

Development of a Parametric Membrane Design Software

MASTER-THESIS

A thesis submitted in partial fulfillment
of the requirements for the degree of

Master Membrane Structures

submitted to

Anhalt University of Applied Sciences

Faculty of Architecture,
Facility Management and Geo Information

by

Dipl.-Ing. HTL Thomas Stephan Dirk Dührsen
24.08.1971, Bern, Switzerland

Matrikel number
4065604

Submission date: 21.03.2019

First tutor: Prof. Dr. Robert Off

Second Tutor: Ing. M.Eng. Peter Novýsedlák, PhD., Archineer M.St

STATEMENT

I hereby declare that the work presented in this Master Thesis, entitled
Development of a Parametric Membrane Design Software

is entirely my own and that I did not use any sources or auxiliary
means other than those referenced.

Thun, March 2019

Thomas Stephan Dirk
Dührsen

*We should continually be striving to transform every art into a science:
in the process, we advance the art.*

— Donald E. Knuth

ABSTRACT

With this Master Thesis I am showing how the power and benefits of reusable and interchangeable logic and algorithms can be applied to the workflow of designing membrane structures especially for the steps of sketching, form finding, structural analysis, three dimensional visualizations and patterning. The focus is on membranes made of fabrics, but cable nets and structures covered with ETFE¹ cushions could also be dealt with.

The main work this document is based on was the development of a software library which seamlessly integrates into the parametric design plugin Grasshopper[®][19] of the Rhino3D CAD² system. But since the focus of the study program is on *Tensile Architecture*, aspects of software engineering are treated and documented only on a very high level. The focus is rather on the use cases for membrane design and visualization of the results from form finding and structural analysis.

The test results and examples listed in the document show that it was possible to finish all relevant assignments from the study program with this software. In addition, and even more important, the software is now being used successfully in real live customer projects to build beautiful membranes.

The reader should have a general understanding of tensile architecture and tasks commonly executed when designing membranes. Experience with working in 3D CAD software is assumed as well.

¹ Ethylene tetrafluoroethylene

² Computer Aided Design

An algorithm must be seen to be believed.

— Donald E. Knuth

ACKNOWLEDGMENTS

Many thanks to Peter Novýsedlák with his passion for Membranes and the Finite Element Method. His Master Thesis presentation *Tensile Structures – Numerical Design Techniques* [15] at the IMS Institute in March 2017 inspired me to pick up this topic and implement a software package for parametric design and analysis of membrane structures. Peter provided me with well documented Matlab scripts defining the core algorithms for form finding and analysis of membrane structures. These algorithms form the basis of my work and they were also used for testing purposes. Peter supported me with his extensive knowledge whenever I had a question or needed a second opinion.

CONTENTS

1	INTRODUCTION	1
2	MEMBRANE SOFTWARE	3
2.1	Background and Personal Needs	3
2.2	Market Overview	4
2.2.1	Software Evaluation Criteria	6
2.3	Requirements for a Parametric Membrane Software	6
2.3.1	Integrated System	7
2.3.2	Form Finding	7
2.3.3	Structural Analysis	7
2.3.4	Patterning	8
2.3.5	Detailing / Shop Drawings	8
2.3.6	Visualization	8
2.3.7	Bill of Material / Cost Estimation	8
3	CONCEPTS AND IMPLEMENTATION	9
3.1	Introduction	9
3.2	The Grasshopper [®] Plugin	9
3.3	Runtime Environment	10
3.4	Module Architecture	11
3.5	Tools and Development Environment	12
3.6	Behavioral Key Concepts	13
3.6.1	Smart Component Groups	13
3.6.2	Message Bubbles	15
3.6.3	Appearance of Smart Components	15
3.6.4	Component State	16
3.6.5	Unified Icons	17
3.7	Main Workflows	17
3.7.1	Workflow for Sketching	18
3.7.2	Workflow for Engineering	18
3.8	Model Definition	19
3.8.1	High Level Abstraction	19
3.8.2	Conceptual Elements	20
3.9	Mesh Definition and Handling	21
3.9.1	Mesh Creation	22
3.9.2	Adoption of Mesh Size	22
3.9.3	Mesh Granularity – Impact on Accuracy of Results	23
3.10	Form Finding	23
3.10.1	Warp Orientation	25
3.10.2	Limitations of Projecting Warp Direction	25
3.10.3	Considering Corner Connection Details for Form Finding	26
3.11	Load Cases	27

3.11.1	Wind Pressure Coefficients	27
3.11.2	Snow Roof Shape Coefficients	28
3.12	Structural Analysis	29
3.12.1	Finite Element Method (FEM)	29
3.12.2	Wrinkling Check	30
3.12.3	Enveloped Results	31
3.12.4	Cascading Solvers	32
3.13	Results Visualization	32
3.13.1	Geometry Preview	33
3.13.2	Colormaps to Visualize Numerical Values	34
3.13.3	Additional Visualizations for Values or Results	34
3.14	Patterning	35
3.14.1	Panel Boundaries	36
3.14.2	Flattening	37
3.14.3	Compensation	38
3.14.4	Seam Allowance	39
3.14.5	Checking Fabric Width	40
3.15	Detailing and 3D Visualization	41
3.15.1	Membranedetails	42
4	USER INTERFACE	43
4.1	Appearance in Grasshopper	43
4.1.1	Panels and their Order	43
4.1.2	Order within Panels	44
4.1.3	Information provided	45
4.2	Panel: Controls	46
4.2.1	Component: Direction (Pamela)	46
4.2.2	Component: List Multi Selector (Pamela)	48
4.2.3	Component: Vector Data (Pamela)	49
4.2.4	Component: XYZ Slider (Pamela)	51
4.3	Panel: Design	52
4.3.1	Component: Create Links (Pamela)	54
4.3.2	Component: Brep Mesher (Pamela)	55
4.3.3	Component: Edge Mesher (Pamela)	57
4.3.4	Component: Mesh Edges (Pamela)	59
4.3.5	Component: Ridge/Valley Cable (Pamela)	61
4.3.6	Component: Vertices on Curve (Pamela)	62
4.3.7	Component: Mesh Conditioner (Pamela)	63
4.3.8	Component: Pull Mesh to Curve (Pamela)	64
4.3.9	Component: Pull mesh to points (Pamela)	66
4.3.10	Component: Geodesic Curve (Pamela)	67
4.3.11	Component: Mesh Path (Pamela)	68
4.4	Panel: Material	69
4.4.1	Component: Cross Sections Database (Pamela)	69
4.4.2	Component: Material Database (Pamela)	70
4.4.3	Component: Cross Section Picker (Pamela)	71
4.4.4	Component: Material Picker (Pamela)	72

4.4.5	Component: Fabric Picker (Pamela)	74
4.4.6	Component: Material (Pamela)	75
4.4.7	Component: Membrane Material (Pamela)	77
4.4.8	Component: Cross Section Parameter (Pamela)	78
4.4.9	Component: Material Parameter (Pamela)	78
4.4.10	Component: Membrane Material Parameter (Pamela)	79
4.5	Panel: Model Definition	79
4.5.1	Component: 1D Element (Pamela)	80
4.5.2	Component: Membrane 2D Element (Pamela)	81
4.5.3	Component: Support (Pamela)	83
4.5.4	Component: Model Builder (Pamela)	85
4.5.5	Component: Copy Final Model (Pamela)	86
4.5.6	Component: 1D Element Parameter (Pamela)	87
4.5.7	Component: 2D Element Parameter (Pamela)	88
4.5.8	Component: Model Parameter (Pamela)	88
4.5.9	Component: Node Parameter (Pamela)	88
4.5.10	Component: Support Parameter (Pamela)	88
4.6	Panel: Model Loads	89
4.6.1	Component: Load Coefficients SUI (Pamela)	89
4.6.2	Component: Load Case GER (Pamela)	91
4.6.3	Component: Load Case SUI (Pamela)	92
4.6.4	Component: Load Calculations GER (Pamela)	94
4.6.5	Component: Load Calculations SUI (Pamela)	95
4.6.6	Component: Load Case (Pamela)	97
4.6.7	Component: Membrane Snow Shape Coefficients (Pamela)	98
4.6.8	Component: Membrane Pressure Coefficients (Pamela)	100
4.6.9	Component: Snowload (Pamela)	101
4.6.10	Component: Windpressure (Pamela)	102
4.6.11	Component: Load Case Parameter (Pamela)	103
4.7	Panel: Solver	103
4.7.1	Component: Force Density Solver (Pamela)	104
4.7.2	Component: Finite Element Method Solver (Pamela)	105
4.7.3	Component: Execute Matlab (Pamela)	107
4.8	Panel: Views	108
4.8.1	Component: Mesh Details (Pamela)	109
4.8.2	Component: Model Explorer (Pamela)	110
4.8.3	Component: Model Viewer (Pamela)	112
4.8.4	Component: Contour Lines (Pamela)	113
4.8.5	Component: 1D Axial Forces (Pamela)	115
4.8.6	Component: Mesh Slope (Pamela)	116
4.8.7	Component: 1D Results View (Pamela)	117
4.8.8	Component: 2D Results View (Pamela)	119
4.8.9	Component: Snow Roof Shape Evaluation (Pamela)	120
4.8.10	Component: Wind cp Evaluation (Pamela)	121

4.8.11	Component: 1D Element Handle (Pamela)	122
4.8.12	Component: 2D Element Handle (Pamela)	124
4.8.13	Component: Model Handle (Pamela)	126
4.8.14	Component: Node Handle (Pamela)	127
4.8.15	Component: Support Handle (Pamela)	128
4.8.16	Component: Colormap (Pamela)	130
4.8.17	Component: Mesh Values (Pamela)	131
4.9	Panel: Detailing	133
4.9.1	Component: Connectorplate (Pamela)	133
4.9.2	Component: Pfeifer 860 (Pamela)	135
4.9.3	Component: Rectangular anchor plate (Pamela)	136
4.9.4	Component: Round anchor plate (Pamela)	137
4.10	Panel: Membrane	139
4.10.1	Component: Membrane Pimper (Pamela)	139
4.11	Panel: Patterning	140
4.11.1	Component: Flattener (Pamela)	141
4.11.2	Component: Create Splitline (Pamela)	142
4.11.3	Component: Split Mesh (Pamela)	143
4.12	Panel: Utility	144
4.12.1	Component: Log Configuration (Pamela)	144
4.12.2	Component: LaTeX Doc (Pamela)	145
4.12.3	Component: Import Results (Pamela)	146
4.12.4	Component: Write To File (Pamela)	148
4.12.5	Component: Input Collector (Pamela)	149
4.12.6	Component: Run Executable (Pamela)	150
5	TESTING AND PROOF OF CONCEPT	153
5.1	Introduction	153
5.1.1	Unit Testing	153
5.1.2	System Testing	154
5.2	Testing against Reference implementation	154
5.2.1	Test Scenarios	155
5.2.2	Test Cases	155
5.2.3	Results Verification	155
5.2.4	Verification of Convergence Behavior	156
5.2.5	Verification of Numerical Values	158
5.3	Example Hypar with Ridge Cable	159
5.3.1	Task	159
5.3.2	Preliminary Remarks	159
5.3.3	Design	160
5.3.4	Model Definition	160
5.3.5	Form Finding	166
5.3.6	Loads	167
5.3.7	Serviceability / Deflection	168
5.3.8	Stress Analysis	169
5.3.9	Conclusion	171
5.4	The Real-life Proof of Concept	172

6	SUMMARY	173
6.1	Limitations	174
6.2	Outlook	175
6.2.1	Business Model	176
6.2.2	Integration with other Software	176
6.3	Personal Findings	177
	BIBLIOGRAPHY	179

LIST OF FIGURES

Figure 3.1	Grasshopper® main window (source: www.wikipwdia.org)	10
Figure 3.2	Rhino technology overview (source: developer.rhino3d.com)	11
Figure 3.3	internal structure	12
Figure 3.4	Trello project management tool	12
Figure 3.5	software usability(source: blog.rocketsoftware.com)	13
Figure 3.6	Ridge/Valley cable component as example for smart component group	14
Figure 3.7	Model Builder component with message bubble	16
Figure 3.8	appearance of smart component groups	16
Figure 3.9	colors indicating the components state	17
Figure 3.10	examples of icons	17
Figure 3.11	generic behavior of computer programs	17
Figure 3.12	Sketching workflow	18
Figure 3.13	Engineering workflow	19
Figure 3.14	model definition	20
Figure 3.15	elements of a geometry	21
Figure 3.16	Model Builder Component in use	21
Figure 3.17	warp on twisted membrane	25
Figure 3.18	generic corner plate	26
Figure 3.19	context of structural analysis	29
Figure 3.20	FEM algorithm	29
Figure 3.21	Classdiagram for FEM datamodel	30
Figure 3.22	Enveloped Results	31
Figure 3.23	FEM Solver	32
Figure 3.24	result types in context	33
Figure 3.25	preview of model	33
Figure 3.26	colormap applied to mesh values	34
Figure 3.27	Contourlines on mesh surface	35
Figure 3.28	Patterning process	36
Figure 3.29	Grasshopper definition for patterning	37
Figure 3.30	panel layout using geodesic lines	37
Figure 3.31	flattened and oriented 2D panels	38
Figure 3.32	detail of compensated panel	39
Figure 3.33	measurements for cable pockets	40
Figure 3.34	measurements for ridge or valley cable pockets	40
Figure 3.35	dimensions of flattened and compensated panel	41
Figure 3.36	Dimensions of connector plate[16]	41
Figure 3.37	Visualizations of connector and anchor plates	42
Figure 4.1	all panels in the tab <i>Pamela</i>	43

Figure 4.2	order of the components in the panel <i>Design</i>	45
Figure 4.3	overview of panel <i>Controls</i>	46
Figure 4.4	Screenshot of <i>Direction (Pamela)</i>	47
Figure 4.5	Screenshot of <i>List Multi Selector (Pamela)</i>	49
Figure 4.6	Screenshot of <i>Vector Data (Pamela)</i>	50
Figure 4.7	Screenshot of <i>XYZ Slider (Pamela)</i>	52
Figure 4.8	overview of panel <i>Design</i>	53
Figure 4.9	Screenshot of <i>Create Links (Pamela)</i>	55
Figure 4.10	Screenshot of <i>Brep Mesher (Pamela)</i>	57
Figure 4.11	Screenshot of <i>Edge Mesher (Pamela)</i>	59
Figure 4.12	Screenshot of <i>Mesh Edges (Pamela)</i>	60
Figure 4.13	Screenshot of <i>Ridge/Valley Cable (Pamela)</i>	62
Figure 4.14	Screenshot of <i>Vertices on Curve (Pamela)</i>	63
Figure 4.15	Screenshot of <i>Mesh Conditioner (Pamela)</i>	64
Figure 4.16	Screenshot of <i>Pull Mesh to Curve (Pamela)</i>	65
Figure 4.17	Screenshot of <i>Pull mesh to points (Pamela)</i>	66
Figure 4.18	Screenshot of <i>Geodesic Curve (Pamela)</i>	67
Figure 4.19	Screenshot of <i>Mesh Path (Pamela)</i>	68
Figure 4.20	overview of panel <i>Material</i>	69
Figure 4.21	Screenshot of <i>Cross Sections Database (Pamela)</i>	70
Figure 4.22	Screenshot of <i>Material Database (Pamela)</i>	71
Figure 4.23	Screenshot of <i>Cross Section Picker (Pamela)</i>	72
Figure 4.24	Screenshot of <i>Material Picker (Pamela)</i>	73
Figure 4.25	Screenshot of <i>Fabric Picker (Pamela)</i>	75
Figure 4.26	Screenshot of <i>Material (Pamela)</i>	76
Figure 4.27	Screenshot of <i>Membrane Material (Pamela)</i>	78
Figure 4.28	overview of panel <i>Model Definition</i>	79
Figure 4.29	Screenshot of <i>1D Element (Pamela)</i>	81
Figure 4.30	Screenshot of <i>Membrane 2D Element (Pamela)</i>	83
Figure 4.31	Screenshot of <i>Support (Pamela)</i>	84
Figure 4.32	Screenshot of <i>Model Builder (Pamela)</i>	86
Figure 4.33	Screenshot of <i>Copy Final Model (Pamela)</i>	87
Figure 4.34	overview of panel <i>Model Loads</i>	89
Figure 4.35	Screenshot of <i>Load Coefficients SUI (Pamela)</i>	90
Figure 4.36	Screenshot of <i>Load Case GER (Pamela)</i>	92
Figure 4.37	Screenshot of <i>Load Case SUI (Pamela)</i>	94
Figure 4.38	Screenshot of <i>Load Calculations GER (Pamela)</i>	95
Figure 4.39	Screenshot of <i>Load Calculations SUI (Pamela)</i>	96
Figure 4.40	Screenshot of <i>Load Case (Pamela)</i>	98
Figure 4.41	Screenshot of <i>Membrane Snow Shape Coefficients (Pamela)</i>	99
Figure 4.42	Screenshot of <i>Membrane Pressure Coefficients (Pamela)</i>	100
Figure 4.43	Screenshot of <i>Snowload (Pamela)</i>	101
Figure 4.44	Screenshot of <i>Windpressure (Pamela)</i>	102
Figure 4.45	overview of panel <i>Solver</i>	103
Figure 4.46	Screenshot of <i>Force Density Solver (Pamela)</i>	105

Figure 4.47	Screenshot of <i>Finite Element Method Solver (Pamela)</i>	107
Figure 4.48	Screenshot of <i>Execute Matlab (Pamela)</i>	108
Figure 4.49	overview of panel <i>Views</i>	108
Figure 4.50	Screenshot of <i>Mesh Details (Pamela)</i>	110
Figure 4.51	Screenshot of <i>Model Explorer (Pamela)</i>	111
Figure 4.52	Screenshot of <i>Model Viewer (Pamela)</i>	113
Figure 4.53	Screenshot of <i>Contour Lines (Pamela)</i>	114
Figure 4.54	Screenshot of <i>1D Axial Forces (Pamela)</i>	116
Figure 4.55	Screenshot of <i>Mesh Slope (Pamela)</i>	117
Figure 4.56	Screenshot of <i>1D Results View (Pamela)</i>	118
Figure 4.57	Screenshot of <i>2D Results View (Pamela)</i>	120
Figure 4.58	Screenshot of <i>Snow Roof Shape Evaluation (Pamela)</i>	121
Figure 4.59	Screenshot of <i>Wind cp Evaluation (Pamela)</i>	122
Figure 4.60	Screenshot of <i>1D Element Handle (Pamela)</i>	124
Figure 4.61	Screenshot of <i>2D Element Handle (Pamela)</i>	125
Figure 4.62	Screenshot of <i>Model Handle (Pamela)</i>	127
Figure 4.63	Screenshot of <i>Node Handle (Pamela)</i>	128
Figure 4.64	Screenshot of <i>Support Handle (Pamela)</i>	129
Figure 4.65	Screenshot of <i>Colormap (Pamela)</i>	131
Figure 4.66	Screenshot of <i>Mesh Values (Pamela)</i>	133
Figure 4.67	overview of panel <i>Detailing</i>	133
Figure 4.68	Screenshot of <i>Connectorplate (Pamela)</i>	135
Figure 4.69	Screenshot of <i>Pfeifer 860 (Pamela)</i>	136
Figure 4.70	Screenshot of <i>Rectangular anchor plate (Pamela)</i>	137
Figure 4.71	Screenshot of <i>Round anchor plate (Pamela)</i>	139
Figure 4.72	overview of panel <i>Membrane</i>	139
Figure 4.73	Screenshot of <i>Membrane Pimper (Pamela)</i>	140
Figure 4.74	overview of panel <i>Patterning</i>	141
Figure 4.75	Screenshot of <i>Flattener (Pamela)</i>	142
Figure 4.76	Screenshot of <i>Create Splitline (Pamela)</i>	143
Figure 4.77	Screenshot of <i>Split Mesh (Pamela)</i>	144
Figure 4.78	overview of panel <i>Utility</i>	144
Figure 4.79	Screenshot of <i>Log Configuration (Pamela)</i>	145
Figure 4.80	Screenshot of <i>LaTeX Doc (Pamela)</i>	146
Figure 4.81	Screenshot of <i>Import Results (Pamela)</i>	147
Figure 4.82	Screenshot of <i>Write To File (Pamela)</i>	149
Figure 4.83	Screenshot of <i>Input Collector (Pamela)</i>	150
Figure 4.84	Screenshot of <i>Run Executable (Pamela)</i>	151
Figure 5.1	Screenshot of <i>Execute Matlab (Pamela)</i>	155
Figure 5.2	parallel solver setup	158
Figure 5.3	distance of nodal coordinates	159
Figure 5.4	Dimensions for Hypar with ridge cable	160
Figure 5.5	Complete model definition for task 1	161
Figure 5.6	Mesh creation and refinement	162
Figure 5.7	Assembling the data model	164
Figure 5.8	Load case definitions	165

Figure 5.9	Daisy-chain of solvers	165
Figure 5.10	Solver results summary	166
Figure 5.11	Displacements due to snow	169

LIST OF TABLES

Table 3.1	Results comparison for different mesh sizes	24
Table 3.2	default cp values	28
Table 3.3	default roof shape coefficients	28
Table 5.1	Initial and refined mesh	162
Table 5.2	Mesh after form finding	166
Table 5.3	Wind load on Hypar	167
Table 5.4	Snow load and enveloped results	168
Table 5.5	Snow load and enveloped results	168
Table 5.6	Membrane Prestress	169
Table 5.7	Membrane stress due to snow load	170
Table 5.8	Maximum membrane stress	170
Table 5.9	Axial forces	171

LISTINGS

Listing 5.1	Convergence of PAMELA Implementation	156
Listing 5.2	Convergence of Matlab Implementation	157

ACRONYMS

API	Application Programming Interface
BREP	Boundary Representation
CAD	Computer Aided Design
CPU	Central Processing Unit
DOF	Degree of Freedom

DPI	Dots Per Inch
ETFE	Ethylene tetrafluoroethylene
FEM	Finite Element Method
GUI	Graphical User Interface
PAMELA	Parametric Membrane Load Analysis
UML	Unified Modeling Language

1

INTRODUCTION

Science is what we understand well enough to explain to a computer. Art is everything else we do.

— Donald E. Knuth

While sketching with paper and pencil and creating stocking models are practicable techniques for the form finding process of membrane structures if no software shall be used, highly specialized software is required for structural analysis and cutting pattern generation since the complex calculations cannot be done by hand within a reasonable time. If a software shall be used for form finding as well, in my opinion it is important that this software is easy to use and must be very responsive to changes. When a design idea comes up, it must be possible to visualize it in a flash before the idea flies away again.

The process step of form finding traditionally belongs to the field of work of architects, while structural analysis requires the expertise of an engineer specialized in membranes. Best results are achieved when form finding and structural analysis are combined in an iterative process to discover the ideal solution. This requires close collaboration between architects and engineers, and it is important that both use a common foundation for their work, which in turn rises the requirement to share data and drawings.

The well-known agile development method *Scrum* thrives on the principles that outstanding performance is achieved when teams are small and self-organizing units of people and when such teams are fed with objectives, not with executable tasks. [27]

How can a membrane software enable teams to work based on objectives?

The solution implemented by myself and presented in this Master Thesis is to apply the power and benefits of reusable and interchangeable logic and algorithms to the design and analysis of membrane structures - all fully integrated into one of the leading 3D CAD systems available. The different logic components can be arranged, connected and activated based on the current needs and ideas to best support the workflow of choice. Ideas in this context can be of technical or design related nature. And to further enhance collaboration and reduce errors and divergences introduced by file conversions, the full process of sketching, form finding, analysis, patterning, creating shop drawings

Scrum is a simple yet incredibly powerful set of principles and practices that helps teams deliver products in short cycles, enabling fast feedback, continual improvement, and rapid adaptation to change.[21]

as well as 3D visualizations - all these steps can be done in the same environment and based on the same drawing files.

Back to *Scrum*: Which members are needed to form a small team which develops membrane projects based on objectives? The team needs an Architect and an Engineer - or even better, both in one person as an Archineer®[9] - and a Grasshopper. Not Grasshopper the insect, but the algorithmic modeling environment for Rhino3D CAD system[19]. In addition, the team is assisted by Pamela. Ok, not Pamela as a person but rather PAMELA¹ as a software library.

PAMELA stands for *Parametric Membrane Load Analysis* and is the working name for the software library I have developed, the why and the how of this development and especially how PAMELA can be used to support the process of building beautiful membranes is the content of this document. The focus is mainly on membranes made of fabrics, but cable nets and structures covered with ETFE cushions could also be dealt with.

The reader should have a general understanding of Tensile Architecture and tasks commonly executed when designing membrane structures. Experience in working with 3D CAD software is assumed as well.

CHAPTER 2 reflects my motivation to tackle the software development project and outlines the workflows that are intended to be supported. It also includes an overview of systems available on the market and the advantages expected from the implemented solution.

CHAPTER 3 is focused on the use cases covered and key concepts and algorithms implemented. The system's design is outlined on an appropriate level.

CHAPTER 4 explains the elements provided on the graphical user interface of Grasshopper® to support in a flexible way all process steps from form finding to cutting pattern generation.

CHAPTER 5 proofs that the system is working and can be used for design and analysis of membrane structures.

CHAPTER 6 assesses the results and identifies possibilities for further development.

¹ Parametric Membrane Load Analysis

2

MEMBRANE SOFTWARE

*We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.
A programmer who subconsciously views himself as an artist
will enjoy what he does and will do it better.*

— Donald E. Knuth [11]

2.1 BACKGROUND AND PERSONAL NEEDS

From my first study I have a degree in computer science and after a few years in the profession, I later trained to become a sail maker. Currently I am self employed and am mainly building shade sails and smaller, temporary membranes.

From my work with membranes grew the need for a software which supports me in this field. Frequently I used a free version of *Formfinder V3.5.1*[7] for form finding and shade analysis. By modeling the membrane corners as *Tennect*[2], it was even possible to get appropriate force vectors. For larger projects I am working with specialized engineers as subcontractors to do structural analysis and patterning by using their own software.

One of my reasons to join the studies was to learn how membrane software work, gain experience using such software and to get a good understanding of the common functional requirements for such software to do a proper evaluation. After all membrane software are not low-priced and if one decides to invest in such a product and accompanying training, the decision should be well-founded.

In March 2017 I started the master studies in Membrane Structures at Anhalt University of Applied Sciences. *IMS*[9] students are eligible to use a student version of the membrane software *ixCube*[30] for their studies. Unfortunately this software does not run error free on my computer (Microsoft Windows 10 on Apple Mac Book Pro using Parallels Desktop V14 for virtualization) and fellow students have reported similar issues while using *ixCube* on computers running Microsoft Windows as primary operating system. Unpredictable crashes are frequent and attempting to load an existing project results in errors. Furthermore the elements of the user interface do not adopt

well to displays with high DPI¹, at least I am not able to recognize / distinguish the different icons unless I reduce the screen resolution. Reducing the screen resolution to use a (not especially cheap) CAD software is a contradiction in terms. I have decided not to invest in different computer hardware just to run this program.

Some lecturers of the study programme, especially

- Prof. Dr. Günther Filz for CM1 Architecture
- Prof. Dr. Kai-Uwe Bletzinger for CM2 Membrane Programs / Numerical Theorie
- Dr. Gregor Grunwald for CM3 Structural Design and Detail
- Prof. Dr. Lars Schiemann for CM5 Structural Design Concepts (Dimensioning)
- Dr. Switbert Greiner and Dipl.-Ing. Alfred Rein for OM4 Foldable and Umbrellas

encouraged us to use generative algorithms to explore new shapes for our assignments. They in particular mentioned Grasshopper[®] [19], a graphical algorithm editor tightly integrated with Rhino's 3-D modeling tools.

Peter Novýsedlák held his Master Thesis presentation *Tensile Structures – Numerical Design Techniques* [15] at the *IMS Institute* [9] in March 2017. Afterwards we got into conversation and the idea arose that I could take Peter's work as a foundation stone to develop a software for parametric design of membranes based on Grasshopper[®]. First just to do some simple form finding, but later extendable also to do structural analysis and patterning. Both of the latter are now included as well, to a certain extent.

This project was started out of personal interests and needs. I saw it as a challenge to implement it and last but not least it is about the learning experience. Commercial aspects were not in the foreground. A lot of time was invested and more than 12'000 lines of source code came together.

2.2 MARKET OVERVIEW

Over the past years I have come across several commercially or even free membrane software. I usually had a look at them, tried to get a demo license and did some evaluation testing. Since this mostly happened before I was even thinking of ever writing a master thesis on this topic, the trials and examinations were not done in a very scientific way but rather driven by the projects I was working on at this time.

¹ Dots Per Inch

In this sense, the following summary should be understood as a non-exhaustive overview of membrane software currently on the market accompanied with some thoughts and insights dated to the time when I used the software. These findings can be 5 years old and facts may have changed in the meantime. The list is in alphabetical order.

- Carat++ [23]: A finite element program for simulation, structural optimization and form finding developed at TU Munich. Carat++ is used as the backend for Kiwi3d (see below).
- COMPAS[26]: Just very recently discovered open-source, Python-based computational framework for collaboration and research in architecture, engineering and digital fabrication.
- Dlubal [5]: Dlubal Software is well established on the market for structural engineering software. Recent addition to support membranes.
- Easy [25]: Has been on the market for ages. Software is divided into several modules which interact based on input and output files. This allows for some flexibility to customize the workflows.
- ExactFlat [6]: 3D to 2D digital patterning and nesting.
- Formfinder [7]: Well known software for form finding. Underlying algorithms are based on EASY (see above).
- ixCube [30]: See comments in 2.1
- K3 Tent [10]: Just recently discovered, company based in Russia.
- Kangaroo [17]: Kangaroo is an interactive physics/constraint solver and Grasshopper plugin for designers.
- Karamba [18]: Karamba3D is a parametric structural engineering tool which provides accurate analysis of spatial trusses, frames and shells.
- Kiwi3D [24]: A recent development based on Rhino / Grasshopper. Meshfree isogeometric FE analysis. Uses Carat++ as solver.
- membranes24 [14]: Once seemed as an interesting project to me, but website is no longer available, latest activity on Twitter dates back to 2013.
- Membranedetail [13]: Grasshopper scripts for detailing of membranes. Good option to integrate with PAMELA.
- MPanel [12]: Based on Rhino, set of products targeting different customer segments (I like the approach).

- NDN Membrane [1]: Full FEM solver targeted on membrane structures. Includes patterning etc. Looked very promising, but when I wanted to trial, the supported operating systems were already outdated. Now website seems to be hacked or offline.
- RhinoMembrane [4]: Grasshopper plugin to interface with ix-Cube (see above).
- WinTess [29]: Just recently discovered, company based in Turkey, documentation found on web used as inspiration to implement wind cp values.

In addition to commercially or freely available software some companies have developed their own solutions.

2.2.1 Software Evaluation Criteria

As mentioned in 2.1, no detailed evaluation of a software was done. In order to evaluate a software, various criteria would have to be considered. The most important ones are:

- Functionality required / offered and how the current / envisioned workflows are supported
- Interfaces / compatibility with other software in use
- Pricing / total cost of ownership (especially initial investment for hardware and software, recurring license fees, training of employees, support and maintenance)
- Quality / references
- Hardware and software requirements
- Support and documentation (availability, languages, costs)
- Options and costs for trial. At the end, the users must be comfortable using the software.

2.3 REQUIREMENTS FOR A PARAMETRIC MEMBRANE SOFTWARE

As described in 2.1 the development was started as a personal challenge and because I think it's an excellent topic for a master thesis in the field of membrane structures. There has been no written requirements specification prior to starting the development. This section summarizes the requirements for a membrane software based on my own requirements and activities in connection with the implementation of membrane projects as a self employed owner of a small

company. Other, especially larger companies may have different requirements.

My current membrane projects are all based on fabrics, therefore ETFE is not in the focus. But future use of ETFE should not be prevented.

2.3.1 Integrated System

Interfaces between different computer programs are always a possible source for errors and incompatibilities. In addition, manual steps are often necessary to export the data from one program and import them into the next program with the danger of using the wrong version of the file. Personally, I prefer a system in which all necessary tasks can be solved rather than a combination of several specialized applications.

Based on this I expect from a membrane software at least to support the steps of the whole design process as described in the following sections. No switching between different applications shall be required.

2.3.2 Form Finding

Form Finding must support two modes, sketching and accurate form finding:

- For sketching a quick and easy way to define a membrane outline which is instantly form found / relaxed is needed. Sketching is an iterative process, it must be possible to take advantage of the parametric environment and show how the form varies when parameters are changed (live preview).
- Accurate form finding is needed for subsequent structural analysis and patterning.

From my daily work, the most important evaluations after form finding are:

- Shade analysis
- Water drainage and ponding analysis

2.3.3 Structural Analysis

Common functionality of software used for structural analysis is also relevant for membrane structures. Among others this includes:

- comfortable definition of load cases (especially wind and snow)
- evaluation of stresses
- evaluation of displacements for the membrane under load conditions

2.3.4 Patterning

Cutting pattern generation should be based on geodesic lines which will lead to most economic material use. An adequate 3D to 2D flattening mechanism must deliver better results than standard Rhino command *Squash*.

Assessment criteria for the accuracy of the flattening process include:

- change of surface area (ideally 0.0)
- change of edge length for each edge (ideally 0.0)

2.3.5 Detailing / Shop Drawings

For the detailed design it should be possible to automatically generate some standard elements like ear plates, clamping etc. based on the current model data (dimensions, force vectors) and specific parameters.

2.3.6 Visualization

The FEM model used for analysis is based on work point and center-line geometry. When cross section and material information is applied to the different members it should be possible to automatically generate basic 3D visualizations.

The visualizations can not only be used to check the model against colliding members but also as a sales instrument. Therefore it's necessary that visualizations can be generated at each stage in the workflow, already starting with sketching.

2.3.7 Bill of Material / Cost Estimation

From the FEM model that was used for analysis it should be possible to derive an initial bill of material which can be used for rough cost estimations.

3

CONCEPTS AND IMPLEMENTATION

The psychological profiling [of a programmer] is mostly the ability to shift levels of abstraction, from low level to high level. To see something in the small and to see something in the large.

— Donald E. Knuth

3.1 INTRODUCTION

My motivation to write the PAMELA software library and a broad overview of commercially available membrane software are presented in chapter 2. The focus of this chapter is on the development of my own software library, the architecture of the system, the methodologies and key concepts applied and the algorithms implemented.

The focus is on the use cases for membrane design and visualization of the results from structural analysis, aspects of software engineering are treated and documented only on a very high level.

3.2 THE GRASSHOPPER[®] PLUGIN

Grasshopper[®] is a visual programming language and environment that runs within the Rhinoceros 3D CAD application. Programs are created by dragging *components* onto a canvas. The outputs of these *components* are then connected to the inputs of subsequent *components*. [28].

Figure 3.1 shows the Grasshopper[®] main window. Standard *components* are usually dedicated to a very specific functionality which makes the application very versatile but on the other hand quickly leads to large number of *components* on the canvas. A large number of *components* results in many connecting wires and it's easy to lose the overview. Grasshopper[®] documents are often referred to as *Spaghetti*. The key concepts developed and implemented to avoid "*Spaghetti*" as good as possible are described in section 3.6.

From within a Grasshopper[®] definition it's possible to reference items of an existing Rhino drawing (lines, points etc.) as well as previewing resulting geometries from Grasshopper[®] directly in the Rhino drawing. As long as geometries are in preview mode, they can be changed without affecting the real drawing. With the *bake* command it's finally possible to create the previewed geometries in the drawing.

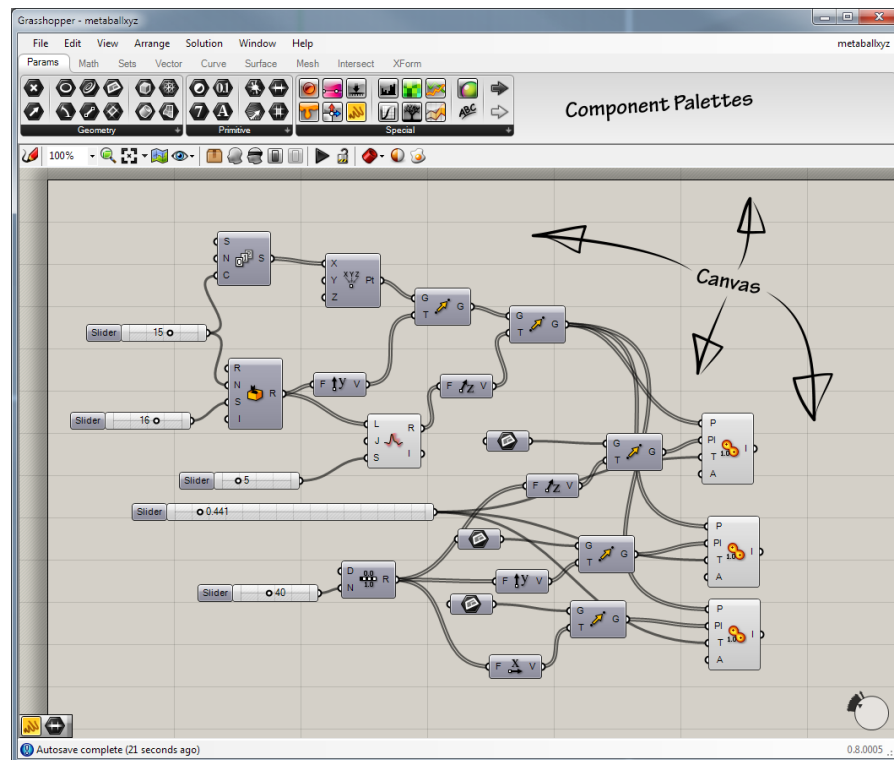


Figure 3.1: Grasshopper[®] main window (source: www.wikipedia.org)

3.3 RUNTIME ENVIRONMENT

Figure 3.2 shows the different layers and technologies Rhino is built of. All .NET plugins that ship with Rhino for Windows and Rhino for Mac, including the Python interpreter, reference *RhinoCommon*. The layer *RhinoCommon* represents the API¹ that can be accessed by 3rd party plugins to use native Rhino functionality. 3rd party plugins reside in the layer *.NET Plugins*.

As programming languages, especially C# and Python are supported. I have chosen to use C# and not Python as programming language for several reasons:

- Python is an interpreted language, from a precompiled language like C# better performance can be expected.
- C# is closely related to the programming languages C++ and also Java, which I both know from past projects.
- For Python different distributions (flavors) exist. 3rd party libraries sometimes support only certain flavors and might not be guaranteed to be compatible with Grasshopper. As an example, Matlab offers a Python API, but this API is not compatible with the Python supported by Rhino Grasshopper.

¹ Application Programming Interface

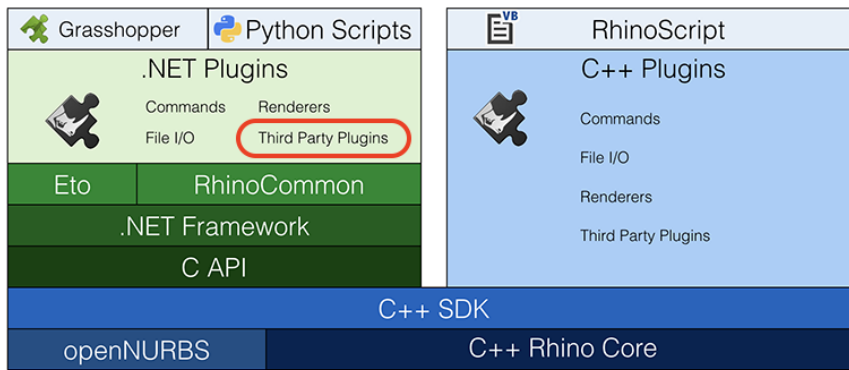


Figure 3.2: Rhino technology overview (source: developer.rhino3d.com)

When using *RhinoCommon* as the API to interface Rhino and Grasshopper[®], chances are good that the custom plugin developed is compatible with Rhino on Windows as well as Rhino on OS X (Apple Macintosh), although executables for each operating system need to be compiled individually. Unfortunately the development and release schedule for Rhino for Windows and Rhino for Macintosh are not synchronized. On Windows, the most current release is Rhino 6. On OS X, the current version is 5.

Special care needs to be taken when implementing GUI² extensions. It's recommended to use the Eto framework for this kind of extensions. On OS X, Eto has been supported from the beginning and therefore is included in the current release (5). On Windows, Eto support starts with Rhino release 6.

Current development is based on and compiled against Rhino 5 for Windows and uses GUI extensions as little as possible, mainly only for popup menus.

3.4 MODULE ARCHITECTURE

One of the goals of the development is to keep things as simple and universal as possible. For the deployment of PAMELA currently only one file called Pamela.gha needs to be copied into the libraries folder of Grasshopper[®] and the functionality can be used.

Internally, a layered approach has been applied. Figure 3.3 shows the internal structure of the library. All the core algorithms were implemented without any dependencies to the representation layer to make them as universal as possible. The core algorithms only rely on RhinoCommon and other standard libraries. All components which depend on the Grasshopper user interface are placed in the layer

² Graphical User Interface

Grasshopper Components. This layer depends on the core algorithm, RhinoCommon and 3rd party libraries as well.

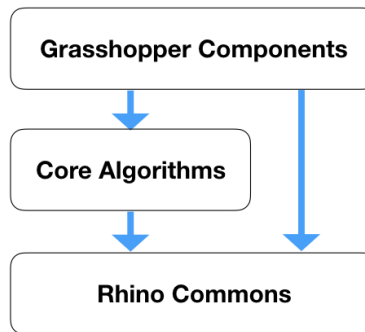


Figure 3.3: internal structure

3.5 TOOLS AND DEVELOPMENT ENVIRONMENT

For the development of the PAMELA software library the following tools are used:

- Visual Studio 2017 - development environment for C#. Programming, compiling, debugging, buliding the library.
- GIT - revision control system for source code as well as LaTeX documentation
- Visual Studio Code on OS X - LaTeX Workshop extension as text processor for documentation (especially this Thesis)
- Trello - web based project management tool used to track issues (errors) as well as ideas / wish list for future development.

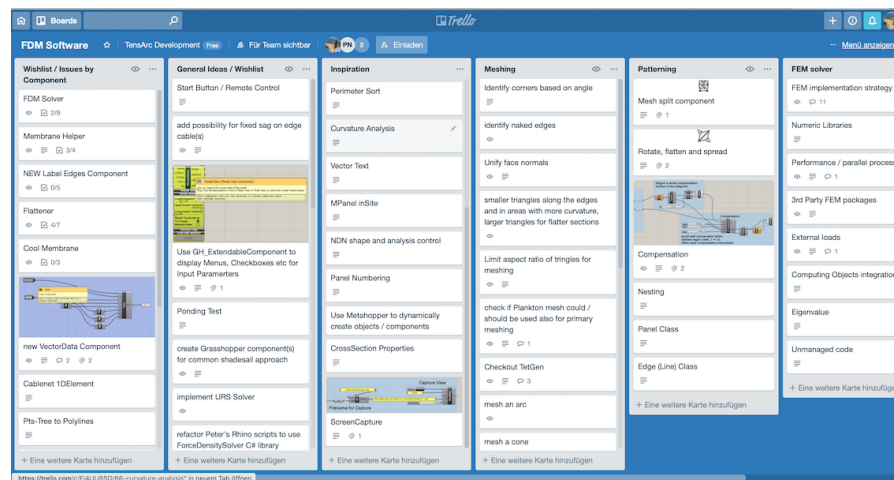


Figure 3.4: Trello project management tool

Figure 3.4 shows a screenshot of the project management tool *Trello*. For the project a separate board was created with lists to group the issues. Within the lists separate cards are created for each Grasshopper *components* or topic.

3.6 BEHAVIORAL KEY CONCEPTS

The usability of a software is the controlled aspect of user experience design that ensures the end-user doesn't strain or encounter problems with the use of a product or website's user interface. The first goal of usability is efficiency and effectiveness while aesthetic value comes after a product has proven to be usable. [3]

Figure 3.5 shows the properties affecting software usability.

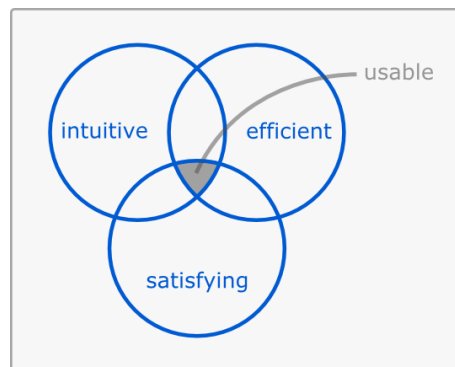


Figure 3.5: software usability(source: blog.rocketsoftware.com)

If a Grasshopper[®] document is perceived as *Spaghetti* (see section 3.2), this obviously indicates that usability has to be classified as inadequate.

Great care has been taken to increase the usability of PAMELA components mainly by introducing *Smart Component Groups* and extensive use of *Message Bubbles*. The two concepts are described in more detail below.

3.6.1 Smart Component Groups

Figure 3.6 shows the Ridge/Valley Cable component as an example of a Smart Component Group. Please note, that all items visible in the picture have been instantiated and placed by the user with a single click.

The main aspects of smart component groups are:

- In addition to the component incorporating the main functionality (algorithm), a set of input parameters and additional components is instantiated automatically as well. This significantly reduces the number of required clicks to build a solution.

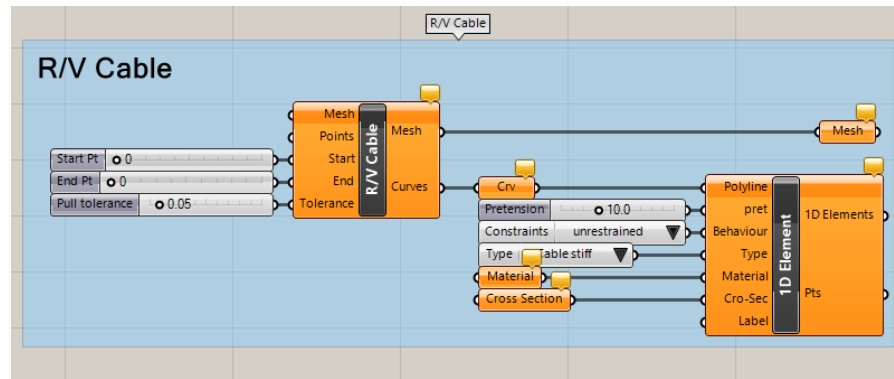


Figure 3.6: Ridge/Valley cable component as example for smart component group

- A tidy look is achieved by automatically placing and aligning all related components.
- Geometry preview is turned on or off for each component or parameter based on the context it is used.
- The components are automatically grouped and the group is labelled according to the main component of the group using increased font size. This allows to keep a better overview in a large definition. The labels stay recognizable even when the document zoom is reduced.
- When placement of the group is changed on the document canvas (drag and drop of blue area), all components are moved with the group.
- The input parameters are automatically configured to contain meaningful default values and settings.
- Common functionality is reused. As shown by this example, a ridge or valley cable will always require settings common to any cables in a valid model, the components of a 1D Element (see 4.5.1) are automatically included.

3.6.1.1

Example default values: The Grasshopper[®] default settings for a number slider are a range from 0.0 to 1.0 with 3 decimal places and an initial value of 0.250. For the Ridge/Valley Cable component the number sliders to select start and endpoint have been configured to represent integer values > 0 and the maximum value is set according to the length of the list of supplied points. As default, the first point in the list is selected. RhinoCommon uses 0 based indexing, therefore the first position corresponds with index 0.

3.6.1.2 Advantages

By using Smart Component Groups as opposed to implementing custom GUI extensions the full flexibility and modularity of the Grasshopper definition remains preserved. The user can individually replace or modify any component or parameter according to his / her needs. Finally, it is this modularity that makes Grasshopper so powerful and universally applicable.

3.6.2 Message Bubbles

Grasshopper[®] components allow to display a custom message in a message bubble located at the bottom end of the component. This functionality is hardly used by any standard component. The message bubbles only appear at a certain zoom level. When zooming out to get a better overview of the whole definition, the message bubbles disappear automatically.

Message bubbles are used extensively within PAMELA software package to give further indications regarding the current state of the object by means of:

- the current number of objects contained
- is the object valid or not
- progress information, especially for *Solvers* (see 4.7)
- tolerance required / achieved
- values of selected properties
- etc.

Figure 3.7 shows the Model Builder component (see 4.5.4) and its message bubble after placement. Since no inputs have been provided, the assembled model does not contain any elements (all counts are 0) and as a consequence the model is neither ready for Force Density Method nor for Finite Element Method processing.

3.6.3 Appearance of Smart Components

Figure 3.8 shows an overview of a detail of a Grasshopper definition with reduced zoom factor. When zooming out, the message bubbles automatically disappear. The text fields used to label the groups on the other hand remain clearly visible. This behavior greatly increases readability of a PAMELA definition.

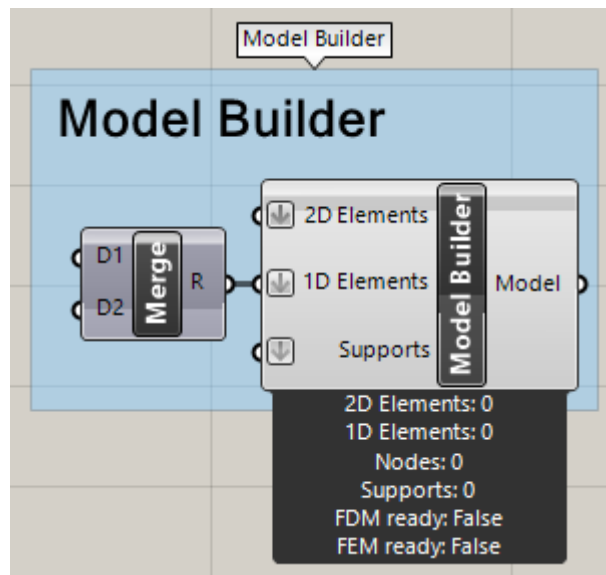


Figure 3.7: Model Builder component with message bubble

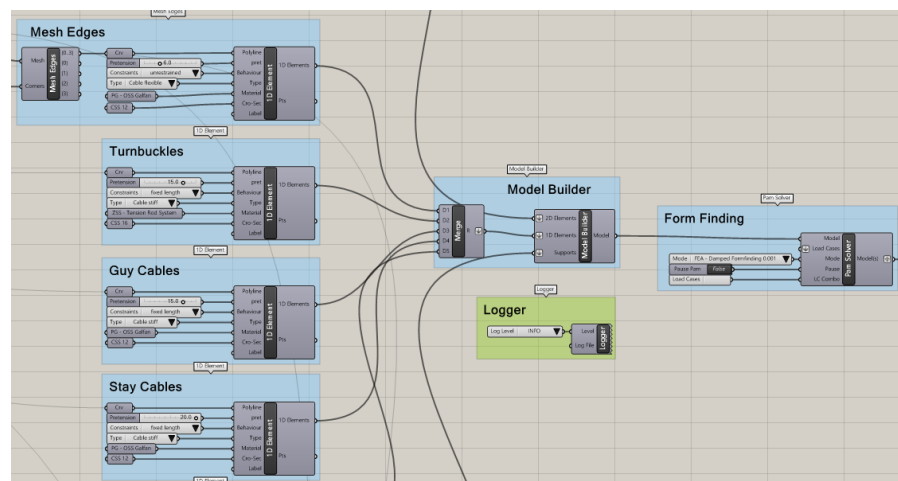


Figure 3.8: appearance of smart component groups

3.6.4 Component State

Figure 3.9 shows how components are colored according to their state:

- Components which have all necessary data to calculate a correct result are shown *grey*.
- Components are drawn *orange* when at least one required input parameter failed to collect data or incorrect or contradictory data is provided and execution is prevented.
- If the input data leads to an incorrect result, the component is colored red.

This concept is common and supported by Grasshopper, but it is by no means enforced by default. Therefore I have taken great care to ensure all components are always displayed according to their state.



Figure 3.9: colors indicating the components state

3.6.5 Unified Icons

A picture is worth more than a thousand words - a well-known saying that is implemented in computer programs by using *Icons* to increase the usability. A good icon must be easily recognizable and therefore I paid attention to unify the appearance of the different icons as good as possible. This was not always easy since the typical icon is based on a bitmap with the size 24x24 dots and vector graphics are currently not supported.

Figure 3.10 shows some of the icons developed. In chapter 4 all icons are shown as part of the components description.



Figure 3.10: examples of icons

3.7 MAIN WORKFLOWS

The generic behavior of a computer program can be reduced to a processing step which takes some input and creates some kind of output. For this to work, either the processing needs to be adopted to the input or the input must be structured according to the requirements from processing. Figure 3.11 shows the generic behavior of a software.



Figure 3.11: generic behavior of computer programs

In the case of PAMELA one of the main processing steps is *Form Finding* of the membrane or cable net. Form finding is executed as

part of the *Sketching* workflow as well as of the *Engineering* workflow. Both workflows are described in more detail in the next sections.

3.7.1 Workflow for Sketching

Sketching is the Artist's (or Architect's) tool to define the overall visual appearance of the membrane structure, especially the membrane itself. A quick and easy way to define a membrane outline which is instantly form found / relaxed is needed. Formfinder [7] works well for this and I used the behavior for inspiration. The form found shape must not be 100% accurate, specific material properties can be ignored but prestress ratio must be considered. Sketching is an iterative process. It's important to get a result quickly and to be able to judge the appearance and easily adjust it. Figure 3.12 shows the main steps of the sketching workflow.



Figure 3.12: Sketching workflow

Several methods have been implemented to accomplish the task of form finding, they are described in 3.10 in more detail. The Force Density Method Solver (see 4.46) is ideally suited for sketching.

3.7.2 Workflow for Engineering

The Engineering workflow is based on the output of the sketching workflow. It actually incorporates the steps of the sketching workflow into the *Model Definition* step. The visual appearance of the resulting shape is the main assessment criteria in the sketching workflow. In the engineering workflow, this review must cover additional aspects and criteria, including:

- water drainage and ponding
- curvature analysis
- aerodynamic behaviour
- air ventilation
- shade analysis
- deflection / displacement when loads are applied

More details are added to the model to complete the specification which then can be used to do proper form finding, structural analysis

and to use the output for patterning and other production related steps. Figure 3.13 shows the main steps of the engineering workflow.

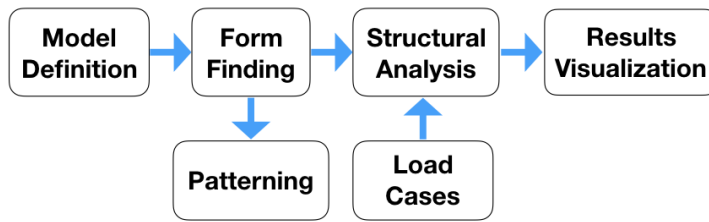


Figure 3.13: Engineering workflow

The workflow starts with the definition of the model to assemble the input data. Once *Form Finding* is executed the next step is *Structural Analysis* considering the different *Load Case* definitions. The *Results Visualization* is mainly about visualizing stresses and forces, displacements, deflections and resulting shapes. As input for *Patterning*, again the result of *Form Finding* is used.

ADAPTABILITY As already mentioned in chapter 1, the different logic components implemented can be arranged, connected and activated based on the current needs and ideas to best support the workflow of choice. For a cable net structure for example patterning is not required and there is no need to analyze ponding behavior since this is usually not an issue.

3.8 MODEL DEFINITION

In the case of PAMELA the key algorithm used for form finding as well as structural analysis is the implementation of the Finite Element Method (FEM³) as described in detail by Peter Novýsedlák in his Master Thesis [15]. The data assembled in the model definition must suit the needs of this algorithm.

3.8.1 High Level Abstraction

On a very high level of abstraction the input data required for FEM can be split into

- nodes (work points)
- system lines (centerline geometry)
- triangular polylines (mesh faces)
- boundary conditions (behavioral restrictions)

³ Finite Element Method

- internal forces (prestress)
- external forces (loads)

3.8.2 Conceptual Elements

This section lists the main conceptual elements used to define a model. A complete overview and further explanations regarding all components implemented is presented in chapter 4, and some use cases and scenarios are presented in chapter 5.

The main elements required to model a membrane structure:

- The membrane itself is treated as so called 2D Element since the fabric is usually very thin compared to length and width and the thickness is given by the fabric. Membranes are drawn by means of meshes, triangular meshes in particular. A detailed description can be found in 4.5.2.
- Cables, trusses and beams are modeled as 1D Elements having a start and an endpoint corresponding with the centerline geometry. 1D Elements are drawn by lines, polylines are automatically split into segments. A detailed description can be found in 4.5.1.
- Fixation points are modeled as Constraints, called Supports. If as an example a membrane is fixed by a Keder profile, a polyline can be used to define all mesh vertices used as fixation points. A detailed description can be found in 4.5.3.

Figure 3.14 shows the main elements required as listed above.

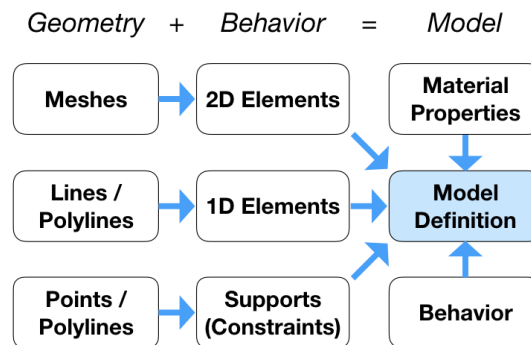


Figure 3.14: model definition

As an example figure 3.15 shows a geometry where all the main elements are labelled according to their type.

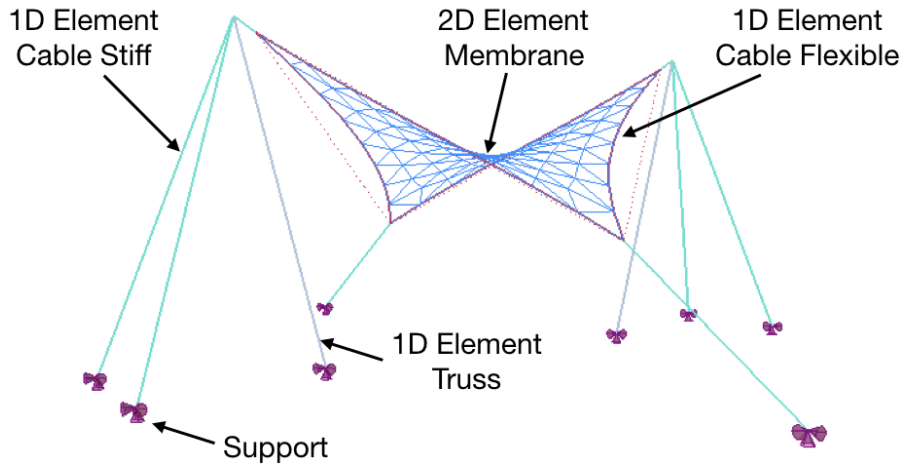


Figure 3.15: elements of a geometry

ASSEMBLING THE MODEL The *ModelBuilder Component* (see 4.5.4) is used to correctly assemble and validate the model. For instance duplicate definitions of supports or other elements are ignored / unified.

Figure 3.16 shows the Model Builder Component in use.

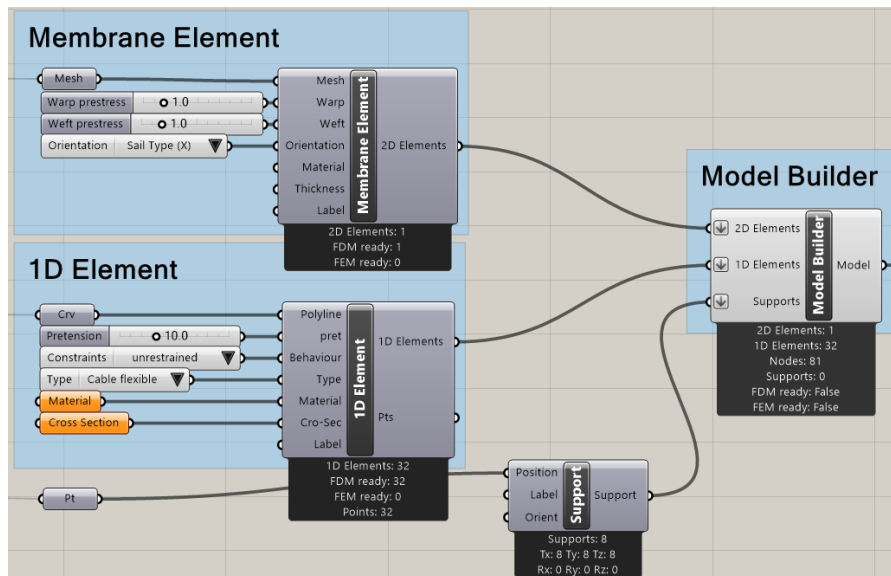


Figure 3.16: Model Builder Component in use

3.9 MESH DEFINITION AND HANDLING

The mesh used to model the membrane element plays a very central role in the implementation. Mesh creation must be easy, intuitive and fast to support sketching (see 3.7.1), on the other hand is the geometry of the mesh directly used for form finding, structural analysis and

patterning. Opposed to some other membrane software which create intermediate surfaces based on the mesh to do patterning, in PAMELA the original mesh is directly used for patterning and flattening as well.

3.9.1 Mesh Creation

The algorithms of PAMELA are based on triangular meshes. Quadrangular meshes are currently not supported, but every quadrangular mesh can easily be transformed into a triangular mesh by spitting the quadrangular mesh faces along one of the diagonals. Splitting a face along the shorter diagonal usually leads to triangular mesh faces with better aspect ratios than if the long diagonal is used for splitting. Therefore, splitting along the short diagonal is preferred. The triangular meshes do not necessarily need to consist of faces of regular size or aspect ratio but (almost) equilateral mesh faces are advantageous. The process of creating a mesh for three dimensional membranes can become a challenge itself and Grasshopper offers only very limited support. The built in functions work best on flat surfaces. Not really the case for three dimensional membranes. Therefore distinguished meshing components have been developed:

- *Edge Mesher*: This component can be used to create a mesh for a membrane characterized by a single closed edge line, like traditional hypars etc. The membrane may not contain any holes. This mesher works directly based on the supplied edge curve, no intermediate surface is required for the meshing process. For more details on this component see section 4.3.3
- *Brep Mesher*: When a membrane contains one or more holes (like typical cone structures) an intermediate surface needs to be created first. This surface then can be used as input for the so called Brep Mesher. BREP stands for Boundary Representation and is a term commonly used in CAD programs. For more details on this component see section 4.3.2

3.9.2 Adoption of Mesh Size

Modern desktop and laptop computers have CPU's with multiple cores which allow parallel execution of multiple sub-tasks. This is called multi-threading and thus, the computers are very powerful. But multi-threading must be supported by the software used. The Grasshopper plugin of Rhino 5 does not support or allow for any multi-threading from within any code directly executed by a plugin. Rudimentary support for multi-threading has been implemented in Grasshopper shipping with Rhino version 6. The current development of PAMELA is based on Rhino version 5 and therefore no multi-threading is sup-

ported and performance can become an issue for more complex structures when many iterations are needed to approximate a solution.

The performance requirements of FEM based computations increase exponentially with the size of the FEM model: Each node is represented by 6 DOF⁴. The stiffness matrix which is one of the core elements of the FEM algorithm in turn consists of one row and one column for each DOF. A problem statement with 10 nodes leads to a stiffness matrix of the size 60 x 60 and therefore consists of 3'600 values. For a problem statement with 100 nodes the stiffness matrix already contains 360'000 values.

Finer meshes definitely lead to more detailed and more accurate results. But the main characteristics and behavior of a solution like stress distribution and maximum stresses remain the same. My tests based on various geometries like hypars have shown that the relevant maximum values for stresses for example differ only minimal between solutions with coarser or finer mesh for the same geometrical shape.

For sketching (see 3.7.1) a coarse mesh delivers the needed responsiveness with sufficient results accuracy.

3.9.3 Mesh Granularity – Impact on Accuracy of Results

The results shown in table 3.1 are based on the same geometry and load cases but with different meshing parameters applied to the membrane.

This example shows that a coarser mesh can be used to do preliminary dimensioning of a structure while the main characteristics remain preserved. But the computation time is reduced massively. Breaking down the computation time to the number of faces:

- coarser mesh: 4.5s / 143 faces = 0.03 s/face
- finer mesh: 339s / 684 faces = 0.5 s/face

Especially in a parametric environment like Rhino Grasshopper where solutions are changed frequently to iteratively determine the best solution, performance and responsiveness are key factors.

3.10 FORM FINDING

Form finding is a process to find the equilibrium state of a cable-membrane structure at a given stress level and with specified boundary conditions[15]. Form finding is used in the sketching workflow as well as in the engineering workflow. Two methods have been implemented to accomplish the task of form finding, the algorithms are based on the work of Peter Novýsedlák[15]:

4 Degree of Freedom

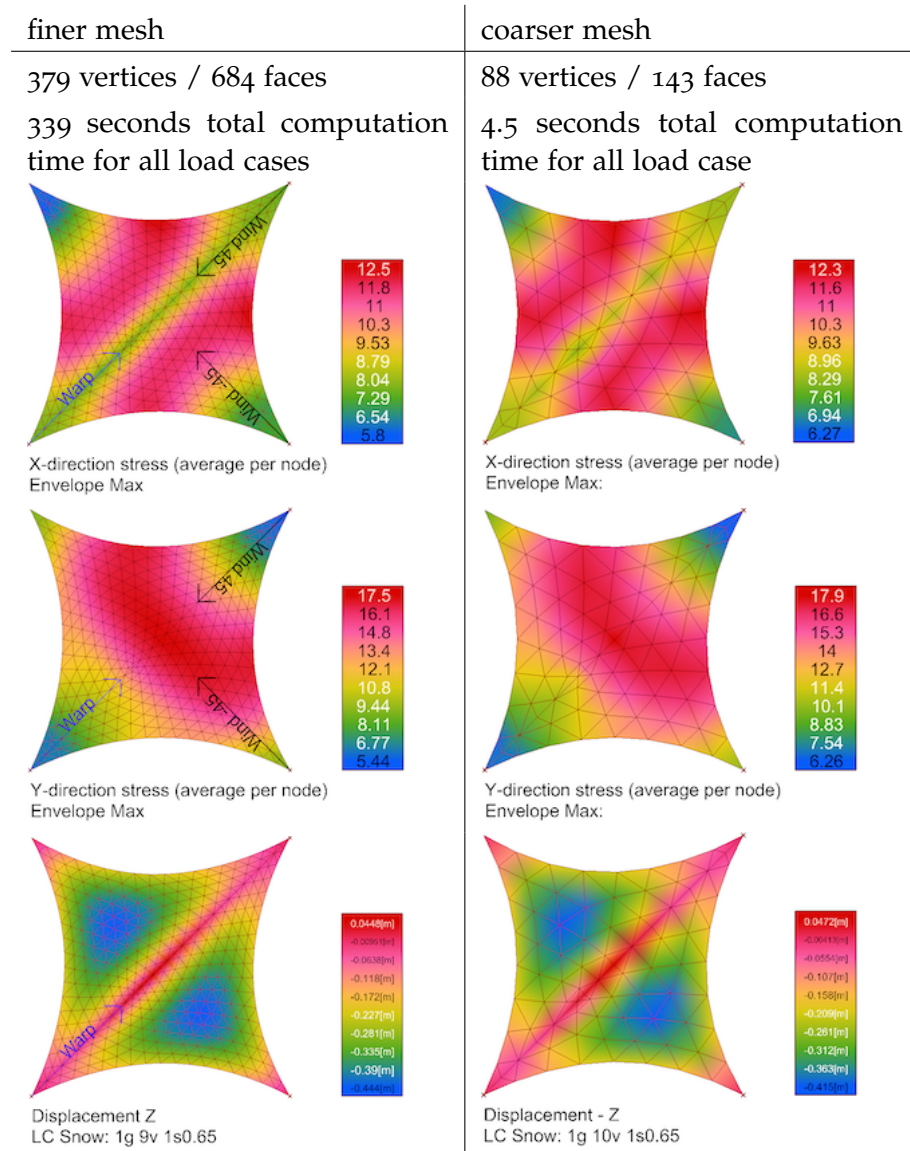


Table 3.1: Results comparison for different mesh sizes

- The Force Density Method Solver (see 4.46) is a good choice if only a membrane and its edge, ridge or valley cables and their prestress ratio are considered. This solver needs only one iteration to find a valid solution and is therefore very quick and ideally suited for sketching.
- The Finite Element Method Solver (see 4.47) allows to select the desired level of accuracy and therefore usually needs multiple iterations to determine the solution. Furthermore it requires material properties to be specified and therefore it is more common to use it in the engineering workflow and not in the sketching workflow.

For best accuracy, form finding must consider self-weight and prestress and therefore also material properties and warp orientation of the fabric.

3.10.1 Warp Orientation

For membrane materials with an anisotropic behavior, which implies having different mechanical or physical properties when measured in the different axes, the warp orientation must be considered in the engineering workflow and as a result also in the whole production process.

In the Finite Element Method the orientation of each mesh face (triangle) regarding warp direction must be known to properly determine its behavior. The implemented and for membrane software very common approach to determine warp direction for each mesh face is to define warp by means of a single orientation vector for the whole membrane. This vector is then used to create a plane to project the membrane onto and from this projection, the warp orientation is determined for all areas of the membrane. For cone type structures, usually the Z-axis is used as warp orientation to emulate warp running from bottom boundary to top boundary of the membrane.

3.10.2 Limitations of Projecting Warp Direction

The approach described regarding warp orientation works especially for rather flat membranes. How about twisted structures as shown in figure 3.17? The red line denotes the warp direction and when projected to a plane, the projected direction changes considerably, a uniform direction cannot be determined. If this structure is form found with unequal prestress for warp and weft and a traditional approach of projecting a uniform warp direction is used, the result can not be accurate.

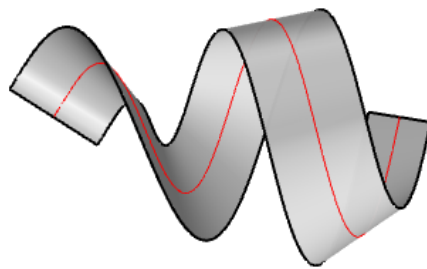


Figure 3.17: warp on twisted membrane

Another aspect most membrane software I trialed seem to ignore at the stage of form finding is the seam layout. To avoid wrinkles along the seams, the seams need to be aligned in a way that the

angle between the cutting line and warp are (almost) the same on both adjacent panels. To optimize this, it must be possible to define warp orientation on a per panel basis, especially if the prestress ratio between warp and weft shall be different than 1.0. Therefore, not considering the panels and seam layout at this step leads to less accurate results.

The two aspects mentioned in this section require further research and investigations. Most likely this topic could be used as subject for a Masters-Thesis by itself.

3.10.3 Considering Corner Connection Details for Form Finding

Often corner plates are ignored during form finding and added later. As a result, the membrane is modelled with pointy corners and cut-backs for corner plates are defined later. This implies, that each corner plate is specified and manufactured individually according to the angle of the membrane. Especially for small or medium sized membranes, it can be more economic to create a set of standard corner plates and reuse them in multiple projects.

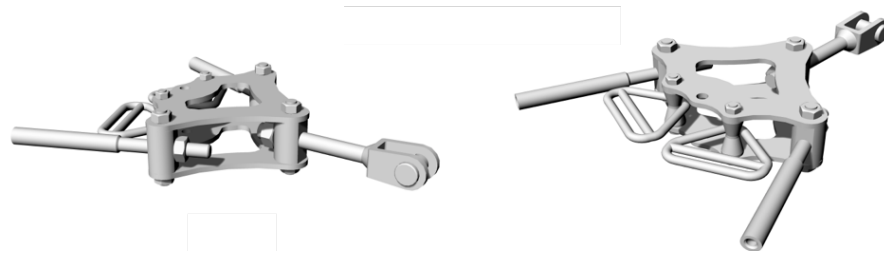


Figure 3.18: generic corner plate

Figure 3.18 shows the generic cornerplate for smaller membranes I developed in cooperation with the danish company BlueWave, a manufacturer of sailboat hardware. Working with standardized corner plates requires that their dimensions and mechanical behavior are already included in the overall model definition used for form finding since the corners of the mesh are not pointy any more. This complicates the mesh generation and especially referencing of cornerpoints (now there are two points for each physical corner). The basic functionality to add such a corner plate is implemented by the Membrane Pimper Component (see 4.10.1). Since it's functionality is rather on a prototyping level it has been excluded from the examples shown in this document.

3.11 LOAD CASES

For structural analysis (3.12) the same FEM algorithm is used as for form finding, but executed with different parameters and especially applying external loads.

The following loads are supported:

- Self-weight (automatically calculated based on material properties and cross section specified)
- Prestress (warp / weft)
- Wind (external) pressure, direction, for cp values see 3.11.1
- Snow load, for roof shape coefficients see 3.11.2
- Internal pressure (important for ETFE cushions)
- Point loads, closes point of the model will be selected

An additional factor can be specified for each load. Several components have been developed to make it easier to select wind and snow loads in general or based on Swiss or German codes. Section 4.6 gives an overview of components that can be used to define load cases.

3.11.1 Wind Pressure Coefficients

Correct determination of wind pressure coefficients (cp values) for membrane structures is very delicate. In the national codes exist no detailed guidelines which are applicable to membrane structures of any shape. The most reliable way to determine aerodynamic behavior of a membrane structure is by means of wind tunnel testing. But usually this is not applicable and too expensive. A common approach is to let the user assign different cp values to selected areas of the membrane.

In PAMELA a more versatile and comfortable approach has been implemented as standard: For each mesh face the orientation is evaluated based on the angle between the surface normal and the defined wind direction. Cp values are assigned according to the angles in steps of 10 degrees. Positive values are used for wind pressure, negative values for wind suction.

The wind cp determination is implemented as *Strategy Pattern*[8] and therefore is exchangeable by definition. The default values can also be changed manually if other values shall be used.

Table 3.2 lists the cp values used by default.

angle	cp	angle	cp	angle	cp
0	-0.4	70	-1.0	130	0.6
10	-0.4	80	-0.9	140	0.8
20	-0.6	90	-0.8	150	0.8
30	-0.8	100	-0.2	160	0.8
40	-0.8	110	0.2	170	0.8
50	-0.8	120	0.4	180	0.8
60	-0.9				

Table 3.2: default cp values

Working with different cp values results in different external loads on the membrane which in turn affects the resulting membrane stresses and shape. This needs to be considered when comparing the results of different membrane software.

3.11.2 Snow Roof Shape Coefficients

For snow loads an algorithm similar to determining wind pressure coefficients has been implemented to determine the roof shape coefficients based on the pitch (0 degrees means horizontal). Table 3.3 lists the roof shape coefficients used by default.

pitch	coefficient
0 to 30 degrees	1.0 (no snow drift)
30 to 60 degrees	0.8 to 0.0, interpolated
60 to 90 degrees	0.0 (total snow drift)

Table 3.3: default roof shape coefficients

The value of 1.0. for the range from 0 to 30 degrees is according to the Swiss code SIA 261[20] which foresees a value in the range from 0.8 to 1.0. The German code foresees a value of 0.8. I prefer to use 1.0 since in the flat areas not only snow drift but also snow deposition from steeper sections can appear.

The determination of roof shape coefficients is implemented as *Strategy Pattern*[8] and therefore is exchangeable by definition. The default values can also be changed manually if other values shall be used (see 4.6.7).

3.12 STRUCTURAL ANALYSIS

For structural analysis the material properties are taken into account as well as load cases. Each load case (reflecting a combination of different loads) is treated individually, a dedicated copy of the model is created to assign the results of the FEM computations.

Figure 3.19 shows the structural analysis in its context.

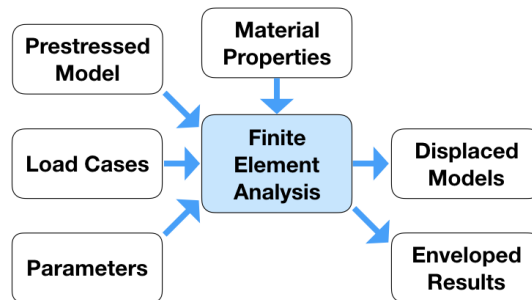


Figure 3.19: context of structural analysis

3.12.1 Finite Element Method (FEM)

The Finite Element Method is described in detail by [15]. Figure 3.20 gives a broad overview of the algorithm implemented in PAMELA. The loop is executed until either one of the following, user defined criteria is met:

- the precision is < than the required precision
- the maximum number of iterations is reached

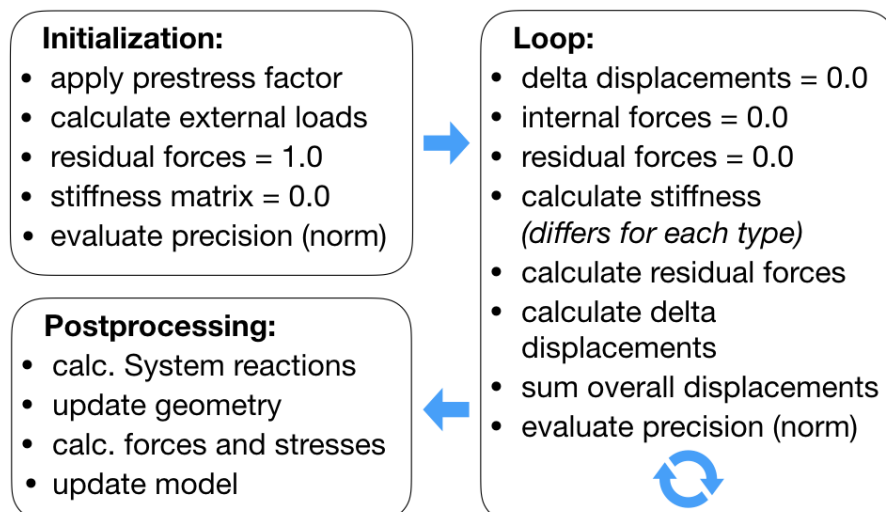


Figure 3.20: FEM algorithm

OBJECT ORIENTED PROGRAMMING Since the used programming language C# is an object oriented programming language the corresponding concepts can be used and implemented. Figure 3.21 shows the UML⁵ diagram for the FEM datamodel implemented. The FEM Datamodel contains a list of FEM Elements. FEM Element is implemented as an abstract class and derived classes are forced to implement the ComputeStiffness() method. This allows each type of fem object to transparently supply an individual implementations of the ComputeStiffness() method use in the FEM algorithm.

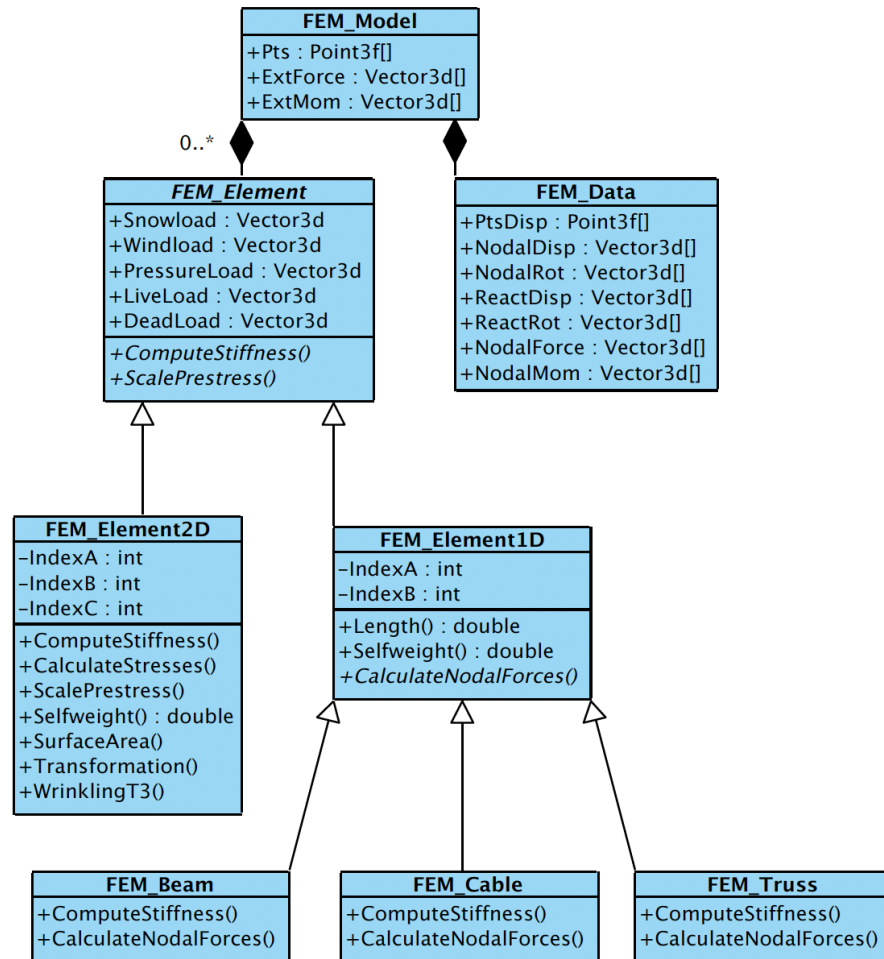


Figure 3.21: Classdiagram for FEM datamodel

3.12.2 Wrinkling Check

Membranes and cables can only take tensile forces and no compression. Therefore a wrinkling check has been implemented for these members. The checks are executed in each iteration of the solving algorithm. If wrinkling occurs, most likely the solution will not converge. To

⁵ Unified Modeling Language

avoid wrinkling of a given geometry, the prestress must be increased. In combination with the load definitions a prestress factor can be set manually which is applied to all membrane and cable elements consistently. Higher prestress requires a more rigid structure which is usually more expensive and less manageable. Therefore the minimal prestress factor working for all load cases needs to be elaborated manually using several iterations.

One of the findings from working on the assignments from *CM6 Studio Detailing* is that the implemented wrinkling check is very sensitive. Especially models with different prestress for warp and weft are sometimes difficult to find a converging solution. Further investigations in this area are work in progress.

To turn the wrinkling check off, the modeFactor can be set to a value <1.0 as for example 0.99999

3.12.3 Enveloped Results

To correctly dimension a structure, the worst case scenario of all applicable load cases must be considered. This is achieved by calculating enveloped results. Therefore, the following algorithm is implemented in the solver components:

1. Two additional result set are created, one for the minimum values, one for the maximum values.
2. For each result value (displacement, stress) the corresponding min and max value of all result sets is determined and stored.

Figure 3.22 visualizes the algorithm applied to create the enveloped results.

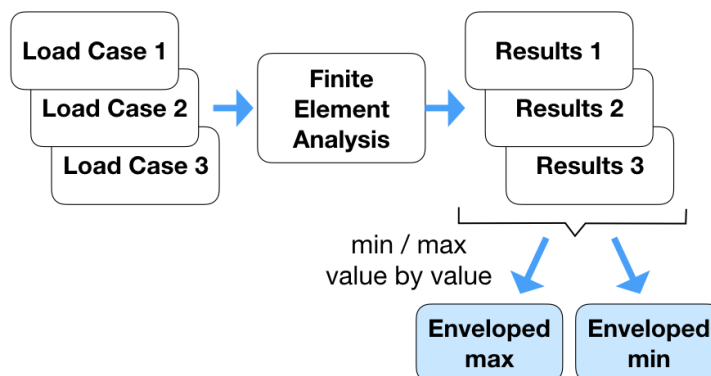


Figure 3.22: Enveloped Results

Enveloped results are only computed when multiple load cases are applied and calculated one after the other.

3.12.3.1 Definition of Minimum

For positive values, the minimum value is the one closest to 0. For negative values, the most negative value is taken as the minimum. To analyze forces the following logic must be considered:

- the biggest tensile force is stored in the result set representing the max values
- the biggest compressive force (most negative) is stored in the result set representing the min values

Figure 3.23 shows an instance of the PamSolver Component when two load cases are applied. For each result set (including enveloped results), an individual output parameter is created on the right hand side of the component.

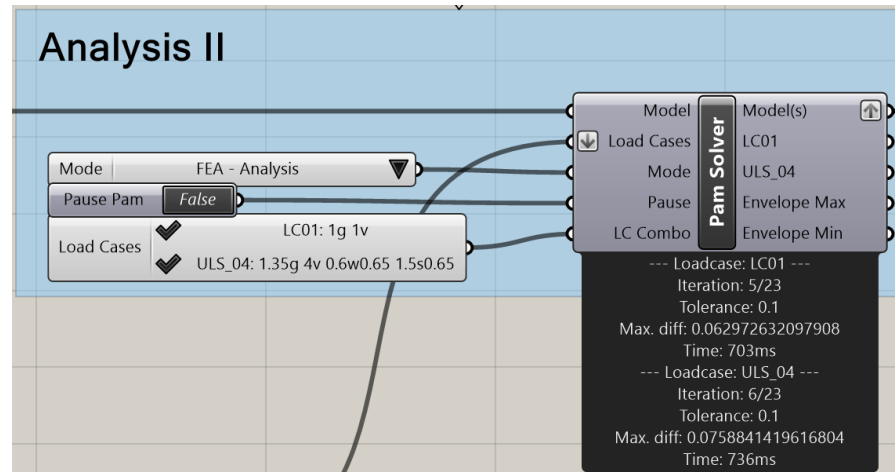


Figure 3.23: FEM Solver

3.12.4 Cascading Solvers

The modular approach of PAMELA allows to use several solvers in parallel or in a row. Running solvers in parallel was very useful when testing the implementation against the original Matlab codebase (see 5.2.5). But the usual case is to have solvers not in parallel but rather in a series:

- the first Solver for form finding
- next one for Structural Analysis with self-weight and prestress. The result will also be used for patterning.
- third solver for structural analysis applying all load cases

3.13 RESULTS VISUALIZATION

Several dedicated components have been developed for results visualization. As shown in figure 3.24, results can be of different nature:

- (changed) geometry

- numerical values
- reactions like shade

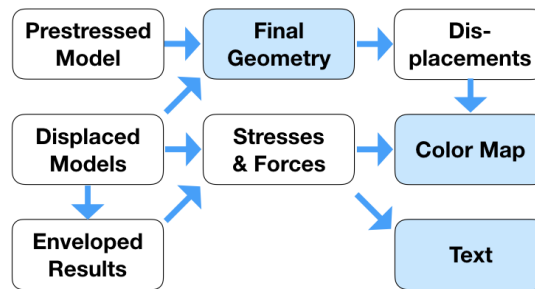


Figure 3.24: result types in context

Different approaches to visualize the results have been implemented based on the nature of the data. These are outlined in the next sections.

3.13.1 Geometry Preview

Some components have a custom preview method implemented to visualize the resulting geometries, usually based on the type of the output parameter(s). Figure 3.25 (and 3.15 as well) show the preview of the model after form finding:

- the mesh is shown in its final shape
- initial / original positions of 1D Elements are shown as dotted lines.
- 1D Elements are drawn in different colors based on type
- Supports are drawn

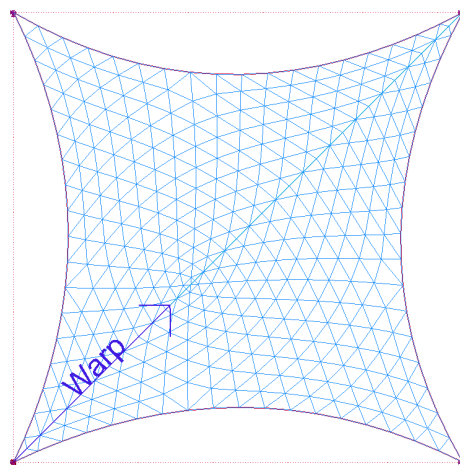


Figure 3.25: preview of model

3.13.2 Colormaps to Visualize Numerical Values

A common approach to visualize numerical values is to use so called colormaps. First the range of the values needs to be determined and then each value a color from the colormap is assigned according to the values position in the range. Colormaps are implemented for different usages:

- coloring text labels used to represent the values. Therefore, each value is drawn in the corresponding color. This is used when axial forces of 1D Elements are visualized (see 4.8.5)
- coloring geometry objects, for example lines or cylinders drawn in different colors
- drawing a mesh surface in so called false colors. Each vertex of the mesh is colored according to the mapped value. To determine the color of the surface area between the vertices, the values of the vertices are interpolated. Figure 3.26 shows an example of visualizing stresses for a mesh. The component 2D Results (see 4.8.8) offers a selection of different stresses or displacements to be visualized either by means of a colored mesh surface or by printing the numbers directly.

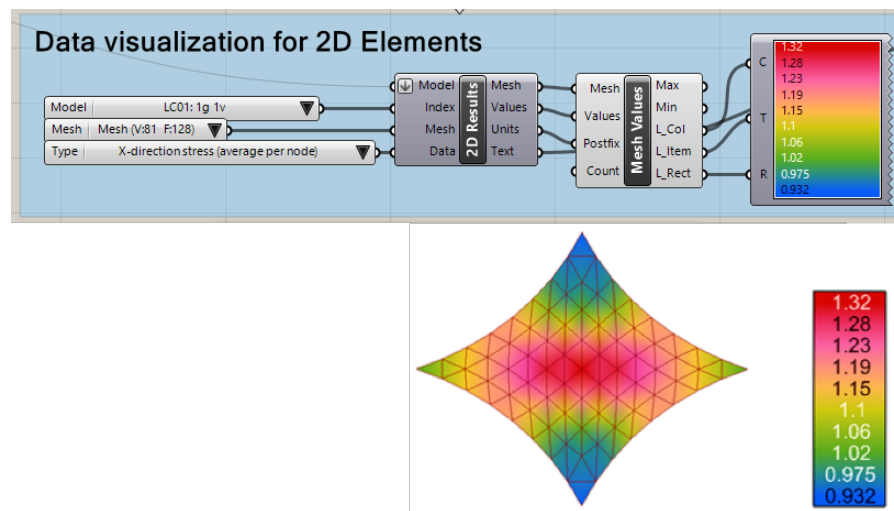


Figure 3.26: colormap applied to mesh values

3.13.3 Additional Visualizations for Values or Results

Most components dealing with the visualization of results are placed in the Panel *Views*. Please refer to section (see 4.8) for a full overview.

3.13.3.1 Contour Lines

The Contour Lines Component (4.8.4) as shown in figure 3.27 draws a variable number of contour lines on meshes or surfaces. This works great to analyze the flow of water or areas where ponding could become an issue.

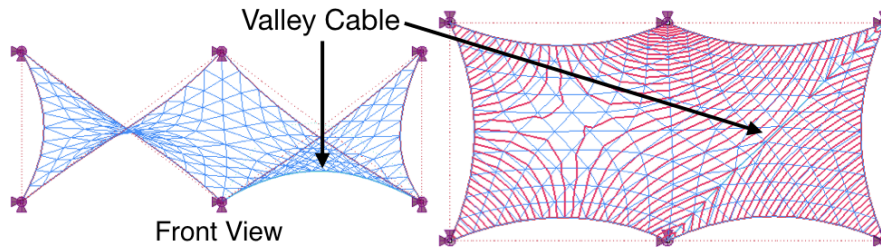


Figure 3.27: Contourlines on mesh surface

3.13.3.2 Slope Analysis

The Mesh Slope Component (4.8.6) allows to analyze the slope of the individual mesh faces. This can be convenient to estimate water flow, exposure to certain loads etc.

3.13.3.3 Shade Analysis

For shade analysis no specific components have been developed yet since some standard components provided by Grasshopper can be used.

By developing some custom components for shade analysis of membranes the usability of this analysis could be further improved.

3.14 PATTERNING

Usually, the form finding of a tensile structure results in non-developable surfaces. Therefore the surface can not be projected onto a plane explicitly, but has to be cut into several pieces, flattened and compensated [15]. The mesh used to model the membrane takes a central position also for the process of *Patterning*. The output mesh from analysis with self weight and applied prestress is used as starting point for patterning, no new mesh is created.

Figure 3.28 shows the process which can be summarized as follows (references to the corresponding components are given in brackets):

1. Determination of panel boundaries by geodesic lines (4.3.10)
2. Refinement of mesh along geodesic lines to create cutting lines along internal mesh edges (4.11.2)

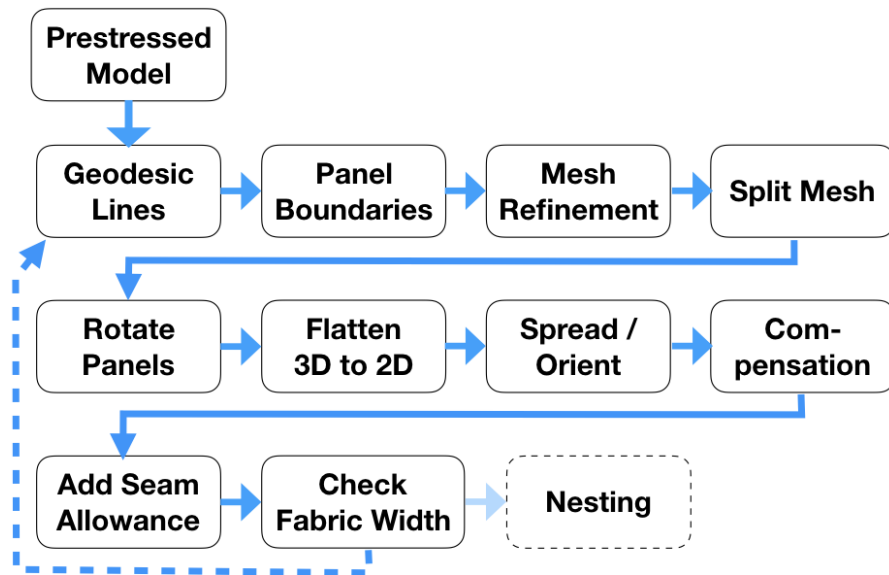


Figure 3.28: Patterning process

3. Split (cut) the mesh along the geodesic lines (4.11.3), label / mark the resulting panels for later reference
4. Rotate the resulting panels (meshes sections) to make sure average of face normals is pointing in direction of Z-axis (prevents panel from flipping upside down)
5. Flatten the panels from 3D to 2D surface, placed on XY plane (4.11.1)
6. Spread / distribute panels in a meaningful order to avoid overlapping
7. Apply compensation to the individual panels
8. Add seam allowances
9. Check panel width against fabric width. If needed, adjust layout defined in step 1 and repeat subsequent steps.
10. add markers, reference lines etc needed for production

3.14.1 Panel Boundaries

Only determination of panel boundaries is currently a mainly manual process. For all the other steps mentioned, components have been developed to assist and automate the process. References are listed in brackets and figure 3.29 shows the corresponding Grasshopper definition to create the geodesic lines and refine and split the mesh.

Figure 3.30 shows the result of the first task in the patterning process, the determination of the panel boundaries. To achieve this result, the following steps were executed:

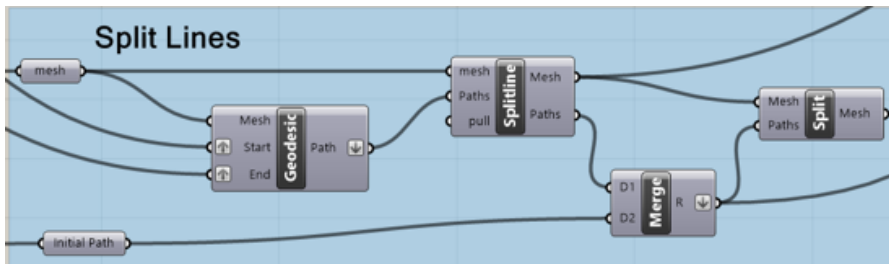


Figure 3.29: Grasshopper definition for patterning

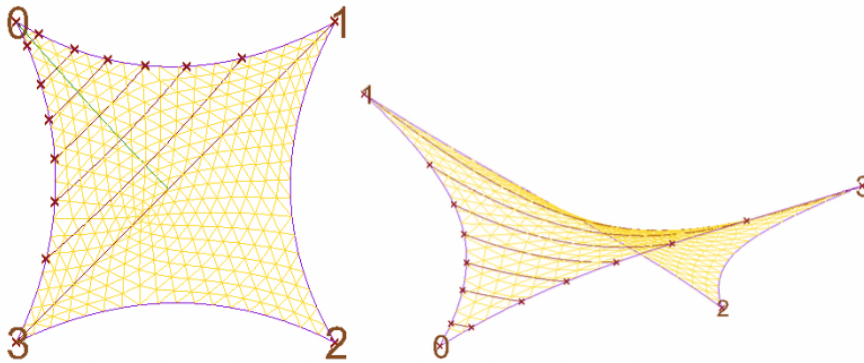


Figure 3.30: panel layout using geodesic lines

1. A geodesic line has been laid between corners 1 and 3 to create the central seam.
2. A geodesic line (green) has been laid between corner 0 and its closest point on the resulting geodesic line from the first step
3. The green geodesic has been split into segments with a length of approximately 90% of fabric width (depends on panel length, curvature etc.).
4. Temporary planes have been defined based on warp orientation and tangents of the green line for each segment boundary
5. Geodesic lines have been laid between corresponding intersection points of the membrane edges with the temporary planes

These steps need to be adopted based on membrane shape and size, fabric width and desired appearance of the overall seam layout.

3.14.2 Flattening

The algorithm implemented to flatten the panels is adopted to the force density method used for form finding and is working directly with the already existing meshes. This guarantees highest possible accuracy. Not all surfaces are developable. Therefore the results of the flattening process must be checked afterwards for accuracy. The assessment criteria for the accuracy of the flattening process include:

- change of surface area (ideally 0.0)
- change of edge length for each edge (ideally 0.0)

Figure 3.31 shows the top view of 7 flattened panels of one half of a membrane. After initial flattening, the panels have been oriented to have warp direction aligned with the X-axis and to be nicely spaced from each other. Furthermore, the refinement of the mesh along the cutting lines becomes clearly visible.

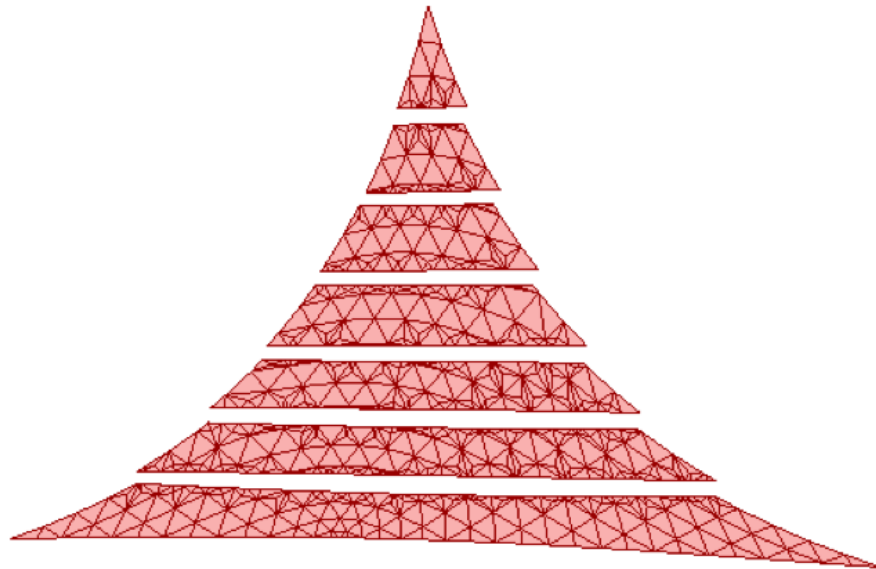


Figure 3.31: flattened and oriented 2D panels

3.14.3 Compensation

To activate prestress in the fabric of the installed membrane it is necessary to change size of the panels according to material specific compensation values. The key concept is to define compensation values for each panel by area. With this approach an area can be excluded from compensation in one or both directions, what in turn makes decompensation needless.

1. A reference point is chosen. The reference point must be included in the area of reduced / ignored compensation if such area is defined.
2. For each mesh vertex on a naked edge of the panel the vector specifying the vertices position relative to the reference point is calculated.
3. The x and y components of the vector are multiplied with (1 - compensationfactor) for the respective direction
4. The final positions for each vertice is calculated based on reference point + compensated vector

5. A closed polyline is drawn connecting all final positions.

Figure 3.32 shows the result of applying compensation values to one of the panels.

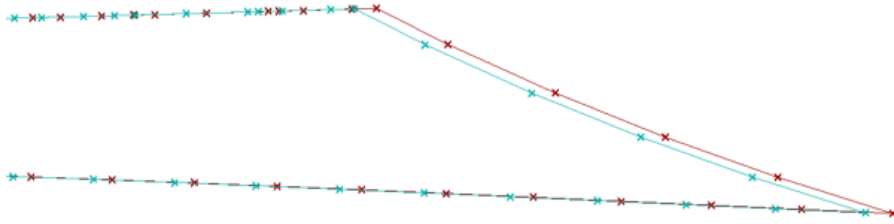


Figure 3.32: detail of compensated panel

The functionality for compensation has only been prototyped so far and is not yet included in the PAMELA library.

3.14.4 Seam Allowance

Cutting lines usually denote some kind of connection border. This can be a seam between two lines (for example overlap) or connection to Keder or cable pocket etc. Depending on connection type, material, processing method (welding, sewing etc.) different seam allowance is needed which needs to be respected and patterns adopted accordingly. This part is work in progress and will be added to PAMELA once more experience is gained. Nevertheless, some information regarding the applicable algorithms is given in the following subsections.

3.14.4.1 Connection Seams

For connection seams, the panels can be modified according to the following rules:

- $0.5 \cdot \text{seam width}$ is added to all edges which are connection seams. For a seam width of 50mm, a parallel curve with distance 25mm is drawn on the outside of the panel. This becomes the cutting line.
- Based on the cutting line, a parallel curve with distance 50mm is drawn on the inside of the panel. This is the marker line for the seam overlap.
- The system line is divided into (a random number of) equal length segments. For each segment a short marker perpendicular to the system line is drawn as guiding line for the welding process.

3.14.4.2 Edge Cable Pockets

Edge cables pockets are made out of 2 pieces, the membrane panel with a certain allowance and an additional flap which is a mirrored copy of the edge. The dimensions depend on the cable diameter. Figure 3.33 shows the principle. The reference is always the system line which is located at the center of the cable.

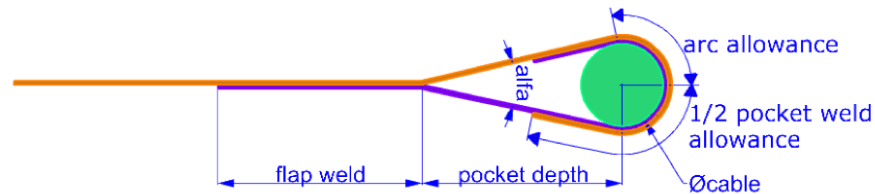


Figure 3.33: measurements for cable pockets

According to Stranghoner [22], the angle alfa must not be larger than 24 degrees, otherwise peeling may occur. A Microsoft Excel spreadsheet has been created to calculate all necessary values based on cable diameter and angle alfa. All numbers are rounded, seam widths in steps of 10mm. The numbers from the table can be applied to all edge cable pockets according to the cable size.

3.14.4.3 Ridge or Valley Cable Pockets

In the areas of ridge or valley cables, seam width is doubled. An additional pocket is used to keep the cable(s) in place. Depending on the curvature of the membrane, these pockets are optional. Figure 3.34 shows the principle to apply for ridge or valley cable pockets.

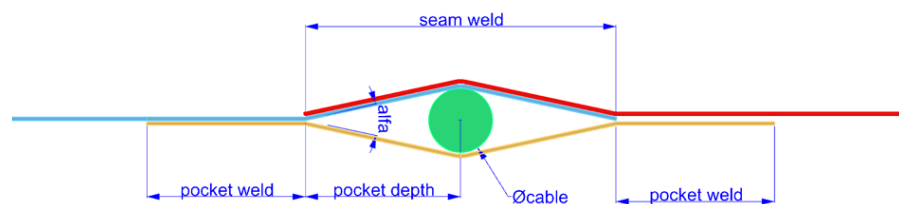


Figure 3.34: measurements for ridge or valley cable pockets

3.14.5 Checking Fabric Width

The oriented panels allow for a first check of the dimensions based on system lines. Seam allowance needs to be added first to check total width against fabric width. Figure 3.35 shows the dimensions of one of the panels.

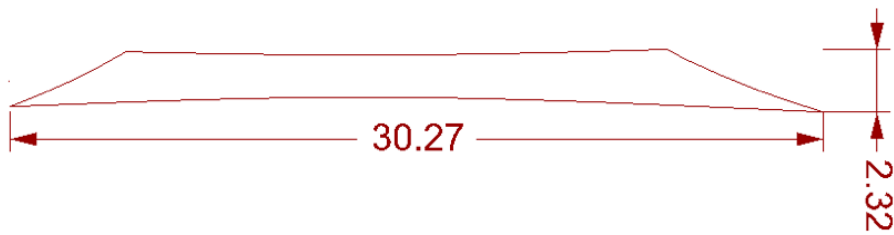


Figure 3.35: dimensions of flattened and compensated panel

3.15 DETAILING AND 3D VISUALIZATION

For most structures, cables, shackles, turnbuckles etc need to be connected to masts, beams or anchor plates. To accommodate this in a standardized way, three main components have been developed:

- The Connector Plate is a fin or earplate to be welded on a supporting structure like a mast or an anchor plate. The shape is automatically optimized based on the force vector and base plane. The dimensions as shown in figure 3.36 are based on the recommendations from Pfeifer for their system 860[16], although different dimensions can be used without any problem.
- The rectangular and round anchor plates can be used in conjunction with the connector plate or for other purposes. The size, number of holes, diameters etc. are all variable and can be specified by parameters.

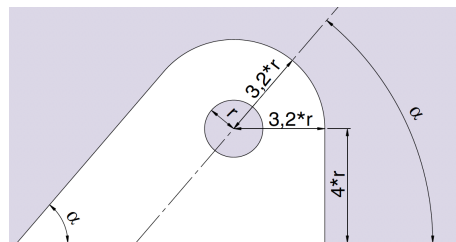


Figure 3.36: Dimensions of connector plate[16]

The components listed above are described in more detail in section 4.9. Figure 3.37 shows the resulting visualizations based on these components from their use for CM₃ Structural Design. It's worth to note that these components make available specific output parameters for 3D drawings as well as for 2D drawings. This greatly simplifies the task of creating shop drawings as well as 3D visualizations.

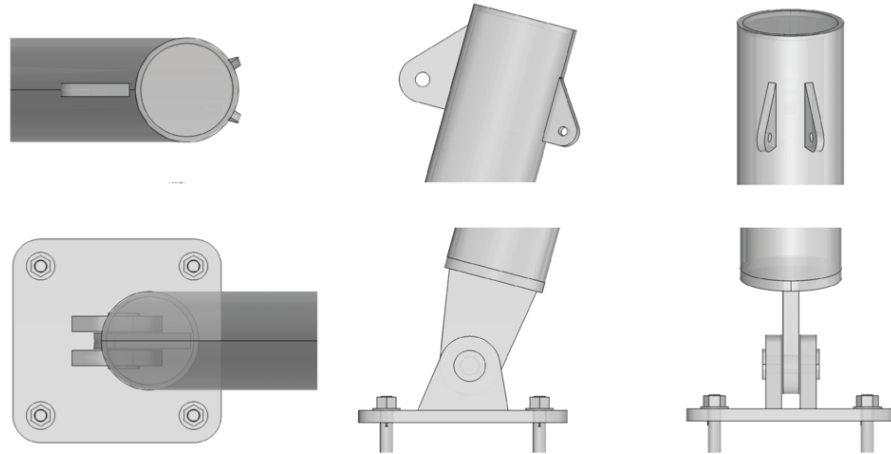


Figure 3.37: Visualizations of connector and anchor plates

3.15.1 Membranedetails

For corner details of a membrane, currently only the generic corner-plate as mentioned in 3.10.3 has been developed. To model corner details, clampings, mast structures, anchorages etc. Grasshopper templates are available from www.membranedetail.com[13]. Since these definitions are all based on Grasshopper, they can be easily integrated.

4 | USER INTERFACE

Science is knowledge which we understand so well that we can teach it to a computer; and if we don't fully understand something, it is an art to deal with it.

— Donald E. Knuth

Section 3.4 has shown that the Grasshopper plugin to Rhino3D CAD system is used as the main runtime environment for all user interactions. The *User Interface* of the PAMELA software library adds a set of (currently) 80 components to the already existing Grasshopper components and parameters.

This chapter serves as an overview and reference for the comprehensive set of Grasshopper Components developed for this project. The corresponding Workflows and use cases from which the need to develop the different components was derived are described in chapter 3. Usage examples can be found in chapter 5 where the system is "put into action". Section 5.3 describes in detail the process of form-finding, analysis and patterning for a hyper with ridge cable.

4.1 APPEARANCE IN GRASSHOPPER

When a Grasshopper component is implemented, one needs to specify a category and subcategory to determine the appearance in Grasshopper user interface. The category becomes the *Tab* and the subcategory the *Panel*. Each tab can contain a collection of multiple panels. *Pamela* is being used as the category (and therefore name of the tab) for all components to make sure they are grouped together and can easily be identified. It would also be possible to insert the components into other, already existing panels. Figure 4.1 shows the tab and all associated panels in Grasshopper.

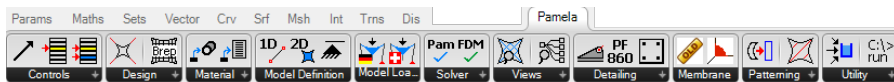


Figure 4.1: all panels in the tab *Pamela*

4.1.1 Panels and their Order

The order of the panels (subcategories) within a Grasshopper tab is alphabetical and cannot be changed. When selecting the names for

Note: The order of the subsequent sections of this chapter corresponds with the subcategories (panels) in Grasshopper and therefore with the appearance of the components in Grasshopper.

the subcategories, care was taken to ensure that an alphabetical order was created from left to right according to the workflow steps. To support this an additional trick was used: for some groups the name was prefixed with a space. In this way it is for example possible that the group *Solver* appears before the group *Detailing*.

The following structure was implemented:

- Controls -> see section 4.2 for all components
- Design -> see section 4.3 for all components
- Material -> see section 4.4 for all components
- Model Definition -> see section 4.5 for all components
- Model Loads -> see section 4.6 for all components
- Solver -> see section 4.7 for all components
- Views -> see section 4.8 for all components
- Detailing -> see section 4.9 for all components
- Membrane -> see section 4.10 for all components
- Patterning -> see section 4.11 for all components
- Utility -> see section 4.12 for all components

4.1.2 Order within Panels

With an additional property called *Exposure* it's possible to create an additional hierarchical grouping within a panel. Every exposure value ends up in a different panel section. However, within a section (exposure) the order is alphabetical and there's nothing one can do about that. When selecting the values for the exposure properties, care was taken to ensure that commonly used components appear first and common workflow is supported in a top-down order. As an example figure 4.2 shows the panel *Design* which is structured as follows:

- *Create Links Component* (-> see section 4.3.1) appears first, because it can be used to model the connection points of a membrane with the supporting structure (link cables, shackles, turnbuckles etc) by means of line elements with usually fixed length.
- The next group contains the two components *Brep Mesher* (-> see section 4.3.2) and *Edge Mssher* (-> see section 4.3.3) which can be used to create the mesh required to do form-finding for membranes.

- *Mesh Edges* (-> see section 4.3.4), *Ridge / Valley Cable* (-> see section 4.3.5) and *Vertices on Curve* (-> see section 4.3.6) components are contained in the third group because they require a mesh as input parameter and can be used to specify how the membrane is held in place and tensioned by means of defining model elements based on mesh vertices and mesh edges.
- The remaining two groups contain components to further modify a mesh or find a path or geodesic line on the mesh surface.

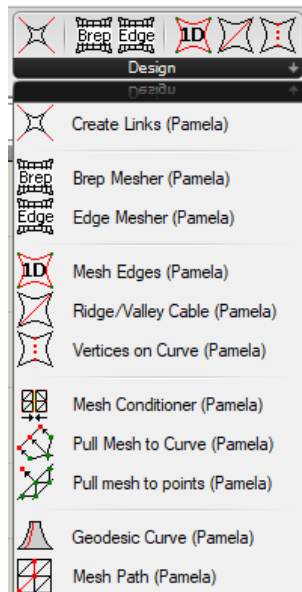


Figure 4.2: order of the components in the panel *Design*

The same logic - top-down order based on workflow and frequency of use - has been applied to all panels but is not further explained for each panel.

4.1.3 Information provided

For each component in addition to a short overview and the listing of all input and output parameters a screenshot is included which shows how the component appears directly after its placement on the Grasshopper document without any further modification / connection to other components. Whenever it makes sense, default values are assigned to the input parameters. As an example, the plane *World.XY* is commonly used when a reference plane needs to be specified.

In section 3.6.4 the coloring of components according to their state is described. It becomes obvious that not all components are in a valid state right after placement and therefore some are shown *orange* and some are shown *grey*. But initial state should never be error (*red*).

4.2 PANEL: CONTROLS

Figure 4.3 shows all icons of the components grouped under the panel *Controls*

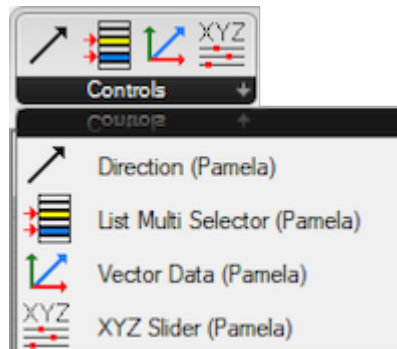



Figure 4.3: overview of panel *Controls*

4.2.1 Component: Direction (Pamela)

4.2.1.1 Synopsis

Icon	
Purpose	Definition of a vector to specify direction of wind, warp etc. The output vector is unitized by default. When enabled, a preview arrow is drawn in the viewport with correct length and orientation and a text label is placed along the arrow to indicate the purpose of use. Use the dropdown menus to change length, color etc.
Nickname	Direction
Type Name	Pamela.GH.Controls.Direction_Component
Location	Controls

4.2.1.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Reference	Plane	Reference plane for start point of vector and rotation.
Direction	Point	Optional point for directional reference. If not provided, the X-axis of the reference plane is used.
Rotation	Number	Rotation around Z-axis in degrees.
Inclination	Number	Inclination from Z-axis in degrees.

4.2.1.3 Output Parameters

ID	Type	Description
Vector	Vector	Resulting vector considering all input parameters. By default, output is unitized.

4.2.1.4 Menu items

The following component menu items exist in addition to the standard drop down menu items:

- Label Text and Position
- Preview Color
- Length
- Thickness

4.2.1.5 Appearance

Figure 4.4 shows the component *Direction (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

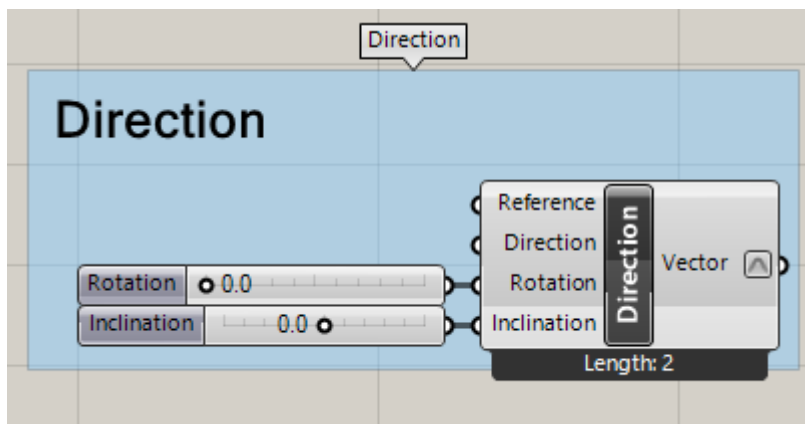



Figure 4.4: Screenshot of *Direction (Pamela)*

4.2.2 Component: List Multi Selector (Pamela)

4.2.2.1 Synopsis

Icon	
Purpose	Easily select one or more items from a list through the attached slider(s). Zoom-in to modify the number of number sliders to select items.
Nickname	Items
Type Name	Pamela.GH.Controls.ItemMultiSelector_Component
Location	Controls

4.2.2.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
List	Generic Data	List to select items from.
all	Boolean	On first output parameter ALL items of the list are returned instead of only the selected ones, therefore ignoring all other inputs and bridging input list to output list.
index	Integer	Index of first element to select.

4.2.2.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
List	Generic Data	List of all elements selected by any of the inputs.

4.2.2.4 Additional Information

The selectable range of numbers of the Sliders (maximum value) is automatically set according to the list length and adjusted when the size of the input list changes.

4.2.2.5 Appearance

Figure 4.5 shows the component *List Multi Selector (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly. .

Note: This component has a variable number of parameters. In Grasshopper, zoom in to discover them (+ and - signs will become visible)

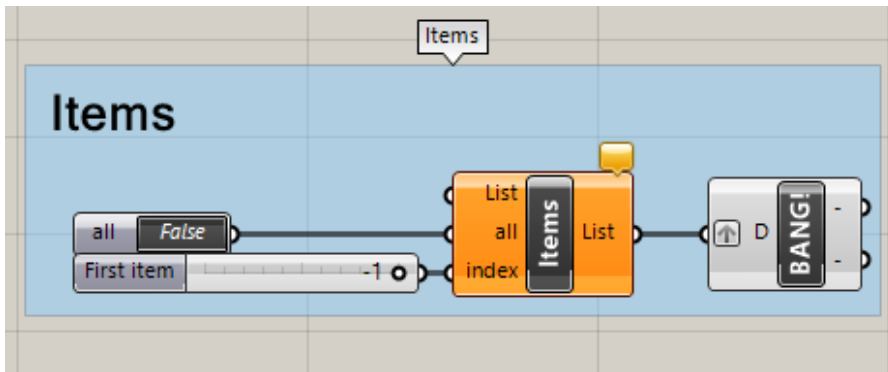


Figure 4.5: Screenshot of *List Multi Selector (Pamela)*

4.2.3 Component: Vector Data (Pamela)

4.2.3.1 Synopsis

Icon	
Purpose	Modify a vector by splitting it into it's components and allowing to combine / select a subset (e.g. only X and Z) of the components as output.
Nickname	Vector Data
Type Name	Pamela.GH.Controls.VectorData_Component
Location	Controls

4.2.3.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Vec	Vector	Vector(s) to decompose
X	Boolean	X component of Vector (true / false)
Y	Boolean	Y component of Vector(true / false)
Z	Boolean	Z component of Vector(true / false)
abs	Boolean	Outputs are based on absolute (non-negative) values.
unit	Boolean	Unitize output vector.
mag	Number	Factor to multiply vector components with.

4.2.3.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Vector	Vector	Resulting vectors considering all input parameters.
X	Number	X component of resulting vectors.
Y	Number	Y component of resulting vectors.
Z	Number	Z component of resulting vectors.
Length	Number	Length of resulting vectors.
Text	Text	Textual description of selected input parameters.

4.2.3.4 Appearance

Figure 4.6 shows the component *Vector Data (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

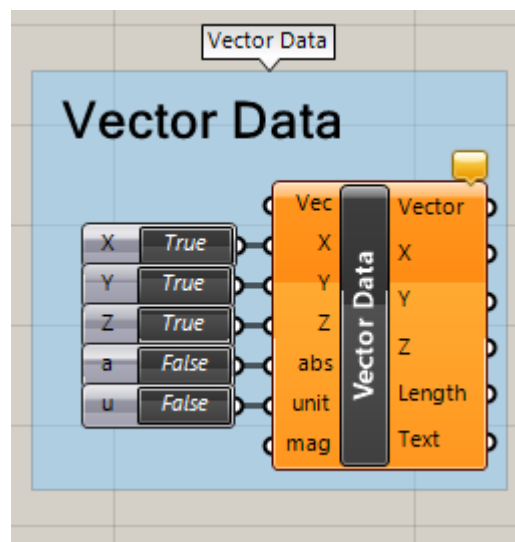



Figure 4.6: Screenshot of *Vector Data (Pamela)*

4.2.4 Component: XYZ Slider (Pamela)

4.2.4.1 Synopsis

Icon	
Purpose	Three sliders combined. Double-click on the slider knobs to modify min / max and also precision.
Nickname	XYZ
Type Name	Pamela.GH.Controls.XYZSliderComponent
Location	Controls

4.2.4.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Plane	Plane	Optional reference plane for transformation relative to World.XY
X Domain	Domain	Optional range (Domain) for X
Y Domain	Domain	Optional range (Domain) for Y
Z Domain	Domain	Optional range (Domain) for Z

4.2.4.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
X	Number	X value
Y	Number	Y value
Z	Number	Z value
Pt	Point	Point object constructed from X, Y and Z values in combination with reference plane.
V	Vector	Vector constructed from X, Y and Z values in combination with reference plane.

4.2.4.4 Appearance

Figure 4.7 shows the component *XYZ Slider (Pamela)* after placement on the Grasshopper document.

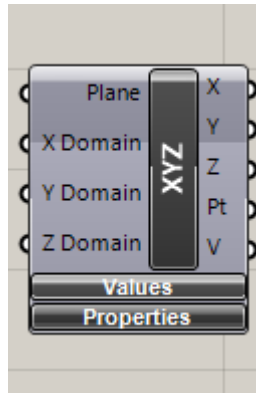


Figure 4.7: Screenshot of *XYZ Slider (Pamela)*

4.3 PANEL: DESIGN

Figure 4.8 shows all icons of the components grouped under the panel *Design*

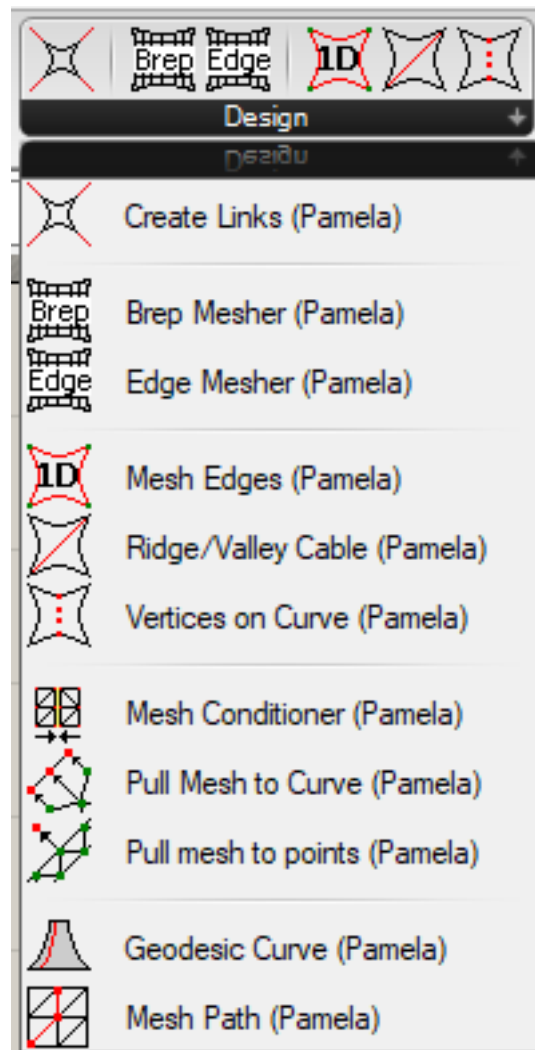



Figure 4.8: overview of panel *Design*

4.3.1 Component: Create Links (Pamela)

4.3.1.1 Synopsis

Icon	
Purpose	Create a set of fixed length Links for a membrane. They can be used to model the connection points of a membrane with the supporting structure (link cables, shakles, turnbuckeles etc). A link may also have length 0. The last length supplied is repeated / used for all remaining links. Zoom-in to add more digit scrollers to select individual numbers. The Endpoints of the output lines are to be used as cornerpoints of the membrane.
Nickname	CreateLinks
Type Name	Pamela.GH.Design.CreateLinks_Component
Location	Design

4.3.1.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Pts	Generic Data	List of points (polyline) to use as fixed / supported endpoints.
rev	Boolean	Reverse order of output points.
Len	Number	List of lengths to use for this link.

4.3.1.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Perimeter	Curve	Closed polyline connecting the inner points. Use it as edge for meshing.
Corner Pts	Point	List of points to use as cornerpoints of the membrane.
Links (1D)	Line	List of lines according to selected link lengths > 0. Can be used to create fixed length 1D Elements.
Fixation Pts	Point	List of points to use as outer endpoints (most likely fixed / supported endpoints).

4.3.1.4 Appearance

Figure 4.9 shows the component *Create Links (Pamela)* and all embedded components and parameters after placement on the Grasshopper

document. The components are automatically grouped and the group is labelled accordingly. .

Note: This component has a variable number of parameters. In Grasshopper, zoom in to discover them (+ and - signs will become visible)

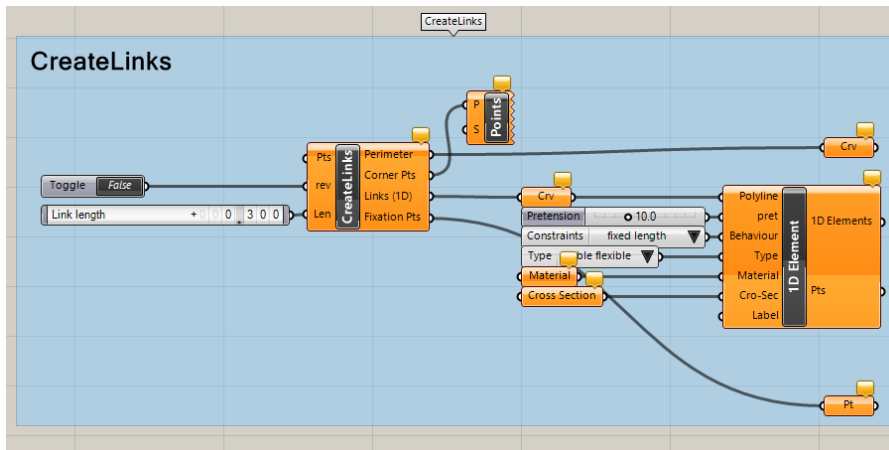



Figure 4.9: Screenshot of *Create Links (Pamela)*

4.3.2 Component: Brep Mesher (Pamela)

4.3.2.1 Synopsis

Icon	
Purpose	Create meshes from curves or surfaces.
Nickname	Brep Mesher
Type Name	Pamela.GH.Design.BrepMesher_Component
Location	Design

4.3.2.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Breps	Brep	List of polysurfaces.
Tolerance	Number	Intersection tolerance to be used during geometry splitting.
Length	Number	Target average edge length for the resulting meshes. Small values can increase calculation time significantly!
Scaling	Number	Scaling of the mesh in conjunction with mesh size. Values around 1 are good.
Iterations	Integer	Number of iterations applied to smoothen differences in triangle sizes.
Flip	Boolean	Change direction of mesh normals for the joined mesh. By default face normals are oriented to minimize the angle between the Z-axis and the arithmetic average of all face normals (angle < 90 degrees).
Pause	Boolean	Pause Mesher while changing values of preceding components to avoid unnecessary heavy computations.

4.3.2.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Joined Mesh	Mesh	Joined Mesh.
Meshes	Mesh	List of Meshes. Consider joining them to one larger Mesh.
Curves	Curve	List of polylines corresponding to the input curves

4.3.2.4 *Appearance*

Figure 4.10 shows the component *Brep Mesher (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

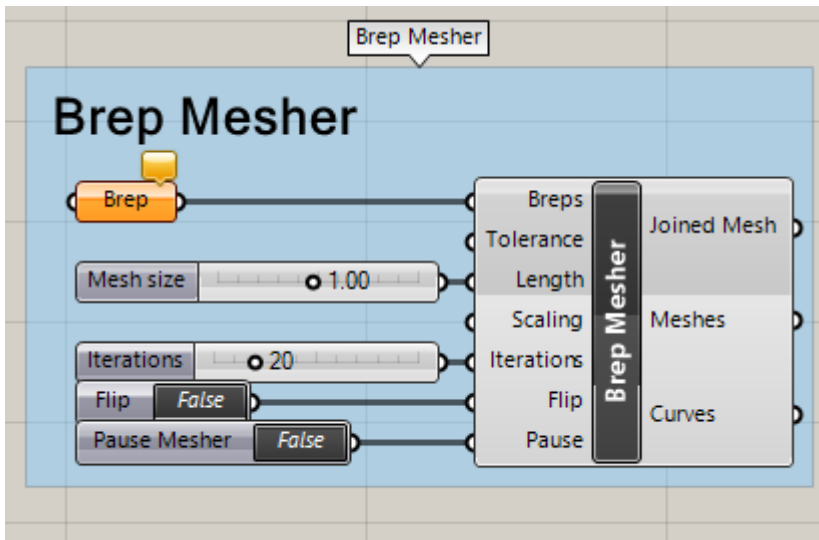



Figure 4.10: Screenshot of *Brep Mesher* (Pamela)

4.3.3 Component: Edge Mesher (Pamela)

4.3.3.1 *Synopsis*

Icon	
Purpose	Create mesh from edge curves
Nickname	Edge Mesher
Type Name	Pamela.GH.Design.EdgeMesher_Component
Location	Design

4.3.3.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Edges	Curve	Input curves. Best results are achieved for closed curves or when individually supplied segments can be joined to a closed curve.
Resolution	Integer	Mesh Resolution. Number of segments per edge.
Relax	Number	Relaxation step
Iterations	Integer	Number of iterations
Flip	Boolean	Change direction of mesh normals. By default face normals are oriented to minimize the angle between the Z-axis and the arithmetic average of all face normals (angle < 90 degrees).
Pause	Boolean	Pause Mesher while changing values of preceding components to avoid unnecessary heavy computations.

4.3.3.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Output Mesh
Corners	Point	Points contained in supplied input curves. At least some of these points are most likely to be used as corners of the membrane or for supports.

4.3.3.4 *Appearance*

Figure 4.11 shows the component *Edge Mesher (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

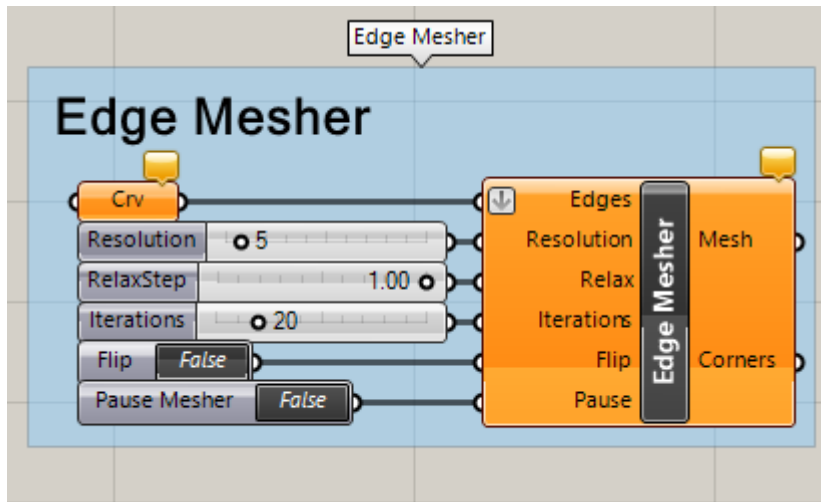



Figure 4.11: Screenshot of *Edge Mesher* (Pamela)

4.3.4 Component: Mesh Edges (Pamela)

4.3.4.1 Synopsis

Icon	
Purpose	Polylines of path along naked edges between corner-points. Can be used to define 1D Elements for edge cables of membranes.
Nickname	Mesh Edges
Type Name	Pamela.GH.Design.MeshEdges_Component
Location	Design

4.3.4.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	The input mesh to analyze.
Corners	Generic Data	Points or Polyline connecting corner points of interest. Make sure the order of the points is adequate.

4.3.4.3 Output Parameters

ID	Type	Description
All Edges	Curve	List of polylines connecting all vertices on the naked edges between the supplied corner points.
0	Curve	First Edge Polyline
1	Curve	2nd Edge Polyline
2	Curve	3rd Edge Polyline
3	Curve	4th Edge Polyline

4.3.4.4 Appearance

Figure 4.12 shows the component *Mesh Edges (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly. .

Note: This component has a variable number of parameters. In Grasshopper, zoom in to discover them (+ and - signs will become visible)

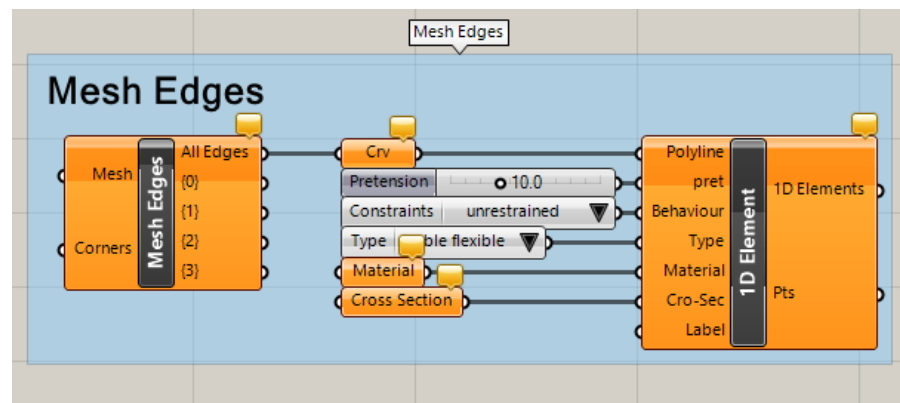



Figure 4.12: Screenshot of *Mesh Edges (Pamela)*

4.3.5 Component: Ridge/Valley Cable (Pamela)

4.3.5.1 Synopsis

Icon	
Purpose	Design ridge or valley cables on a mesh. The mesh is refined to provide a path along (internal) edges following a geodesic line from start to end. IMPORTANT: Use the output mesh to further specify the model.
Nickname	R/V Cable
Type Name	Pamela.GH.Design.RidgeValleyCable_Component
Location	Design

4.3.5.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	The input mesh to analyze.
Points	Generic Data	List of points to select start and end-points from, usually you can assign the cornerpoints of the membrane.
Start	Integer	Index of start point(s)
End	Integer	Index of end point(s)
Tolerance	Number	Maximum distance an intersection point is moved (pulled) towards an existing vertex. Values > 0 can be used to avoid very tiny mesh triangles, the mesh path is straightened anyway by applying tension on the designed cable.

4.3.5.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Modified output Mesh. Use this mesh to further specify the model.
Curves	Curve	Polylines connecting all vertices on the geodesic line from start to end, denoting the ridge or valley cable.

4.3.5.4 Appearance

Figure 4.13 shows the component *Ridge/Valley Cable (Pamela)* and all embedded components and parameters after placement on the

Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

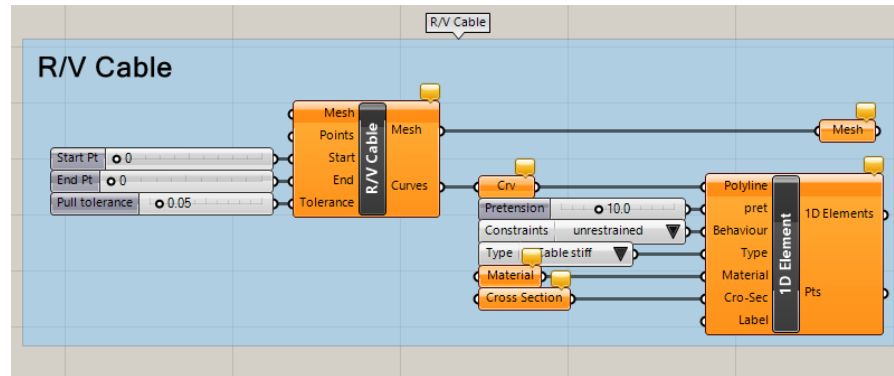


Figure 4.13: Screenshot of *Ridge/Valley Cable (Pamela)*

4.3.6 Component: Vertices on Curve (Pamela)

4.3.6.1 Synopsis

Icon



Purpose

Find all vertices of a mesh which are located directly on a curve. This can be used to select naked edges based on boundary curves like arches, rings, beams etc which were used to create the mesh. As an example, these points can be used as input for support definitions.

Nickname Vert on Crv

Type Name Pamela.GH.Design.VerticesOnCurve_Component

Location Design

4.3.6.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	The input mesh to analyze.
Crv	Curve	Curve to search intersecting vertices.
Tolerance	Number	Maximum allowable distance of vertices to the curve to be included in result set.

4.3.6.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Pts	Point	Positions of vertices on curve.

4.3.6.4 Appearance

Figure 4.14 shows the component *Vertices on Curve (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

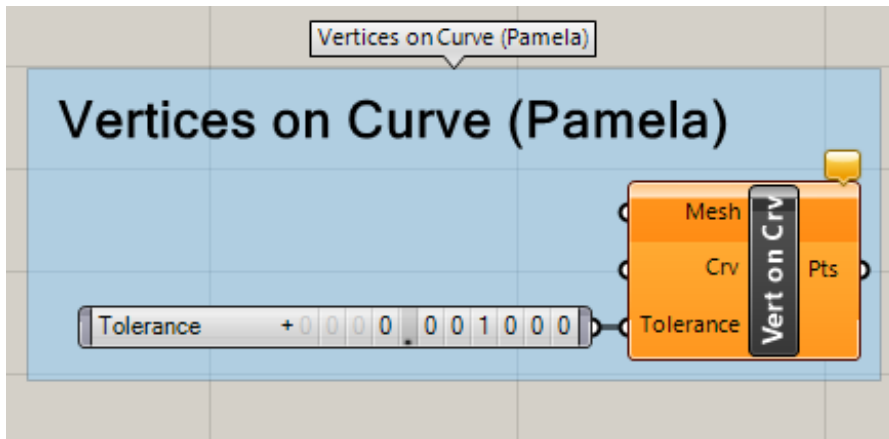



Figure 4.14: Screenshot of *Vertices on Curve (Pamela)*

4.3.7 Component: Mesh Conditioner (Pamela)

4.3.7.1 Synopsis

Icon	
Purpose	Mesh conditioner to join and weld multiple meshes, remove duplicate vertices, unify face winding etc. Especially useful when meshes are created or modified with 3rd-party tools or directly in Rhino.
Nickname	Join & Weld
Type Name	Pamela.GH.Design.MeshConditioner_Component
Location	Design

4.3.7.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Mesh(es) to join, weld and remove any other inconsistencies.
Flip	Boolean	Change direction of mesh normals. By default face normals are oriented to minimize the angle between the Z-axis and the arithmetic average of all face normals (angle < 90 degrees).

4.3.7.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Output Mesh

4.3.7.4 Appearance


Figure 4.15 shows the component *Mesh Conditioner (Pamela)* after placement on the Grasshopper document.



Figure 4.15: Screenshot of *Mesh Conditioner (Pamela)*

4.3.8 Component: Pull Mesh to Curve (Pamela)

4.3.8.1 Synopsis

Icon	
Purpose	Each vertex on the mesh path is moved to the closest point of the supplied curve. If no start and end vertices are supplied, the vertices closest to the end-points of the curve are used.
Nickname	PullMeshToCurve
Type Name	Pamela.GH.Design.PullMeshToCurve_Component
Location	Design

4.3.8.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	The input mesh to modify.
crv	Curve	Curve to pull vertices to.
start	Point	Vertex of mesh to use as starting point for mesh path. If not provided, vertex closest to start of curve is used.
end	Point	Vertex of mesh to use as end point for mesh path. If not provided, vertex closest to end of curve is used.
ne	Boolean	If set to true, only vertices of naked edges are considered.

4.3.8.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Output Mesh
Pts	Point	Positions of vertices on curve.

4.3.8.4 *Appearance*

Figure 4.16 shows the component *Pull Mesh to Curve (Pamela)* after placement on the Grasshopper document.

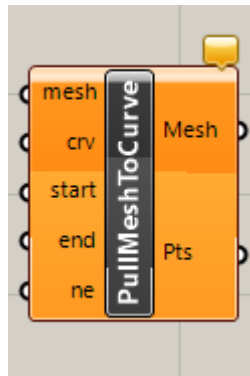



Figure 4.16: Screenshot of *Pull Mesh to Curve (Pamela)*

4.3.9 Component: Pull mesh to points (Pamela)

4.3.9.1 Synopsis

Icon	
Purpose	For each point supplied, the coordinates of the closest vertex of the mesh are changed to the location of the point.
Nickname	PullMeshToPoints
Type Name	Pamela.GH.Design.PullMeshToPoints_Component
Location	Design

4.3.9.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	The input mesh to modify.
pos	Generic Data	Target positions
ne	Boolean	If set to true, only vertices of naked edges are considered.

4.3.9.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Output Mesh

4.3.9.4 Appearance

Figure 4.17 shows the component *Pull mesh to points (Pamela)* after placement on the Grasshopper document.

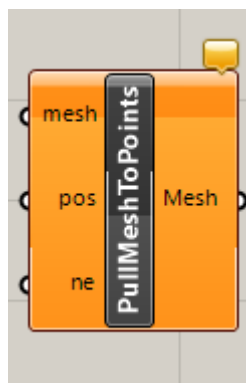



Figure 4.17: Screenshot of *Pull mesh to points (Pamela)*

4.3.10 Component: Geodesic Curve (Pamela)

4.3.10.1 Synopsis

Icon	
Purpose	Shortest route between two points on mesh surface. Polylines are created based on intersection points with mesh edges and vertices.
Nickname	Geodesic
Type Name	Pamela.GH.Design.GeodesicLine_Component
Location	Design

4.3.10.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Input mesh to analyze.
Start	Point	Start point(s)
End	Point	End point(s)

4.3.10.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Path	Curve	Polyline(s) representing the shortest path connecting start and end points.

4.3.10.4 Appearance


Figure 4.18 shows the component *Geodesic Curve (Pamela)* after placement on the Grasshopper document.



Figure 4.18: Screenshot of *Geodesic Curve (Pamela)*

4.3.11 Component: Mesh Path (Pamela)

4.3.11.1 Synopsis

Icon	
Purpose	Shortest path on a mesh along edges (going through vertices) connecting source and destination points.
Nickname	Mesh Path
Type Name	Pamela.GH.Design.MeshPath_Component
Location	Design

4.3.11.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Input mesh to analyze.
Start	Point	Start point(s)
End	Point	End point(s)

4.3.11.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Path	Curve	Polyline(s) representing the shortest path connecting start and end points.

4.3.11.4 Appearance

Figure 4.19 shows the component *Mesh Path (Pamela)* after placement on the Grasshopper document.



Figure 4.19: Screenshot of *Mesh Path (Pamela)*

4.4 PANEL: MATERIAL

Figure 4.20 shows all icons of the components grouped under the panel *Material*

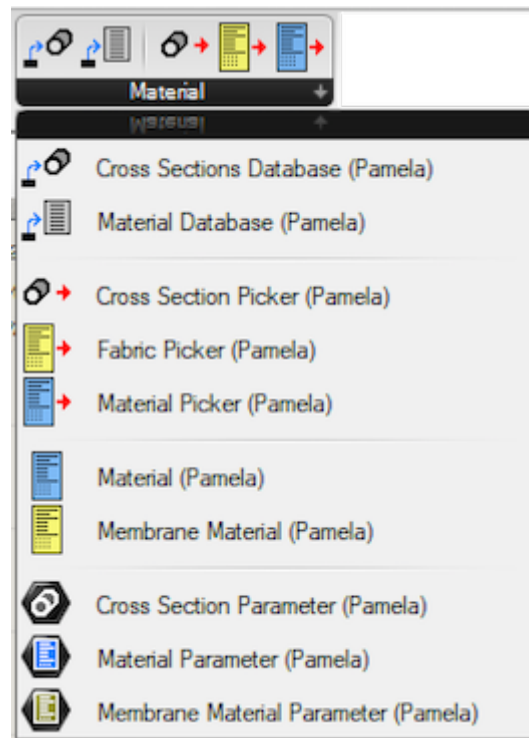



Figure 4.20: overview of panel *Material*

4.4.1 Component: Cross Sections Database (Pamela)

4.4.1.1 *Synopsis*

Icon	
Purpose	Load cross section properties from file.
Nickname	Cro-Sec DB
Type Name	Pamela.GH.Material.CrossSectionLoader_Component
Location	Material

4.4.1.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
File	Text	Uri of file to read

4.4.1.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Cro-Secs	CrossSection	List of Cross Section objects

4.4.1.4 Appearance

Figure 4.21 shows the component *Cross Sections Database (Pamela)* after placement on the Grasshopper document.

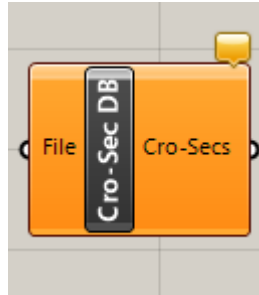


Figure 4.21: Screenshot of *Cross Sections Database (Pamela)*

4.4.2 Component: Material Database (Pamela)

4.4.2.1 Synopsis

Icon	
Purpose	Load properties of all Material types from file
Nickname	Material DB
Type Name	Pamela.GH.Material.MaterialLoader_Component
Location	Material

4.4.2.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
File	Text	Uri of file to read

4.4.2.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Membranes	Membrane Material	List of Membrnae Material objects
Materials	Material	List of Material objects

4.4.2.4 Appearance

Figure 4.22 shows the component *Material Database (Pamela)* after placement on the Grasshopper document.

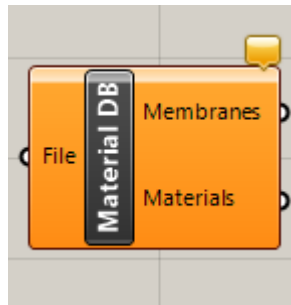


Figure 4.22: Screenshot of *Material Database (Pamela)*

4.4.3 Component: Cross Section Picker (Pamela)

4.4.3.1 Synopsis

Icon	
Purpose	Select a cross section based on hierarchical grouping criterias
Nickname	Cross Section
Type Name	Pamela.GH.Material.CrossSectionPicker_Component
Location	Material

4.4.3.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Cross Sections	CrossSection	List of Cross Section objects
Group	Text	Group identifier to select specific set of Cross Section
Family	Text	Family identifier to select specific set of Cross Section. The families are unique within a group.
Name	Text	Unique name of Cross Section within set specific to Group and Family.

4.4.3.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Cross Section	CrossSection	Cross Section selected.

4.4.3.4 Appearance

Figure 4.23 shows the component *Cross Section Picker (Pamela)* after placement on the Grasshopper document.

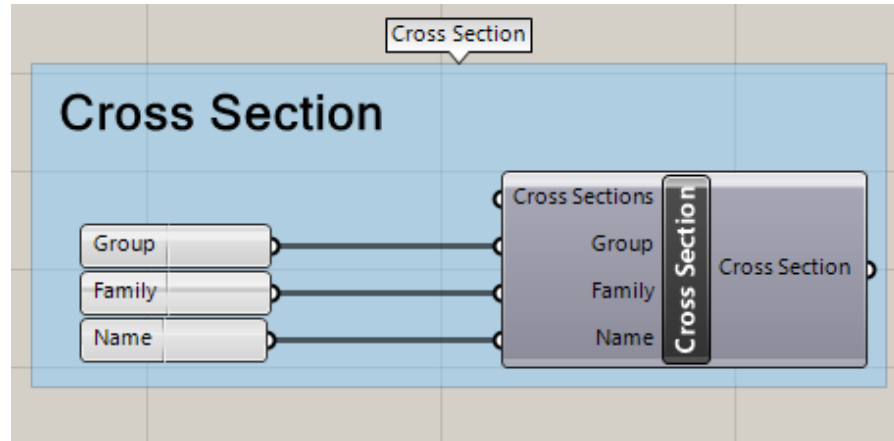



Figure 4.23: Screenshot of *Cross Section Picker (Pamela)*

4.4.4 Component: Material Picker (Pamela)

4.4.4.1 Synopsis

Icon	
Purpose	Select a material based on hierarchical grouping criterias
Nickname	Material
Type Name	Pamela.GH.Material.MaterialPicker_Component
Location	Material

4.4.4.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Materials	Material	List of Material objects
Group	Text	Group identifier to select specific set of Material
Family	Text	Family identifier to select specific set of Material. The families are unique within a group.
Name	Text	Unique name of Material within set specific to Group and Family.

4.4.4.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Material	Material	Material selected.

4.4.4.4 Appearance

Figure 4.24 shows the component *Material Picker (Pamela)* after placement on the Grasshopper document.

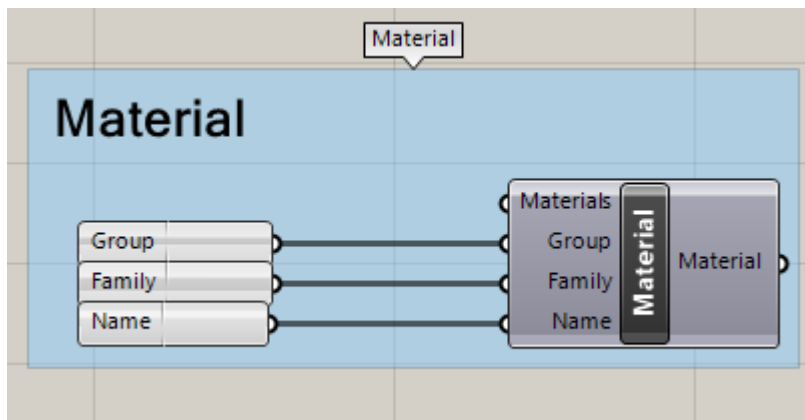



Figure 4.24: Screenshot of *Material Picker (Pamela)*

4.4.5 Component: Fabric Picker (Pamela)

4.4.5.1 Synopsis

Icon	
Purpose	Select a Membrane material (fabric) based on hierarchical grouping criterias.
Nickname	Membrane
Type Name	Pamela.GH.Material.MembraneMaterialPicker_Component
Location	Material

4.4.5.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Membrane Materials	Membrane Material	List of Membrane Material objects
Group	Text	Group identifier to select specific set of Membrane Material
Family	Text	Family identifier to select specific set of Membrane Material. The families are unique within a group.
Name	Text	Unique name of Membrane Material within set specific to Group and Family.

4.4.5.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Membrane Material	Membrane Material	Membrane Material selected.
Thickness	Number	Thickness per layer of the membrane material.

4.4.5.4 Additional Information

In contrast to standard materials, the thickness of membrane materials usually depends on the chosen material and cannot be changed. Therefore

the thickness is also stored in the material database and not in the database of the cross sections.

4.4.5.5 Appearance

Figure 4.25 shows the component *Fabric Picker (Pamela)* after placement on the Grasshopper document.

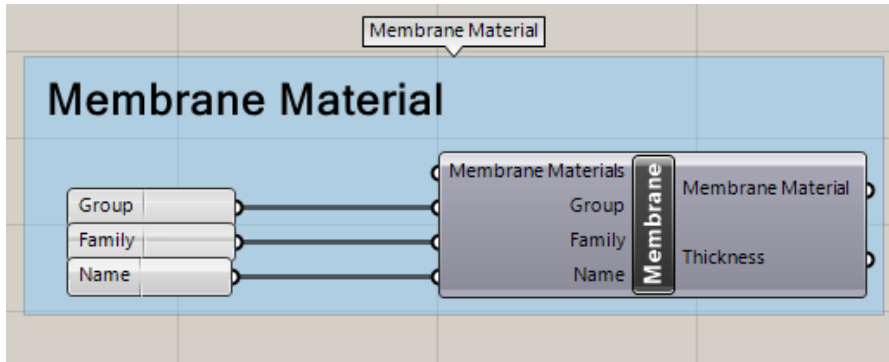



Figure 4.25: Screenshot of *Fabric Picker (Pamela)*

4.4.6 Component: Material (Pamela)

4.4.6.1 Synopsis

Icon	
Purpose	Material Properties.
Nickname	Material
Type Name	Pamela.GH.Material.Material_Component
Location	Material

4.4.6.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Group	Text	Toplevel grouping criteria for materials.
Family	Text	Second level grouping criteria for materials. The families must be unique within a group.
Name	Text	Specific name of Material
E	Number	Young's Modulus [kPa]
G	Number	Shear modulus G [kPa]
nu	Number	Poisson Constant []
gamma	Number	Specific weight [kg/m ³]
fy	Number	Yield strength [kPa]
alphaT	Number	Coefficient of thermal expansion [1/K]

4.4.6.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Material	Material	Material properties

4.4.6.4 *Appearance*

Figure 4.26 shows the component *Material (Pamela)* after placement on the Grasshopper document.

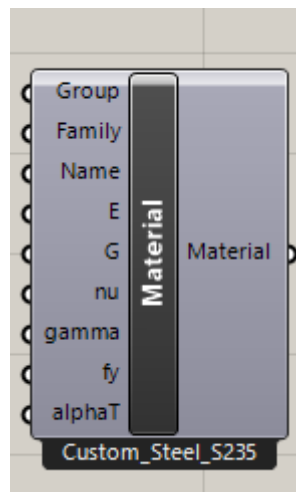



Figure 4.26: Screenshot of *Material (Pamela)*

4.4.7 Component: Membrane Material (Pamela)

4.4.7.1 Synopsis

Icon	
Purpose	Material properties for membranes.
Nickname	Membrane Material
Type Name	Pamela.GH.Material.MembraneMaterial_Component
Location	Material

4.4.7.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Group	Text	Toplevel grouping criteria for materials.
Family	Text	Second level grouping criteria for materials. The families must be unique within a group.
Name	Text	Specific name of Material
E warp	Number	Young's Modulus E in warp direction [kPa]
E weft	Number	Young's Modulus E in weft direction [kPa]
G warp	Number	Shear modulus G in warp direction [kPa]
G weft	Number	Shear modulus G in weft direction [kPa]
nu warp	Number	Poisson Constant nu in warp direction []
nu weft	Number	Poisson Constant nu in weft direction []
gamma	Number	Specific weight [kg/m ³]
Thickness	Number	Thickness in [m]
alphaT	Number	Coefficient of thermal expansion [1/K]

4.4.7.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Material	Membrane Material	Material properties

4.4.7.4 Appearance

Figure 4.27 shows the component *Membrane Material (Pamela)* after placement on the Grasshopper document.

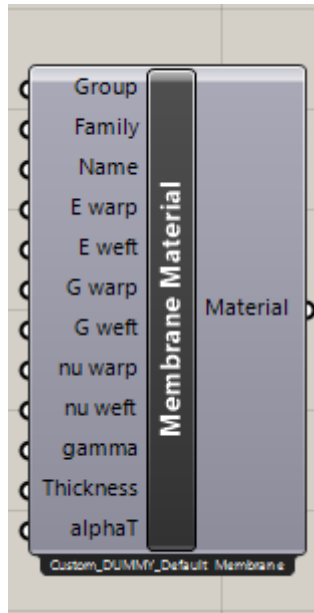



Figure 4.27: Screenshot of *Membrane Material (Pamela)*


4.4.8 Component: Cross Section Parameter (Pamela)

4.4.8.1 Synopsis

Icon	
Purpose	Parameter object for cross sections
Nickname	Cross Section
Type Name	Pamela.GH.Material.Param_CrossSection
Location	Material


4.4.9 Component: Material Parameter (Pamela)

4.4.9.1 Synopsis

Icon	
Purpose	Parameter object for isotropic material.
Nickname	Material
Type Name	Pamela.GH.Material.Param_Material
Location	Material

4.4.10 Component: Membrane Material Parameter (Pamela)

4.4.10.1 Synopsis

Icon	
Purpose	Parameter object for orthotropic materials like membrane fabrics
Nickname	Membrane Material
Type Name	Pamela.GH.Material.Param_MembraneMaterial
Location	Material

4.5 PANEL: MODEL DEFINITION

Figure 4.28 shows all icons of the components grouped under the panel *Model Definition*

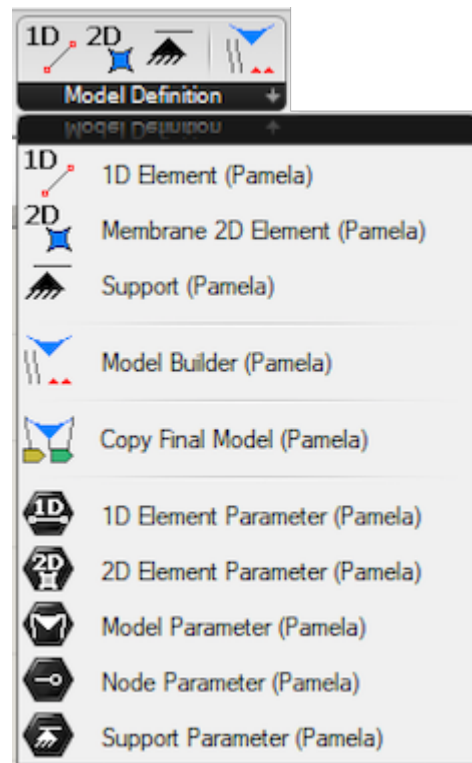



Figure 4.28: overview of panel *Model Definition*

4.5.1 Component: 1D Element (Pamela)

4.5.1.1 Synopsis

Icon	
Purpose	Create 1D Elements from lines and polylines.
Nickname	1D Element
Type Name	Pamela.GH.ModelDef.Element1D_Component
Location	Model Definition

4.5.1.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Polyline	Curve	Polyline or line.
pret	Number	Pretension to apply on the 1D element(s).
Behaviour	Integer	0 = unrestrained, 1 = fixed length, 2 = completely fixed. Used for FDM, irrelevant for FEM.
Type	Integer	1 = cable stiff, 2 = cable flexible, 3 = truss, 4 = beam. Required for FEM, irrelevant for FDM.
Material	Material	Material specification. Only used for FEM, ignored for FDM.
Cro-Sec	CrossSection	Cross Section specification. Only used for FEM, ignored for FDM.
Label	Text	Label for identification

4.5.1.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
1D Elements	1D-Element	1D Elements according to specified input values / polyline segments.
Pts	Point	List of distinct points of all 1D Elements.

4.5.1.4 Appearance

Figure 4.29 shows the component *1D Element (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

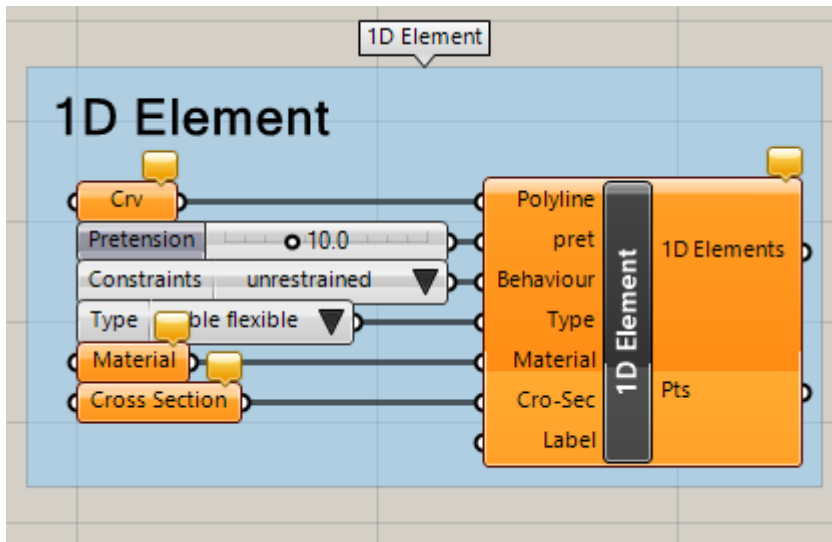



Figure 4.29: Screenshot of 1D Element (Pamela)

4.5.2 Component: Membrane 2D Element (Pamela)

4.5.2.1 Synopsis

Icon	
Purpose	Create 2D Elements from Mesh representing a Membrane.
Nickname	Membrane Element
Type Name	Pamela.GH.ModelDef.Element2DMembrane_Component
Location	Model Definition

4.5.2.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Mesh
Warp	Number	Membrane prestress in warp direction.
Weft	Number	Membrane prestress in weft direction.
Orientation	Generic Data	Assign a vector (or text) for warp direction. Use vector based on X and Y like "1.0, 0.0, 0.0" for sail-type structures and "0.0, 0.0, 1.0" (warp in Z) for conical-type structures.
Material	Membrane Material	Material specification. Only used for FEM, ignored for FDM.
Thickness	Number	Membrane Thickness in [m]. Required for FDM, ignored for FEM (taken from Material properties instead.)
Label	Text	Optional label for identification.

4.5.2.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
2D Elements	2D-Element	2D Elements according to specified input values

4.5.2.4 *Appearance*

Figure 4.30 shows the component *Membrane 2D Element (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

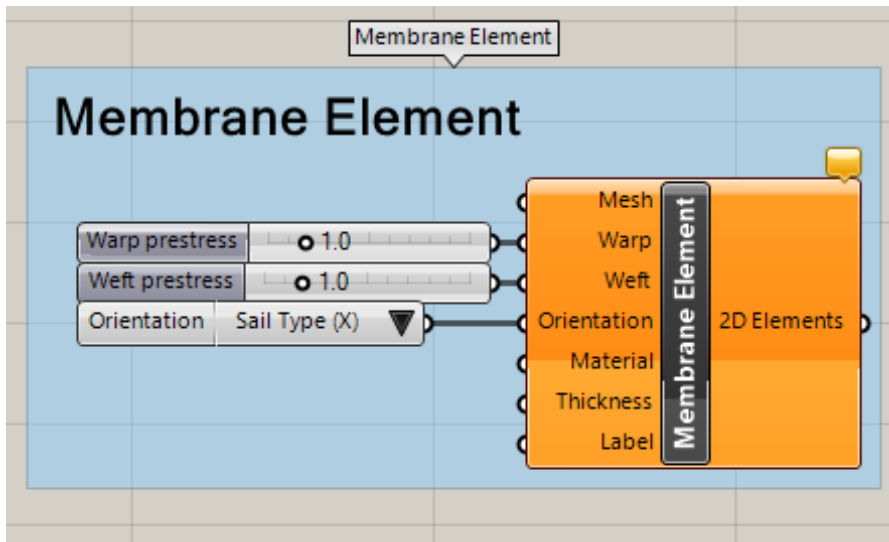


Figure 4.30: Screenshot of *Membrane 2D Element (Pamela)*

4.5.3 Component: Support (Pamela)

4.5.3.1 Synopsis

Icon	
Purpose	Creates supports / constraints at nodes (point coordinates). Drop-down menu lets you select translations/rotations which should be zero.
Nickname	Support
Type Name	Pamela.GH.ModelDef.Support_Component
Location	Model Definition

4.5.3.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Position	Generic Data	Input coordinates, accepts points as well as polylines.
Label	Text	Name / descriptive text
Orient	Plane	Initial support displacements and rotation. By default the global coordinate system is used.

4.5.3.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Support	Support	Resulting constraints.

4.5.3.4 *Menu items*

The following component menu items exist in addition to the standard drop down menu items:

- Tx fixed
- Ty fixed
- Tz fixed
- Rx fixed
- Ry fixed
- Rz fixed
- Fix all
- Free all

4.5.3.5 *Appearance*

Figure 4.31 shows the component *Support (Pamela)* after placement on the Grasshopper document.

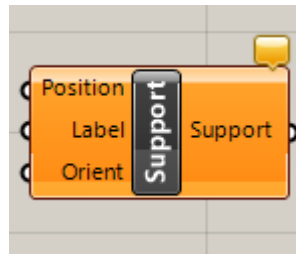



Figure 4.31: Screenshot of *Support (Pamela)*

4.5.4 Component: Model Builder (Pamela)

4.5.4.1 Synopsis

Icon	
Purpose	Assembles all inputs into a generic datamodel which can be used for formfinding or analysis.
Nickname	Model Builder
Type Name	Pamela.GH.ModelDef.ModelBuilder_Component
Location	Model Definition

4.5.4.2 Input Parameters

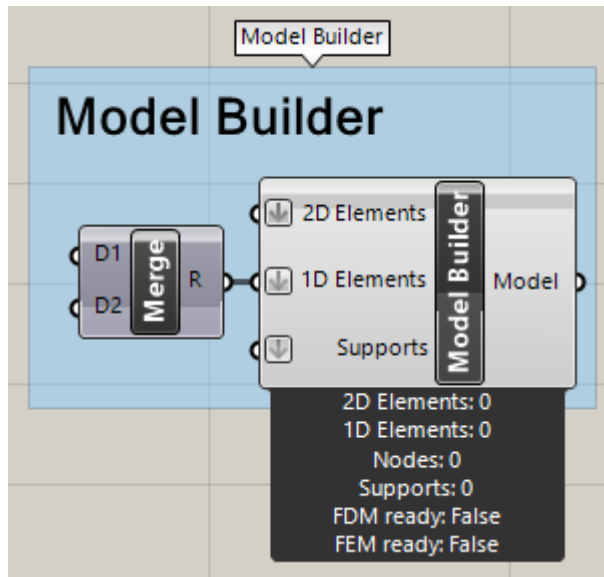
<i>ID</i>	<i>Type</i>	<i>Description</i>
2D Elements	2D-Element	Collection of all 2D Element definitions to include.
1D Elements	1D-Element	Collection of all 1D Elements definitions to include.
Supports	Support	Collection of all Supports definitions to include.

4.5.4.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	The complete Model ready to be used as input to FDM or FEM solver.


4.5.4.4 Appearance

Figure 4.32 shows the component *Model Builder (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

Figure 4.32: Screenshot of *Model Builder (Pamela)*

4.5.5 Component: Copy Final Model (Pamela)

4.5.5.1 Synopsis

Icon	
Purpose	Create a new model with input values copied from final values / displaced positions. Drop down menu options exist to select data to copy.
Nickname	Copy Final
Type Name	Pamela.GH.ModelDef.ModelCopy_Component
Location	Model Definition

4.5.5.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing Model to copy final state.

4.5.5.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	The new Model ready to be used as input to Pam solver for analysis.

4.5.5.4 *Menu items*

The following component menu items exist in addition to the standard drop down menu items:

- Copy Prestress Values

4.5.5.5 *Appearance*

Figure 4.33 shows the component *Copy Final Model (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

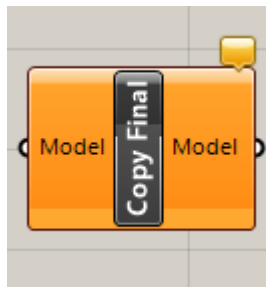



Figure 4.33: Screenshot of *Copy Final Model (Pamela)*


4.5.6 Component: 1D Element Parameter (Pamela)

4.5.6.1 *Synopsis*

Icon	
Purpose	Parameter object for 1D modeling elements based on lines.
Nickname	1D Element
Type Name	Pamela.GH.ModelDef.Param_Element1D
Location	Model Definition


4.5.7 Component: 2D Element Parameter (Pamela)

4.5.7.1 Synopsis

Icon	
Purpose	Parameter object for 2D modeling elements based on meshes.
Nickname	2D Element
Type Name	Pamela.GH.ModelDef.Param_Element2D
Location	Model Definition


4.5.8 Component: Model Parameter (Pamela)

4.5.8.1 Synopsis

Icon	
Purpose	Parameter object for assembled data models.
Nickname	Model
Type Name	Pamela.GH.ModelDef.Param_Model
Location	Model Definition


4.5.9 Component: Node Parameter (Pamela)

4.5.9.1 Synopsis

Icon	
Purpose	Parameter object for Node / point locations
Nickname	Node
Type Name	Pamela.GH.ModelDef.Param_Node
Location	Model Definition

4.5.10 Component: Support Parameter (Pamela)

4.5.10.1 Synopsis

Icon	
Purpose	Parameter object for Support definitions.
Nickname	Support
Type Name	Pamela.GH.ModelDef.Param_Support
Location	Model Definition

4.6 PANEL: MODEL LOADS

Figure 4.34 shows all icons of the components grouped under the panel *Model Loads*



Figure 4.34: overview of panel *Model Loads*

4.6.1 Component: Load Coefficients SUI (Pamela)

4.6.1.1 Synopsis

Icon	1,2,3 🇨🇭
Purpose	Coefficients for country specific load calculations for Switzerland according to SIA 261.
Nickname	Coeff
Type Name	Pamela.GH.ModelLoads.LoadCoeffSUI_Component
Location	Model Loads

4.6.1.2 Input Parameters

ID	Type	Description
Zone	Integer	Referenzzone für Staudruck gemäss SIA-261, Anhang E
Gelände	Integer	Geländekategorie gemäss SIA-261 6.2.1.2 Tabelle 4

4.6.1.3 Output Parameters

ID	Type	Description
qp0	Number	Staudruck qp0[kN/m ²]
zg	Number	Gradientenhöhe zg
alfa r	Number	Bodenrauigkeit alfa r

4.6.1.4 Appearance

Figure 4.35 shows the component *Load Coefficients SUI (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

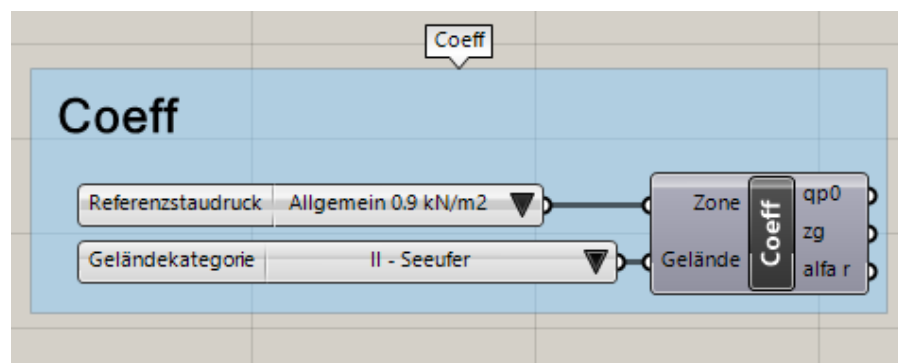



Figure 4.35: Screenshot of *Load Coefficients SUI (Pamela)*

4.6.2 Component: Load Case GER (Pamela)

4.6.2.1 Synopsis

Icon	
Purpose	Load case based on country specific coefficients and calculations for Germany (simplified approach).
Nickname	Load Case GER
Type Name	Pamela.GH.ModelLoads.LoadCaseGER_Component
Location	Model Loads

4.6.2.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Name	Text	Specific name of Load Case
g factor	Number	Factor to multiply dead load (self weight).
v factor	Number	Factor to multiply prestress.
Wind dir	Generic Data	Wind direction as Vector (or text).
Wind cp	Generic Data	Wind pressure coefficient (Strategy).
Wind	Number	Load from wind pressure in [kN/m ²]
w factor	Number	Factor to multiply wind load.
Snow	Number	Snow Load in [kN/m ²]
Snow mui	Generic Data	Roof shape coefficient (Strategy).
s factor	Number	Factor to multiply snow load.
Pressure	Number	Load from (internal) pressure in [kN/m ²]
p factor	Number	Factor to multiply pressure load.
Point	Point	Coordinates of point to apply load. Load is applied to node closest to specified point.
Point Load	Generic Data	Point load(s) as Vector (or text) in [kN]
pl factor	Number	Factor to multiply point load.

4.6.2.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
LoadCase	LoadCase	LoadCase properties

4.6.2.4 Appearance

Figure 4.36 shows the component *Load Case GER (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

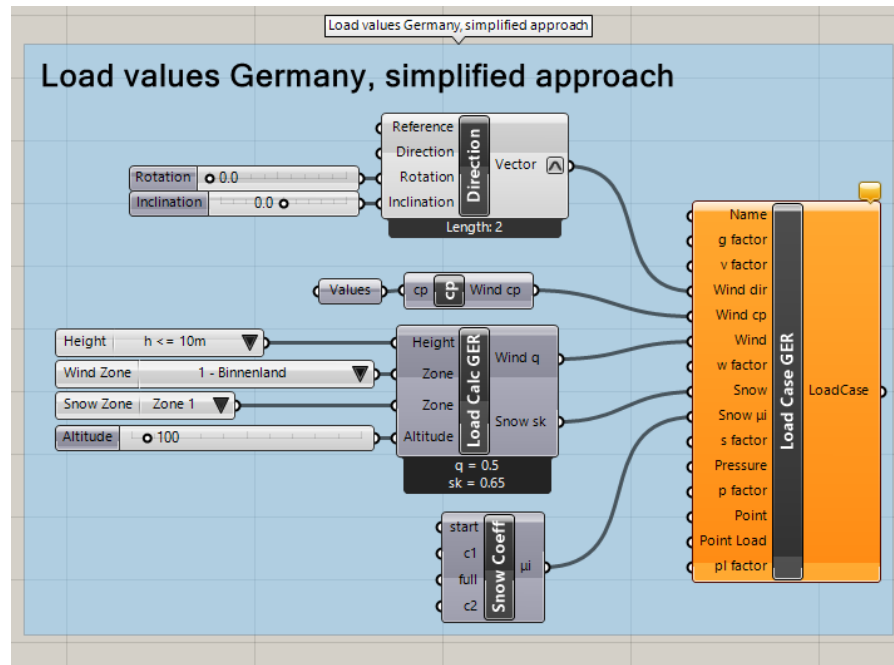


Figure 4.36: Screenshot of *Load Case GER (Pamela)*

4.6.3 Component: Load Case SUI (Pamela)

4.6.3.1 Synopsis

Icon	
Purpose	Load case based on country specific coefficients and calculations for Switzerland.
Nickname	Load Case SUI
Type Name	Pamela.GH.ModelLoads.LoadCaseSUI_Component
Location	Model Loads

4.6.3.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Name	Text	Specific name of Load Case
g factor	Number	Factor to multiply dead load (self weight).
v factor	Number	Factor to multiply prestress.
Wind dir	Generic Data	Wind direction as Vector (or text).
Wind cp	Generic Data	Wind pressure coefficient (Strategy).
Wind	Number	Load from wind pressure in [kN/m ²]
w factor	Number	Factor to multiply wind load.
Snow	Number	Snow Load in [kN/m ²]
Snow mui	Generic Data	Roof shape coefficient (Strategy).
s factor	Number	Factor to multiply snow load.
Pressure	Number	Load from (internal) pressure in [kN/m ²]
p factor	Number	Factor to multiply pressure load.
Point	Point	Coordinates of point to apply load. Load is applied to node closest to specified point.
Point Load	Generic Data	Point load(s) as Vector (or text) in [kN]
pl factor	Number	Factor to multiply point load.

4.6.3.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
LoadCase	LoadCase	LoadCase properties

4.6.3.4 *Appearance*

Figure 4.37 shows the component *Load Case SUI (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

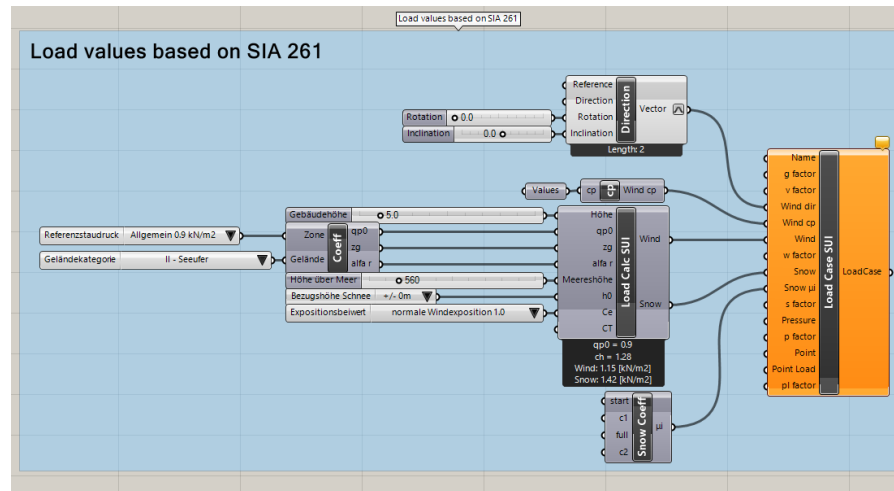


Figure 4.37: Screenshot of Load Case SUI (Pamela)

4.6.4 Component: Load Calculations GER (Pamela)

4.6.4.1 Synopsis

Icon	
Purpose	Country specific load calculations for Germany
Nickname	Load Calc GER
Type Name	Pamela.GH.ModelLoads.LoadCalcGER_Component
Location	Model Loads

4.6.4.2 Input Parameters

ID	Type	Description
Height	Number	Height of the building [m], <= 25m. Consider highest point (required for wind load)
Zone	Number	Reference zone for wind, simplified approach.
Zone	Integer	Reference zone for snow, simplified approach.
Altitude	Number	Altitude above sea level [m] (required for snow load)

4.6.4.3 Output Parameters

ID	Type	Description
Wind q	Number	Wind Pressure [kN/m ²]
Snow sk	Number	Snow Weight [kN/m ²]

4.6.4.4 Appearance

Figure 4.38 shows the component *Load Calculations GER (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

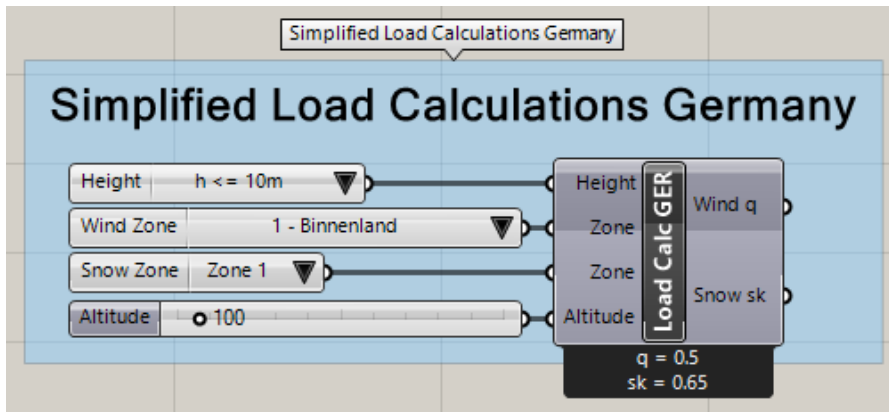


Figure 4.38: Screenshot of *Load Calculations GER (Pamela)*

4.6.5 Component: Load Calculations SUI (Pamela)

4.6.5.1 Synopsis

Icon	
Purpose	Country specific load calculations for Switzerland according to SIA 261.
Nickname	Load Calc SUI
Type Name	Pamela.GH.ModelLoads.LoadCalcSUI_Component
Location	Model Loads

4.6.5.2 Input Parameters

ID	Type	Description
Höhe	Number	Gebäudehöhe [m]. Consider highest point (required for wind load)
qp0	Number	Staudruck qp0 (required for wind load)[kN/m ²]
zg	Number	Gradientenhöhe zg (required for wind load)
alfa r	Number	Bodenrauigkeit alfa r (required for wind load)
Meereshöhe	Number	Höhe über Meer [m] (required for snow load)
ho	Number	Bezugshöhe [m] (required for snow load)
Ce	Number	Expositionsbeiwert (used for snow load)
CT	Number	Thermischer Beiwert (used for snow load)

4.6.5.3 Output Parameters

ID	Type	Description
Wind	Number	Wind Pressure [kN/m ²]
Snow	Number	Snow Weight [kN/m ²]

4.6.5.4 Appearance

Figure 4.39 shows the component *Load Calculations SUI (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

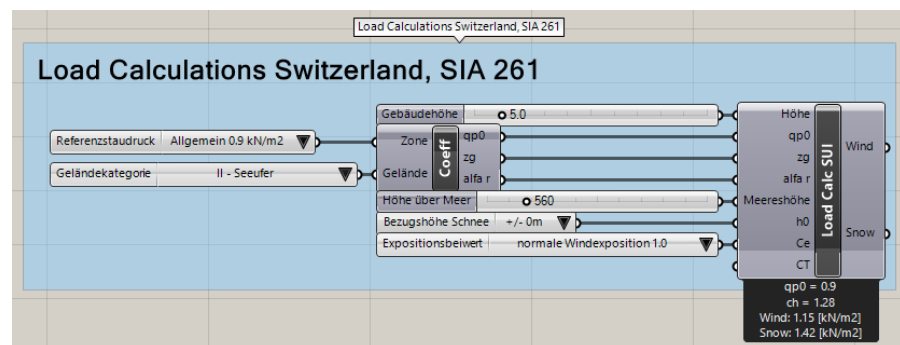



Figure 4.39: Screenshot of *Load Calculations SUI (Pamela)*

4.6.6 Component: Load Case (Pamela)

4.6.6.1 Synopsis

Icon	
Purpose	Load Case Properties.
Nickname	Load Case
Type Name	Pamela.GH.ModelLoads.LoadCase_Component
Location	Model Loads

4.6.6.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Name	Text	Specific name of Load Case
g factor	Number	Factor to multiply dead load (self weight).
v factor	Number	Factor to multiply prestress.
Wind dir	Generic Data	Wind direction as Vector (or text).
Wind cp	Generic Data	Wind pressure coefficient (Strategy).
Wind	Number	Load from wind pressure in [kN/m ²]
w factor	Number	Factor to multiply wind load.
Snow	Number	Snow Load in [kN/m ²]
Snow mui	Generic Data	Roof shape coefficient (Strategy).
s factor	Number	Factor to multiply snow load.
Pressure	Number	Load from (internal) pressure in [kN/m ²]
p factor	Number	Factor to multiply pressure load.
Point	Point	Coordinates of point to apply load. Load is applied to node closest to specified point.
Point Load	Generic Data	Point load(s) as Vector (or text) in [kN]
pl factor	Number	Factor to multiply point load.

4.6.6.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
LoadCase	LoadCase	LoadCase properties

4.6.6.4 Appearance

Figure 4.40 shows the component *Load Case (Pamela)* after placement on the Grasshopper document.

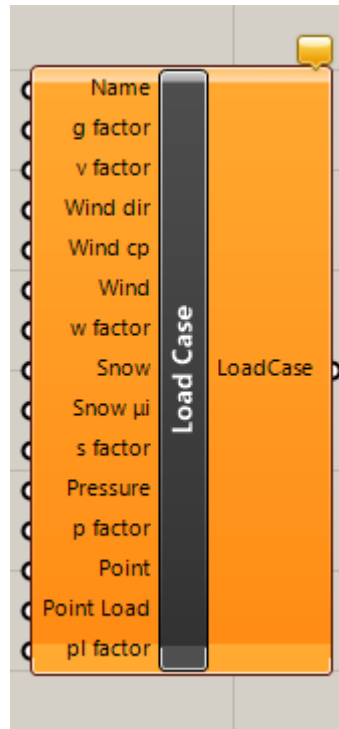



Figure 4.40: Screenshot of *Load Case (Pamela)*

4.6.7 Component: Membrane Snow Shape Coefficients (Pamela)

4.6.7.1 Synopsis

Icon	
Purpose	Generic evaluation of roof shape coefficients for snow loads on membranes based on angle between surface normal and Z-axis of horizontal plane.
Nickname	Snow Coeff
Type Name	Pamela.GH.ModelLoads.SnowRSgenericic_Component
Location	Model Loads

4.6.7.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
start	Number	Angle (degrees) between surface normal and Z-axis of horizontal plane when snow drift starts. For angles smaller than start no snow drift is considered.
c1	Number	Roof shape coefficient for start angle.
full	Number	Angle (degrees) for full snow drift. For angles larger than this full (c2) snow drift is considered.
c2	Number	Roof shape coefficient for full / complete snow drift.

4.6.7.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
mui	Generic Data	Roof shape coefficient (Strategy).

4.6.7.4 Appearance


Figure 4.41 shows the component *Membrane Snow Shape Coefficients (Pamela)* after placement on the Grasshopper document.



Figure 4.41: Screenshot of *Membrane Snow Shape Coefficients (Pamela)*

4.6.8 Component: Membrane Pressure Coefficients (Pamela)

4.6.8.1 Synopsis

Icon	
Purpose	Generic evaluation of wind pressure coefficients for membranes based on angle between surface normal and wind direction.
Nickname	cp
Type Name	Pamela.GH.ModelLoads.WindCPgeneric_Component
Location	Model Loads

4.6.8.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
cp	Text	Key/Value pairs structured as angle=cp for the angles from 0 to 180 degrees in steps of 10 degrees. Positive cp values for pressure, negative values for suction. For membranes negative values are usually in the direction of the surface normal, positive values in opposite direction.

4.6.8.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Wind cp	Generic Data	Wind pressure coefficient (Strategy).

4.6.8.4 Appearance

Figure 4.42 shows the component *Membrane Pressure Coefficients (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

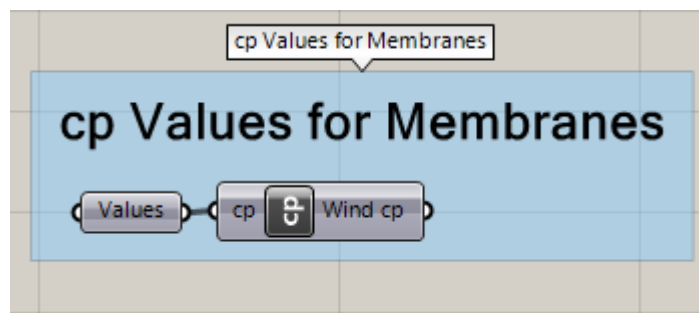


Figure 4.42: Screenshot of *Membrane Pressure Coefficients (Pamela)*

4.6.9 Component: Snowload (Pamela)

4.6.9.1 Synopsis

Icon	❄️❄️
Purpose	Calculation of snow load based on snow type and thickness.
Nickname	Snow
Type Name	Pamela.GH.ModelLoads.Snowload_Component
Location	Model Loads

4.6.9.2 Input Parameters

ID	Type	Description
Density	Number	Density according to type of snow. Wet snow is heavier. [kN/m ³]
Depth	Number	Depth or height of snow in [m].

4.6.9.3 Output Parameters

ID	Type	Description
Snow	Number	Snow Weight [kN/m ²]

4.6.9.4 Appearance

Figure 4.43 shows the component *Snowload (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

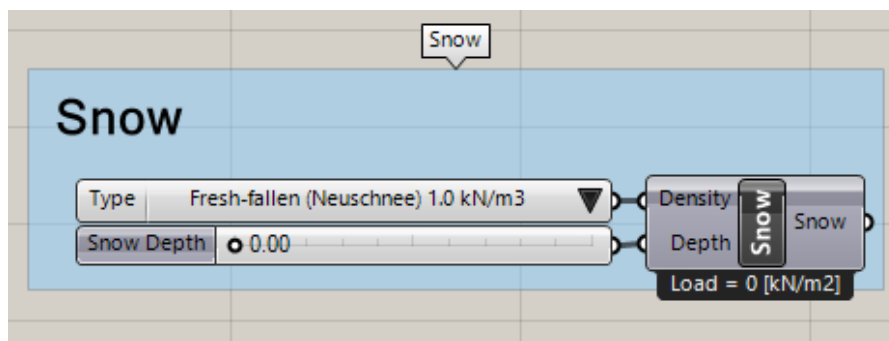



Figure 4.43: Screenshot of *Snowload (Pamela)*

4.6.10 Component: Windpressure (Pamela)

4.6.10.1 Synopsis

Icon	
Purpose	Windspeed to pressure conversion.
Nickname	Wind
Type Name	Pamela.GH.ModelLoads.Windpressure_Component
Location	Model Loads

4.6.10.2 Input Parameters

ID	Type	Description
Bf	Integer	Wind speed in Beaufort (max value)
m/s	Number	Wind speed in m/s
km/h	Number	Wind speed in km/h

4.6.10.3 Output Parameters

ID	Type	Description
Wind	Number	Maximal wind pressure from input values. [kN/m ²]

4.6.10.4 Appearance

Figure 4.44 shows the component *Windpressure (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

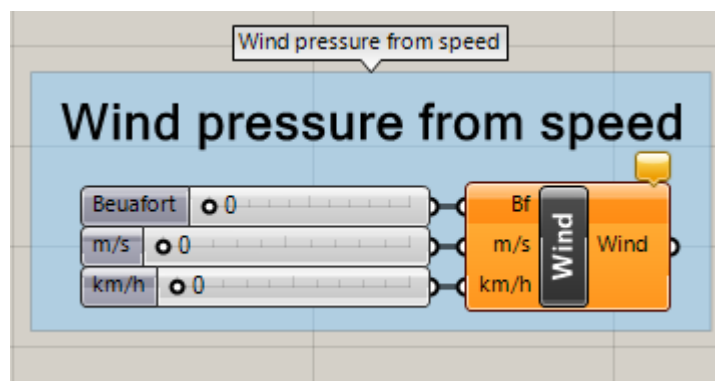



Figure 4.44: Screenshot of *Windpressure (Pamela)*

4.6.11 Component: Load Case Parameter (Pamela)

4.6.11.1 Synopsis

Icon	
Purpose	Load Case data for analysis.
Nickname	Load Case
Type Name	Pamela.GH.ModelLoads.Param_LoadCase
Location	Model Loads

4.7 PANEL: SOLVER

Figure 4.45 shows all icons of the components grouped under the panel *Solver*

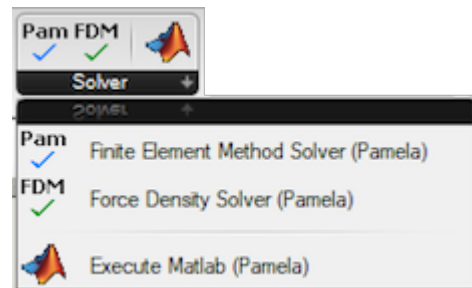


Figure 4.45: overview of panel *Solver*

4.7.1 Component: Force Density Solver (Pamela)

4.7.1.1 Synopsis

Icon	FDM ✓
Purpose	linear (one-step) form finding of tensile structures using generalized Force Density method(FDM). The generalization is based on rewriting the original equations of FDM to Finite Element(FE) environment and implementing also the surface finite elements. The final geometry is obtained by solving the system of linear equations of the form $u = \text{inv}(K) * F$. Where u is the final position of nodes, K is the system stiffness matrix and F is the external loading and influence of the supports.
Nickname	FDM Solver
Type Name	Pamela.GH.Solver.FDMSolver_Component
Location	Solver

4.7.1.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	The completely assembled model to use for computing.
Pause	Boolean	Pause Solver while changing values of preceding components to avoid unnecessary heavy computations.

4.7.1.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Model with results.

4.7.1.4 Menu items

The following component menu items exist in addition to the standard drop down menu items:

- Tolerance
- Iterations

4.7.1.5 Appearance

Figure 4.46 shows the component *Force Density Solver (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

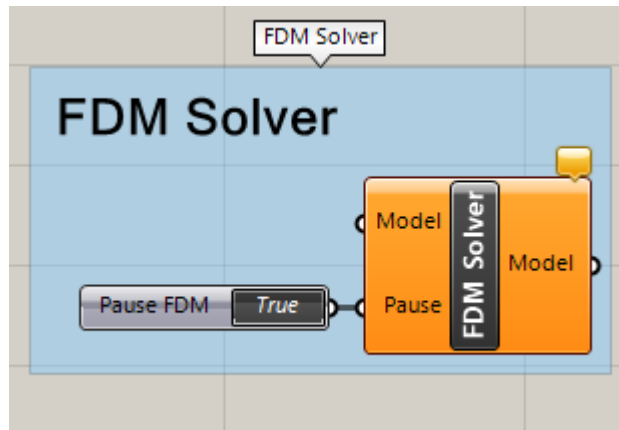


Figure 4.46: Screenshot of *Force Density Solver (Pamela)*

4.7.2 Component: Finite Element Method Solver (Pamela)

4.7.2.1 Synopsis

Icon	Pam ✓
Purpose	Form finding and analysis of tensile structures using Finite Element Method (FEM) or alternatively the Force Density Method (FDM).
Nickname	Pam Solver
Type Name	Pamela.GH.Solver.PamSolver_Component
Location	Solver

4.7.2.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	The completely assembled model to use for computing.
Load Cases	LoadCase	Load Case definitions.
Mode	Number	Factor to apply to the material property (Young's modulus)
Pause	Boolean	Pause Solver while changing values of preceding components to avoid unnecessary heavy computations.
LC Combo	Integer	Select the load cases to apply.

4.7.2.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model(s)	Model	All Model(s) with results. If load cases were selected, additional output parameters are created, one per active load case.

4.7.2.4 *Menu items*

The following component menu items exist in addition to the standard drop down menu items:

- Tolerance
- Iterations

4.7.2.5 *Appearance*

Figure 4.47 shows the component *Finite Element Method Solver (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

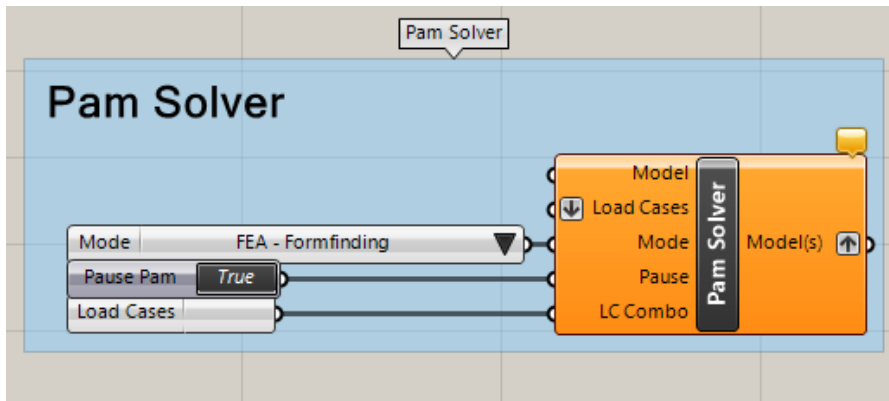



Figure 4.47: Screenshot of *Finite Element Method Solver (Pamela)*

4.7.3 Component: Execute Matlab (Pamela)

4.7.3.1 Synopsis

Icon	
Purpose	Execute Matlab
Nickname	Matlab
Type Name	Pamela.GH.Solver.ExecuteMatlab_Component
Location	Solver

4.7.3.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Matlab	Text	Full path to the Matlab executable
Workdir	Text	Path to the directory of the matlab script file
Script	Text	Name of the script to execute. Must be placed in working directory
Arg	Text	Arguments to pass to the executable
Exit	Boolean	Set to true to automatically close Matlab execution window when script terminates.
now!	Boolean	Set to true to trigger write data to the file.

4.7.3.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Trigger	Boolean	Is set to true once Matlab has finished. False when input trigger is false.

4.7.3.4 Appearance

Figure 4.48 shows the component *Execute Matlab (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

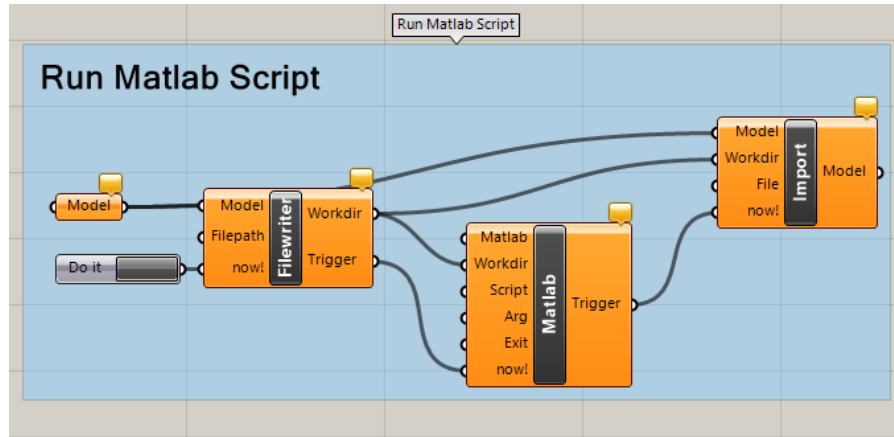


Figure 4.48: Screenshot of *Execute Matlab (Pamela)*

4.8 PANEL: VIEWS

Figure 4.49 shows all icons of the components grouped under the panel *Views*

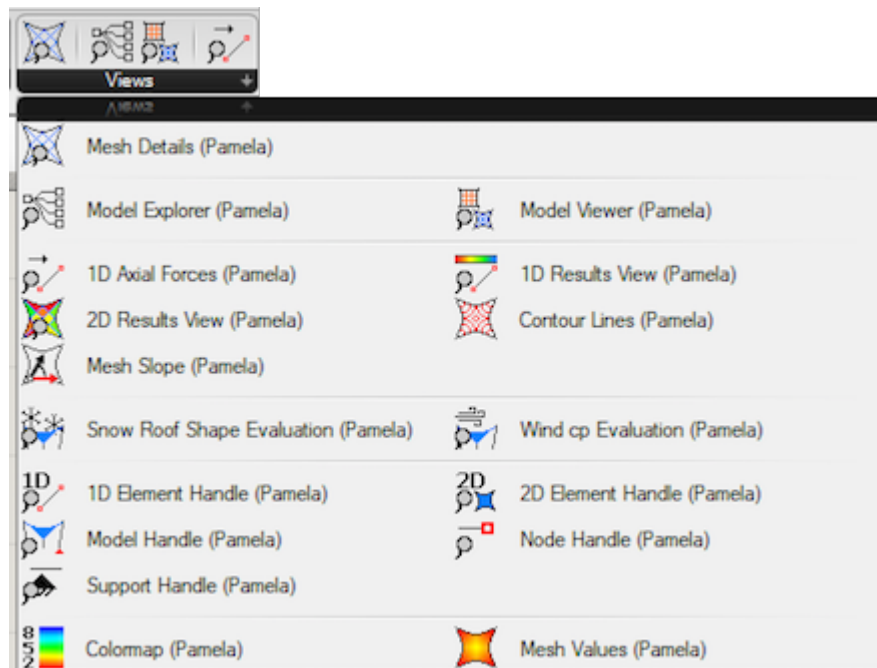



Figure 4.49: overview of panel *Views*

4.8.1 Component: Mesh Details (Pamela)

4.8.1.1 Synopsis

Icon	
Purpose	Break up Mesh into its components
Nickname	Mesh Details
Type Name	Pamela.GH.ResultViews.MeshDetails_Component
Location	Views

4.8.1.2 Input Parameters

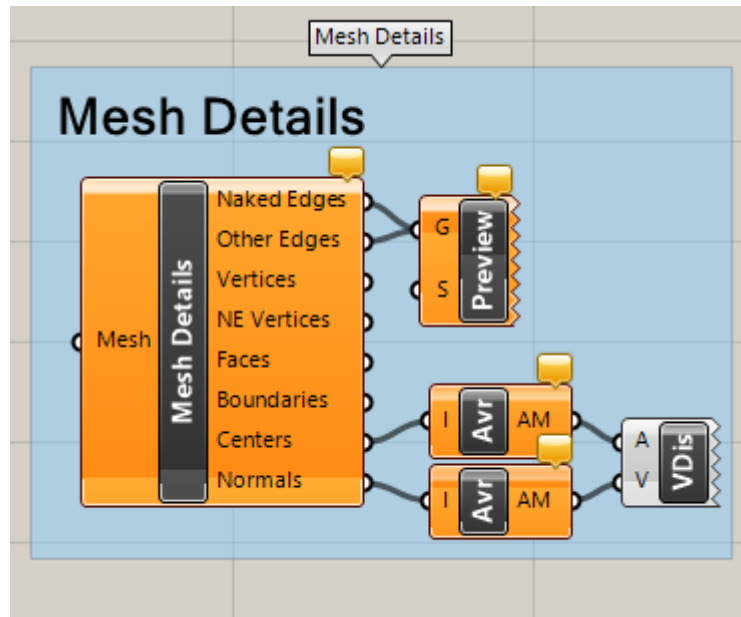
<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Input mesh

4.8.1.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Naked Edges	Line	Edges with valence 1 (a single adjacent face)
Other Edges	Line	Edges with valence 2 or higher (multiple adjacent faces)
Vertices	Point	All mesh vertices
NE Vertices	Point	Mesh vertices on naked edges
Faces	Mesh face	Faces - indices of vertices
Boundaries	Curve	Boundary polyline for each mesh face
Centers	Point	Center-points of all faces
Normals	Vector	Normal vectors for all faces

4.8.1.4 Appearance

Figure 4.50 shows the component *Mesh Details (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

Figure 4.50: Screenshot of *Mesh Details* (Pamela)

4.8.2 Component: Model Explorer (Pamela)

4.8.2.1 Synopsis

Icon	
Purpose	Decomposes the model into smaller element definitions.
Nickname	Model Explorer
Type Name	Pamela.GH.ResultViews.ModelExplorer_Component
Location	Views

4.8.2.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing Model to extend with additional definitions.

4.8.2.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
2D Elements	2D-Element	All 2D Elements matching filter criterias.
1D Elements	1D-Element	All 1D Elements matching filter criterias.
Nodes	Node	All Nodes matching filter criterias.
Supports	Support	All Supports matching filter criterias.
Copy	Model	A new Model with input values copied from final values / displaced positions.

4.8.2.4 Appearance

Figure 4.51 shows the component *Model Explorer (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

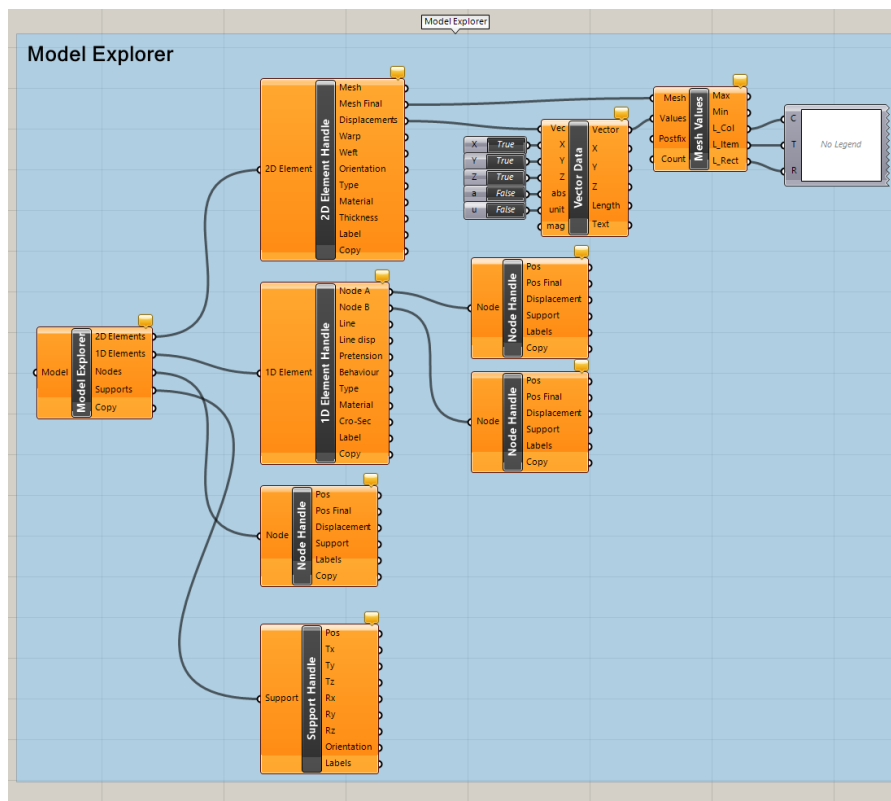



Figure 4.51: Screenshot of *Model Explorer (Pamela)*

4.8.3 Component: Model Viewer (Pamela)

4.8.3.1 Synopsis

Icon	
Purpose	Provides easy access to the geometrical elements of a Model for results visualization.
Nickname	Model Viewer
Type Name	Pamela.GH.ResultViews.ModelViewer_Component
Location	Views

4.8.3.2 Input Parameters

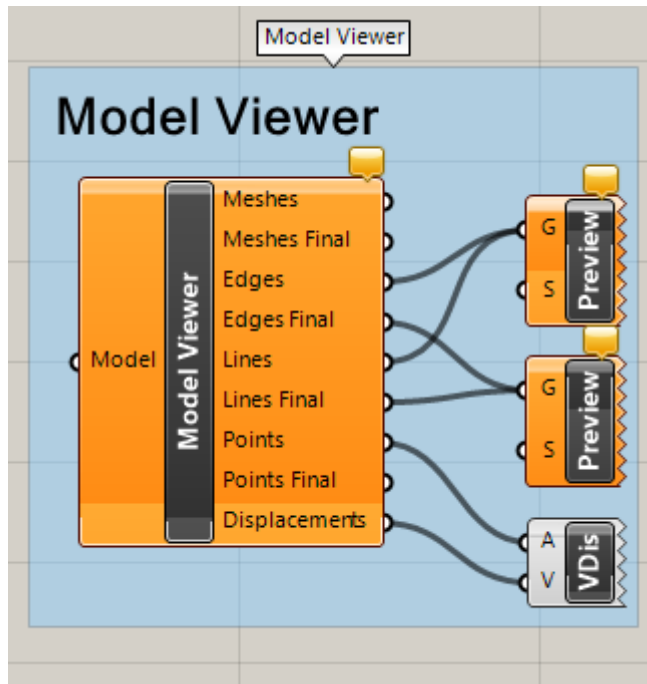
<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing Model to analyze / visualize.

4.8.3.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Meshes	Mesh	Initial input meshes.
Meshes Final	Mesh	Meshes after formfinding / analysis.
Edges	Line	Edges from initial meshes.
Edges Final	Line	Edges from mesh after formfinding / analysis.
Lines	Line	Linesegment with initial properties
Lines Final	Line	Linesegment with displaced position and final properties
Points	Point	Coordinates of ALL initial positions (Point3d)
Points Final	Point	Coordinates of ALL displaced / final position (Point3d)
Displacements	Vector	Vector3d of displacement for ALL nodes.


4.8.3.4 Appearance

Figure 4.52 shows the component *Model Viewer (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

Figure 4.52: Screenshot of *Model Viewer (Pamela)*

4.8.4 Component: Contour Lines (Pamela)

4.8.4.1 Synopsis

Icon	
Purpose	Draw a variable number of contour lines on a Mesh, Brep or Surface
Nickname	Contour Lines
Type Name	Pamela.GH.ResultViews.ContourLines_Component
Location	Views

4.8.4.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Meshes	Generic Data	Meshes, Surfaces or Breps to draw contour lines on.
Count	Integer	Number of Lines to draw on each input Mesh, Surface or Brep.
Lowest	Integer	Lowest / last line to draw.
Highest	Integer	Highest / first line to draw.

4.8.4.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
all	Curve	All contour lines.
closed	Curve	Closed contour lines. Check for ponding.

4.8.4.4 Menu items

The following component menu items exist in addition to the standard drop down menu items:

- Preview Color
- Thickness

4.8.4.5 Appearance

Figure 4.53 shows the component *Contour Lines (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

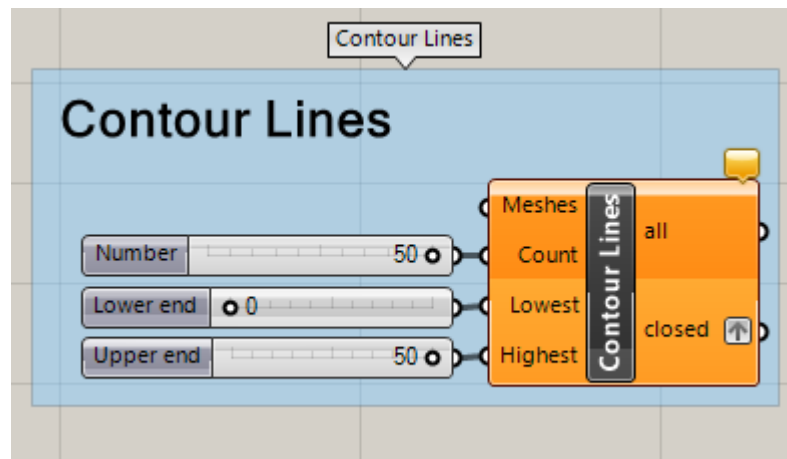



Figure 4.53: Screenshot of *Contour Lines (Pamela)*

4.8.5 Component: 1D Axial Forces (Pamela)

4.8.5.1 Synopsis

Icon	
Purpose	Visualization of resulting axial forces for 1D Elements.
Nickname	1D Axial
Type Name	Pamela.GH.ResultViews.Element1DAxialForces_Component
Location	Views

4.8.5.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing model(s) to choose data to analyze.
Index	Integer	Model to analyze, in case multiple models are present. 0 based Index.

4.8.5.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Lines	Line	Line elements at displaced positions
Plane	Plane	Reference plane of the results vector.
Values	Number	Axial forces according to selected data.
Units	Text	Units according to selected data, can be used for labels.
Text	Text	Text according to selected mode, can be used to label legends.

4.8.5.4 Appearance

Figure 4.54 shows the component *1D Axial Forces (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

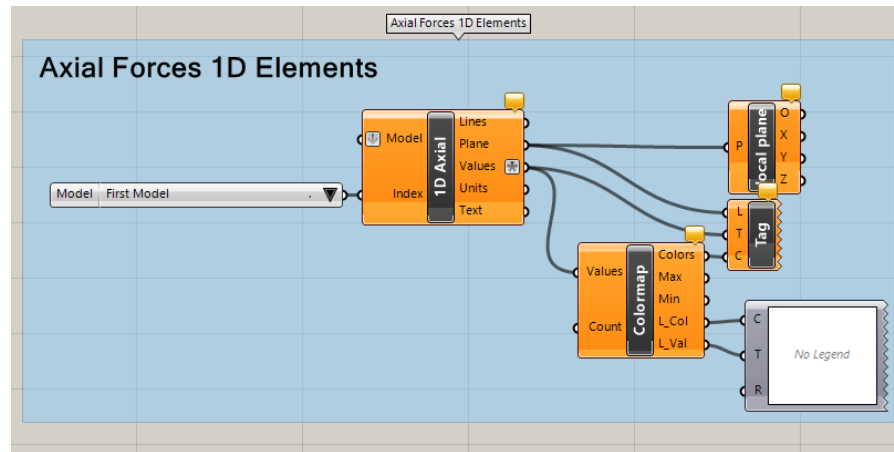


Figure 4.54: Screenshot of 1D Axial Forces (Pamela)

4.8.6 Component: Mesh Slope (Pamela)

4.8.6.1 Synopsis

Icon



Purpose

Slope or pitch analysis for a mesh. Can be used to detect possible ponding issues.

Nickname Mesh Slope

Type Name Pamela.GH.ResultViews.MeshSlope_Component

Location Views

4.8.6.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Generic Data	Mesh to analyze slope or inclination
Mode	Integer	0 = pitch in degrees, 1 = slope in %

4.8.6.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Generic Data	Analyzed mesh, unmodified.
Values	Number	Slope or pitch values for each face of the mesh.
Units	Text	Units according to selected mode, can be used for labels.
Text	Text	Text according to selected mode, can be used to label legends.

4.8.6.4 Appearance

Figure 4.55 shows the component *Mesh Slope (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

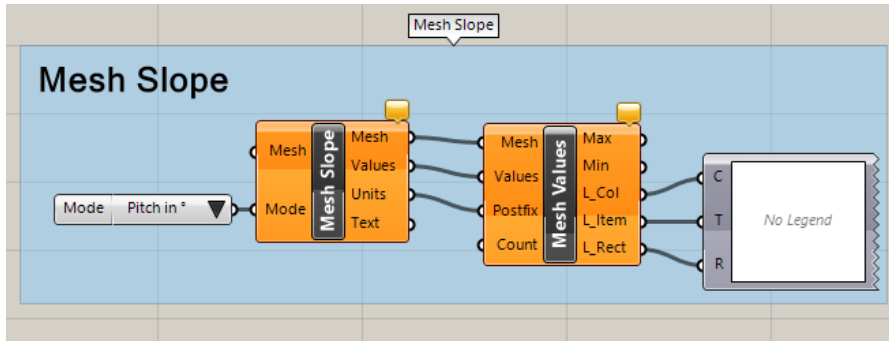



Figure 4.55: Screenshot of *Mesh Slope (Pamela)*

4.8.7 Component: 1D Results View (Pamela)

4.8.7.1 Synopsis

Icon	
Purpose	Visualization of resulting forces and moments for 1D Elements.
Nickname	1D Results
Type Name	Pamela.GH.ResultViews.Model1DResults_Component
Location	Views

4.8.7.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing model(s) to choose data to analyze.
Index	Integer	Model to analyze, in case multiple models are present. 0 based Index.
Scale	Number	Factor to scale displacements. Impacts the lines output.
Data	Integer	Type of values to analyze and visualize. 0 based Index.

4.8.7.3 Output Parameters

ID	Type	Description
Lines	Line	Line elements at displaced positions
Plane	Plane	Reference plane of the results vector.
Type	Integer	Structure of result values. 0 = single value, 1 = vector with 3 components.
Values	Vector	Values according to selected data, in local coordinates of the element
Units	Text	Units according to selected data, can be used for labels.
Text	Text	Text according to selected mode, can be used to label legends.

4.8.7.4 Appearance

Figure 4.56 shows the component *1D Results View (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

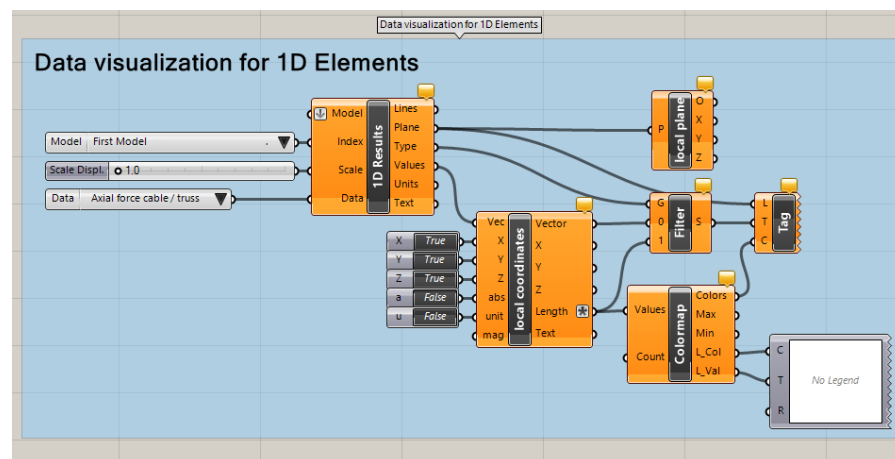



Figure 4.56: Screenshot of *1D Results View (Pamela)*

4.8.8 Component: 2D Results View (Pamela)

4.8.8.1 Synopsis

Icon	
Purpose	Visualization of resulting stresses and other values for 2D Elements.
Nickname	2D Results
Type Name	Pamela.GH.ResultViews.Model2DResults_Component
Location	Views

4.8.8.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing model(s) to choose mesh to analyze.
Index	Integer	Mesh to analyze, in case multiple models are present. 0 based Index.
Mesh	Integer	Mesh to analyze, in case multiple meshes are present. 0 based Index.
Data	Integer	Type of values to analyze and visualize. 0 based Index.

4.8.8.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Generic Data	Analyzed mesh, unmodified.
Values	Number	Numeric data value for each face or vertex of the mesh.
Units	Text	Units according to selected mode, can be used for labels.
Text	Text	Text according to selected data, can be used to label legends.

4.8.8.4 Appearance

Figure 4.57 shows the component *2D Results View (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

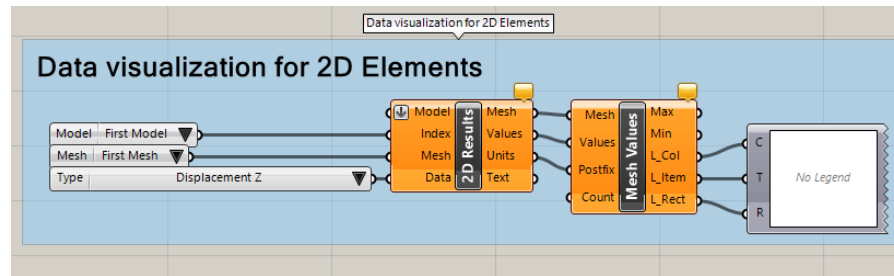



Figure 4.57: Screenshot of *2D Results View (Pamela)*

4.8.9 Component: Snow Roof Shape Evaluation (Pamela)

4.8.9.1 Synopsis

Icon	
Purpose	Evaluates the snow roof shape coefficients based on the provided strategy (algorithm).
Nickname	Roof Shape
Type Name	Pamela.GH.ResultViews.SnowRSEvaluation_Component
Location	Views

4.8.9.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Input mesh to evaluate values.
Snow mui	Generic Data	Roof shape coefficient (Strategy).

4.8.9.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Analyzed mesh, unmodified.
mui Values	Number	Snow roof shape coefficients per Mesh Face

4.8.9.4 Appearance

Figure 4.58 shows the component *Snow Roof Shape Evaluation (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

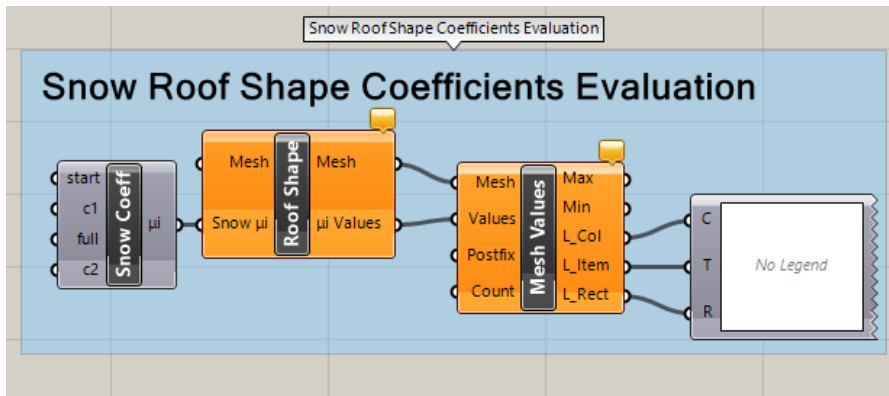



Figure 4.58: Screenshot of *Snow Roof Shape Evaluation (Pamela)*

4.8.10 Component: Wind cp Evaluation (Pamela)

4.8.10.1 Synopsis

Icon	
Purpose	Evaluates the wind cp values based on the provided strategy (algorithm).
Nickname	Wind cp
Type Name	Pamela.GH.ResultViews.WindCPEvaluation_Component
Location	Views

4.8.10.2 Input Parameters

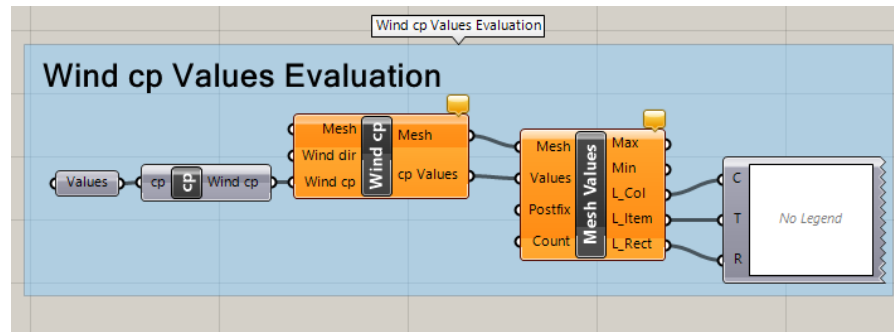
<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Input mesh to evaluate values.
Wind dir	Generic Data	Wind direction as Vector (or text).
Wind cp	Generic Data	Wind pressure coefficient (Strategy).

4.8.10.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Analyzed mesh, unmodified.
cp Values	Number	Wind pressure coefficients per Mesh Face


4.8.10.4 Appearance

Figure 4.59 shows the component *Wind cp Evaluation (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

Figure 4.59: Screenshot of *Wind cp Evaluation (Pamela)*

4.8.11 Component: 1D Element Handle (Pamela)

4.8.11.1 Synopsis

Icon	
Purpose	Decompose 1D Elements into common Rhino data types for further use.
Nickname	1D Element Handle
Type Name	Pamela.GH.ResultViews.Element1DHandle_Component
Location	Views

4.8.11.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
1D Element	1D-Element	Existing 1D Element definition.

4.8.11.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Node A	Node	Node denoting start of line.
Node B	Node	Node denoting end of line.
Line	Line	Linesegment with initial properties
Line disp	Line	Linesegment with displaced position and final properties
Pretension	Number	Originally assigned stress.
Behaviour	Number	0 = unrestrained, 1 = fixed length, 2 = completely fixed
Type	Integer	1 = cable, 2 = truss, 3 = beam, 4 = geodesicLine
Material	Material	Material properties
Cro-Sec	CrossSection	Cross Section properties
Label	Text	Label previously assigned to this element.
Copy	1D-Element	A new Element with input values copied from final values / displaced positions.

4.8.11.4 *Appearance*

Figure 4.60 shows the component *1D Element Handle (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

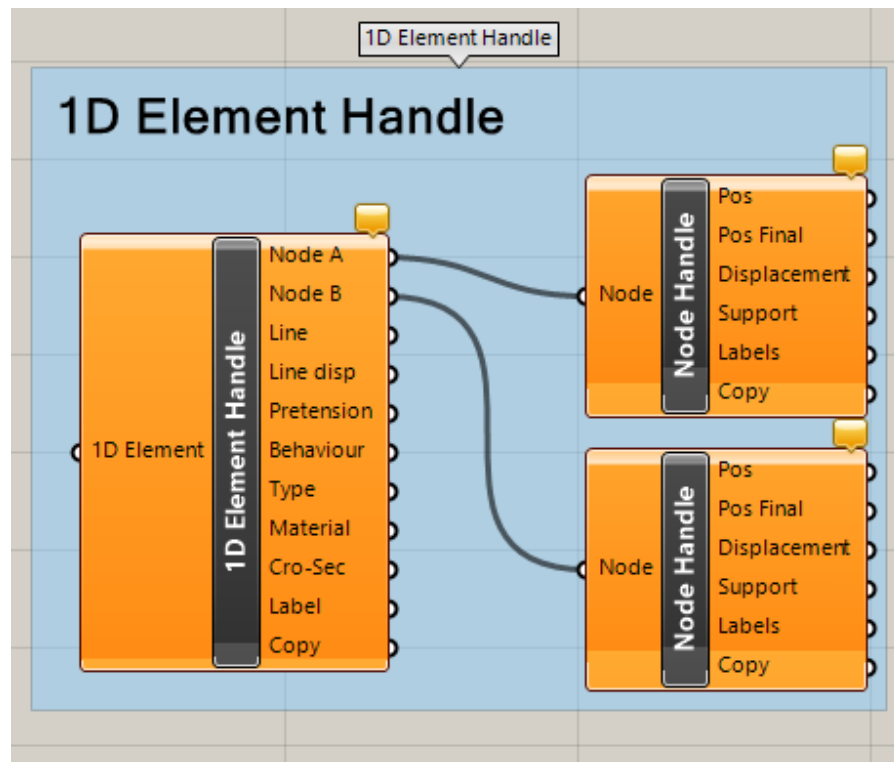



Figure 4.60: Screenshot of *1D Element Handle* (Pamela)

4.8.12 Component: 2D Element Handle (Pamela)

4.8.12.1 Synopsis

Icon	
Purpose	Decompose 1D Elements into common Rhino data types for further use.
Nickname	2D Element Handle
Type Name	Pamela.GH.ResultViews.Element2DHandle_Component
Location	Views

4.8.12.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
2D Element	2D-Element	Existing 2D Element definition.

4.8.12.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Initial input mesh.
Mesh Final	Mesh	Mesh after formfinding / analysis.
Displacements	Vector	Vector3d of displacement for ALL nodes.
Warp	Number	Membrane prestress (ratio) assigned in warp direction.
Weft	Number	Membrane prestress (ratio) assigned in weft direction.
Orientation	Vector	Warp direction assigned.
Type	Integer	1 = membrane, 2 = shell
Material	Membrane Material	Material properties
Thickness	Number	Membrane thickness in [m]
Label	Text	Label previously assigned to this element.
Copy	2D-Element	A new element with input values copied from final values / displaced positions.

4.8.12.4 Appearance

Figure 4.61 shows the component *2D Element Handle (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

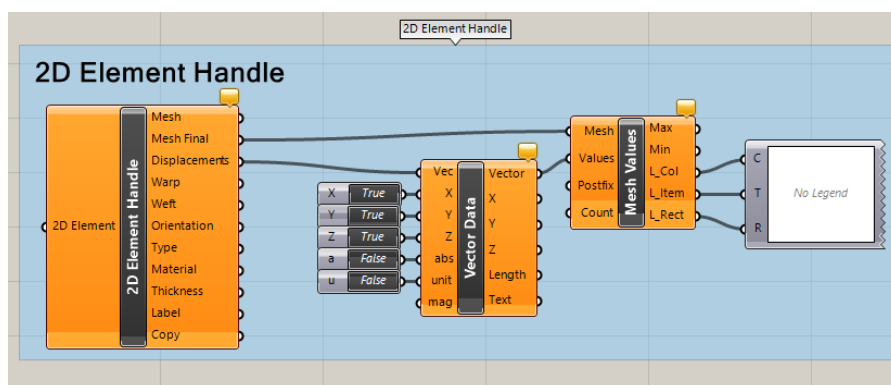



Figure 4.61: Screenshot of *2D Element Handle (Pamela)*

4.8.13 Component: Model Handle (Pamela)

4.8.13.1 Synopsis

Icon	
Purpose	Decomposes the model into smaller element definitions.
Nickname	Model Handle
Type Name	Pamela.GH.ResultViews.ModelHandle_Component
Location	Views

4.8.13.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Existing Model to extend with additional definitions.

4.8.13.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
2D Elements	2D-Element	All 2D Elements matching filter criterias.
1D Elements	1D-Element	All 1D Elements matching filter criterias.
Nodes	Node	All Nodes matching filter criterias.
Supports	Support	All Supports matching filter criterias.
Copy	Model	A new Model with input values copied from final values / displaced positions.

4.8.13.4 Appearance

Figure 4.62 shows the component *Model Handle (Pamela)* after placement on the Grasshopper document.

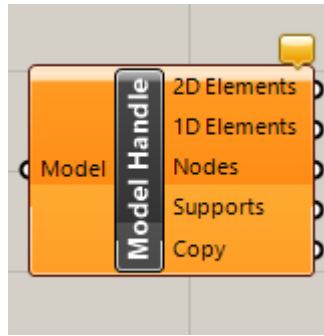



Figure 4.62: Screenshot of *Model Handle (Pamela)*

4.8.14 Component: Node Handle (Pamela)

4.8.14.1 Synopsis

Icon	
Purpose	Decompose Node into common Rhino data types for further use.
Nickname	Node Handle
Type Name	Pamela.GH.ResultViews.NodeHandle_Component
Location	Views

4.8.14.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Node	Node	Existing Node definition.

4.8.14.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Pos	Point	Coordinates of initial position (Point3d)
Pos Final	Point	Coordinates of displaced / final position (Point3d)
Displacement	Vector	Vector3d of Displacement
Support	Support	Support condition assigned to this Node
Labels	Text	Labels previously assigned to other objects containing this node
Copy	Node	A new Node with input values copied from final values / displaced position.

4.8.14.4 Appearance

Figure 4.63 shows the component *Node Handle (Pamela)* after placement on the Grasshopper document.

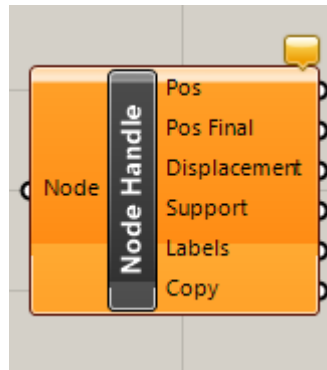


Figure 4.63: Screenshot of *Node Handle (Pamela)*

4.8.15 Component: Support Handle (Pamela)

4.8.15.1 Synopsis

Icon	
Purpose	Decompose Support into common Rhino data types for further use.
Nickname	Support Handle
Type Name	Pamela.GH.ResultViews.SupportHandle_Component
Location	Views

4.8.15.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Support	Support	Existing support definition.

4.8.15.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Pos	Point	Coordinates of location (Point3d)
Tx	Boolean	Indication if displacement in X direction is prevented.
Ty	Boolean	Indication if displacement in Y direction is prevented.
Tz	Boolean	Indication if displacement in Z direction is prevented.
Rx	Boolean	Indication if rotation around X-axis is prevented.
Ry	Boolean	Indication if rotation around Y-axis is prevented.
Rz	Boolean	Indication if rotation around Z-axis is prevented.
Orientation	Plane	Plane of orientation.
Labels	Text	Label previously assigned to the node

4.8.15.4 *Appearance*

Figure 4.64 shows the component *Support Handle (Pamela)* after placement on the Grasshopper document.

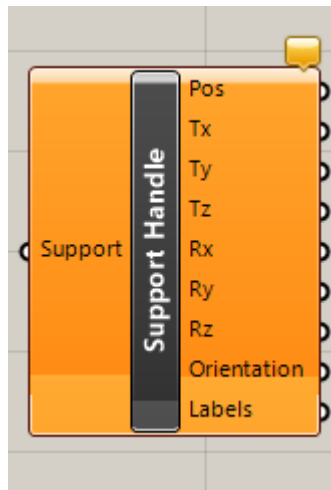



Figure 4.64: Screenshot of *Support Handle (Pamela)*

4.8.16 Component: Colormap (Pamela)

4.8.16.1 Synopsis

Icon	
Purpose	Create a list of colours according to the list of input values. In addition, colors and tags for a legend are provided.
Nickname	Colormap
Type Name	Pamela.GH.ResultViews.Colourmap_Component
Location	Views

4.8.16.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Values	Number	Values to generate colors for.
Count	Integer	Number of items to generate for the legend. If necessary, the number is adjusted to be <= the number of data values provided.

4.8.16.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Colors	Colour	Colors representing the data items
Max	Number	Maximum value of the input data.
Min	Number	Minimum value of the input data.
L_Col	Colour	Colors for the legend items.
L_Val	Number	Values for the legend items.

4.8.16.4 Menu items

The following component menu items exist in addition to the standard drop down menu items:

- Blue-Magenta-Yellow
- Blue-Green-Yellow-White
- Blue-Green-Yellow-Magenta-Red

4.8.16.5 Appearance

Figure 4.65 shows the component *Colormap (Pamela)* and all embedded components and parameters after placement on the Grasshopper

document. The components are automatically grouped and the group is labelled accordingly.

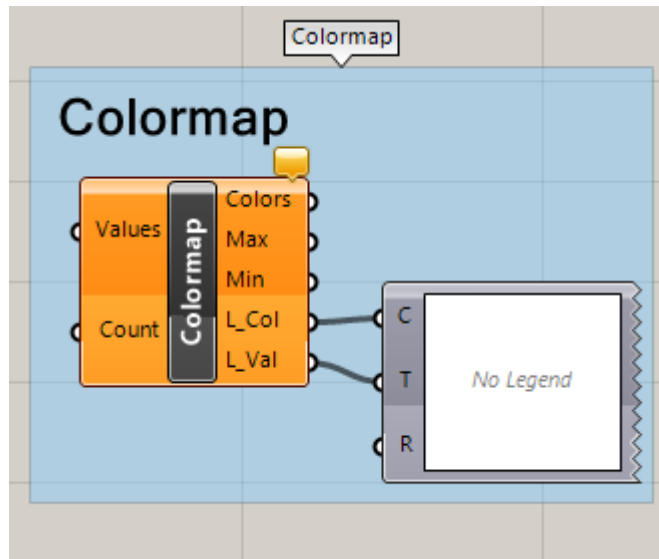


Figure 4.65: Screenshot of *Colormap* (Pamela)

4.8.17 Component: Mesh Values (Pamela)

4.8.17.1 Synopsis

Icon



Purpose

Display a mesh colored according to the list of input values per Mesh Face or Vertex (one or the other). In addition, colors and tags for a legend are provided.

Nickname Mesh Values

Type Name Pamela.GH.ResultViews.MeshValues_Component

Location Views

4.8.17.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Input mesh to display values.
Values	Number	Values to display.
Postfix	Text	Text (best only single character) to append to each value when displayed.
Count	Integer	Number of items to generate for the legend. If necessary, the number is adjusted to be \leq the number of data values provided.

4.8.17.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Max	Number	Maximum value of the input data.
Min	Number	Minimum value of the input data.
L_Col	Colour	Colors for the legend items.
L_Item	Text	Text (values) for the legend items.
L_Rect	Rectangle	Default rectangle to show legend directly beside mesh in the viewport. Delete the connection if not convenient.

4.8.17.4 *Menu items*

The following component menu items exist in addition to the standard drop down menu items:

- Draw mesh edges
-
- Blue-Green-Yellow-Magenta-Red
- Blue-Green-Yellow-White
- Blue-Magenta-Yellow
-
- Show texts instead of colored mesh surface
- Text Size
- Values as text
- Vertex ID's as text
- Face ID's as text

4.8.17.5 *Appearance*

Figure 4.66 shows the component *Mesh Values (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

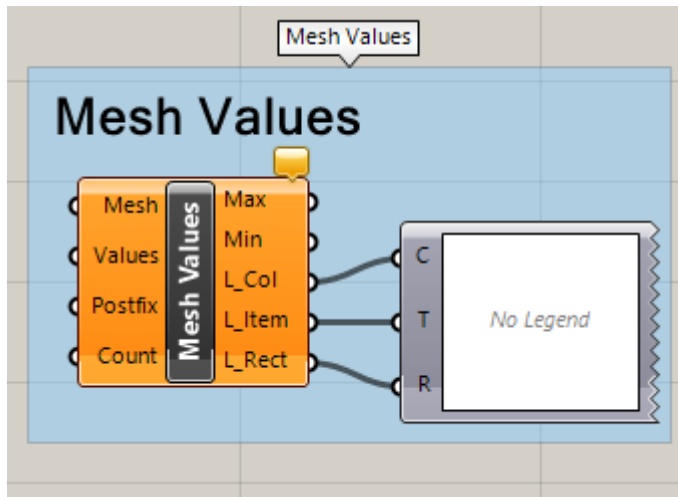


Figure 4.66: Screenshot of *Mesh Values* (Pamela)

4.9 PANEL: DETAILING

Figure 4.67 shows all icons of the components grouped under the panel *Detailing*

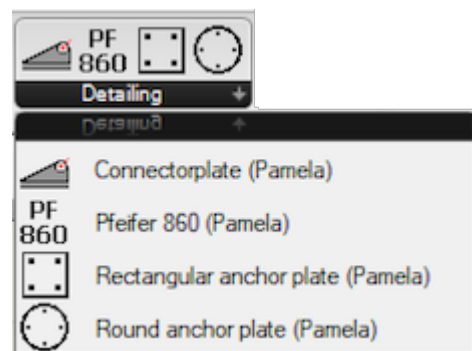


Figure 4.67: overview of panel *Detailing*

4.9.1 Component: Connectorplate (Pamela)

4.9.1.1 Synopsis

Icon	
Purpose	Creates a connectorplate on the reference base plane aligned with the force vector.
Nickname	Connectorplate
Type Name	Pamela.GH.Detailing.ConnectorplateComponent
Location	Detailing

4.9.1.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
baseP	Plane	Reference plane to connect the plate.
forceV	Vector	Force vector.
pinPt	Point	Pinpoint of force vector to connected structure. Required if no pivotpoint is defined.
pivPt	Point	Pivotpoint of force vector (centerpoint of hole). Required if no pinpoint is defined.
d	Number	Diameter of hole.
r	Number	Outer radius
dist	Number	Distance between hole centerpoint and connecting structure .
thick	Number	Thickness of the extrusion.
minLen	Number	Minimal length of resulting brep on connecting structure.
maxLen	Number	Maximal length of resulting brep on connecting structure.

4.9.1.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
connP	Plane	Reference plane rotated around its Z-Axis to align X-Axis with projection of force vector.
forceV	Vector	Force vector.
pinPt	Point	Pinpoint of force vector to connected structure.
pivPt	Point	Pivotpoint of force vector (centerpoint of hole).
centerLn	Line	Centerline of base surface on connecting structure.
brep	Brep	Resulting 3D brep (border representation).
proj	Brep	2D projection of the resulting outline.
drawP	Plane	Plane of 2D projection drawing. Can be used as reference when 2D projection shall be drawn in different location.
Info	Text	Processing and statistical information.

4.9.1.4 *Appearance*

Figure 4.68 shows the component *Connectorplate (Pamela)* after placement on the Grasshopper document.

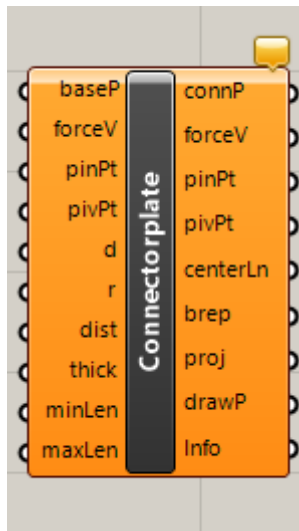


Figure 4.68: Screenshot of *Connectorplate (Pamela)*

4.9.2 Component: Pfeifer 860 (Pamela)

4.9.2.1 *Synopsis*

Icon	PF 860
Purpose	Parameterset for connectorplate Type 860 from Pfeifer.
Nickname	PF860
Type Name	Pamela.GH.Detailing.Pfeifer860ParameterComponent
Location	Detailing

4.9.2.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
d	Number	Diameter of hole in [m].

4.9.2.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
d	Number	Diameter of hole in. Corresponds to input value.
rOut	Number	Outer radius.
dist	Number	Distance from hole centerpoint to connecting structure .
thick	Number	Thickness of the extrusion.

4.9.2.4 Appearance


Figure 4.69 shows the component *Pfeifer 860 (Pamela)* after placement on the Grasshopper document.



Figure 4.69: Screenshot of *Pfeifer 860 (Pamela)*

4.9.3 Component: Rectangular anchor plate (Pamela)

4.9.3.1 Synopsis

Icon	
Purpose	Creates a rectangular or square anchor plate on the reference base plane aligned with the force vector.
Nickname	Rect Anchor Plate
Type Name	Pamela.GH.Detailing.RectAnchorPlateComponent
Location	Detailing

4.9.3.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
base plane	Plane	Reference plane to connect the plate.
center line	Line	Centerline of connected plate.
size x	Number	Size in x direction parallel.
size x	Number	Outer radius
thickness	Number	Thickness of plate.
holes x	Integer	Number of holes in X-direction per side.
holes y	Integer	Number of holes in Y-direction per side.
diameter	Number	Diameter of holes.
edge	Number	Distance of holes from edge of plate.
fillet	Number	Fillet radius for the 4 corners of the plate. Use 0.0 for none, otherwise hole radius is applied for default.

4.9.3.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
brep	Brep	Resulting 3D brep (border representation).
proj	Brep	2D projection of the resulting outline.
drawP	Plane	Plane of 2D projection drawing. Can be used as reference when 2D projection shall be drawn in different location.
centerPts	Point	Centerpoints of all holes, on top surface of the plate.
Info	Text	Processing and statistical information.

4.9.3.4 Appearance

Figure 4.70 shows the component *Rectangular anchor plate (Pamela)* after placement on the Grasshopper document.

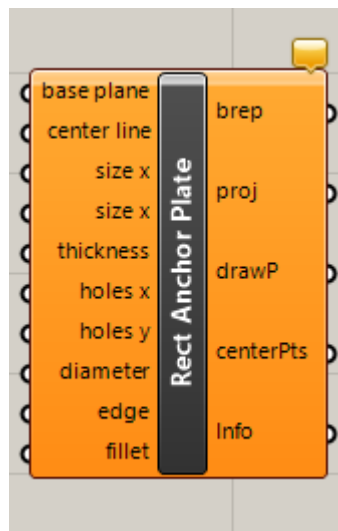



Figure 4.70: Screenshot of *Rectangular anchor plate (Pamela)*

4.9.4 Component: Round anchor plate (Pamela)

4.9.4.1 Synopsis

Icon	
Purpose	Creates a round anchor plate on the reference base plane aligned with the force vector.
Nickname	Round Anchor Plate
Type Name	Pamela.GH.Detailing.RoundAnchorPlateComponent
Location	Detailing

4.9.4.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
base plane	Plane	Reference plane to connect the plate.
reference line	Line	Reference centerline of connected plate.
plate radius	Number	Radius of plate.
count	Integer	Total number of holes.
size	Number	Size in angular degrees
offset	Number	Rotational offset in angular degrees.
centerRadius	Number	Radius of circular centerline
diameter	Number	Diameter / width of holes.
thickness	Number	Thickness of plate.

4.9.4.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
brep	Brep	Resulting 3D brep (border representation).
proj	Brep	2D projection of the resulting outline.
drawP	Plane	Plane of 2D projection drawing. Can be used as reference when 2D projection shall be drawn in different location.
centerPts	Point	Centerpoints of all holes, on top surface of the plate.
Info	Text	Processing and statistical information.

4.9.4.4 *Appearance*

Figure 4.71 shows the component *Round anchor plate (Pamela)* after placement on the Grasshopper document.

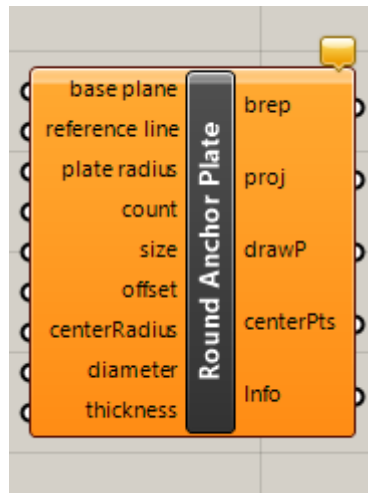


Figure 4.71: Screenshot of *Round anchor plate (Pamela)*

4.10 PANEL: MEMBRANE


Figure 4.72 shows all icons of the components grouped under the panel *Membrane*



Figure 4.72: overview of panel *Membrane*

4.10.1 Component: Membrane Pimper (Pamela)

4.10.1.1 Synopsis

Icon	
Purpose	Utility component to add corner plates to mesh.
Nickname	Membrane Pimper
Type Name	Pamela.GH.Membrane.MembranePimperComponent
Location	Membrane

4.10.1.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	The input mesh to analyze.
corner	Point	Corner point of the membrane.
left	Point	left cable connection point of the membrane.
right	Point	right cable connection point of the membrane.

4.10.1.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	Resulting Mesh with new Geometry.
cornerInd	Integer	Indices of cable endpoints.
info	Text	Statistical information gathered during input processing.

4.10.1.4 *Appearance*

Figure 4.73 shows the component *Membrane Pimper (Pamela)* after placement on the Grasshopper document.

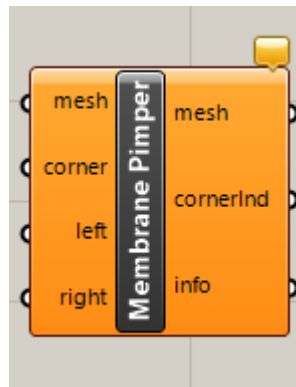


Figure 4.73: Screenshot of *Membrane Pimper (Pamela)*

4.11 PANEL: PATTERNING

Figure 4.74 shows all icons of the components grouped under the panel *Patterning*

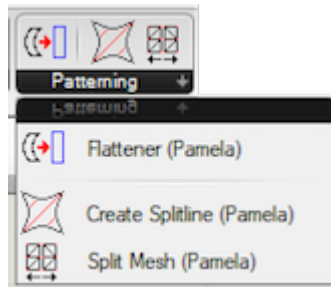



Figure 4.74: overview of panel *Patterning*

4.1.1.1 Component: Flattener (Pamela)

4.1.1.1.1 Synopsis

Icon	
Purpose	Flattens a non-developable (curved in two directions) 3-D mesh into a flat 2-D pattern.
Nickname	Flattener
Type Name	Pamela.GH.Patterning.FlattenerComponent
Location	Patterning

4.1.1.1.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	The input mesh to flatten.
refPts	Point	Optional list of reference points.
iter	Integer	Number of iterations to execute. Maximum is limited to 100.

4.1.1.1.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Resulting Mesh with new Geometry.
refPts	Point	Final positions of reference points on flattened output mesh.


4.1.1.1.4 Appearance

Figure 4.75 shows the component *Flattener (Pamela)* after placement on the Grasshopper document.

Figure 4.75: Screenshot of *Flattener (Pamela)*

4.11.2 Component: Create Splitline (Pamela)

4.11.2.1 Synopsis

Icon	
Purpose	Prepare (refine) mesh to be split along polylines (Geodesic lines are a good choice) on the mesh.
Nickname	Splitline
Type Name	Pamela.GH.Patterning.SplitLineComponent
Location	Patterning

4.11.2.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	The input mesh to modify.
Paths	Curve	The polylines (intersection points with mesh) to refine the mesh along.
pull	Number	Maximum distance an intersection point is moved (pulled) towards an existing vertex.

4.11.2.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Refined mesh with new geometry.
Paths	Curve	The refined mesh paths (polylines) which can be used to split the mesh along.

4.11.2.4 Appearance

Figure 4.76 shows the component *Create Splitline (Pamela)* after placement on the Grasshopper document.



Figure 4.76: Screenshot of *Create Splitline (Pamela)*

4.11.3 Component: Split Mesh (Pamela)

4.11.3.1 Synopsis

Icon	
Purpose	Split a Mesh along polyline(s). Best results when using polylines representing mesh paths (e.g. output of geodesic line combined with splitline component).
Nickname	Split
Type Name	Pamela.GH.Patterning.SplitMeshComponent
Location	Patterning

4.11.3.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	The input mesh to split.
Paths	Curve	The mesh paths (polylines) to split the mesh along.

4.11.3.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Mesh	Mesh	Resulting Meshes with new geometry.

4.11.3.4 Appearance

Figure 4.77 shows the component *Split Mesh (Pamela)* after placement on the Grasshopper document.



Figure 4.77: Screenshot of *Split Mesh (Pamela)*

4.12 PANEL: UTILITY

Figure 4.78 shows all icons of the components grouped under the panel *Utility*

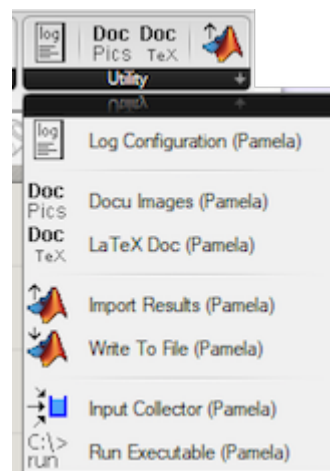



Figure 4.78: overview of panel *Utility*

4.12.1 Component: Log Configuration (Pamela)

4.12.1.1 Synopsis

Icon	
Purpose	Initializes the logging framework.
Nickname	Logger
Type Name	Pamela.GH.Utility.Logger_Component
Location	Utility

4.12.1.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Level	Integer	Level: 0=DEBUG, 1=INFO, 2=WARNING, 3=ERROR
Log File	Text	Path (including filename) to store logfile.

4.12.1.3 Appearance

Figure 4.79 shows the component *Log Configuration (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

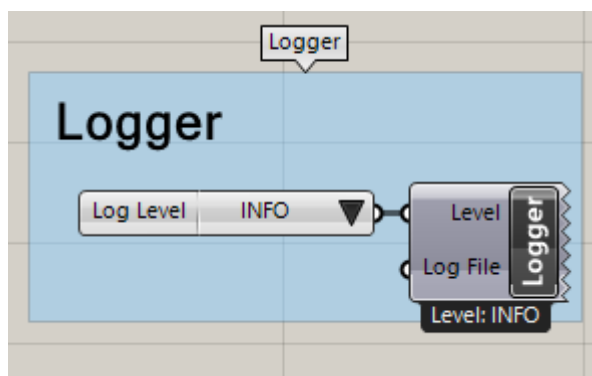


Figure 4.79: Screenshot of *Log Configuration (Pamela)*

4.12.2 Component: LaTeX Doc (Pamela)

4.12.2.1 Synopsis

Icon	Doc TeX
Purpose	Create LaTeX documentation for all Grasshopper components of this assembly. The following entities are created: 1.) a document with all type descriptions 2.) a list of all subcategories 3.) a .png file for each icon. REMARK: make sure you use the DocImages component first to generate the screenshots of the components placed on the canvas for them to be included and scaled according to text width.
Nickname	Docu Text
Type Name	Pamela.GH.Utility.Documentation_Component
Location	Utility

4.12.2.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Path	Text	Path (including dummy filename) to store generated files. Icons will be placed in subfolder.
now!	Boolean	Set to true to create documentation and write data to the filesystem.

4.12.2.3 Appearance

Figure 4.80 shows the component *LaTeX Doc (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

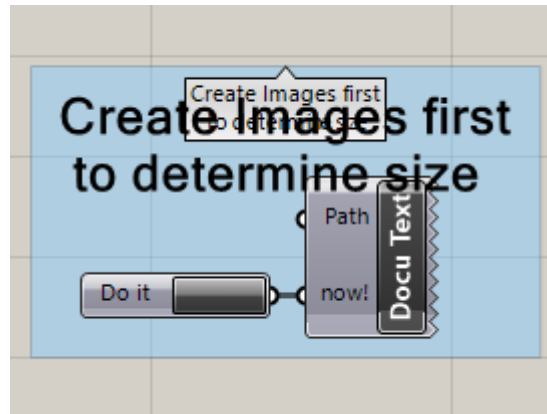


Figure 4.80: Screenshot of *LaTeX Doc (Pamela)*

4.12.3 Component: Import Results (Pamela)

4.12.3.1 Synopsis

Icon	
Purpose	Import final X, Y, Z coordinates from file generated by Matlab scripts and assign them to a model as final / displaced positions. This component is mainly used by the Execute Matlab Solver Component.
Nickname	Import
Type Name	Pamela.GH.Utility.ImportResults_Component
Location	Utility

4.12.3.2 *Input Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Model representing initial state.
Workdir	Text	Working directory of Matlab. Results file must be located in a subdirectory called Export.
File	Text	Name of the file containing the results data
now!	Boolean	Set to true to trigger read data from file.

4.12.3.3 *Output Parameters*

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	Model with results.

4.12.3.4 *Appearance*

Figure 4.81 shows the component *Import Results (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

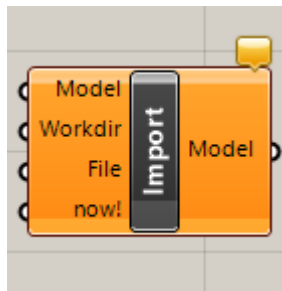



Figure 4.81: Screenshot of *Import Results (Pamela)*

4.12.4 Component: Write To File (Pamela)

4.12.4.1 Synopsis

Icon	
Purpose	Write the Pamela Datamodel to a file. This file can be used in Matlab scripts for further processing or results verification. This component is mainly used by the Execute Matlab Solver Component.
Nickname	Filewriter
Type Name	Pamela.GH.Utility.WriteModelToFile_Component
Location	Utility

4.12.4.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Model	Model	The assembled Pamela Model to write to file.
Filepath	Text	Full path including filename for output file.
now!	Boolean	Set to true to trigger write data to the file.

4.12.4.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
Workdir	Text	Path to working directory for Matlab and script file.
Trigger	Boolean	Is set to true once the file is written. False when input trigger is false.

4.12.4.4 Appearance

Figure 4.82 shows the component *Write To File (Pamela)* and all embedded components and parameters after placement on the Grasshopper document. The components are automatically grouped and the group is labelled accordingly.

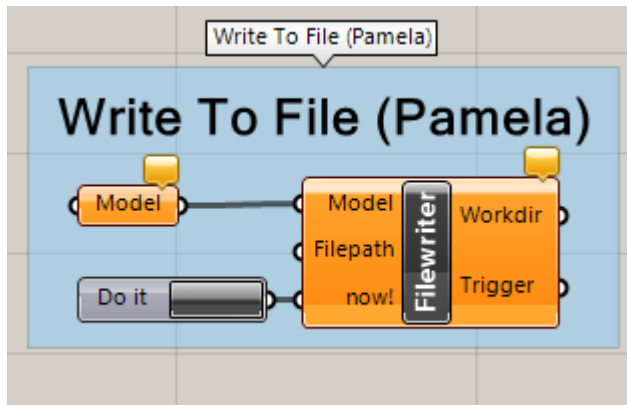


Figure 4.82: Screenshot of *Write To File (Pamela)*

4.1.2.5 Component: Input Collector (Pamela)

4.1.2.5.1 Synopsis

Icon	
Purpose	Collects input definitions from active Rhino document.
Nickname	IC
Type Name	Pamela.GH.Utility.InputCollectorComponent
Location	Utility

4.1.2.5.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
mesh	Mesh	Meshes to create 2D elements.
crv	Curve	Curves to create 1D elements.
pt	Point	Point objects for supports.

4.1.2.5.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
info	Text	Key / value pairs of the user texts assigned to the object.

4.1.2.5.4 Appearance

Figure 4.83 shows the component *Input Collector (Pamela)* after placement on the Grasshopper document.

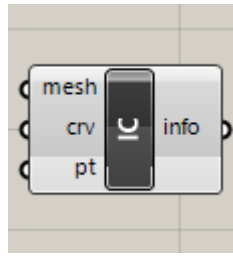


Figure 4.83: Screenshot of *Input Collector (Pamela)*

4.12.6 Component: Run Executable (Pamela)

4.12.6.1 Synopsis

Icon	C:\> run
Purpose	Run an executable with attributes
Nickname	RunExec
Type Name	Pamela.GH.Utility.RunExecComponent
Location	Utility

4.12.6.2 Input Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
T	Text	Full path to the executable
T	Text	Arguments to pass to the executable
B	Boolean	Boolean parameter
B	Boolean	Wait for exit code

4.12.6.3 Output Parameters

<i>ID</i>	<i>Type</i>	<i>Description</i>
S	Text	Exit code from the program

4.12.6.4 Appearance

Figure 4.84 shows the component *Run Executable (Pamela)* after placement on the Grasshopper document.

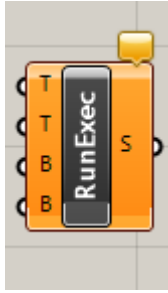


Figure 4.84: Screenshot of *Run Executable (Pamela)*

5

TESTING AND PROOF OF CONCEPT

If you find that you're spending almost all your time on theory, start turning some attention to practical things; it will improve your theories. If you find that you're spending almost all your time on practice, start turning some attention to theoretical things; it will improve your practice.

— Donald E. Knuth

5.1 INTRODUCTION

This chapter shows how the accuracy of the implementation was verified and proofs that the system is working and can be used to design and analyze membrane structures.

Software testing is a field of expertise by itself. Different complementary approaches exist and the effort to test a software can often be compared to the effort needed to write the program. A basic distinction of the test methodology is based on how the system is viewed:

- In *black box testing*, the artifact to test is treated as a black box, the functionality is tested based on an outside view. *Does the result correspond with the expected result for the given input?* This methodology is mainly used to validate the final results of a workflow.
- In contrast, *white box testing* is focusing on the internals of the implementation. These can be things like *is all memory released when no longer used?* etc. This methodology is mainly used when the program is executed in debugging mode and the source code can be executed step by step.

5.1.1 Unit Testing

To test functionality of a specific component or algorithm individually, I mainly followed the concept of unit testing. Therefore, testing is split up in several modules, each of them testing a specific algorithm or logic. I created individual Grasshopper definitions for each functionality and algorithm to be covered by unit tests. An important rule of unit testing is to always execute all unit tests. This allows to discover unexpected side effects (changes for module A effecting module B due to changes made in shared code C).

5.1.2 System Testing

Testing of workflows and overall results has been done in several steps:

- Verification of the algorithms against the Matlab reference implementation
- Over the past 2 years I used PAMELA to complete all relevant assignments from the study program at IMS
- Real live testing using PAMELA for customer projects

Numerical testing against 3rd party membrane software was done only very limited since I currently don't have access to any such software running on my computer. But since the Matlab implementation was previously tested by Peter Novýsedlák, it is assumed that the results of this reference implementation are correct and can be used to substitute testing against another membrane software.

The steps listed above are discussed in more detail in this chapter.

5.2 TESTING AGAINST REFERENCE IMPLEMENTATION

For results verification against Matlab implementation it was important to use exactly the same values and properties in both systems. Same values means for example exactly the same point coordinates and indexing for each vertice of the mesh etc. To ease and automate this process, I have implemented dedicated components which are described in detail in section 4.7.3.

Figure 5.1 shows the component *Execute Matlab (Pamela)* and all embedded components and parameters. It's actually a group of three components, each of them dedicated to a specific task:

1. The first one reads the data model, converts the data structure and writes the Matlab specific file to disk. once the file is written, the second component is triggered.
2. The second one calls Matlab as a native process on the operating system and waits until Matlab returns. Soon as this signal is received, the third component is triggered.
3. The last component called Importer reads the result data created by Matlab from disk, converts the data and appends the data to the original data model.

Having the functionality split into three components further allows to use them individually as well. A common use case was to disable the middle component and start Matlab in debug mode individually.

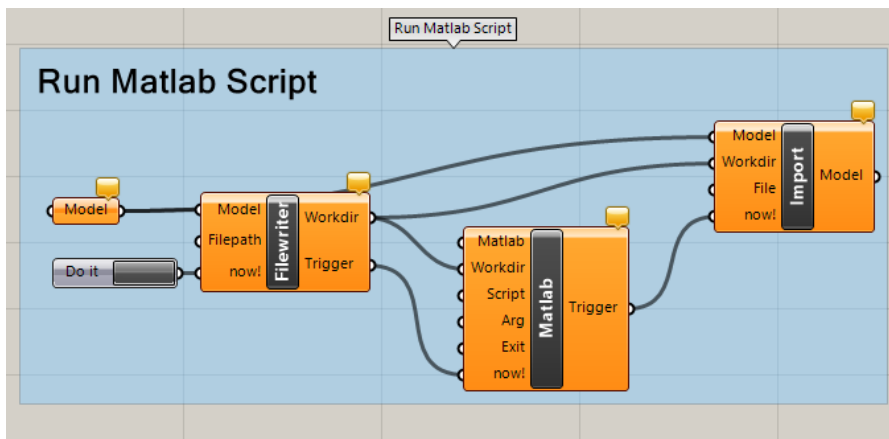


Figure 5.1: Screenshot of *Execute Matlab (Pamela)*

5.2.1 Test Scenarios

Results comparison against Matlab implementation was done on a very detailed level:

1. A cable net only, therefore with a model containing only 1D elements and supports but no membrane / 2D elements
2. A model just based on a membrane and supports, no edge cables or other 1D elements
3. several models combining 1D and 2D elements and supports

5.2.2 Test Cases

Based on the models mentioned above, several test cases with different prestress ratios and warp directions were executed for the different analysis modes:

- Form finding with modeFactor property set to 0.0, therefore ignoring material properties
- Damped form finding with modeFactor set to 0.001. This configuration is mainly used to form find cable nets.
- Analysis (modeFactor = 1.0) without any loads
- Analysis (modeFactor = 1.0) with prestress and self-weight only
- Analysis (modeFactor = 1.0) with prestress, self-weight and point loads.

5.2.3 Results Verification

With the detailed testing several errors in my implementation were detected. Some of them were introduced by myself, others originated

in 3rd party libraries. The most unexpected misbehavior was the fact that the implementation of the square root function provided by Microsoft for the data type double delivered different results than the one incorporated in Matlab. The differences were very small, usually in the of range 0.00000001. But the difference resulted in wrong values in the element stiffness matrix which summed up with the iterations and led to a different convergence behavior or even worse, prevented convergence at all. Double checking the results with Excel and also a pocket calculator proofed that Microsoft implementation was wrong. Things like this you usually don't expect and they can lead to very time-consuming testing and debugging. The solution to fix this error was to implement my own square root function.

Verification of the results was done in two steps:

1. Verification of convergence behavior (see 5.2.4)
2. Verification of numerical values (see 5.2.5)

5.2.4 Verification of Convergence Behavior

Verification of convergence behavior is done based on logfile output. PAMELA has a logging functionality integrated which can be activated using the Log Configuration component (see 4.12.1). If the level is set to *Info*, tolerance and displacement increment is output at the end of each iteration.

Matlab scripts are configured to output the same information to the console which allows to easily compare the behavior of both implementations. For illustration, one set of logfile data is included.

Listing 5.1: Convergence of PAMELA Implementation

```
Membrane with edge cables
Supports on all membrane corners
FEA Form finding damped (0.001)
Model: Frey

Iter: 1/51 Tol: 0.0001 Max. diff: 1 Displ incr: 2.59119003100387
Iter: 2/51 Tol: 0.0001 Max. diff: 0.231166154872148 Displ incr:
0.545067279532629
Iter: 3/51 Tol: 0.0001 Max. diff: 0.0766645231833856 Displ incr:
0.183244426748704
Iter: 4/51 Tol: 0.0001 Max. diff: 0.0340303698988302 Displ incr:
0.0808544021734926
Iter: 5/51 Tol: 0.0001 Max. diff: 0.0180408974908638 Displ incr:
0.042933669172953
Iter: 6/51 Tol: 0.0001 Max. diff: 0.0104649959645834 Displ incr:
0.0248891116506604
Iter: 7/51 Tol: 0.0001 Max. diff: 0.00639799030766625 Displ incr
: 0.0152188511910136
```

```

Iter: 8/51 Tol: 0.0001 Max. diff: 0.00403631101140875 Displ incr
      : 0.00960072451757556
Iter: 9/51 Tol: 0.0001 Max. diff: 0.00260659091574162 Displ incr
      : 0.00619996334631589
Iter: 10/51 Tol: 0.0001 Max. diff: 0.00171644022098323 Displ
      incr: 0.00408273606185146
Iter: 11/51 Tol: 0.0001 Max. diff: 0.00114903733008164 Displ
      incr: 0.00273306309433529
Iter: 12/51 Tol: 0.0001 Max. diff: 0.000780763138978998 Displ
      incr: 0.00185712207159313
Iter: 13/51 Tol: 0.0001 Max. diff: 0.000537291358615094 Displ
      incr: 0.00127798743883579
Iter: 14/51 Tol: 0.0001 Max. diff: 0.000374038524018902 Displ
      incr: 0.00088968500247322
Iter: 15/51 Tol: 0.0001 Max. diff: 0.000262958159833781 Displ
      incr: 0.000625466786210727
Iter: 16/51 Tol: 0.0001 Max. diff: 0.00018643571030735 Displ
      incr: 0.000443453756811303
Iter: 17/51 Tol: 0.0001 Max. diff: 0.00013319964476075 Displ
      incr: 0.000316826259364233
Iter: 18/51 Tol: 0.0001 Max. diff: 9.57938115659238E-05 Displ
      incr: 0.000227853838167884
Nonlinear FEA was used for Form Finding !
FEM finished: Iteration: 18/51 Tolerance: 0.0001 Max. diff:
9.57938115659238E-05

```

Listing 5.2: Convergence of Matlab Implementation

```

Iteration: 1 crit: 1 Displacement increment: 2.5912
Iteration: 2 crit: 0.23117 Displacement increment: 0.54507
Iteration: 3 crit: 0.076665 Displacement increment: 0.18324
Iteration: 4 crit: 0.03403 Displacement increment: 0.080854
Iteration: 5 crit: 0.018041 Displacement increment: 0.042934
Iteration: 6 crit: 0.010465 Displacement increment: 0.024889
Iteration: 7 crit: 0.0063981 Displacement increment: 0.015219
Iteration: 8 crit: 0.0040364 Displacement increment: 0.0096009
Iteration: 9 crit: 0.0026067 Displacement increment: 0.0062001
Iteration: 10 crit: 0.0017165 Displacement increment: 0.004083
Iteration: 11 crit: 0.0011492 Displacement increment: 0.0027334
Iteration: 12 crit: 0.00078083 Displacement increment: 0.0018573
Iteration: 13 crit: 0.00053734 Displacement increment: 0.0012781
Iteration: 14 crit: 0.00037403 Displacement increment: 0.00088966
Iteration: 15 crit: 0.00026291 Displacement increment: 0.00062536
Iteration: 16 crit: 0.00018643 Displacement increment: 0.00044344
Iteration: 17 crit: 0.0001332 Displacement increment: 0.00031683
Iteration: 18 crit: 9.5819e-05 Displacement increment: 0.00022791

Nonlinear FEA was used for Form Finding !
Converged successfully at 18 iterations
Convergence criteria 9.5819e-05 is smaller than prescribed
tolerance 0.0001
Final norm of residual forces 0.0026465
Last displacement increment 0.00022791

```

5.2.5 Verification of Numerical Values

Once the same convergence behavior was achieved all numerical values were compared in detail. Numerical values are:

- displacements / point coordinates
- stresses / forces

To verify the accuracy of the displacements / point coordinates a model was created where the two solvers run in parallel. Since the same point order was used, it was possible to compare the coordinates of the two result sets easily. Figure 5.2 shows the relevant part of the model used for numerical comparison. Vectors are computed based on the points with the same index in both result sets. The length of the resulting vectors is used to color the mesh for visualization.

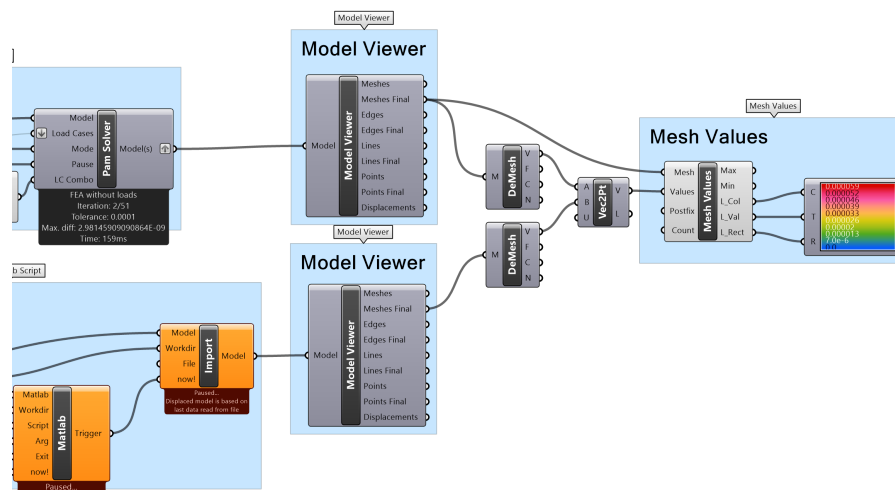


Figure 5.2: parallel solver setup

Figure 5.3 shows the visualization of the results, the absolute distance of the point coordinates for identical point indices. The largest difference is 0.000735 which is well acceptable.

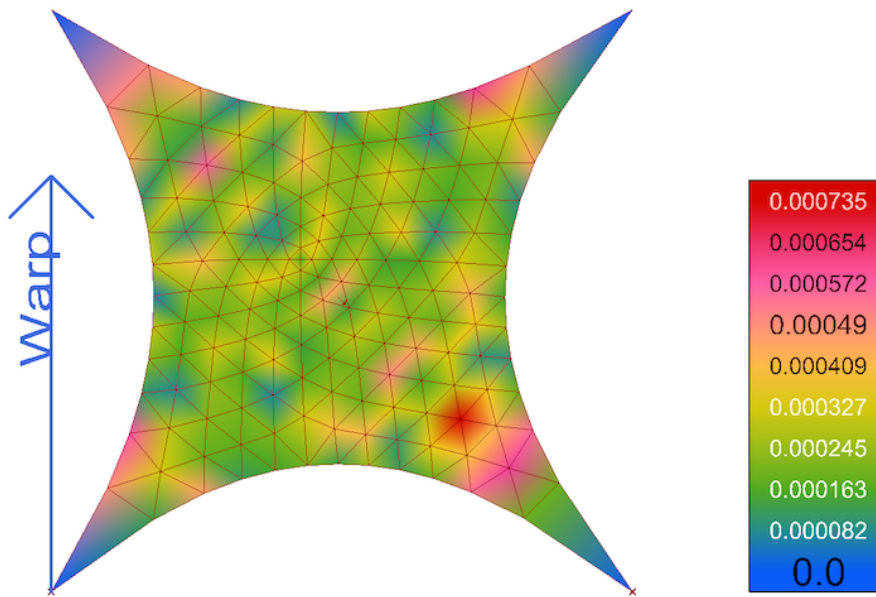


Figure 5.3: distance of nodal coordinates

5.3 EXAMPLE HYPAR WITH RIDGE CABLE

In this section the results for one of the tasks of the assignment in *CM6 Studio Detailing* are reproduced to demonstrate how the process of form finding and structural analysis can be implemented for a simple *Hypar* with ridge cable. Patterning is not described at this point because this example is already used for the explanations in section 3.14.

5.3.1 Task

Figure 5.4 shows the geometry used for this task. It must be noted that a size of 20m x 20m seems rather big for a traditional *Hypar* structure. As a consequence, the required prestress in the membrane and especially the resulting axial forces in the cables are remarkable high.

As prestress ratio for warp/weft/cables 1:1:10 is specified. The given value for snow load is 0.65kN/m² and for wind load 1kN/m².

5.3.2 Preliminary Remarks

For structural analysis of a membrane, usually different load cases are defined and combinations of these individual load cases with specific load factors are considered to evaluate maximum and minimum values of stresses and forces, also known as enveloped results. The task only defines individual loads like snow, wind 45 or wind -45 degrees, no

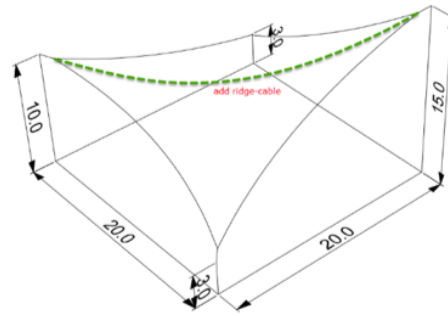


Figure 5.4: Dimensions for HyPar with ridge cable

combinations and respective factors are given. Therefore the results are processed accordingly:

- Besides the external load specified, each load case considers also prestress and self-weight.
- For each load, a separate load case is defined and evaluated individually, no load combinations like snow and wind are evaluated.
- For dimensioning of the membrane and cables, envelope results of all load cases are considered.

5.3.3 Design

Since warp direction is not specified by the task, I chose warp to run in the direction from highpoint to highpoint. The main reason for this is to align it with the direction of the ridge cable and its pocket.

The Dimensions of 20m x 20m seem to be fairly big for a HyPar shaped membrane and high forces in the cables and structure are expected.

5.3.4 Model Definition

Figure 5.5 shows the complete model used for form finding and analysis including meshing, material specifications, load definitions and results visualization. It consists of 20 main component groups which are later described in more detail.

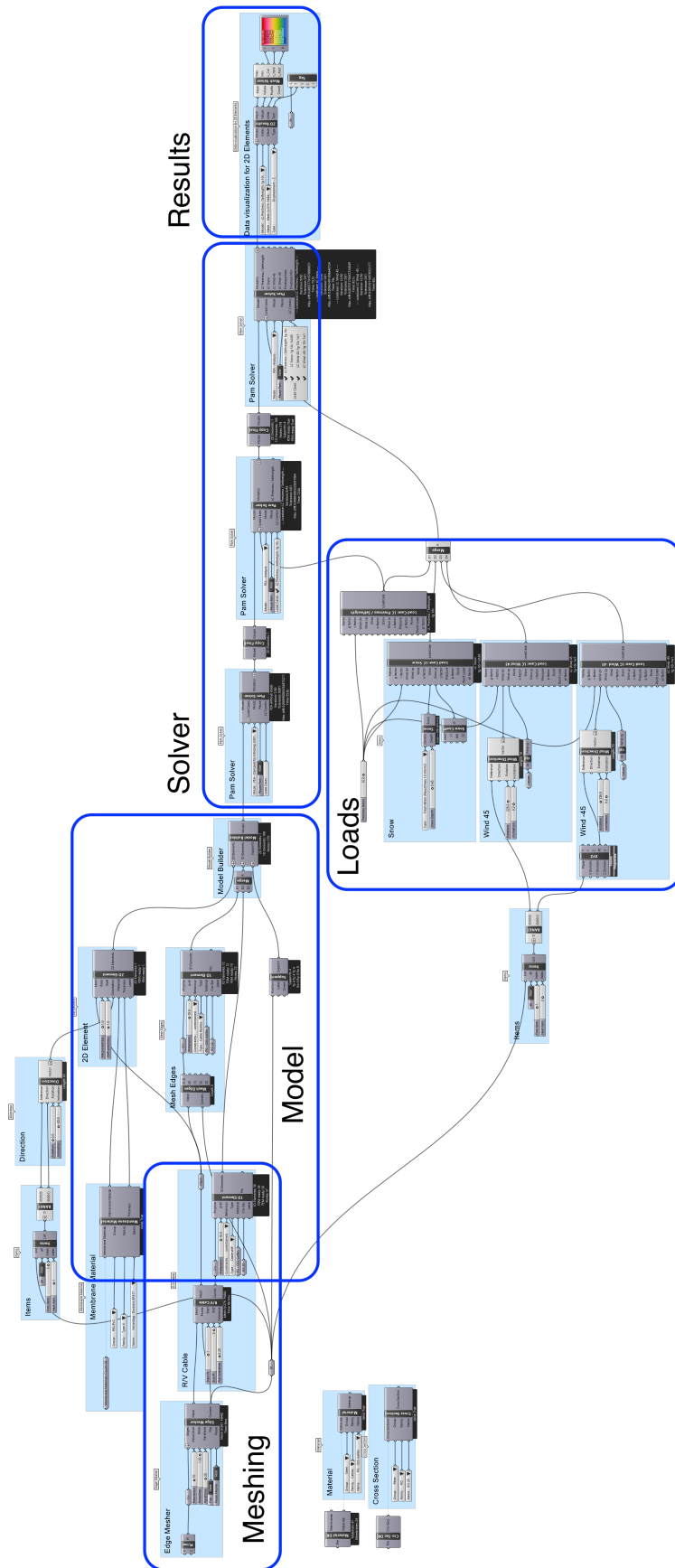


Figure 5.5: Complete model definition for task 1

The meshing is split in 2 parts, shown by figure 5.6:

- The preliminary mesh created based on the 3-dimensional outline of the membrane. 10 was chosen for the resolution parameter which lead to 19 nodes along each edge.
- Local mesh refinements along the path of the ridge cable which runs from corner 1 to corner 3. This group also contains the specification for the cable like pretension, behavior, material and cross-section. With this refinement along the diagonal it will later when the cutting patterns are created be possible to place a seam and cut the mesh exactly along the ridge cable.

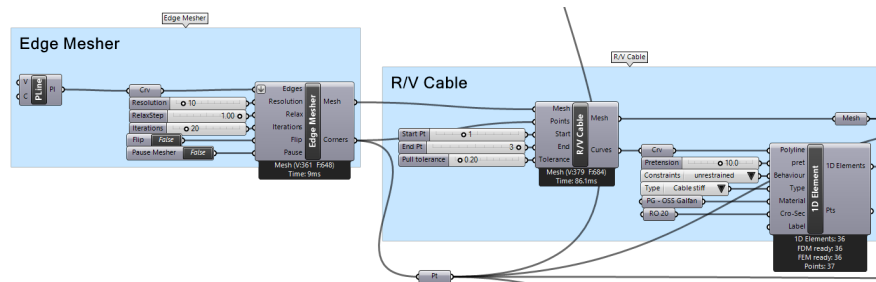


Figure 5.6: Mesh creation and refinement

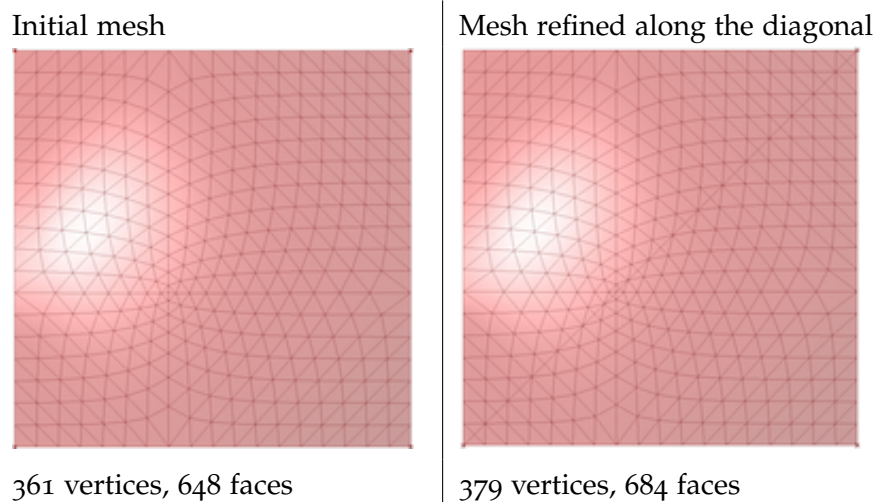


Table 5.1: Inital and refined mesh

It is important to understand that all subsequent definitions in the model must be based on the refined mesh and not the initial mesh.

The core part of the model definition is rather simple.

MEMBRANE ELEMENT

- The single mesh is used as input for a so called 2D Element (or Membrane Element)

- Prestress for warp and weft is set according to the tasks definition to 1 for warp and weft identical.
- Material and corresponding thickness is taken from the material database. As a starting value a PES/PVC fabric of Type II was assigned.

EDGE CABLES

- The Mesh Edges component is used to automatically identify the 4 cables running along the edges of the membrane. The corner points are used as differentiating factor.
- Since all cables have the same prestress a single 1D Element (or Line Element) specification is used and shared among the 4 cables
- Pretension value is set to 10
- Open spiral strand cables with 20mm in diameter are selected for material and cross section

RIDGE CABLE

- The definition of the ridge cable was included in the mesh refinement process shown by figure 5.6
- The same material specifications as for the edge cables are used

SUPPORTS

- Since there are no additional structural elements involved in the analysis, the 4 corners of the membrane are used as location input for the supports.
- The supports are fixed against movement in x, y and z direction, but they are free to rotate.

All model elements mentioned above are used as input for the model builder which assembles a consistent and validated data model based on these inputs. Since material specifications have been provided for all elements, the model is ready to be used by a Finite Element Method solver. Without the material specification, only the Force Density Solver could be used.

Figure 5.7 shows the components involved to define and assemble the data model.

The 4 load cases are defined individually according to the definitions in the task:

- 0.65 kN/m² for snow load
- 1.0 kN/m² for wind

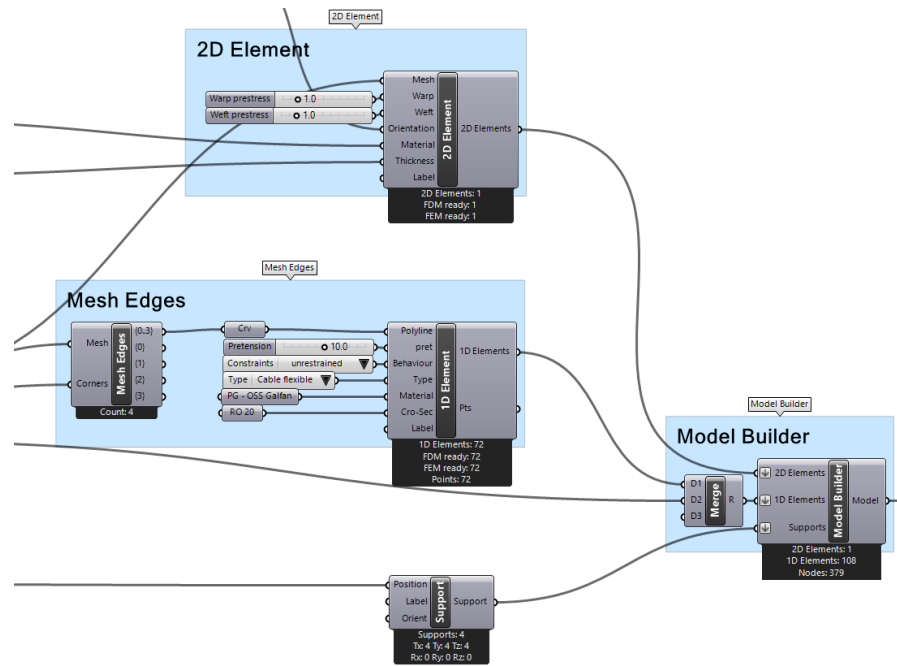


Figure 5.7: Assembling the data model

- Wind direction is set to 45 degrees from high point to high point and - 45 degrees from low point to low point respectively
- For the snow load roof shape coefficients are assigned to each triangle of the mesh automatically. As we will see later, only the area of the highest corner of the membrane will be affected by some snow drifting. The remaining part of the membrane is rather flat and therefore no snow drift is taken into account.
- For the wind loads pressure coefficients (also called cp values) are automatically assigned to each triangle of the mesh based on it's orientation compared to the wind direction.
- The prestress factor was elaborated iteratively. The load case wind from direction – 45 degrees was the one requiring the highest prestress to avoid wrinkling in all regions of the membrane.

Figure 5.8 shows the load case definitions used.

For form finding and analysis a chain of 3 solvers is being used.

- The first solver is just used for form finding. Loads are not taken into account and material properties are multiplied with 0.001.
- The second solver uses the geometry output of the first solver, prestress and self-weight are applied and material properties considered full. This intermediate step could be omitted, but the output geometry is a perfect basis to analyze the resulting

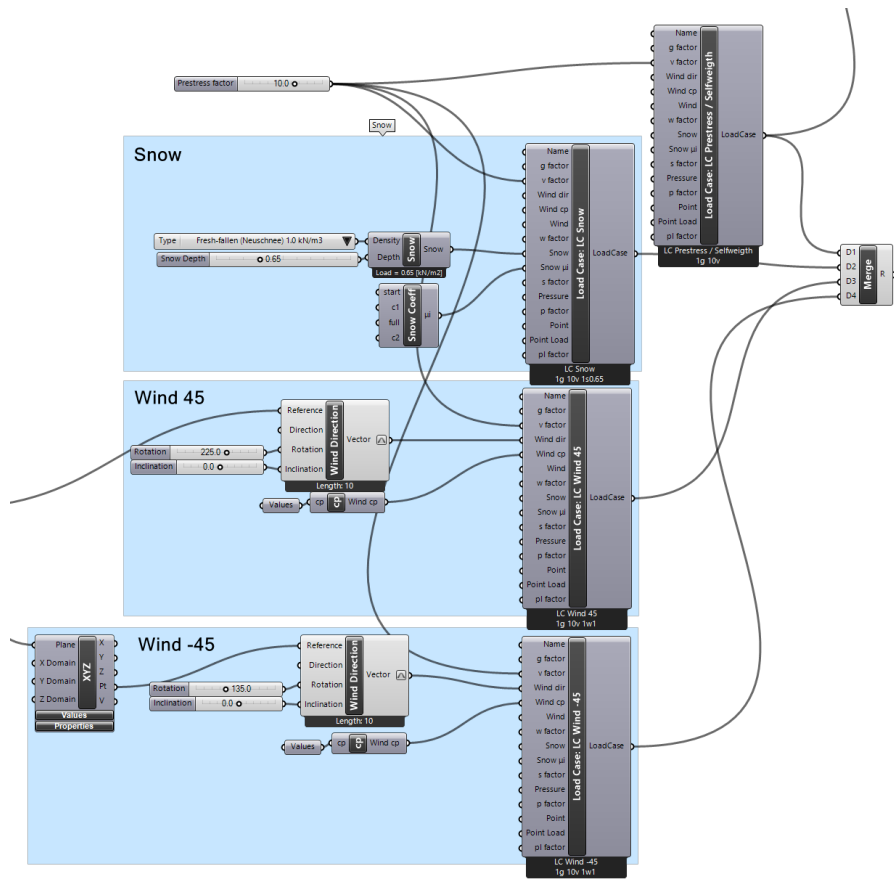


Figure 5.8: Load case definitions

displacements of the subsequent steps where load cases are applied.

- The third solver process applies all selected load cases one after the other and stores the resulting stresses and displacements individually. If multiple load cases are applied, the envelope minimal and envelop maximal values are calculated in addition and results stored individually as well. Applying 4 load cases (or load scenarios) therefore results in 6 output models.

Figure 5.9 shows the daisy chain of multiple solvers.

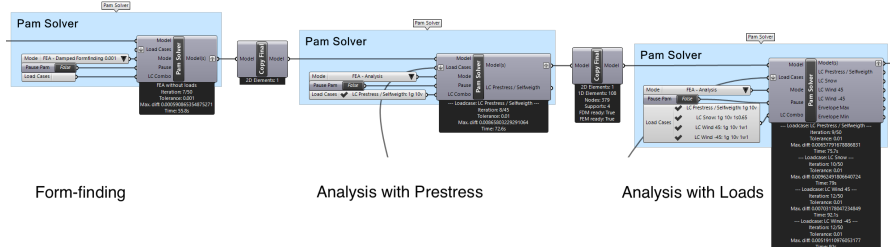
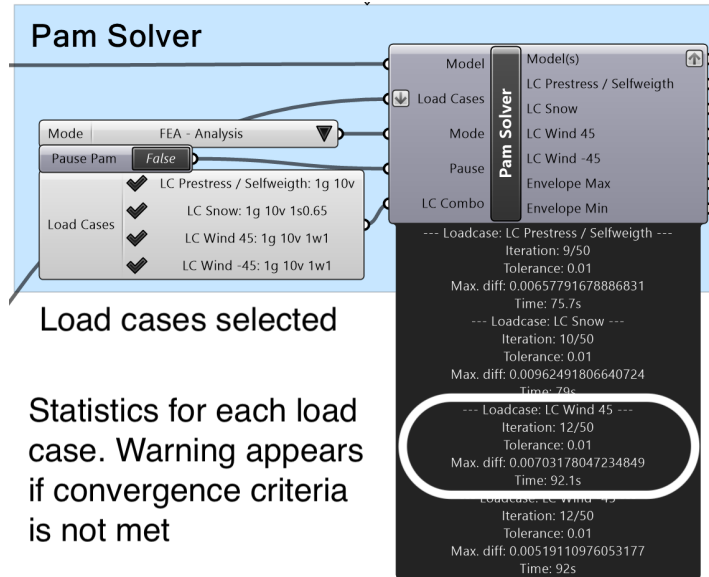


Figure 5.9: Daisy-chain of solvers

Figure 5.10 shows how each solver documents the progress and results summary of the execution in the black message bubble. If the convergence criteria for at least one load case is not met, the component turns orange and a warning message is displayed.



Load cases selected
 Statistics for each load case. Warning appears if convergence criteria is not met

Figure 5.10: Solver results summary

5.3.5 Form Finding

Table 5.2 shows the membrane after initial form finding and after formfinding with prestress and selfweight applied. The mesh on the right represents the final geometry.

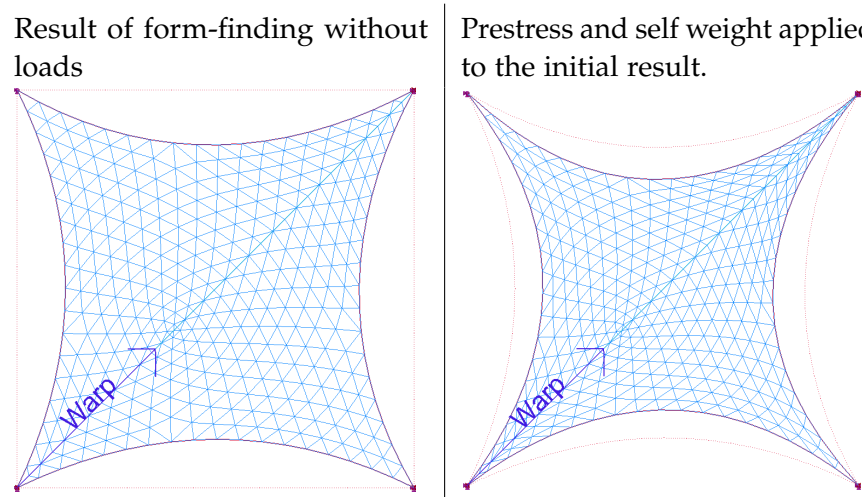


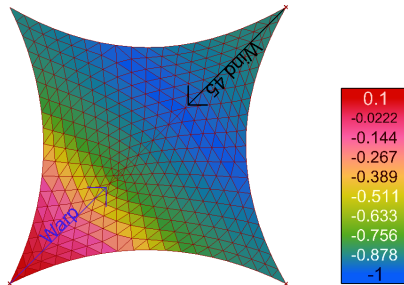
Table 5.2: Mesh after form finding

5.3.6 Loads

As mentioned before, specific algorithms are applied to automatically calculate adequate values for wind pressure coefficients and roof shape coefficients for snow loads. Therefore the values applied differ from the definitions in the assignment but I regard these values to be more accurate mainly because a wider range of values are applied and the steps between adjacent mesh triangles are smoother.

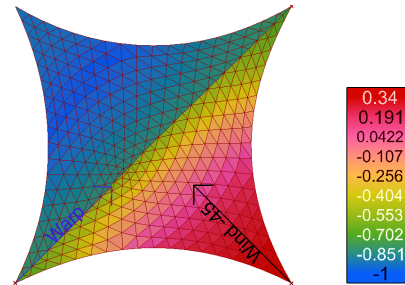
Table 5.3 shows the impact of the wind direction when wind load is applied.

Wind from direction of top right corner which is the highest point of the geometry results in uplift in this area. Some downward pressure occurs at the high point in the bottom left corner.



Wind - relative pressure [kN/m²]
LC Wind 45: 1g 10v 1w1

Wind in direction from low point to low point results in some downward pressure on the windward side of the membrane. Wind suction (uplift) is experienced on the leeward side.

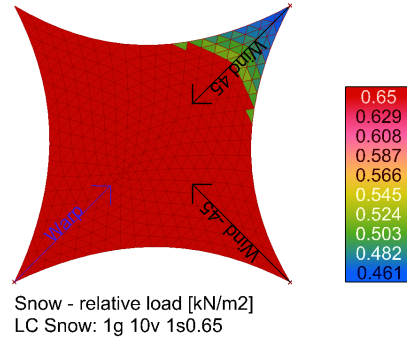


Wind - relative pressure [kN/m²]
LC Wind -45: 1g 10v 1w1

Table 5.3: Wind load on Hypar

Table 5.4 shows the impact of snow load and the enveloped maximum for external forces.

Snow load is acting uniform for the largest part of the membrane. Only in the area of the highest corner some snow drift is expected.



Snow and wind loads are acting in different directions. The envelope maximum of all external forces in any direction is given by the length of the corresponding resulting forces vector.

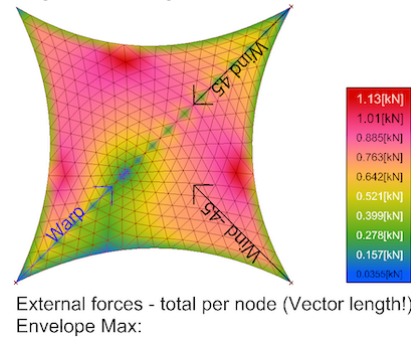


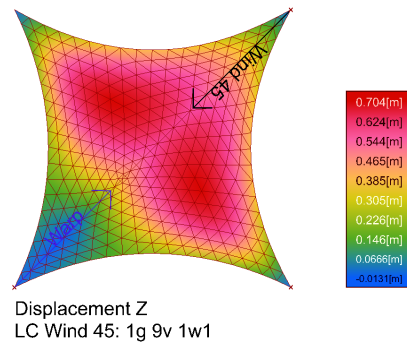
Table 5.4: Snow load and enveloped results

5.3.7 Serviceability / Deflection

The deflection of a membrane under loads is a common criteria to assess the serviceability. Although displacements can occur in any direction, for this structure the focus is on displacements up- or downwards and therefore in Z-direction.

Table 5.5 shows the resulting displacements depending on the direction of the applied wind loads.

Displacement in Z-direction due to wind 45 degrees is mainly upwards due to wind suction. The cable is reducing the movement only minimally along the ridge since this movement does not apply additional tension on the cable.



The resulting displacements in Z direction due to wind from -45 degrees correspond well with the findings from analyzing the reacting forces based on the cp values.

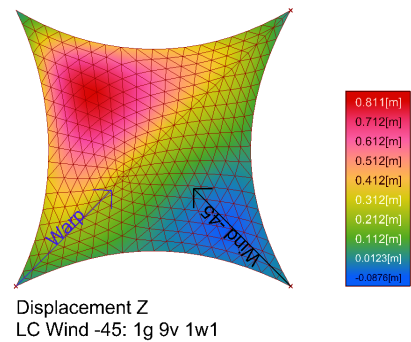


Table 5.5: Snow load and enveloped results

When the membrane is loaded with snow the ridge cable comes into play and significantly reduces deformation along the diagonal from highpoint to highpoint. Figure 5.11 shows that the largest deformations occur in the areas with the largest distance from edge and/or ridge cables.

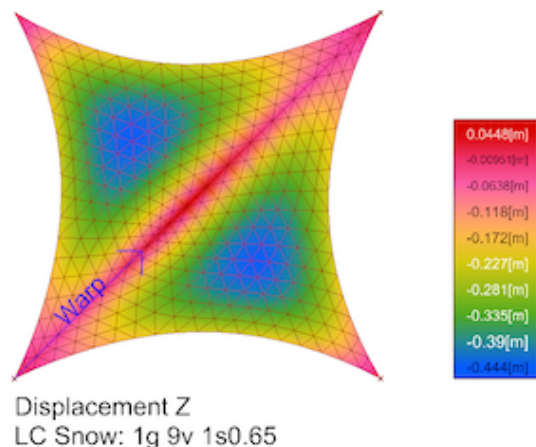


Figure 5.11: Displacements due to snow

5.3.8 Stress Analysis

5.3.8.1 Membrane Prestress

Table 5.6 shows the resulting prestress in warp (x) and weft (y) direction.

The X-direction corresponds with the warp direction and therefore runs parallel to the ridge cable.

The resulting prestress in Y direction is slightly higher than in X-direction, most likely because of the missing cable in this direction.

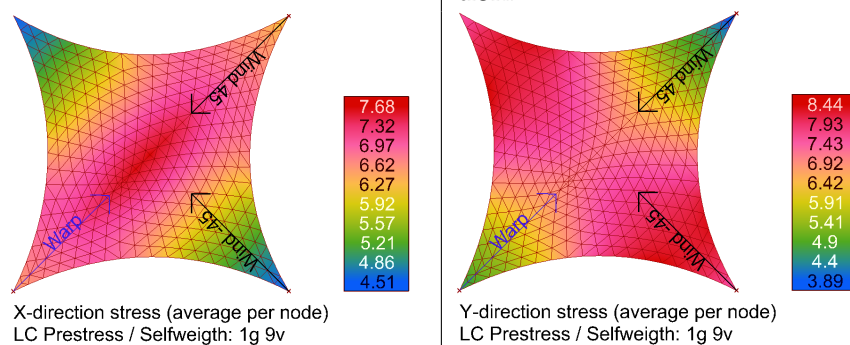


Table 5.6: Membrane Prestress

5.3.8.2 Membrane Stress for Loadcas Snow

Table 5.7 shows the resulting membrane stresses in warp (x) and weft (y) direction due to snow load.

Looking at the maximum stress in X-direction the impact of the ridge cable becomes clearly visible.

In Y-direction, the highest stresses from snow load appear in the corners of the low points.

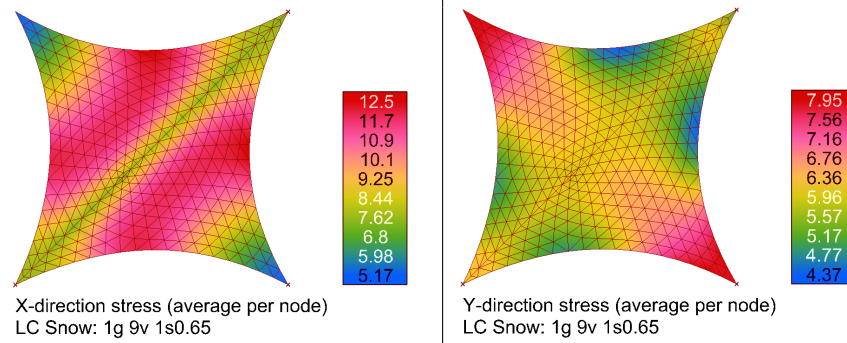


Table 5.7: Membrane stress due to snow load

5.3.8.3 Maximum Membrane Stresses (Enveloped Result)

Table 5.8 shows the maximum membrane stresses in warp (x) and weft (y) direction. These maximum values will be used to select the fabric based on tensile strength (reduction factors need to be applied).

The enveloped result for stress in X-direction look very similar as the result for snow load only. This load case seems to be the most relevant for stress in warp direction.

The maximum stresses in Y-direction are higher than in X-direction because there is no supporting cable running in this direction.

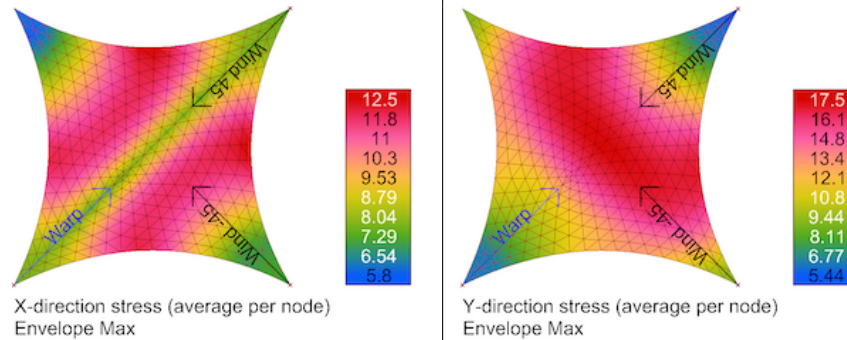
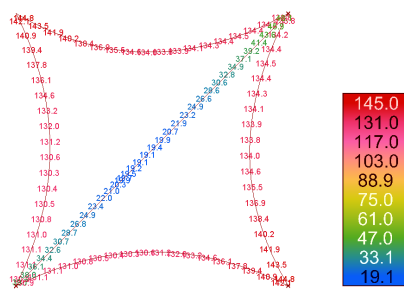


Table 5.8: Maximum membrane stress

5.3.8.4 Axial Forces in the Cables

Table 5.9 shows the resulting axial forces in the cables.

As expected, prestress leads to fairly harmonic axial forces in the edge cables.



The maximum forces (envelope results) in any of the edge cables is 252kN. For the ridge cable up to 115kN is expected.

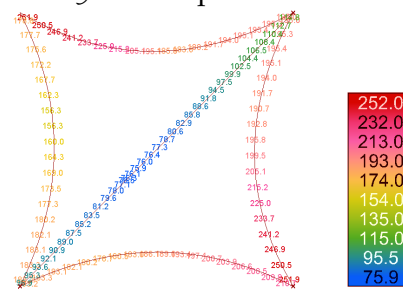


Table 5.9: Axial forces

5.3.9 Conclusion

This example proves that the full engineering workflow for the design of a membrane structure as described in 3.7.2 can be executed with the developed software library PAMELA.

The results of the analysis show that the maximum stresses in the membrane are in Y-direction and imposed by the wind loads and not by the snow load. Maximum stress in Y-direction is 17.5 and in X-direction “only” 12.5. Since most membrane materials have a (slightly) higher tensile strength in warp (X) direction, one could come to the conclusion that the orientation of the fabric should be changed by 90 degrees. But the warp direction was chosen based on the direction of the ridge cable. Now the question arises if the cable, which was introduced to stabilize the membrane for snow loads, runs in the right direction? Defined maximum for snow load is 0.65kN/m^2 , defined maximum for wind load is 1.0kN/m^2 . I was curious to see how the membrane behaves without the ridge cable and if the same material specifications would be sufficient. Thanks to the parametric design, this was easy to do: disable the ridge cable component and the solver starts recomputing since the change in the model is discovered automatically. All result views are automatically updated afterwards.

5.3.9.1 Membrane without Ridge Cable

To get the solution converge without the ridge cable, prestress factor had to be increased from 9.0 to 9.5. The results of the analysis are not presented in detail but just summarized:

- Slightly higher prestress is required, but the membrane could still be build using a fabric of Type II.

- The maximum stress in X-direction is still lower than the stress in Y-direction, therefore the orientation of warp direction could be changed by 90 degrees.
- Omitting the ridge cable from high-point to high-point will lead to lower overall costs.
- An option to further investigate would be to have a cable running from low-point to low-point to stabilize the membrane against wind suction. It's well possible that by this cable the forces in the edge cables could be reduced and cables of type PG40 become sufficient for all edge cables (currently 2x PG40 and 2x PG55).
- These findings are all based on the given load cases and especially wind directions and the associated cp values. If any of these factors is changed, the results change as well.

5.4 THE REAL-LIFE PROOF OF CONCEPT

It's good and important to crack numbers, and as the section 5.2 shows, it sometimes can be very important to start digging for the 6th decimal place. The real proof of whether something is working or not, however, is still provided by the fact that it is implemented, built. Last but not least, the difference between theory and practice is that there is no difference in theory.

Following this principle and the slogan "use what you sell", I have used PAMELA as often as possible in customer projects. First only for form finding and visualization, then for patterning and finally for structural analysis as well. So I could continuously test, improve and extend the implementation. A true iterative, agile development process.

6

SUMMARY

This work shows that it's possible to develop a fully parametric software for form finding, structural analysis, patterning and 3D visualizations of membrane structures as an easy deployable and installable plugin to Rhino Grasshopper. The interface components provided allow the software to be used by anybody with understanding of membranes and CAD, especially no programming skills are required. By introducing the concept of smart component groups (3.6.1) the usability was improved significantly, the minimal number of clicks required to build a solution speeds up the process noticeable and also reduces the risk of errors and inconsistent configurations. Some of the implemented concepts are regarded as unique selling propositions.

DOCUMENTATION To be able to take full advantage of the benefits, however, the provided components must be known, which is associated with learning effort at the beginning. If the software is to be made available to a wider circle of users in the future, appropriate examples must be provided. The analysis of a hyper described in section 5.3 can be regarded as such an example. In order to keep documentation up to date, special functions were implemented which generate LaTeX-compliant descriptions of the components targeted at end users and not programmers directly from source code (see 4.12.2). To capture the dynamic parameters of the components, this tool actually instantiates each component, parses the parameters, adds the component to the Grasshopper canvas and automatically creates a screenshot which in turn is included in the documentation. With this tool it is possible to generate up-to-date documentation at any time. It is easy to convert LaTeX based documentation to PDF or other formats.

PROOF The proof for the correct functioning is provided by comparing the results with the reference implementation based on Matlab scripts. The Matlab reference implementation was previously tested by its author and is therefore assumed to deliver correct results. The test results and examples listed in the document show that it was possible to finish all relevant assignments from the study program with the PAMELA software library, including form finding, structural analysis, patterning, detailing and 3D visualizations. In addition, and even more important, the software is now being used successfully in real live customer projects to build beautiful membranes.

FULFILLMENT OF REQUIREMENTS In chapter 2 I have summarized my own requirements for a parametric membrane software. The next section contains a list of limitations and topics with room for improvements. My overall summary is that I am very happy with the overall result and I am frequently using the PAMELA package which supports my workflows and perfectly suits my needs.

6.1 LIMITATIONS

Although I am very satisfied with what I have achieved, the current implementation has certain limitations. They are listed in the following in random order.

PROGRESS INFO Structures with a larger number of DOF's require a certain amount of processing time. The Grasshopper user interface currently does not support any kind of reporting progress information. This can lead to a "frozen" appearance of Grasshopper and Rhino while computations are executed. The current workaround is to open the logfile to follow the progress reported to the log (loglevel *Info* is ideally suited for this).

PERFORMANCE The performance of a software has a great impact on the usability. As described in 3.9.2 support for multithreading is added in Grasshopper only starting with Rhino for Windows, Version 6.0. A goal for the future development is to migrate the whole software library to Rhino 6.0 and take full advantage of the additional concepts available.

PLATFORM SUPPORT It is a goal to natively support the two operating systems Microsoft Windows as well as Mac OS X. Section 3.3 points out that this will be possible once the library is migrated to Rhino version 6 and the Eto framework is supported. This is regarded as requiring minor effort only.

LICENSING Currently no licensing module is implemented which I take as a reason not to make the software available to other people at the moment until it's clear if a commercialization will be considered or not.

USE CASES The current implementation has been continuously improved and adapted and proofed to be very useful. But there are still many more use cases that could be envisioned as helpful and good to implement.

MESHING / REMESHING Meshing is key functionality of the whole program and process. The currently implemented meshers work well, but it is not out of the question that the function could be further optimized. As pointed out in 3.9 3rd party meshes can already be used today if required by a special case.

WARP ORIENTATION In section 3.10.2 it was pointed out that the current approach of projecting warp direction has some limitations. This will be an interesting field for further research and solutions could become additional unique selling propositions.

ETFE Currently internal pressure can be specified as part of the load cases and therefore structures based on ETFE cushions could be dealt with as well. But this has not been tested or used so far.

NATIONAL CODES For structural analysis the model already contains material properties and cross section data for all elements. Together with the resulting stresses and forces, checking of the model against national codes would be possible. Time constraints so far did not allow to implement such functionality.

PATTERNING The most important components for the patterning process have been developed and are included in the library: Geodesic lines based on mesh surface, refinement of a mesh to have a path following the geodesic (or any other) line, splitting (dividing) a mesh along a mesh path and flattening meshes (panels) from 3D to 2D. On the other hand functionality in order to find the optimal locations for the geodesic lines or to orient and lay out the panels as well as for adding compensation values have only been prototyped so far and usability of them could be further improved. But first more experience with this process must be gained.

6.2 OUTLOOK

Now that my studies are coming to an end, the time pressure is also easing a bit and my focus has to shift back to customer projects to earn a living. I personally want to continue this project and it would also be great to see other professionals using it to build their membranes. Since it incorporates a fully parametric approach, it would also be well suited to be used at Universities to teach parametric design and membranes. A lot of time has been invested up to now and great care was taken not to include any third party libraries which would prevent a commercialization.

6.2.1 Business Model

A major decision that needs to be taken is how this software is made available to others. An open source approach where the software is given away for free has the potential to quickly build a community which in turn generates valuable feedback and ideas for future development. On the other hand the main focus is membranes and it is not obvious that skilled software engineers would join the community and actively contribute to the further development.

If commercialization is considered, a business case including an in depth market analysis needs to be written. Additional manpower to continue the development and to provide a reliable support infrastructure for the world wide community of membrane professionals would be needed.

Implementing a licensing module allowing to grant other people access to the software for a trialling period would allow to get some first feedback and to further investigate the market potential. Therefore, such a module should be implemented as soon as possible.

6.2.2 Integration with other Software

Due to the flexible architecture chosen, it would also be possible to integrate PAMELA with other software. Some options are:

- Karamba [18] is a fantastic structural analysis package for Grasshopper but with no or only very limited support for membranes. Karamba could be used for the general structural analysis tasks and PAMELA for the membrane specific workflows and evaluations.
- The Carat++[23] solver used by Kiwi3D[24] works based on input files. Integrating this solver into PAMELA can be envisioned the same way Matlab is currently integrated (see 5.2).
- Just very recently I discovered the software library COMPAS[26] developed at the Block Research Group at ETH Zurich. This Python based library is also available for Rhino and Grasshopper and incorporates interesting functionality regarding mesh creation and modification as well as different algorithms for form finding. Combining the power of COMPAS and PAMELA could lead to a very powerful toolset and a community of highly skilled researchers.
- Kangaroo[17] as an interactive physics/constraint solver and Grasshopper plugin seems to have a broad community of users but does not especially support membrane specific workflows and requirements.

6.3 PERSONAL FINDINGS

By implementing this software library I gained insight knowledge of the the Finite Element Method and algorithms applied in the process of designing and dimensioning membrane structures. In order to gain the ability to write this software program I had to learn a new programming language (C#) and in particular I had to explore the 3rd party library RhinoCommon, which has a very large feature set but is only very sparsely documented. I got to know Grasshopper as a very powerful tool and am fascinated by the possibilities offered by parametric design. Last but not least I discovered the power of LaTeX as a document preparation system to write this Master Thesis.

I am happy and proud to now have a software library which I can use for my daily work with membranes.

Don't let membranes drive you crazy...
... get crazy about membranes!
PAMELA will happily assist :-)

BIBLIOGRAPHY

- [1] Martin Brown. *NDN Membrane Software*. URL: <https://dokumentips/documents/ndn-membrane-software.html> (visited on 03/10/2019).
- [2] Carl Stahl ARC GmbH. *TENNECT formfinder | Carl Stahl ARC GmbH - YouTube*. URL: <https://www.youtube.com/watch?v=scuvs5fx8yc> (visited on 03/10/2019).
- [3] Bill Cole. *What is software usability?* URL: <https://blog.rocketsoftware.com/2014/11/software-usability/#.XIX61i1oRYg> (visited on 03/11/2019).
- [4] Gerry D'Anza. *RhinoMembrane v 3.0 For Rhino 6.0 | Food4Rhino*. URL: <https://www.food4rhino.com/app/rhinomembrane-v-30-rhino-60> (visited on 03/10/2019).
- [5] Dlubal Software. *Dlubal Structural Analysis and Design Software*. URL: <https://www.dlubal.com/en-US/solutions/application-areas/software-for-form-finding-and-cutting-pattern-membrane-structures> (visited on 03/10/2019).
- [6] ExactFlat. *ExactFlat*. URL: <https://www.exactflat.com/> (visited on 03/10/2019).
- [7] Formfinder Software GmbH. *Formfinder*. URL: <https://www.formfinder.at/> (visited on 03/10/2019).
- [8] E Gamma, R Helm, R Johnson, and J Vlissides. "Design Patterns – Elements of Reusable Object-Oriented Software." In: *A New Perspective on Object-Oriented Design* (1995). ISSN: 02016361. DOI: 10.1093/carcin/bgs084. arXiv: dd.
- [9] IMS e. V. *Archineer®*. URL: <https://www.ims-institute.org/membrane-structure-program/master-archineerr/archineerr.html> (visited on 03/07/2019).
- [10] K3Tent. *K3-Tent*. URL: <http://k3-tent.com/> (visited on 03/10/2019).
- [11] Donald E. Knuth. *Knuth: Computer Programming as an Art*. 1974. URL: <http://www.paulgraham.com/knuth.html> (visited on 03/07/2019).
- [12] MPanel Software Solutions LLC. *MPanel Software*. URL: <https://mpanel.com/> (visited on 03/10/2019).
- [13] Membranedetail. *Membranedetail*. URL: <https://www.membranedetail.com/> (visited on 03/10/2019).
- [14] Membranes24. *Membranes24*. URL: <https://www.youtube.com/watch?v=ANGvZNSpFAQ> (visited on 03/10/2019).

- [15] Peter Novýsedlák. *Tensile Structures – Numerical Design Techniques*. 2017.
- [16] PFEIFER SEIL- UND HEBETECHNIK GMBH. *PFEIFER-Zugstabsystem Typ 860*. URL: www.pfeifer.de (visited on 03/17/2019).
- [17] Daniel Piker. *Kangaroo3d*. URL: <http://kangaroo3d.com/> (visited on 03/10/2019).
- [18] Clemens Preisinger. *Karamba3D – parametric engineering*. URL: <https://www.karamba3d.com/> (visited on 03/10/2019).
- [19] Robert McNeel & Associates. *Grasshopper - algorithmic modeling for Rhino*. URL: <https://www.grasshopper3d.com/> (visited on 03/08/2019).
- [20] Schweizerischer Ingenieur- und Architektenverein (SIA). “Norm SIA 261(2003) Einwirkungen auf Tragwerke.” In: *SIA 261:2003 und SIA 261/1:2003* (2003).
- [21] Scrum Alliance. *What is Scrum?* URL: <https://www.scrumalliance.org/> (visited on 03/07/2019).
- [22] Natalie Stranghöner et al. “Prospect for European Guidance for the Structural Design of Tensile Membrane Structures.” In: *Science and Policy Report* (2016). DOI: 10.2788/967746.
- [23] TUM. *Carat ++*. URL: <https://www.st.bgu.tum.de/software/forschung/carat/> (visited on 03/10/2019).
- [24] TUM and structure. *Kiwi!3D*. URL: <https://www.kiwi3d.com/> (visited on 03/10/2019).
- [25] Technet GmbH. *EASY*. URL: <https://www.technet-gmbh.com/loesungen/leichte-flaechentragwerke/> (visited on 03/10/2019).
- [26] Tom Van Mele, Andrew Liew, Tomas Mendéz, and Matthias Rippmann. *compas: A framework for computational research in architecture and structures*. 2017. URL: <https://compas-dev.github.io/main/overview.html>.
- [27] Gunther Verheyen. *Scrum is not an acronym | Ullizee*. URL: <https://guntherverheyen.com/2014/01/09/scrum-is-not-an-acronym/> (visited on 03/07/2019).
- [28] Wikipedia. *Wikipedia: Grasshopper 3D*. URL: https://en.wikipedia.org/wiki/Grasshopper_3D.
- [29] WinTess. *WinTess*. URL: <http://www.wintess.com/> (visited on 03/10/2019).
- [30] IxRay ltd. *ixCube*. URL: <http://www.ixray-ltd.com/> (visited on 03/10/2019).