|  |
| --- |
| Parametric Modelling |
| Frame Bridge Model  Date XX. August 2023  Author(s) Sophia Kuhn |
|  |
|  |

**Table of** **Contents**

[Summary 1](#_Toc143521531)

[1 Usage Instructions 2](#_Toc143521532)

[1.1 Data Sampling 2](#_Toc143521533)

[1.2 Geometry generation 2](#_Toc143521534)

[1.3 Structural Analysis 2](#_Toc143521535)

[1.4 Results Postprocessing 3](#_Toc143521536)

[2 Parameter 4](#_Toc143521537)

[2.1 Slab Parameter 4](#_Toc143521538)

[2.2 Wall Parameter 6](#_Toc143521539)

[2.3 Parapet Parameter 7](#_Toc143521540)

[2.4 Structural Model Parameter 8](#_Toc143521541)

[2.5 Material Parameter 8](#_Toc143521542)

[2.6 Plate Section Parameter 9](#_Toc143521543)

[3 Sampling 11](#_Toc143521544)

[3.1 Sampling Algorithm 11](#_Toc143521545)

[3.2 Sampling Implementation 11](#_Toc143521546)

[4 Parametric Grasshopper File 12](#_Toc143521547)

[4.1 Structural Model 12](#_Toc143521548)

[4.1.1 Definition of Modelling Origin 12](#_Toc143521549)

[4.1.2 Plate 12](#_Toc143521550)

[4.1.3 Walls 16](#_Toc143521551)

[4.1.4 Parapet 17](#_Toc143521552)

[4.1.5 Modelling options 18](#_Toc143521553)

[4.1.6 Meshing 21](#_Toc143521554)

[4.1.7 Support Conditions 23](#_Toc143521555)

[4.2 Rhino Model/ File Generation 24](#_Toc143521556)

[4.2.1 Layer Generation 24](#_Toc143521557)

[4.2.2 Document User Text 24](#_Toc143521558)

[4.2.3 File Saving 25](#_Toc143521559)

[4.3 Dependencies or Restrictions 25](#_Toc143521560)

[5 Evaluation Framework 27](#_Toc143521561)

[5.1 Iron python script: Build, Evaluate and Visualise Structure 27](#_Toc143521562)

[5.1.1 Define Structure 27](#_Toc143521563)

[5.1.2 Analyse 27](#_Toc143521564)

[5.1.3 Extract, save and Visualise Results 27](#_Toc143521565)

[5.2 Analysis Results 27](#_Toc143521566)

[6 Iteration Loop for Data Genertaion 30](#_Toc143521567)

[7 Appendix A – 31](#_Toc143521568)

[7.1 …. 31](#_Toc143521569)

Summary

This is a dynamic documentation of the parametric Non-Linear Finite Element Analysis Pipeline for Concrete Frame Bridges.

# Installation

## Installation Instructions

1. Installation of Rhino 7: [Rhino - Downloads (rhino3d.com)](https://www.rhino3d.com/download/)

Rhino is a CAD software, which is used for visualization, drawings and parametrizes geometry. Make sure it is Rhino 7 and not the newest Version Rhino 8. (Unfortunately, the free 90 Day test version only exists for Rhino 8). At IBK we have licenses that we will request for you with which you can use Rhino 7. Please install the English version of Rhino (the language can be selected on the download webpage).

(If you want to use the Grasshopper generation file, then you also need to install the grasshopper plug-in ANEMONE. You can download the plugin-file [here](https://www.food4rhino.com/en/app/anemone) and the instruction how to install it are [here](https://www.food4rhino.com/en/faq#users-install-grasshopper-plugin))

1. Install Ansys 2024: <https://itshop.ethz.ch/>

For DBAUG employees: You can download the Ansys Research 2024 version via the IT Shop of ETH.

For DBAUG students: You can download the Ansys Teaching 2024 version via the IT Shop of ETH.

(I am currently using Ansys Research 2022. However, I will also change to Research 2024 and change the source code so it also works with this version. Status: In progress)

1. Install Anaconda: [Free Download | Anaconda](https://www.anaconda.com/download)

This is a very convenient way of installing Python on your computer. Personal Tip: Make sure you don’t have any other python version on your device. This can sometimes lead to complications.

1. Install GitHub Desktop: <https://docs.github.com/en/desktop/installing-and-authenticating-to-github-desktop/installing-github-desktop>

This is a program to push and pull to GitHub repositories. You can also use equivalent programs or do it directly in the terminal. Login with your GitHub account.

1. Clone repository: <https://github.com/SophiaKuhn/StructuralEvaluationOfCFB>

Via GitHub Desktop you can clone our repository. Practise for working on the repository: Work always on a branch and not on main. Merge you changes from your branch to main only when your merge request got accepted. This helps to keep an overview over the changes within the repository and also allows working on the repository parallelly.

1. Install StrucEngLib: <https://strucenglib.ethz.ch/strucenglib_plugin/install_for_ansys/>

This is a Package for Rhino that was developed at the Chair of Concrete Structures and Bridge Design (by Dr. Marius Weber). Follow the installation instructions “Install for ansys” via the link.

( In case you need to make changes to the source code of the dependencies such as StrucEngLib or compas\_fea you should follow the developer installation instruction for the StrucEngLib [here](https://strucenglib.ethz.ch/strucenglib_plugin/developer/python_developer_mode/). This does not work for compas, so do it for all other 3 dependencies. You will probably also need to downgrade the scipy version to 1.11.4 to resolve an error regarding compass and scipy when running a script)

1. Install CMMUsermat: <https://github.com/kfmResearch-NumericsTeam/CMM_Usermat/wiki/01-Getting-Started>

This enables the material Non-Linear FE-Analysis with Ansys. It was developed by Dr. Karel Thoma. Follow the instructions provided in the link.

1. Install further dependencies to your strucEngLib python environment

Open the anaconda command prompt. Activate the strucenglib3 environment, that was automatically created when installing the StrucEngLib plugin. Then install the following packages:

pip install ipykernel

pip install scikit-optimize

pip install pandas

## Testing your setup

To test you set up you can test run the examples provided in the “\_testSetup” folder of the repository.

**Geometry Generation**

Simply open the Grasshopper file and follow the instructions in the file.

**NLFE\_Analysis**

You can test if you correctly set up the software dependencies by running the 2 provided examples. To run these examples, open the Rhino file. Enter the command “EditPythonFile” and navigate and open to the corresponding .py file. Then run the examples.

Hint: Check the defined path. Either change it to where you want the Ansys Calculations to be saved. Or make sure that the defined path exists.

# Pipeline Overview

We have developed a parametric pipeline that automatically performs Non-Linear Finite-Element Analysis (NLFE-Analysis) for simple concrete frame bridges (CFBs).

The pipeline takes bridge feature vectors X as input and outputs the structural verification results in a vector Y.

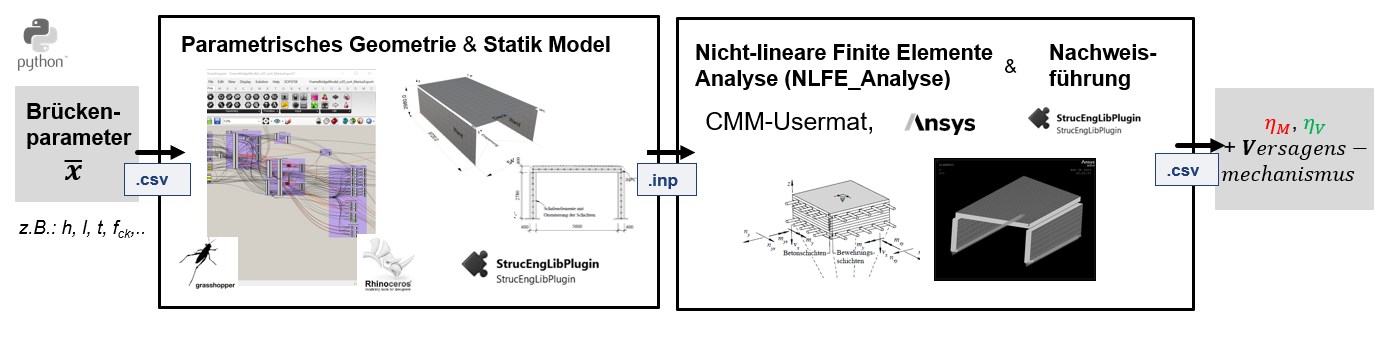


Figure 1: Parametric Pipeline Overview

The pipeline consists of 4 separate parts, which are run separately.

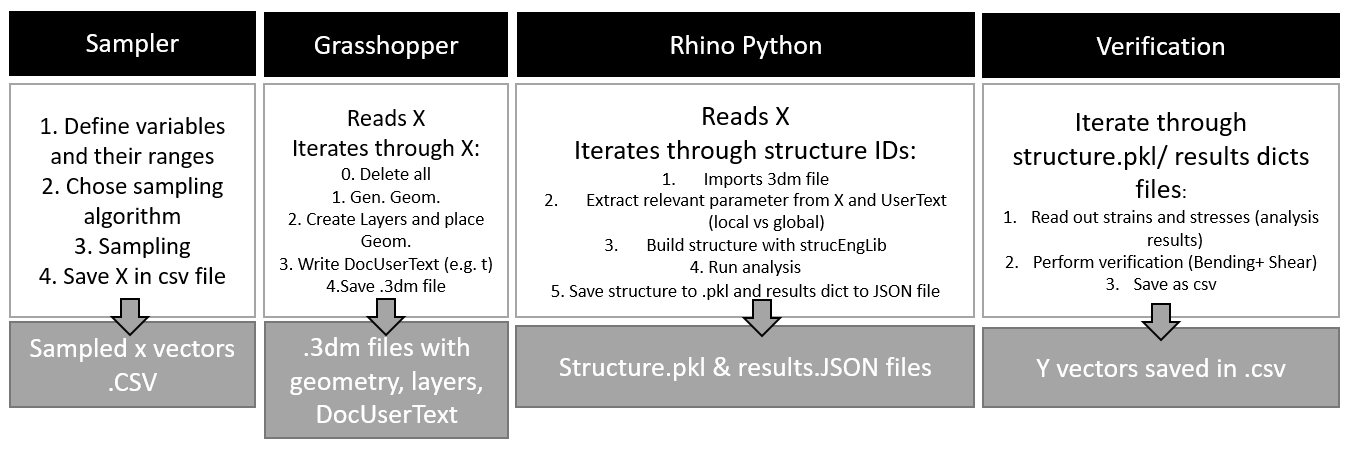


Figure 2: The 4 individual components and their summarized functionality of the parametric NLFA-Pipeline

# Instructions on using the pipeline

## Sampler:

* Open Jupiter notebook: CFB\_sampling.ipynb
* Select sampling strategy and run corresponding notebook cells

## Geometry generation

* Open file: “CFB\_geometryGeneration.gh”
* Follow the green text instructions.
* (Hint: You can use this file independently by connecting the parameter sliders to the parameter group)

## Structural Analysis

1. Open an empty Rhino File
2. Command: “EditPythonScript”
3. Navigate to the directory and open file: CFB\_NLFEAnalysis.py
4. Input: Sample Index: idx\_s, Start and end ID
5. Run scripts

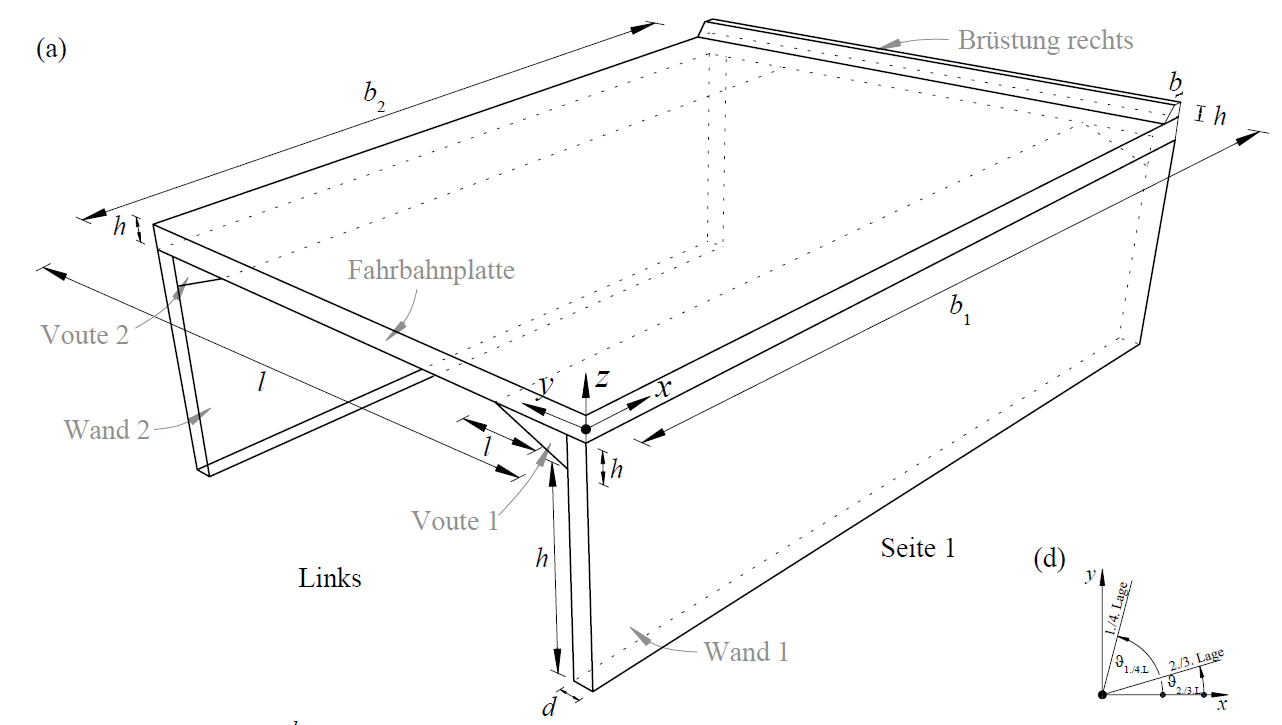
## Results Postprocessing

* Open Jupyter notebook: CFB\_Verification.ipynb & Run

# Parameter

In this section the parameter of the parametric model are listed and defined. The parameter themselves and their corresponding utilized ranges/ categories are derived from a pre-study of existing Bridges. This section defines all parameters that are variables of the parametric model and therefore sampled by the sampler.

In the Geometric Model in Grasshopper only the plate, wall, and subset of the structural modelling parameters are needed and therefore imported. The remaining parameter are sampled and then directly read by the python script within Rhino.



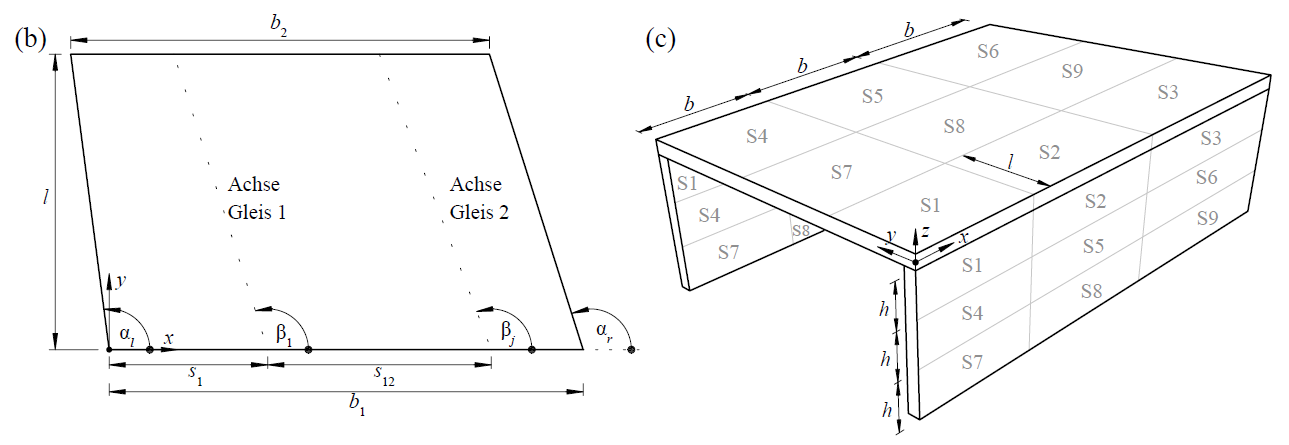


Figure 3: Parameter Overview (Result from WP 1.2, taken from the Zwischenbericht by Marius Weber)

## Slab Parameter

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Width 1 | Legth of Wall 1 | b1 | mm |
| Width 2 | Legth of Wall 2 | b2 | mm |
| Span | Span of plate between the two walls | L | mm |
| Angle | Angle between Wall Axis and Deckplate edge. | alpha\_l | Deg. |
| Section 1 Width (Side 1) | Proportion of width b1 that falls into section 1 | b\_sec1\_b1 | - |
| Section 2 Width (Side 1) | Proportion of width b1 that falls into plate section 2 | b\_sec2\_b1 | - |
| Section 1 Width (Side 2) | Proportion of width b2 that falls into plate section 1 | b\_sec1\_b2 | - |
| Section 2 Width (Side 2) | Proportion of width b2 that falls into plate section 2 | b\_sec2\_b2 | - |
| Outer Section Length | Proportion of span that fall into the two outer plate sections (1&2) | l\_sec | - |
| Thickness of deck slab plate |  | h\_FP | mm |



Figure 4: Plan View of Plate

Discarded plate parameter:

* Alpha\_r: Results from total widths and alpha\_l
* Width of Section 3: Results from total width (b1,b2 and section widths)

## Wall Parameter

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Wall Thickness | Wall Thickness (constant over whole hight) | t\_w | mm |
| Wall Hight | Hight of Wall from top of fundament to lower point of deck slab plate (or lower point of voute if existing) | h\_w | mm |
| Hight of Sections 1-3 | Proportion of the wall hight which falls into wall section 1 to 3 | h\_S1-3 | - |
| Hight of Sections 7-9 | Proportion of the total wall hight which falls into wall section 7-9 | h\_S7-9 | - |



Figure 5: Wall 1



Figure 6: Cross Section View

Discarded Parameter:

* Hight of Section 4-6: results from total hight minus the 2 other section hights

## Structural Model Parameter

Structural modelling parameter that are used in the Grasshopper file:

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Mesh size factor | Controlls size of generated mesh (factor is multiplied with minimum plate thickness to get maximum mesh size) | mesh\_size\_factor | - |

## Material Parameter

**Concrete Material Properties**

The concrete material properties are the same for the whole structure. So no different concrete is defined for the deck slab and the individual walls.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Unit Weight of Reinforced concrete | Set constant to 0.0000025 | roh | N/mm3 |
| Compressive Stregth of Concrete | Zylinder/ Cube? | fcc | MPa |
| Ultimate compressive strain in the concrete | Claulated according to EN 1992-1-1:2004 (Table 3.1)- | ecu | - |

Calculation of ecu from fcc according to EN 1992-1-1:2004 (Table 3.1):



**Reinforcement Steel Properties**

The reinforcement material properties are the same for the whole structure. So no different reinforcement material properties are defined for the deck slab and the individual walls. We assume a idealised biaxial stress-strain behaviour for the steel.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| Yield stregth of reinforcement steel |  | fsy | MPa | 390 |
| Ultimate strength of reinforcement steel | This parameter is sampled with a factor. Which is sampled and then afterwards it is multiplied with fsy. This allows that the ultimate strehth always lies above the yield strength. This factor is set to 1.08 at the moment. | fsu | MPa | (1.08) \*fsy |
| Ultimate strain of reinforcement steel | We assume a constant ultimate strain of 0.05 (as all the used reinforcement fullfill this) | esu | - | 0.045 |

## Plate Section Parameter

These are the parameters defining the geometry of the individual plate cross sections.

**General Plate section parameter:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Top concrete cover thickness | The concrete cover is set to a constant value of 30. | oo | mm |
| Bottom concrete cover thickness | The concrete cover is set to a constant value of 30. | uu | mm |

**Of deck slab plate sections:**

The reinforcement numbering is in the direction of the local z-axis. The first layer is the bottom and 4 the top layer in the deck slab plate.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Rebar diameter of the nth reinforcement layer in the deck slab plate |  | dn\_plate | mm |
| Spacing of reinforcement in deck slab plate |  | s\_plate | mm |

**Of Wall sections:**

The reinforcement numbering is in the direction of the local z-axis. The first layer is the inner layer and 4th the outter layer in the deck slab plate.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** |
| Rebar diameter of the nth reinforcement layer in the wall plates |  | dn\_walls | mm |
| Spacing of reinforcement in the wall plates |  | s\_walls | mm |

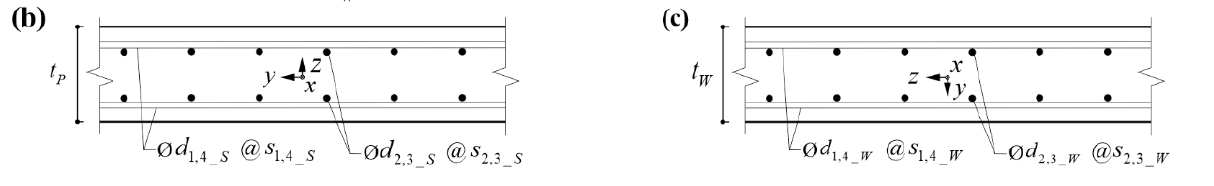


Figure 7: Sketch of plate sections of deck slab and walls.

The reinforcement content is calculated using the corresponding diameter and spacing value:

Reinforcement content of a plate (1m thickness):

as\_ **=** (diameter\*\*2 \*m.pi)**/**(4\*spacing) [mm2/1m]

## Load Parameter

### Permanent Loads

The dead load of the modelled elements is calculated using the defined unit weight of reinforced concrete (see 2.5).

**Superimposed dead load**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| Unit weight of seal | (Abdichtung) | gamma\_A | N/mm3 | 0.0000021 |
| Hight of seal | (Abdichtung) | h\_A | mm | 5 |
| Unit weight of gravel bed | (Schotterbett) | gamma\_G | N/mm3 | 0.0000018 |
| Hight of gravel bed | (Schotterbett) | h\_G | mm |  |

Calculation and generation of superimposed dead load:

**Earth pressure**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| Unit weight of backfill material | Hinterfüllungsmaterial | gamma\_E | N/mm3 | 0.0000020 |
|  |  | phi\_k | ° | 30 |
|  |  |  |  |  |
|  |  |  |  |  |

Note: While the earth pressure resulting from the backfill has a triangle shape (like shown in Fig. ), we apply the resulting earth pressure force uniformly over the wall of the structure.

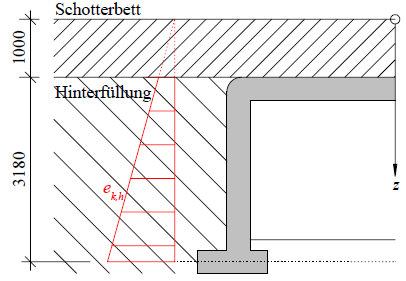


Figure 8: Earth pressure on structure

Calculation and generation of earth pressure:

### Variable Loads

**(Normalspurverkehr nach SIA)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| Gleisabstand | Abstand der Bahngleisachse vom linken Ende (x=0)  (sampled as a factor of b1) | s | mm | 0,1 |
| Bahngleiswinkel | Winkel zwischen y-Achse und Bahngleisachse | beta | ° | 0, -180 |
| Load of concrete sleeeper | (Betonschwelle) | q\_Gl | N/mm | 4.8 |
| Width of Concrete sleepers | (Betonschwelle) | b\_Bs | mm | 2500 |
| Hight of train tracks | Hight measured from the top of the deck slab to the bottom of the concrete sleepers (Betonschwelle) | h\_Strich | mm | Calculated from gravel bed hight and the hight of the seal layer, Betonschwellen hight?  h\_G+ h\_A (ca. h\_G) |
| Nominelle Achslast | Abhängig von der streckenklasse (SIA 269/1 11.2), für D4 = 225 kN | Q\_k | N | 225\*1000 |
|  | ????? | y\_A | mm | 5000 |

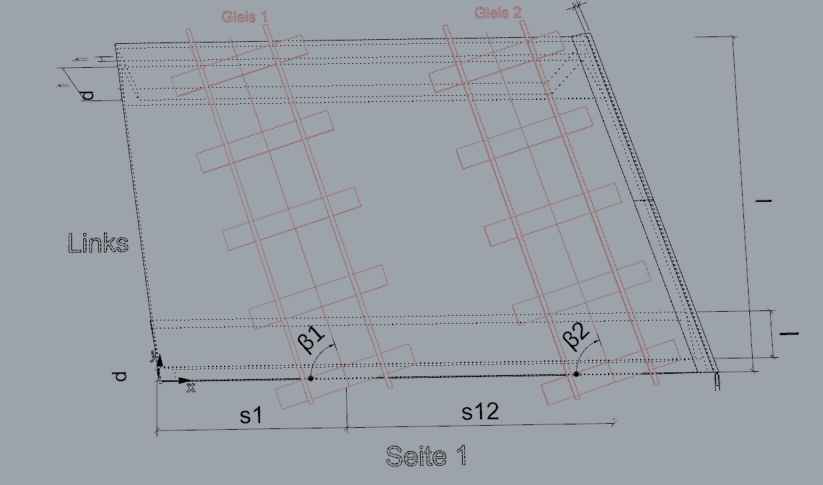


Figure 9: Definition of the parameters defining the train track position and directions

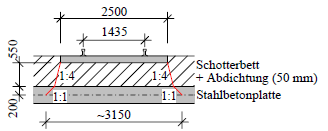
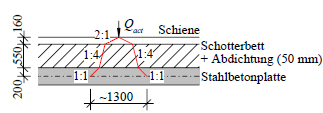
 

Figure 10: Lastausbreitung in Quer- und Längsrichtung

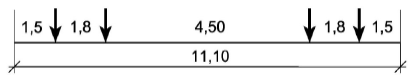


Figure 11:Aktualisiertes Lastmodell nach SIA 269/1

**Earth pressure resulting from variable loads**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| Gleisabstand | Abstand der Bahngleisachse vom linken Ende (x=0)  (sampled as a factor of b1) | s | mm | 0,1 |

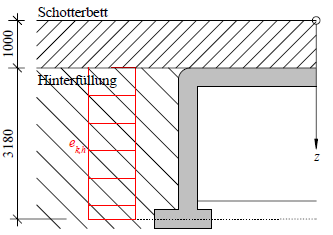
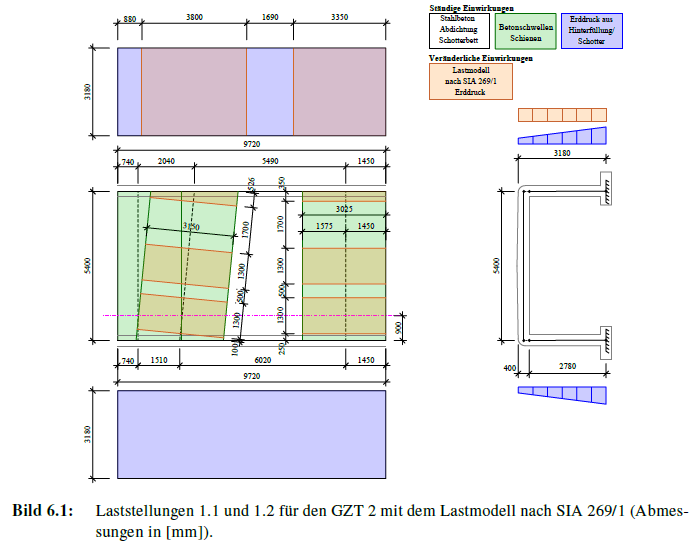


Figure 12: Earth pressure loading resulting from live loads

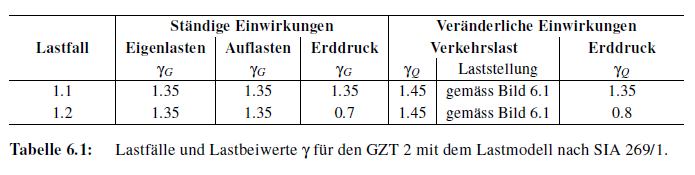
Calculation and generation of loading:

### Load Combination

The following Image shows how the above defined loads are applied to the structure.



Within this study we only consider GZT 2 with the loading model according to SIA 269/1 (like shown in the table). In this study we only look at that load case 1.1. We however do not check the load case 1.2 and GST 2 Engleisungsmodelle 1 &2.Additionally, we do not consider GZT 4 (Ermüdung).



Erdruck aus verändelrichen Lasten auch mt 1.45! (live load behandelt)

To discuss with Anne: Eigelanlast Loadfaktor 1.2 statt 1.35?

Further loading simplifications/ assumptions:

* Wind, Temperature and Snow loads are neglected.

# Sampling

## Sampling Algorithm

Currently the LHC sampling algorithm is used. In the future change to Sobol.

And possibly even active learning or adaptive sampling is needed.

## Sampling Implementation

The sampling is implemented in Python in a Jupyter Notebook.

The implementation is currently completely separate and not connected to other stages of the data generation loop.

In the notebook one has to define:

* idx\_s : Index of Sampling iteration (Defines which sampling iteration is currently executed, and is written into file name)
* n\_samples: number of samples (Defines how many samples should be sampled).

Running the notebook creates n samples of X vectors, which are sapled with the above defined sampling algorithm. These samples are then saved in a csv file.

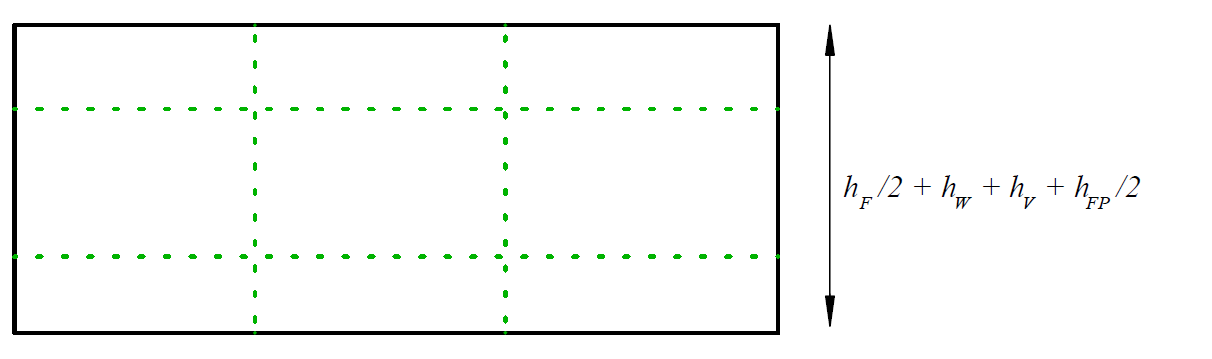
# Parametric Grasshopper File

## Structural Model

### Definition of Modelling Origin

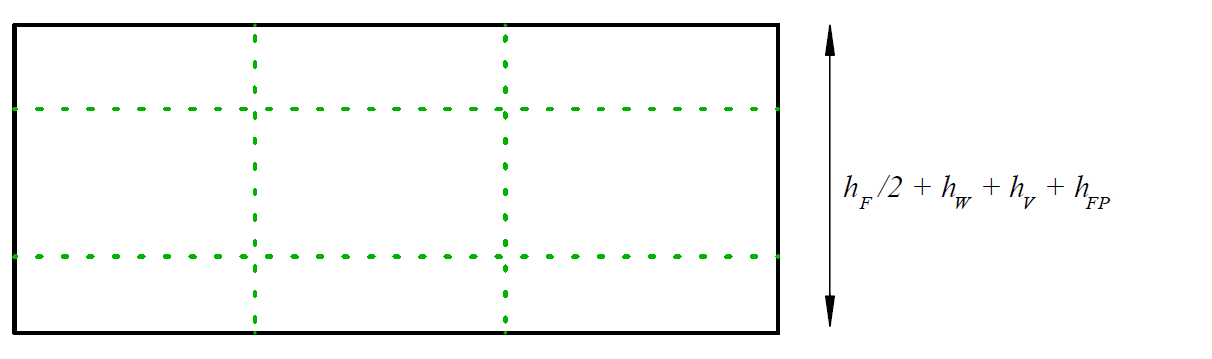
Without Offset modelling:

h\_w\_el**=**(h\_F/2) + h\_w + h\_v **+**(h\_FP/2)



With Offset modelling:

h\_w\_el**=**(h\_F/2) + h\_w + h\_v **+**(h\_FP)



Origin Definition: (0,0,h\_w\_el)

### Plate

The following figures show the parametric model is set up in grasshopper. The Figures define the utilized names and numbering system of the parametric model.





Figure 13: Plate model numbering

### Walls

The following figures show the parametric model is set up in grasshopper. The Figures define the utilized names and numbering system of the parametric model.

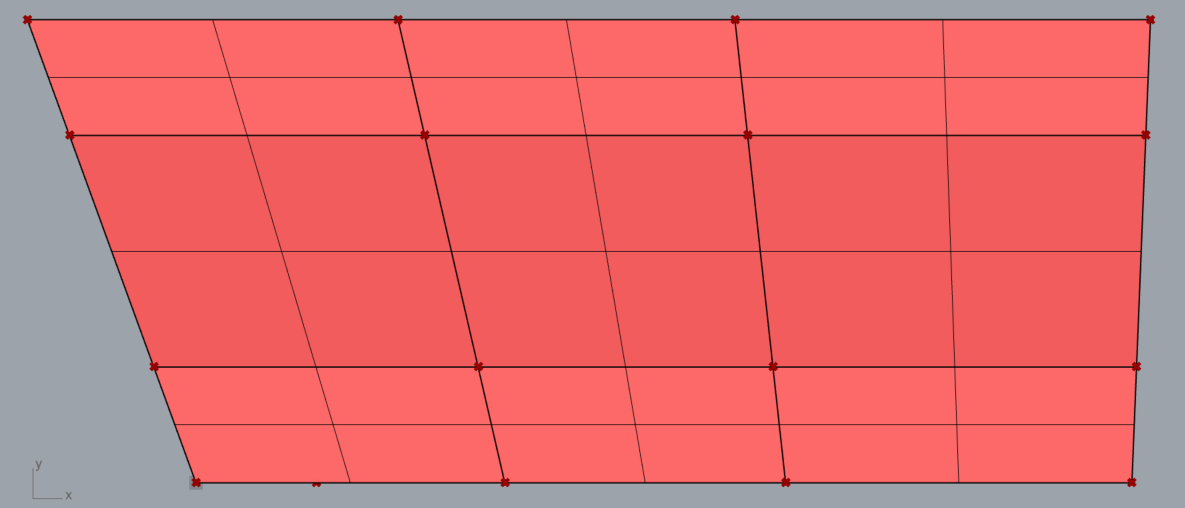


Wo die walls anfangen -> je nach offset modelling und no/ with mpcs

Auf basis welches h\_ws genau jetzt die Proportion berechnet wird von den Seiten sectionen

#### Voute

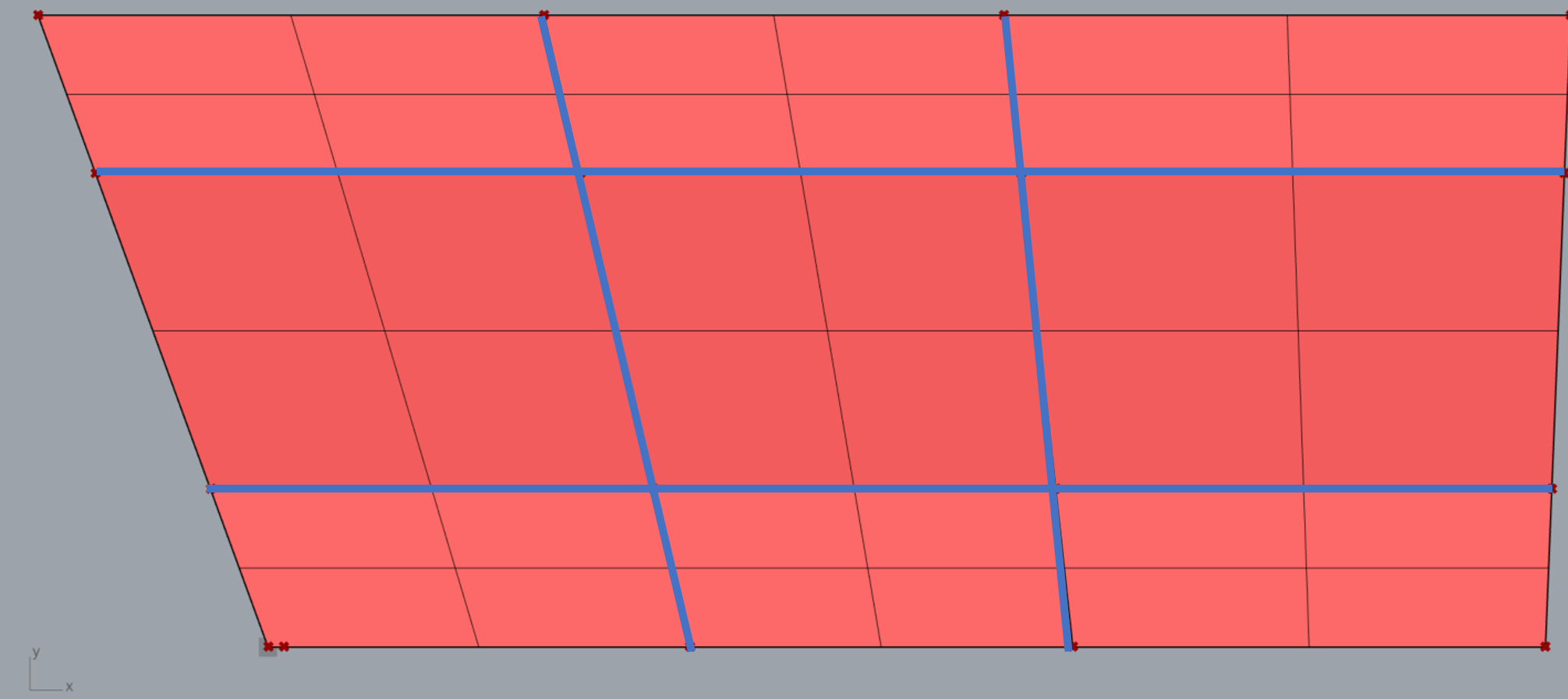
To model the voute and to calculate the corresponding plate thicknesses, it is distinguished between 4 different cases.



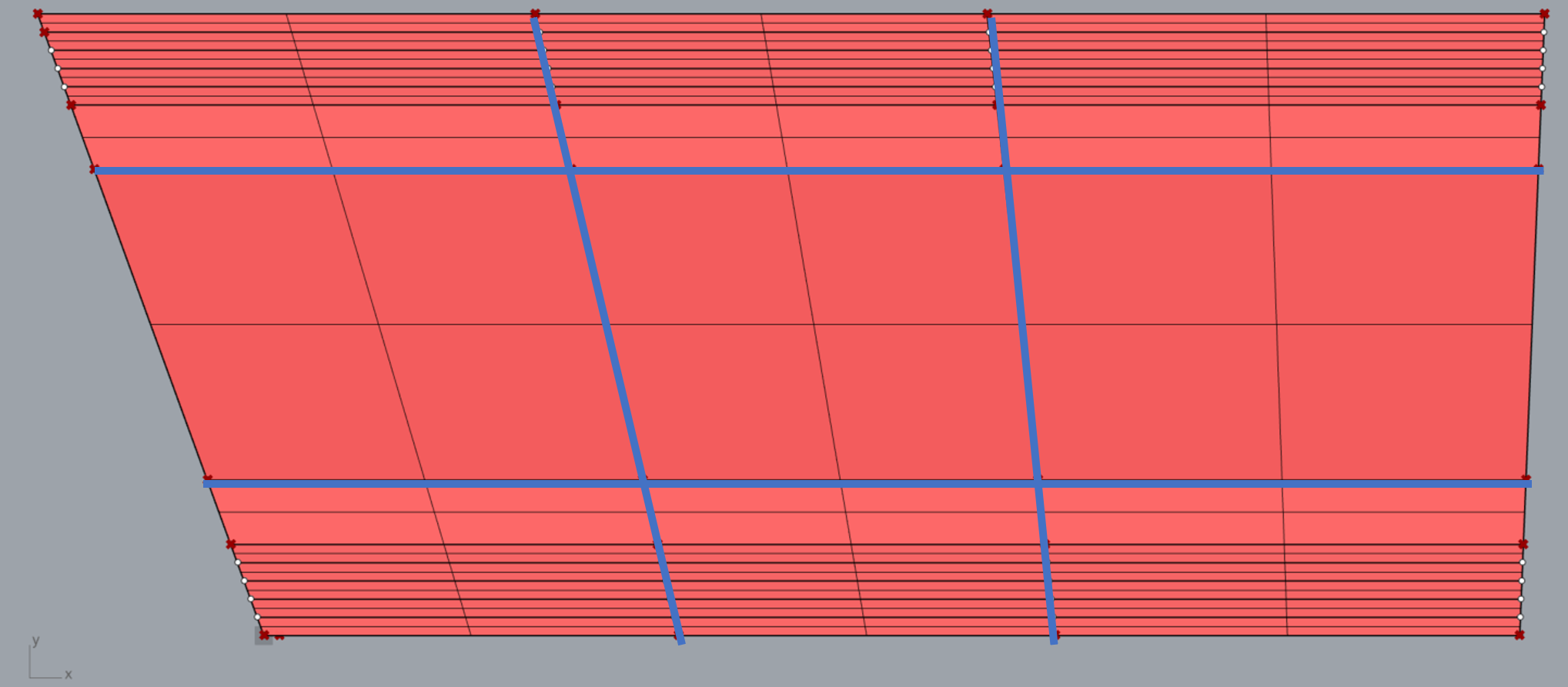
# initialize modeling cut of

cutof\_v= 0.8\* mesh\_size\_factor \* h\_FP

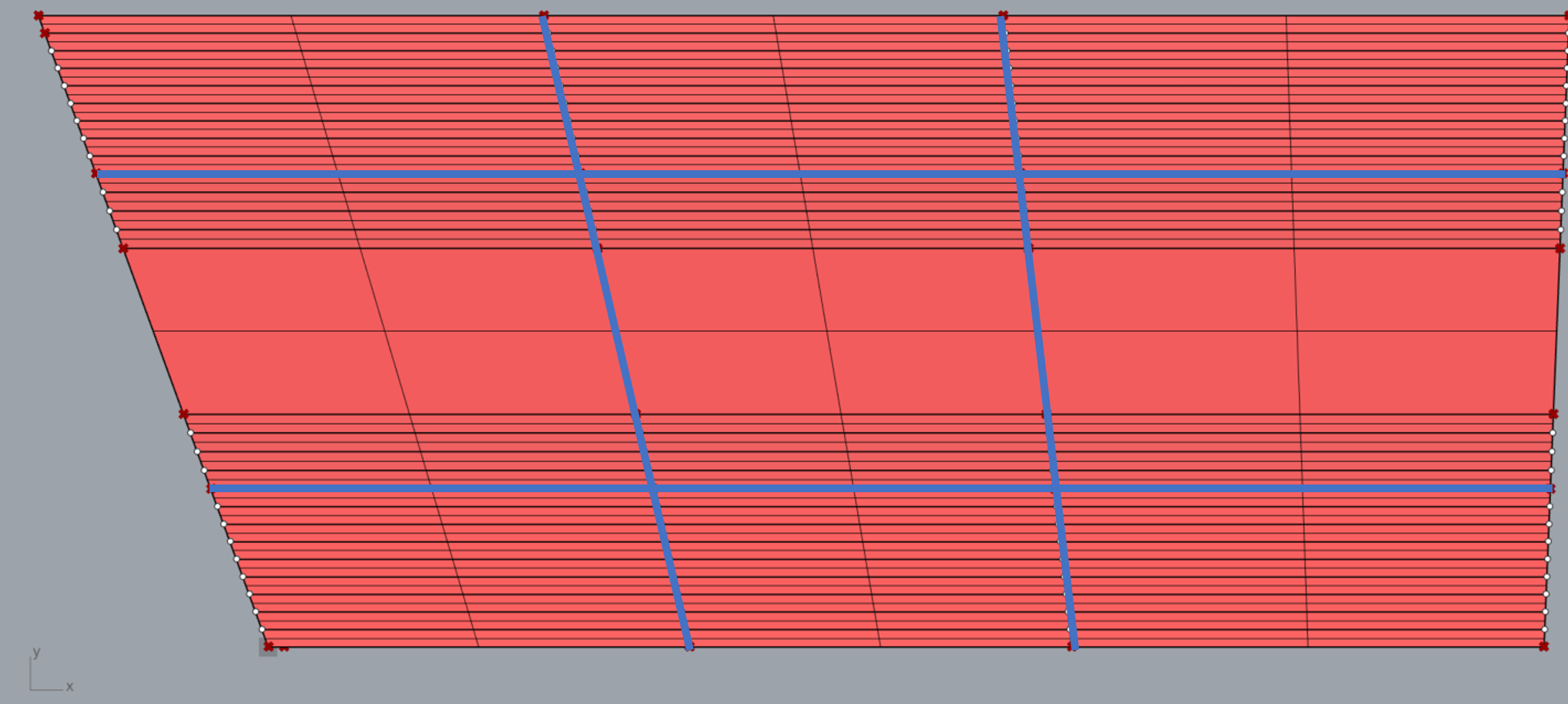
1. **Case 0:** h \_v < cutof\_v **or** l\_v < cutof\_v:
   1. the voute is not moddled if its length and hight is smaller than the cutoff length. 🡪 Simply 9 sections all with thrickness h\_FP
   2. Example: h\_FP=200 mm, mesh\_size\_factor =1 🡪 cutof\_v=199.89 mm 🡪 If either h\_v or l\_v smaller than cutoff\_v, then the voute is not modelled
   3. This is the case described in the plate section (cf. Section 2.2)

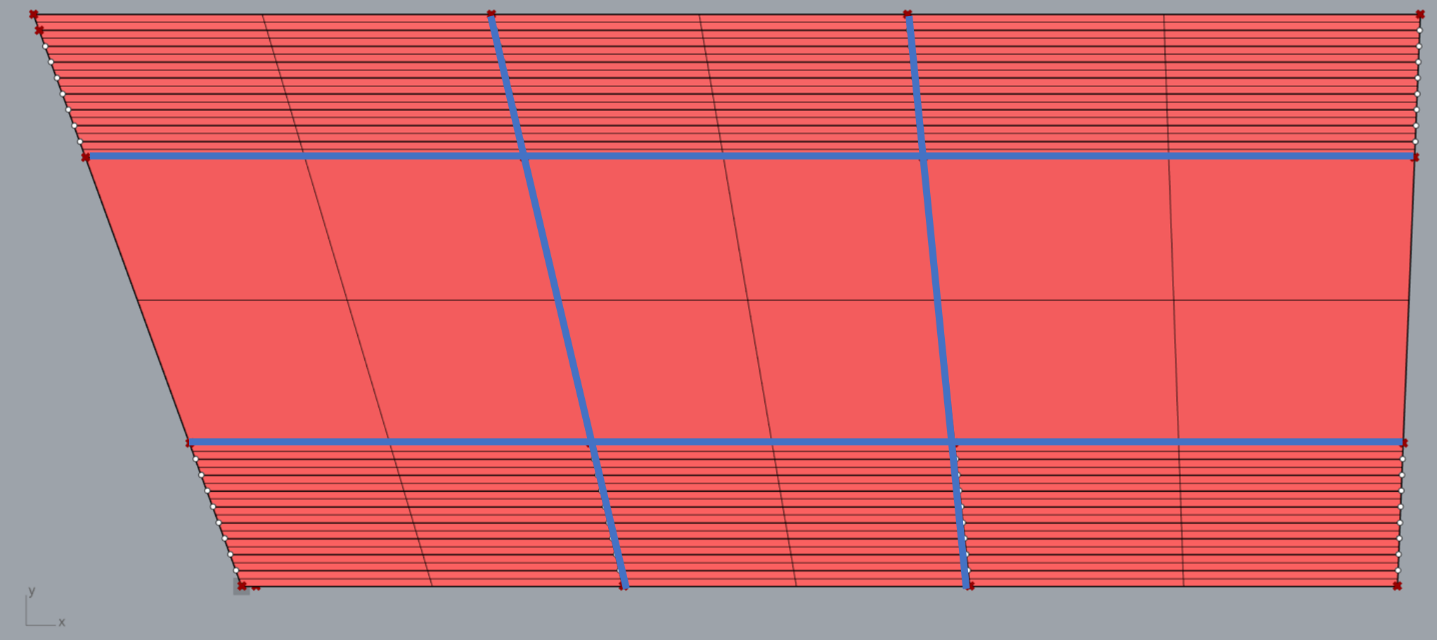


1. **Case 1:** (l\_v+t\_w/2) < l\_sec\*L -cutof\_v:
   1. The voute is reasonably large to be moddled and the length of the voute is smaller than l\_sec
   2. So there are vouten elements in the outer plate sections, aswell as non voute plate elements. In the inner sections of the plate (7-9) there are only elements with normal (h\_FP) thickness



1. **Case 2:** (l\_v+t\_w/2) > l\_sec\*L+cutof\_v
   1. The voute is longer than l\_sec\*L+cutof\_v



1. **Case 3:** Else
   1. So in the case that (l\_v+t\_w/2) > l\_sec\*L -cutof\_v AND (l\_v+t\_w/2) < l\_sec\*L+cutof\_v
   2. In this case the voute is roughly as long as the l\_sec\*L. This means in the outer plate sections we now have only voute elements with variable thickness. However in the inner plate sections (7-9) we only have elements with the normal Plate hight (h\_FP)

For each case the quad elements and their corresponding plate thicknesses calculated.

Here add how the hight of the voute elemnts is calculated.

Inlcude pictures of the 4 different cases (here not the meshed but only the quad lements) and then later in the meshing section provide screenshots of the 4 cases when its meshed.

From here onwards, it includes old notes, that are relevant for the model with vouten and angle etc.

### Parapet

The above Figures show the modelling of Plate and walls in the case no parapet is present. In this section the case is shown when a parapet (=Brüstung) exists.

The definition of the sections 1 to 9 stays the same for both the plates and the walls. Meaning that the distance for example between node 1 and 16 remains the same. To model the parapets however parallel to the ends and with a width of bb the parapet are modelled as separate quads.



Figure X: Node and quad definition of plate in the case of a parapet on the left (at x=0) and on the right (at x=b1).

The same as shown above is also done for both walls.

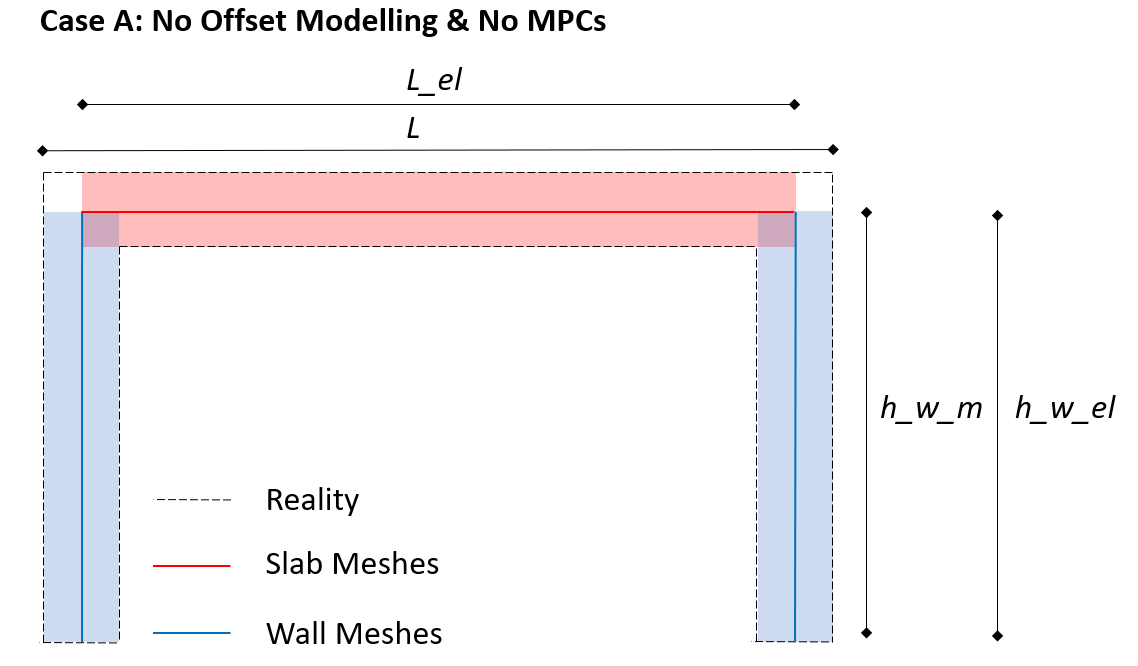
To be done hier sind die punkte nummerierung nur anders, und in der platte gibt’s extra sections, aber in wall keine extra sections.

Quad numbering picture with updated numbers and additional parapet quads, however make clear that they all go in the same section (as shown in Figure at the start and indicated by the numbering)

### Modelling options

I here show 4 different modelling options. We chose to go with the 4th option.

#### Case A:No offset modelling and no MPCs



Length of modelled plate element: L\_el=L-t\_w

Hight (distance) between plate element and lowest point of wall element: h\_w **=**(h\_F/2) + h\_w **+**(h\_FP/2)

Hight of Wall Element that is modelled: h\_w=h\_w\_el **=**(h\_F/2) + h\_w + h\_v **+**(h\_FP/2)

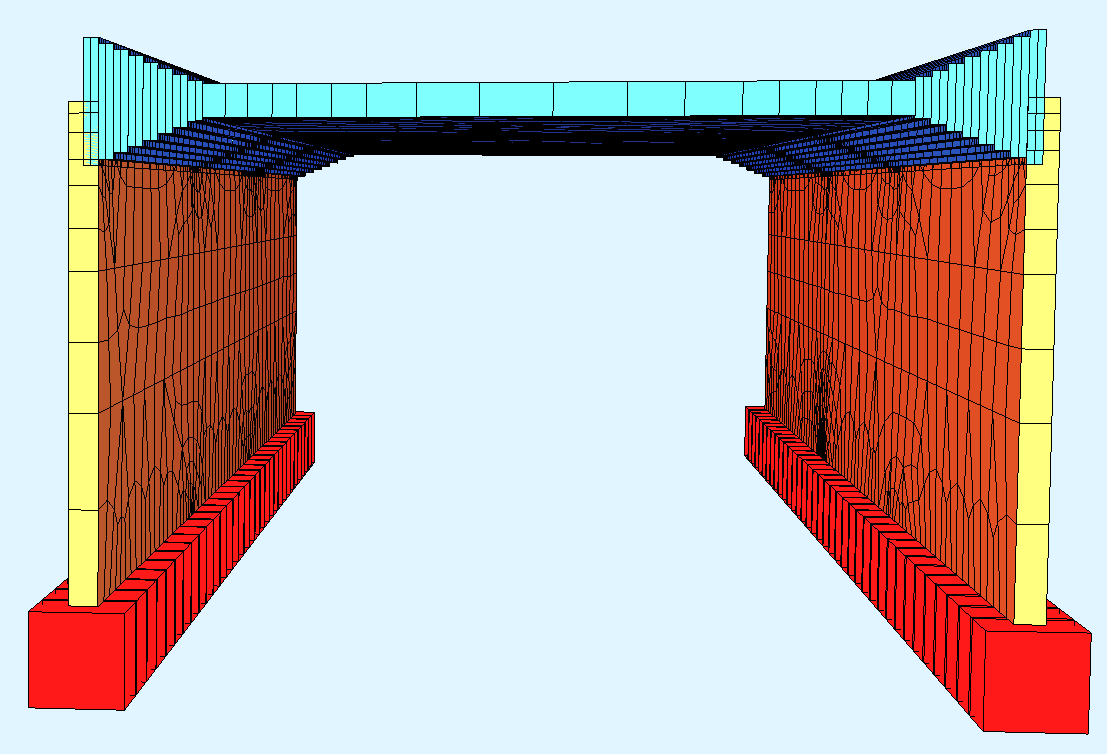
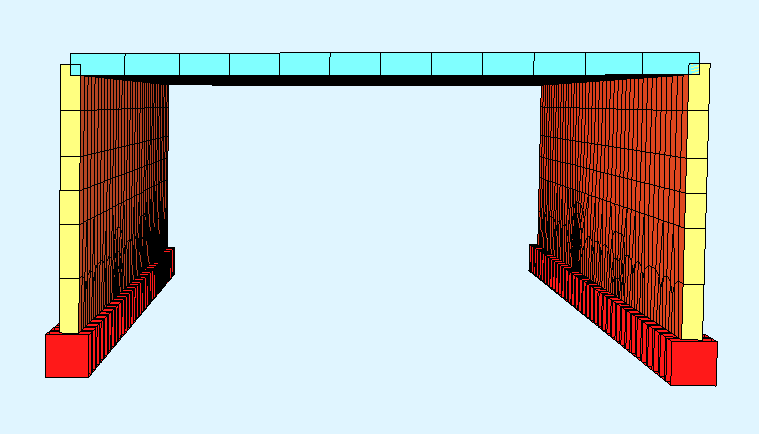
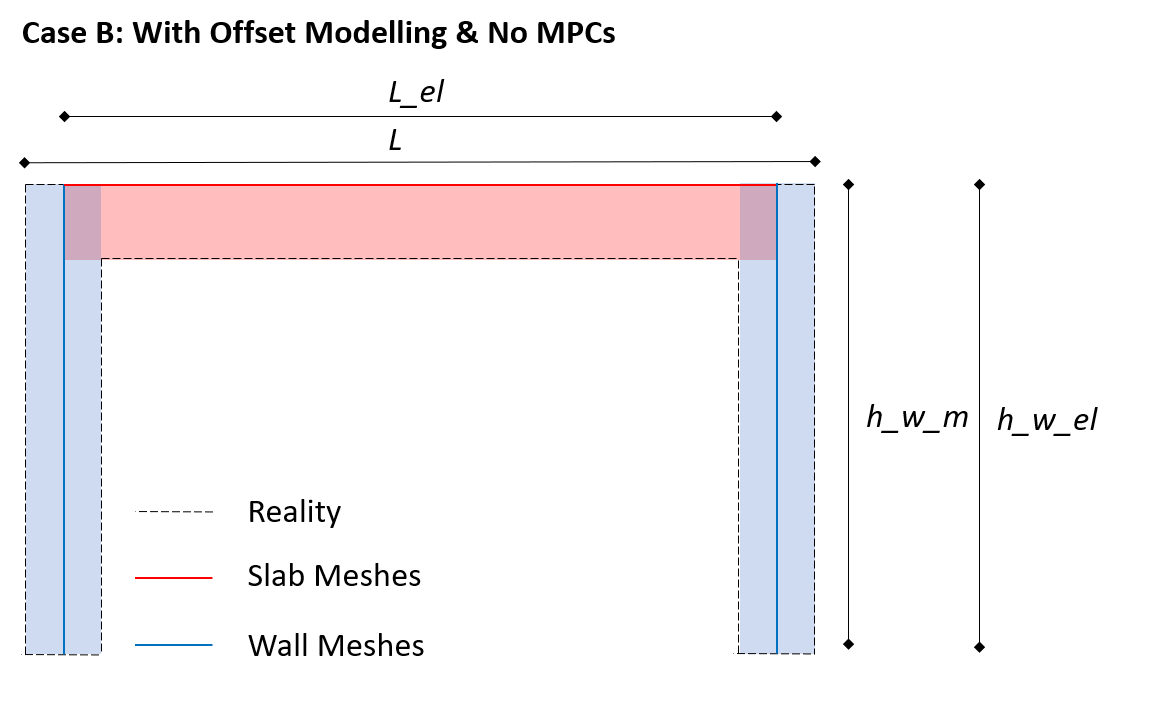


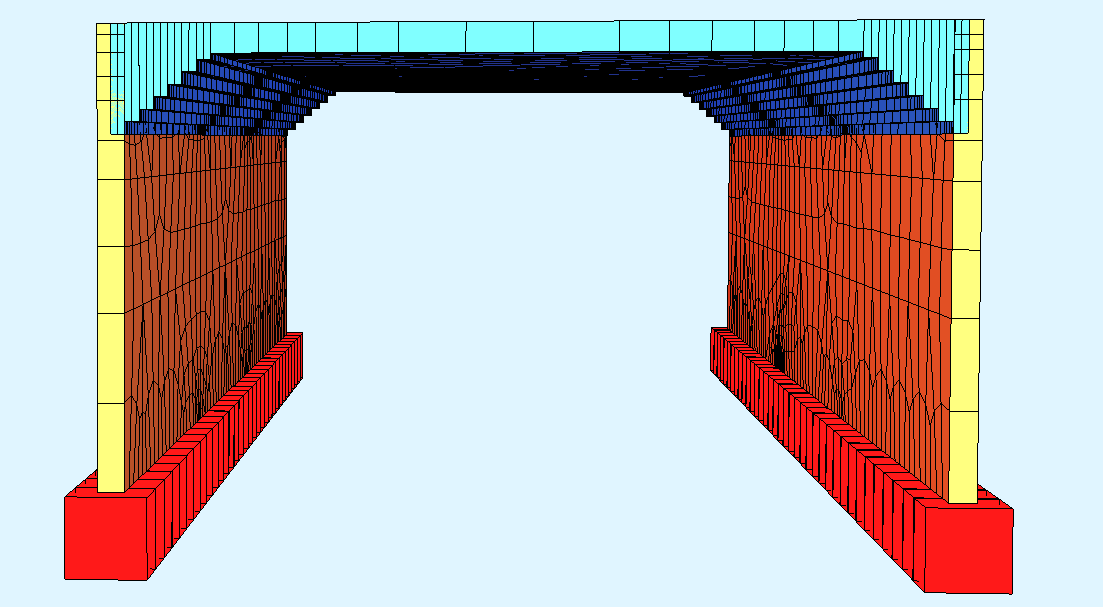
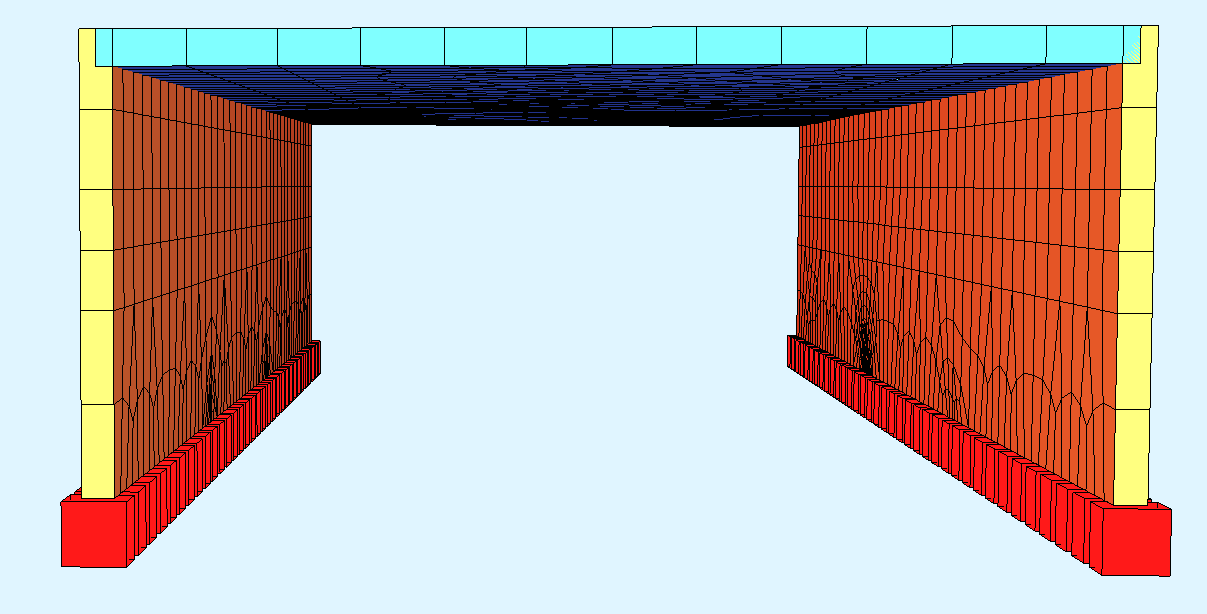
Figure 14: Structural System (without offset modelling and rigid links) (a) without voute, (b) with voute.

#### Case B: With offset modelling and no MPCs

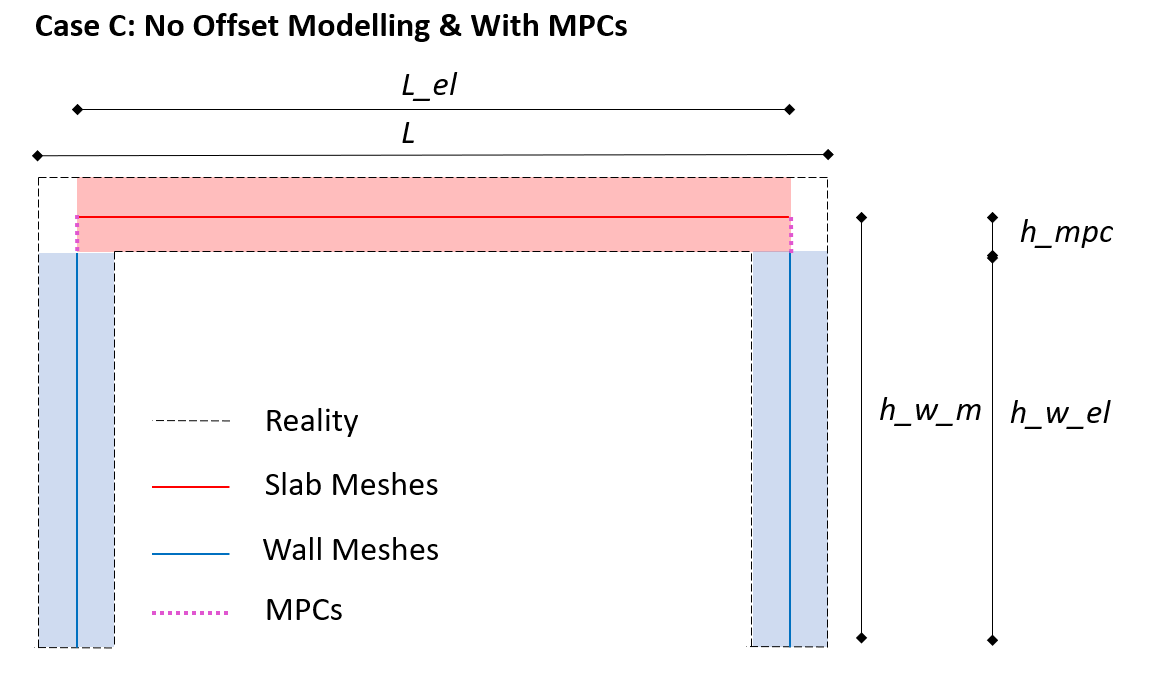


Length of modelled plate element: L\_el=L-t\_w (As we only model the slab with offset modelling, the Length of the slab elements (=distance between the wall elements) stays the same for all cases listed in this chapter.

Hight of Wall Element that is modelled: h\_w=h\_w\_el **=**(h\_F/2) + h\_w + h\_v **+**(h\_FP)



#### Case C: No offset modelling and with MPCs



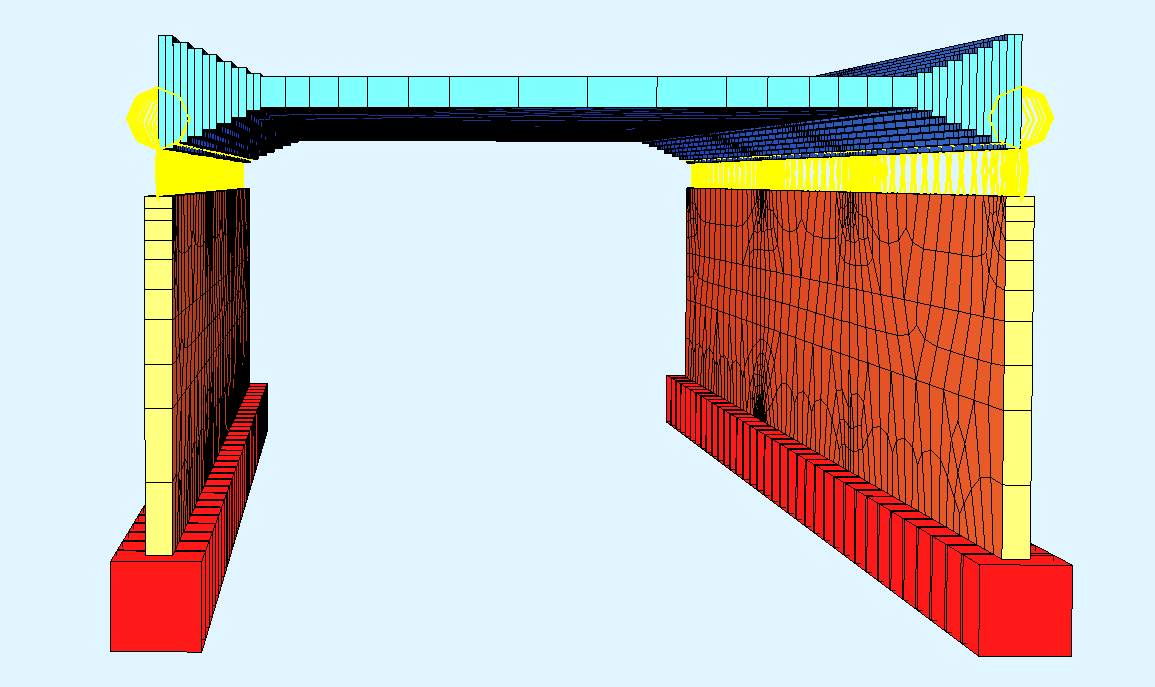
Length of modelled plate element: L\_el=L-t\_w

Hight (distance) between plate element and lowest point of wall element: h\_w\_m **=**(h\_F/2) + h\_v+ h\_w **+**(h\_FP/2)

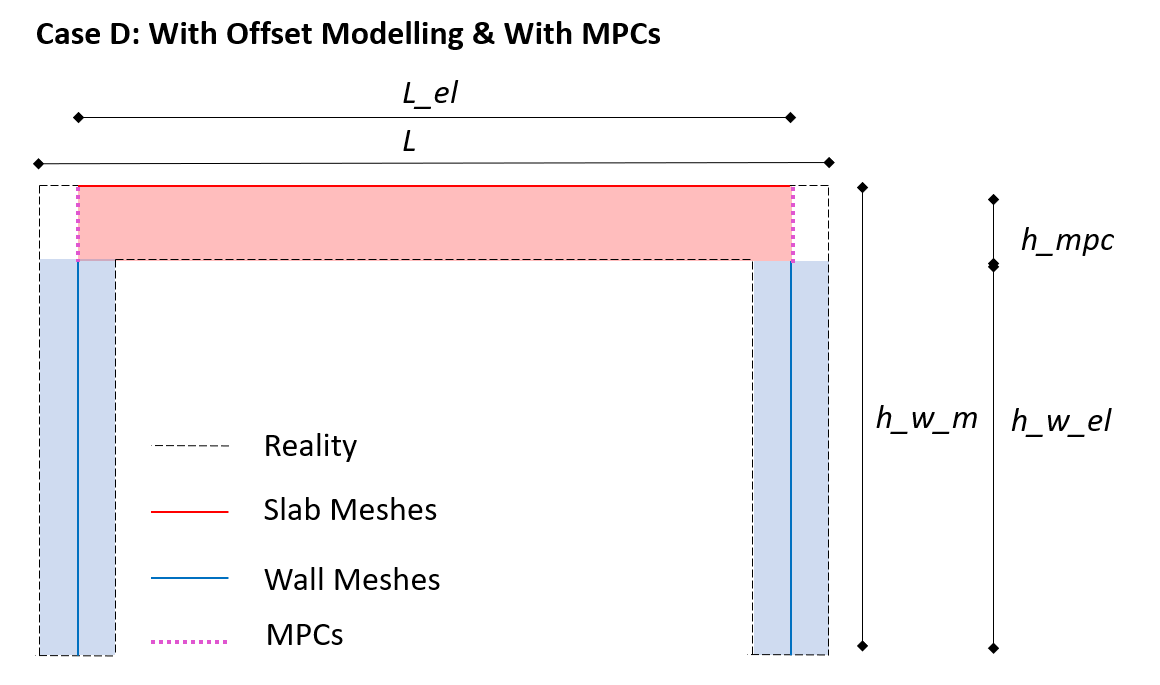
Hight of MPC elements: rlink\_h= h\_FP/2+h\_v (In Case 0 h\_v set to 0, cf. Secion )

Hight of Wall Element that is modelled: h\_w\_el = (h\_F/2) + h\_w (+h\_v in case of case 0)

(It always applies: h\_w\_el + h\_MPCs = h\_w\_m)



#### With offset modelling and with MPCs



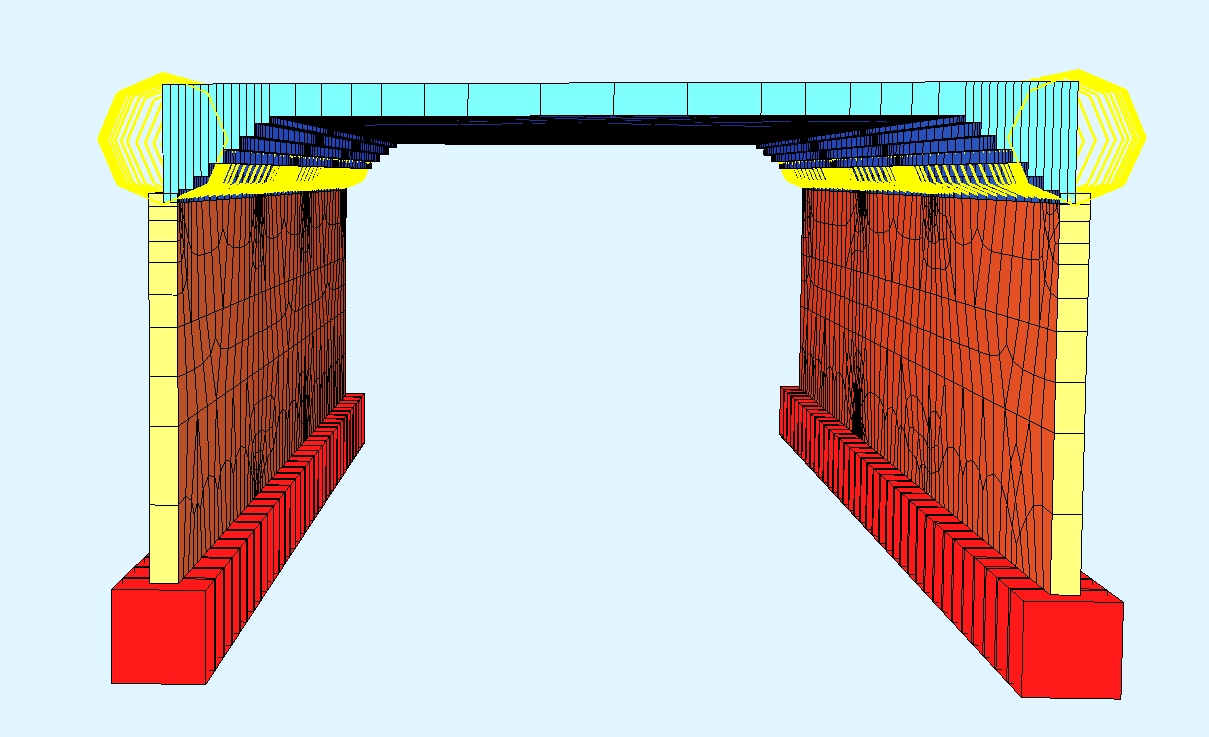
Length of modelled plate element: L\_el=L-t\_w

Hight (distance) between plate element and lowest point of wall element: h\_w\_m **=**(h\_F/2) + h\_v+ h\_w **+**(h\_FP)

Hight of MPC elements: rlink\_h= h\_FP+h\_v (In Case 0 h\_v set to 0, cf. Secion )

Hight of Wall Element that is modelled: h\_w\_el = (h\_F/2) + h\_w (+h\_v if case 0)

(It always applies: h\_w\_el + h\_MPCs = h\_w\_m)



### Meshing

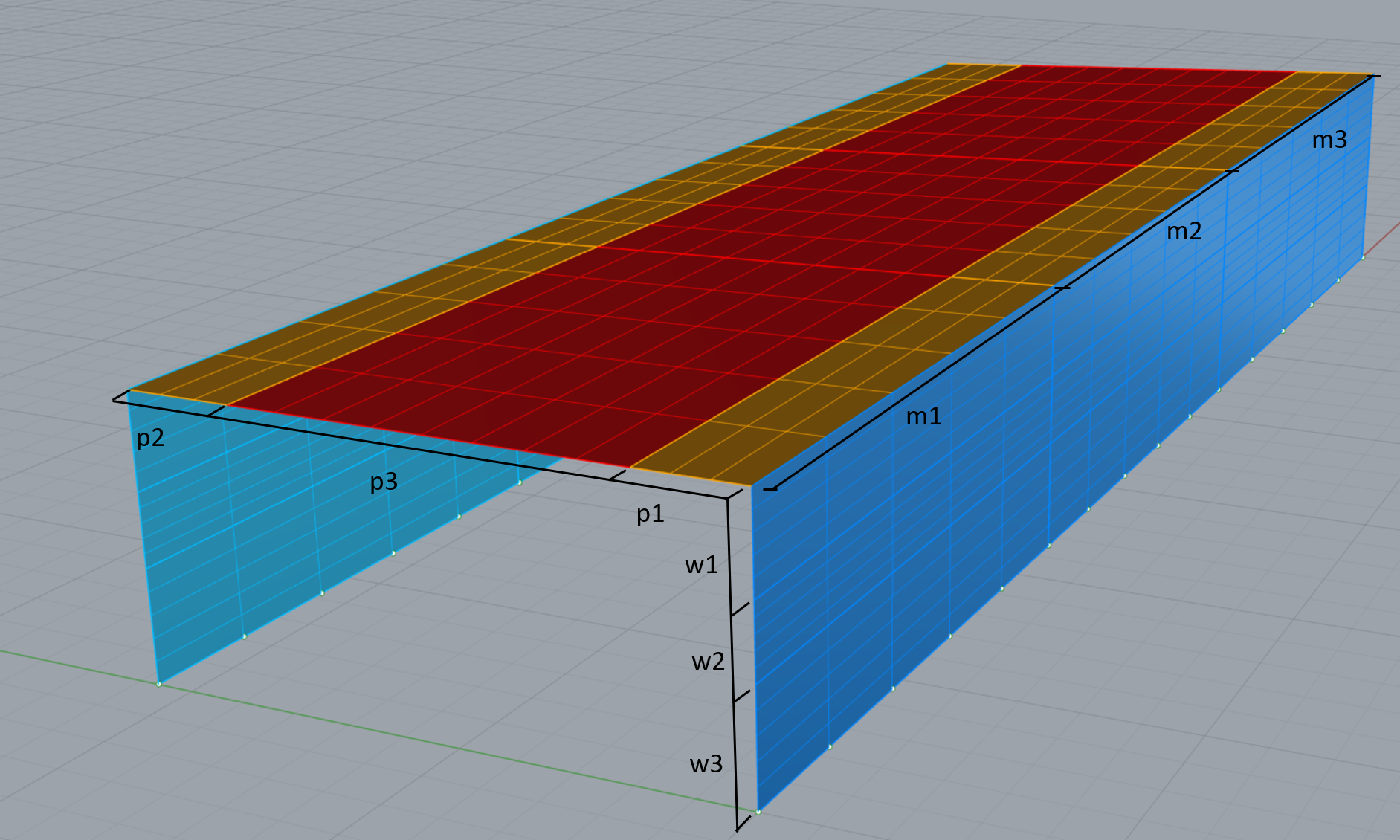
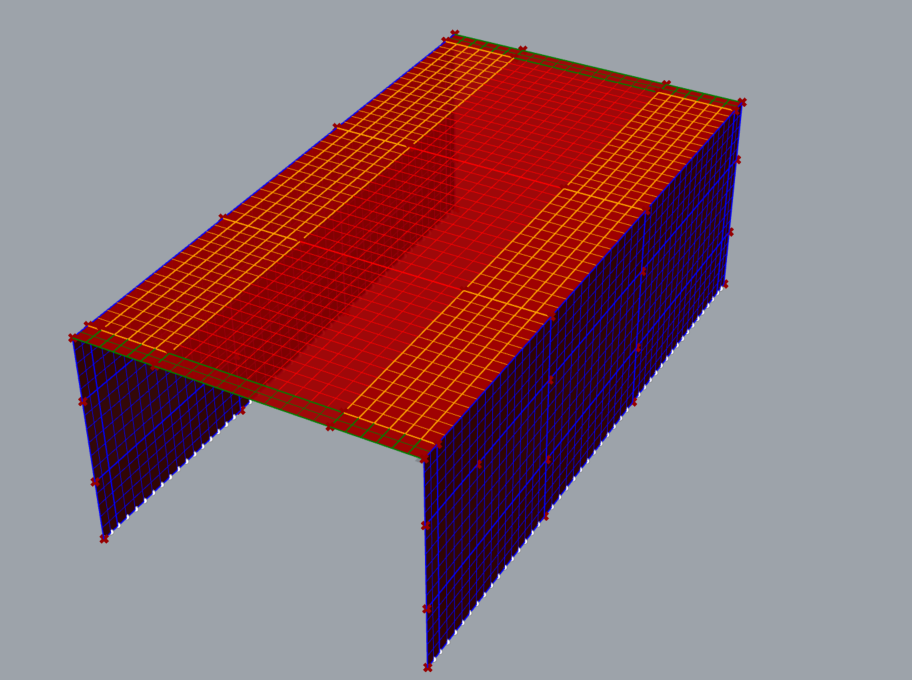


Figure 15: Frame bridge mesh example with meshing parameter (for the case that there is no voute)



The mesh sizes (rather mesh counts) are calculated separately for each section depending on the “mesh size factor” and the current plate/ wall thickness. This makes sure that all quad elements are meshed with roughly equal mesh sizes. Depending on the parameter combinations, individual mesh counts have to be rounded up, which results in slightly smaller mesh sizes in individual elements (but it is made sure all meshes are smaller than the defined mesh size). For each mesh the u\_count and v\_count value (=number of mesh elements of a side) is calculated and given to the corresponding mehing components. The calculation of mesh counts is executed in the plate\_generation and wall\_generation component. The mesh count values that are calculated are defined in the Figure above for the case where no voute is modelled. As the voute elements are already sized according to the mesh size factor and the plate thickness, the meshing only needs one element per vouten element in y direction.

Example: When the factor is set to one. All mesh sizes are equal or smaller than the corresponding element thickness. All plate (including the voute element) meshing elements have a meshing size equal or smaller than the thickness of the plate. All wall elements have meshing elements that are sized equal or smaller than the wall thickness.

#####TODO: Make 4 pictures of the 4 cases! Then we can see the difference, where p1 and p2 mean, and that the voute elements are width is already chosen according to masch\_size\_factor!

### Support Conditions

The nodes of the lowest mesh elements are extracted and defined as support points. This is done for wall 1 and wall 2. The support points are then saved in the layers *elset\_support\_wall1* and *elset\_support\_wall2*.

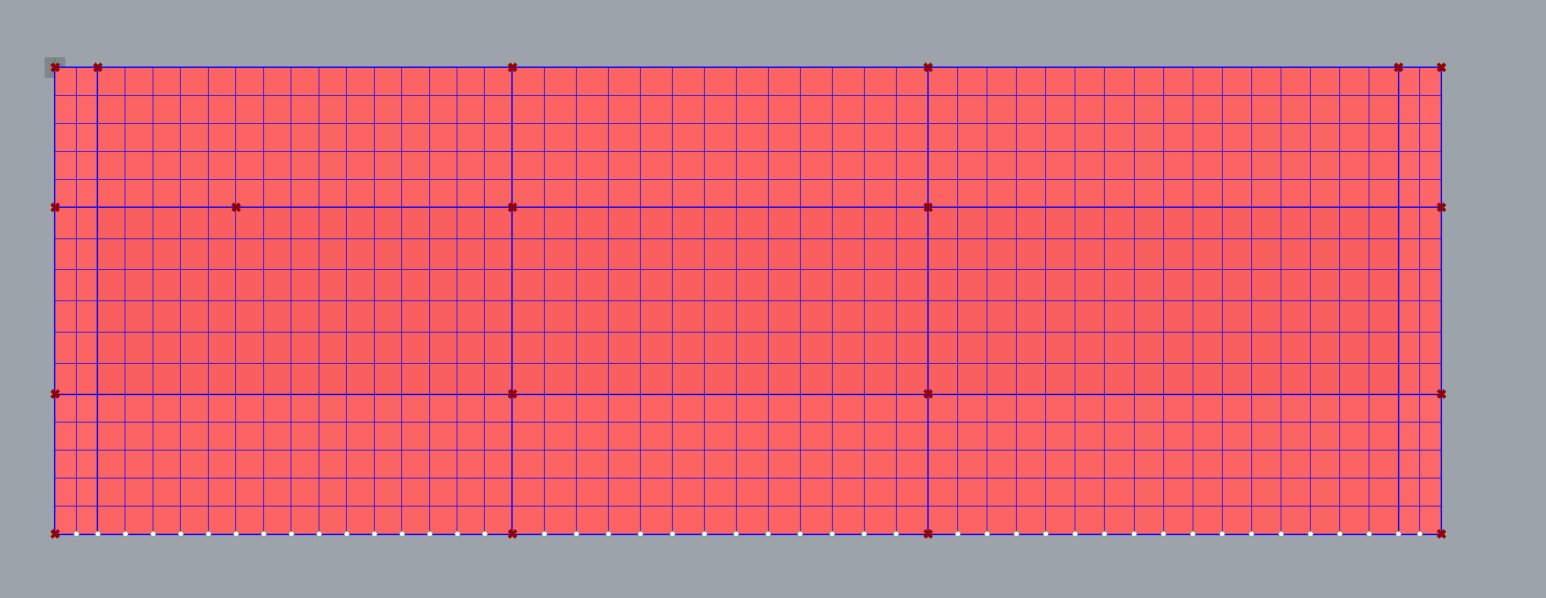


Figure 16: Wall mesh with extracted support points.

## Rhino Model/ File Generation

In this section I summarize how the generated geometry and the corresponding parameters is given from the Grasshopperfile to the rhino file.

### Layer Generation

In a Grasshopper Python Component a separate layer is created for each element group that has differing properties. Or in other words: A layer only contains elements which have all the same properties (meaning the same material, the same plate thickness etc.).

Methods:

* Add a layer: sc.doc.Layers.**Add**("nset\_support\_wall1", col.Gray)
* Get path to layer: layer\_path= sc.doc.Layers.**FindByFullPath**("nset\_support\_wall1", 0)
* Set a current active layer: sc.doc.Layers.**SetCurrentLayerIndex**(layer\_path, **False**)
* Add points to the active layer: sc.doc.Objects.**AddPoint**(support\_nodes\_wall1[i])
* Add meshed to the active layer: sc.doc.Objects.**AddMesh**(parap1[i])
* …

### Document User Text

The «DocumentUserText» is used to pass on all necessary parameters from the GH-File to the the Rhino-Document. The «DocumentUserText» is a dict.

There are some keys of the Dict that contain general information:

|  |  |
| --- | --- |
| key |  |
| “idx”  (This is the index/ identifier of the corresponding parameter combination) | [int] |
| «layer\_names»  This key inherits all layer names that this specific bridge has (to be iterated through) | List of strings (delimiter: ,) |
|  |  |

And there is a key for every layer that is generated. This dict entry saves the individual properties of the elements in each that layer. Each layer contains elements which all have the same properties.

|  |  |
| --- | --- |
| **Layer Group** | **List of Features saved** |
| Nset\_support  e.g.: | Fixed/unfixed |
| Elset\_plate & wall  e.g. Layers:  “elset\_wall\_2\_8”; “elset\_plate\_v\_1\_1”; “elset\_plate\_7” | fck, fsd, d\_top, s\_top, d\_bot, s\_bot, t |
| elset\_parapet  e.g.: ”elset\_parapet\_1” | fck, fsd, d\_top, s\_top, d\_bot, s\_bot, t |
| Elset\_mpcs  e.g.: elset\_ layer\_names |  |

Attention: So far, the user text is not deleated! So there are more keys than layers. One has to iterate through the layer names and not all existing keys. (To be fixed)

Methods: The command rs.**SetDocumentUserText**(key,value) is used to save the list of properties.

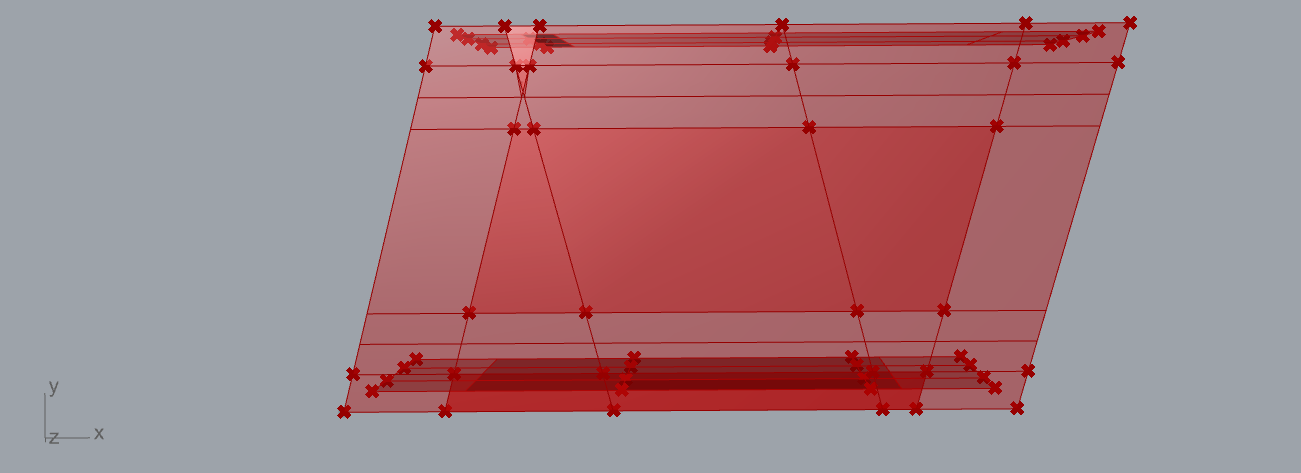
### File Saving

After all the geometry is generated, the layers are created and the right geometry is saved within each layer and the Document user text is written, the .3dm file is saved.

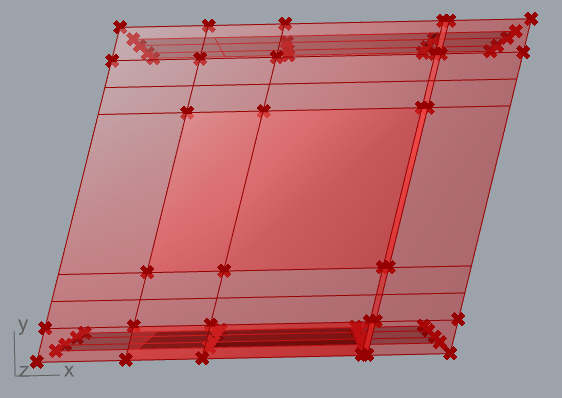
## Dependencies or Restrictions

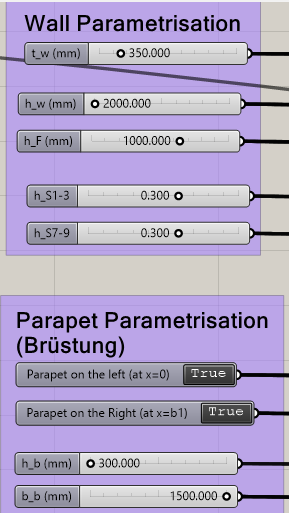
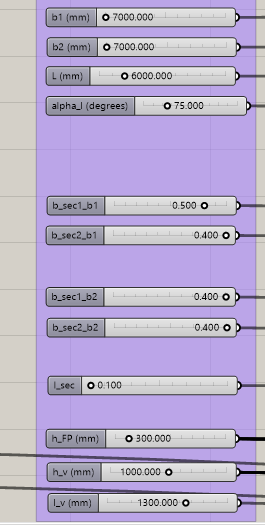
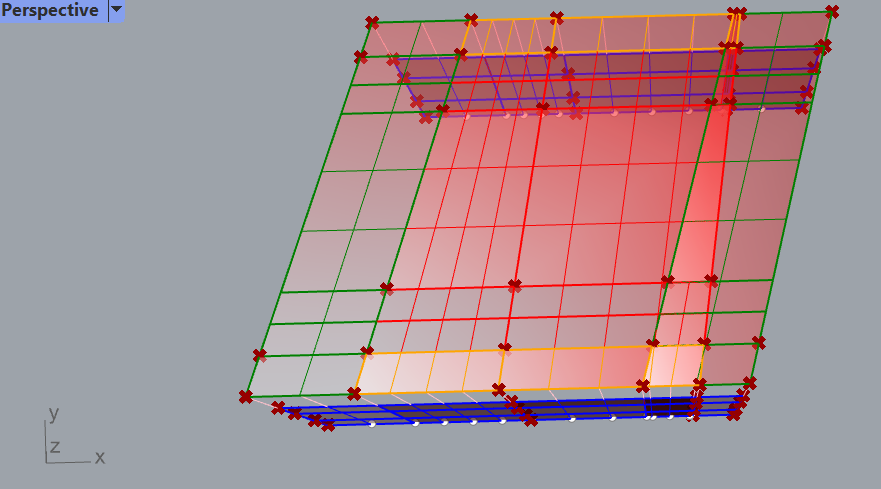
First and last section have to be larger than the Brüstungs width.

Otherwise this happenes:



Or this: (Overlapping of section 3 and Brüstung on the right)





# Evaluation Framework

The evaluation Framework can be accessed via Rhino. Utilising the StrucEngLibrary (= a library in the compas ecosystem) and using iron python scripts within python a structure model is constructed, which is analysed with the FEM software ansys.

## Iron python script: Build, Evaluate and Visualise Structure

This is an iron python script that utilises StrucEngLib and Compas functions to

* (i) define a structure with its geometry, material properties, section properties, support conditions and loads
* (ii) Analyse the defined structure (Sandwitchmodel or NLFE)
* (iii) Extract, save and Visualise analysis results.

### Define Structure

### Analyse

The results are stored in a dict which is saved in the structure object.

### Extract, save and Visualise Results

The structure object is saved as a pickle file.

Afterwards individual result values are extracted from the results dict in the structure object. This is done by reading the pickle file into a jupyter notebook, and extracting the parameters that we want.

### Function Documentation (StrucEngLib)

**Normalspurbahnverkehr\_load\_generator**(structure=mdl,name='Gleis', l\_Pl=L\_el, h\_Pl=**float**(rs.**GetDocumentUserText**(plate)), s=5000, beta=-30, q\_Gl=4.8, b\_Bs=2500, h\_Strich=300, Q\_k=225\*1000, y\_A=5000)

Parameter:

Return Value:

## Analysis Results

Results at Node:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| total displacements | esulting displacements, resulting out of displacement components ux, uy and uz at the nodes | um |  | Calculated out of ux, uy,uz |
| Displacement in global x direction | at the nodes | ux |  |  |
| Displacement in global y direction | at the nodes | uy |  |  |
| Displacement in global z direction | at the nodes | uz |  |  |

Results at element:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| membrane foces in local x-direction nx | For each element | sf1 |  |  |
| membrane forces in local y-direction ny | For each element | sf2 |  |  |
| shear forces in local xy-direction nxy | For each element | sf3 |  |  |
| transverse shear forces on local x-plane vy | For each element | sf4 |  |  |
| transverse shear forces on local y-plane vx | For each element | sf5 |  |  |
| bending moments around local x-direction m\_x | For each element | sm1 |  |  |
| bending moments around local y-direction m\_y | For each element | sm2 |  |  |
| twisting moments in x- and y-directions m\_xy | For each element | sm3 |  |  |

Results at each Gauss Point/ Integration point:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property** | **Description** | **Variable** | **Unit** | **Ranges/ Categories** |
| membran spannung in x richtung (top layer) | Concrete stress in glob? x direction in top layer | sig\_x\_top |  |  |
| membran spannung in y richtung (lop layer) | Concrete stress in glob? Y-direction in bottom layer | sig\_y\_top |  |  |
| shear spannung in der plattenebene (top layer) |  | tau\_xy\_top |  |  |
| Effective compressive strength of concrete | Effective concrete compressive strength that results from the reuction of the single-axis compressive strength fcd, due to the corresponding transverse strain state e1 (automaticly calculated in CMM-Usermat) | fcd,eff\_top |  |  |
| membran spannung in x richtung (bottom layer) | Concrete stress in glob? x direction in bottom layer | sig\_x\_bot |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  | loc\_y\_glob\_y |  |  |
|  |  | …. |  |  |
|  |  | coor\_intp\_layer\_x\_bot |  |  |
|  |  | …. |  |  |

# Iteration Loop for Data Genertaion

Currently the data generation is done in multiple individual steps, which all have to executed after eachother:

* (i) Sampling: Generation of (design-space filing) X vectors
* (ii) Geometry Generation: Generation of rhino.3dm files which include the geometry in the correct layer, relevant parameters in the User text
* (iii) Evaluation: The files that are generated in Step ii are imported iteratively into en empty rhino fule using an iron python script within rhino. In each iteration a single generated .3dm file is imported together with the corresponding X vector (imported from the csv file). Then the structure definition and analysis is done. And then for each iteration the structure object is saved.
* (iv) Extraction of perfromence attributes: While we save the structure objects which contain all the information. In an jupyter notebook: we iterate through all structure objects and extract the cooresponding perfromance attribute values for each structure and save all these values into a csv file. 🡪 extraction of Y vectors.

# Parameter study

In order to prove the functionality and usability of the framework without taking too much time, we decided to do the parameter study as well as the model training in multiple Steps. So we start with only a reduced set of parameter, which results in a reduced amount of bridge variability. And then step by step we want to increase the bridge variability, which results in more parameter and potentially larger parameter ranges.

Generally, the more parameter we have the more data points we have to evaluate to be able to train a reasonable machine learning model.

In the following I am documenting the different parameter study and modelling step we are taking.

## Step 1: Orthogonal Frame bridges without Vouten and Parapets

For step 1 we built a separate gh model, that only allowes for the here considered confrete frame bridge varaiability.

### Step 1.1 Homogeneous Reinforcement

At very first we vary the parameters for a whole element. Meaning we don’t vary any parameters in the individual 9 sections per element. We rather sample one reinforcement for each element (incl. Walls and deck slab).

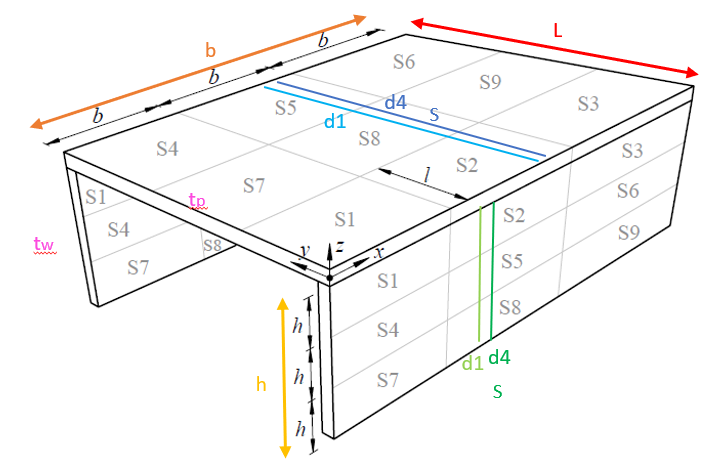


Figure 17: Visualisation of sampled parameters in parameter study step 1.1

The ranges where chosen following the data analysis of the SBB database of concrete frame bridges.

We have 16 variables, which are listed together with the ranges that were used for sampling in the tables below.

Geometric

|  |  |  |
| --- | --- | --- |
| **Variable** | **Unit** | **Ranges/ Categories** |
| L | mm | (3000,8000) |
| b1 | mm | (10000,20000) |
| t\_p | mm | (200,800) |
| t\_w | mm | (200,800) |
| h\_w | mm | (2000,5000) |

Reinforcement layout

|  |  |  |
| --- | --- | --- |
| **Variable** | **Unit** | **Ranges/ Categories** |
| d\_1\_ plate | mm | (10,12,14,16,18,20,22,26,30) |
| d\_4\_ plate | mm | (10,12,14,16,18,20,22,26,30) |
| s\_plate | mm | (75,100,150,200,250) |
| d\_1\_ walls | mm | (10,12,14,16,18,20,22,26,30) |
| d\_4\_ walls | mm | (10,12,14,16,18,20,22,26,30) |
| s\_walls | mm | (75,100,150,200,250) |

Material parameter

|  |  |  |
| --- | --- | --- |
| **Variable** | **Unit** | **Ranges/ Categories** |
| fcc | MPa | (40.,90) |
| fsy | MPa | (500., 700.) |
| Fsu\_fac | - | (1.0, 2.0) |

Loading Parameter

|  |  |  |
| --- | --- | --- |
| **Variable** | **Unit** | **Ranges/ Categories** |
| s | - | (0., 1.) |
| beta | ° | (-90, 90) |

Some parameters are calculated from the sampled parameters after sampling: Ecu, fsu (see variable definition in chapter 2.5). The remaining parameter are fixed for this step of the parameter study.

|  |  |  |
| --- | --- | --- |
| **Variable** | **Unit** | **Values** |
| b2 | mm | b1 |
| alpha\_l | ° | 90 |
| l\_sec | - | 0.33 |
| b\_sec1\_b1 | - | 0.33 |
| b\_sec2\_b1 | - | 0.33 |
| b\_sec1\_b2 | - | 0.33 |
| b\_sec2\_b2 | - | 0.33 |

…..

# Appendix A –

## Rhino hints

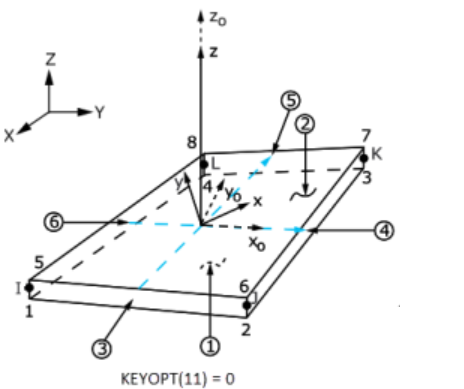
Visualise local coordinate system of a plane with Surface tools

And the local Z acsis is always out of the plane.

## Ansys Code Notes

<https://www.mm.bme.hu/~gyebro/files/ans_help_v182/ans_elem/Hlp_E_SHELL181.html>

The number the plates are assigned: (for the load application definition in the inp file)



## How to Visualize Ansys Model

1. Find .inp file of calculation

Anschauen der Berechnungen/ Ueberpruefungen der Berechnungen:

- Mechanical ADLP Product Launcher öffnen

- File name geben -> run

- .inp file in VS öffnen

- in Command line in launcher den text aus dem .inp file reinkopieren (bis zu der Stelle an der man den code laufen lassen will) --> Enter

- In Menu bar: Plot --> elements

- /eshape,1,1 (zeigt dir auch noch die dicken an)