

Parametric Design of steel structures

Fundamentals of parametric design using Grasshopper

By using parametric modelling tools, designers can simulate a high number of design-variants with relatively low effort. Visual programming software's like Grasshopper significantly lowers the threshold for beginners to start building parametric models. This paper discusses how using Grasshopper as a parametric design tool can be applied by structural engineers to improve well-informed decision making, when designing a steel structure. Finally, worth mentioning further possibilities parametric design workflows can bring are discussed such as optimisation strategies, optimising sustainability and including structural connections in the parametric model.

Keywords parametric design; parametric engineering; visual programming; Grasshopper; Karamba3D; steel structures

1 New basics for engineers

Parametric design is a phrase which is becoming more prevalent in the Architecture Engineering and Construction industry (AEC). It allows the designer to generate and analyse multiple variants, which speeds up the decision-making process and drastically increases the capacity of the work of an engineer. In making better informed decisions, parametric design is inevitably becoming a fundamental skill for structural engineers.

The aim of this introductory paper is to help the industry towards a more data-oriented future and to provide engineers better insight of parametric design based on the course “Parametric design for steel structures”, which has been developed by the Dutch Steel construction Institute Bouwen met Staal. In this course the fundamentals of modelling a parametric steel structure are thought out in a step-by-step approach such that knowledge is gradually build up. In this paper, a short overview of the fundamentals of parametric design for steel structures including

straightforward examples for demonstrating purposes will be given.

1.1 Visual programming

A significant contributor to the popularity of the phrase Parametric design are visual programming packages. Visual programming is a no-code type of programming where code is encrypted into components. Each of these components has inputs and outputs. With wires these components can be connected to each other. Forming a network of logic where the output of one component serves as the input for the other. This network of logic processes input into a defined output. Compared to conventional programming, visual programming is significantly more intuitive, making it relatively easy to learn for beginners.

In the AEC there are two main visual programming packages, Grasshopper (Fig. 1) and Dynamo (Fig. 2). Grasshopper works together with Rhino, is geometry driven and computes quickly. Dynamo works together with Revit, is object driven and because these objects contain a bigger amount of information, computing takes longer. Furthermore, there are some visual differences between the packages, Grasshopper makes use of vivid colours and logos where Dynamo looks more serious. For this introduction Grasshopper is chosen for its quick computing. This is an important feature since it will react rapidly and in doing so it provides beginners instant feedback, whether the defined logic works as desired.

Learning visual programming can be seen as a intermediate step into understanding how computers process data and consequently making it more easy to be able how to program, see Fig. 3. Most engineers are accustomed to working with Excel for run-of-mill engineering

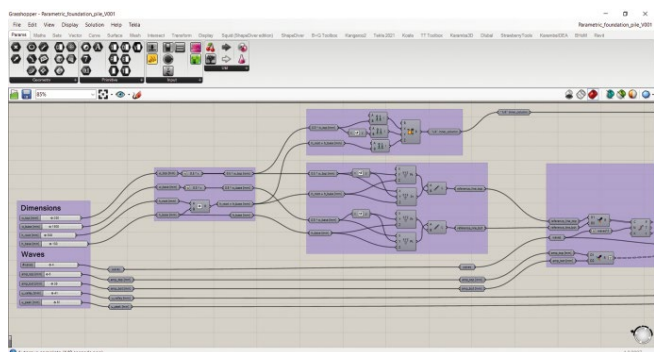


Fig. 1 Grasshopper

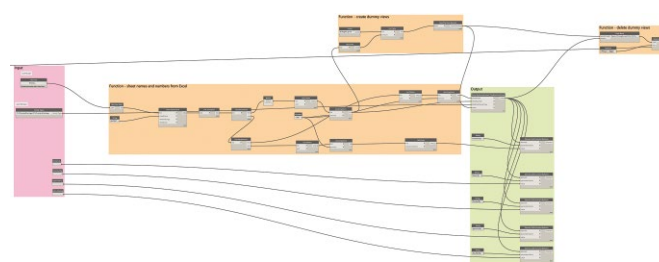


Fig. 2 Dynamo

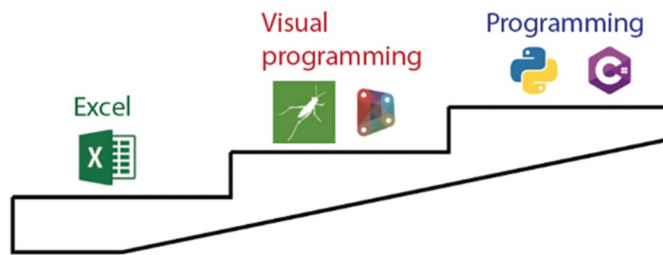


Fig. 3 “Staircase” of increasing possibilities and complexity

processes. Where Excel is limited, visual programming will give you an extra set of tools to make well-informed design decision. The same applies for visual programming: once experienced one will face its limits. Programming will then be an interesting alternative, which will be much easier to adopt since one is familiar with computational concepts and processes.

1.2 The scope of parametric design for steel structures

Visual programming software, like Grasshopper, have many different types of applications. This paper focuses on how structural engineers can use Grasshopper to design steel structures. These steel structures are often described by using points and lines (see Fig. 4). We shall further focus here on such geometries. Hence surfaces, which are often used to describe concrete structure, as displayed in Fig. 5, will not be covered here. Although, having good basic knowledge about simple lines structures will make it easier to learn how to work with more complex geometry such as surfaces.

1.3 Fundamentals of making a parametric model for structural engineers

All geometry discussed in the examples of this paper, shall be created to be compliant for finite-element-method-calculations. This means a different approach is used compared to simple drawings with a generic CAD program. Firstly, lines are defined from node to node. Typically, in a CAD tool drawing a straight line will be a single straight-line segment, see truss in Fig. 6. This prevents that the intersection nodes are being found and will result in the truss not behaving as desired in the analysis. Secondly, in finite element packages curves do not exist, there are only (straight) 1D elements. Therefore, when an arch is modelled, this arch must be discretised into multiple line segments.

1.4 Computational thinking

In parametric design one creates the logic which will generate the desired structure. Instead of drawing the anticipated final structure, one will have to break down the creation of the structure into a series of steps to generate it. For example, a tube, which has a length

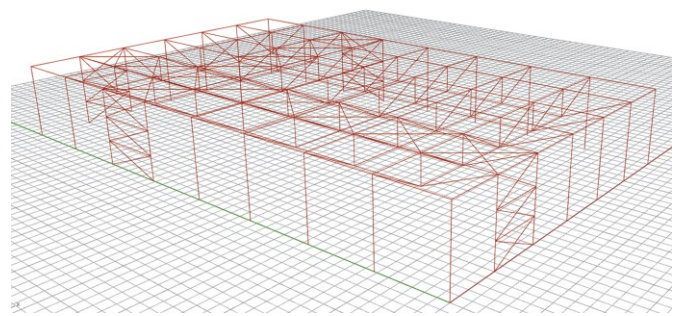


Fig. 4 Lines and points are covered here

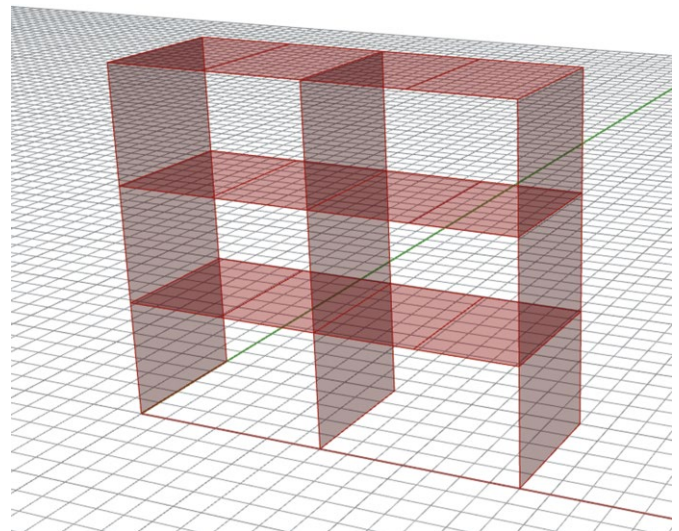


Fig. 5 Surfaces are not covered here

and a radius as shown in Fig. 7. On the right of this image the Grasshopper canvas is shown. Here, multiple components with input and output are shown which fulfil a step in order to generate the tube. The first step is to define the first point at the origin $\{0,0,0\}$, next a second point is modelled with a variable z-coordinate $\{0,0,z\}$. Between these points a line can be defined and lastly this line can be converted into a pipe with a set radius.

Breaking down your desired structure into series of computational steps or rules is referred to as computational thinking. The key to successfully create a parametric model is to be able to define the steps needed upfront. In this article two simple examples are discussed, where first the series of computational steps needed are defined and after these steps are scripted with Grasshopper components.

2 Parametric Geometry

Before a parametric engineering model can be created, parametric geometry is needed. In this chapter, two examples are shown where parametric geometry is created. In the ensuing chapter this geometry is used to create the parametric engineering model. In these examples the se-

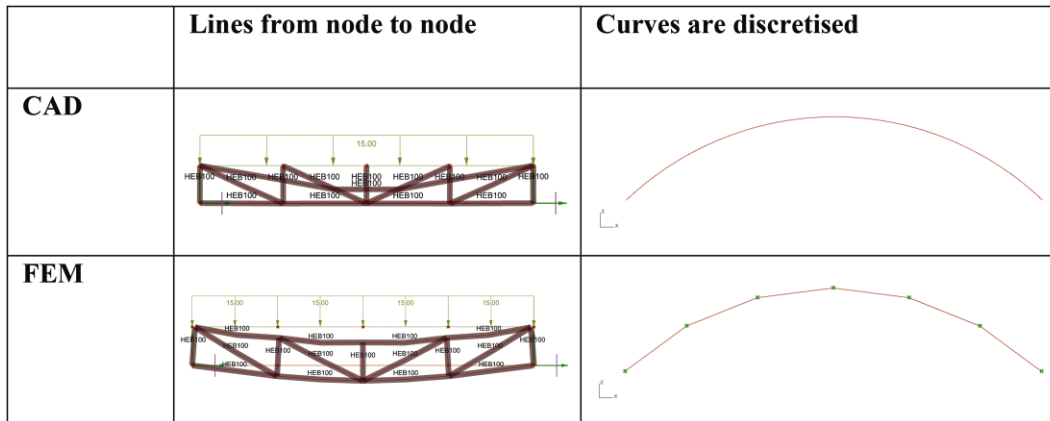


Fig. 6 Lines must reach from node to node, curves are discretised

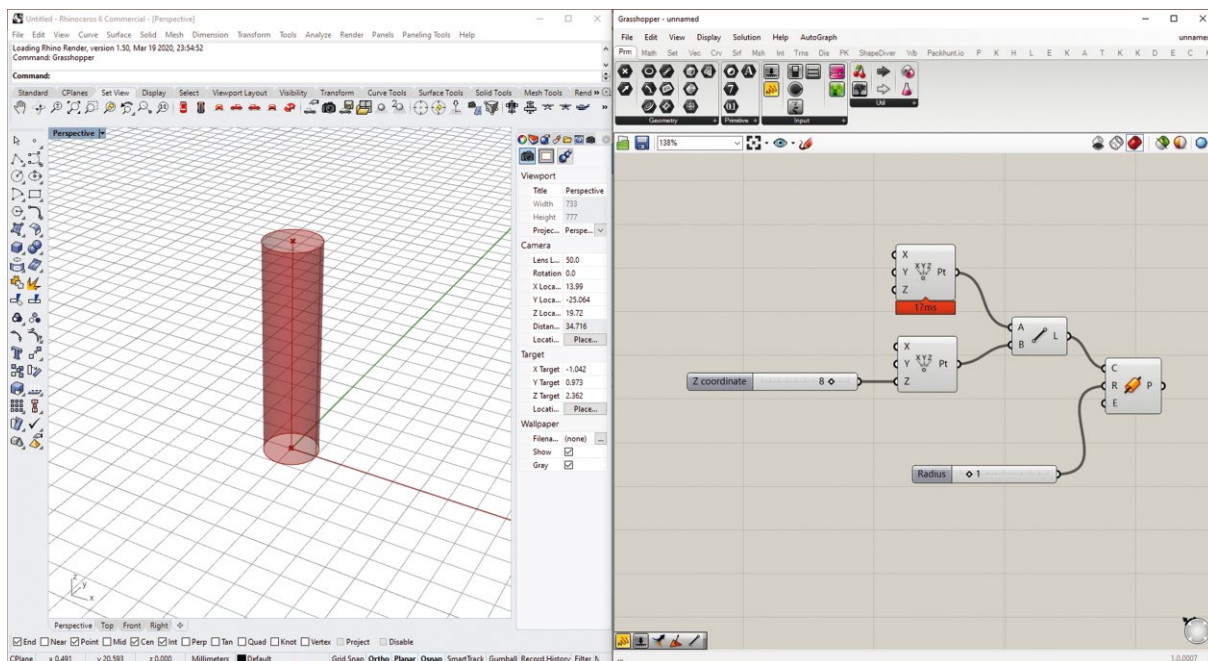


Fig. 7 Series of steps needed to generate a parametric tube

ries of steps needed to generate the final structure will be elaborated.

2.1 Arch

An arch structure shall be designed, but it is not yet known, which height leads to the most efficient structure. In such a case a parametric model can help gaining more insight into the multiple solutions. Firstly, the logic on how the arch is generated must be created.

An arch is defined by three points A, B and C, so first we need to define variable coordinates of these three points. When we want to create this arch in the XZ plane, we need coordinates accordingly. Point A will be at the origin so the default values of zero will suffice. Point C will have a X-coordinate equal to the length of the structure. Point B will have an X-coordinate equal to half of the length and its Z-coordinate will be equal to the height of the structure. Linking these three points to the Grasshopper

per component, named Arc_3pt, will produce a continuous arch (Fig. 8).

As mentioned before, we are not interested in continuous arcs, instead we need discretised arcs. Therefore, extra logic will be added to the script. Firstly, we will divide the continuous arc into a random number, for now we choose five segments. This will give us a list of six points. From this list we need to retrieve the start points and end points required to create the line segments. In this case we need 5 line-segments, so we need 5 start points and 5 end points. To get the start points we use the original list of six points where we remove the last point, by the index “-1”. To get the end points we remove the first point with index “0” (Grasshopper starts counting at zero). Lastly these two lists of 5 points can be linked to the “Line” component producing 5 line-segments (Fig. 9).

Finalising the last step results in a parametric model, where the length the height and the number of discretised line-segments are variable.

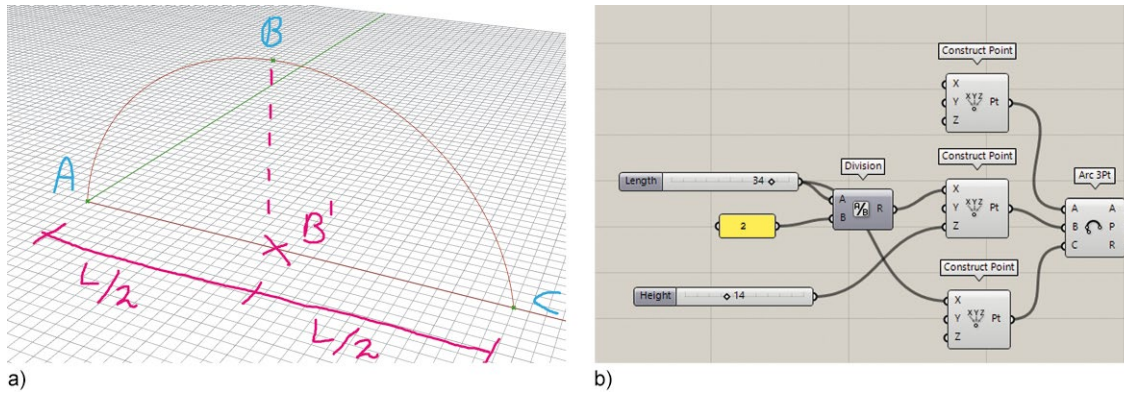


Fig. 8 Creating a parametric arc; a) computational plan, b) Grasshopper components

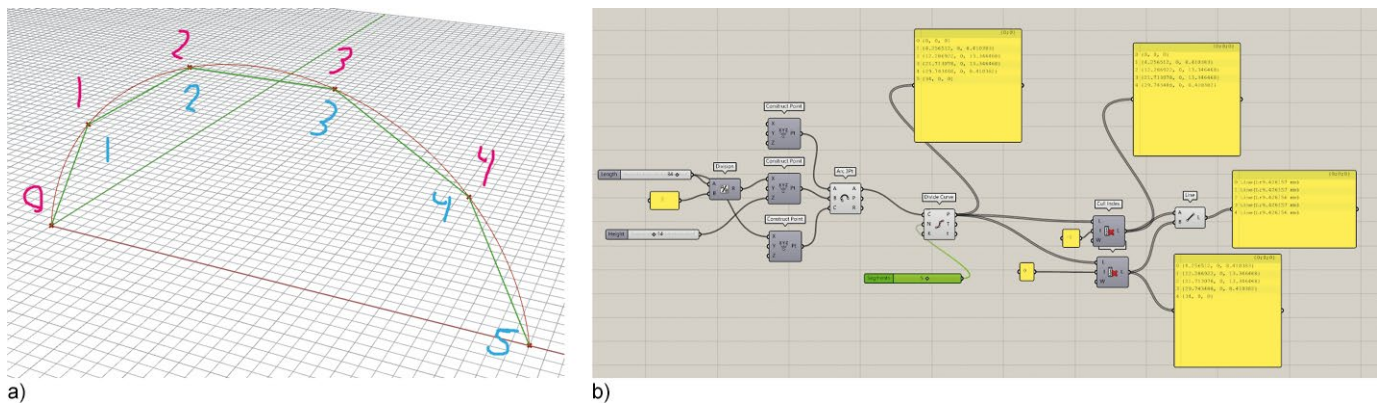


Fig. 9 Discretising the arc into multiple line-segments; a) computational plan, b) Grasshopper components

2.2 Truss

A more steel structure related example is a parametric truss. The aim is now to design a truss system for a certain span, where total weight is minimised. Here the best amount of truss segments and the best truss typology shall be derived to find the most efficient solution. In this example the parametric definition of a Vierendeel truss and an N-truss will be shown.

Designing the truss starts with defining the point grid system. In this case we will design the truss in the XZ-plane. First of all, we start with a point at the origin {0.0.0}, we move this point over the X-axis with a length equal to the defined by the slider defined length (blue). Between this points a line is created, this line is then divided into multiple segments creating a list of points representing the bottom nodes of the truss (pink). The list of points can also be moved in the Z-axis with a length equal to the defined height, which creates a new list of point representing the top nodes of the truss (yellow), see Fig. 10.

The point grid is now in place, so the next thing to do is defining the lines. First lines are being created where the start point are the bottom nodes and where the end points are the top nodes of the truss. Secondly, in the list of bottom nodes line segments are created in a similar procedure to the arch. Thirdly, the same procedure as previous is executed for the top nodes. These steps com-

bined will produce the parametric Vierendeel geometry (Fig. 11).

The Vierendeel geometry can be upgraded into an N-truss by including diagonals. To execute the diagonals as desired the correct start and end points needs to be defined. To make the truss symmetric in every situation the number slider, determining the number of possible truss segments, is set to only even numbers. For the N-truss diagonals may be present in two directions from top left to bottom right (blue) and from bottom left to top right (pink) (Fig. 12).

To achieve the right relationship between the points, the list of points at the top and the list of point at the bottom of the truss point grid are manipulated. To create the start points of the blue diagonals, the list of top points is split at a dynamically specified index, such that the required points remain. The definition of the index can be made parametric by including a formula based on the length of the list. For example: $(\text{List length} - 1)/2 = \text{index}$. For this example, an index of $(5-1)/2=2$ is found, meaning that the points with index 0 and 1 (output List A) will be split from the points with index 2, 3 and 4 (output List B). Next, the same procedure is made for the blue diagonals end points. To split the list of bottom points a different formula is used: $(\text{List length} + 1)/2 = \text{index}$. Resulting in $(5+1)/2=3$. From this list of three items the first item is deleted, which gives us the required end points of the blue diagonals.

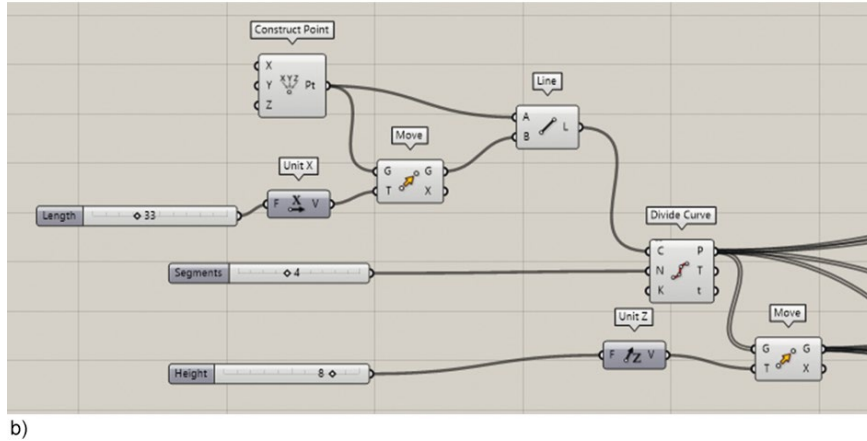
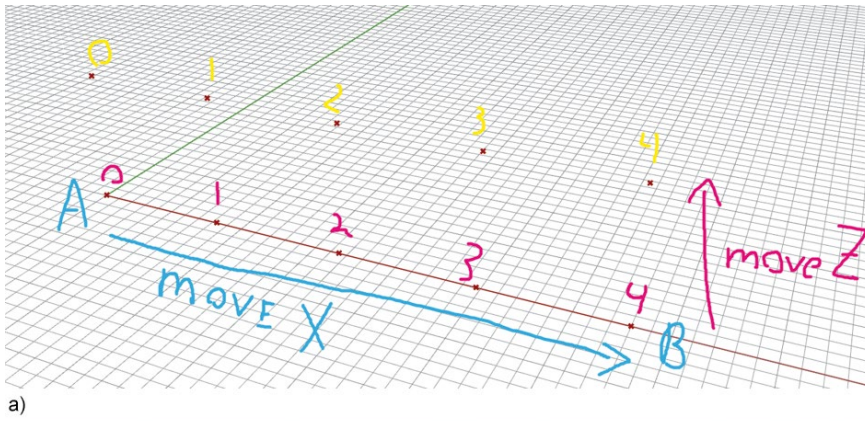


Fig. 10 Creating the point grid of the truss; a) computational plan, b) Grasshopper components

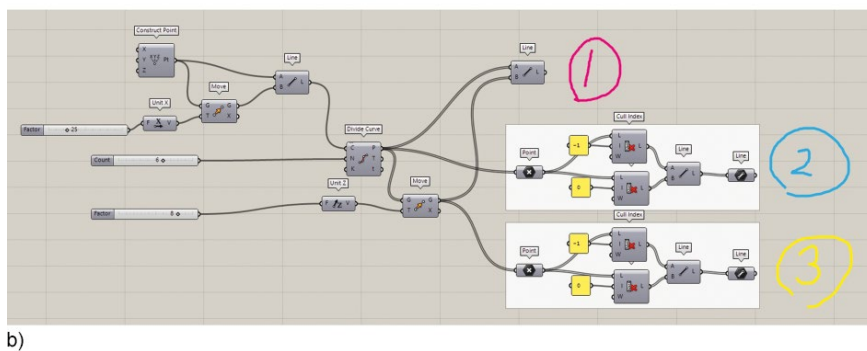
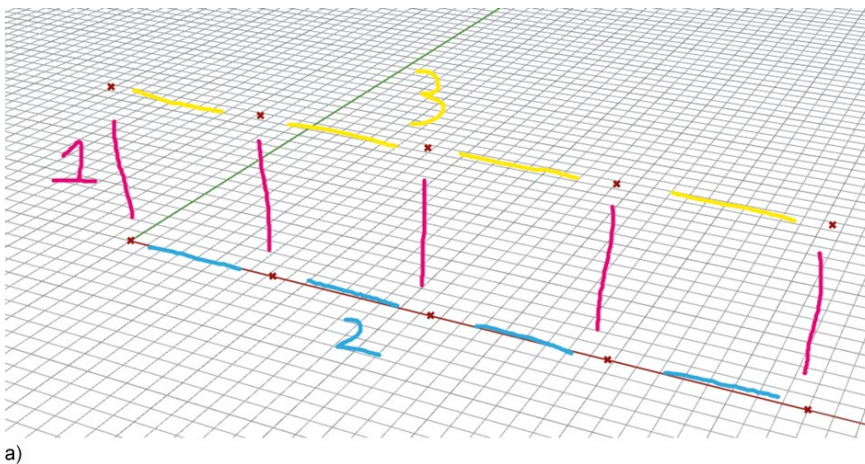


Fig. 11 Line definitions for the Vierendeel truss; a) computational plan, b) Grasshopper components

The pink diagonals are created by performing the exact same steps, but in contraction to the blue diagonals now output list B supplies the required points.

The result is a parametric N-truss where the height, length and number of segments are variable. Next, a filter can be included separating the geometry from the Vieren-

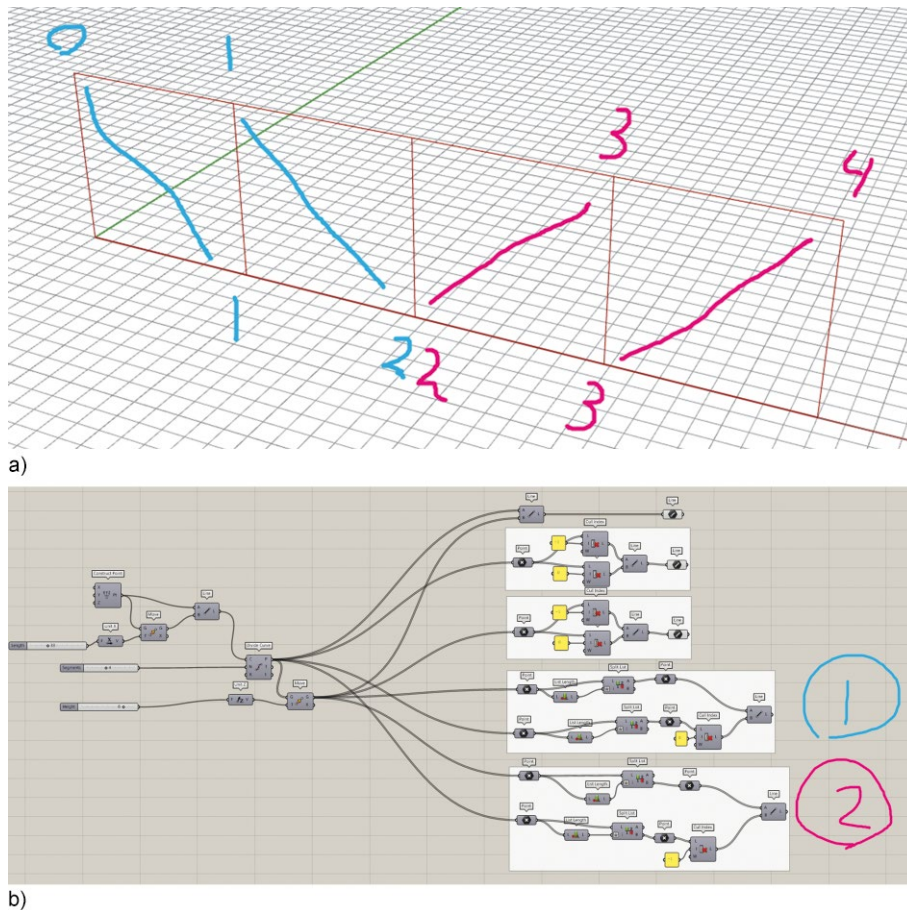


Fig. 12 Line definitions for the N-truss diagonals; a) computational plan, b) Grasshopper components

deed and the N-truss. Giving an additional variable for the truss typology.

Using the same steps other truss typologies can be defined as well. Besides the Vierendeel and the N truss also the Z, M and W truss may be included in the parametric model, see Fig. 13.

So far straightforward examples have been given to demonstrate how a parametric geometry can be created by defining a series of procedural steps. To recap, for parametric models point grids are created, subsequently these point grids are modified and lastly relationships between the points are defined generating the desired line geometry. More complex 3D structures can be developed by using similar procedural steps. For example, a steel hall, a curved tribune stand or a curved twisting tower (Fig. 14).

3 Parametric Engineering

The parametric geometry definition is the input for the structural analysis model. Karamba3D is a grasshopper plug-in, which is used to do structural analyses with finite elements [1]. With different Grasshopper-components this analysis model can be assembled. Like any other finite element software, the same information is needed to assemble the engineering model: geometry, sections, steel grades, loads and boundary conditions. Only the line elements representing beam members and the nodes repre-

senting the support from the already build parametric geometry definition must be linked (Figs. 15, 16). Since the set-up of assembling the analysis model is generally the same, it can be expedient to build a template which can be reused and modified easily.

3.1 Arch

The arch, described previously, can now be expanded into a parametric structural analysis model. The discretised line segments and the points representing the support are connected to the predefined Karamba3D template (Fig. 17). In the template, HEB300 is selected for the cross-section, supports are restricted in X, Y, Z and phi X (to prevent out of plane rotation), lastly the structure is loaded with a uniform line load of 50 kN/m.

A span of 10 m is assumed. With this analysis model we can answer the predefined question:

At which height of the arch are bending moment at their minimum?

To answer this question a “brutal force analysis” is performed on the model with Colibri, part of the plug-in TT toolbox [2]. Colibri saves the required results for every simulated variant in one Excel database. In this case, the maximum (hogging) and minimum (sagging) bending moment for the different arch heights. These results may be

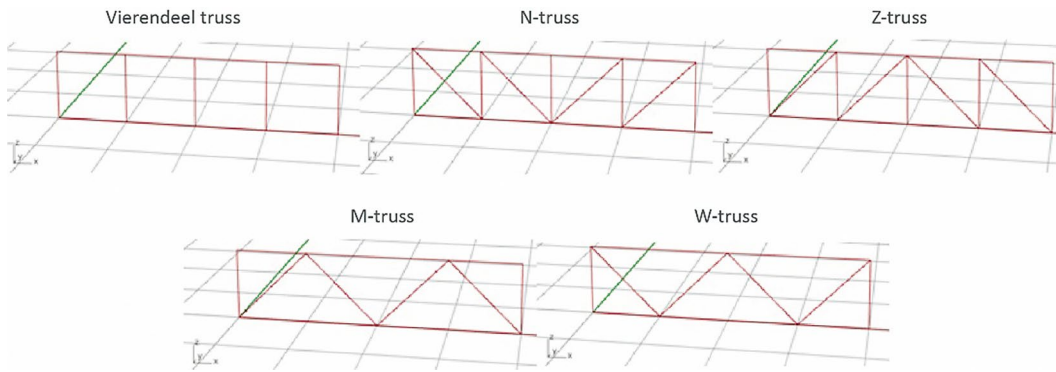


Fig. 13 Truss typologies

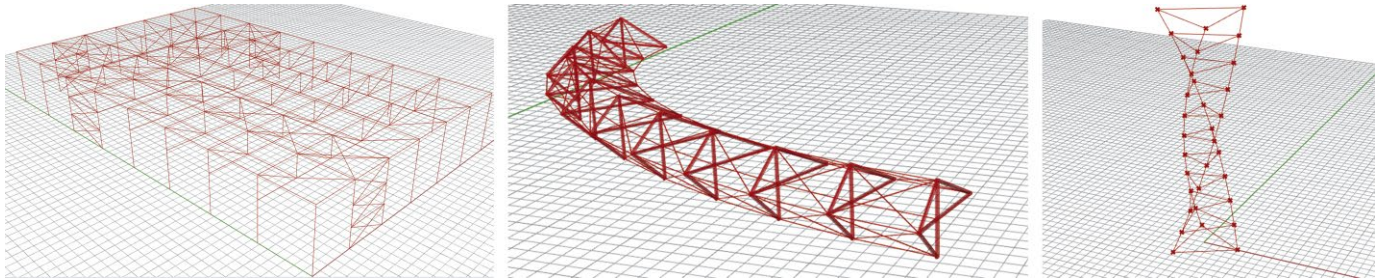


Fig. 14 Parametric geometry of complex 3D structures

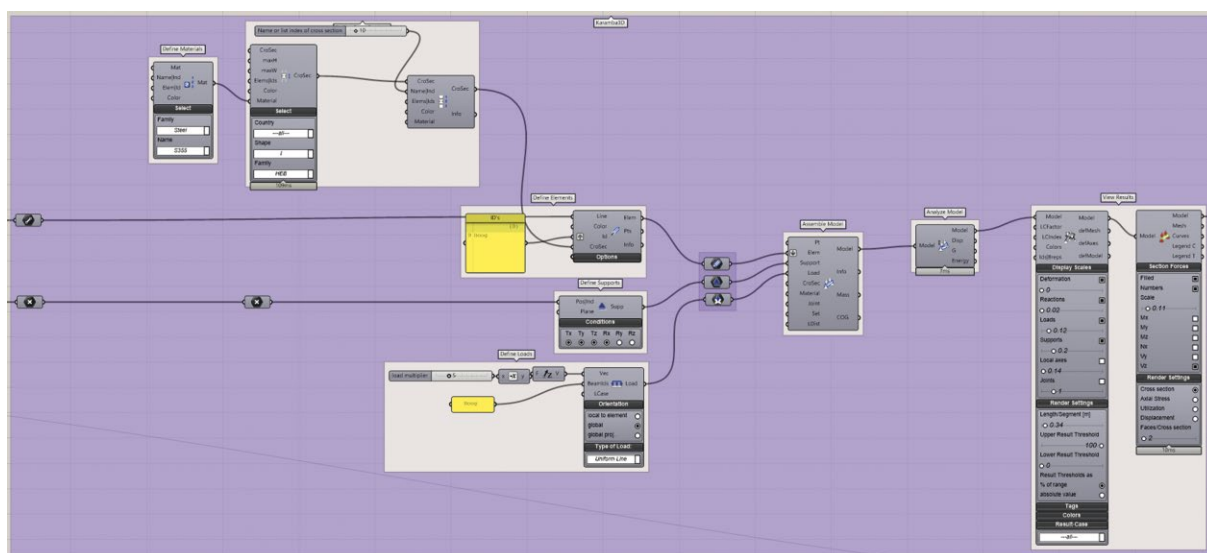


Fig. 15 Karamba3D's components required for building the analysis model

plotted into one graph giving insight in which height is considered optimal.

The graph in Fig. 18, shows the smallest sagging moment of -12.9 kNm can be found at a mid-span height of 1.6 meter.

3.2 Truss

Next, the truss described previously, can similarly be expanded into a parametric structural analysis model (Fig. 19).

In contradiction to the arch, the first support is fixed and the second is a rolling support. Instead of a list of line seg-

ments a tree (list with sub lists) of line segments forms the input. Every sub list in the tree represents a different group of elements. For example: Top chords, bottom chords, vertical and diagonal elements are separately grouped in these sub list.

I-sections cross-sections are assigned for the top and bottom chord members and rectangular hollow sections are assigned to the verticals and diagonals members, both in S355. A size optimisation algorithm is included that optimises the sections based on the loads acting on the element. Connections in the truss are assumed to be rigid and buckling lengths are equal to the element's length (default value). Lastly, the truss will be loaded on the top chord with a uniform distributed load of 30 kN/m .

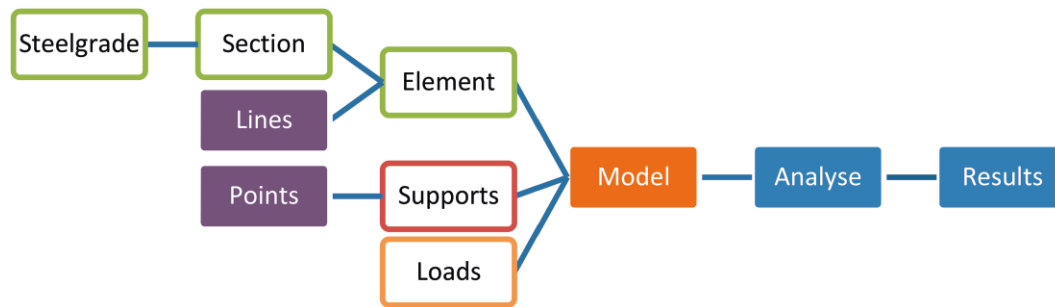


Fig. 16 Abstract relationship definition of information needed for the analysis model

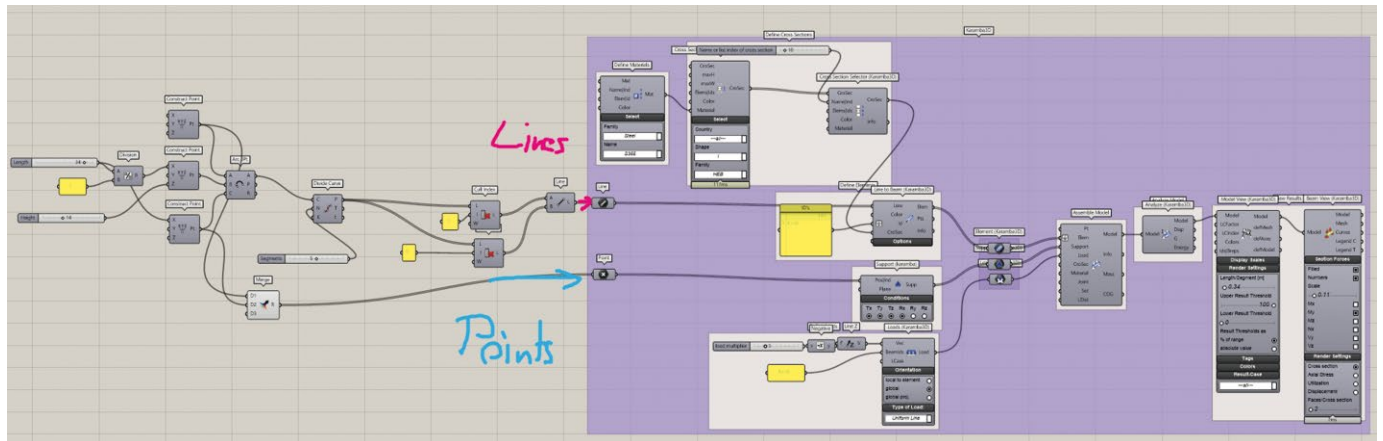


Fig. 17 Parametric engineering model of the arch

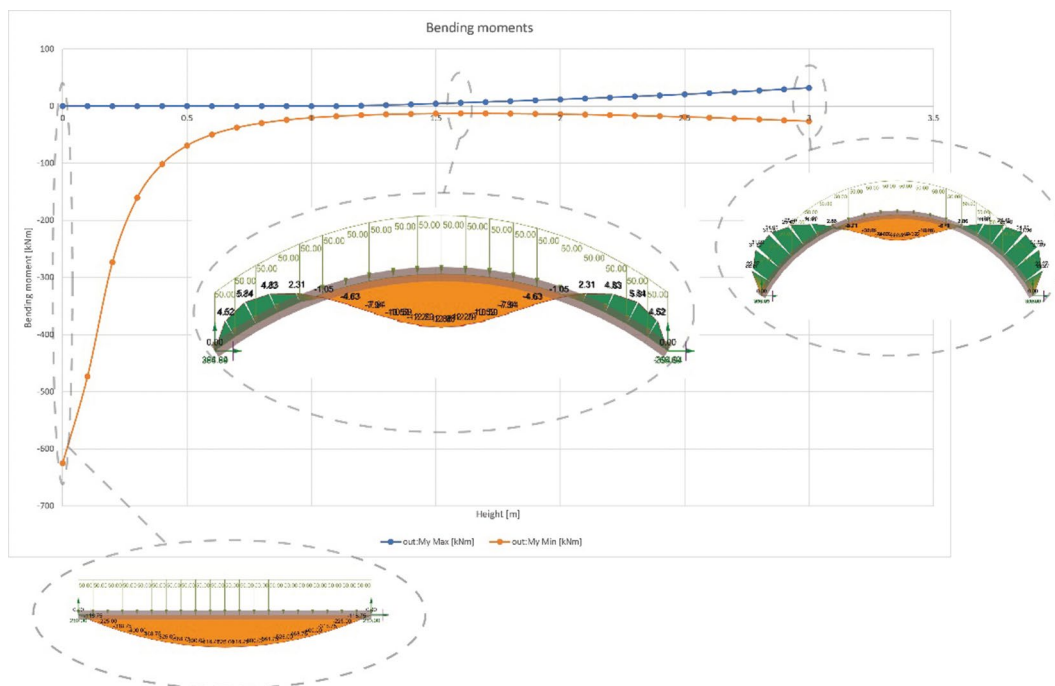


Fig. 18 Arch results

A fixed span of 30 meter and a fixed truss height of 1.5 meter is assumed. With this analysis model we can now answer the predefined question:

Which truss typology and number of truss segments will lead to the minimum total weight?

With help of Colibri a “brutal force analysis” is executed where every possible variant is simulated. The result data of the analysis can be visualised one graph (Fig. 20). In this graph every line represents a different truss typology with its total weight presented on the y-axis. From this graph we can find the minimal weight truss, which

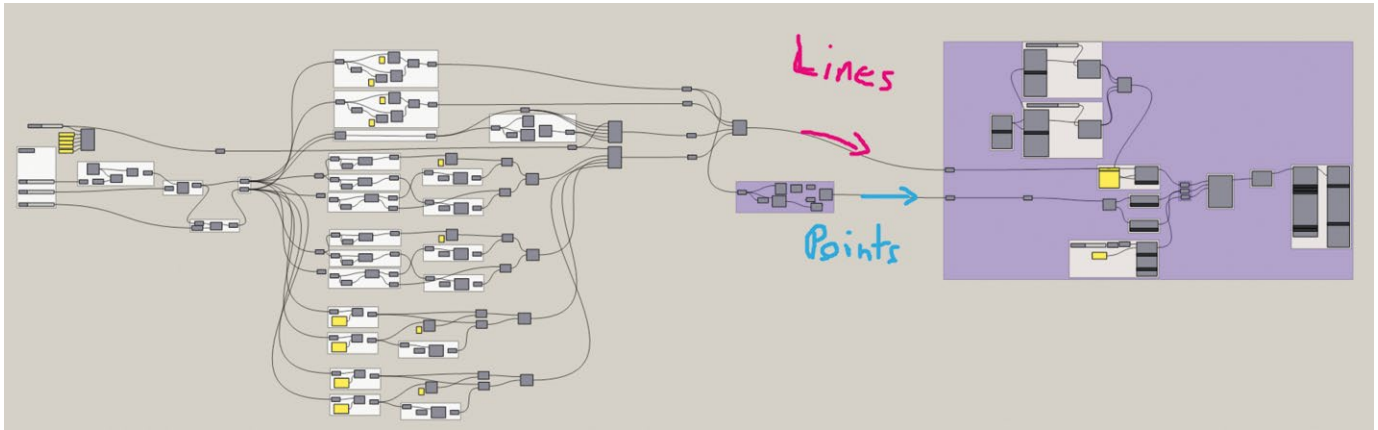


Fig. 19 Parametric engineering model of the truss

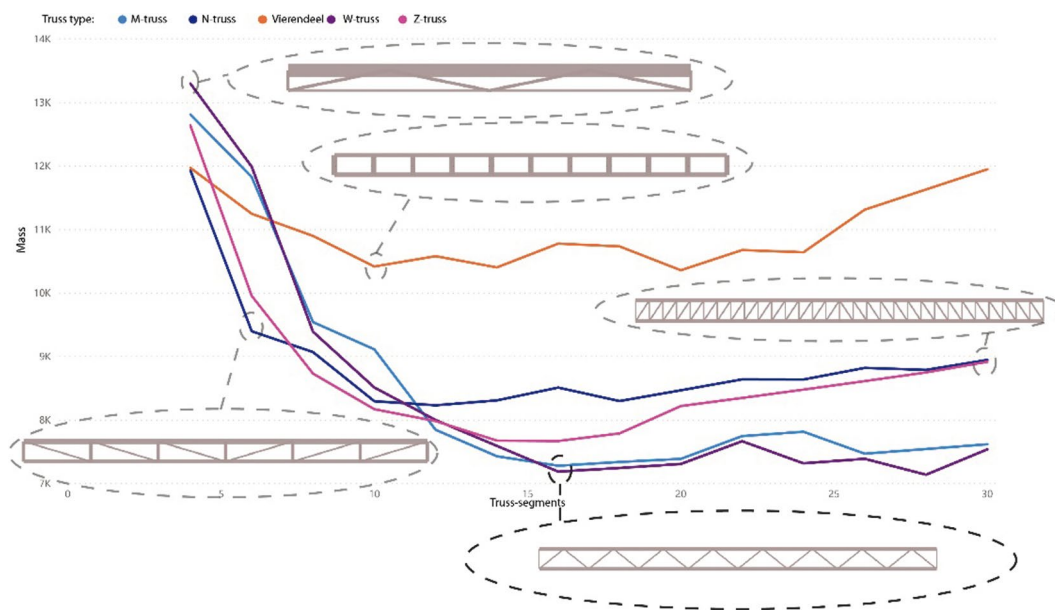


Fig. 20 Truss results

is the W-truss with 16 segments min total weight of 7190 kg.

4 Further possibilities in parametric design

Grasshopper holds many possibilities and application, mainly due to the fact that users can develop their own components in python, C# or VBA. This enables the user to encapsulate existing code or algorithms within Grasshopper. These special purpose developed components are often shared as plug-ins and made available to download. In this paragraph worth-mentioning possibilities that Grasshopper entails are briefly discussed.

4.1 Optimisation strategies

In the examples discussed in previous sections, a “brutal force analysis” has been used. In this analysis method every possible variant is simulated. This can become problematic in terms of computational time, if the model

has a high number of different parameters with wide ranges. In such cases, algorithms can be used to reduce computational time.

A likely alternative is using a genetic algorithm. In such an algorithm a population (relatively small number of different variants) is created. Variants with a good result or outcome (e.g. minimal weight) are combined to produce offspring, comparable to the survival of the fittest in nature. After multiple generations optima can be found in the solution landscape (Fig. 21). Grasshopper has plug-ins like Octopus and Galapagos enabling users to use these genetic algorithms in their optimisation process.

The Karamba3D website includes examples in which genetic algorithms are applied to structures. Including examples like a roof structure where the position of its supports are variable or a truss structures where the number of the diagonals and the position of these diagonals are variables (Fig. 22). Due to the high number of design possibilities in such examples, a genetic algorithm may bring the user swiftly to an optimal.

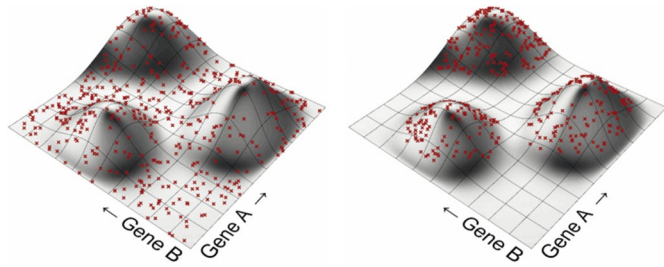


Fig. 21 Finding optima in the solution landscape [4]

4.2 Sustainability

In parametric design workflows multiple variants can be compared. This becomes especially interesting when the material (steel, concrete, or timber) becomes a variable as well. CO₂ emissions can consequently be calculated based on the amount of used material. Plug-in like One-ClickLCA enables the user to link the material outcomes with an environmental database [6].

In combination with code-based analyses, Karamba3D for steel and concrete or Beaver for timber [5], a multi-material CO₂ optimisation model can be created. To further reduce the environmental impact on our planet and to reach our set climate goals, the need for such multi material optimisation processes can be a massive help.

4.3 Integrate connection design in the parametric model

Parametric design enables us to optimise weight by changing the structures geometry and helps engineers to design structures with a smaller carbon footprint. However, in this optimisation process the structural connections are often neglected. A related pitfall is when the designer is on the wrong track, when at a later stage the structural connections require larger dimensions of the cross-sections.

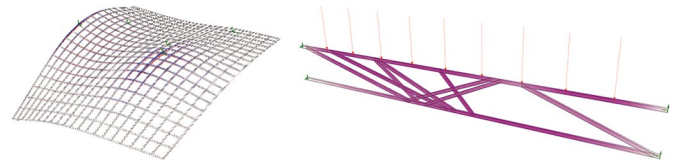


Fig. 22 Genetic algorithm examples Karamba3D [1]

A higher level of detail can be added to the parametric engineering models. By including structural details, an integral analysis process can be created where every connection is analysed. This leads to an optimisation process where cost is optimised rather than total weight. Bouwen met Staal has conducted the SMARTconnection research project, where methods have been developed to integrate steel connections in a parametric optimisation process [7]. This integral process leads to a more buildable design and constructability, since uncertainties that connections can bring when designed at a later stage are eliminated. Part of the research was the development of Karamba-IDEA. This tool allows engineers to export their steel connections modelled in Karamba3D to IDEA StatiCa connection for further analysis on a detail level.

5 Conclusion

Visual programming can offer the engineer a useful extra tool. Due to the large number of plug-ins available, parametric design brings many possibilities in search for better solutions in design projects. Making use of its full potential requires training, especially improving how complex structures can be broken down into a sequence of logical steps. To give the industry the possibility of implementing these new possibilities Bouwen met Staal organises English courses in parametric design bi-annually. Planned dates of upcoming courses can be found on the Bouwen met Staal website.

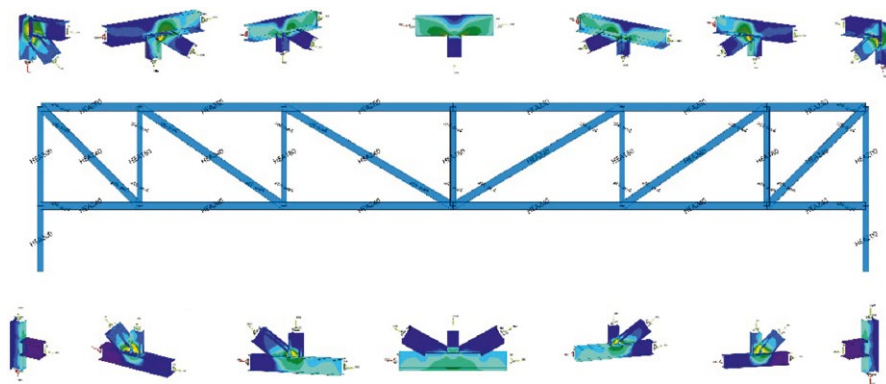


Fig. 23 Integrating detail analysis in the parametric model

References

- [1] Karamba3D (2021) *Karamba3D – parametric engineering* [online]. <https://www.karamba3d.com/> [accessed on 05-05-2021]
- [2] TT toolbox (2021) *Download page* [online]. <https://www.food4rhino.com/app/tt-toolbox> [accessed on 05-05-2021]
- [3] Octopus (2021) *Download page* [online]. <https://www.food4rhino.com/app/octopus> [accessed on 05-05-2021]
- [4] Rutten, D. (2010) *Evolutionary Principles applied to Problem solving* [online]. <https://www.grasshopper3d.com/profiles/blogs/evolutionary-principles> [accessed on 05-05-2021]
- [5] Beaver (2021) *Download page* [online]. <https://www.food4rhino.com/app/beaver> [accessed on 05-05-2021]
- [6] Apellániz, D.; Pasanen, P.; Gengnagel, C. (2021) *A Holistic and Parametric Approach for Life Cycle Assessment in the Early Design Stages*. SimAUD 2021 April 15–17.
- [7] Ajouz, R. (2019) *SMARTconnection* [online]. Zoetermeer: Bouwen met Staal. <https://www.bouwenmetstaal.nl/the-mas/parametrisch-ontwerpen/smartconnection/> [accessed on 05-05-2021]
- [8] Ajouz, R. (2020) *KarambaIDEA download page* [online]. Zoetermeer: Bouwen met Staal <https://www.food4rhino.com/app/karambaidea> [accessed on 05-05-2021]

Author

ir Rayaán Ajouz (corresponding author)
 rayaan@bouwenmetstaal.nl
 Bouwen met Staal
 Louis Braillelaan 80
 2719 EK Zoetermeer, The Netherlands

How to Cite this Paper

Ajouz, R. (2021) *Parametric Design of steel structures: Fundamentals of parametric design using Grasshopper*. Steel Construction. <https://doi.org/10.1002/stco.202100011>