
Element length		7m
Cross-section		D=375mm, t=7,5mm (D/t = 50)
Glass thickness		26,7mm
Total Load [kN]	LC1 - own weight	1905,1
	LC2 - Roof DL	38221,3
	LC3 - Roof LL	4777,4
	LC4 - glass façade	3200,0
	LC5 - wind load	2707,0



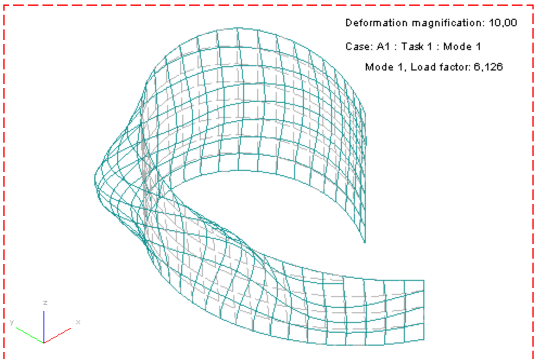


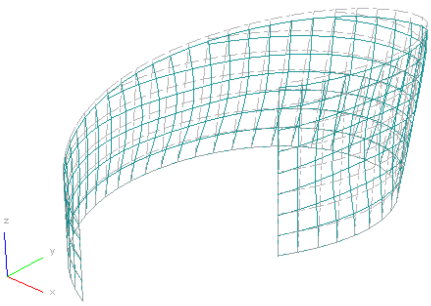
$|U|_{max}$ = 33mm, $|Uz|_{max}$ = 33mm



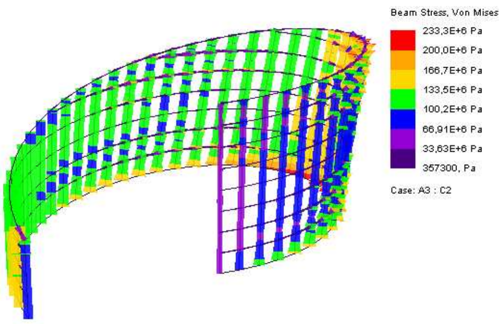


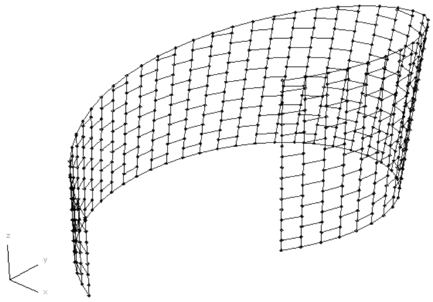
Design case		3
Description		Rectangular grid (beam-column model)
Element length		4m
Cross-section		D=450mm, t=9mm (D/t = 50)
Glass thickness		26,7mm
Total Load [kN]	LC1 - own weight	2718,2
	LC2 - Roof DL	38259,6
	LC3 - Roof LL	4782,6
	LC4 - glass façade	3204,0
	LC5 - wind load	2707,3



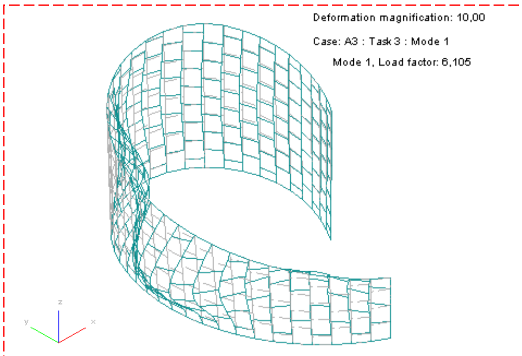


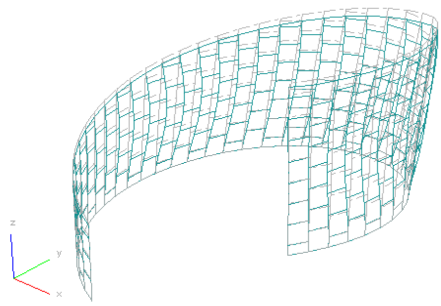
$|U|_{max}$ = 42mm, $|Uz|_{max}$ = 19mm



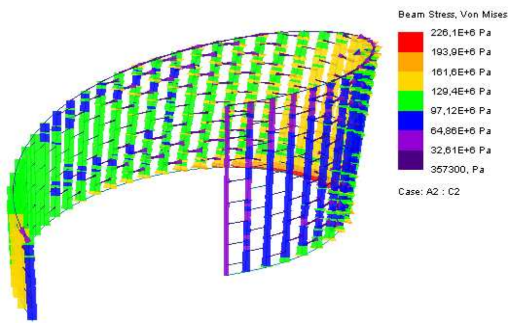


Model		4
Description		Brick-like grid (Discontinue beams)
Element length		4m
Cross-section		D=450mm, t=9mm (D/t = 50)
Glass thickness		26,7mm
Total Load [kN]	LC1 - own weight	2793,3
	LC2 - Roof DL	38258,5
	LC3 - Roof LL	4782,4
	LC4 - glass façade	3214,8
	LC5 - wind load	2707,1





$|U|_{max}$ = 33mm, $|Uz|_{max}$ = 19mm



Chapter 8

Conclusions and recommendations

8.1 Conclusions

8.1.1 Structural definition for the cellular wall

1_ The surface of the cellular wall is a single-curved surface (developable). It can be generated as ruled / lofted surface by two free-form curves (bottom and top boundaries).
2_ The structural topology of the cell-like grid is complex. It represents the wall surface by linear structural elements. Topology of these elements will determine the stiffness properties of the grid structure, also the forces trajectories.
3_ The main load case is vertical load caused by the green roof, and the self-weight of the grid structure is relatively large. Wind load has small proportion in the combined load cases. Therefore, the grid structure is heavily loaded by in-plane loads, and the out-of-plane loads are quite small.
4_ To isolate the cellular wall for structural design and evaluation, the other parts of the building that connect to this wall are translated to supports and imposed loads: fixed foundation, side edges and upper boundary - applied pin connection but free to move in the vertical direction, and the roof and glass layer will bring extra imposed loads to the cellular wall.

8.1.2 Material and construction technologies

1_ Hollow section steel tube is suitable for the grid structure of the cellular wall. On one hand, precast concrete is not popular in the building market of China. On the other, tubular structures have various applications in China, especially for lattice structures, lots of precedence can be found. It is easy to be fabricated and assembled.
2_ Rectangular hollow section profiles are chosen, because: most of the beam elements in this grid structure are under bi-axial bending and the torsional moments cannot be ignored. [Based on the analysis results in Chapter 5.3.3]
3_ The construction technology is mainly determined by the welding work – all the individual hollow section steel tubes are welded together. For in-situ welding, the elements must be carefully coded / coordinated for exact positioning. One proposal is to weld the steel tubes on separated frameworks, and then position and weld the large welded segments on site to finish the construction. This solution can make the welding work easier. Information of the wall surface can be generated for building these frameworks. [Chapter 6.1]

8.1.3 Structural design and evaluation

In this thesis study, two main parts of structural analysis were performed:

Part I – basic grid types study

Three basic grid types have been analysis and compared. [Chapter 5.3] From the analysis results, the following conclusions can be made:

1_The buckling capacity is strongly improved by the curvature of the facade:
The buckling load factors of the grid structures on curved panels are 15 times (for hexagonal grid; 8 times for rectangular grid and around 30 times for triangular grid) than on the flat panels.

2_ The grid types and orientation (related to the curvature of the surface) influence the stiffness properties of the grid structures:
Triangular grid has large in-plane and out-of-plane stiffness, thus it shows the highest buckling capacity in the case of curved panels (much higher than the other two grid types). Hexagonal grid has a relatively large out-of-plane stiffness but little in-plane stiffness, it also shows large buckling load factor. Rectangular grid (Note: orientated as beam-columns!) has low level in both in-plate and out-of-plane stiffness properties, it shows the worst buckling behavior – the smallest buckling load factor and a very large buckling area.
These stiffness properties have been further confirmed by linear-static-analysis. For example: wind load causes the smallest deformation in the triangular grid and hexagonal grid. The deformation of the rectangular grid is approximately 8 times larger. The hexagonal grid has the largest total deformation (combined load cases). This is caused by its small in-plane stiffness.

3_ Besides the stiffness properties and boundary conditions, the geometry of the surfaces influence the (shell-like) buckling of the grid structure:
Grid structures built on the extruded surface - hexagonal grid has much better buckling capacity then the rectangular grid. But when the actual geometry of the facade (including the ‘outward’ incline level and the slope of the upper boundary) is applied to the grid structure, the hexagonal grid doesn’t show this advantage any more (see Page 69).

4_The failure modes of the grid structures are different:
For the hexagonal grid, the deformation and the strength determine the dimensions – as design criteria; while for triangular grid and rectangular grid, the buckling problem is critical – buckling load determines the dimensions. (Chapter 7.7)

In the specific load cases of the cellular wall, the main load is the weight of the green roof (vertically loaded on the wall), while wind load has relatively small proportion. Thus, a pure hexagonal (cell-like) grid is not a very economic structural form: very weak in-plane stiffness, carrying large bending moments. Material cannot be used in a very efficient way; therefore the resulting structure will be quite heavy. Solutions to efficiently increase the total in-plane stiffness is recommended, for example, design special joints (nodes) to create rigid connection which can restrict the rotation, combine the pure hexagonal grid with triangular grid in the structural topology, etc.

Part II – Actual design conditions: cell-like grid optimization by parametric models

In this design case, the objective of the optimization is efficiency of the material utility, which means the structural elements are settled according to structural requirements. The optimization strategy is meaningful, since the geometry and topology of the structure is irregular, and the imposed loads are also not uniformly distributed. Experiments and comparison shows some level of efficiency has been achieved in the modified structures.

1_ structured grid models (Chapter 7.5.1)
Without modification or optimization, the grid structure with regular size/cell-density results in un-evenly distributed forces and high-level stress. When change Rsc to introduce some margin of randomness, some grid elements deform (some grid deformation might cause unfavorable shape) in the unloaded structure, therefore the resulted deformation when loaded is worse than the regular grid.

2_a_ modified structured grid, by inserting triangular elements (Chapter 7.5.2)
The grid structure with inserting triangles was stiffer than the original model. More moment-free joints thus the material can be used in more efficient way. But in most of the cases, when triangles are interested to the network, stress concentration might occur nearby, which makes it difficult to control the inserting. Several iterations (modify grid – structural analysis – feedback cycle) are needed to optimize the interesting.

2_b_ modified structured grid, by locally doubling up hexagonal elements (Chapter 7.5.2)
By locally double up the hexagonal grid, the grid structure also got stiffness to restrict the deformation. The stress distribution was more even than the original structure, and with lower values. The implementation of double-up grid is easier than inserting triangular elements. The modified pattern with local double rhythm has appearance of fractal geometry.

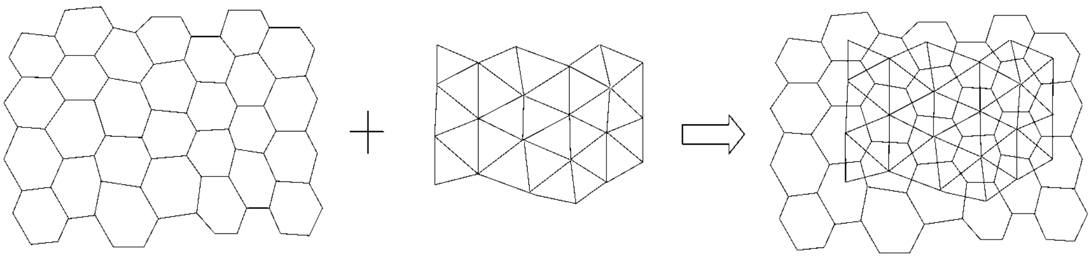
3_ unstructured grid models (Chapter 7.5.3)
When the local densities of the grid structure are fine tuned up with the imposed load cases (structural requirements), the material will be used in an efficient way – can be read from the analysis results – better forces distribution and low stress level. The evenly distributed reaction forces are also good for the foundation. The local densities/grid sizes are changed smoothly, which brings nice design aesthetic.

4_ Member design – apply different cross-sections (Chapter 7.6)
Another method is to apply different profiles (cross-sections) for individual beam elements according to the structural requirements. Although it will bring extra requirements for construction – carefully coded and stored etc, this approach provides quite an efficient structure.

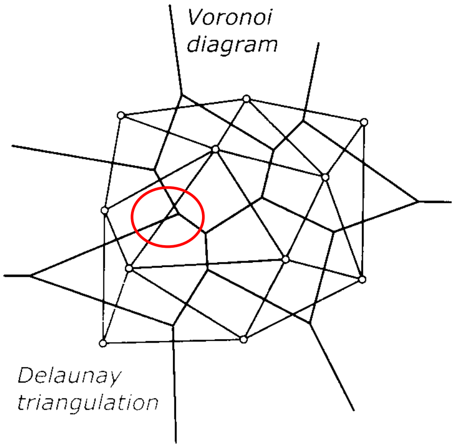
Other solutions

In the modified structured grid models, by locally cutting-out triangles will cause stress concentration, which cannot efficiently increase the total stiffness. A suggested method is to corporate Voronoi diagram with the associated Delaunay triangulation, efficiently getting advantages of the stiff triangle components.

Example Process includes the definition of two point-sets:
[**Black**] Point-set 1 - grid points to generate the original cell-like grid (Voronoi Diagram)
[**Red**] Point-set(s) 2 - grid points (selected from Point-set 1) to generate the local interested Delaunay Triangulation



Note: Solution should be found to prevent unfavorable ‘cross’ between Voronoi diagram and the associated Delaunay triangulation (an example is showed in the figure below).



8.2 Recommendations

8.2.1 SNHM projects

- **Design conditions**

The design conditions in this thesis study are based on several assumption and decisions, for example the glass layer has been integrated with the primary structure by point supported system – which brings extra loads from the weight of glass panels. If any one of these conditions is changed, the structural design should be modified. In that situation, parametric design shows advantages, the complete process doesn't need to be change, but quickly repeat the generation, analysis and evaluation.

- **Joint design**

Designing of the joints for the cellular wall structure is a complex task, which was not included in this thesis study. Some optional designs were sketched (see Appendix). Further study and detailing design for the joins (large bending moment) should be done, to enhance the in-plane stiffness of the grid and increase the efficiency of material utility.

- **Further checking**

1_ Horizontal reaction forces at the connection with the green roof: Verify if the roof and the shear wall on other side can provide a fully horizontal support to the cellular wall.

2_Vertical deformation |Uz|: |Uz| might cause the cracking problem on the other side of the roof or the mid-span columns. It should be evaluated, but not yet verified by standards in this thesis study.

3_ Member buckling: To reduce the weight on steel, smaller wall thickness plates should be adopted in the design of box section. If the width-thickness ratio of steel plates for welded thin-wall box members is quite large, local buckling on the compressed plates occur easily. This analysis should also be implemented.

4_ Influences by the local floors: Local floors that attach to the cellular wall will create intermediate supports and bring extra loads to the grid structure.

- **Geometric tricks**

Some geometric tricks have been proposed (chapter 5.5), for creating irregular patterns, but in the mean time, reducing the different structural elements. These solutions can be involved in the grid structure generation. Further investigation can be performed to implement some of these in the geometry. However, these geometric tricks cannot be easily cooperated with the optimization concept - 'adaptive pattern' by structural requirements.

8.2.2 General grid structures design

Grid structures are popular in modern architecture, and a large part of the special structures nowadays are grid structures. Therefore, more systematical study could be performed.

- **Structural behaviors**

In 2D grid structure (grid on flat panels), the grid types and orientations determine the in-plane stiffness property and the forces trajectories. When the surface is changed to a curved panel, the determination of the stiffness properties becomes more complicate. The relationship between the orientation of the grids and the curvature of the surface significantly influence the out-of-plane stiffness. Limited items are taken into account in this thesis study, and some of the unexpected results are not given in-depth explanations. Further research could be performed to get more in-sign knowledge and experience of these kinds of grid structures.

- **Materials and construction technologies**

Because the specific design conditions gave some clear clues, decision was made in advance for the structural material and construction technologies in this thesis study. More research could be performed to investigate new materials and construction technologies for similar grid structures. It would be useful in other grid structures design in the future.

8.2.3 Adaptive pattern

This thesis proposed a concept of 'Adaptive pattern'. The basic purpose was to explore a grid structure in which all the elements were tuned up by different design constrains.
_ Create a parametric model which can be generated by user's pre-defined rules/inputs
_ Build up design exploration diagram and network of all the constrains
_ Implement the parametric model with inputs from the constrains network, to get 'optimal' structures

- **Generation tools**

Several strategies have been experimented to generate flexible patterns, there are some points can be improved in each of them:

1_ Circle packing strategy
Only 2D generations were experimented, no complex surface was introduced. Further experiments could be performed to apply this strategy to free surfaces. One drawback of the 2D circle-packing algorithms is the memory cost – the data structure should be improved.

2_ Attract & repel strategy
In the experiment, the points behave according the same rules and cannot adjust them. Further investigation could be done by introducing intelligent agents, which can be able to learn from the environment. In that case, more geometric principles could be included in the generation process.

3_ Mapping tool
The uv mapping tool (described in Chapter 6.4) was created according to the specific geometric rules of the surface. It is not the only solution. Therefore, other mapping principles can be found.

- **Design Exploration**

A Multi-criteria/design exploration by different constrains was the basic method of "Adaptive pattern". However, focus on the structural optimization, other design constrains were not fully taken into account in the design process of this thesis study. Further research for "Adaptive pattern" could be performed to better use these methods.

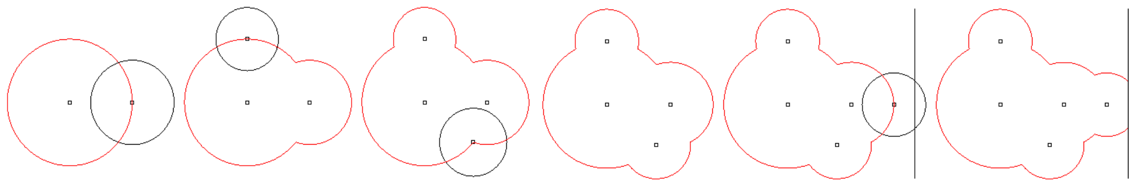
Appendix A: Algorithm and Pseudo-code of the 2D Circle-packing Experiment

Objective of the 2D circle packing experiment

The objective is to generate a point-set within a certain region, according to predetermined local (point-point) distances. At each position i [x,y] or [u,v] of this region, the point-point distances ($D_{default}$) are predetermined. The procedure attempts to add one point at a time looking to satisfy the condition $D_p \approx D_{default}$, where D_p is the distance between the currently tested point and its nearest neighbor;

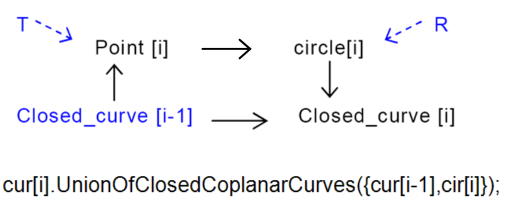
Circle packing 1

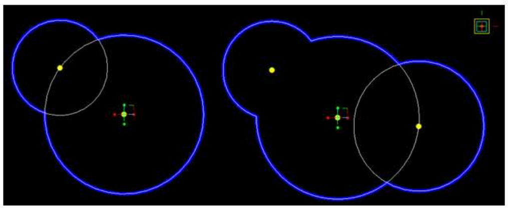
Generation procedure is illustrated in the figure bellow. Every time when a new grid point is created, a corresponding circle will be draw. The radius of the circle will be determined by the location of this grid point (center of the circle). The difficulty of this strategy is to define the 'closed-curve' (**Red**), which is determined by the outer circumference of all the draw circles, as the efficient path to generate new grid points.



```
< Pseudo-code >
int Np;         // the number of points we want to generate
Draw the first point and circle;
Void Newpoints() {
        If (closed-curve == true && i< Np) {
                i++;
                Create a random point i on closed-curve;
                Create circle i;        // center is point i, radius by point i.position(Ddefault)
                Calculateclosed-curve(point i, circle i);       // update a new close-curve
                Newpoints();        // recursion: continue the generation, until there is no efficient
                                       closed-curve the outer circumference is out of the boundary or
                                       required number of grid points has been created;
        }
```

Implementation in GenerativeComponents is represented in the figures bellow: One feature called 'UnionOfClosedCoplanarCurves' was used to define the close-curve [**Blue**]. With help of this feature, it was easy to build up the iteration loop. But it can't be implemented to many iteration cycles to deal with large amounts of points/curves, because of the memory limitation.

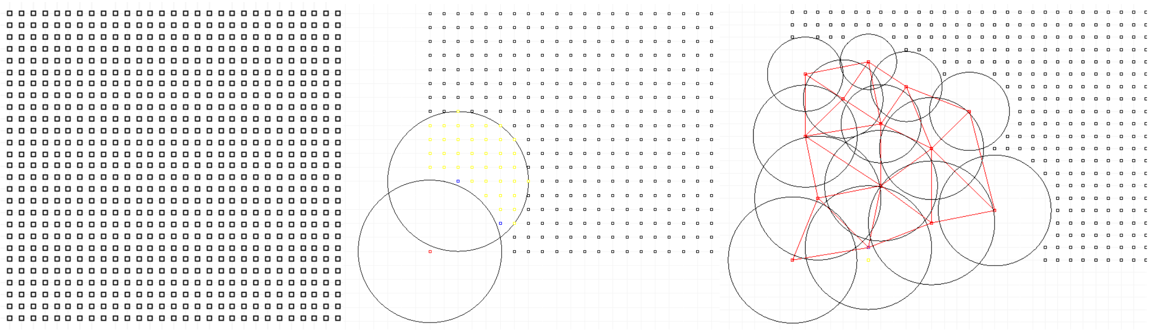




Point.byParameterAlongCurve (T)
Circle.byCenterRadius (R)
Curve.byUnionOfClosedCoplanarCurves

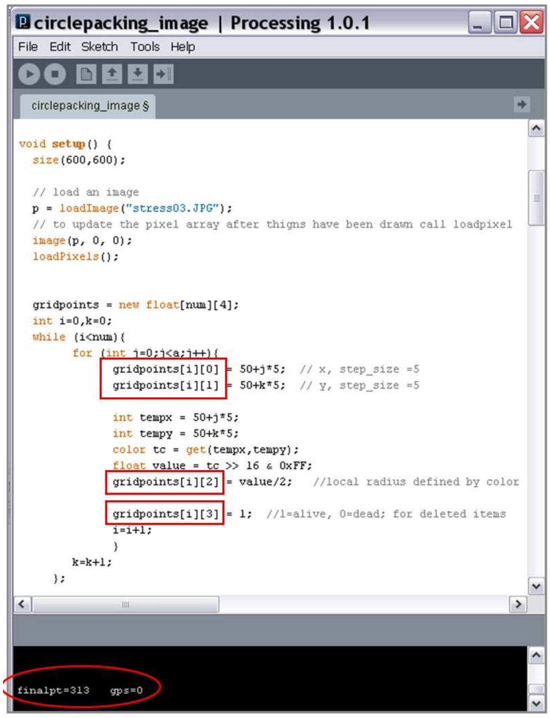
Circle packing 2

In this case, the rules were the same as the former method, but an inverse procedure - selection and removal from the default grid point, as illustrated in the figure bellow.



3 point-lists were defined in this method:
DefaultPoints[] – [**Black**] its items will be removed when they are located inside the generated circles, and it will be a null list after all the selection and removal;
FinalList[] – [**Red**] items that have been selected from default grid as effective points will be stored in this list, to provide the resulted point-set after iteration;
ClosestPoints[] – [**Blue**] every time when a new point is selected and a corresponding circle is created, a group of closest points (closest to the new selected point) will be updated, and the next selection will be from this group

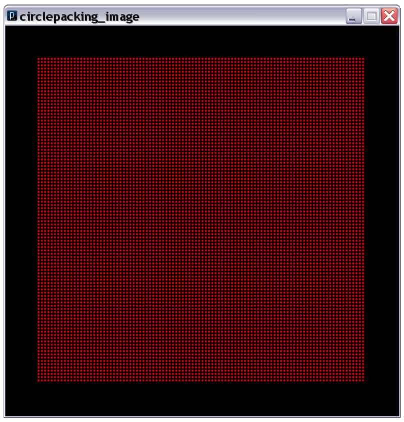
```
< Pseudo-code >

List/Array   DefaultPoints[]    FinalList[]   Current_ClosestPoints[]   New_ClosestPoints[]
Initialize DefaultPoints[];    // generate default grid points
Initialize New_ClosestPoints[];    // by select one point from DefaultPoints[]

Void Newpoints() {
        if (New_ClosestPoints[] != Null) {
                Current_ClosestPoints[] = New_ClosestPoints[];
                while (Current_ClosestPoints[] != Null) {
                        Point() {
                                select one point from Current_ClosestPoints[];
                                add to FinalList[];        // add point[i]
                                }
                        Circle() {
                                center = point[i];
                                radius = point[i].position.D_default;
                                }
                        Check() {
                                Distance = point[i] to each remaining grid points;
                                if (Distance < Radius) Delete remaining grid points from all the lists;
                                update New_ClosestPoints[];  //find the closest neighbor(s) to point[i];
                                }
                                                }
                Newpoints();     // recursion
                                }

// after the loop, report the DefaultPoints[] ('Null') and FinalList[]
```



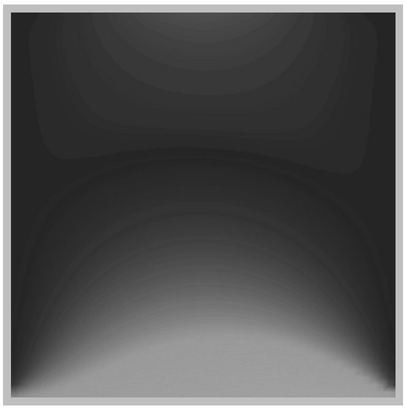
Figures: Implementation in processing

Input information
For each grid point (gridpoints[i]), there are 4 items to define its properties:
gridpoints[i][0] – x coordinate
gridpoints[i][1] – y coordinate
gridpoints[i][2] – local radius signed by color value from background image
gridpoints[i][3] – exist or not, for deleting items 1=alive, 0=dead (inefficient point)

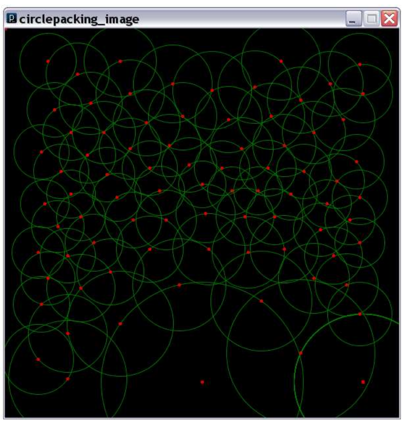
Reported information
finalpt=313
Resulted grid points in the FinalList[]
gps=0
Remaining grid points in the DefaultPoints[]
– after efficient generation, should be ‘0’



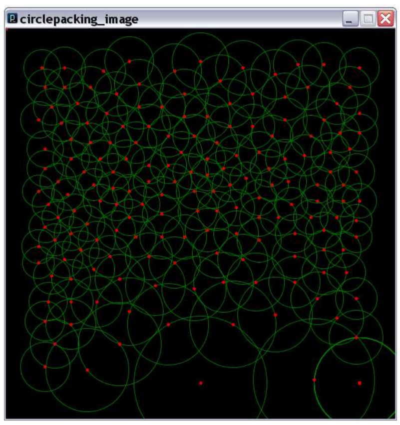
Default grid points



Background image (stresses graph)



Output I

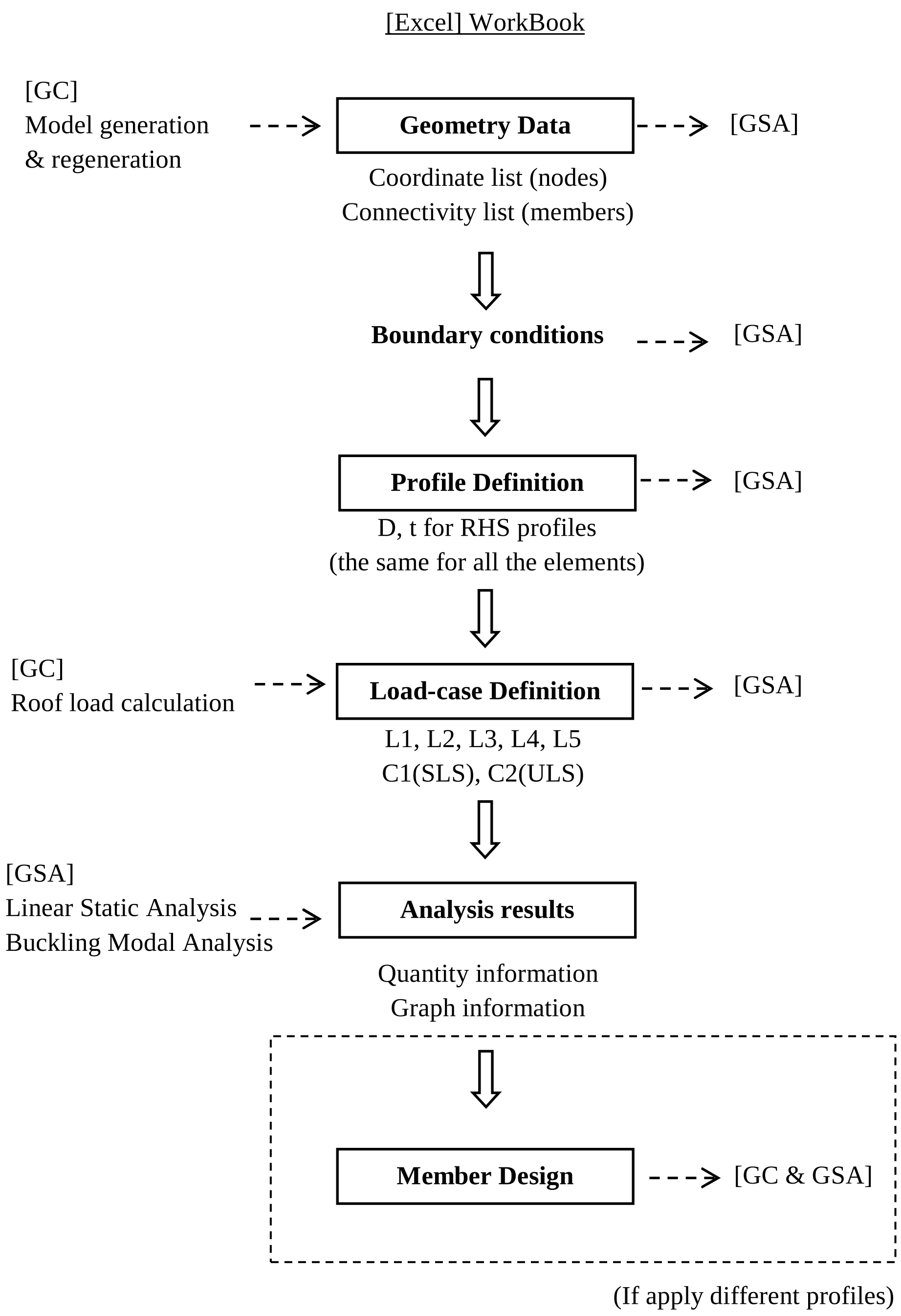


Output II

Sample: in this case, the local distances were determined by color values – the final point-set distribution was corresponding to the color gradient. A gray image (top, left) was used as background; color value of each position in this image was get and signed to the local distance (radius). The rule between color value and radius can be defined freely, to get different grid densities (bottom, left & right).

Appendix B: Inputs and Outputs of the parametric design cases

Work Flow



A combined [Excel] WorkBook was created for each parametric model, to aid the whole process – from model generation to analysis results – interfacing the modeling tool [GC] and structural analysis tool [GSA].

Example: Implementation of unstructured grid model

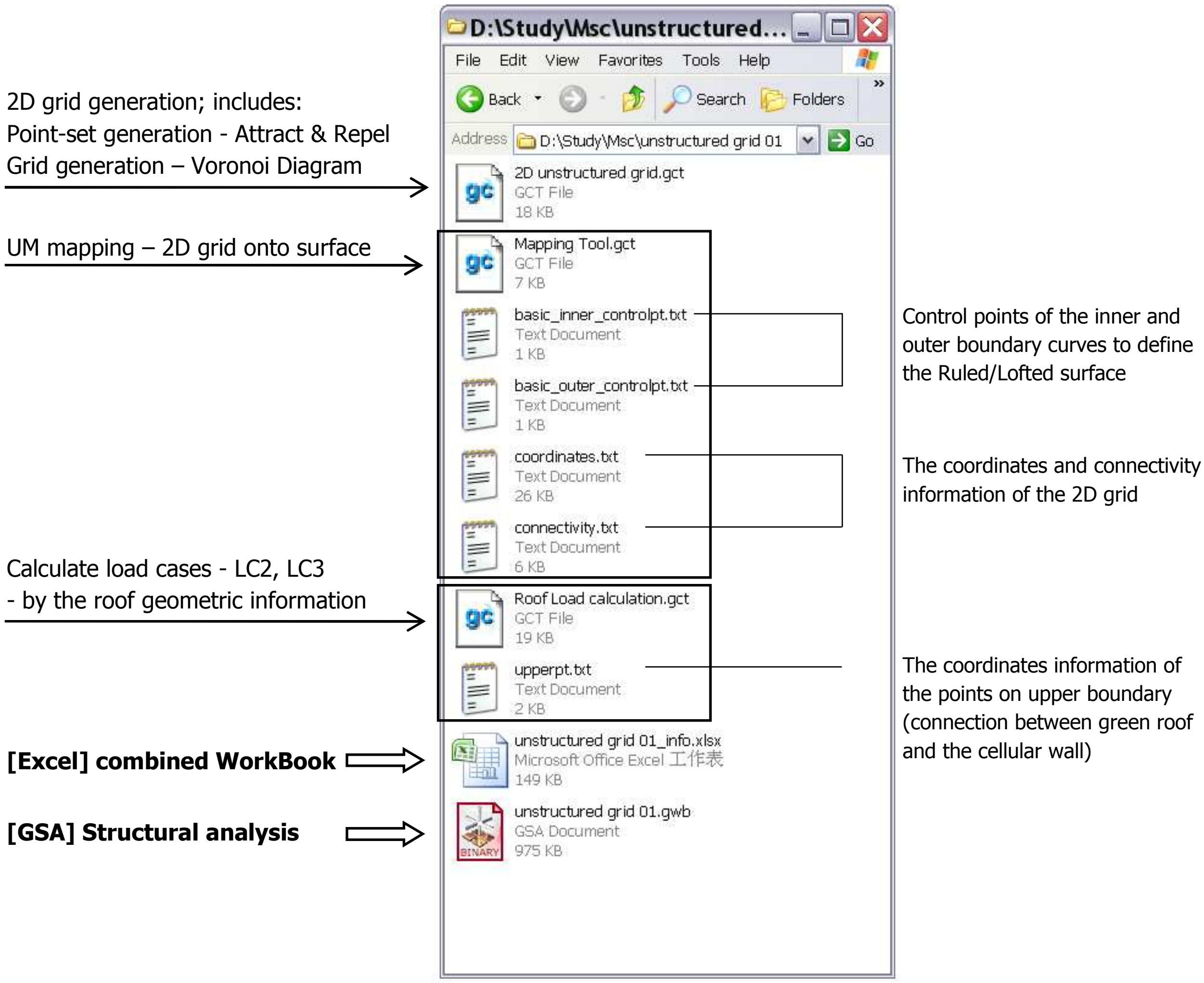
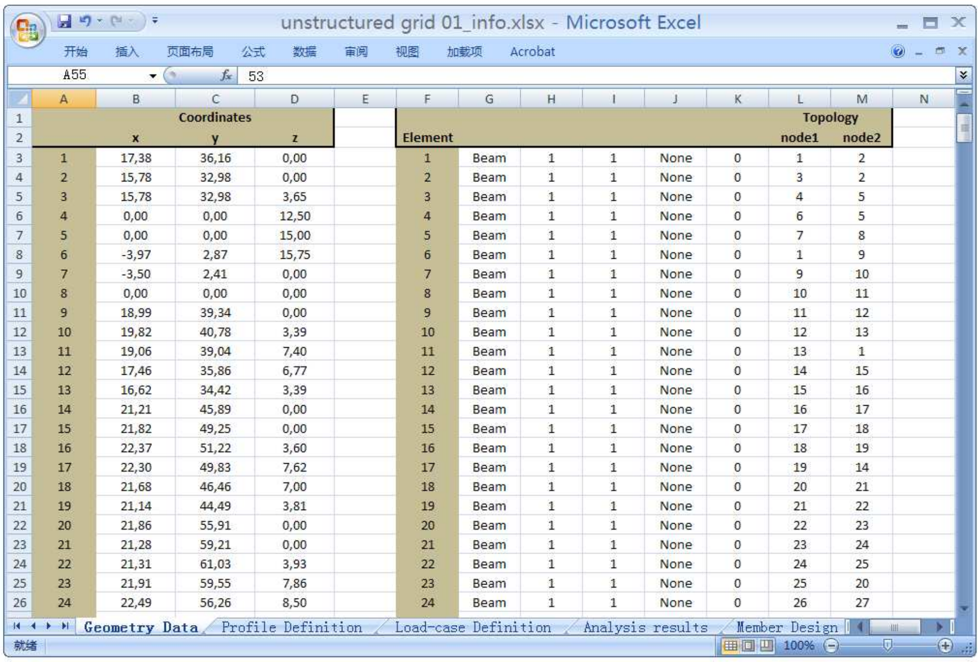
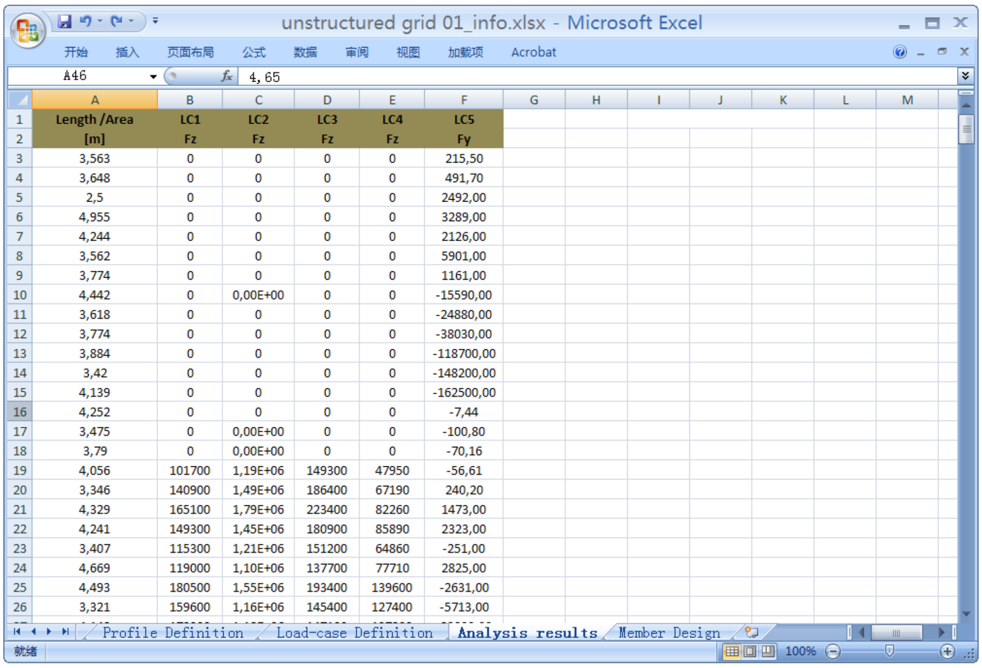


Figure above shows the documents used in this parametric model. Geometry data (and relative information) from GC was written to the Excel WorkBook, and the required inputs were prepared in this WorkBook for GSA. Analysis results from GAS were recorded into the WorkBook.

Note:
For the current version of GenerativeComponents (Version 08.11.05.36), it's easier to read point files and connectivity files from txt format. All the txt. files were prepared as input files for GC, to generate points (coordinates) and lines (connectivity).
For the outputs from GenerativeComponents, it's chosen to communicate GC with Excel via features 'WriteExcelRange'. By pointing to specific ranges of the Excel sheets, the data can be correctly recorded and directly applied to the other calculation.



	Coordinates				Element						Topology	
	x	y	z								node1	node2
1	17,38	36,16	0,00		1	Beam	1	1	None	0	1	2
2	15,78	32,98	0,00		2	Beam	1	1	None	0	3	2
3	15,78	32,98	3,65		3	Beam	1	1	None	0	4	5
4	0,00	0,00	12,50		4	Beam	1	1	None	0	6	5
5	0,00	0,00	15,00		5	Beam	1	1	None	0	7	8
6	-3,97	2,87	15,75		6	Beam	1	1	None	0	1	9
7	-3,50	2,41	0,00		7	Beam	1	1	None	0	9	10
8	0,00	0,00	0,00		8	Beam	1	1	None	0	10	11
9	18,99	39,34	0,00		9	Beam	1	1	None	0	11	12
10	19,82	40,78	3,39		10	Beam	1	1	None	0	12	13
11	19,06	39,04	7,40		11	Beam	1	1	None	0	13	1
12	17,46	35,86	6,77		12	Beam	1	1	None	0	14	15
13	16,62	34,42	3,39		13	Beam	1	1	None	0	15	16
14	21,21	45,89	0,00		14	Beam	1	1	None	0	16	17
15	21,82	49,25	0,00		15	Beam	1	1	None	0	17	18
16	22,37	51,22	3,60		16	Beam	1	1	None	0	18	19
17	22,30	49,83	7,62		17	Beam	1	1	None	0	19	14
18	21,68	46,46	7,00		18	Beam	1	1	None	0	20	21
19	21,14	44,49	3,81		19	Beam	1	1	None	0	21	22
20	21,86	55,91	0,00		20	Beam	1	1	None	0	22	23
21	21,28	59,21	0,00		21	Beam	1	1	None	0	23	24
22	21,31	61,03	3,93		22	Beam	1	1	None	0	24	25
23	21,91	59,55	7,86		23	Beam	1	1	None	0	25	20
24	22,49	56,26	8,50		24	Beam	1	1	None	0	26	27



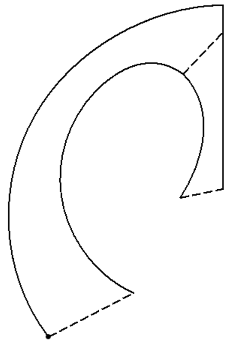
Length /Area	LC1	LC2	LC3	LC4	LC5
[m]	Fz	Fz	Fz	Fz	Fy
3,563	0	0	0	0	215,50
3,648	0	0	0	0	491,70
2,5	0	0	0	0	2492,00
4,955	0	0	0	0	3289,00
4,244	0	0	0	0	2126,00
3,562	0	0	0	0	5901,00
3,774	0	0	0	0	1161,00
4,442	0	0,00E+00	0	0	-15590,00
3,618	0	0	0	0	-24880,00
3,774	0	0	0	0	-38030,00
3,884	0	0	0	0	-118700,00
3,42	0	0	0	0	-148200,00
4,139	0	0	0	0	-162500,00
4,252	0	0	0	0	-7,44
3,475	0	0,00E+00	0	0	-100,80
3,79	0	0,00E+00	0	0	-70,16
4,056	101700	1,19E+06	149300	47950	-56,61
3,346	140900	1,49E+06	186400	67190	240,20
4,329	165100	1,79E+06	223400	82260	1473,00
4,241	149300	1,45E+06	180900	85890	2323,00
3,407	115300	1,21E+06	151200	64860	-251,00
4,669	119000	1,10E+06	137700	77710	2825,00
4,493	180500	1,55E+06	193400	139600	-2631,00
3,321	159600	1,16E+06	145400	127400	-5713,00

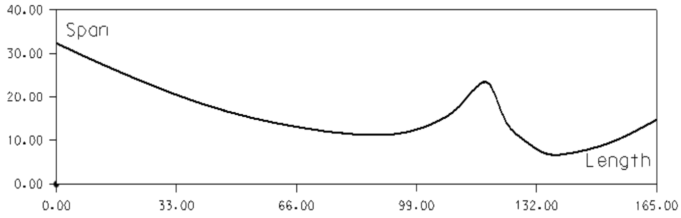
Fig. Sample sheet of the Excel WorkBook

Note: GSA has the ability to make use of ‘COM Export Functions’ to communicate with Excel and vice versa. ‘Command-control’ can be investigated to build up auto interface between GSA and Excel. But in this Msc thesis study, the interface was done by hand.

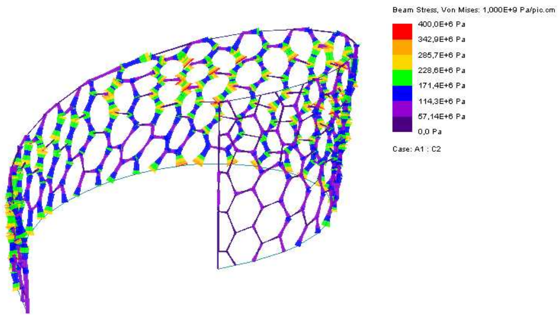
In the point-set (grid points) generation by Attract & Repel model, the definition to local distances were referenced to:

- the roof load distribution, since it has the largest proportion in the design load cases



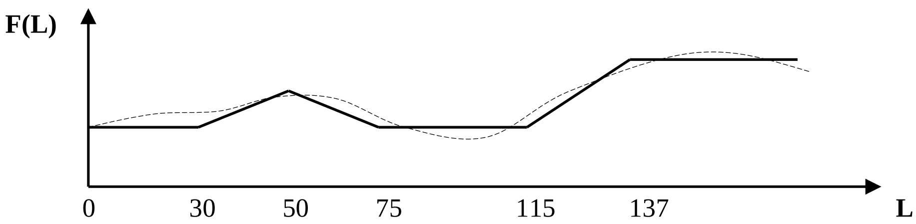


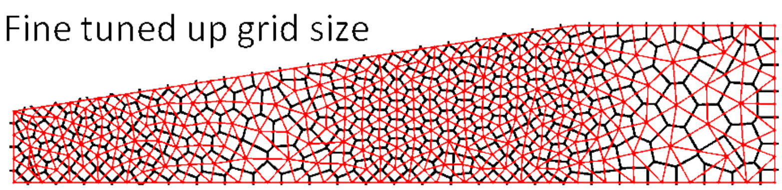
- the analysis results of regular grid, which shows the material usage



To make sure the speed of the iteration and reduce the computer memory cost, the definition of the local (point-point) distances were simplified by piecewise function, instead of curve. For example:

$$F(L) = \begin{cases} 4.5 & L \le 30 \\ 4.5+(L-30)/10 & 30<L\le 50 \\ 6.5-(L-50)/12.5 & 50<L\le 75 \\ 4.5 & 75<L\le 115 \\ 4.5+(L-115)/5.5 & 115<L\le 137 \\ 8.5 & L>137 \end{cases}$$

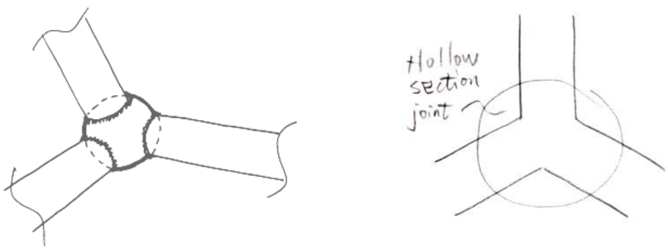




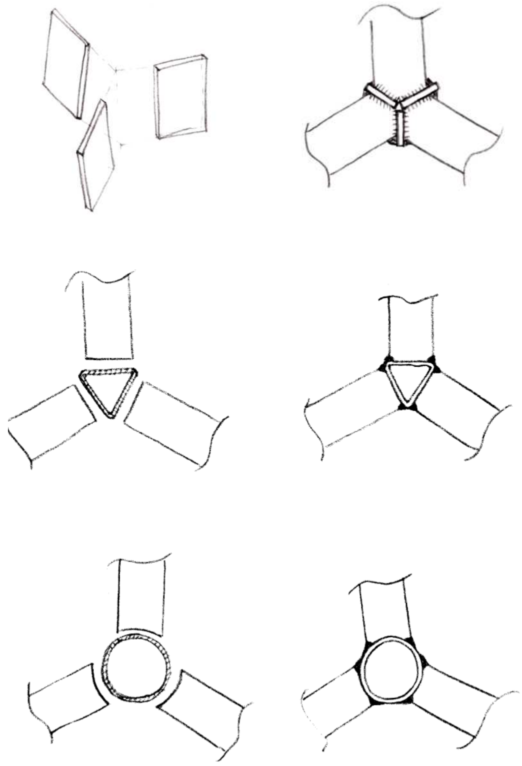
Appendix C: Sketches of detailing (connections)

Nodes of the primary grid structure

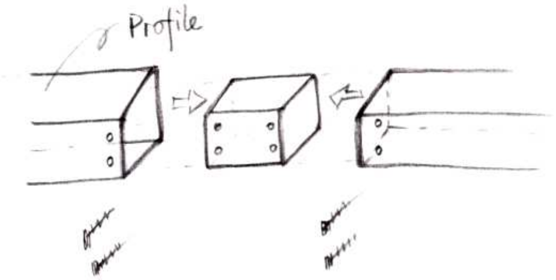
The joints of the cell-like grid structure are very important, since the largest combined forces occur at these nodes. Besides the structural requirements, the 3D prototype of the joints is also quite complicate. Connectivity between the **straight elements** is difficult due to the curvature of the surface, for example, there will be some requirements to the gaps (in-between) for welding.

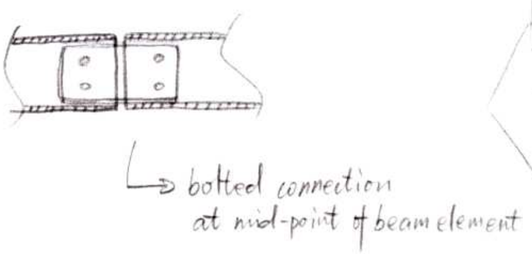


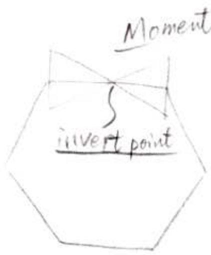
Solid ball connection is very easy for construction complicate 3D prototype of such an irregular grid structure. But in this case, because of the large dimensions of the beam element (the width of the RHS around 0.5m), the connections (nodes) cannot be solid. Several prototypes of the hollow section joints are exampled by sketching. The continuity of steel plates can make sure the efficiency of the load transfer in the joints.



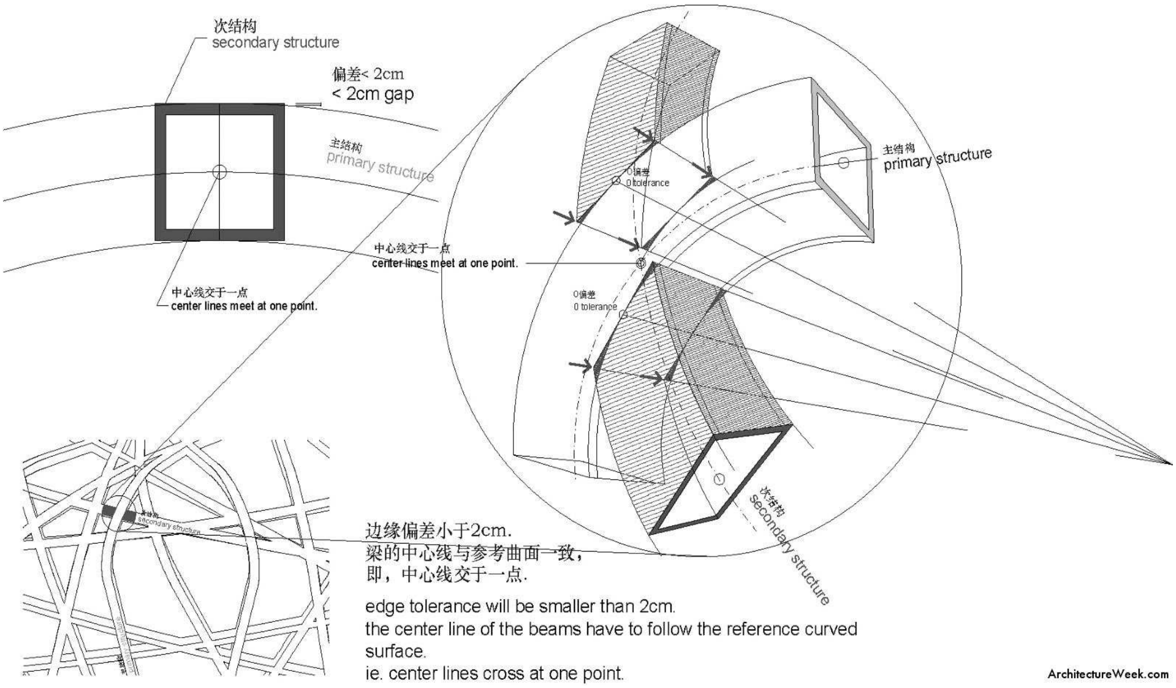
In addition, the beam elements can also be connected by bolted connection in the middle point of the beams, where the invert point of the moment locates.







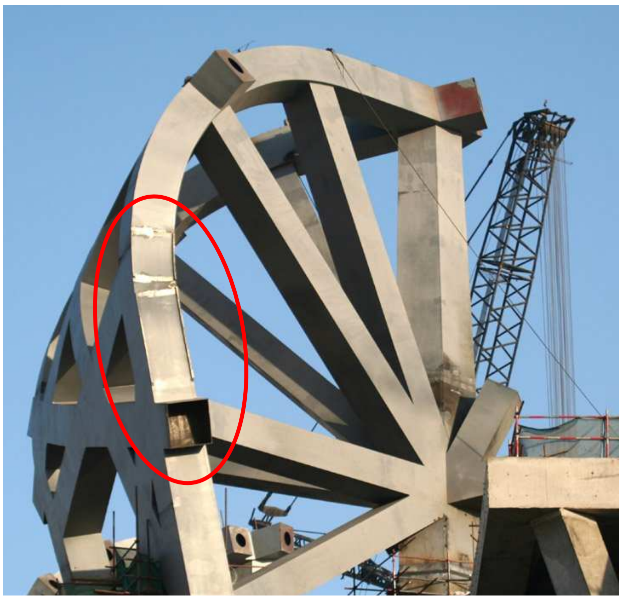
Nodes of the Bird’s Nest (Source: Architectureweek.com)



In the project of Bird’s Nest, most of the steel beam elements (RHS) are curved and/or twisted.

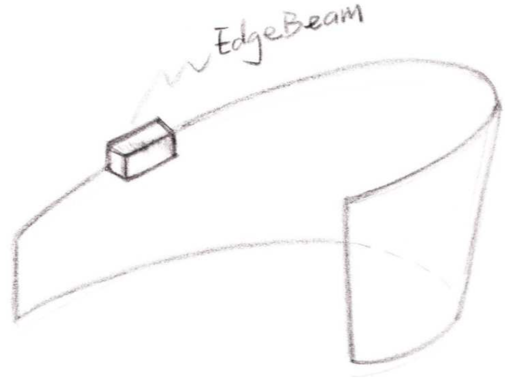
The edge tolerances (gaps) between the primary structure and secondary structure are controlled within 2cm, while the beam elements have dimensions of 1m width and 20mm thickness.

From the figure (right), the continuity of the beam elements go through nodes (cross-points of RHS profiles) can be found.



Green Roof to the cellular wall

Since the detail design of the green roof and the construction procedure is not yet determined, the detailing of the connections is only conceptual.



Edge beam at top was chosen for the grid structure – to create a tensile ring and transmit the forces between the roof and the grid structure.

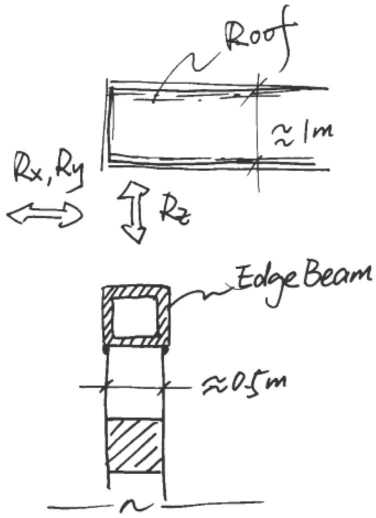
A concrete roof structure was proposed (see Chapter 2.2 & Chapter 7.4). For this concept, three sample connections are sketched here:

1_ by anchor encased in concrete

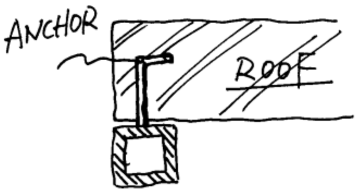
2_ by steel bar cooperated with the reinforcements in concrete roof

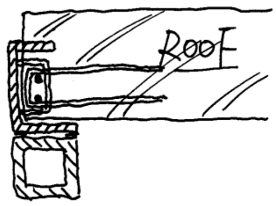
(In these two cases, the construction of the concrete roof will be finished after the installation of the grid structure.)

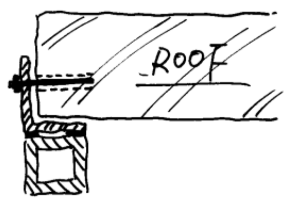
3_ prefabricate concrete roof, connected by screws/studs to the edge beam

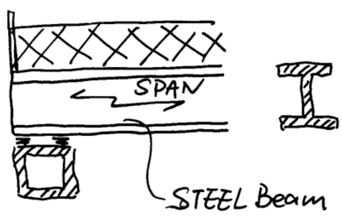


If steel beams over the roof span are chosen for the roof structure, the edge beam can be welded/fixed to the steel beams.







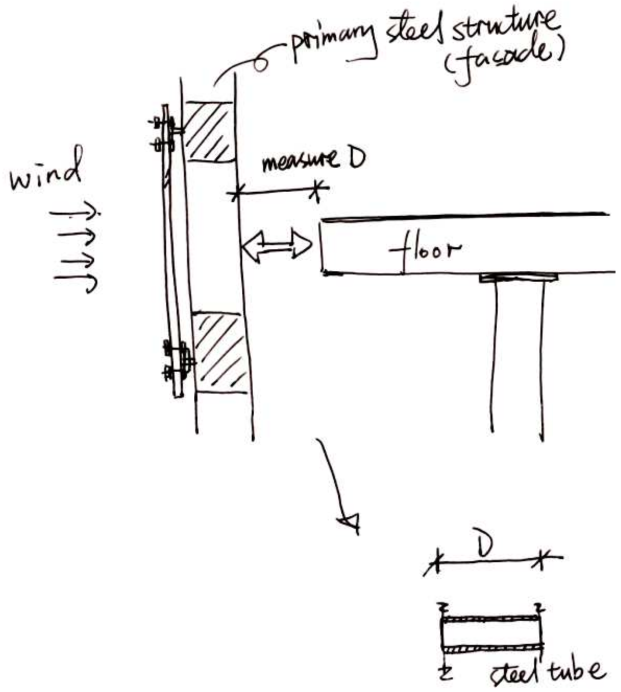


(Local) floors to the cellular wall

A proposed solution to the connection between local floor structure and the cellular wall is:

- keep them separated until the construction is finished and deformations of the structures are settled down under the permanent loads
- measure the distances between the local floor structures and the wall, attach the floor to the wall with horizontal supports

Thus, the floor structures won't bring significant extra load to the cellular wall, but only create horizontal supports to deal with wind load.



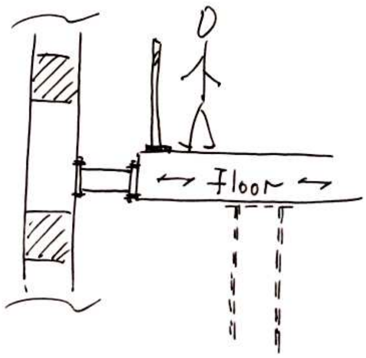
Reasons:

In one hand, as for shell structures, intermediate supports are not favorable. Intermediate supports also have little contribution to the grid structure, but result in large local forces.

— provide horizontal support for wind load

— no extra load from the floor to facade

In another, consider the adaptability of the building, advice was made to create a clear distinction between a primary structure and a secondary structure (see Chapter2.2.1). The cellular wall is a main component of the primary structure, while local floors belong to the secondary structure.



Cellular wall to the ground (foundation)

To create a neat and nice view, it is advised to connect the grid structure to the foundation without applying any visible edge beam.

Two examples are sketched here:

1_ simple connected to the foundation with steel plate and anchors (as figure)

2_ encased to the foundation pier

Edge beam (steel/concrete) can also be applied, but hided underground.