LAB University of Applied Sciences

Parametric modeling in structural design

LAB University of Applied Sciences
Double Degree Program in Civil and Construction Engineering
2020
Leonid Vasilev

Abstract

Author(s)	Type of Publication	Published
Vasilev, Leonid	Thesis, UAS	2020
	Number of Pages	
	65	

Title of Publication

Parametric modeling in structural design

Name of Degree

Civil Engineer (UAS)

Name, title, and organization of the supervising teacher

Antti Roiha, LAB University of applied sciences

Name, title, and organization of the client

Timo Lehtoviita, LAB University of applied sciences

Abstract

The purpose of this Bachelor's thesis is to investigate parametric modeling in a structural design stage. Building information technologies evolve every day, and now you can not even imagine a building project without a BIM 3D model. In building modeling, we have got two general problems, how to create complicated architecture and how to do it faster. The thesis was compared and observed two main programs for algorithmic modeling Dynamo, Grasshopper. The main idea of the topic is to find which of the two programs better fits for structural parametric design. the result of the thesis is parametric models of the structural portal frame, the result of work with difficult various shapes, and exploration of work with the analysis part via parametric design.

Also, this Thesis is a part of the BIMICE project and the result included guidelines for Grasshopper and Dynamo models.

The thesis was commissioned by LAB university civil engineering

Keywords

Parametric modeling, Dynamo, Grasshopper

Contents

1 Introduction and Aim

1.1 Introduction.

With the development of BIM technologies, the 3D design has become an integral part of any building project. It allows making design with high accuracy, systematizing, and speeding up the design, while eliminating the appearance of various errors. (Nancy Reyes 2020)

Because of this, there was a jump in the construction industry. A large-scale project that is difficult to implement by direct modeling began to appear. And parametric modeling became a solution to such problems.

Parametric modeling is the next step for a designer, a tool that allows solving more complex architectural solutions and systematizing the modeling process.

1.2 Aim of the thesis.

This thesis aims to explore the issue of parametric modeling by comparing two main programs for 3d parametric design Autodesk Dynamo and Grasshopper for Rhinoceros. The functional is examined by the example of working with structural design in combination: Dynamo – Revit, Grasshopper – Tekla.

It is important to note that in this project the goal was also set to isolate from comparisons of BIM software as much as possible, excluding those moments in which it directly concerns parametric modeling.

For this thesis, work was done to research to work in the above programs from a beginner to a confident user. The process of investigation is divided into 4 stages for each of the programs: first acquaintance, work with difficult shapes, parametric portal frame, basic structural analysis.

- First acquaintance. This part described the process of the first connection and experience as a new user. The first connection, finding information and guidelines, a common description of Interface and tools for work.
- Work with difficult shapes. Description of a process creating space frame and wood arc. Assignment for wood arc was a mission to understand how both programs work with creating elements from 0 with non-ordinary form and placement of them to the BIM model. For a spaceframe is an assignment to look at work with creating frame on early prepared non-ordinary surface.

- Parametric portal frame. Experience in creating a fully parametric model of steel industrial frame. With the brace system for roofs and walls as well as between columns in 3 different variations
- Structural Analysis. Some information about the possibility of parametric structural analysis. And an example of creating a parametric 2d portal frame analytical model.

2 Background

2.1 Parametric modeling

Parametric modeling is modeling using the parameters of the model elements and the relationships between these parameters. Parameterization allows for a short time to "play" (by changing parameters or geometric relationships) various design schemes and avoid fundamental errors.(Parametric design, 2015)

In contrast to direct modeling, a parametric model is constructed by using logical connections that make modeling similar to programming. Commonly the result of parametric modeling is not a model but a script is created by using visual programming or by using text programming or both ways together.

The parametric model of the building combines the 3D model and external data. The model is updated correctly when changing its elements. All elements of the model are linked by dependencies; when the parameter is changed, the model is updated automatically.

To understand the idea of this thesis, it is necessary to understand the work of algorithmic modeling inside the Dynamo and the Grasshopper. since they have the same basic idea, it is enough to consider only one software to understand the essence, it will be Grasshopper.

Grasshopper and Dynamo are visual programming software in which a script is written by connecting different nodes. (Parametric design, 2015)

Nodes are commands responsible for a particular function. they are of 2 types:

The former simply generate information, such an example is the number slider, you can see it in the figure 1, which does not take any information into itself, but simply creates a numerical value.

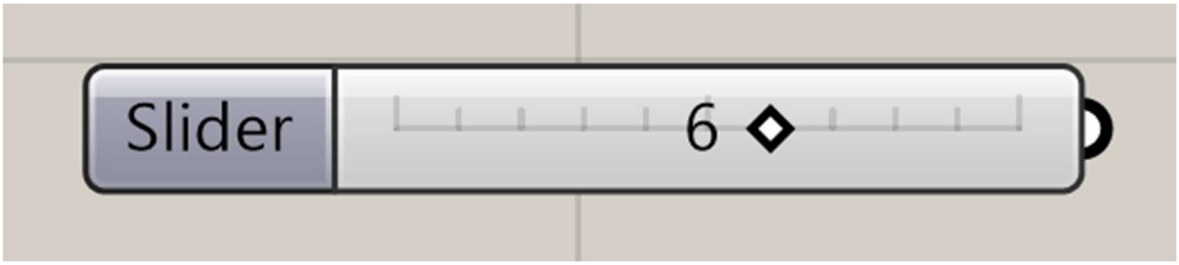


Figure 1. Number Slider Node in Grasshopper

The latter process information into something else, you can see an example in the figure 2. There are most of such nodes and it is from them that the script is built, they can be

characterized by 3 key concepts: the task they must perform, the information that they receive (INPUTS), and the information that they give (OUTPUTS).

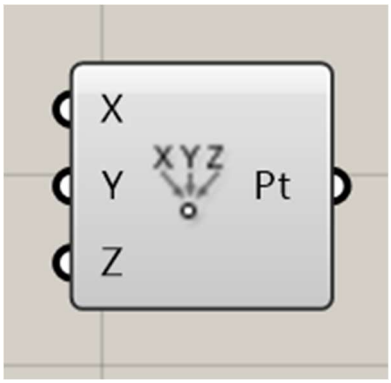


Figure 2. Point by Coordinates Node in Grasshopper

Below, in the figure 3, is an example of using nodes: there is a node for creating a line, 2 nodes for creating points, and number sliders that are used to set coordinates for points. By connecting all of them, you get a task for building a line.

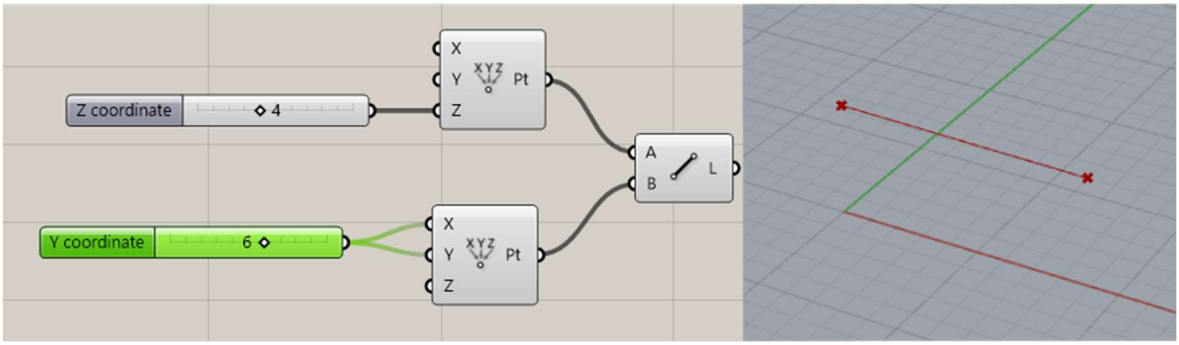


Figure 3. Method for creating lines in Grasshopper

In direct modeling, you usually put elements by choosing a place in 3d/2d view, and both of two lines work separately

To summarize in general, parametric modeling is not modeling at all, it is just a sum of different commands, which are formed in the task for the computer, and then the computer creates a model by using this task. You can see an example in the figure 4.

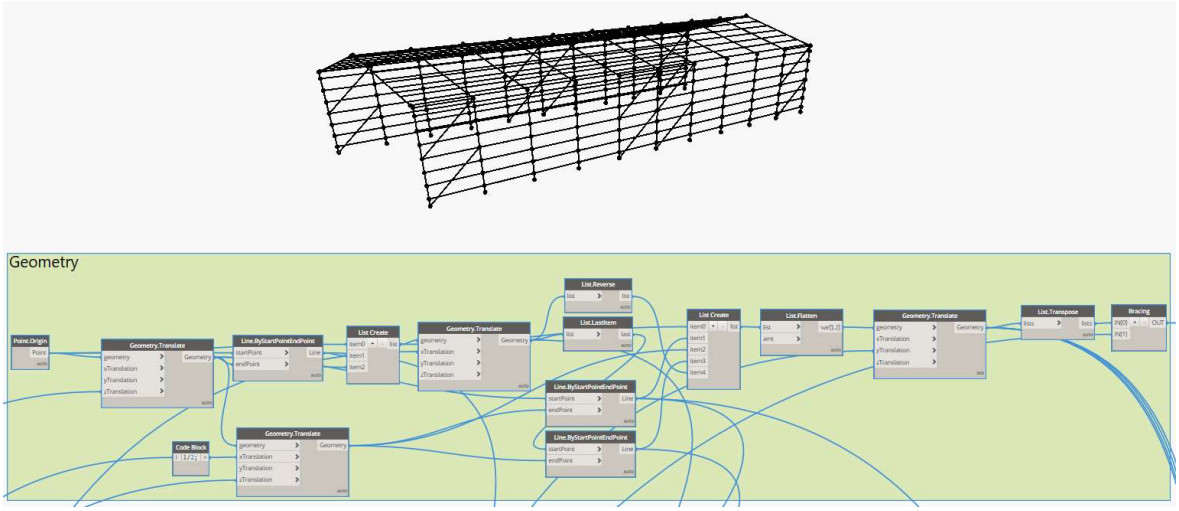


Figure 4. Visual programming in Dynamo for Revit.

Parametrization may be noted in the environment as a phenomenon that created a new way of Architecture. The possibility to create difficult shapes by using mathematical formulas investigated Parametric Architecture, that mostly impossible to create using direct modeling. An example of such an architecture is Gaha SOHO created by architect Zaha Hadid, you can see the picture in the figure 5 (Rob Deutscher, 2013)

Figure 5. Gaha SOHO, Beijing, China. Architect Zaha Hadid

This thesis explores two main tools for nowadays for this kind of case.

2.2 Software to use

2.2.1 Grasshopper combination

Grasshopper is an additional free software for parametric modeling that is provided for Rhinoceros software for 3D CAD modeling. In this thesis, we do not use Rhinoceros at all, but it is important to notice it. Since 6 version of Rhino Grasshopper at once goes in complect inside Rhinoceros. But if you use an older version. You need to upload grasshopper to its official website. (Grasshopper 3D, 2020)

As a BIM modeling program is used Tekla Structures. Tekla Structures is a software product for creating BIM models of buildings or structures. Tekla possesses tremendous engineering tools for structural design. In the process of researching the question of parametric modeling, the standard Tekla tools were not used, only the one that is proposed as an extension to Grasshopper. (Tekla, 2020)

For Structural Analysis is used extension for Grasshopper Karamba3D. Karamba3D is a parametric structural engineering tool that provides accurate analysis of spatial trusses, frames, and shells. Karamba3D is fully embedded in the parametric design environment of Grasshopper, a plug-in for the 3d modeling tool Rhinoceros. (Karamba3D, 2020)

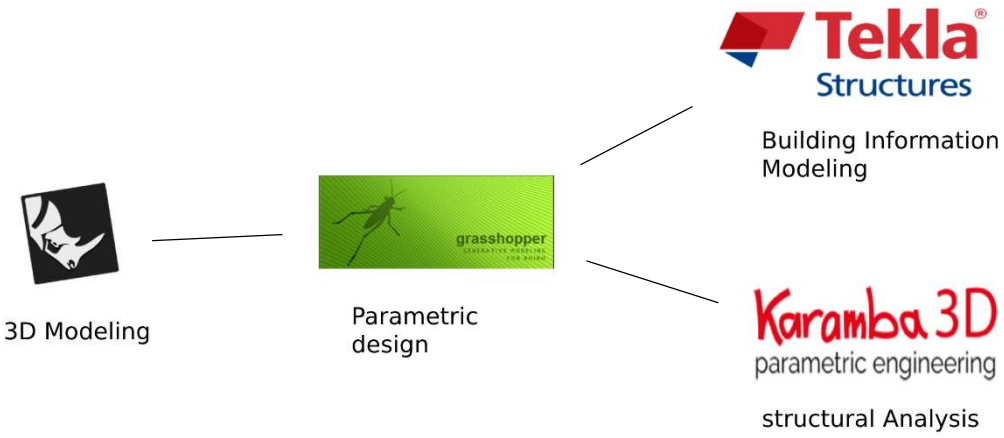


Figure 6. Software used in combination with Grasshopper

2.2.2 Dynamo combination

Dynamo is a visual programming software developed by Autodesk. There are several different versions of this software, such as dynamo studio, dynamo sandbox, and dynamo for Revit, and the dynamo can also be used with other Autodesk programs (Formit, Robot, Advanced Steel, etc.) (Autodesk, 2020)

In this thesis, Dynamo for Revit and Dynamo sandbox was used.

In general, these programs are the same and perform the same functions, except that the dynamo for Revit comes in one package with Revit, which allows you to use all the dynamo functionality and Revit library for the dynamo.

In turn, Dynamo Sandbox is free software, but it has limited functionality, it is limited to the fact that you cannot upload your packages, cannot import or export DWG files, it also does not have a library for roars, and therefore you cannot work with roars in it(Autodesk, 2020).

Revit 2021 was used as a counterbalance to Tekla Structures for BIM software. Revit is a solution for many BIM design departments such as architectural structural design and MEP design. In the process of writing a diploma work, the goal was set to isolate as much as possible from the functionality of the direct modeling of BIM software. Therefore, the Revit functional was used on a final basis, mainly libraries in Dynamo were used. (Revit, 2018)

It is also worth noting that for Revit there is the possibility of connecting with a grasshopper, but in this thesis, it was decided to use only the conical version: Dynamo – Revit

Autodesk Robot Structural Analysis (Robot) is a structural analysis and design software. It allows you to virtually analyze structures of any type and shape, as well as design elements of these structures (steel, reinforced concrete, wood) (Autodesk, 2020).

the Robot functionality as a program for structural analysis is not used, in this project, to create an analytical model and its analysis, an extension package was used, which is responsible for adding the Robot functionality to Dynamo and the possibility of creating a parametric analytical model inside the Dynamo and further exporting it to the Robot.

Figure 7. Software used in combination with Dynamo

3 Process of Modeling

3.1 Dynamo

3.1.1 First connection

To start the process of BIM parametric modeling with Dynamo is needed only to have Autodesk Revit.

Dynamo has got its website with information, guidelines, video lessons, and a forum with quite a big community base. To start to create script information from video lessons enough. All these links easy to find in the main menu of a program, also there are put some basic Dynamo script samples.

There are several possibilities to view the generated geometry in Dynamo: directly look in the Dynamo programming environment, in Revit, or use a special “Watch 3d” node.

All basic nodes in Dynamo have only one output connector. This means that each node has only one function or task to complete. Most nodes have got a description that conveys the full essence of work this node.

Dynamo has got extra tool packages that expand the functionality of Dynamo differently. You can find packages directly in Dynamo or by google search.

Important to notice the Dynamo Player's existence. This tool allows us to not create or modify script but directly to play it in Revit. The menu is easy to understand and work. It allows even for people that do not know how to use Dynamo easy to use scripts for Revit. Important to remember that as Revit, Dynamo can work with files that were created in an older version of the program, but the older version of Dynamo cannot open files that are created in a new one.

Dynamo has the possibility of creating scripts by using Python 3 Language.

Generally to summarize the first steps in Dynamo, For Revit users, it is easy to start to work with Dynamo, it has got a lot of good quality tutorials and an active Forum where you can ask or find information about your problem.

3.1.2 Work with difficult shapes

WOODEN ARC

The process of writing this script is divided into 3 stages:

1. creation of geometry
2. assigning this geometry to the family
3. placing the family in the Revit model.

Only base nodes were used to create the script.

To create geometry in Dynamo and place it in Revit, important to understand the process of placing a model in Revit. Almost all geometry in Revit has its own separate .rfa file called a family, when you create geometry you put it in family and then place a family in the Revit model. Each family has its type from general geometry to various types of the structural frame. In this script, there was a task of placing geometry in Revit using only Dynamo.

The main node for executing this script is “FamilyType.ByGeometry” , you can see it in the figure 8, It acts as a link between the geometric body and Revit, transforming the body into a family. INPUT

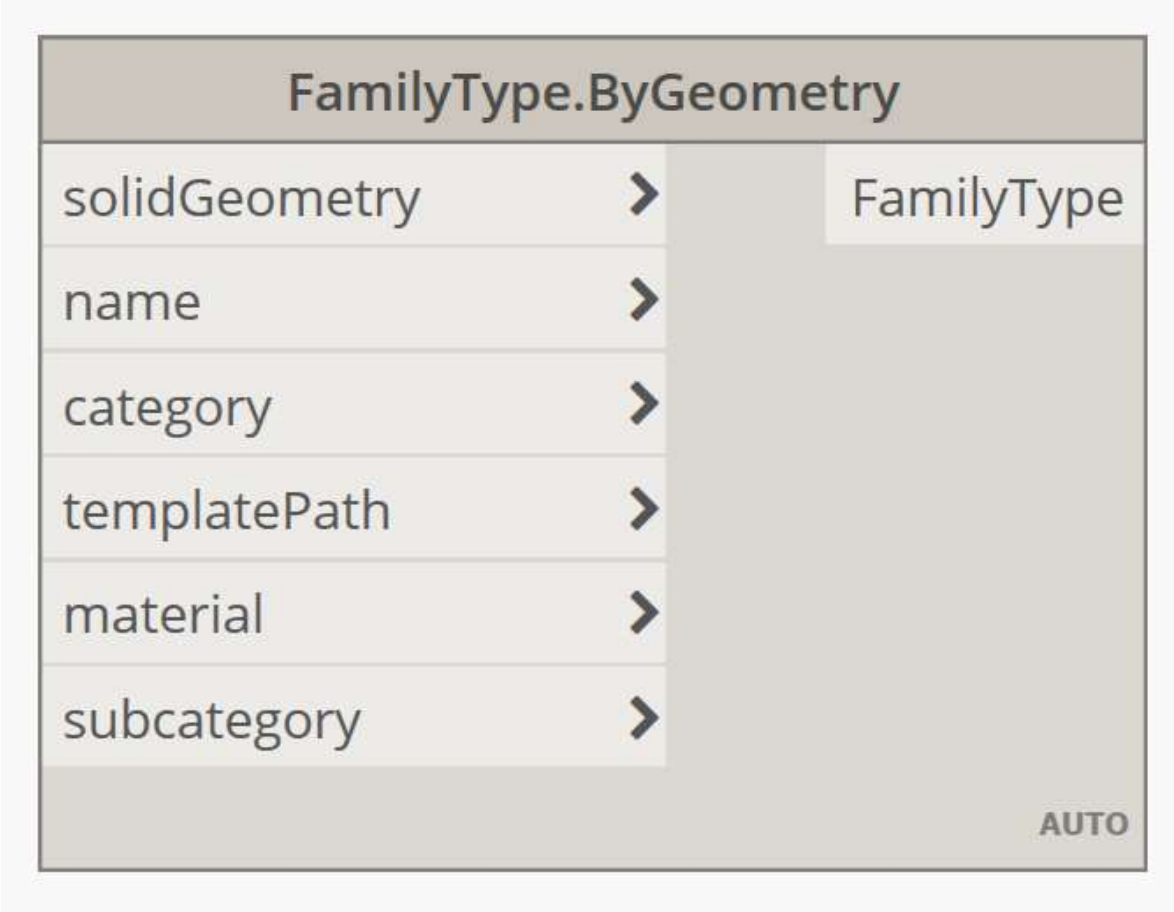


Figure 8. The node responsible for creating the dynamo geometry family

So, to create a model inside Dynamo is needed to fulfill several conditions: create a geometry, you can see it in the figure 9, set a name, a structure type, attach a link to a template file, and set material.

starting with the first, you need to create the geometry.

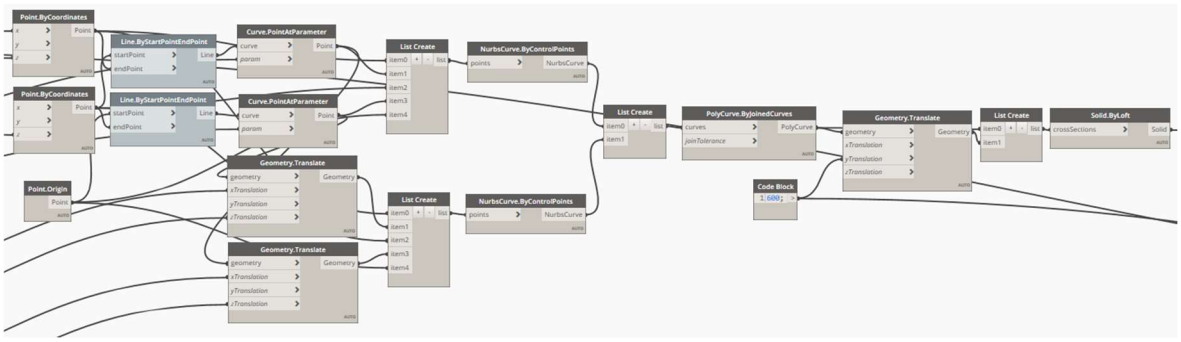


Figure 9 Algorithm for creating geometry.

The complexity of the geometry created directly depends on its shape, but the result should always be solid. In this model, the process of creating geometry was not something specific, it included the creation of points along which the curves included in the 2 curves were built, which were later combined into a polycurve, then it was copied to the distance of the thickness of our structure and the final geometry, you can see it in the figure 10, was formed by both polycurves.

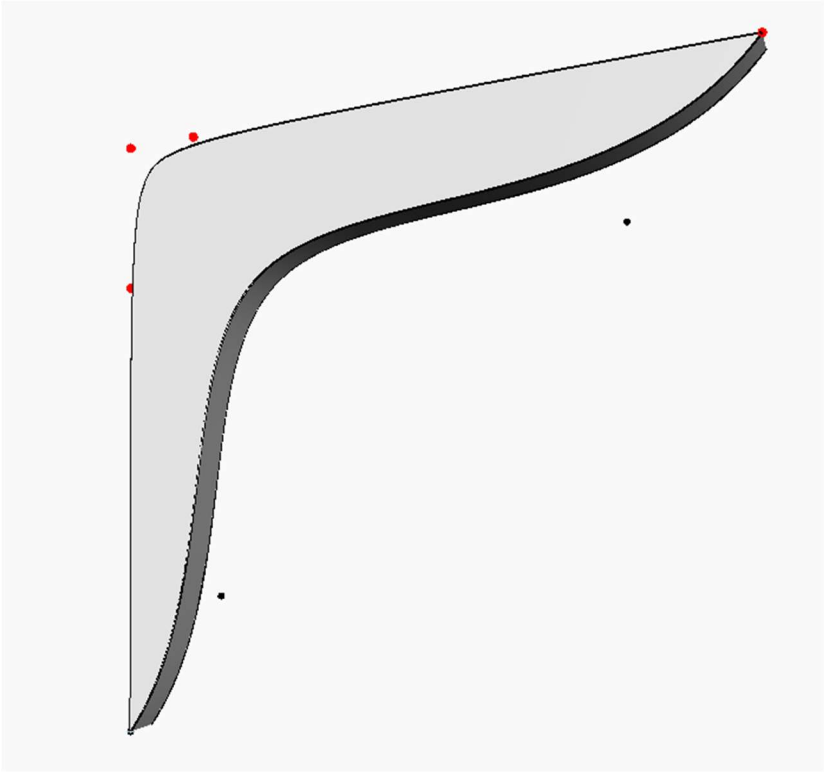


Figure 10. Dynamo solid object

Accordingly, after creating the geometry, you must specify the rest of the required data for creating the geometry, you can see it in the figure 11,.

For the name day, you need to specify any name.

In the category, set the type of your family, then how it will be identified in Revit. This is done using a special node that opens access to the list of all categories within Revit.

A template file is a file that will be used to define a family, and the creation of a family begins with the selection of a template file. it is enough to select any template file, which one does not matter since the geometry is created inside the Dynamo.

For the material, a node is used which, by name, finds the name of the material inside Revit.

Once all these criteria have been set, the geometry family will be created.

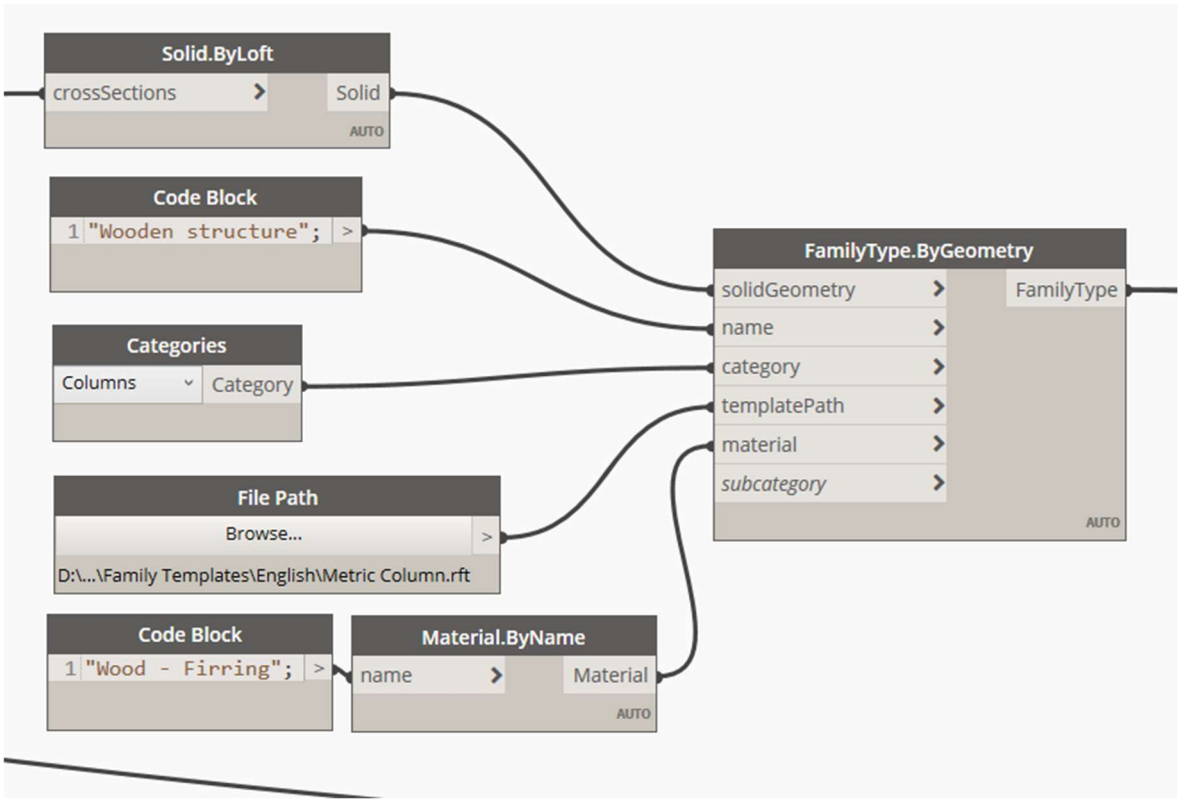


Figure 11. Algorithm for creating a family

Accordingly, at this stage, the geometry was initialized with the name that we specified and saved in Revit, but not placed in the building model itself.

There are several possible options for its placement. The first is to place the elements manually inside the Revit, in this case, since we created the family, we can change its geometry inside the Dynamo, you can see it in the figure 12, and it will be updated for all placed elements inside the Revit, but it will not work to manipulate the position.

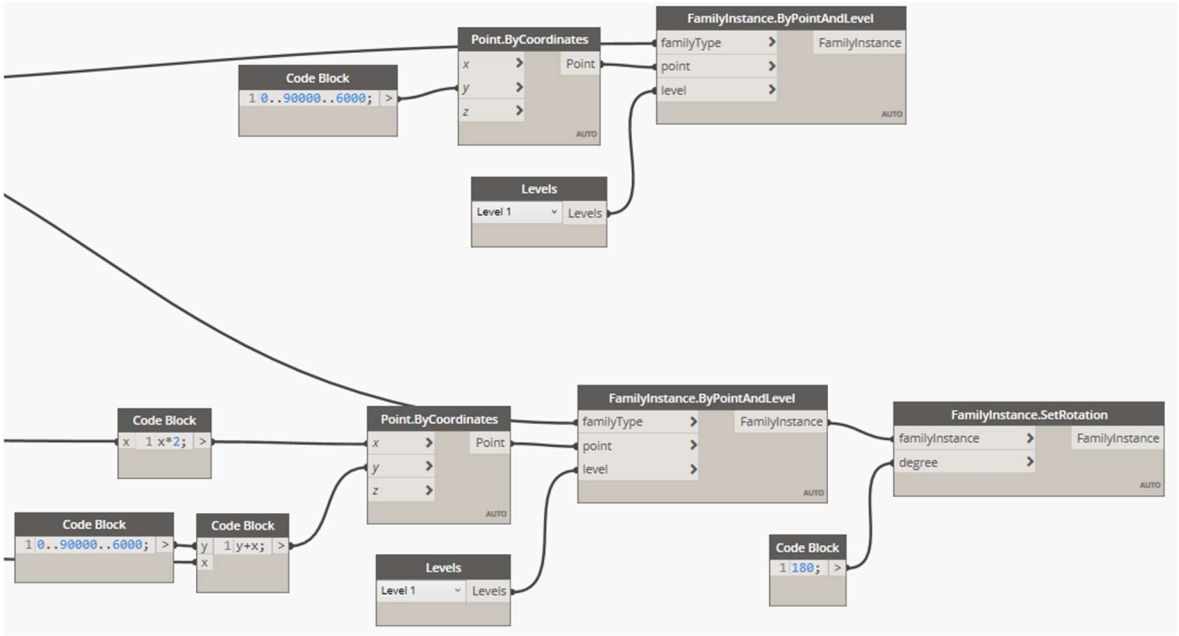
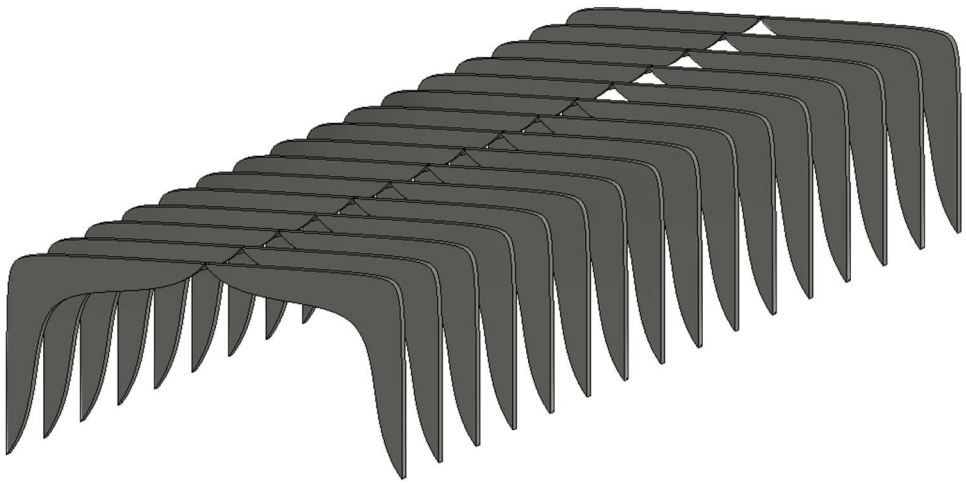


Figure 12. Algorithm for the location of the family inside Revit

The second method of placement is through Dynamo, for this, a special "FamilyInstance.BypointAndLevel" node was used, whose task is to place elements by level and points. Also, to rotate the geometry 180 degrees, the " FamilyInstance.SetRotation" node was used to position the second part of the arch.

You can see the result of Revit model in the figure 13.



Picture 13. The result of the script in Revit

SPACE FRAME

The process of creating this script can be divided into 3 stages:

1. Creating a surface,
2. Creating geometry of lines along the surface
3. Positioning the BIM model along these lines.

The first stage can be of 2 types: when there is already a finished surface, you can see it in the figure 14, and when you need to create a new one inside Dynamo, you can see it in the figure 15.

In the first case, all we need to do is export this surface from Revit to Dynamo and convert it to geometry. it is worth noting that along with the surface, other elements of the plane geometry, such as lines and curves, are also transformed and transferred, and then a surface must be selected from the geometry list.

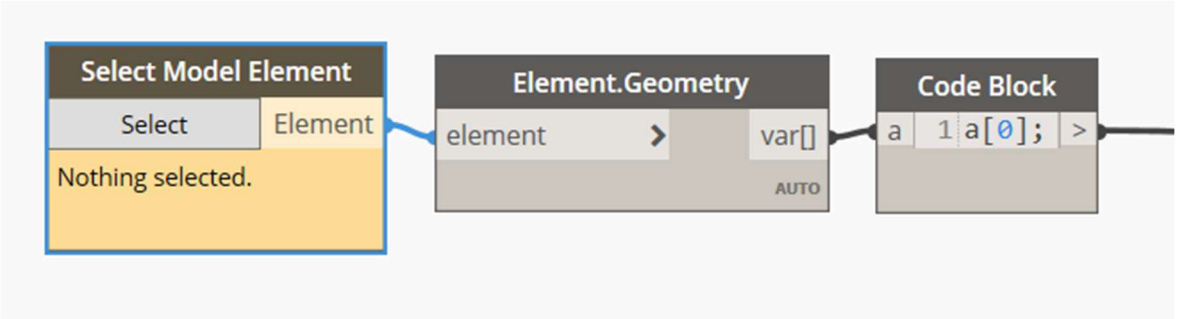


Figure 14. Method of obtaining geometry from Revit

In the second case, build a new one using the dynamo toolkit, in the case of this example, the method of creating a surface by 2 curves was used.

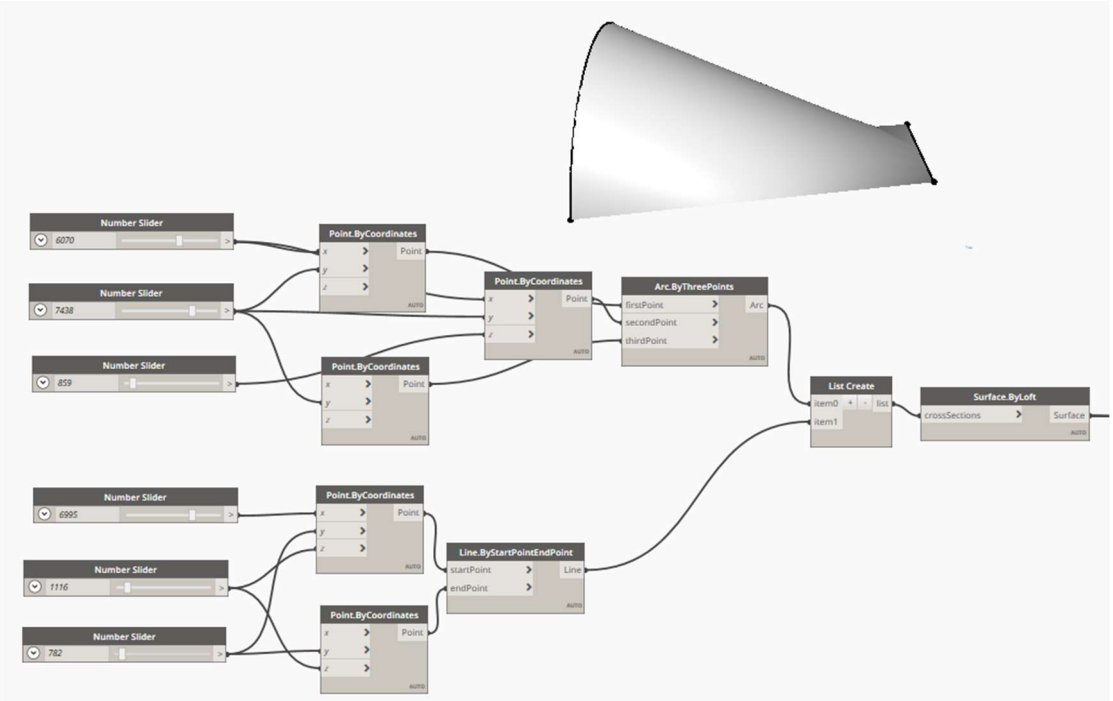


Figure 15. Method of creating a surface inside the Dynamo

The 2nd stage is the main one for this script. There are several methods for its implementation, but 3 main ways can be distinguished: using the standard library of nodes, you can see it in the figure 16, resorting to using the built-in Python language, or finding a custom package in the public domain that performs this function.

the custom package "LunchBox" has the necessary custom node to create the geometry of the save frame. But this script will not be used.

The process of creating a script using the standard library of Dynamo nodes is a possible and feasible process, but due to the need for non-linear management of lists of information, this method is overloaded and inconvenient.

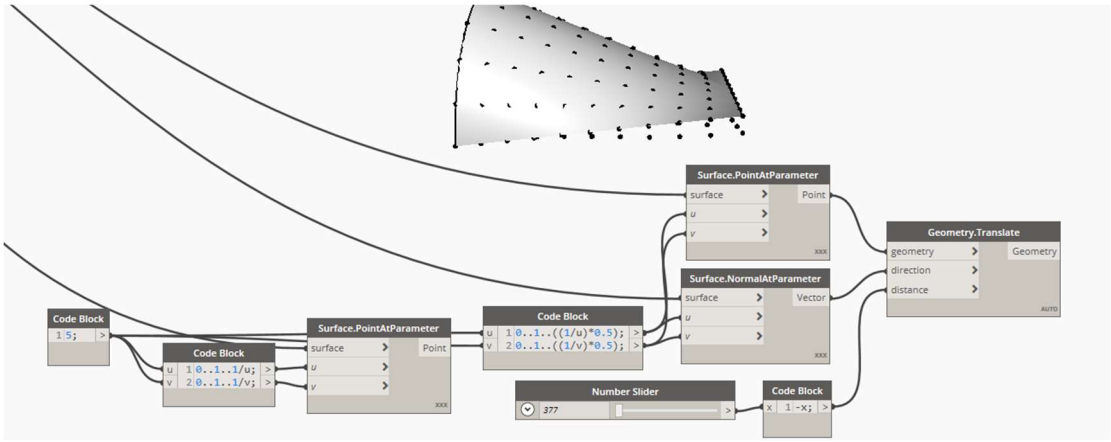


Figure 16. The idea of a method for creating space frames using standard Dynamo nodes

The process of creation using Python was chosen as the main method, you can see it in the figure 17 and figure 18, with due knowledge of the syntax of the language, this method is the most optimal and fastest when it is necessary to work with lists.

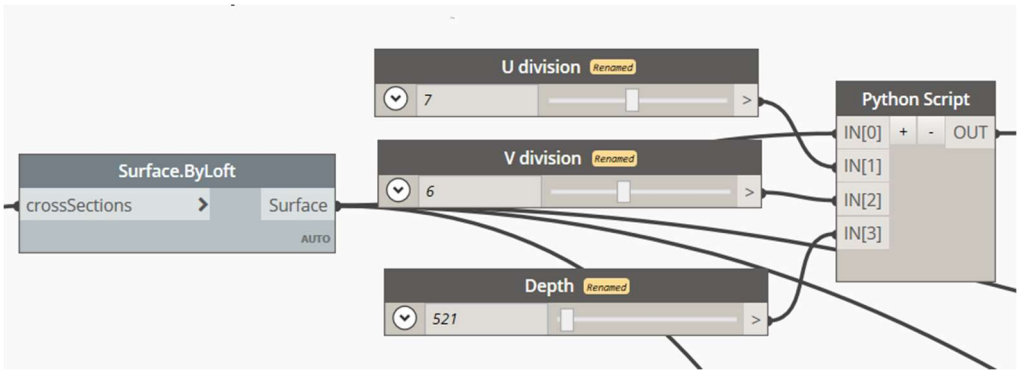


Figure 17. Information for the Python script and the Python script itself

To write a script inside Python, the same node syntax is used as in Dynamo itself. to write it, you need to know python synaxis and the concept of using conditions and loops.

```python
import sys
import clr

clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

surf = IN[0]
U = IN[1]
V = IN[2]
depth = IN[3]

Udiv = 1.0 / U
Vdiv = 1.0 / V
deep = Surface.Offset(surf, depth)
ls = []
for i in range(int(U) + 1):
    for j in range(int(V) + 1):
        pt1 = Surface.PointAtParameter(surf, i * Udiv, j * Vdiv)

        if i < U:
            pt2 = Surface.PointAtParameter(surf, (i + 1) * Udiv, j * Vdiv)
            ln1 = Line.ByStartPointEndPoint(pt1, pt2)
            ls.append(ln1)
        if j < V:
            pt2 = Surface.PointAtParameter(surf, i * Udiv, (j + 1) * Vdiv)
            ln1 = Line.ByStartPointEndPoint(pt1, pt2)
            ls.append(ln1)

        if i < U and j < V:
            pt1 = Surface.PointAtParameter(deep, (i * Udiv) + (0.5 * Udiv), (j * Vdiv) + (0.5 * Vdiv))
            if i < U - 1:
                pt2 = Surface.PointAtParameter(deep, ((i + 1) * Udiv) + (0.5 * Udiv), (j * Vdiv) + (0.5 * Vdiv))
                ln1 = Line.ByStartPointEndPoint(pt1, pt2)
                ls.append(ln1)
            if j < V - 1:
                pt2 = Surface.PointAtParameter(deep, (i * Udiv) + (0.5 * Udiv), ((j + 1) * Vdiv) + (0.5 * Vdiv))
                ln1 = Line.ByStartPointEndPoint(pt1, pt2)
                ls.append(ln1)

            PtA = Surface.PointAtParameter(surf, i * Udiv, j * Vdiv)
            PtB = Surface.PointAtParameter(surf, (i + 1) * Udiv, j * Vdiv)
            PtC = Surface.PointAtParameter(surf, i * Udiv, (j + 1) * Vdiv)
            PtD = Surface.PointAtParameter(surf, (i + 1) * Udiv, (j + 1) * Vdiv)
            mid = Surface.PointAtParameter(deep, (i * Udiv) + (0.5 * Udiv), (j * Vdiv) + (0.5 * Vdiv))

            lineA = Line.ByStartPointEndPoint(PtA, mid)
            lineB = Line.ByStartPointEndPoint(PtB, mid)
            lineC = Line.ByStartPointEndPoint(PtC, mid)
            lineD = Line.ByStartPointEndPoint(PtD, mid)

            ls.append(lineA)
            ls.append(lineB)
            ls.append(lineC)
            ls.append(lineD)

OUT = ls
```

Figure 18. Code is written in Python

After creating the geometry of the lines, the process of creating a BIM model begins. For this, the "StructuralFraming.BeamByCurve" node is used, , you can see it in the figure 19, which is directly responsible for creating a structural beam inside Revit along a curve from Dynamo. To create a model, you just need to select the type of beam from those available inside the Revit model, select a level that is conventionally necessary to create a BIM model but does not affect the position of the geometry, and connect the curves in the Python script.

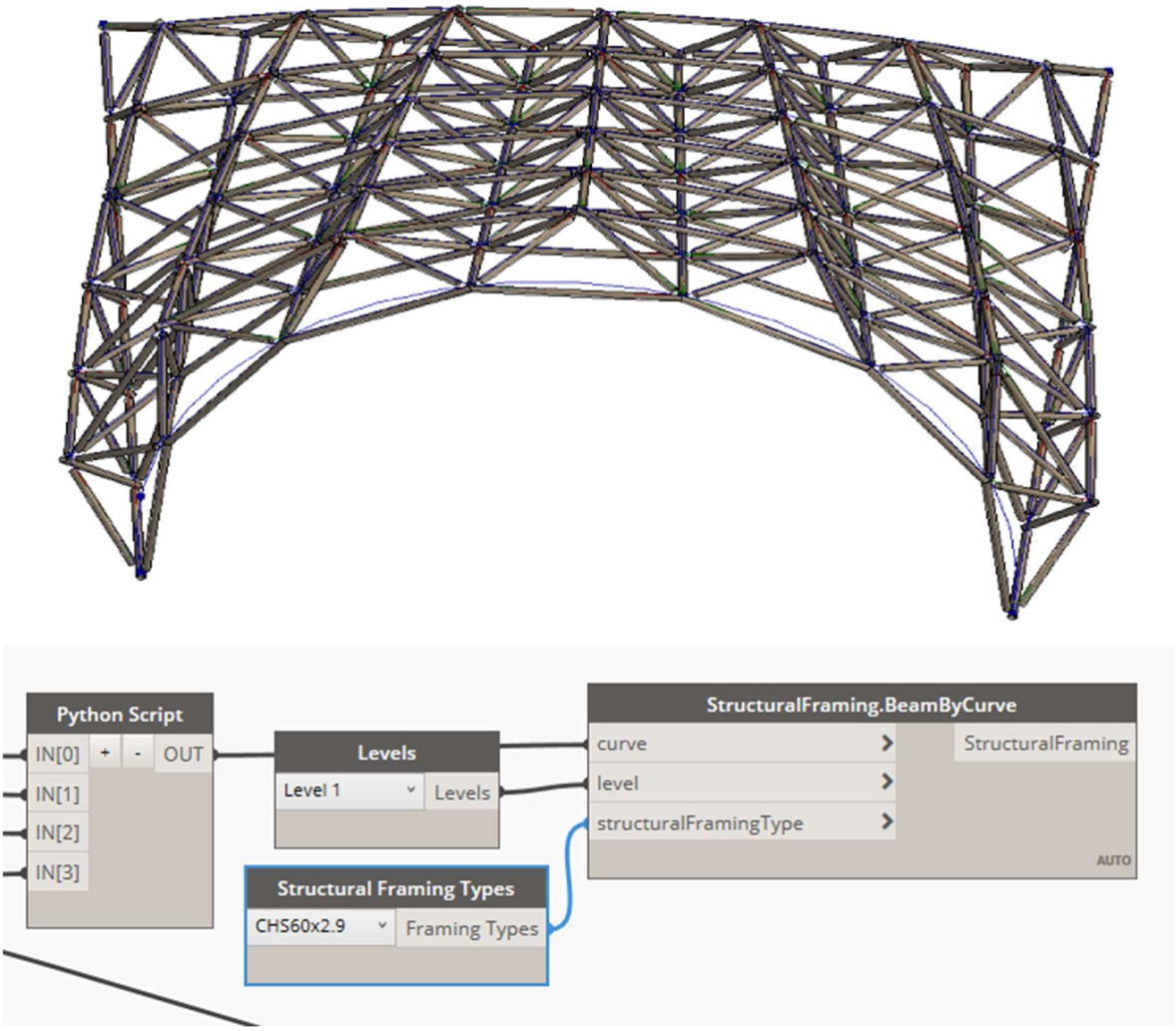


Figure 19. Frame forming for Revit

3.1.3 Parametric portal frame

The process of writing a script for a structural model is directly dependent on what this model will include. But the essence will always be the same: first, geometry is created, and then a structural model is built using this geometry.

This model consists of columns and beams that form a portal frame, bracing between columns with 3 types of variation, and bracing between walls and roof.

The geometry creation process for this script is to create lines for beams and columns and bracing.

Geometry creation begins with creating a 2D frame and then copying it to the length of the building with a column pitch. to generate this sequence, the block code is used with its special syntax, you can see it in the figure 20,:

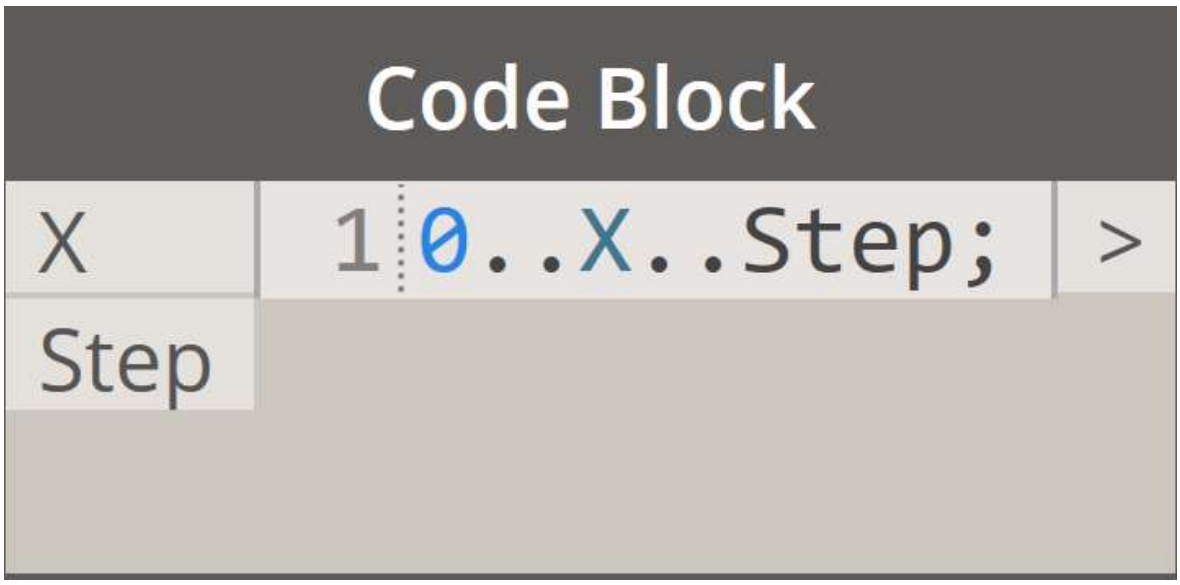


Figure 20. Generating a sequence of numbers using a code block

A sequence of numbers is created where:

0: This is the start of counting

X: This is the end of the countdown

Step: The step with which this sequence goes.

The geometry itself consists of points and lines. which were created with the help of nodes for creating or copying geometry, you can see it in the figure 21.

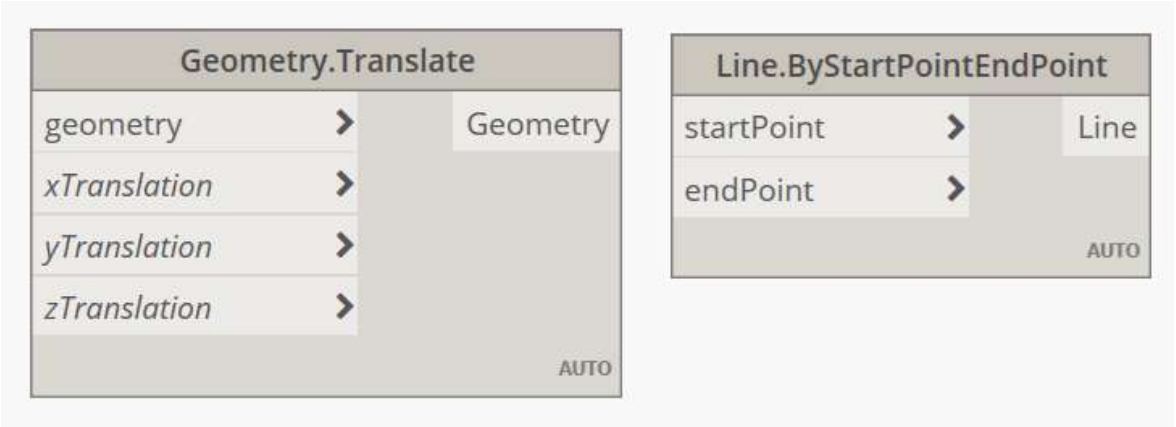


Figure 21. The main nodes used to generate lines

An important part of creating geometry is occupied by the process of managing and organizing the lists, you can see it in the figure 22, in this script it was important to understand in what order the elements of the wireframe are located. therefore, it was important to combine elements into a list in a logical order, and somewhere it was necessary to edit lists of elements.

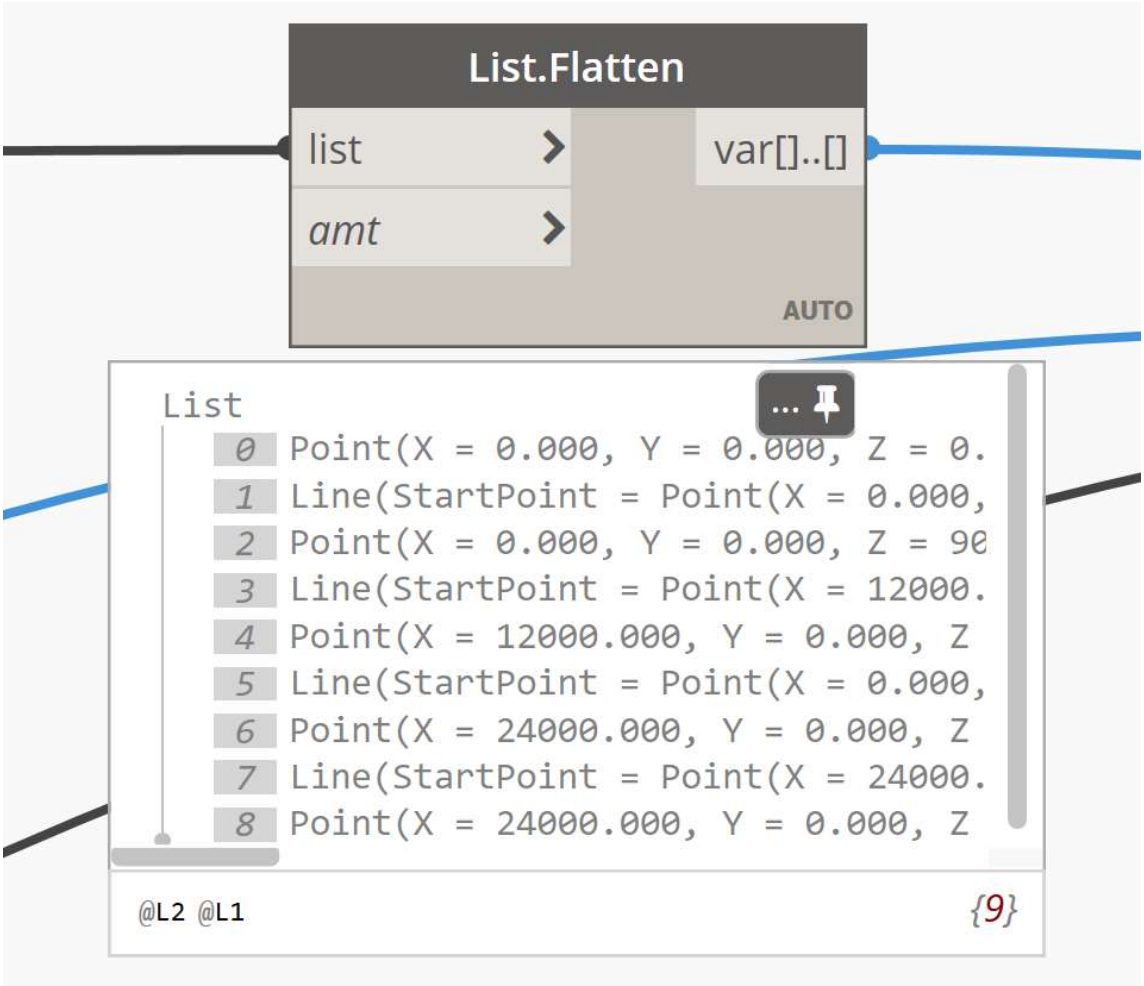


Figure 22. The sequence of items in a list

Elements in this list form a 2D portal frame in exact order: that is, first comes the lower point of the column, then the column line, after the last point of the column, line 1 of the beam, the endpoint of 1 beam, etc. This process is necessary so that after the elements from this list will be taken by their ordinal numbers, to build connections between columns or form a frame for Revit, and knowing the local sequence simplifies the process.

The result of performing all the higher actions will be the main portal frame of the building with the required column step for the entire length of the building, you can see it in the figure 23.

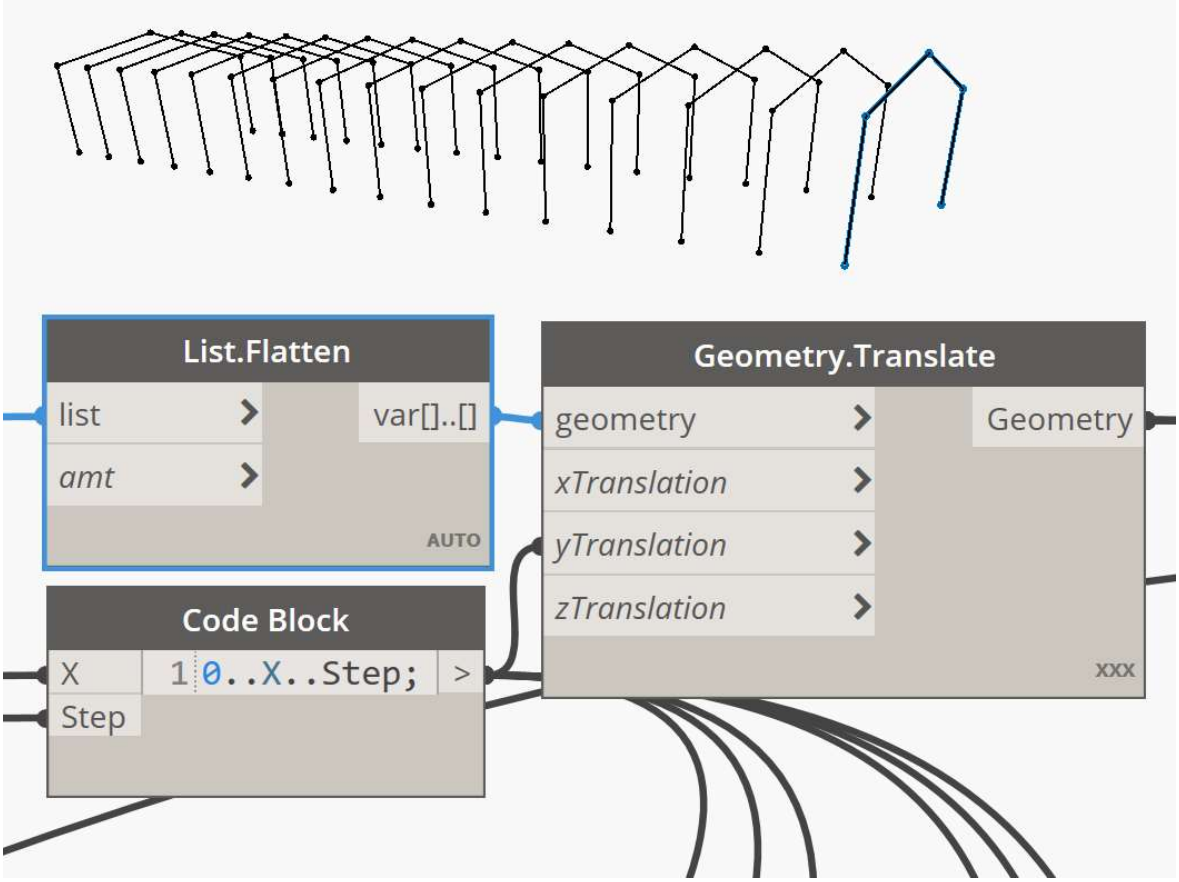


Figure 23. Script for copying the frame to the length of the building

A python script was built to create bracing between the columns. into which 2 values were supplied: 1 is the final list of all elements of the portal frame, and the second is a number slider that was responsible for the choice of the bracing option.

The very process of writing a script inside python is not very different from the process of creating nodes inside a dynamo, all we do is also create geometry and manage lists of elements, only the tool and style change. Accordingly, everything that is done in python can be done in a dynamo, but python is a more compact and convenient option when it is necessary to work with conditions, non-linear management of lists, or loops.

The process of this script can be described by selecting some elements from the list and creating the bracing based on the condition of which option we need, you can see it in the figure 24.

```python
import sys
import re
import clr

clr.AddReference('ProtoGeometry')
from Autodesk.DesignScript.Geometry import *

A = IN[0]
bo = IN[1]
lst = []

middle = len(A) // 2


def brace(a, b, c):
    if c == 1:
        def a1(x, y):
            lst.append(Line.ByStartPointEndPoint(A[a][x], A[b][y]))
            lst.append(Line.ByStartPointEndPoint(A[a][y], A[b][x]))
            lst.append(Line.ByStartPointEndPoint(A[a][y], A[b][y]))

        a1(0, 2)
        a1(8, 6)

    if c == 2:
        def a2(x, y, z):
            m_1 = Curve.PointAtParameter(A[a][z], 0.5)
            m_2 = Curve.PointAtParameter(A[b][z], 0.5)
            ln_m = Line.ByStartPointEndPoint(m_1, m_2)
            m_m = Curve.PointAtParameter(ln_m, 0.5)
            m_top = Curve.PointAtParameter(Line.ByStartPointEndPoint(A[a][y], A[b][y]), 0.5)
            lst.append(ln_m)
            lst.append(Line.ByStartPointEndPoint(A[a][y], A[b][y]))
            lst.append(Line.ByStartPointEndPoint(A[a][x], m_m))
            lst.append(Line.ByStartPointEndPoint(A[b][x], m_m))
            lst.append(Line.ByStartPointEndPoint(m_1, m_top))
            lst.append(Line.ByStartPointEndPoint(m_2, m_top))

        a2(0, 2, 1)
        a2(8, 6, 7)

    if c == 3:
        def a3(x, y, z):
            m_1 = Curve.PointAtParameter(A[a][z], 0.5)
            m_2 = Curve.PointAtParameter(A[b][z], 0.5)
            lst.append(Line.ByStartPointEndPoint(m_1, m_2))
            lst.append(Line.ByStartPointEndPoint(A[a][y], A[b][y]))
            lst.append(Line.ByStartPointEndPoint(A[a][x], m_2))
            lst.append(Line.ByStartPointEndPoint(m_1, A[b][y]))

        a3(0, 2, 1)
        a3(8, 6, 7)


brace(0, 1, bo)
brace(-2, -1, bo)
brace(middle, middle + 1, bo)
brace(middle - 1, middle, bo)

OUT = lst
```

Figure 24. Python script for creating braces between columns

As a result, the scripts get lines in 3 different variations, you can see it in the figure 25.

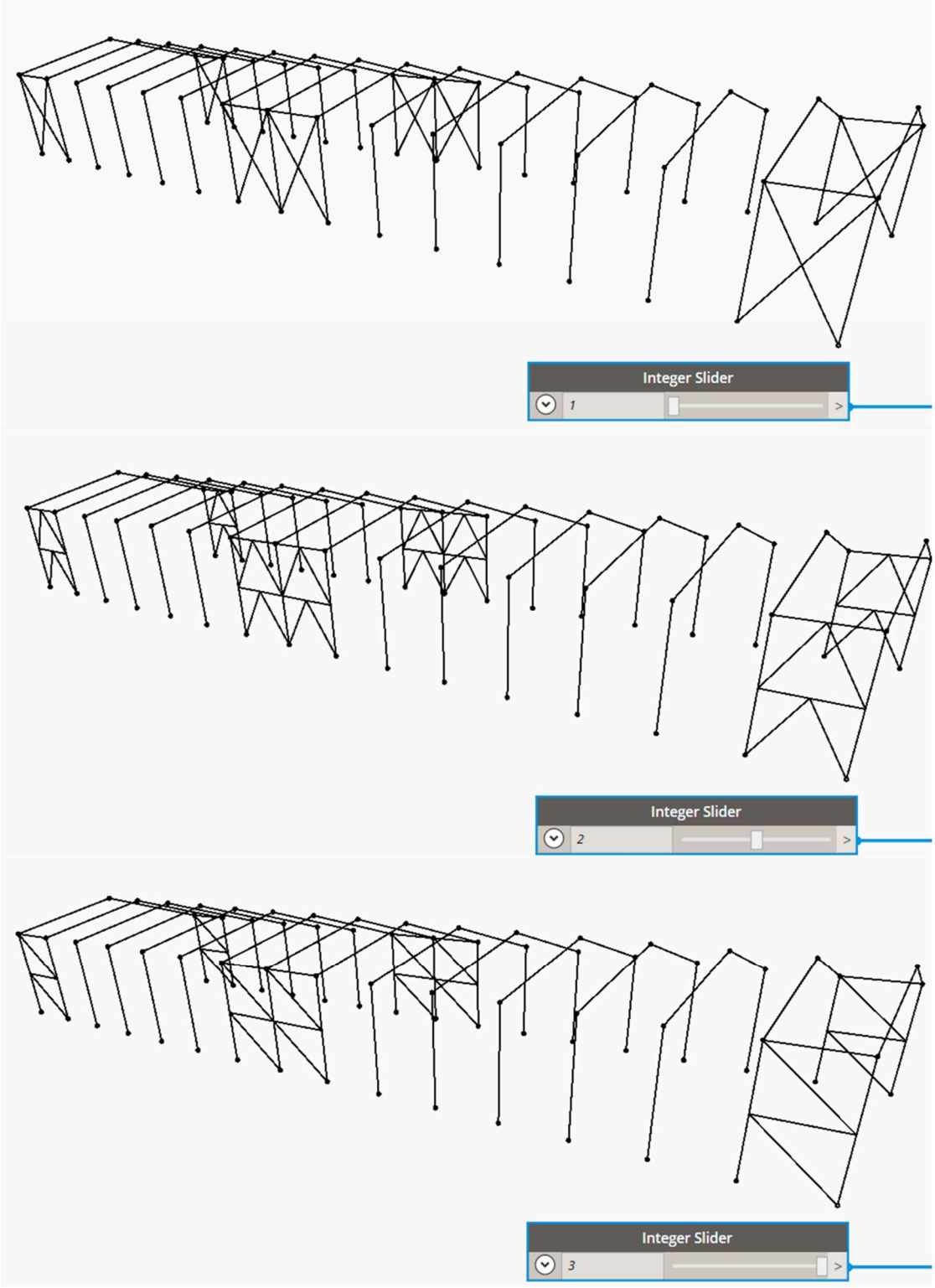


Figure 25. 3 variations of the frame created in Dynamo

After that, the process of creating a connection of beams and walls begins, the structure and logic of this script are the same, so it is enough to create for one and copy for the other, you can see it in the figure 26.

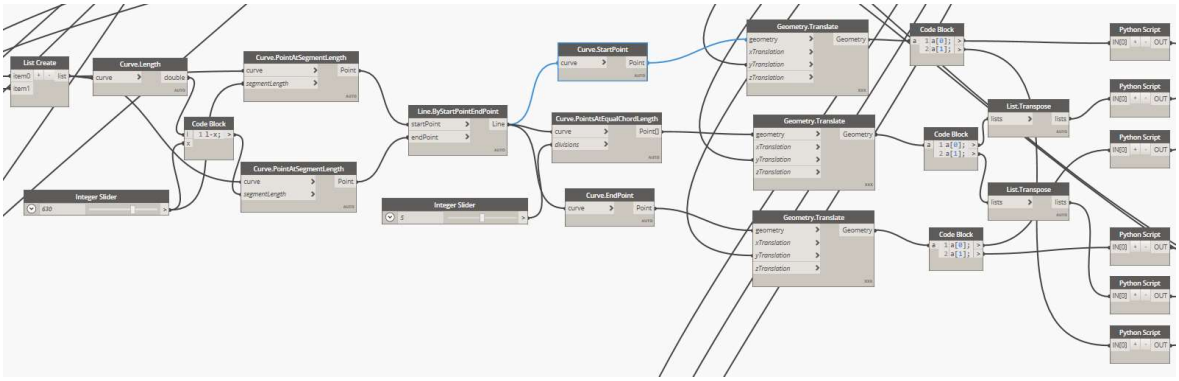


Figure 26. Script for creating a roof brace

This script takes the lines of the beams of the portal frame, creates indents from both ends at the selected distance for each line, and 2 new lines are drawn along the new lines. Inside these lines, points are created at equal intervals, the number of which is adjusted using the slider, then these points are copied to all the following lines, and lines are created between them that is bracing.

The same method is appropriate for creating bracing for walls, so this script is simply copied, and the lines of columns are fed into the values in which we supplied the lines of the beams.

The most massive part is the part of creating a BIM model for a given geometry, or rather its adjustment. To create the geometry itself, you need 3 nodes, you can see them in the figure 27:

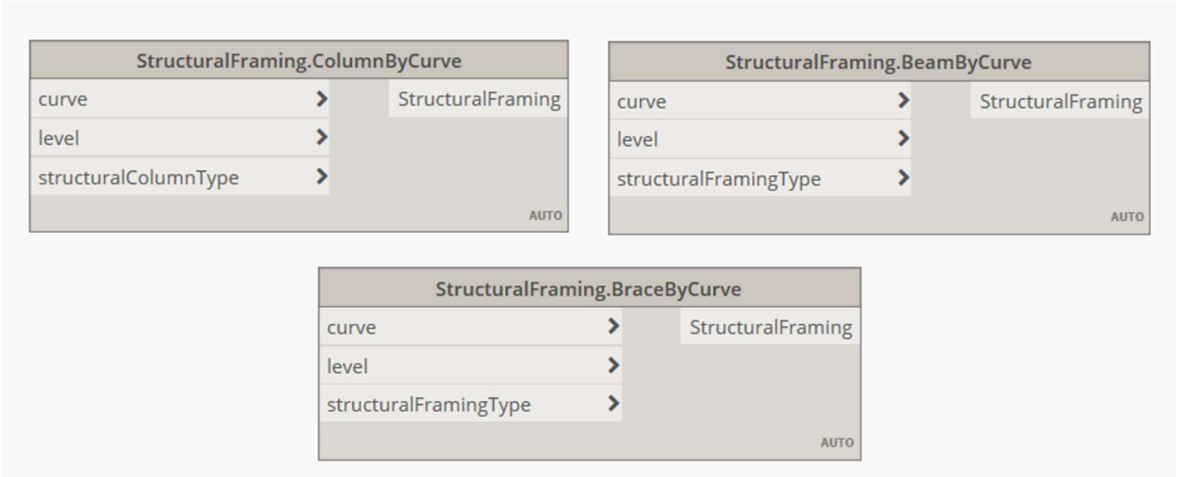
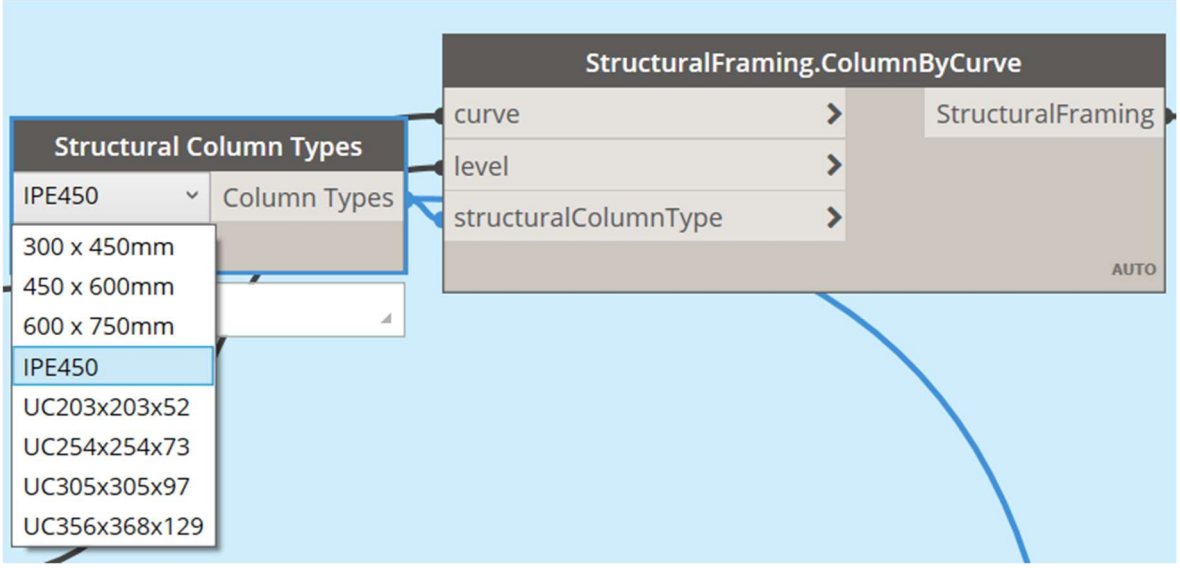


Figure 27. Nodes for the formation of a structural frame from curves in Dynamo

Accordingly, the necessary lines are selected along which the frame will be built. The level and section of the frame are also selected, you can see it in the figure 28. It is important to

note that you can use those element families that were loaded into this model. For this model, families from the Revit core library for Finland were used.



Picture 28. Structural Column Library

The frame is built strictly along the lines and the line always intersects the center of the section, to build a section with an offset or some rotation, you need to create a node that changes the parameter for the element inside the Dynamo, you can see it in the figure 29.

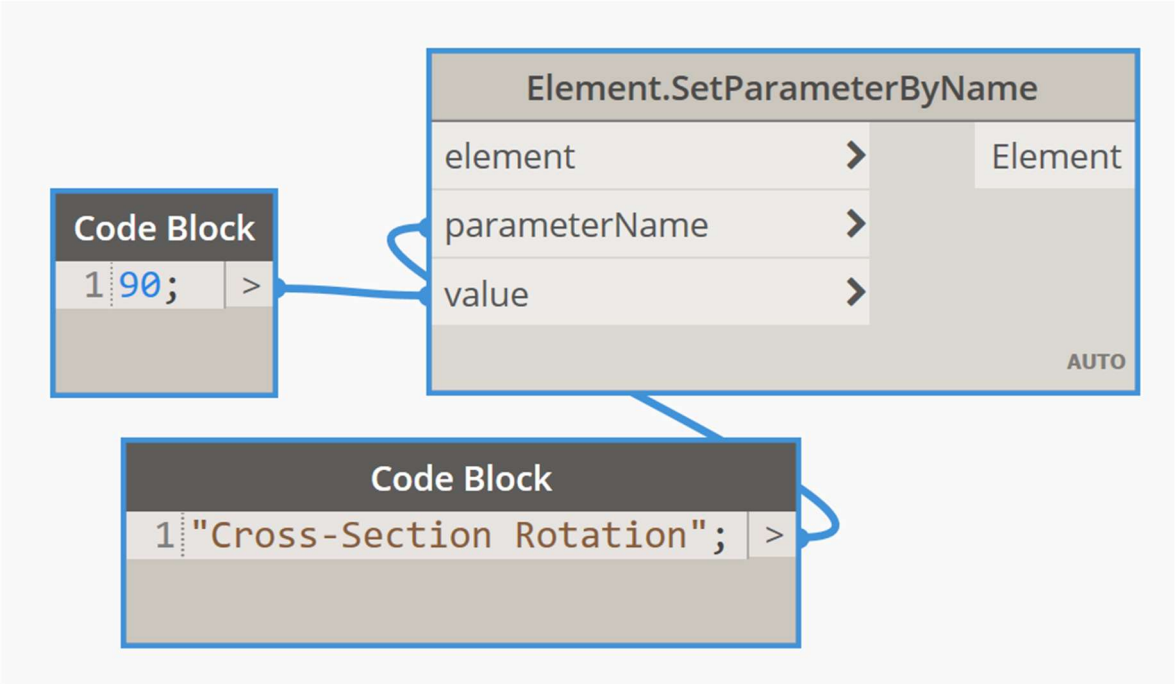


Figure 29. Setting a parameter by name

Each element in Revit has its parameter sheet, which can be edited or simply viewed, you can see it in the figure 30. To change these properties inside Dynamo, you need to respectively specify the name of the desired parameter and give it the desired value.

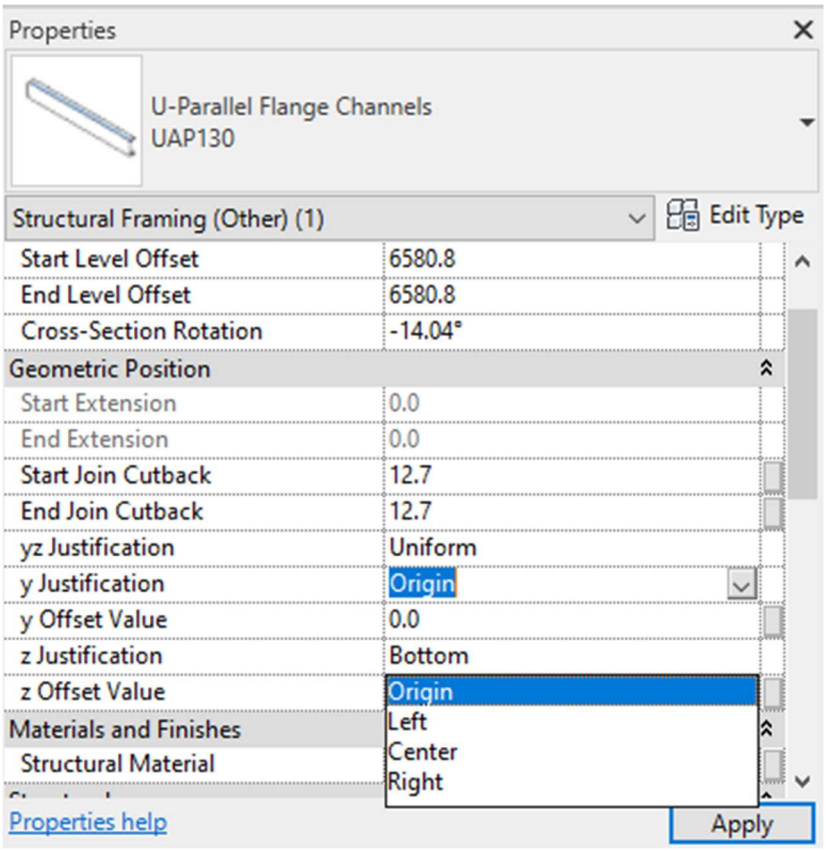


Figure 30. Item parameters inside Revit

You create a foundation by placing a foundation family at a specified point, you can see it in the figure 31.

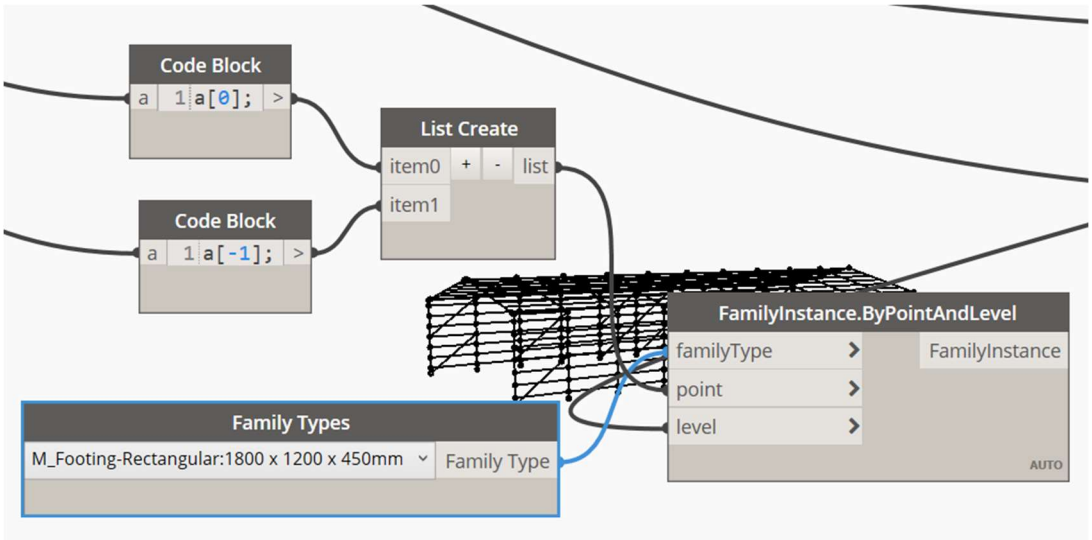


Figure 31. Foundation script

After creating all the structural elements and editing the required parameters, how it will be built in Revit, you can see it in the figure 32.

Figure 32. The skeleton resulting from the script

The most rational way to create connections between metal structures using the Dynamo is to use the Dynamo player and an additional package "Autodesk Steel Connections 2020". This method allows, by choosing a framework within Revit, to build various connections between them. These scripts independently select the necessary elements from those proposed for the selected type of connections, you can see it in the figure 33.

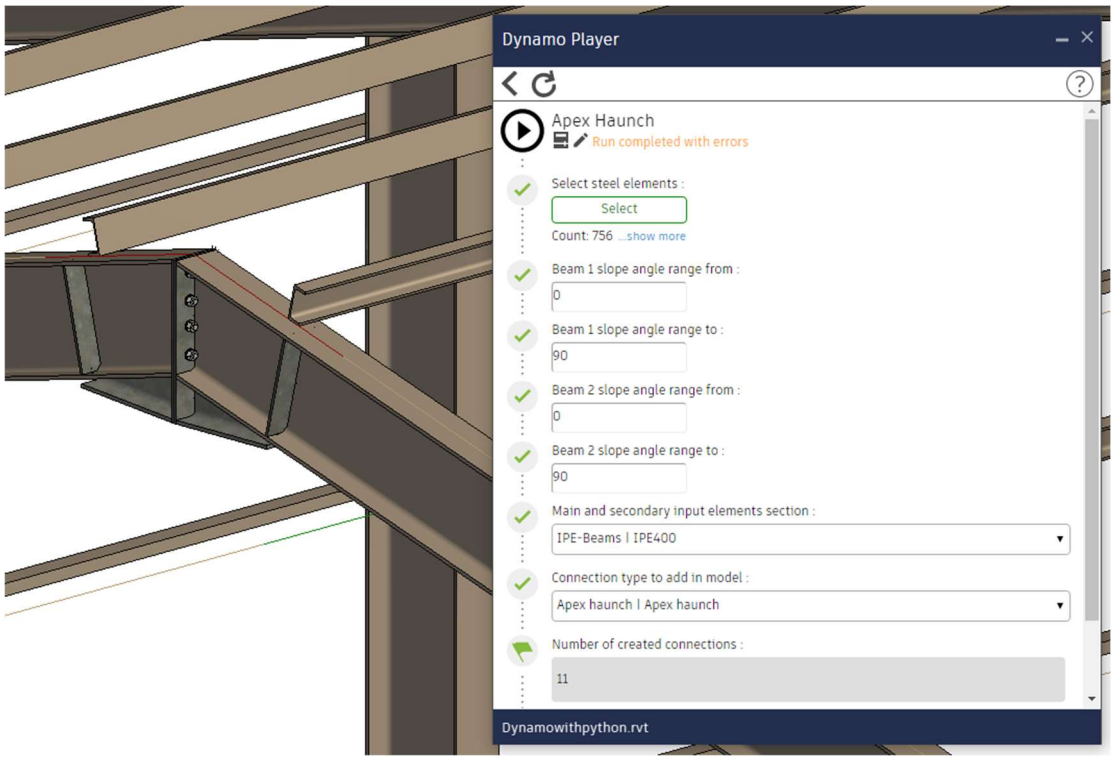


Figure 33. Creating steel connections in Dynamo Player

3.1.4 Structural Analysis

Before attempting to write this script, an error was detected in which Dynamo for Revit 2021 could not be connected to the Robot 2021. To solve this problem, the Dynamo Sandbox was used, Robot 2021 connects with it without problems.

The scriptwriting process can be divided into parts:

1. Creating 2D geometry in a dynamo,
2. Converting geometry to an analytical model in Robot,
3. Adding sections, materials, and supports,
4. Task of loads
5. Analysis of the resulting structure.

Geometry creation consists of creating lines and points. The geometry itself is a 2D portal frame, you can see it in the figure 34.

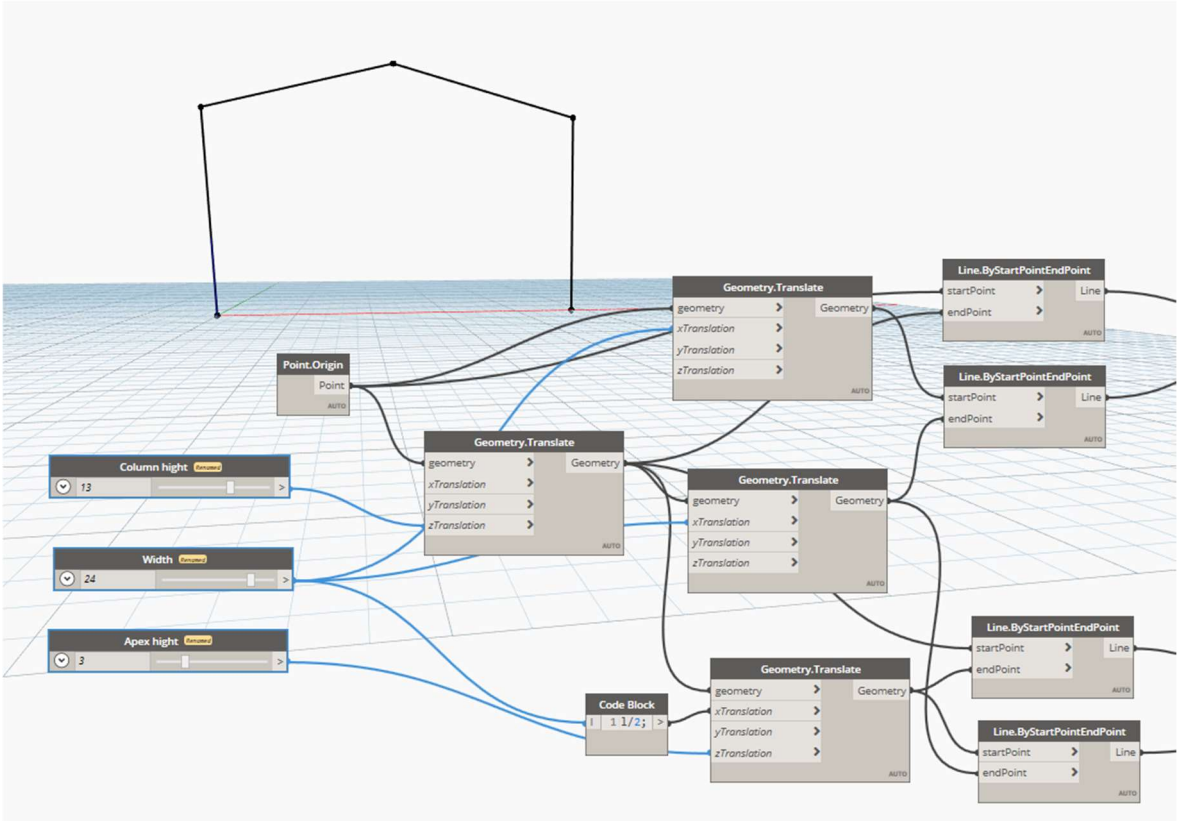


Figure 34. Creation of 2D geometry

according to the resulting geometry lines, an analytical model is built in the robot for this, it is enough to select the lines using the nodes, you can see it in the figure 35:

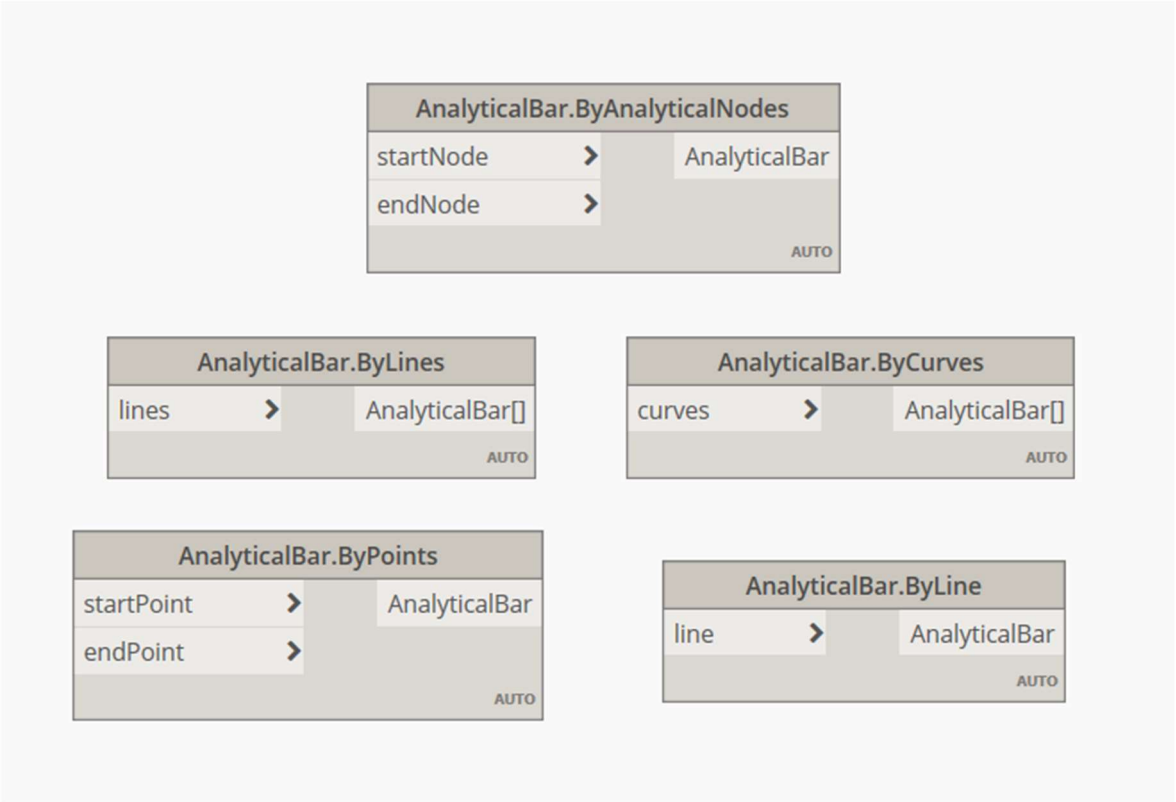


Figure 35. Nodes for creating an analytical model in Robot

After drawing the lines, it is necessary to create supports and set their type, you can see it in the figure 36; they are selected from the supports created in the Robot.

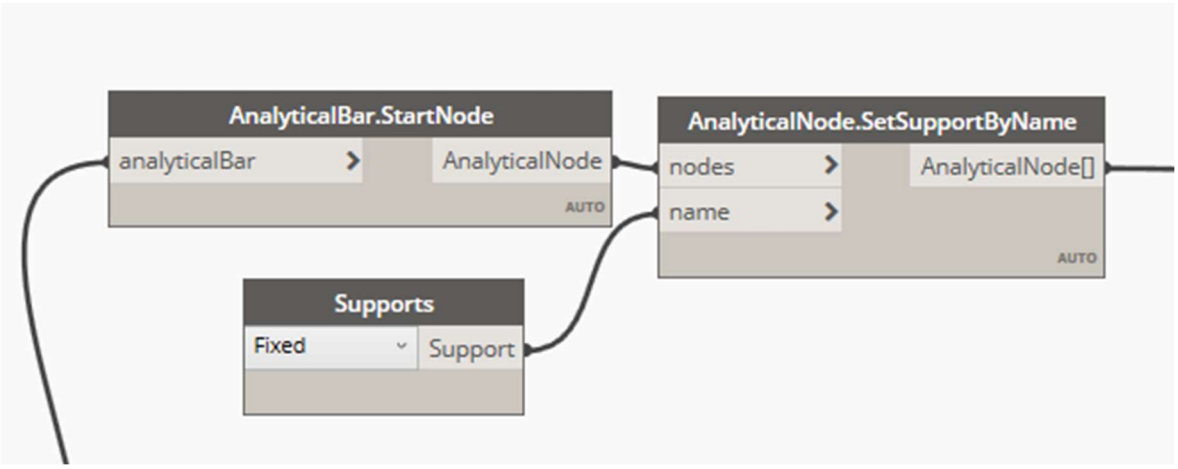


Figure 36. Support script

Sections and Materials are also selected from the extensive list of available elements in Robot. Sections can be directly loaded from the base library or selected from those that are available in this model specifically, you can see it in the figure 36.

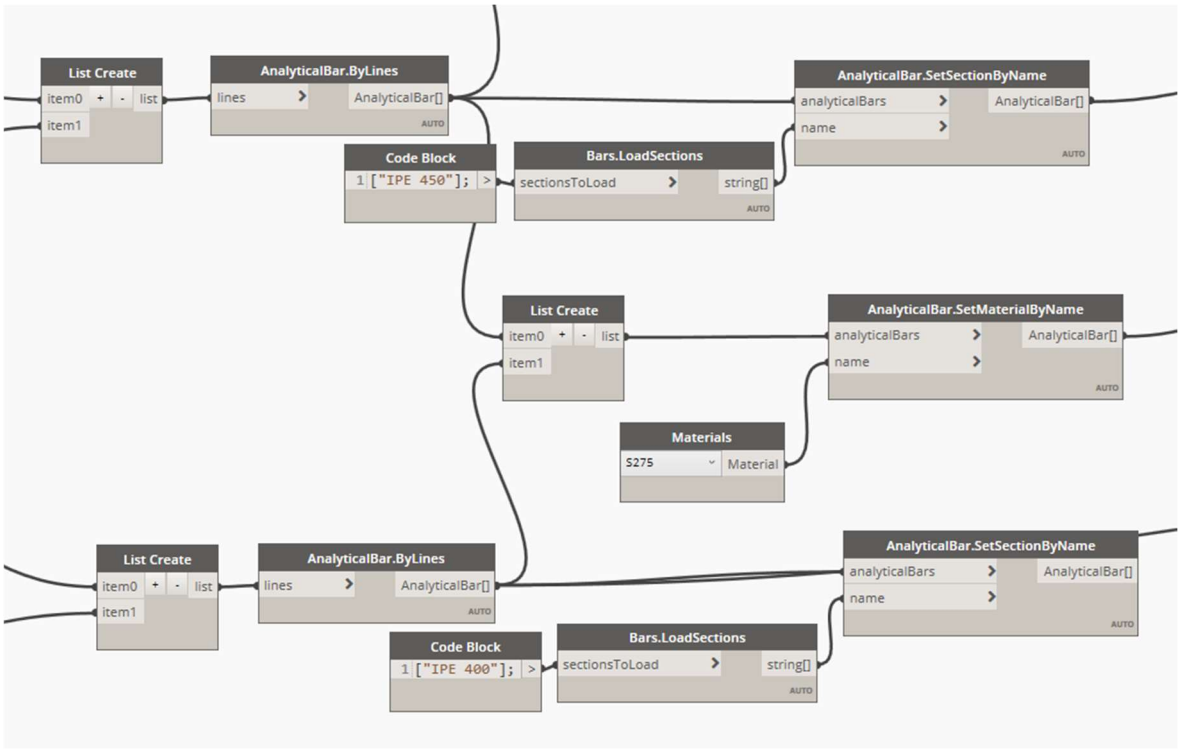


Figure 37. Script for creating sections and materials

The creation of loads consists of creating a load type, name, and load value, you can see it in the figure 38.

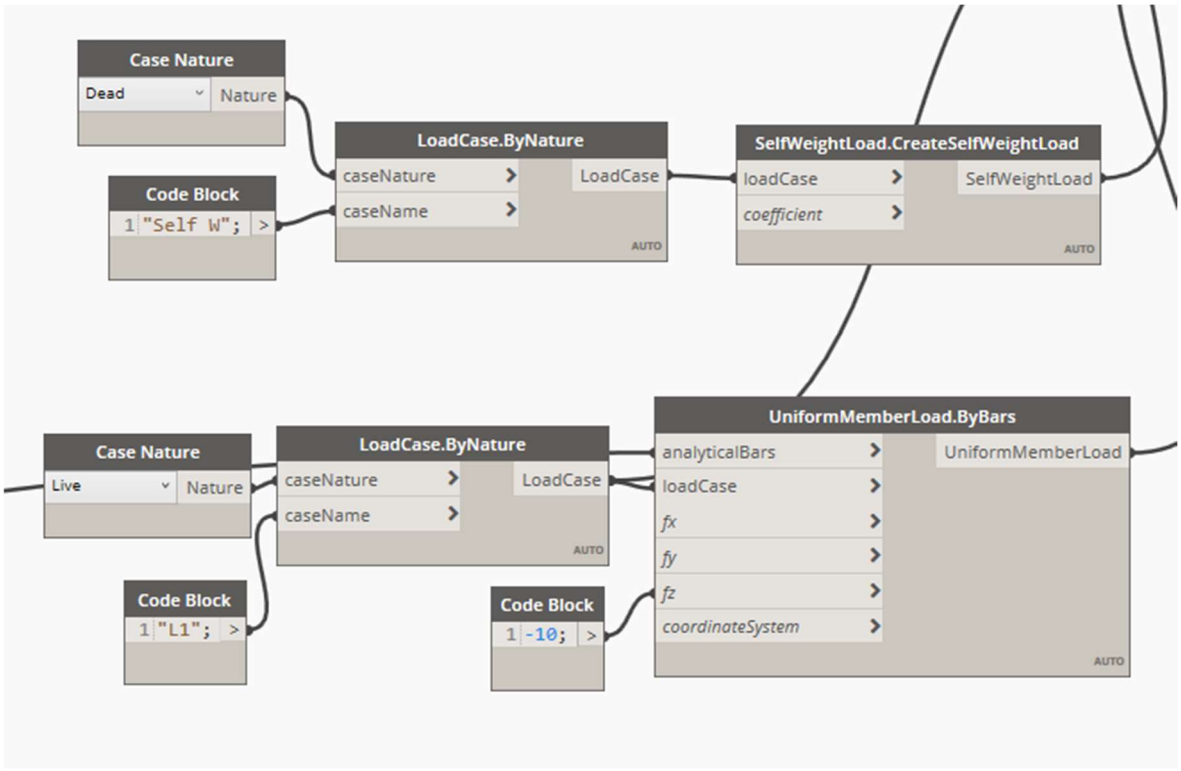


Figure 38. Creating loads

After creating all the necessary elements for the calculation, you need to create a sheet where there will be: all analytical nodes (such as supports), all analytical lines, and all types of loads, as well as their values (not including the value of their weight load) , you can see it in the figure 39.

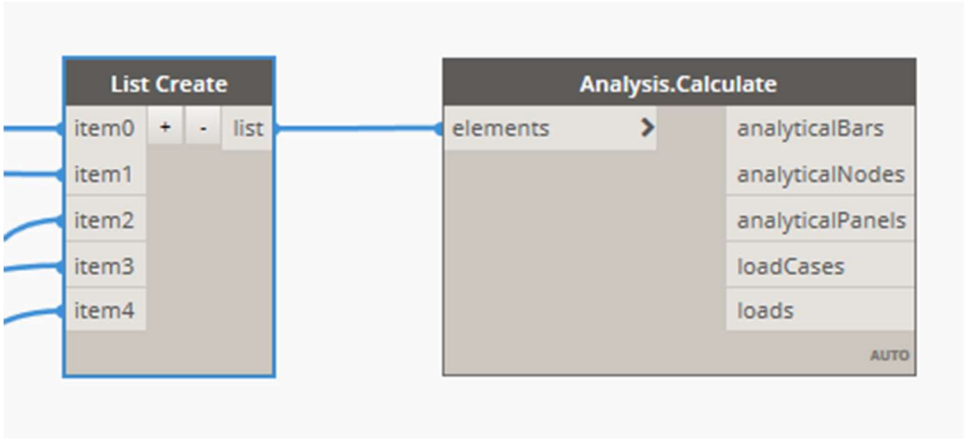


Figure 39. Model analysis in Dynamo

After the analysis, it is possible to obtain different analysis values inside the Dynamo, as well as simply third the diagram inside the Robot or continue working with the model inside the Robot, you can see it in the figure 1.

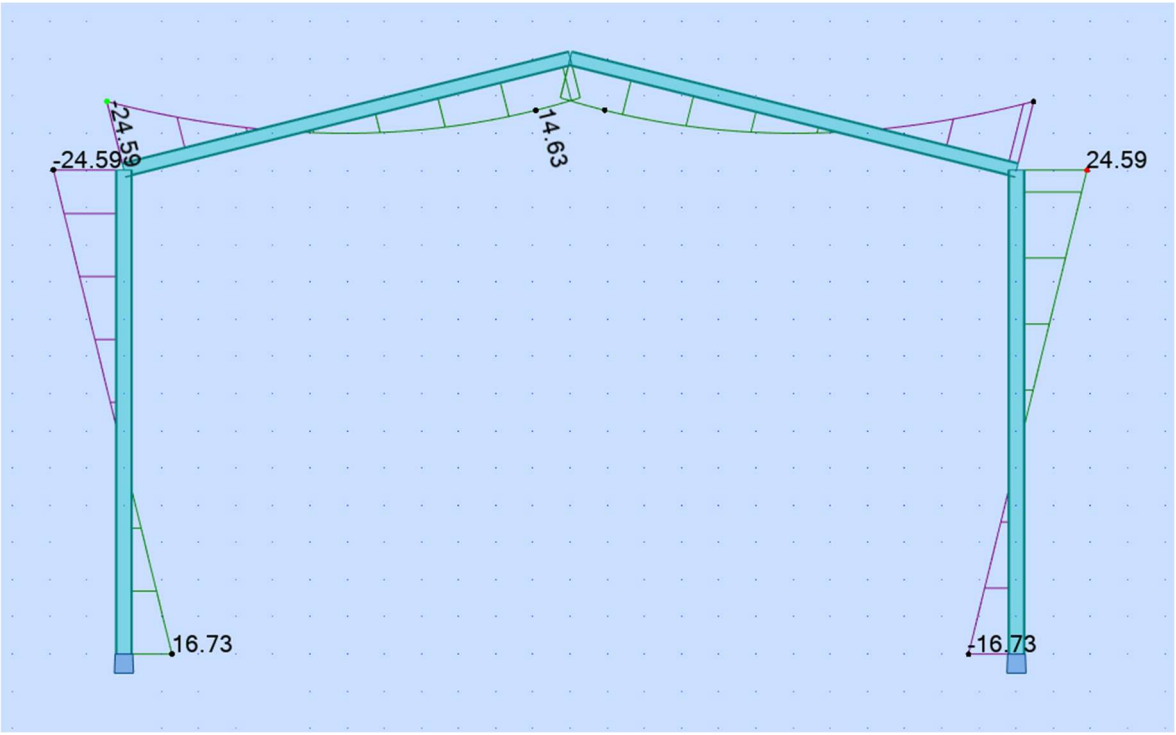


Figure 40. Result in Robot

3.2 Grasshopper

3.2.1 First connection

The process for the first steps with grasshopper is quite massive. At the first, you must have Rhinoceros 6 (it also possible to use an older version, but you should add to upload and installer Grasshopper), after that you should have Tekla structures student or educational version, and it the end you should get and upload Tekla Grasshopper live link for your version of Tekla. Important to notice that there to different live links for the educational and sale version of Tekla, to upload the last one you also should have “Tekla Maintenance”.

Grasshopper has got its website where is possible to find some guidelines for basic geometry work. Also, on the Tekla website, you can find an article showing an example of Tekla Grasshopper live link work and a description of all Tekla nodes.

The interface in Grasshopper is designed for quick use, all the nodes are arranged by the category, each has its picture, which is quite quickly remembered, while it is possible to change the display to the text version. The output from the nodes can contain several elements at one to choose from.

All Geometry created in Grasshopper transfers in the Rhinoceros window, BIM elements can be seen only in Tekla.

Considering the topic of Add -ons for Grasshopper, there are quite a few of them, unfortunately, there is no special interface for searching and installation and inside the program. But the list of all add – ons with description and a link for download is quite easy to find on the official Grasshopper website. The installation process is also straightforward.

Grasshopper can write a script using three different text programming languages: Visual Basic, C#, Python 3.

To make a conclusion grasshopper is not so friendly for new users in one way…

3.2.2 Work with difficult shapes

Wooden Arc.

The process of creating non-ordinary geometry for Tekla using the Grasshopper is divided into 2 stages:

1. creating the geometry.
2. Moving it to Tekla.

The main node for this process is "Item", you can see it in the figure 1, which exports geometry from Grasshopper to Tekla. To do this, the node needs to draw the geometry that needs to be transferred to the flow and, if desired, additional attributes.

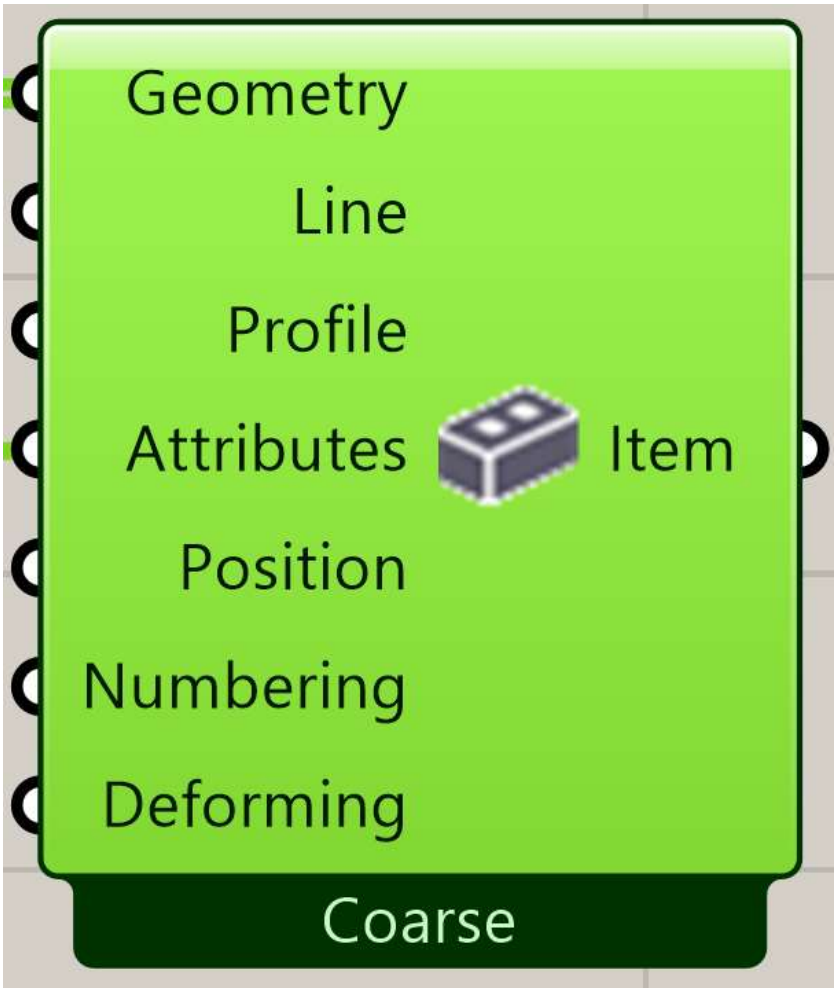


Figure 41. Node for creating a geometry element for Tekla

As a geometry element, you can see it in the figure 42, that is transferred to Tekla, a conditional wooden frame was created. This geometry can also be observed in Rhinoceros. The process of creating this geometry is divided into 2 stages:

1. Creating a single geometric figure.
2. Ccopying and mirroring it.

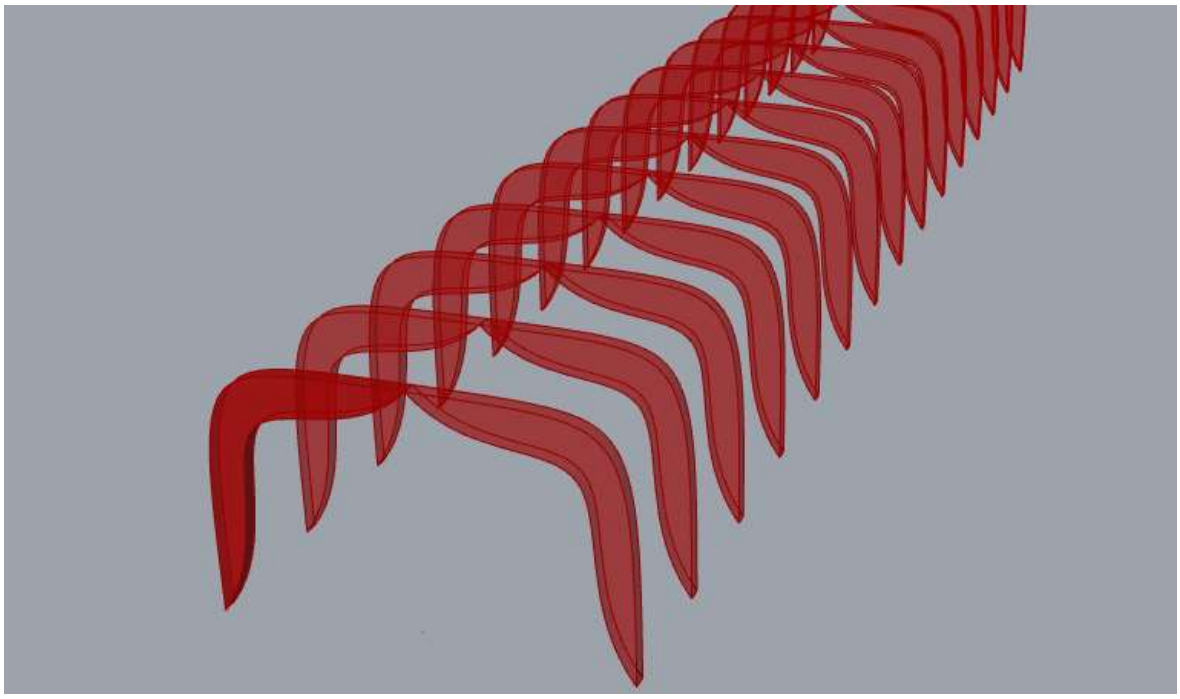


Figure 42. Final Geometry for Export to Tekla

Single geometry creation, you can see it in the figure 43, is about creating a solid object. It breaks down into stages, creating control points, creating lines, adding points on lines, creating curves by points, merging curves into a plane, and extruding a plane into a solid object.

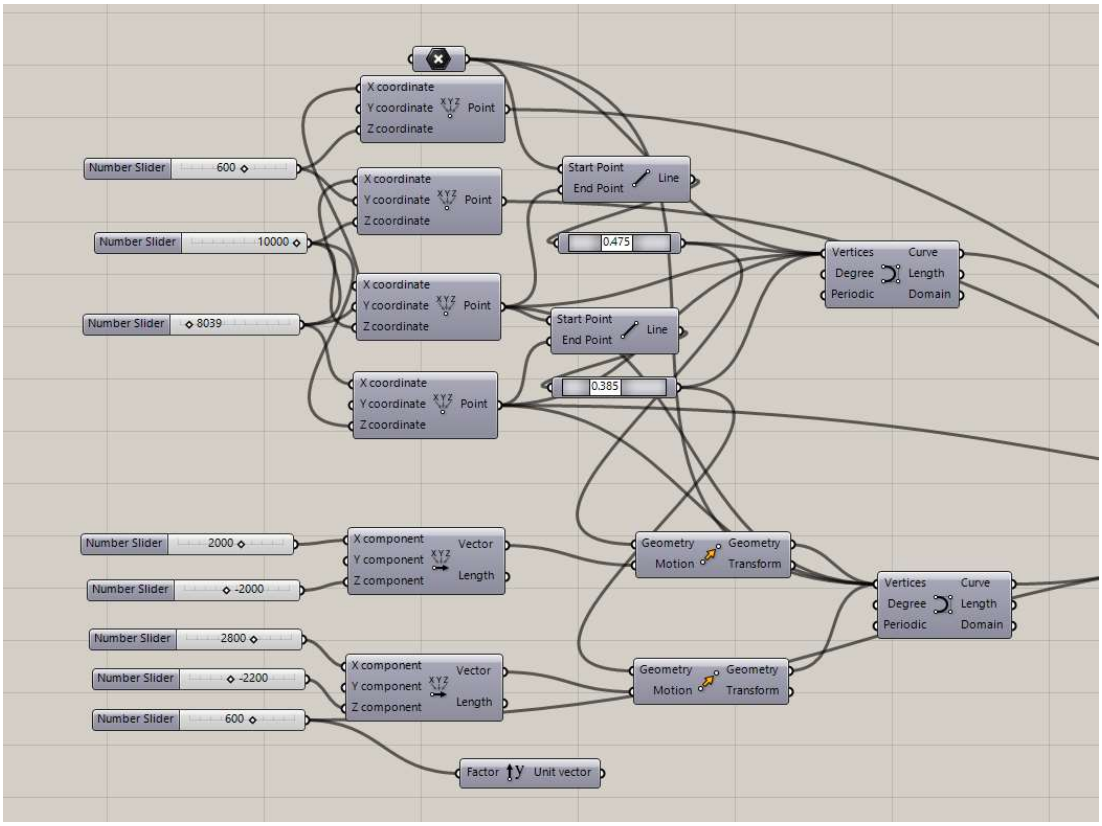


Figure 43. Script for creating a single geometry element

The copying stage, you can see it in the figure 44, consists of creating a series of numbers and copying the original body according to a given vector to the file of a series of numbers. After that, the copied elements are mirrored along the plane.

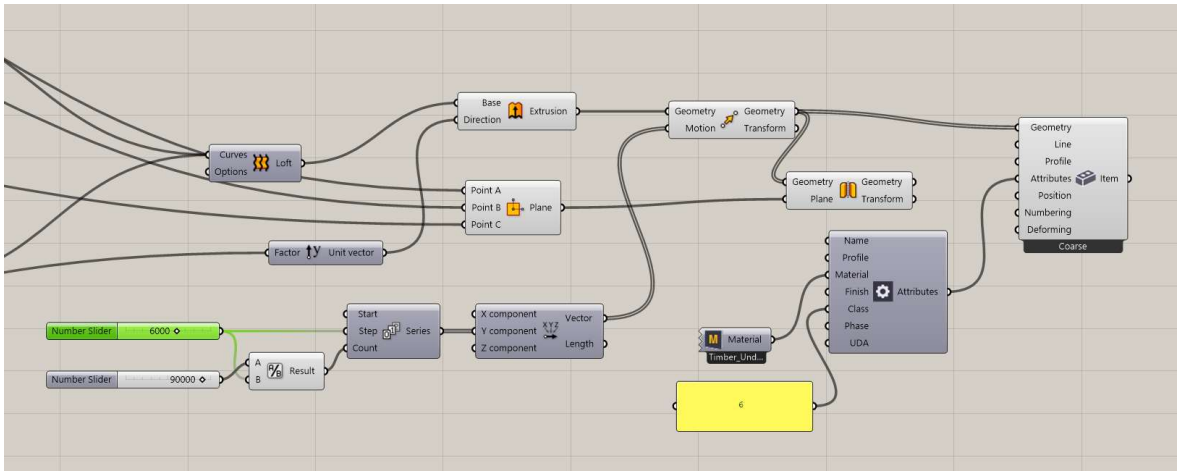


Figure 44. Script for copying and mirroring and exporting to Tekla

After the geometry is created, it is delivered to the "Item" node, and a node for attributes is created where additional parameters such as class, material, profile, etc. are specified if desired. You can see the result in Tekla in the figure 45.

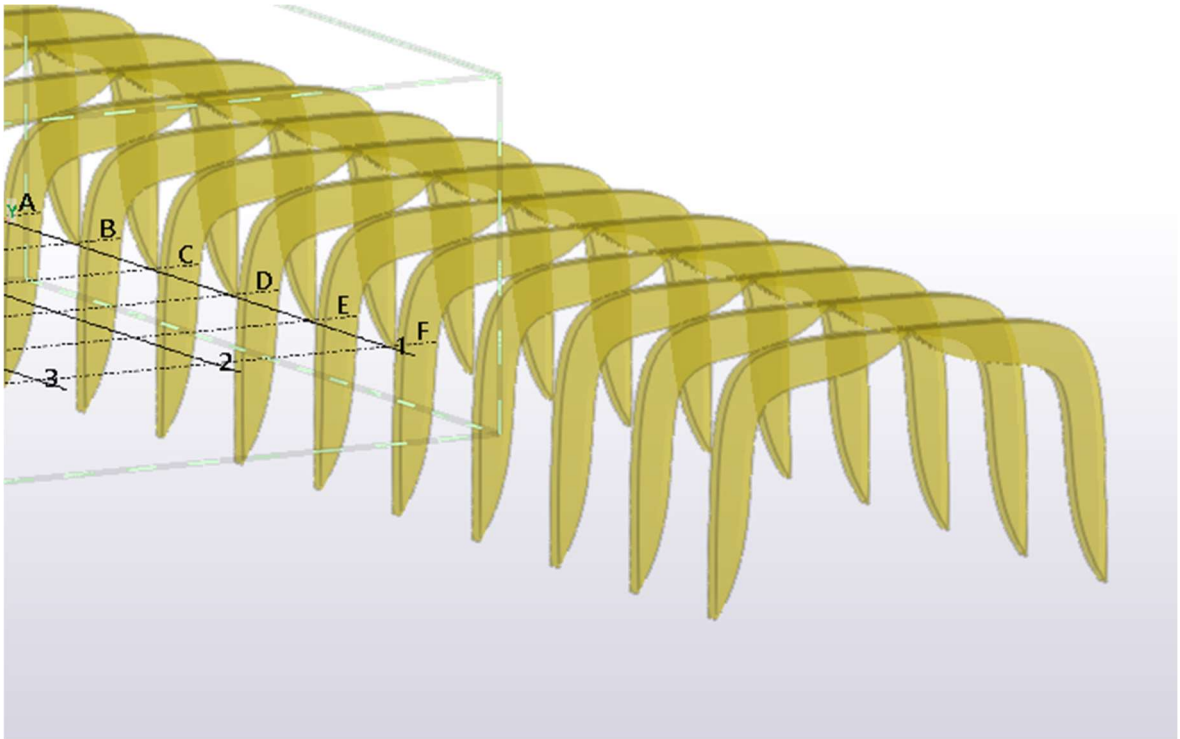


Figure 45. Result in Tekla

Space frame

As a result, a script was created to create a space frame on a given surface inside Rhinoceros, and Tekla, the plane was created inside Rhinoceros. To create a frame on a plane using a grasshopper, you need to create the plane itself. This can be done in several ways, immediately in Grasshopper or in Rhinoceros, also if you need to create a structure on the existing surface inside Tekla, you can recreate this surface in Grasshopper using the Tekla Line, Tekla point nodes, you can see it in the figure 46.

Figure 46. Nodes for importing lines and points from Tekla

After that, you can start writing the script. The script itself can be divided into 4 parts:

1. Creation of divisions on planes and the subsequent creation of different planes along with these divisions, you can see it in the figure 47.
2. Creating midpoints of normal for these planes, you can see it in the figure 48.
3. Regulating lists and creating lines, you can see it in the figure 49.
4. Formation of the frame in Tekla along the established lines, you can see it in the figure 50.

In the first paragraph, a segmentation is created along a given surface, after that the plane is divided along with these segments, and then it is divided into small sub surfaces.

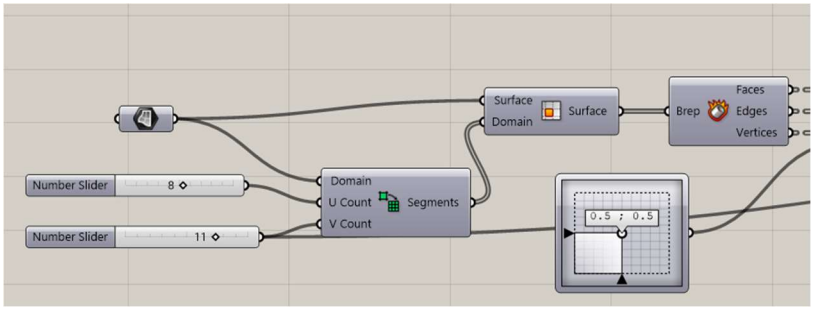


Figure 47. Dividing a surface into segments

The second part creates points on these surfaces according to our coordinates, in this case, it is in the middle and after that lines are created whose starting points are these points and the direction is the normals from these points concerning their surfaces.

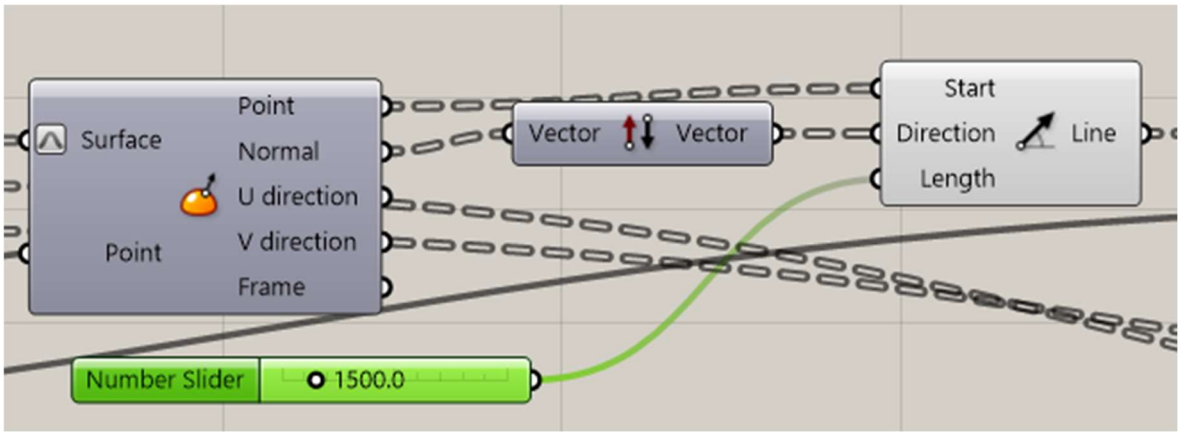


Figure 48. Creating surface normals

Part 3 is reorganizing the lists of items and creating new lists by segmentation. We need to create lines at the necessary points, for this we need to place the points in the necessary lists, so first, the lists are destroyed, and then they are created by a new one. After that, lines are drawn for the frame.

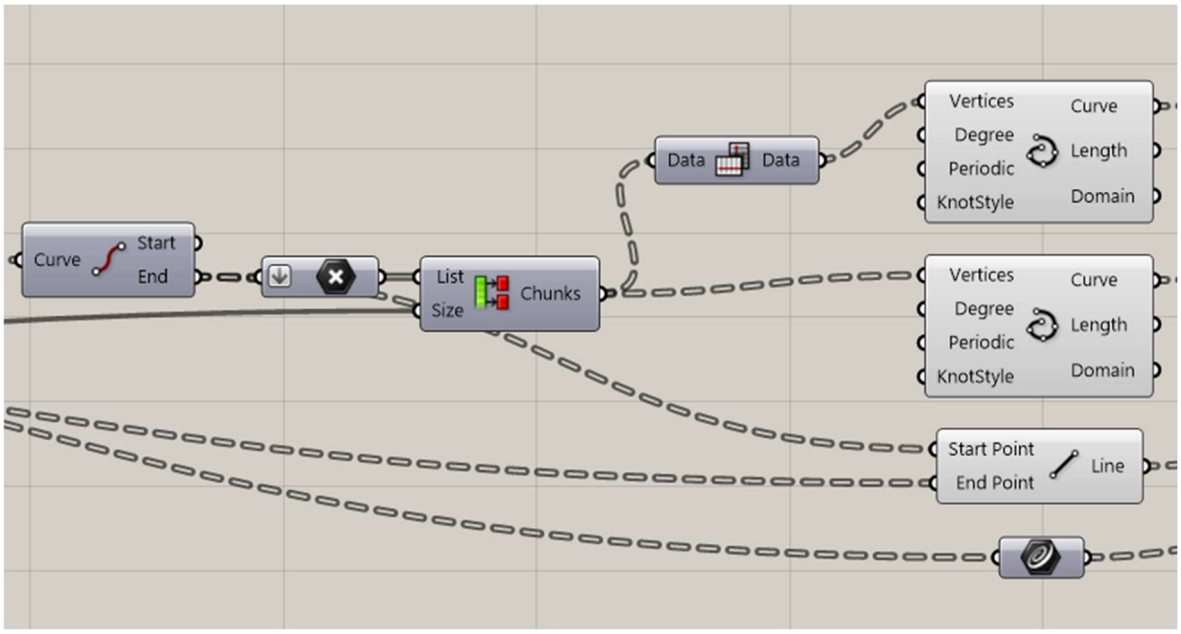


Figure 49. Reorganizing Lists and Creating Lines

After all the lines have been built, you can build a frame inside Tekla, for this you use the "Beam" node inside which all lines are fed, as well as attributes such as material, profile, and others.

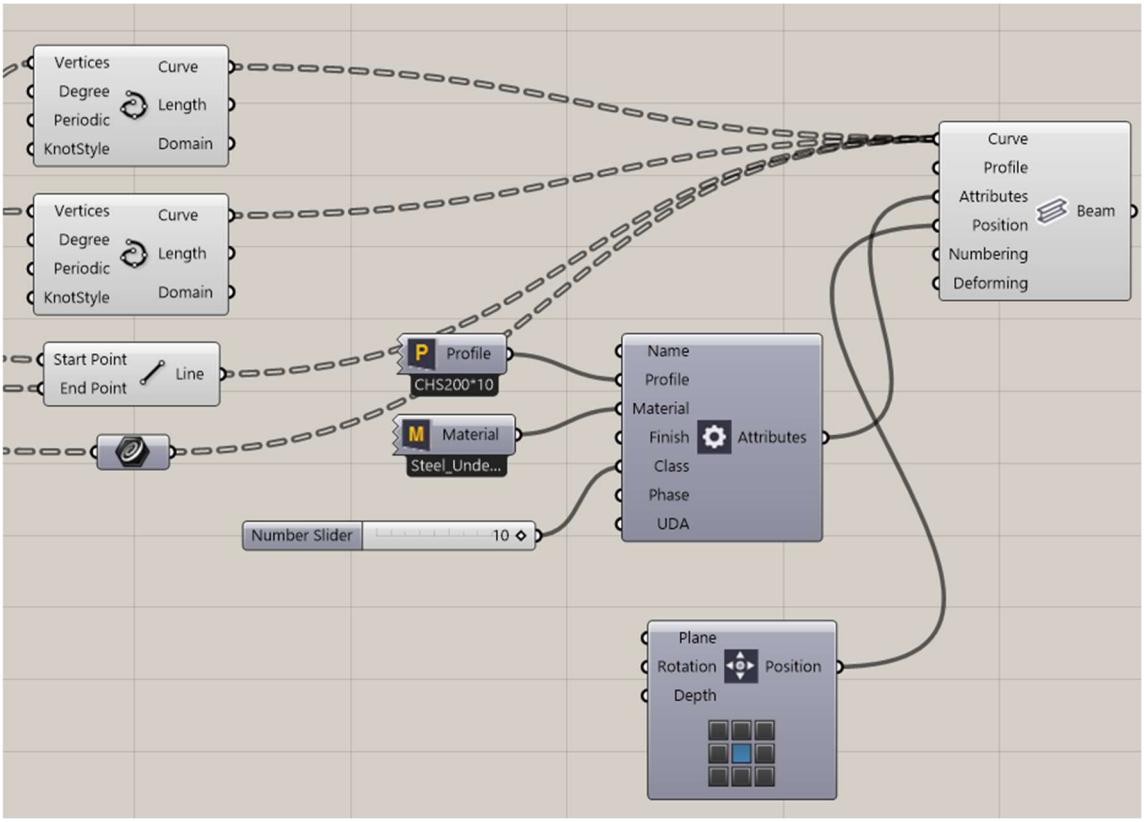


Figure 50. Formation of the frame for Tekla

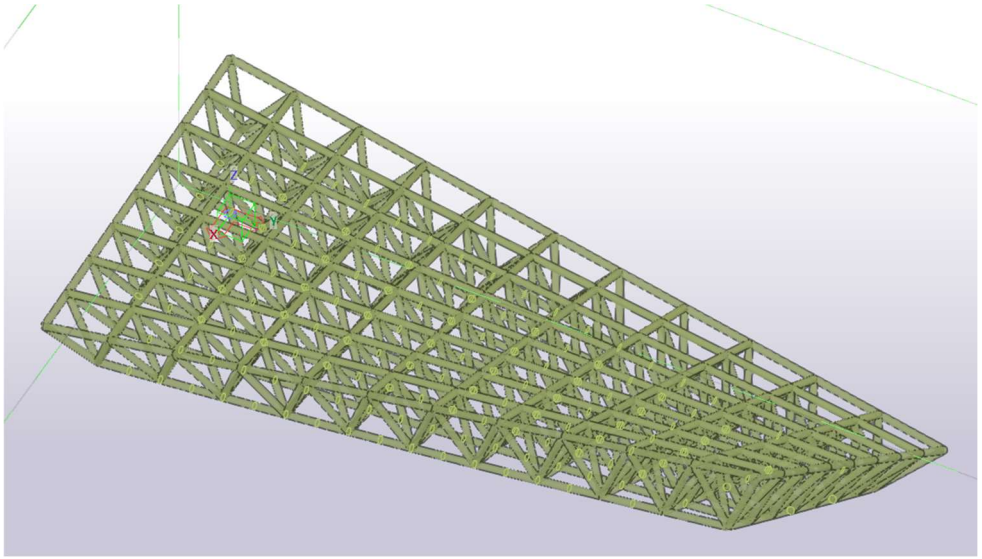


Figure 51. The result of the script in Tekla

3.2.3 Parametric portal frame.

The process of creating this script can be divided into 4 stages:

1. Creation of the geometry of the main framing lines, you can see it in the figure 52
2. Creation of bracing lines between the walls and the roof, you can see it in the figure 54.
3. Creation of bracing lines between the columns with different variations, you can see it in the figure 55 and 56
4. Creation of framing and connections inside Tekla , you can see it in the figure 57, 58, 59.

Line geometry is created by creating points along with the coordinates and creating lines between them, and further copying geometric objects using the "move" node

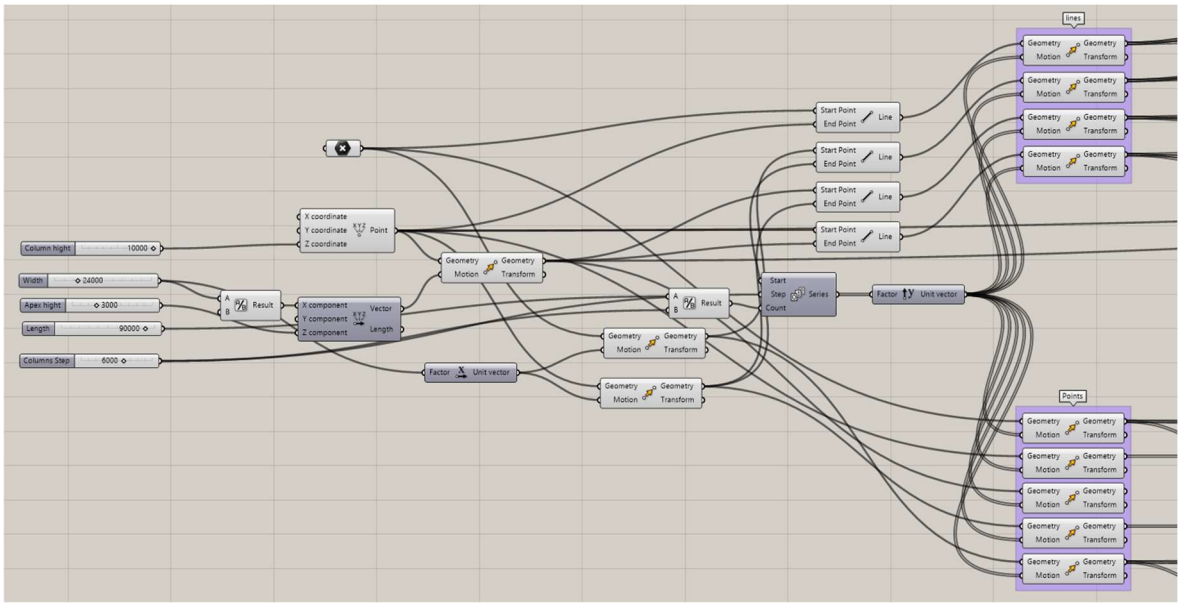


Figure 52. Script for creating the geometry of the mainframe

An important element for copying a 2D portal frame is the "series" node, , you can see it in the figure 53, which will allow you to create the generation of numbers with the desired step. In this case, you need to set the number of elements and the step. The step between the columns will act as a step, and as the amount of the length of the building divided by a step.

Figure 53. Node for creating a sequence of numbers in Grasshopper

creating a brace between walls and a roof is an identical process, so it is enough to create for one and simply copy for the other. The creation process consists of creating reduced lines and further segmentation by points, and the creation of lilies between these points.

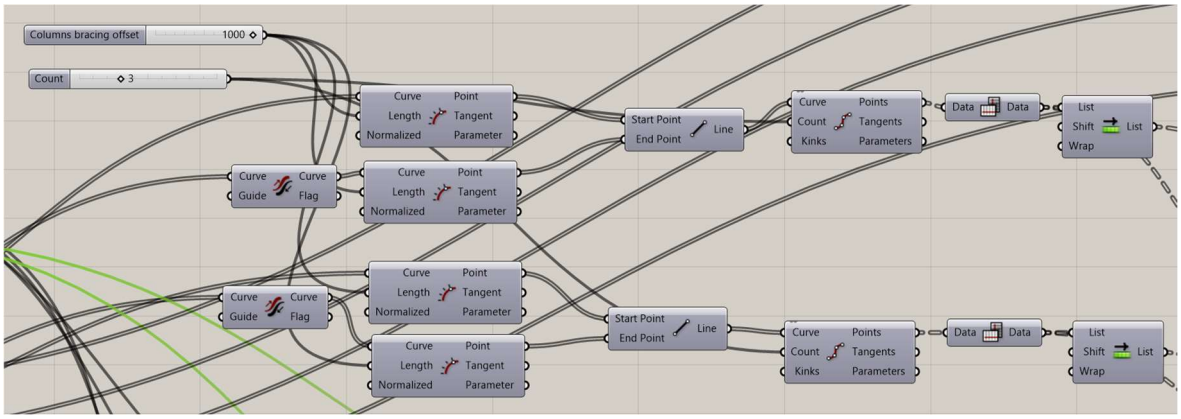


Figure 54. Script for creating a roof brace

To create the brace between the columns, a Python script was used that creates 3 different variations of the connections. The beginning and endpoints of the columns are fed into the script itself, and a slider showing which option will be used. The script uses the same command syntax as Grasshopper itself.

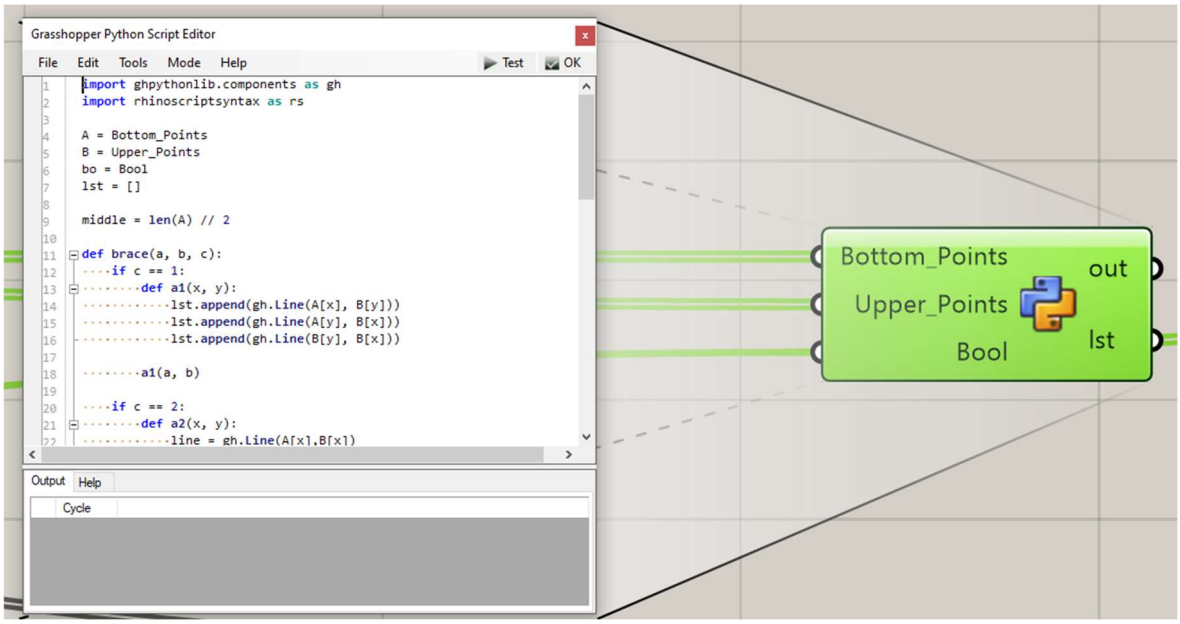


Figure 55. Python script created in Grasshopper for Bracing between columns

After that in Rhinoceros, the geometry of the lines will be created along which the structure in Tekla will be created.

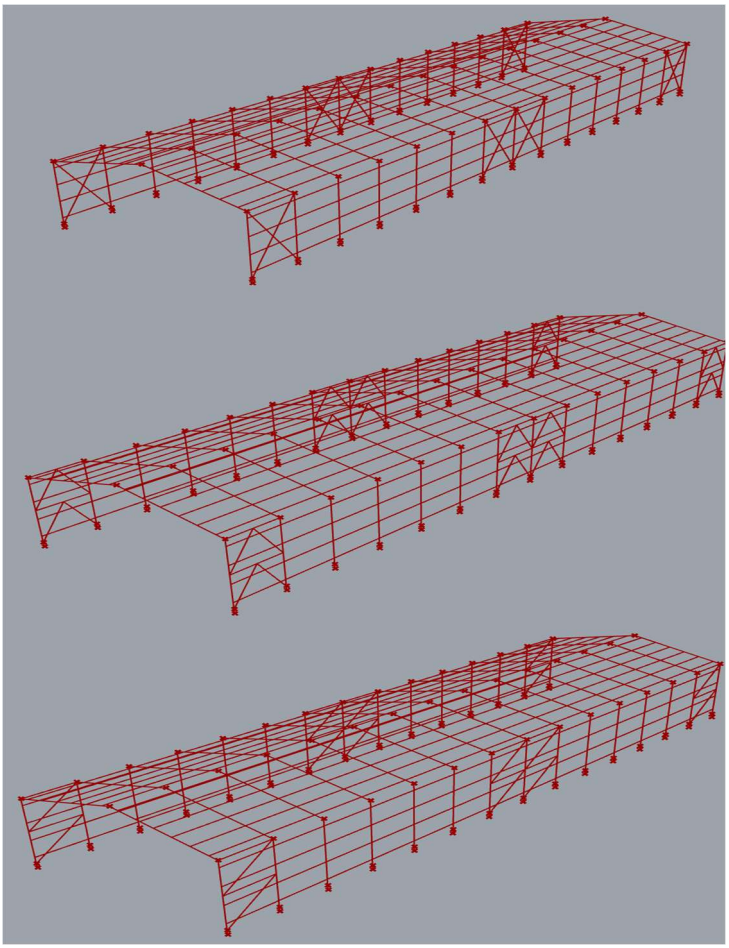


Figure 56. 3 Different geometry variations

To form the skeleton in Tekla through the Grasshopper, two main nodes from the Tekla node package for the Grasshopper will be used: a beam and a column, you can see it in the figure 57. Their main task is to create a structure along curves. It is also necessary to set a profile for the structure, this is done using the "Profile" node, which allows you to use the Tekla profile library and edit the section.

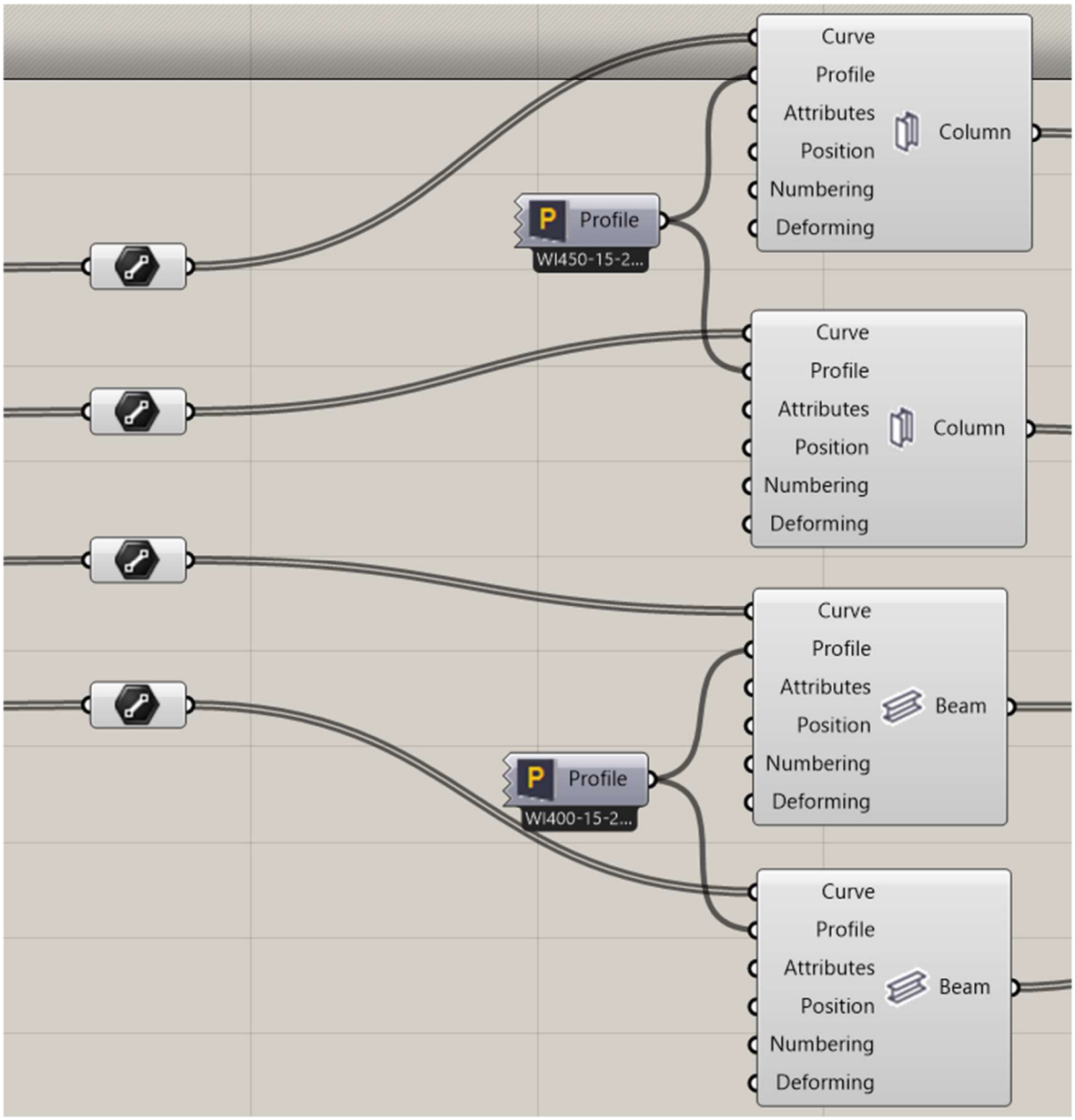


Figure 57. Creating beams and columns for Tekla

The same method is used to create a frame for a brace, specifically for cases of steel and roof it is necessary to carry out additional manipulations to adjust the position and angle of rotation and offset. For these purposes, the "Position" node, you can see it in the figure 58 is used in which you can immediately select the position of the frame to the line, rotation, and offset. To set the rotation angle for a brace on the roof, you need to set the desired rotation angle, for this, a node is used that shows the value of the angle between the vectors.

Also, it is necessary to remove the last element from the list of bracing lines, since it is unnecessary for this, a node is used that removes an element from the list at a given index.

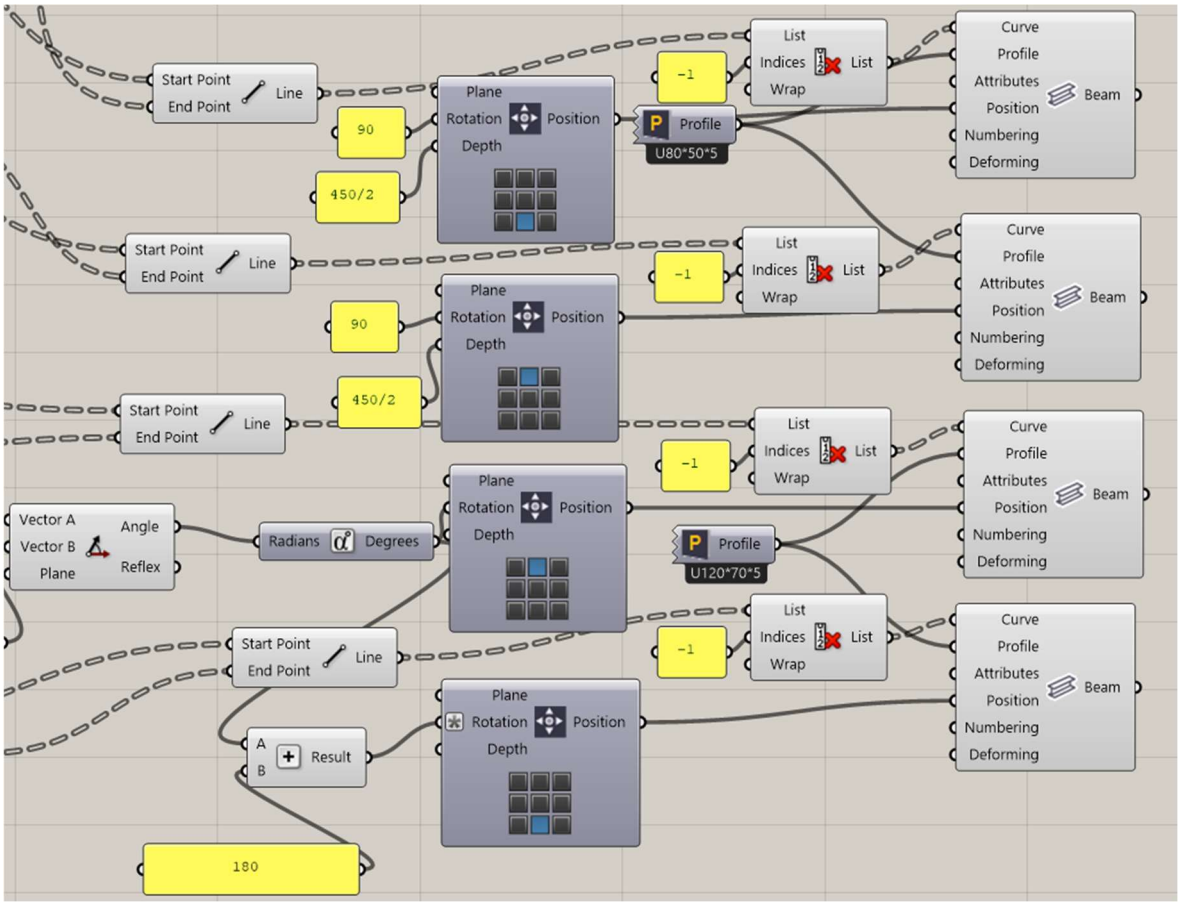


Figure 58. Creation and customization of bracing for Tekla

To create the foundation, the "Pad Footing" node is used, you can see it in the figure 59. The principle of its operation is the same as for a beam or a column, a line is formed along the length of which the foundation will pass, and a section and its dimensions are created.

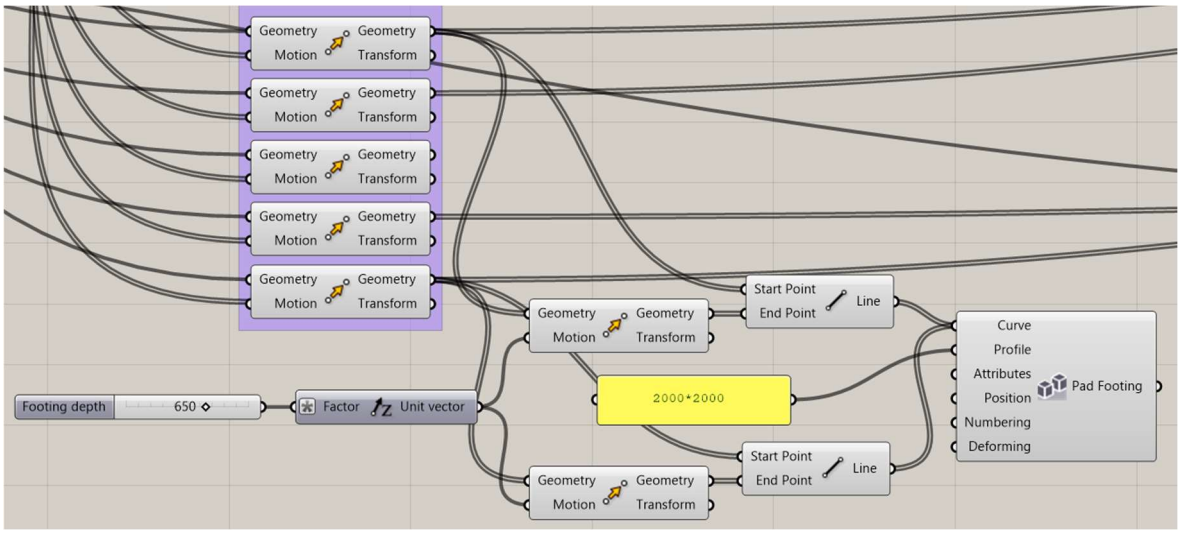


Figure 59. Foundation script for Tekla

Two important components for creating connections between steel structures are nodes: Component and Component Catalog, you can see it in the figure 60. The component catalog gives you complete access to all components available in this Tekla model. And accordingly, when you select a component of the connection of the node, the component transforms the node to create a connection. In the future, all that needs to be done is to attach the frame elements to this node that fit these connections.

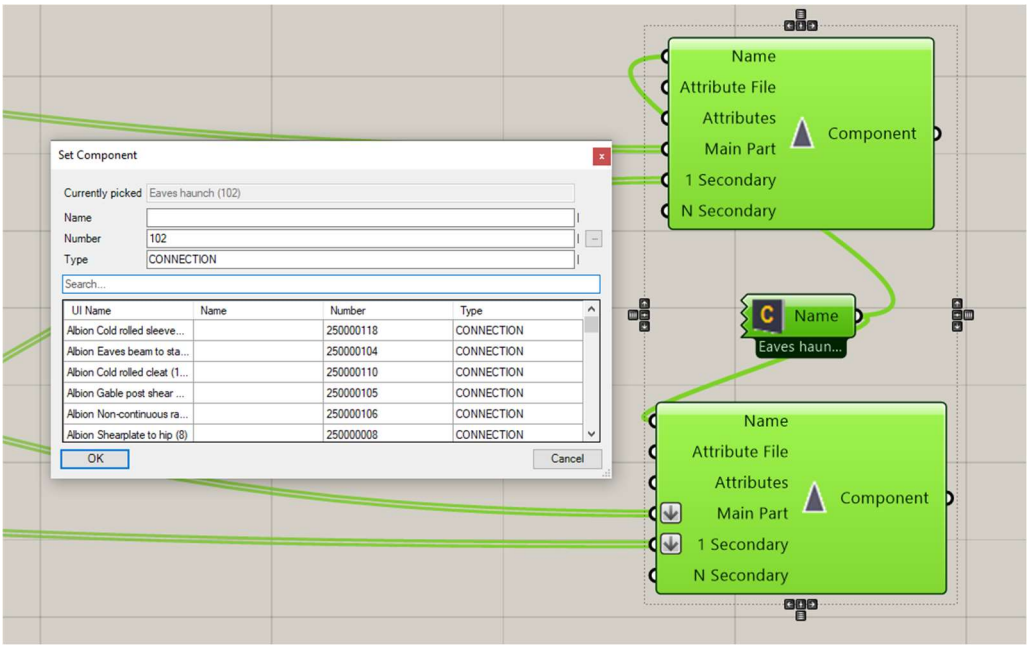
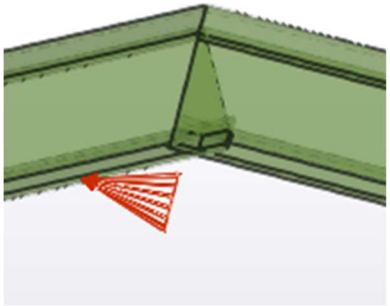


Figure 60. Creating connections between beams and columns

It is important to note that this node is not perfect and there is a possibility that it will not work correctly when copying the node, you can see it in figure 61; if you repeat the procedure, the problem may go away. Also, there are cases when it is not possible to build a connection. In this script, an attempt was also made to create a ridge connection, but Grasshopper did not do it correctly and after repeated attempts to fix it, the urge to abandon this idea.



Picture 61. The faulty connection between beams

As a result of this script, you can see in figure 62, a parametric model of a steel parametric frame inside Tekla is obtained.

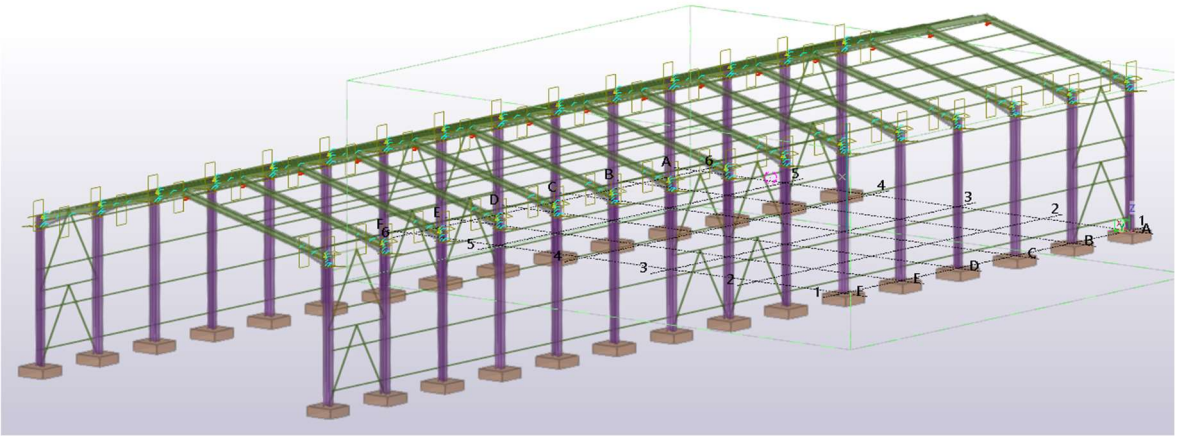


Figure 62. The result of the script

3.2.4 Structural Analysis.

For structural analysis via a grasshopper, an additional package of nodes from "Karamba3d" is used. With the help of this add-on, structural analysis is carried out immediately in Grasshopper and Rhinoceros.

The process itself is divided into 3 stages:

1. Creation of geometry, you can see it in figure 63.
2. Grouping of characteristics for analysis. You can see it in figure 65.
3. Analysis, and visualization of analysis, you can see it in figure 66.

Geometry creation for this model is completely identical when working with a portal frame, only here you create a 2D frame. The difference is only in units of measurement, Karamba3d calculates in meters, so the model needs to be created in meters. First, the main point is created at the origin, the portal frame is copied over the key points, and then lines are created between them.

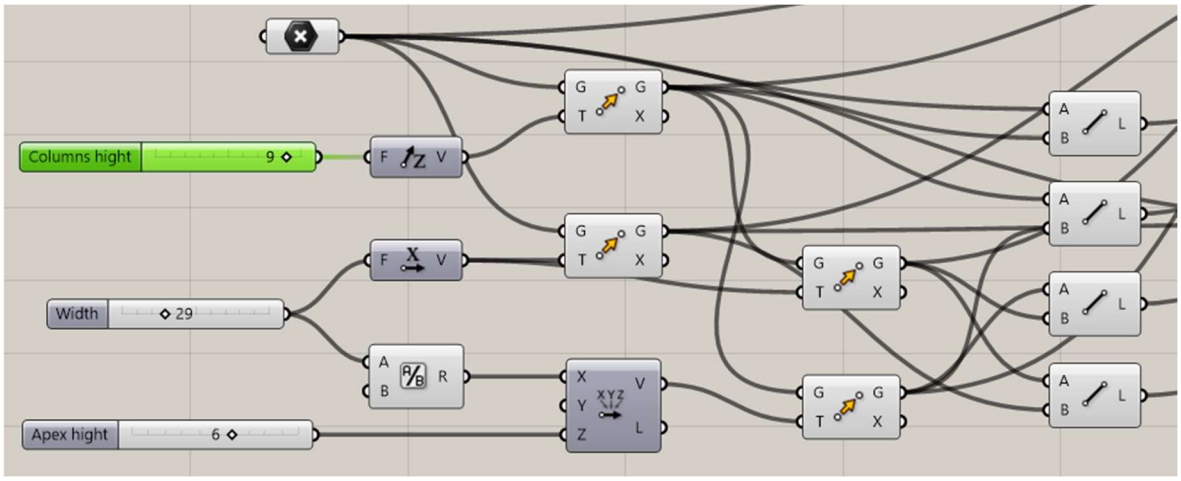


Figure 63. Script for building 2D geometry lines

The next step is to collect all the necessary information for analysis, at this point loads, supports, sections, material, etc. are created. The key node is "Assemble model", you can see it in figure 64, all information is supplied to it

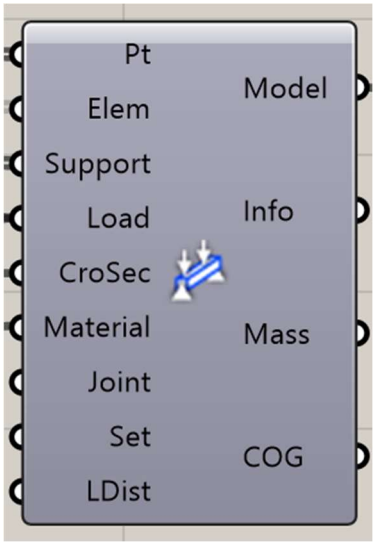


Figure 64. Assemble model node

This node must connect all points of the model and its elements, which are formed from the lines of the same model. Further attributes are optional but required for the analysis to work correctly.

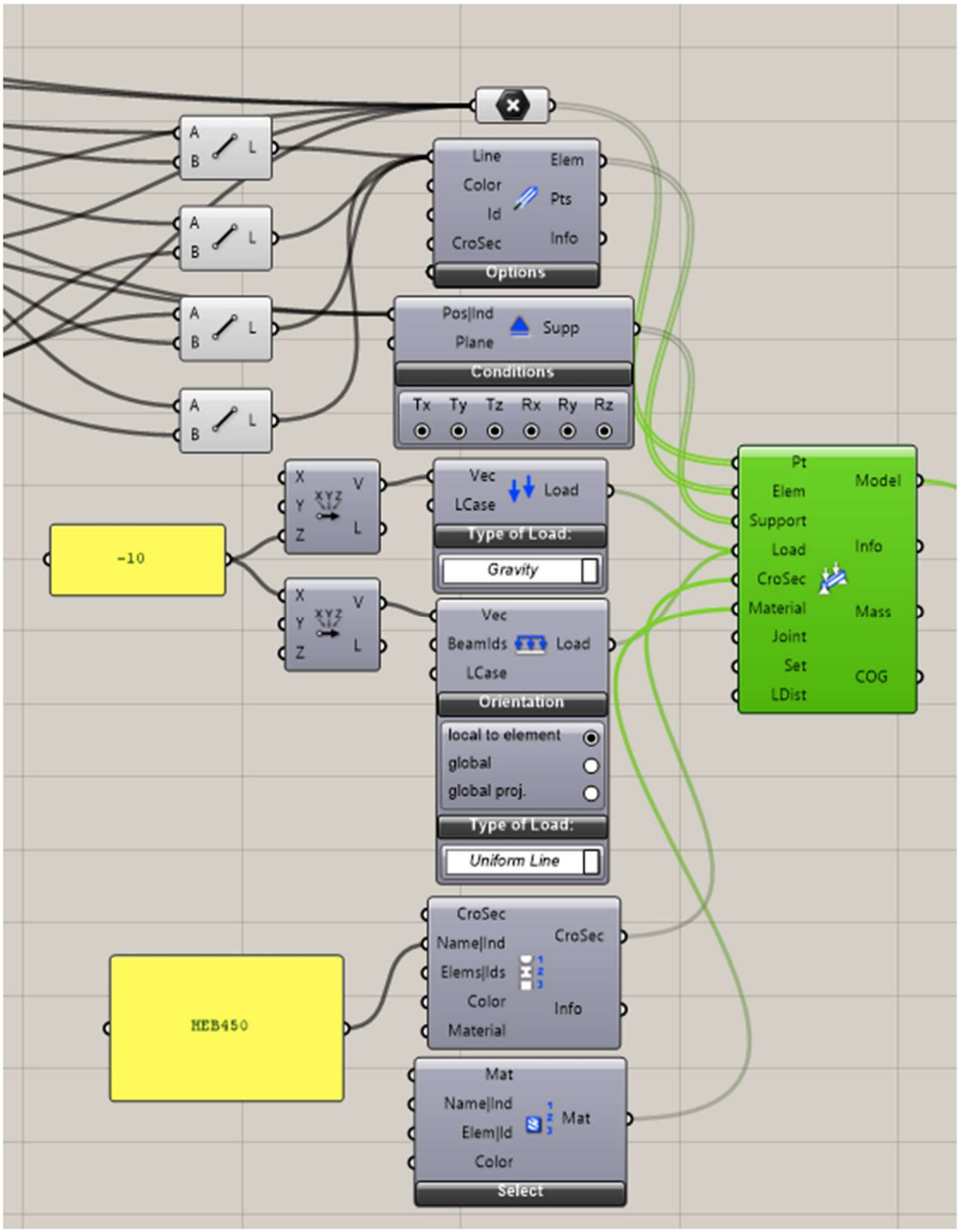


Figure 65. Script for creating attributes for analysis

Karamba3ad allows you to use many ways and variations of attributes from creating different variations of loads to choosing or creating your subtype or material characteristics.

After creating all the necessary elements and attributes, analysis, and subsequent projection of deformations, diagrams, and their values are carried out. There is a wide range of nodes for this with a wide range of possibilities.

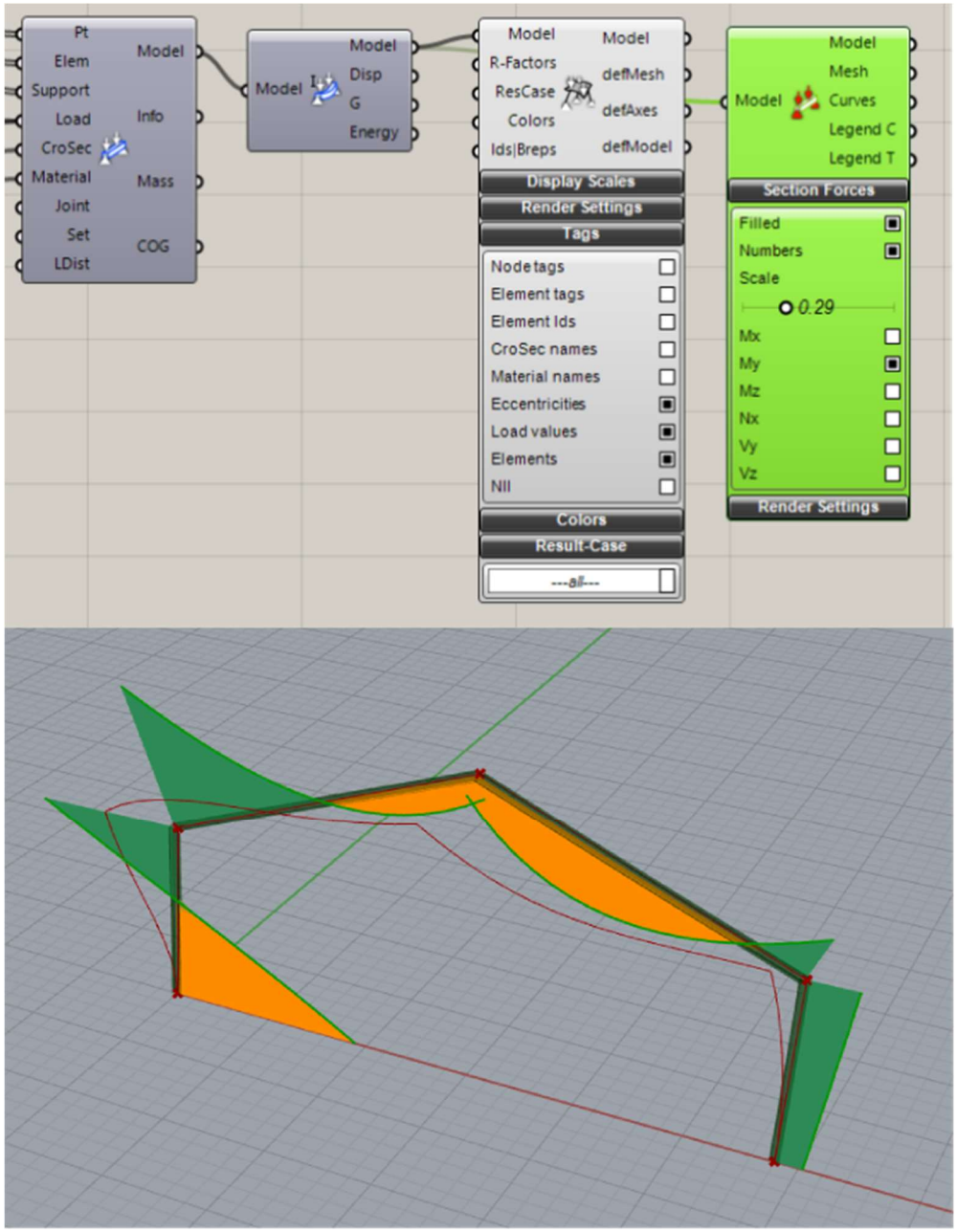


Figure 66. Analysis result

4 Results

4.1 First Connection

At this point, I would like to highlight the main points of comparison that do not fall under a specific model, either speak about software in general or are true for each model.

4.1.1 Finding software information and tutorials

First, it is worth noting the moment of the first connection to this or that software. This process is made much easier for Dynamo because all programs are from one developer and are quite automatically compatible with each other.

In Grasshopper, you need to connect Tekla and Grasshopper, this is done quite simply and quickly. But due to different developers, another problem arises, that the support is different and information on some questions is more difficult to find than on the Dynamo website.

This concerns more questions related to the collaboration of Tekla and Grasshopper, when looking for information about the basic capabilities of Grasshopper nodes, since the program itself is older and the community base is larger, more often it is easier to find or get help.

The same goes for video tutorials or guidelines. Information on the joint work of Tekla and Grasshopper is quite problematic to find and you have to figure it out on your own, and in the case of the Dynamo, which was originally developed to work together with Revit, there are more various manuals for work.

4.1.2 Purpose of the software

The main difference between this software is its tasks. For the Grasshopper, this is Parametric Modeling, that is, the construction of Geometry. Dynamo has more possibilities as software for visual programming.

Since the software was sharpened to work with Revit, it has many different variations of Element regulation, filtering, and processing, and besides that, species regulation, filtering of sheets, templates, etc.

Grasshopper is specifically geared towards parametric modeling and geometry creation. It is important to note that the functionality can be expanded by installing additional packages.

Grasshopper gives more functionality in the early stages of design, and Dynamo with Revit at later stages allows you to quickly prepare parts and drawings for documentation.

4.1.3 Additional Packages.

Grasshopper is an older software and during its existence, it has managed to accumulate a large community base. During this time, a huge number of additions of various functionality have been released from "Lunchbox", which expands the functionality of Parametric geometry modeling to "Karamba3D", which allows you to carry out a full-fledged structure analysis right in the Grasshopper.

Since the Dynamo is a newer program, it is worse with this, but it is possible to quickly and easily add your cost-meshed nodes and a group of nodes to the general environment of Dynamo packages.

It is important to emphasize that all packages rest on the functionality of the software itself and the originality of the developers. So, the advantage of certain packages for a particular software varies from time to time.

In terms of accessibility, all the packages are quite easy to find and download. Either on add-on sites, or for the Grasshopper on the "Food4Rhino" website, and for Dynamo in the program itself, in the package search menu.

4.1.4 The power and convenience of software

Going directly to the comparison of the functional of parametric modeling, it is very important to note the main advantage of Grasshopper over Dynamo. This is the speed of processing information by the program.

When it is necessary to perform operations on the generation or modification of the structure in which there is a huge amount of information, the speed, and safety of Grasshopper are much ahead of that of Dynamo.

Both software has 2 modes of operation:

1. Automatic - when all actions are performed immediately after changes.
2. On enable - when the script execution process is executed only after pressing the start button.

When writing all the scripts for this Thesis in the Grasshopper, only the automatic version was used.

In Dynamo, it was fast and immediately switched to the manual. Because the processing time began to take time and the program began to freeze or crash from overload with data loss after the last save.

As for the speed of use, it can be noted here that both programs have a node library and a node search. The library is implemented equally well in both programs, and the usability is divided more by taste.

As for the search reason, it is worth noting 2 points: the names of the nodes and the understanding of the search query.

In the first case, Grasshopper's Nodes have simple versions of the names as a human language, while at Dynamo they have the form of a programming language. And depending on the person, it will be easier for someone to remember something.

In the case of a search by the name of the node, the Grasshopper has a little better here, the Program finds the required node very quickly, even if there is no full observance of the name, Dynamo, in this case, sometimes may not find the required node by name.

Touching upon the issue of functionality, it is important to note that Dynamo has a block code and its wide functionality, which makes it possible to quickly write some command, take an element from the list by index or set the generation of a sequence of numbers. It is invoked by a simple double click of the left mouse button.

Grasshopper has a similar analog, but it cannot be called quickly and it does not provide such a range of possibilities as the code block in Dynamo, in it, you can visualize lists or set some kind of string, number, and numeric expression.

It is also worth noting the process of combining elements into one Node, in Grasshopper you can immediately connect several elements to the desired node, in the dynamo for this process you always need to create an additional node to create a sheet. In some cases, this process takes time and is inconvenient, in other cases, it allows you to quickly change the order of the elements in the list, find and fix the problem associated with the sheets of elements.

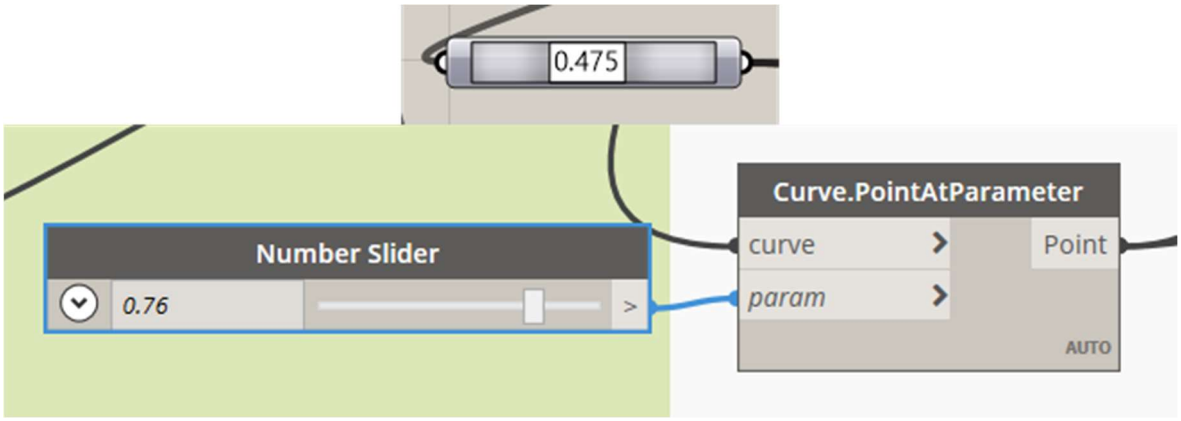
Summing up the power and functionality of the common points of both software, the Grasshopper will noticeably overtake Dynamo.

4.2 Work with difficult shapes

4.2.1 Wooden structure

The creation of geometry in Grasshopper and Dynamo is almost identical, nodes with the same functionality and task were involved, the only difference was in details such as

regulation by lists or as the "Point on a curve in a Grasshopper" node, you can see it in figure 67, which immediately included a number slider and a point function by parameter



Picture 67. Above a point on a curve in a Grasshopper, below a point on a curve in Dynamo

The main difference lies in the way geometry is created and exported to BIM software, you can see it in figure 68. In Grasshopper, this happens immediately on the generated geometry. In Revit, you need to create a family, and then place it in Revit.

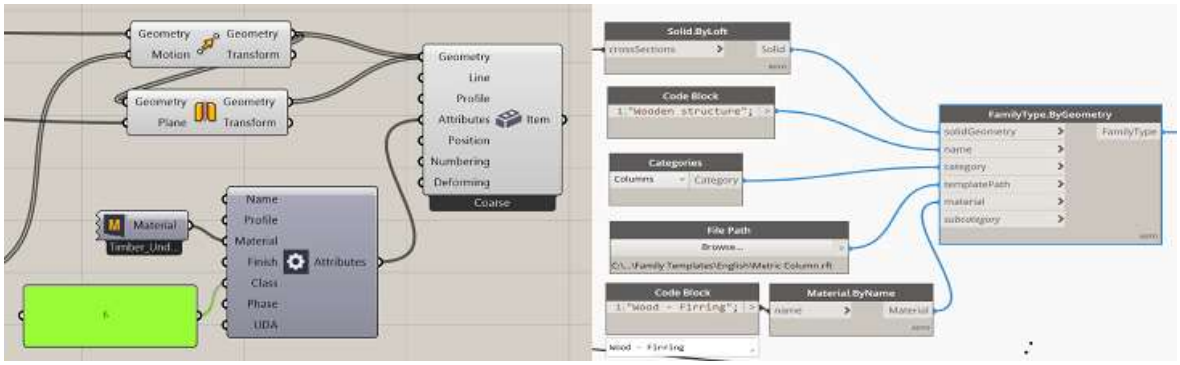


Figure 68. On the left is the formation of an element in the Grasshopper, on the right is the creation of a family in the Dynamo

The process in Revit is more laborious and time-consuming and does not bring benefits. When automatically changing the parameters of the created family, Revit often had errors and the program was closed.

The only advantage of this method is that a family can be created in Dynamo, and placed in a direct way inside Revit, and accordingly, when changing the geometry parameters of the family, the geometry of all placed elements will also be changed. But at the same time, to create another type of family somewhat different from the original, you will have to copy

the script and create a new parametric family, or copy the elements and set them a separate family directly in Revit.

In turn, elements in Tekla do not create a separate family, all that unites them is the formula of the object's shape, when directly copying an element inside Tekla and further changing the script parameters, the shape of the copied element will not change. If you try to change the formula, the position and shape of the object will change.

From this, it turns out that it is faster to create an element in Grasshopper and export it to Tekla, but if you need to arrange such parametric elements and further edit them, the process can be delayed. In Dynamo, an additional step is required to create a family, but if the placement is required directly and further change of parameters, it will be faster.

4.2.2 Space Frame

The main difference is the script creation method in Grasshopper and Dynamo, you can see it in figure 69. The script was first written in Grasshopper using the standard node library. When trying to do the same thing with the same method in Dynamo, the problem of the absence or lack of functionality of the standard node library was encountered. The key difference between the nodes in Grasshopper and Dynamo is that Grasshopper can give more information from one node, it has nodes that offer several output options to choose from, while the dynamo always has one.

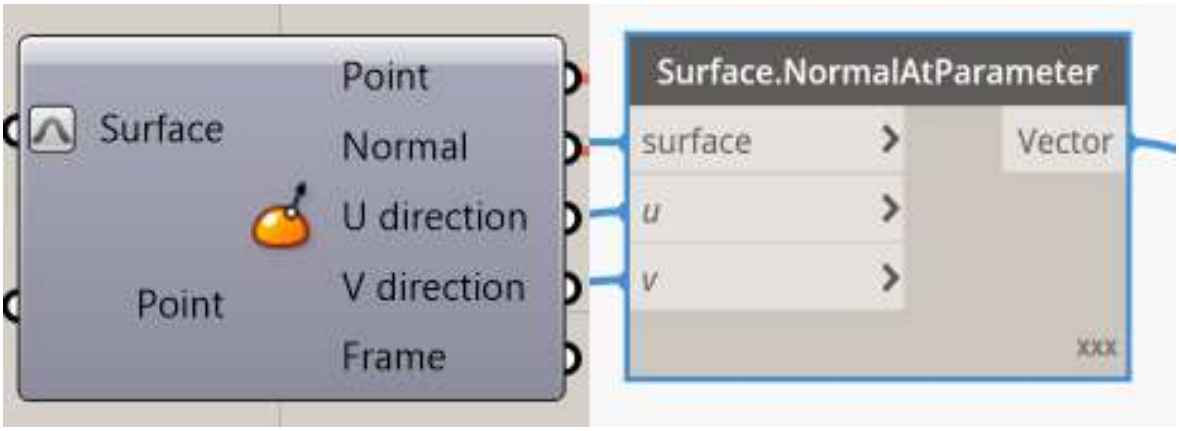


Figure 69. On the left there is a lot of information for the exit to the Grasshopper, on the right, there is one exit for information in the Dynamo

It was not convenient and long to use standard nodes, so it was decided to write a Python script. The complexity of the process of writing a script in python directly depends on your skills. Understanding how lists, conditions, and loops work. Writing this script is no more difficult than creating a script from nodes in Grasshopper. If you do not understand how the Python language works, it will still take time to study it.

If we compare my time for creating a script in both programs, it is about the same, while the time spent on learning Python for this script took much more than the time spent on learning the nodes for the script in the Grasshopper.

If we compare the base node functionality of the nodes, then Grasshopper is much ahead of Dynamo.

When creating a new surface, the capabilities of both programs are approximately at the same level. When choosing surfaces from BIM software, in Dynamo it is easier to do this than in Grasshopper with Tekla. Revit can be used to import any element, and from Tekla only wireframe elements of lines and points, you can see it in figure 70.

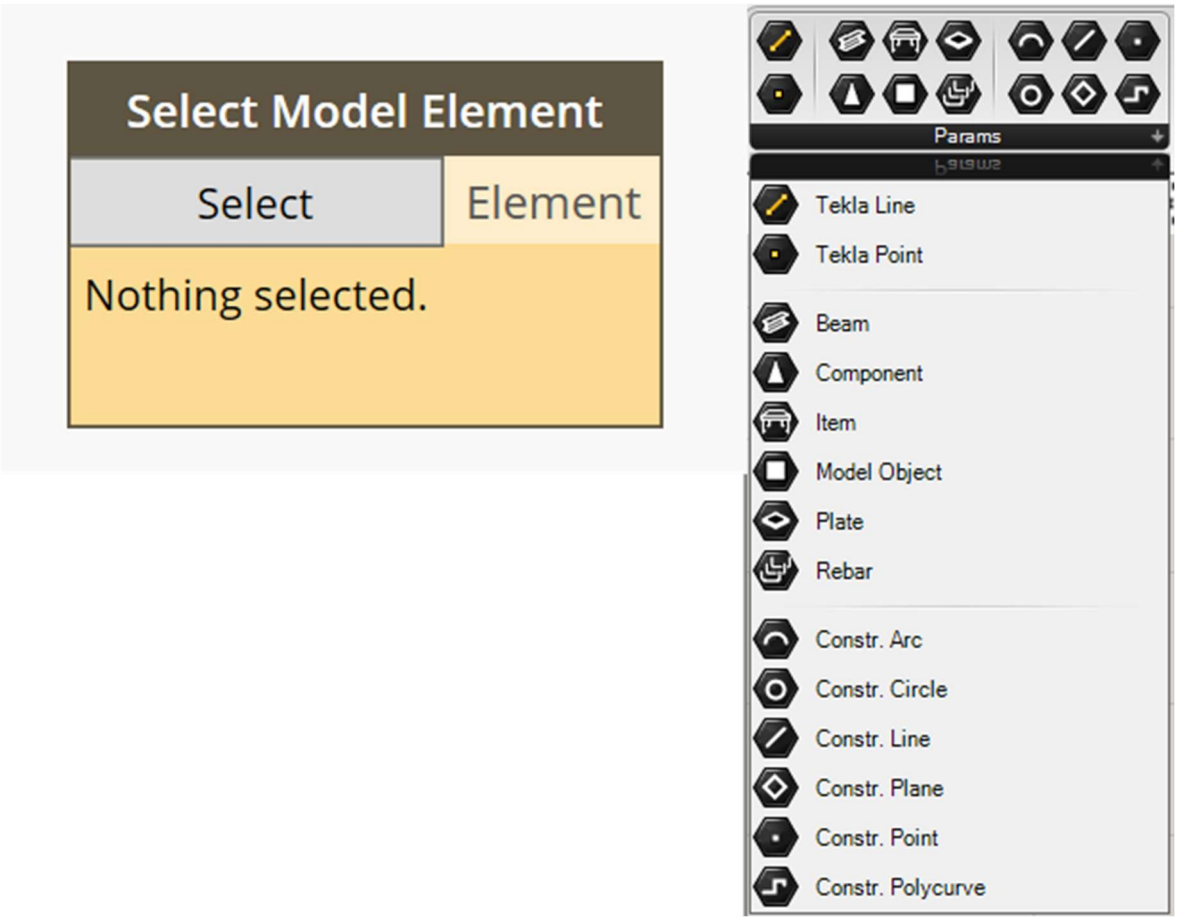


Figure 70. On the left is a node for selecting any element in Dynamo, on the right is a list of nodes for selecting an element in a Grasshopper

In the end, I would like to touch on the topic not at all as Parametric modeling, how much visualization of the results, and specifically the connection of nodes of structural elements.

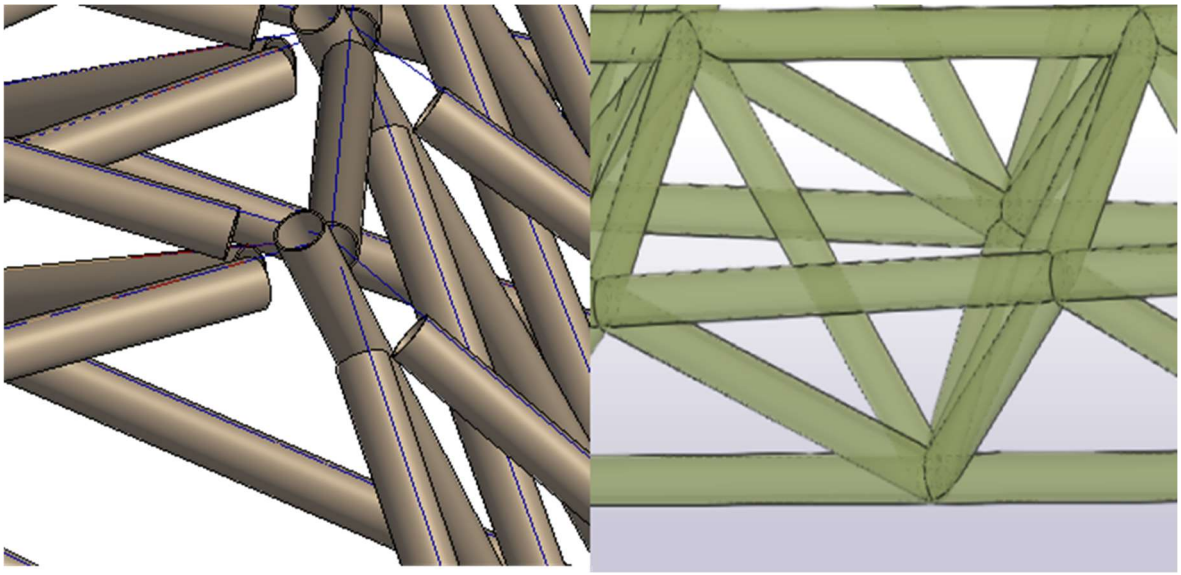


Figure 71. On the left is a node for connecting steel elements created in Dynamo, on the right is a node for connecting steel elements created in a Grasshopper

As you can see from the picture above, Revit does not connect metal objects at once and creates connection bindings by himself. Sometimes it does not come out quite correctly as shown in the picture. It is possible to fix these bindings, but manually for each node, which with many elements is a very laborious task.

4.3 Parametric portal frame

The construction of the geometry of lines in both Grasshopper and Dynamo is a very similar process. It differs only in the regulation of lists of data and small details of individual nodes. In both cases, the process is straightforward.

This time a Python script was used in both scripts. The script itself is 85% identical in both cases in the exclusion of regulation by the lists and syntax features of Dynamo and Grasshopper. It is worth noting the presence of a standard debugger panel in the Grasshopper, you can see it in figure 72, and its absence in Dynamo. Its presence makes it somewhat more convenient to track errors in the script.

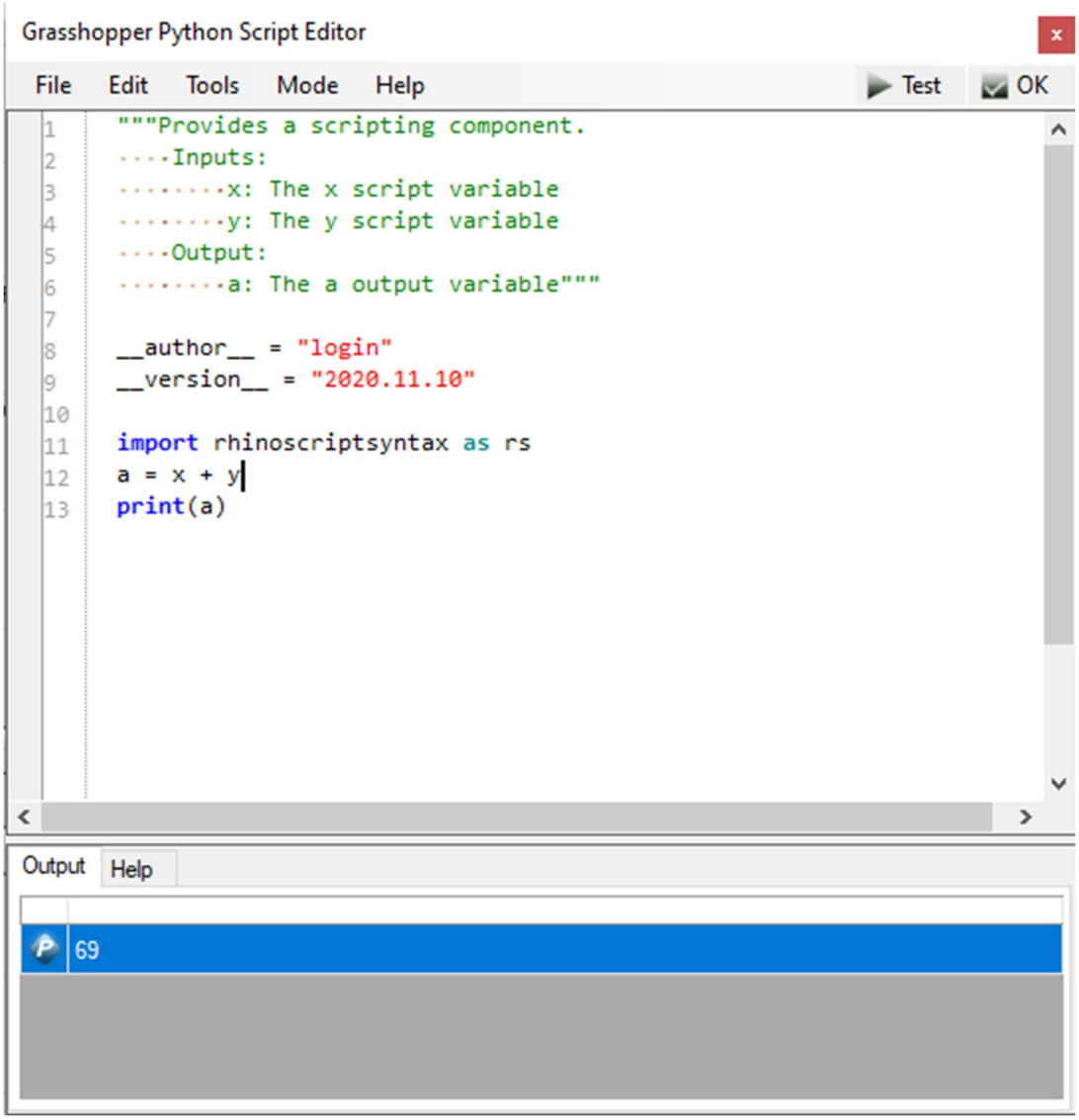


Figure 72. Python script editor with debugger panel in Grasshopper

The main differences begin at the stage of export to BIM software, you can see it in figure 73. First, these differences concern not so much Grasshopper and Dynamo as Revit and Tekla, but specifically their nodes and properties.

All elements created in the Revit model are Families. Accordingly, to place an element, you need to either download it or create it from 0. In Tekla, you need to open the profile library and select from the existing one or replace the values in the form formula with the desired ones

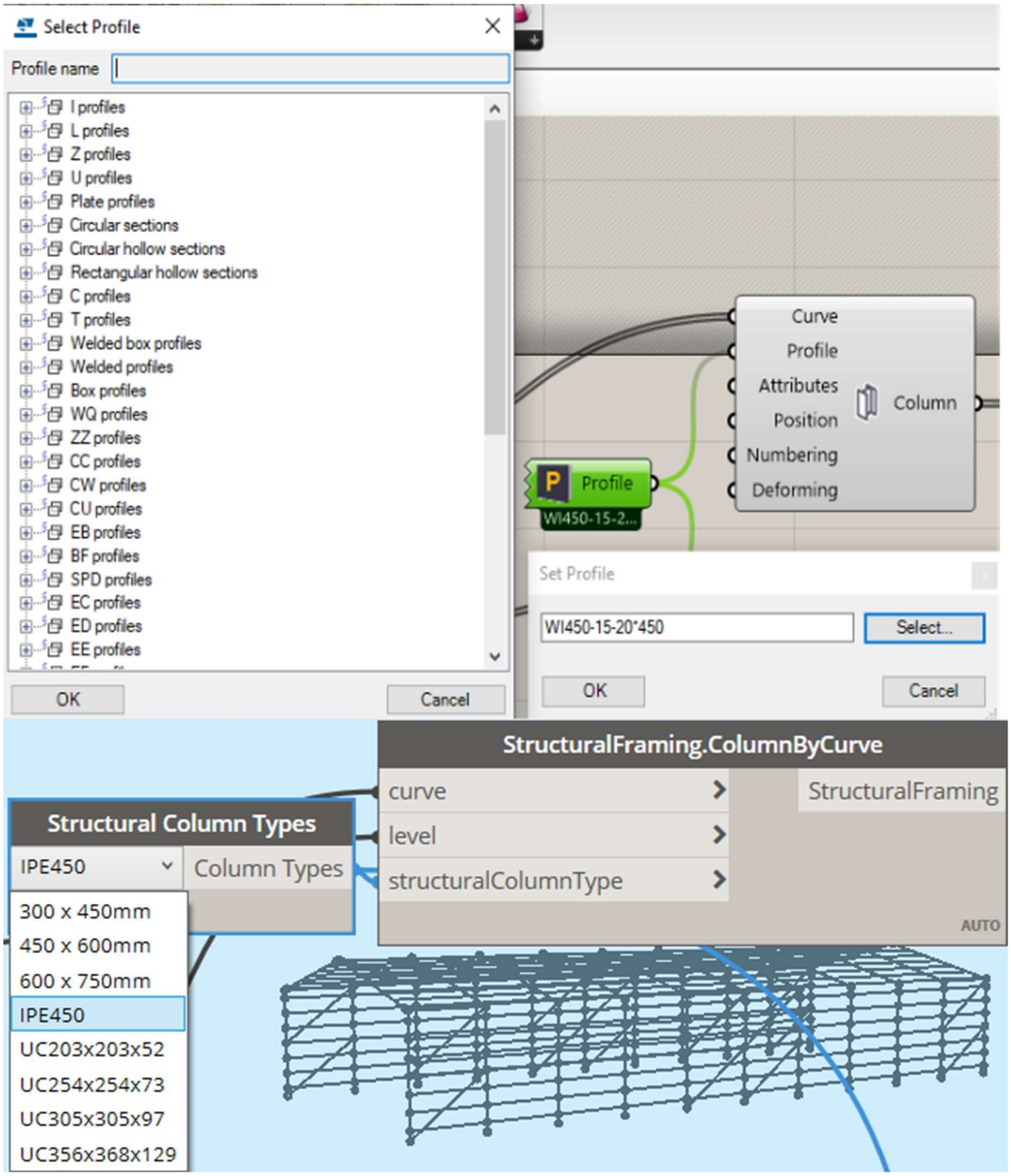


Figure 73. Above, selection of element sections in the grasshopper, below, selection of element sections in Dynamo

Thanks to the "Position" node for Tekla, the task of geometric properties is also a more convenient way than Dynamo, where you need to specify the name of the property

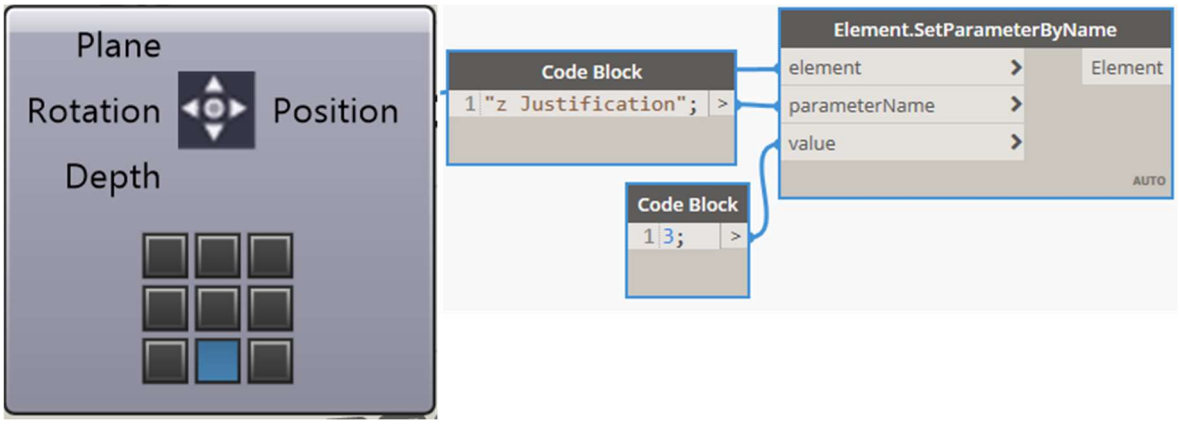


Figure 74. On the left is the node for editing the position of the element parameters in the Grasshopper, on the right is the node for editing the element parameters created in Dynamo

All this makes exporting the same geometry to Tekla much easier and faster than to Revit.

Touching upon the topic of steel connections, in both cases, this option is not ideal, but if you look at the picture with the generation of connections in general, then the situation in Dynamo is better.

To create steel connections in Dynamo, an additional pact of nodes "Autodesk Steel Connections 2020" is used, for creation in a grasshopper, a component node is used.

The algorithm for creating steel connections in Dynamo is much more complicated than in Grasshopper, but this is leveled by the fact that the node package already contains most of the blanks for various connections for the Dynamo player, and in this case, you just need to select the frame type of beam or column and the type of connection and the script will do the rest automatically ...

In the case of Grasshopper and Tekla, everything is simpler and more complicated. To produce a connection, only the desired type of beams and connections must be selected. But at the same time, some connections may not be built at all (This is a shortcoming of the script and may be corrected in the future).

4.4 Structural Analysis

In this case, the functionality of Karamba3D and the Robot of the node package for Dynamo is compared. If we speak for this model, then the functionality of both software copes well with the task.

Like Robot and Kramba3D, they have large libraries of sections, materials, capabilities, and their creation. It is possible to highlight the possibility of creating parametric sections and materials in Karamba3D in the Robot node package, this possibility is absent.

It is also worth highlighting the issue of optimization and errors, in Karamba3d there are almost no problems with this. And in the Robot package, all sorts of errors occur periodically during the analysis and creation of attributes. Errors are both due to an error in the script itself, as well as due to imperfections in the package.

5 Guidelines

For a better understanding and assistance to other students interested in parametric modeling, within the framework of this thesis, training materials were prepared and made for each of the 2 software.

They are 5 videos for each software, you can see it in figure 74, where 1 is an introduction and general information on how to download, install, log in, and where you can find useful basic training materials. The remaining 4 videos are dedicated to the models that were specifically considered in this thesis.

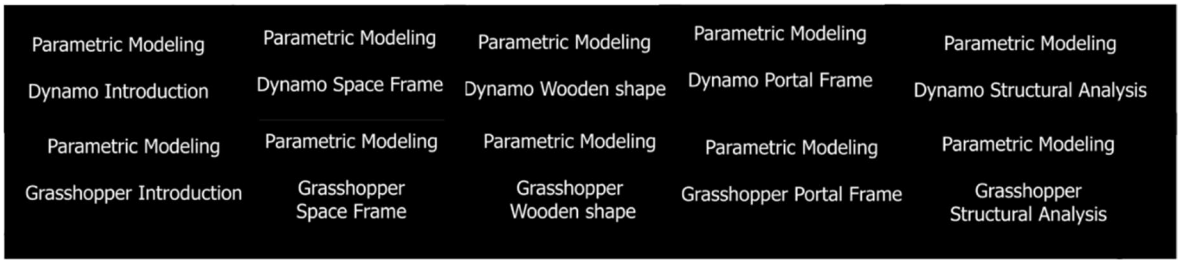


Figure 74. Titles for each video

In addition to video clips, all script files created in the framework of this program are also available. It is important to note that the files for the dynamo can be opened in the Revit version of the dynamo not lower than Revit 21.

6 Conclusions

6.1 Comparison of software

For convenient visualization, let us summarize the analysis in a table:

Table of model results.	
Dynamo	Grasshopper
1. Work with difficult shapes (Wooden Arc)	
To create an element in Revit, you need to set the family. The process is slower. Crashes may occur when changing parameters in automatic mode. You can place families directly and change geometry parameters, while the geometry will keep the position in the BIM model.	You can immediately export the created geometry to Tekla. The process is faster. The model changes almost synchronously with the change in parameters. The position of the exported geometry directly depends on the parameters set in Grasshopper, if you change them in Tekla, further editing in Grasshopper will lead to their cancellation or will not affect them at all.
2. Work with difficult shapes (Wooden Arc)	
Opportunities to unload Data such as a plane or surface from the BIM software (Revit) are more than in Grasshopper. The functionality of standard nodes did not allow you to create a structure quickly and conveniently, a Python script was used. The visualization of the connections of structural elements is not always accurate and convenient immediately after creation.	Opportunities to unload Data such as a plane or surface from BIM software (Tekla) are less than in Dynamo. The functionality of standard mods allows you to quickly create the desired structure on the surface. The visualization is complete, that is, the structure goes to the end of the line even if the elements intersect.
3. Portal Frame	
Geometry is created in the same way in both software.	Geometry is created in the same way in both software.

The script debugger is less convenient than Grasshopper. The elements of the BIM structure that you need to work with must be loaded or created in Revit in advance. Editing geometry parameters is done by name, it takes some time. It takes more time to create a BIM model. Steel connections are created quickly with the Autodesk Steel Connections 2020 add-on node package.	The script debugger is more convenient than in Dynamo. Can immediately show information and errors in a special field below. Items can be loaded and edited immediately from the catalog. Thanks to special nodes, editing geometry parameters is quite fast. It takes less time to create a BIM model. Making connections between steel structures is quick and easy, but some errors and inaccuracies prevent some type of connections.
4. Structural Analysis	
The node package does not always work correctly, and errors may occur. In Dynamo, you can only unload materials, sections, or supports from a Robot. The ability to create elements in Dynamo is available and, in the future, continue to work only in Robot.	The node package works almost autonomously and without errors. The ability to create your parametric subtypes of sections and materials is available. Elements created in Grasshopper are not amenable to direct editing in the future using Rhinoceros.

Table 1. Table of model results.

In the end, you can summarize all the results in one table.

Summary table of results	
Dynamo	Grasshopper
1. Speed of modeling and data processing.	

The data processing speed is low. In automatic mode, there are frequent freezes and departures. The purpose of the nodes by name is not always clear the first time. Searching by name is not always intuitive.	When modeling, automatic mode is mainly used, there are rarely freezes and crashes. It is easy to search for nodes, the purpose of the nodes is not always intuitive.
2. Availability of guidelines and functional plugins.	
The base is smaller but still large. The developer is one because of this, it is easier to find guides when working between different software.	The software is much older, the community base is huge, there are many guides. When working between different software, it is more difficult to get guides or help due to different developers
3. The functionality of the software.	
Specifically, full parametric modeling, the functionality of standard nodes is below. When working specifically with Revit, a large functionality opens for editing and selecting various parameters of an already finished model. There is a Dynamo player that allows even a beginner to use ready-made scripts without starting Dynamo	The functionality of the nodes is great and convenient, just like Dynamo, there is Python, but the C # and VB script are also added to it. The functional is designed directly for modeling and creating geometry. The functional for the regulation of parameters is designed mainly for elements. Tekla's visibility or annotation properties cannot be adjusted.
4. Software availability and connection methods.	
Dynamo goes along with Revit. There is also a free version where you can create only geometry and download node packages. The price of Revit is higher than that of Rhinoceros, but you do not need to take additional software for BIM.	Rhinoceros goes with the Grasshopper. There is a trial version for 90 days with full functionality. Cheaper than Revit, but in addition to this, you need to additionally take BIM software.

Table 1. Table of model results.

Based on the above results, it is easy to see that the Dynamo functional in many aspects is an order of magnitude inferior to Grasshopper. The power of the Grasshopper engine allows for much faster processing of complex geometry elements, and the convenience and ability of the nodes are also superior to Dynamo.

All this allows you to work much faster and more conveniently.

But here it is also worth noting another equally important aspect, what software is used for BIM modeling.

If this is Tekla, then there is only one option left to use the Grasshopper.

For Revit, both Grasshopper (not in ideal work) and Dynamo are available. At the same time, Dynamo comes with Revit. That is, it is important to understand for what purposes parametric modeling will be used, and whether the advantages of Grasshopper equalize its purchase if it is possible to create the same in Dynamo for Revit.

According to the results of this Thesis, it can also be seen that the result is ultimately at about the same level, everywhere scripts were obtained to accomplish certain goals. The only difference lies in the complexity.

6.2 Parametric modeling

During this thesis, various scripts were written and a lot of material on parametric modeling was studied.

The results show that parametric modeling will soon be an integral part of any project, not necessarily in the form of Grasshopper or Dynamo. Creating tasks for a computer, i.e. scripts, greatly simplifies the typical design moments, what a person does in an hour a computer will do in 2 seconds.

In structural design, this will allow you to quickly create a heterogeneous structure of a building, carry out analysis by quickly changing the parameters of an element, create scripts where, according to the conditions and surface, the computer itself will create the most suitable structural solution.

But the main thing is to understand that Parametric modeling is not a solution to all problems and now it is impossible to get away from the direct design. Machine learning does not stand still, and perhaps in the future it will be enough to enter the parameters of the building and it will be designed. But today the creation and optimization of the script take a lot of time and effort. And it is important to understand the specific task of the script, the time that will

be spent on its creation, and most importantly, whether the goal is worth the time spent. In some cases, using straightforward modeling will be much more cost-effective.

7 Sources.

Nancy Reyas.2018. Top 5 Benefits of BIM Construction. https://hmcarchitects.com/news/top-5-benefits-of-bim-construction-2020-05-13/

Wikipedia. Parametric design. 2015. https://en.wikipedia.org/wiki/Parametric_design#:~:text=Parametric%20design%20is%20a%20process,design%20intent%20and%20design%20response.

Wikipedia. Grasshopper3D. 2020. https://en.wikipedia.org/wiki/Grasshopper_3D

Wikipedia. Tekla. 2020. https://en.wikipedia.org/wiki/Tekla

Wikipedia. Revit. 2020. https://ru.wikipedia.org/wiki/Revit

Karamba3D. Main page description of software.2020. https://www.karamba3d.com/

Rob Deutscher. 2013. Galaxy SOHO. https://commons.wikimedia.org/wiki/File:Galaxy_Soho.jpg

Autodesk. Introductory Tutorials for Dynamo. https://dynamobim.org/learn/

Autodesk. The Dynamo Primer. https://primer.dynamobim.org/

Structured Parametric. Parametric Tutorials. https://www.structuredparametrics.com/tutorials

Trimble. Grasshopper-Tekla Live Link. https://teklastructures.support.tekla.com/not-version-specific/en/ext_grasshopperteklalink

David Rutten. Grasshopper getting started. https://vimeopro.com/rhino/grasshopper-getting-started-by-david-rutten

Tomasz Fudala.2019. Portal Frame with Structural Analysis for Dynamo Package – Part 02. https://blogs.autodesk.com/revit/2019/10/28/portal-frame-with-structural-analysis-for-dynamo-package-part-02/

Autodesk.Robot and dynamo connection. Dynamo Forum. https://forum.dynamobim.com/t/robot-and-dynamo-connection/56116

Emmanuel Weyermann. Optimizing Structural Analysis with Dynamo. https://www.autodesk.com/autodesk-university/class/Optimizing-Structural-Analysis-Dynamo-2015#handout

Tadeh Hakopian. Exploring Python Nodes in Dynamo. https://www.autodesk.com/autodesk-university/class/Exploring-Python-Nodes-Dynamo-2019#handout

Scott Davidson. 2019. Your First Python Script in. https://developer.rhino3d.com/guides/rhinopython/your-first-python-script-in-grasshopper/

Karamba3D examples. https://www.karamba3d.com/examples/

Rhino Grasshopper youtube channel. https://www.youtube.com/channel/UCjLDKM9EzNdASaNdjBhTqug