

Combinatorial Meshing for Mechanical FEM

Doctoral Thesis

(Dissertation)

to be awarded the degree Doctor of Engineering (Dr.-Ing.)

submitted by

Henrik Stromberg M.Sc.

from Herten

approved by the Faculty of Mathematics/Computer Science and
Mechanical Engineering, Clausthal University of Technology,

Date of the oral examination

20th of July 2023

Dean: Prof. Dr. rer. nat. Jörg P. Müller

Chairperson of the Board of Examiners: Prof. Dr.-Ing. Gunther Brenner

Supervisor: Prof. Dr.-Ing. Armin Lohrengel

Reviewer: Prof. Dr. Rüdiger Ehlers

Dissertation Technische Universität Clausthal

D 104

Published with a CC-BY-NC-ND License

Contents

List of figures	V
List of tables	VII
List of symbols	IX
List of abbreviations	XIII
Abstract	XV
1 Introduction	1
2 Requirements of Mechanical Engineering toward FEM Software	5
3 State of the Art	11
3.1 Finite Element Method	11
3.1.1 Solved Problem	11
3.1.2 Principal of Virtual Work	12
3.1.3 Numerical Solution	16
3.1.4 Mesh Induced Errors in Finite Element Approximations	19
3.2 Computer Science Fundamentals	20
3.2.1 Geometry Representations	20
3.2.2 Complexity Theory	24
3.2.3 Equation System Solvers	31
3.2.4 Graph Theory	32
3.3 Mesh quality	34
3.3.1 Element Quality	34
3.3.2 Mesh Structure	39
3.4 Meshing	42

3.4.1	Tetrahedral Mesh Generation	42
3.4.2	Hexahedral Mesh Generation	43
3.4.3	Mesh Optimization	47
3.4.4	Grid Based Meshing	48
4	Combinatorial Meshing	53
4.1	Consequences from Engineering Needs	53
4.2	The Concept of Combinatorial Meshing	54
4.3	Graph representation of a mesh	57
5	Generation of Super Elements	61
5.1	Problem Definition	61
5.2	Considered Node Locations	64
5.3	Meshing ASP Model	67
5.4	Results	73
6	Combinatorial Mesh Assembly	77
6.1	Problem Description	77
6.2	Residual Volume Integration	77
6.3	ASP Approach	80
6.4	Heuristic Approach	83
7	Entity Mapping	87
8	Demonstration Results	93
9	Summary	101
10	Outlook	103
	References	105
A	Questionnaire	113
B	Considered Node locations	121
C	Super Elements	125

List of Figures

fig. 2.1:	Industries of study participants (Stromberg et al., 2021)	6
fig. 2.2:	Importance of FEM from the interviewee's point of view (Stromberg et al., 2021)	7
fig. 2.3:	Market share of FEM software tools in survey (Stromberg et al., 2021)	7
fig. 2.4:	User satisfaction in the survey (Stromberg et al., 2021)	8
fig. 2.5:	Count of FEM programs interviewees are familiar with (Stromberg et al., 2021)	9
fig. 2.6:	Requested areas of improvements (Stromberg et al., 2021)	10
fig. 3.1:	Used element types	13
fig. 3.2:	ξ to x space transformation	14
fig. 3.3:	hanging nodes	14
fig. 3.4:	Representation of a stiffness matrix of a single element with a framework	18
fig. 3.5:	Residual area between mesh and geometry for a circle discretized with linear and quadratic elements	19
fig. 3.6:	1D, 2D and 3D simplex	21
fig. 3.7:	Polygon	21
fig. 3.8:	Point in polygon test with ray casting (Hormann and Agathos, 2001)	22
fig. 3.9:	Example of a NURBS surface in its parameter and image space	23
fig. 3.10:	Geometric entities of CAD models	24
fig. 3.11:	Comparison of the complexity of both algorithms	27
fig. 3.12:	Comparison of complexity classes	28
fig. 3.13:	Complexity of meshing algorithms in Ansys	29
fig. 3.14:	Setup of a k-d tree with vector space in the upper illustration and the tree below (example from Bentley (1975))	30

fig. 3.15: Example of an octree	31
fig. 3.16: Example graph representing a road map	32
fig. 3.17: Measured correlation of mesh quality metrics with accuracy and stability for solving a linear elasticity problem (Gao et al., 2017)	35
fig. 3.18: Jacobian formulation (Shepherd, 2007)	36
fig. 3.19: Defective element with positive Normalized Scaled Jacobian (Lobos, 2015)	38
fig. 3.20: Graph of singular edges (Pietroni et al., 2022)	40
fig. 3.21: examples of mesh structure (Pietroni et al., 2022)	41
fig. 3.22: Element improvement in Delaunay triangulation	42
fig. 3.23: Hexahedral meshing with sweeping approach	44
fig. 3.24: Sculpt algorithm (Owen and Shelton, 2015)	49
fig. 3.25: 2D octree of a geometry (Yerry and Shephard, 1984)	50
fig. 3.26: 2D mesh created from octree (Yerry and Shephard, 1984)	51
fig. 4.1: Hanging edge between a hexahedron and two tetrahedrons marked in red	54
fig. 4.2: Manufacturing and meshing of an I-beam with a cut groove	55
fig. 4.3: Results of trivial nodes removal for a circular geometry with unrecoverable elements are marked in red	56
fig. 4.4: Graph representation of a mesh	58
fig. 5.1: Possible reduced triangular faces on Super Elements	62
fig. 5.2: Possible reduced quadrangular faces on Super Elements	63
fig. 5.3: Considered Node Locations for hexahedral Super Elements	65
fig. 5.4: Considered Node Locations for prismatic Super Elements	66
fig. 5.5: Considered Node Locations for pyramid Super Elements	66
fig. 5.6: Considered Node Locations for tetrahedral Super Elements	66
fig. 5.7: Used ASP model	68
fig. 5.8: Legal and illegal combination of normal vectors	69
fig. 5.9: Example for a boundary mesh defining the task to be fulfilled by the ASP model	70
fig. 5.10: Boundary mesh for computing a refinement Super Element	76
fig. 6.1: Bad mesh quality due to instability of trivial RV function	78
fig. 6.2: Target function for node coloring optimization	78

fig. 6.3:	Chosen integration subdomains of a hexahedron	79
fig. 6.4:	Chosen integration subdomains of a tetrahedron	79
fig. 6.5:	Chosen integration subdomains of a prism	80
fig. 6.6:	Chosen integration subdomains of a pyramid	80
fig. 6.7:	ASP model to compute node coloring	82
fig. 6.8:	Computation time for node coloring with ASP	83
fig. 6.9:	Computation time for node coloring with heuristic	84
fig. 7.1:	Convergence of a Newton iteration to a far away root	88
fig. 7.2:	Entity Mapping process	89

List of Tables

tbl. 3.1:	Quality metrics examined by Gao et al. (2017)	34
tbl. 3.2:	Evaluation of mesh structure according to Pietroni et al. (2022)	41
tbl. 3.3:	Application perspective on meshing methods (Pietroni et al., 2022)	45
tbl. 3.4:	Qualities of meshes generated with different methods (Pietroni et al., 2022)	46
tbl. 3.5:	Current research on meshing methods (Pietroni et al., 2022)	47
tbl. 5.1:	Relevant cases for Super Element types	64
tbl. 5.2:	Considered Sub Elements for Hexahedrons (Stromberg et al., 2023)	67
tbl. 5.3:	Multiple examples of generated Super Elements optimized for maximum SJ	74
tbl. 5.4:	Optimization results (Stromberg et al., 2023)	74
tbl. 6.1:	Comparison of results for voting heuristic	85
tbl. 7.1:	Required geometry mapping	87
tbl. 7.2:	Results of Entity Mapping process	90

tbl. 8.1:	Results of Combinatorial Meshing on geometries from the MAMBO dataset	94
tbl. 9.1:	Time complexity of all components with respect to element count of the algorithm	102
tbl. B.1:	Considered Node locations for hexahedrons	121
tbl. B.2:	Considered Node locations for tetrahedrons	122
tbl. B.3:	Considered Node locations for prisms	122
tbl. B.4:	Considered Node locations for pyramids	123
tbl. C.1:	Hexahedral Super Elements with different optimization goals .	125
tbl. C.2:	Incidence matrices for hexahedral Super Elements optimized for maximized SJ	130
tbl. C.3:	Incidence matrices for hexahedral Super Elements optimized for minimized element count	138
tbl. C.4:	Incidence matrices for hexahedral Super Elements optimized for minimized node valence	146
tbl. C.5:	Incidence matrices for manually created tetrahedral Super Elements	155
tbl. C.6:	Incidence matrices for manually created pyramidal Super Elements	155
tbl. C.7:	Incidence matrices for manually created prismatic Super Elements	157

List of symbols

Latin symbols

Symbol	Unit	Explanation
B	-	minimum element block count
$\underline{\underline{B}}$	$\frac{1}{\text{mm}^2}$	deflection strain matrix
\mathbf{C}	$\frac{\text{N}}{\text{mm}^2}$	stiffness tensor
C_t	s	time complexity
C_s	-	space complexity
\mathbf{E}	$\frac{\text{mm}}{\text{mm}}$	strain tensor
\underline{f}	N	external forces
F	-	logically false
\underline{F}	N	force vector
$\underline{\underline{g}}$	$\frac{\text{m}}{\text{s}^2}$	gravitational acceleration
h	$\frac{\text{N}}{\text{mm}^2}$	stress algorithm
\mathbf{H}	$\frac{\text{mm}}{\text{mm}}$	Deformation gradient Tensor
i	-	running index
j	-	running index
$\underline{\underline{J}}$	mm	Jacobian matrix of basis functions
J	mm	Jacobian determinant of element
J_{ENS}	-	Element Normalized Scaled Jacobian
J_s	mm	scaled Jacobian determinant of element
k	-	exemplary integer
$\underline{\underline{K}}$	$\frac{\text{N}}{\text{mm}}$	stiffness matrix
l_k	mm	length of element edge k
\underline{M}	N	mass induced forces

Continued on the following page

Symbol	Unit	Explanation
n_k	-	count of k
\mathbf{N}	$\frac{\text{N}}{\text{mm}^2}$	nominal stress tensor
\underline{N}	-	element basis function
\underline{Q}	-	Element quality vector
\underline{r}	N	internal reaction forces
r_V	-	singular node fraction
r_B	-	minimum element block count divided by element count
t	s	time
\vec{t}	$\frac{\text{N}}{\text{mm}^2}$	stress vector
T	-	logically true
\mathbf{T}	$\frac{\text{N}}{\text{mm}^2}$	stress tensor
\vec{u}	mm	deflection
$\delta \vec{u}$	mm	virtual deflection
\hat{u}	mm	node deflection
$\underline{\hat{u}}$	mm	node deflection vector
w_k	-	Gauss integration weight for integration point k
\vec{x}	mm	global Cartesian coordinates
x_k	mm	x coordinate of node k
y_k	mm	y coordinate of node k
z_k	mm	z coordinate of node k

Greek symbols

Symbol	Unit	Explanation
$\vec{\xi}$	-	local Cartesian coordinates in element
ρ	$\frac{\text{kg}}{\text{mm}^3}$	density
σ	$\frac{\text{N}}{\text{mm}^2}$	Cauchy stress tensor

Indices

In this subscripts the thesis provided in the following table are used.

Index	Explanation
max	maximum
min	minimum

Symbol conventions

Symbol	Explanation
a	scalar
\vec{a}	geometric vector
\mathbf{a}	tensor
\underline{a}	vector
$\underline{\underline{a}}$	matrix
\dot{a}	first temporal derivative
\ddot{a}	second temporal derivative
\hat{a}_k	nodal value for node k
\tilde{a}	Voigt notation of tensor a
$f = \mathcal{O}(g)$	f f is not growing faster than g g (Bachmann–Landau notation)
a_n	a in iteration n

List of abbreviations

Abbreviation	Explanation
APDL	Ansys Parametric Design Language
ARG	Aspect Ratio Gamma
ASP	Answer Set Programming
BREP	Boundary REPresentation
CAD	Computer Aided Design
FE	Finite Element
FEM	Finite Element Method
IMW	Institut für Maschinenwesen of the Technischen Universität Clausthal
ISO	International Organization for Standardization
LP	Logical Programming
NP	Non Polynomial
NURBS	Non-Uniform Rational B-Spline
SAT	SATisfiability
SJ	Scaled Jacobian
STEP	Standard for the Exchange of Product model data
RV	Residual Volume
1D	1 Dimensional
2D	2 Dimensional
3D	3 Dimensional

Abstract

This dissertation advances the research of mesh generation for Finite Element Method simulation for mechanical applications. In order to target further research at user needs, a survey is conducted to identify the most pressing issues in FEM software.

The concept of Combinatorial Meshing is proposed as a novel approach to grid based meshing. While conventional grid based meshing works on trivial Cartesian grids, the use of a Precursor Mesh instead of a grid is proposed. Appropriate Precursor Meshes are selected by analyzing the internal feature structure of the provided CAD data.

The cells of this Precursor Mesh are then filled with precomputed mesh templates – called Super Elements. The selection of appropriate Super Elements is modeled as a combinatorial optimization problem. To solve this problem, Answer Set programming (ASP) and a heuristic approach are compared with respect to their time complexity and result quality.

The resulting mesh is a rough approximation of the target geometry which then has to be fitted to the geometric entities. For this process a novel algorithm is presented which is able to automatically identify the geometric entities on which the surface nodes of the mesh have to be drawn in order to generate high quality meshes and correctly approximate the desired geometry.

For the generation of Super Element Meshes, a novel approach based on ASP is developed. In order to enable meshing with ASP, a graph representation of a mesh is developed and the meshing process is formulated as a graph selection problem. It is then solved with an ASP solver for multiple optimization goals. The graph formulation will also aid the theoretical understanding of meshing complexity.

Acknowledgments

The process which finally lead to finishing this work was a long and often exhausting journey over the better part of four years. As with most journeys I started it with the intend of reaching the destination. I started with two things: Heaps of provisions and a well thought-out plan. What could possibly go wrong? While the plan was quickly becoming obsolete as I read more and conducted experiments, the things I carried with me proved to be of great value. It was the good upbringing and love I got from my parents. I am very thankful for what they have given me.

Over the years the guidance of my supervisor Prof. Dr.-Ing. Armin Lohrengel – director of the Insitut für Maschinenwesen (IMW) – proved to be very valuable. Under his leadership of the IMW a awesome team has formed. He always leads by example and helped his assistants to grow both as professionals but also as human beings. Working for him is characterized by having much freedom and being trusted to make good decisions while simultaneously offering guidance when needed.

I also want to thank Prof. Dr. Rüdiger Ehlers for reviewing my thesis. I highly appreciate that he found the time to do so to strengthen the interdisciplinary character of my thesis.

Such a long journey is impossible to successfully finish without good and helpful company along the way. My wife Isabelle Stromberg was the essence of a helping hand for all the way. She always did everything imaginable in order to support me personally, mentally and in professional discussions. Many ideas in this dissertation were forged in our conversations.

Valentin Mayer-Eichberger and I developed a very fruitful cooperation exchanging ideas between our fields. I am very thankful for the ideas and expertise he had to offer.

This team of the IMW was often of great help in many ways. Christian Heinrich with whom I shared the office was always open for discussing ideas and giving valuable

insight. His brought knowledge of technical literature often impressed and helped me. Over the years Dennis Kaczmarek has become a good friend who helped to make Clausthal a habitable place. Moreover we motivated each other to continue with our work. Besides these personal words I want to thank the whole team of the IMW for their support in all the circumstances which occurred over the years. True friends are those who bring you more chocolate to hospital than you can possibly eat in the weeks there.

I want the specially thank Jessica Schulze-Bentrop for proofreading my thesis and helping me to get its language in an acceptable state.

Maren Kaufmann and Tim Deepe contributed to my work through writing project assignments for me. They were both a pleasure to work with.

I am especially thankful for the help of the Wissenschaftlicher Verein of the IMW and its members in helping to get my questionnaire in as many hands as possible.

Finally what keeps you marching on your journey through the dark night is the sound of good music in your ears. For many nights it was supplied by Heaven Shall Burn who definitely did their part in helping me finish this thesis.

1 Introduction

The use of simulations based on the Finite Elements Method (FEM) has proliferated in all branches of mechanical engineering over the last years. Even though FEM software has become more user friendly it is stagnating at a point at which expert knowledge is still need for many tasks. This demand excludes many small organisations from profiting from FEM simulations as they cannot afford to hire experts.

FEM simulations need a representation of a geometry as a mesh. The quality and usability of the results from FEM simulation massively depend on the quality of this mesh. The main driver of user facing complexity in FEM software is meshing, as the generation of high quality meshes often requires extensive geometry preprocessing and configuration of meshing algorithms. Thus, the existing fully automatic methods often yield unusable results so that – in industrial practice – meshing requires extensive manual work by expert users.

When examining commercial FEM software it becomes clear that they do not implement the state of the art of meshing research, but lag behind substantially. This suggests that developers assume that their users will have no substantial benefits from these advancements. From a researcher's point of view this seems hard to conceive. Thus, this dissertation starts by assessing user needs with a survey (see chapter 2). The main result of this survey is that FEM software users are very dissatisfied with the usability and result quality of available meshing methods in commercial FEM software. Improving algorithms which provide robust fully automatic meshing with high quality should thus be the focus of further research in the field.

This dissertation aims at improving available meshing techniques in order to allow for robust, fully automatic meshing of arbitrary parts with reasonable quality. Initially, different measurements of mesh quality are discussed for their practical applicability in section 3.3 in order to select an appropriate quality metric to be

used in this thesis.

At first, strengths, weaknesses and open problems of known meshing methods are assessed (see chapter 3.4). As a result of this analysis grid based meshing is identified as the ideal starting point to develop an algorithm with the desired characteristics: full automation, high result quality and robustness.

However, the state of the art in grid based meshing has issues which require improvement: The quality of some elements may degrade severely, the generated meshes are orientation sensitive, local refinement of meshes is hard to achieve and the algorithms struggle with accurately mapping the mesh to sharp and smooth geometries without user intervention.

In order to improve the state of the art in grid based meshing the novel concept of combinatorial meshing is developed within this dissertation. The central idea of this concept is modeling meshing as a graph selection problem (see section 4.3). The second novel idea is to solve the meshing problem using logical programming (LP), which is enabled by the representation as a graph selection problem. Using LP to solve meshing allows to compute meshes with proven optimality.

The downside of this approach is its exponential time complexity. However, the de facto requirement for the complexity of meshing algorithms with practical use is linear time complexity (see section 3.2.2). An algorithm is searched which allows to leverage the advantages of LP with linear or near linear time complexity when generating a mesh. As third novel idea, the algorithm developed in this dissertation exploits the structure of the CAD (computer aided design) model tree, thus obtaining problem adapted grids instead of Cartesian grids which are hereafter called Precursor Meshes. The developed LP based algorithm works by precomputing geometry independent mesh segments named Super Elements (see chapter 5). To build the target geometry, the Super Elements are combined in a building block like fashion by filling all cells of the Precursor Mesh with Super Elements (see chapter 6). The so created mesh is then fine tuned to fit the target geometry by assigning all parts of the surface mesh to entities of the target geometry such as curves or corners (see chapter 7). Then all surface nodes are drawn on their assigned geometric entities. The resulting mesh fits the geometry. Element quality can be optimized by moving nodes.

In comparison to state of the art grid based meshing techniques meshes generated

with the algorithm developed within this dissertation have the following advantages:

1. Meshes can be guaranteed not to fall below a minimum element quality.
2. Meshes are insensitive to orientation.
3. Meshes may be locally refined.
4. Meshes can represent sharp and smooth geometric features without manual fixing.

2 Requirements of Mechanical Engineering toward FEM Software

FEM simulations have become the dominating method to simulate elastic mechanical systems. Mechanical simulations require high quality meshes. In contrast to other simulation physical domains, hexahedral elements are highly desirable for elastic mechanics (Roca Navarro, 2009; Kremer et al., 2014). This results in meshing being a main part of the total effort of creating a mechanical FEM simulation. Halpern (1997) attributes up to 90% of the time spend on a simulation by engineers to meshing. A report on the use of resources at the Sandia National Laboratories (USA) states that 50% of man hours on linear simulations are used for meshing related tasks, for nonlinear simulations 62% are mentioned (Harwick et al., 2005)¹.

In order to deepen the understanding of user needs for advancements in FEM, a user survey was undertaken. (Stromberg et al., 2021) The remainder of this chapter presents results from this survey. The questionnaire can be found in appendix A. The study was conducted as an online survey with 44 participants from industry and academia. The largest industry in the sample is mechanical and plant engineering (see fig. 2.1). 70% of the participants work in non management positions. 10% are students in assistant positions, 13% have department management responsibilities while the remaining 7% have a position as managing director. Half of the interviewees have 5 or less years of professional experience, about one quarter between 5 and 10 years and the remaining quarter more than 10 years.

The questionnaire identifies company sizes according to the classification of the European Union in terms of employees and turnover. (European Union, 2003) The employers of 40% of the participants are classified micro and small companies with 50 employees or less. 24% are medium size companies with up to 250 employees and the remaining 36% are large companies.

¹ In this context geometry decomposition, meshing and mesh manipulation are viewed as meshing related tasks. The stated numbers are the sum of these categories from (Harwick et al., 2005).

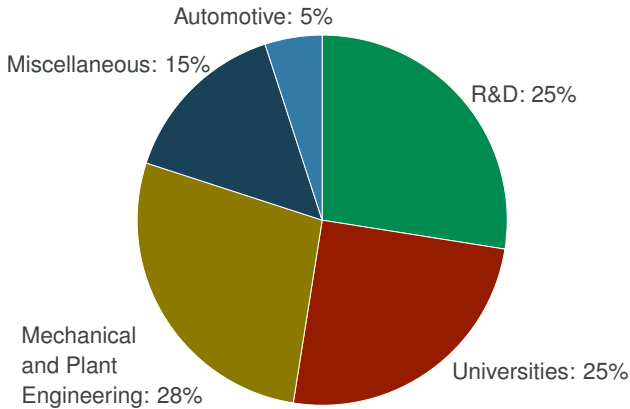


fig. 2.1: Industries of study participants (Stromberg et al., 2021)

93% of the interviewees answer that FEM simulations are either done within their organisation or performed by external engineering service providers. A majority of 70% personally uses FEM software while 21% answer that such work is done by colleagues.

The importance of FEM simulations from the interviewee's point of view is rated high or very high by 82% of the participants (see. fig. 2.2). The given answers do not correlate ($R = 0.26$) with professional experience. When asked for the perceived evaluation of the importance of FEM simulations by the entire organisation the affirmation slightly reduces to 62%.

The participants are asked for the FEM software they use the most. Subsequent questions on general satisfaction and specific wishes for improvement are aimed at the program specified by the participant. The FEM software used by the interviewees allows for the estimation of estimate market shares (see. fig. 2.3).

The reliability of the determined market shares are reduced by the rather small sample size and might be biased slightly towards Ansys, as Ansys is the primary FEM software used at TU-Clausthal. When removing all participants from TU-Clausthal from this evaluation, the market share of Ansys drops to 75%. Even with this uncertainty, Ansys is clearly leading the market. Accordingly, Ansys is used for further tests in this dissertation. Results on the market share for CAD software are included in the original publication.

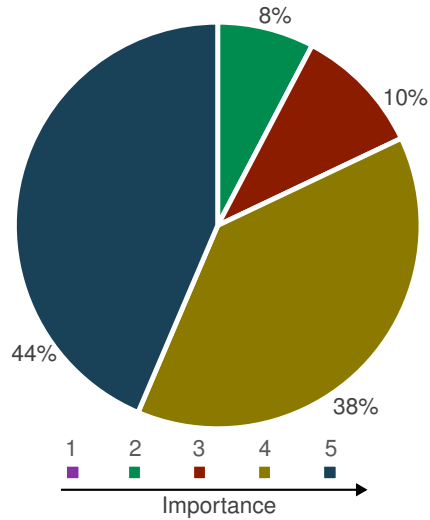


fig. 2.2: Importance of FEM from the interviewee's point of view (Stromberg et al., 2021)

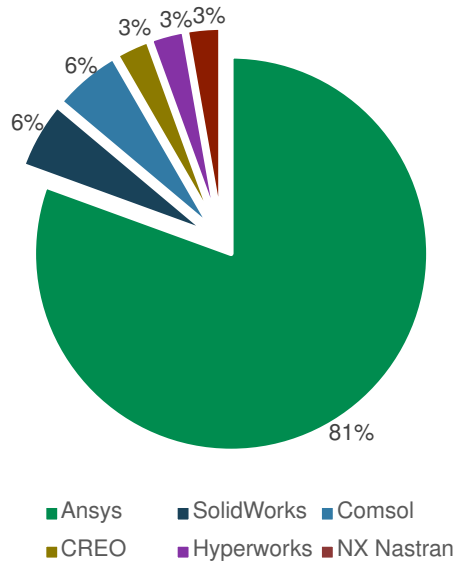


fig. 2.3: Market share of FEM software tools in survey (Stromberg et al., 2021)

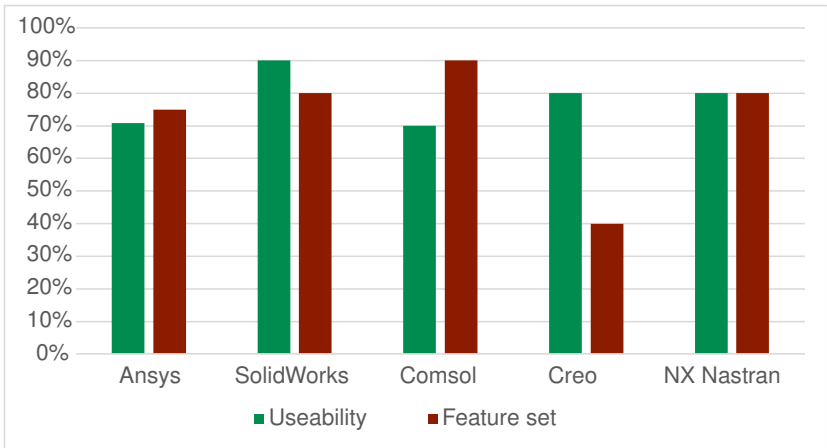


fig. 2.4: User satisfaction in the survey (Stromberg et al., 2021)

The main concerns of the survey are user satisfaction and needs. Participants are asked to evaluate the usability and feature set of their currently most used FEM software. The feature set is generally given high scores. The notable exception here is Creo Simulate. The given wishes for improvement for Creo Simulate explain these low scores as users criticise the lack of features considered basic by them such as computation of reaction forces, contact simulation or geometry optimization in the standard licence package.

Usability is rated similarly for all software packages. This result stands in stark contrast to improvement suggestions made for Ansys which contained strong critique of the usability. For SolidWorks, Comsol Multiphysics and NX Nastran, no suggestions are made by the interviewees which could be compared against the given rating. A possible explanation for the deviation between usability rating and suggestions might be that most survey participants consider the presented user experience to be unavoidable as over 90% of the interviewees only have experience with the FEM software they are rating (see fig. 2.5).

For Ansys significant improvement suggestions are given by the participants. When grouped into categories, lack of software quality is the most frequent complaint. The second most frequent suggestion for improvement is automated meshing with usable quality and improved robustness of the mesher. The third most frequent suggestions are improvement of the documentation and automation. Especially

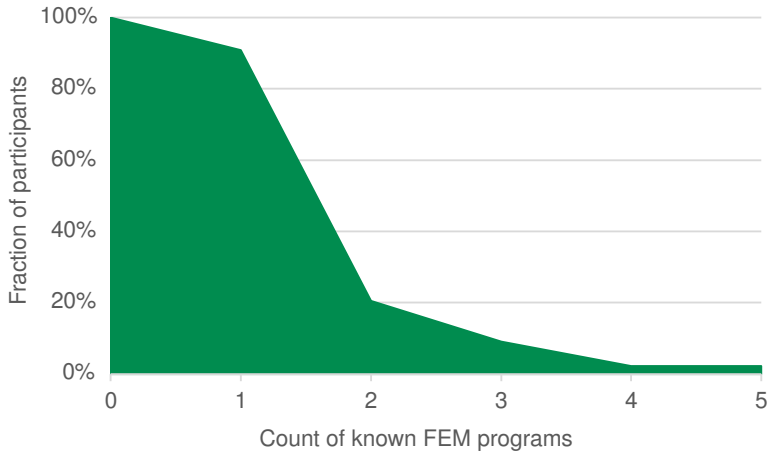


fig. 2.5: Count of FEM programs interviewees are familiar with (Stromberg et al., 2021)

abolishing the proprietary APDL programming language of Ansys is demanded (see fig. 2.6). A more detailed analyses is presented in the publication.

From these results a strong need for the improvement of the mesh quality, and robustness of automated mesh generators can be derived.

Participants are also asked to comment on their willingness to use online FEM tools in a software as a service approach. 62% agreed while 35% disagreed for data protection concerns and 3% disagreed for other reasons.

Lastly the interviewees are asked to comment on the need for an extension of the STEP file format specified by ISO 10303 (ISO, 2020) to contain information on the CAD tree, thus preserving editability. Such a feature was approved by 81% of the participants.

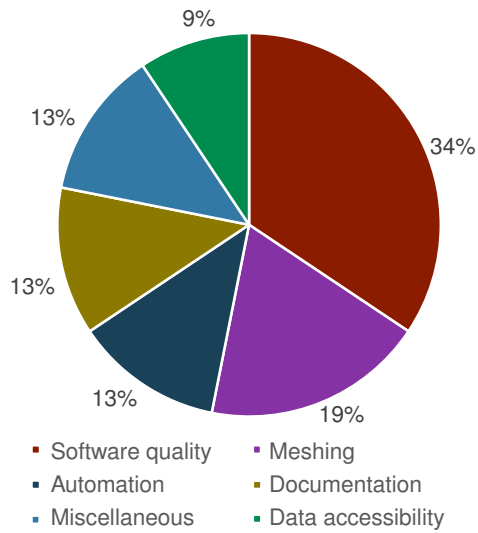


fig. 2.6: Requested areas of improvements (Stromberg et al., 2021)

3 State of the Art

3.1 Finite Element Method

This section provides an introduction to the Finite Element Method. The explanation is based on works by Wriggers (2001) and Bathe (2006). More details can be found in these sources.

3.1.1 Solved Problem

In mechanical engineering, stresses, deflections and other mechanical quantities have to be calculated in order to allow for the adequate dimensioning of parts. The knowledge to perform such calculations was formed in the 18th century. It was more then likely a driving force behind the industrial revolution, as it allowed for a great reduction of prototyping cycles.

Early mechanical laws were restricted to point masses and trivial geometry. Their extension onto an ambiguous continuum yields partial differential equations.

$$\rho \ddot{\vec{u}}(\vec{x}) = \rho \vec{g} + \vec{\nabla} \cdot \boldsymbol{\sigma}(\vec{x}) \quad (3.1)$$

$$\boldsymbol{\sigma}(\vec{x}) = h(\mathbf{E}(\vec{x}), \dots) \quad (3.2)$$

$$\mathbf{E}(\vec{x}) = \frac{1}{2}(\mathbf{H}(\vec{x}) + \mathbf{H}(\vec{x})^T) \quad (3.3)$$

$$\mathbf{H}(\vec{x}) = \vec{\nabla} \vec{u}(\vec{x}) \quad (3.4)$$

(3.1) constitutes the conservation of momentum. On the left hand side is the product of the acceleration field $\ddot{\vec{u}}(\vec{x})$ and the density. It describes the impulse due to acceleration. The first summand on the right hand side is the product of density and the vector of gravitational acceleration \vec{g} . It constitutes the impulse caused by

gravity. Further such terms for other field forces such as electromagnetism may be added. The second summand on the right hand side is the gradient of the stress field $\sigma(\vec{x})$.

3.1.2 Principal of Virtual Work

The principal of virtual work allows to acquire a weak formulation of (3.1). It is known as Galerkin's method. (Galerkin, 1915) The stress algorithm h in (3.2) computes the local stress from strain $\mathbf{E}(\vec{x})$ and other factors such as the strain velocity for viscoelastic materials. Strain is computed as the symmetric part of the deformation gradient \mathbf{H} in (3.3).

These laws constitute a partial differential equation system, which can be solved using the Method of Finite Elements. In order to do so (3.1) is transformed into a weak formulation through the principal of virtual work (see (3.5)).

$$\int_V \delta \vec{u}(\vec{x}) \cdot \rho \ddot{\vec{u}}(\vec{x}) \, dV + \int_V \sigma(\vec{x}) \cdot \delta \mathbf{E}(\vec{x}) \, dV = \int_A \delta \vec{u}(\vec{x}) \cdot \vec{t}(\vec{x}) \, dA + \int_V \delta \vec{u}(\vec{x}) \cdot \rho \vec{g} \, dV \quad (3.5)$$

For this transformation arbitrary virtual deflections $\delta \vec{u}$ are applied to the body which result in the virtual strain $\delta \mathbf{E}$. The virtual deflections have to be 0 wherever displacement boundary conditions are applied. The virtual work of external forces and the resulting potential energy of the body is integrated over the whole body. \vec{t}_0 are stresses induced into the body via its surface. As $\delta \vec{u}$ is unknown it must be factored out. To achieve this, B is defined in (3.6).

$$\underline{\underline{B}} = \frac{\delta \mathbf{E}}{\delta \vec{u}} \quad (3.6)$$

By substituting $\delta \mathbf{E}$ in (3.5) $\delta \vec{u}$ can be factored out (see (3.7)). As $\delta \vec{u}$ is required to be exclusively 0 for \vec{x} with deflection boundary conditions, the left factor of (3.7) will fulfill the equation locally for all Dirichlet boundary conditions. For all other \vec{x} the

right factor must be 0.

$$\delta \vec{u}(\vec{x}) \cdot \left[\int_V \rho \vec{\ddot{u}}(\vec{x}) dV + \int_V \underline{\underline{\sigma}}(\vec{x}) \cdot \underline{\underline{B}}(\vec{x}) dV - \int_A \vec{t}(\vec{x}) dA - \int_V \rho \vec{g} dV \right] = 0 \quad (3.7)$$

The primary solution to a given problem is the deflection field $\vec{u}(\vec{x})$. Other solutions such as stresses are derived from it. In order to solve (3.7) for an actual problem the integrals have to be solved. This is enabled by approximating the problem domain with elements of trivial geometry such as simplices (in 2D triangles). For these trivial geometries (see 3.1) integration schemes can be found to integrate polynomials exactly. Accordingly the solution is modeled as a piecewise polynomial function.

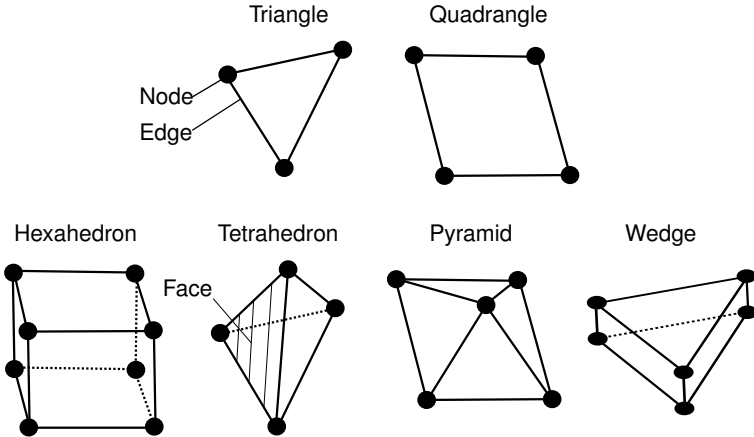
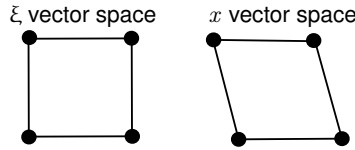


fig. 3.1: Used element types

The mesh consists of elements of various types such as triangles, quadrangles, wedges or hexahedrons. Each element has one basis function for each of its nodes. Basis functions have to be 1 at their corresponding node and 0 at all other nodes of the element. They are only defined within the element. That creates the piecewise polynomial characteristic of the solution. Elements have an internal vector space $\vec{\xi}$ in which the element has its ideal shape and unit size (see fig. 3.2).

fig. 3.2: ξ to x space transformation

The solution $\vec{u}(\vec{\xi})$ is the sum of all basis functions of the element for the given $\vec{\xi}$ weighted with the deflection at each node \hat{u}_k (see (3.8)). Linear or quadratic basis functions $\underline{N}(\vec{\xi})$ are typically used for mechanical FEM applications.

As a discontinuity in the deflection field $\vec{u}(\vec{x})$ causes a singularity in stress and strain according to (3.3), the basis functions between elements must be continuous over element boundaries. This requirement may be circumvented by novel approaches such as non conforming finite element methods but is enforced in classical FEM. As a consequence so called hanging nodes as shown in fig. 3.3 are prohibited.

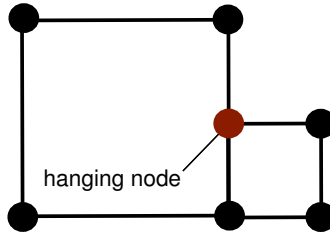


fig. 3.3: hanging nodes

$$\underline{u}(\vec{\xi}) = \sum_{i=0}^k \hat{u}_i \underline{N}_i(\vec{\xi}) \quad (3.8)$$

Polynomial basis functions can be derived trivially with respect to the ξ vector space. However its derivatives with respect to the x vector space are required in order to formulate $\underline{B}(\vec{x})$. For this purpose a transformation $\vec{\xi} \rightarrow \vec{x}$ is required. The most widely used element types are isoparametric. This means, that the same basis functions are used to represent the solution and to transform from ξ space to x space (see (3.8) and (3.9)).

$$\underline{x}(\vec{\xi}) = \sum_{i=0}^k \hat{x}_i \underline{N}_i(\vec{\xi}) \quad (3.9)$$

Hence the node locations \hat{x}_k , which are constant with respect to $\vec{\xi}$ (3.9), can be derived trivially to obtain $\frac{\partial \underline{x}}{\partial \vec{\xi}}$. With these derivatives the Jacobian matrix J of $\underline{x}(\vec{\xi})$ can be assembled (see (3.10)). Its transposed inverse (see (3.11)) can be used to transform ξ space derivatives of $\underline{N}(\vec{\xi})$ into x space derivatives (see (3.12)).

$$\underline{J} = \begin{bmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_1} \\ \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_2} \\ \frac{\partial x_1}{\partial \xi_3} & \frac{\partial x_2}{\partial \xi_3} & \frac{\partial x_2}{\partial \xi_3} \end{bmatrix} \quad (3.10)$$

$$\underline{J}^{T-1} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x_1} & \frac{\partial \xi_2}{\partial x_1} & \frac{\partial \xi_2}{\partial x_1} \\ \frac{\partial \xi_1}{\partial x_2} & \frac{\partial \xi_2}{\partial x_2} & \frac{\partial \xi_2}{\partial x_2} \\ \frac{\partial \xi_1}{\partial x_3} & \frac{\partial \xi_2}{\partial x_3} & \frac{\partial \xi_2}{\partial x_3} \end{bmatrix} \quad (3.11)$$

$$\begin{bmatrix} \frac{\partial \xi_1}{\partial x_1} & \frac{\partial \xi_2}{\partial x_1} & \frac{\partial \xi_2}{\partial x_1} \\ \frac{\partial \xi_1}{\partial x_2} & \frac{\partial \xi_2}{\partial x_2} & \frac{\partial \xi_2}{\partial x_2} \\ \frac{\partial \xi_1}{\partial x_3} & \frac{\partial \xi_2}{\partial x_3} & \frac{\partial \xi_2}{\partial x_3} \end{bmatrix} \begin{pmatrix} \frac{\partial N_i}{\partial \xi_1} \\ \frac{\partial N_i}{\partial \xi_2} \\ \frac{\partial N_i}{\partial \xi_3} \end{pmatrix} = \begin{pmatrix} \frac{\partial N_i}{\partial x_1} \\ \frac{\partial N_i}{\partial x_2} \\ \frac{\partial N_i}{\partial x_3} \end{pmatrix} \quad (3.12)$$

With $\frac{\partial N_i}{\partial \vec{x}}$ calculated, \underline{B} can be assembled. In order to do so, (3.7) must be brought into a matrix form using a tensor to matrix convention such as Voigt notation (see (3.13)). In this equation $\underline{N}(\vec{\xi})$ is a matrix of values of the basis functions. $\underline{B}(\vec{\xi})$ contains derivatives of the basis functions in the x vector space. The stress tensor \tilde{T} is computed using the stress algorithm $h(\tilde{E}(\vec{\xi}), \dots)$. For linear elastic material behaviour $h(\tilde{E}(\vec{\xi}), \dots) = \tilde{C}\tilde{E}$ is used. Here \tilde{C} is the Voigt notation of the elasticity tensor. A more detailed explanation is provided by Wriggers (2001).

$$\delta \underline{u}(\vec{x}) \cdot \left[\int_V \underline{N}(\vec{\xi})^T(\vec{\xi}) \rho \underline{N}(\vec{\xi}) \ddot{\underline{u}}(\vec{x}) dV + \int_V \underline{B}(\vec{\xi})^T h(\tilde{E}(\vec{\xi}), \dots) dV - \int_A \underline{N}(\vec{\xi})^T \tilde{t}(\vec{x}) dA - \int_V \underline{N}(\vec{\xi})^T \rho \vec{g} dV \right] = 0 \quad (3.13)$$

$\underline{\underline{B}}$ is the discrete numerical differential operator of FEM. It allows \tilde{E} to be computed from the vector of node deflections in the element \hat{u} (see (3.14)). For large deflections or strains nonlinear terms of $\underline{\underline{B}}$ be can no longer be neglected, so that a different formula is needed.

$$\tilde{E} = \underline{\underline{B}} \hat{u} \quad (3.14)$$

3.1.3 Numerical Solution

In order to solve the integrals in (3.13) the integration domain is split into the used elements, so that they can be solved elementwise (see (3.15)).

$$\delta \vec{u}(\vec{x}) \cdot \left[\sum_{e=0}^{n_e} \int_{V_e} \dots dV_e + \sum_{e=0}^{n_e} \int_{V_e} \dots dV_e - \sum_{e=0}^{n_e} \int_{A_e} \dots dA_e - \sum_{e=0}^{n_e} \int_{V_e} \dots dV_e \right] = 0 \quad (3.15)$$

Within the elements the integrals are solved using Gaussian quadrature. This method allows the exact quadrature of a given polynomial over a unit domain by sampling the polynomial at specified points – the so called Gauss points – and computing a weighted sum of them. The count and position of the samples depends on the order of the integrated polynomial.

$$\delta \vec{u}(\vec{x}) \cdot \sum_{e=0}^{n_e} \sum_{g=0}^{n_{gp}} w_g \left[\left(\underbrace{\underline{\underline{N}}(\vec{\xi}_g)^T (\vec{\xi}_g) \rho \underline{\underline{N}}(\vec{\xi}_g) \vec{\vec{u}}(\vec{x}_g)}_{\underline{\underline{M}}} \right) + \right. \quad (3.16)$$

$$\left. \left(\underbrace{\underline{\underline{B}}(\vec{\xi}_g)^T h(E(\vec{\xi}_g), \dots)}_{\underline{\underline{r}}} \right) - \underbrace{\left(\underline{\underline{N}}(\vec{\xi}_g)^T \vec{t}(\vec{x}) \right) - \left(\underline{\underline{N}}(\vec{\xi}_g)^T \rho \vec{g} \right)}_{\underline{\underline{f}}} \right] = 0$$

$$\begin{aligned}
& \sum_{e=0}^{n_e} \sum_{g=0}^{n_{gp}} w_g \underbrace{\left(\underline{\underline{B}}(\vec{\xi}_g)^T h(\underline{\underline{E}}(\vec{\xi}), \dots) \right)}_{\underline{\underline{r}}} = \\
& \sum_{e=0}^{n_e} \sum_{g=0}^{n_{gp}} w_g \underbrace{\left(\underline{\underline{N}}(\vec{\xi}_g)^T \vec{t}(\vec{x}) \right) - \left(\underline{\underline{N}}(\vec{\xi}_g)^T \rho \vec{g} \right)}_{\underline{\underline{f}}}
\end{aligned} \tag{3.17}$$

If the problem is statically determinate and deformation speeds are low, $\underline{\underline{M}}$ in (3.16) can be neglected. This simplification is chosen as all further deliberations are valid in both situations. In order to obtain node deflections for degrees of freedom without deflection, the displacement boundary conditions (3.16) is rearranged to (3.17).

(3.17) produces an equation system. If the material law represented by $h(\underline{\underline{E}}(\vec{\xi}), \dots)$ is exclusively linear in $\underline{\underline{E}}$, (3.17) is a linear equation system. Other nonlinearities such as contacts or large deflections are beyond the scope of this section.

In this case $\underline{\underline{r}}$ in (3.17) is reduced to $\underline{\underline{B}}(\vec{\xi}_g)^T \underline{\underline{C}} \underline{\underline{B}}(\vec{\xi}_g) \hat{\underline{\underline{u}}}$ with $\underline{\underline{C}}$ being the linear elasticity tensor in Voigt notation. As a result the linear equation system (3.18) is formed as shown in (3.19).¹

$$\underline{\underline{K}} \hat{\underline{\underline{u}}} = \underline{\underline{F}} \tag{3.18}$$

$$\underbrace{\sum_{e=0}^{n_e} \sum_{g=0}^{n_{gp}} w_g \underline{\underline{B}}(\vec{\xi}_g)^T \underline{\underline{C}} \underline{\underline{B}}(\vec{\xi}_g)}_{\underline{\underline{K}}} \hat{\underline{\underline{u}}} = \underbrace{\sum_{e=0}^{n_e} \sum_{g=0}^{n_{gp}} w_g \left(\underline{\underline{N}}(\vec{\xi}_g)^T \vec{t}(\vec{x}) \right) - \left(\underline{\underline{N}}(\vec{\xi}_g)^T \rho \vec{g} \right)}_{\underline{\underline{F}}}
\tag{3.19}$$

Within the equation system matrix $\underline{\underline{K}}$ most elements are zeros. This is referred to as a sparse matrix. Nodes which share elements will cause nonzero values in $\underline{\underline{K}}$ for their respective degrees of freedom.

This equation system (3.18) can be solved for its unknowns – the node deflections – using an equation system solving algorithm. Such algorithms are described in

¹ Splitting of degrees of freedom with deflection boundary conditions to the right hand side is neglected here for simplification.

section 3.2.3. The non zero elements of \underline{K} have a band structure. The secondary bands k_b are introduced by nodes being part of multiple elements.

Mathematically speaking a linear equation system can be solved exactly, if a solution exists. For practical problems the accuracy is limited by the floating point arithmetic error of the machine solving the system. This error is small compared to mesh induced errors and can be neglected. The process described above introduces three errors which are mesh dependent.

The stiffness matrix \underline{K} (3.20) can be visualized as a mechanical framework. Nonzero elements in the matrix are represented by by rods in this mental model. Fig. 3.4 is such a representation for the stiffness matrix in (3.20). Each rod couples four degrees of freedom – two nodes with two dimensions each.

This representation of the problem could be used in post processing of simulation results to extract force flows by gradually merging rods of the framework. Software pursuing this ideas does not exist.

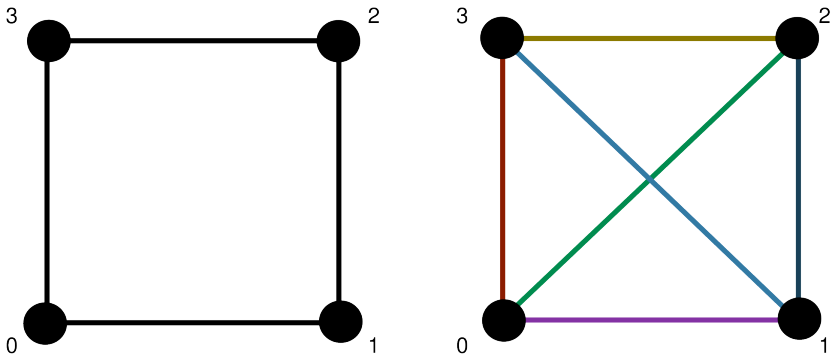


fig. 3.4: Representation of a stiffness matrix of a single element with a framework

$$\begin{bmatrix}
 k_{0,0} & k_{1,0} & k_{2,0} & k_{3,0} & k_{4,0} & k_{5,0} & k_{6,0} & k_{7,0} \\
 k_{0,1} & k_{1,1} & k_{2,1} & k_{3,1} & k_{4,1} & k_{5,1} & k_{6,1} & k_{7,1} \\
 k_{0,2} & k_{1,2} & k_{2,2} & k_{3,2} & k_{4,2} & k_{5,2} & k_{6,2} & k_{7,2} \\
 k_{0,3} & k_{1,3} & k_{2,3} & k_{3,3} & k_{4,3} & k_{5,3} & k_{6,3} & k_{7,3} \\
 k_{0,4} & k_{1,4} & k_{2,4} & k_{3,4} & k_{4,4} & k_{5,4} & k_{6,4} & k_{7,4} \\
 k_{0,5} & k_{1,5} & k_{2,5} & k_{3,5} & k_{4,5} & k_{5,5} & k_{6,5} & k_{7,5} \\
 k_{0,6} & k_{1,6} & k_{2,6} & k_{3,6} & k_{4,6} & k_{5,6} & k_{6,6} & k_{7,6} \\
 k_{0,7} & k_{1,7} & k_{2,7} & k_{3,7} & k_{4,7} & k_{5,7} & k_{6,7} & k_{7,7}
 \end{bmatrix} \quad (3.20)$$

3.1.4 Mesh Induced Errors in Finite Element Approximations

The first mesh induced error occurs at the transformation between x and ξ vector space. For this step the element basis functions are used. However, a given basis function cannot exactly match any given contour (see fig. 3.5). For example a circle can only be approximated using polynomials. Thus a low Residual Volume (RV) between mesh and geometry is desirable.

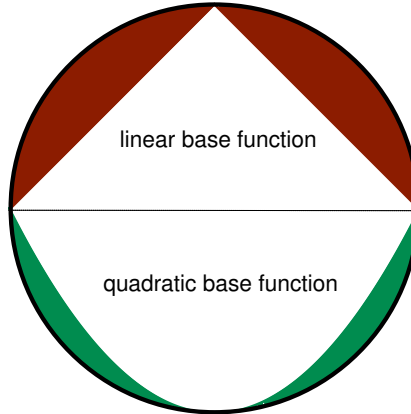


fig. 3.5: Residual area between mesh and geometry for a circle discretized with linear and quadratic elements

The second error caused by mesh quality is the sampling error. Deflection can only be computed at nodes, stress and strain only at Gauss points. For all points inbetween only an interpolation can be provided, rendering the simulation blind for any local effects in this interval. As a result local stress peaks between Gauss

points cannot be detected by an FEM simulation. A mesh should have dense nodes and Gauss points in areas of fluctuating solution quantities.

Third the distortion on an element between x and ξ , space causes excessive stiffening of the elements as Stricklin et al. (1977) and others have found. This shows, that the quality of \underline{B} is very dependent on element distortion.

Nonlinear problems have further associated sources for errors.

3.2 Computer Science Fundamentals

This section introduces a variety of concepts from computer science which are vital to this dissertation, but not broadly known in mechanical engineering.

3.2.1 Geometry Representations

Arbitrary geometry can be represented in various forms within computer memory. These forms differ in geometric accuracy and mathematical intricacy. All shown representations theoretically work in n -dimensional vector spaces, but are typically used for 2D or 3D objects.

In order to save memory, geometry is generally stored as a boundary representation (BREP) if possible. This means, that a watertight² surface of the object is saved instead of the object. This is advantageous as the surface contains fewer points, making this approach more memory efficient. Furthermore common algorithms – which are run on geometry, such as collision tests or point in geometry tests – only interact with an object's surface. A BREP has two types of dimensions: its nodal dimension and its element dimension. The nodal dimension is the vector space in which the body exists. The element dimension is the local vector space of its elements. A triangle for example has an internal 2D vector space but may reside in a 2D or 3D vector space. The element dimension must be less than or equal to the nodal dimension.

²A watertight surface is a surface which fully encloses one or multiple volumes.

Polygons are the most widely used type of BREP. Their prevalent use is computer graphics. A polygon represents a body by approximating its surface with linear elements, usually simplices. These are the most trivial geometric elements in n dimensions (e.g. a triangle in 2D). A simplex has edges which connect it to one other simplex of the polygon (see fig. 3.6). In the 2D case the term face refers to an edge.

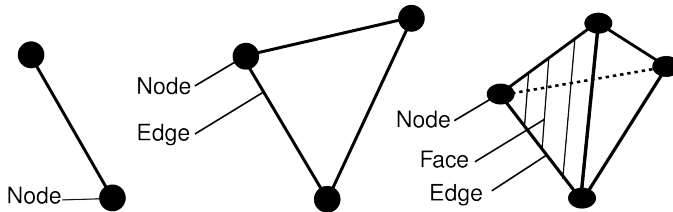


fig. 3.6: 1D, 2D and 3D simplex

A polygon (see fig 3.7) is stored in two matrices. One contains its nodes, the other node IDs of all simplices. The object represented by the polygon is now enclosed in the simplices. Viewed globally the side of the simplices on which the object lies can be found by testing which of the two options yields a finite object. As the global test is expensive to perform in terms of computation time, simplex normal vectors are stored to be able to execute this test locally. For the given storage format there has to be a convention whether the normal vectors point inwards or outwards.

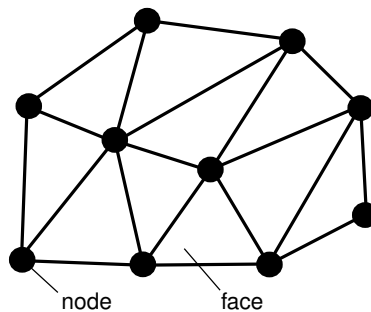


fig. 3.7: Polygon

The accuracy of polygons can be varied by using more or less simplices at a given

spot. However, a polygon will never exactly match a non polygonal geometry such as a sphere. Polygon faces are planar so that resulting collision computations are linear problems. Polygons are mostly used in applications where this limitation is not relevant and simple computation with great speed is required. One such application is to compute propagation of light through the scene in a computer game. Polygon representations are also used in rapid prototyping. The popular STL data format aggregates node coordinates, triangles and normal vectors. For modern rapid prototyping techniques, with lower tolerances, high triangle resolutions are required in order to reduce the geometric approximation error below manufacturing tolerances. This results in large files which are slow to process and increasing inaccuracies when scaling up geometry.

There are several methods to test whether a point is within a polygon. The most reliable according to Hormann and Agathos (2001) is ray casting (see fig. 3.8). In this algorithm, a ray starting from R is cast in an arbitrary direction. If the number of intersections is divisible by 2 the point is outside of the polygon.

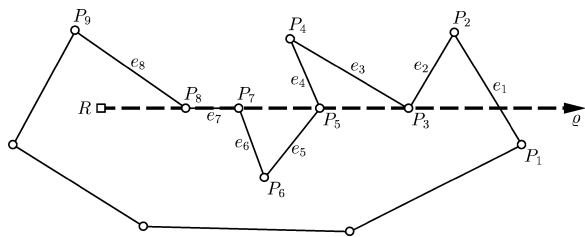


fig. 3.8: Point in polygon test with ray casting (Hormann and Agathos, 2001)

Modern CAD systems use BREPs based on non-uniform rational basis spline (NURBS). NURBS are parametric splines. They take a parameter u as an argument and return a point in any n-dimensional vector space. By inserting values of u for a given interval $[u_{min}, u_{max}]$ into the NURBS function, an approximation of the spline is created. As the NURBS function can be derived symbolically the NURBS's tangent can be computed swiftly. The tensor product of two NURBS curves can be used to create a NURBS surface, which now has two arguments, u and v . Derivatives and normal for NURBS surfaces vectors can be computed symbolically. (Piegl and Tiller, 1997)

In contrast to polygons NURBS splines can exactly represent circles. Extended to

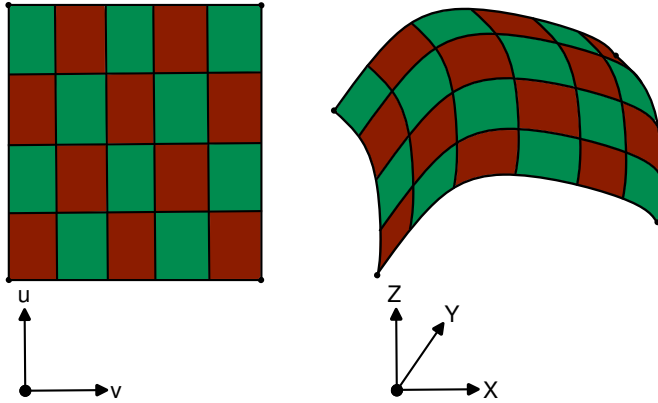


fig. 3.9: Example of a NURBS surface in its parameter and image space

3D this lets NURBS based BREPS exactly represent a cylinder jacket. As such features are very common in mechanical engineering, modern CAD software is founded on the concept of NURBS based BREP geometry representation. As the (u, v) vector space of NURBS surfaces is by definition rectangular, the resulting geometry in the image space is always a deformed rectangle (see fig. 3.9). This limitation makes it very challenging to model arbitrary geometry as shown by Kang and Youn (2015), so CAD software typically resorts to trimming NURBS surfaces with NURBS curves. This is achieved by using a geometrically hierarchical model as shown in 3.10.

With this approach the edges between surfaces of the part are represented by NURBS curves, which in turn start and terminate in geometric nodes. These will be referred to as vertexes within this dissertation to avoid confusion with mesh nodes. Within this data structure, a NURBS surface may be bordered by any number of NURBS curves which are used to trim surfaces into non rectangular shapes, such as the hexagon on the bolt head in fig. 3.10. A NURBS cyclic curve starts and ends with the same vertex. A non cyclic NURBS curve terminates in two different vertexes.

The storage of NURBS BREP models has been standardized by ISO 10303 which implements the STEP (Standard for the Exchange of Product model data) file format. (ISO, 2020) This is a text based file format to exchange CAD models between different CAD software. It stores the resulting geometry and metadata but not the

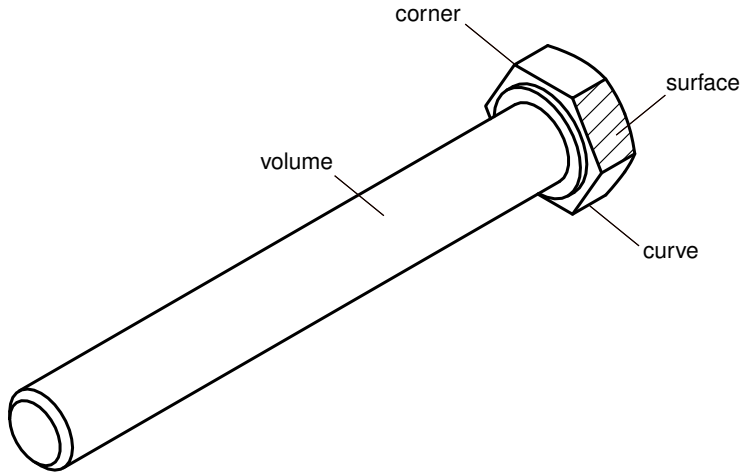


fig. 3.10: Geometric entities of CAD models

user input into the CAD software that generated the geometry. Accordingly it is not possible to easily edit (e.g. change a diameter) of a part provided in STEP format. STEP files are the quasi standard for exchanging CAD files between companies. Other formats with a similar feature set such as IGES exist.

All formats for CAD data described so far contain BREP data. BREPs however are an incomplete description of a solid geometry as they do not describe the part internally. This description though is required for FEM calculations. For this purpose, a volumetric mesh of the element types shown in fig. 3.1 is required. It is stored similar to a polygon mesh with one matrix holding the node coordinates and one incidence matrix for each element type. Each line of the incidence matrices contains the node IDs for the involved nodes for one element.

For the results presented in this dissertation the Open CASCADE CAD kernel is used. (*Open Cascade*, 2022) Surface meshes are generated using GMSH. (Geuzaine and Remacle, 2009)

3.2.2 Complexity Theory

In the context of computer science the term complexity is defined as the amount of resources required to solve a problem or run an algorithm. The main resources

of concern are typically computation time and memory space. The complexity of a problem is the amount of resources consumed by the best possible algorithm. This property might not be known for a given problem. Complexity theory is tasked with identifying the complexity of problems. (Cormen et al., 2001)

The first example problem is the following: given a sorted set A of integers with n elements, an algorithm shall find the index of a given number k .

$$A = \{1, 2, 3, 4, 5, 12\}, k = 3 \quad (3.21)$$

This can be achieved by performing the following algorithm:

```

n = length of A
index = 0
for i = 0 To n {
    if A[i] = k {
        index = i
    }
}
return index

```

For now, time consumption is measured in lines of code executed and space as integers saved. More complex models of resource consumption may be defined. This leads to time complexity $C_{t,l}$ and space complexity $C_{s,l}$ in (3.22), (3.23) for the algorithm above.

$$C_{t,l}(n) = 3n + 4 \quad (3.22)$$

$$C_{s,l}(n) = n + 3 \quad (3.23)$$

In complexity theory resource consumption is viewed with respect to problem size. (Cormen et al., 2001) In the given example this is the number of elements in A . For both time and space the growth is linear in the number of elements in the list.

As a second example the sort of A is exploited. Now searching for k can be achieved with the bisection method (binary search):

```
min_index = 0
max_index = length of A - 1
while min_index <= max_index {
    middle_index = floor ((max_index + min_index) / 2)
    if A[middle_index] == k {
        return middle_index
    }
    else if A[middle_index] < k {
        min_index = middle_index + 1
    }
    else {
        max_index = middle_index - 1
    }
}
return -1
```

The second algorithm has the following complexity:

$$C_{t,b}(n) = 3 + 5 \log_2(n) \quad (3.24)$$

$$C_{s,b}(n) = n + 4 \quad (3.25)$$

The second algorithm will be advantageous in terms of speed. Computation time for the second algorithm only grows logarithmic with n . For any significant n , this difference between the two algorithms will outrun the additive constant 5 as well as the multiplicative constant 3 or 4 respectively. This idea is captured by the \mathcal{O} notation. It groups algorithms and problems in classes by comparing them to benchmark functions. If the benchmark function is an upper bound for the execution time of the algorithm for a sufficiently large n the benchmark function is the complexity class of the algorithm. Accordingly if $f(n) \leq cg(n)$ then $f(n) = \mathcal{O}(g(n))$ may be written to indicate that f is a member of the complexity class g . This comparison is always dominated by the fastest growing term of $f(n)$ so that lower order terms and the multiplicative constant c are neglected. (Cormen et al., 2001) Fig. 3.12 shows some of these complexity classes with problem size n and execution effort N . The two algorithms are thus grouped into the linear and the logarithmic complexity class (see (3.26), (3.27) and fig. 3.11).

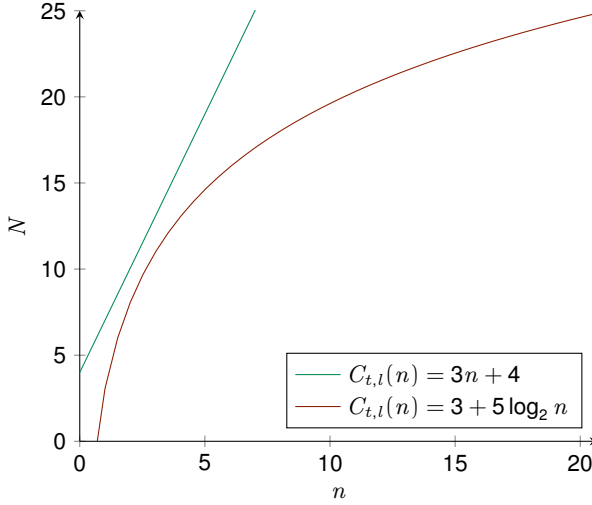


fig. 3.11: Comparison of the complexity of both algorithms

Algorithms are grouped into complexity classes by the type of growth their computation time has with respect to problem size. Problems are assigned to a class if it can be proven that no algorithm can exist which would have a more shallow growth.

$$C_{t,l}(n) = 3n + 4 = \mathcal{O}(n) \quad (3.26)$$

$$C_{t,l}(n) = 3 + 5 \log_2(n) = \mathcal{O}(\log n) \quad (3.27)$$

Fig. 3.12 shows that for large problems the complexity class will dominate execution time and in many cases the overall feasibility of an algorithm. For example there are no known algorithms to perform a prime factorization in polynomial time. This makes the factorization of large numbers practically impossible whereas the inverse is of quadratic complexity. This idea is the mathematical basis of the RSA encryption algorithm. (Rivest et al., 1978) For smaller problems, however, the constant factor in execution time may be more important, making an algorithm with a worse complexity faster in practise. As described in section 3.1, FEM programs generate an equation system which has to be solved. This system may be nonlinear. For linear equation systems the most efficient solver type currently available

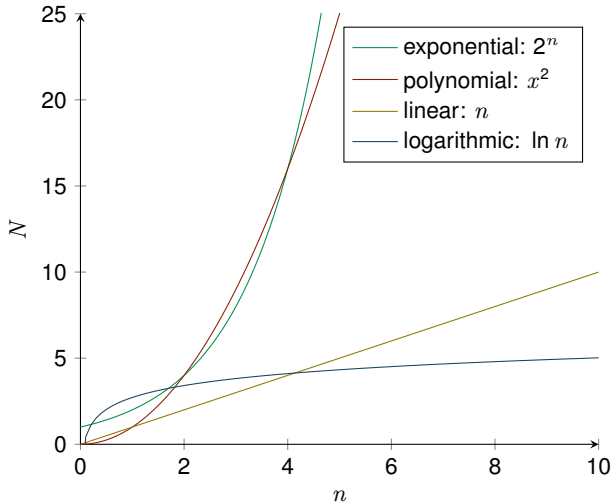


fig. 3.12: Comparison of complexity classes

are multi grid solvers as they have linear complexity ($\mathcal{O}(n)$). (Briggs et al., 2000) For more details see subsection 3.2.3.

As the meshers should not become the bottleneck of any FEM software their time complexity in number of nodes has to be linear or better. Thus it may not contain sub algorithms with significant multiplicative constants that have a complexity above $\mathcal{O}(n)$.

This idea is supported by experimental data gathered from performance tests of the mesher of Ansys Mechanical 2022 R2. (Ansys Inc., 2022) Meshes for two test geometries – B0 and B1 – form the MAMBO test data set (Ledoux, 2022) were generated for different sizings. For B0 a fully hexahedral mesh was generated; for B1 a tetrahedral mesh was created. The runtimes for generating of the meshes are shown in 3.13. Both mesh generators show linear complexity.

Optimizing the complexity of algorithms is in general the most fruitful route to improving software performance. A practical problem with relevance for meshing algorithms is the point proximity problem. Given are a group of points G and a further point P . Now an algorithm is required to return the point of G which is closest to P . The trivial approach to the problem with linear complexity is to iterate over all points in G and compute the distance to P . When now both the number of points in G and the number of points P , which are queried, derived from a common size parameter

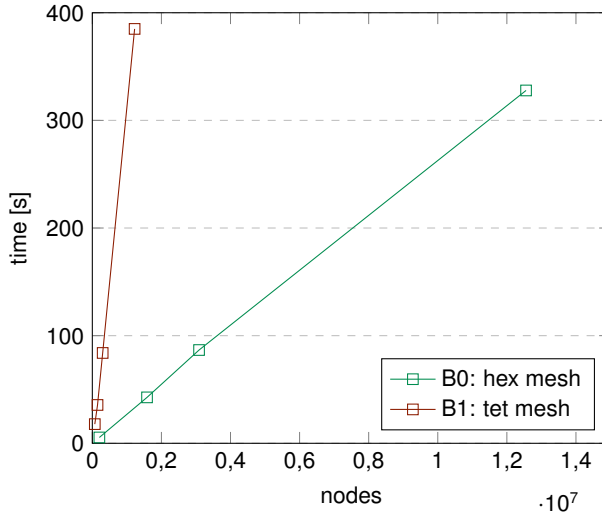


fig. 3.13: Complexity of meshing algorithms in Ansys

such as the mesh size n by a similar function, overall complexity is quadratic in n : $C_t = \mathcal{O}(n^2)$. In order to be able to generate large meshes, an algorithm with a more advantageous complexity is required.

A well known solution to this problem are k-dimensional trees (k-d tree) as described by Bentley (1975). A k-d tree contains the nodes of G and arranges them in a tree which reflects their position. Level by level the tree cyclically iterates over the dimensions of the underlying vector space splitting it in such a way that half of the points present on the current node propagate on each branch (see fig. 3.14). The tree construction is locally terminated by creating a tree leaf (square in fig. 3.14) when the number of points to propagate falls below a threshold. Then a list of these points is written to the leaf.

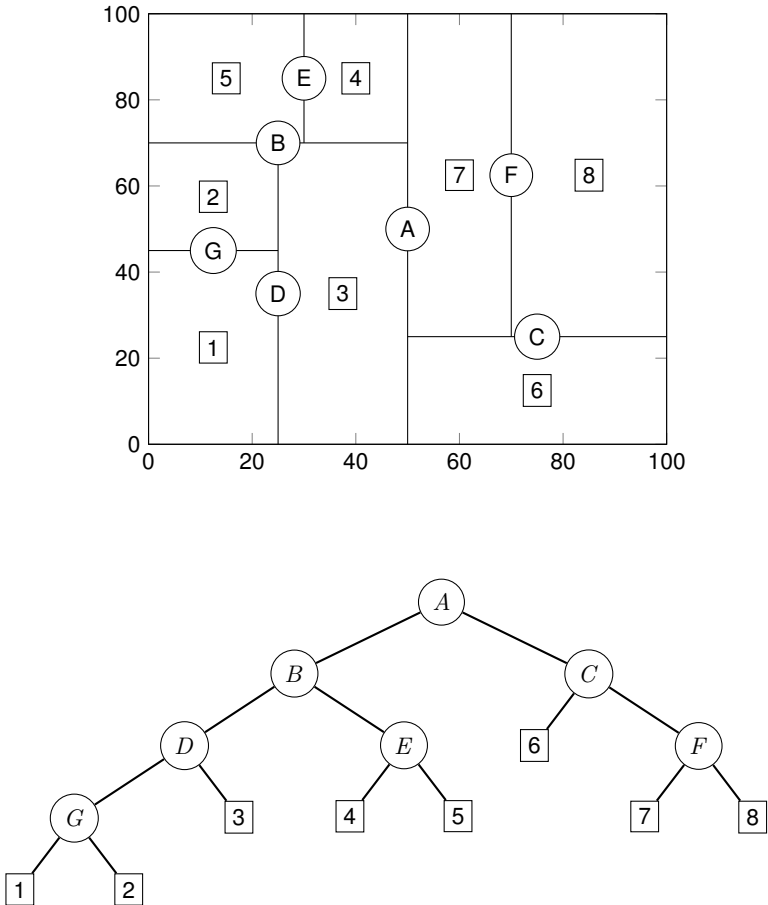


fig. 3.14: Setup of a k-d tree with vector space in the upper illustration and the tree below (example from Bentley (1975))

When looking up the nearest neighbour of a point P , the tree is first searched for the leaf containing P . The distance between P and all points assigned to the leaf is computed and the minimum distance plus the point ID are saved. Now the algorithm runs up the tree back to the root node. On the way it checks at each tree node whether there might be a node on the opposing branch which is closer than the currently best node. If this is the case the currently closest node is substituted. This process has an average complexity of $\mathcal{O}(\log n)$. (Bentley, 1975) Therefore

the case outlined above with $\mathcal{O}(n^2)$ is reduced to $\mathcal{O}(n \log n)$ which reduces it close enough down to the complexity target of $\mathcal{O}(n)$ of multigrid algorithms for typical problem sizes in the magnitude to $n \approx 10^7$.

An alternative to k-d-trees are octrees. While k-d-trees only split one dimension in each level, an octree splits all dimensions at each node (see fig. 3.15). The name is motivated from the eight sub nodes created at one node in the three dimensional case. (Meagher, 1982)

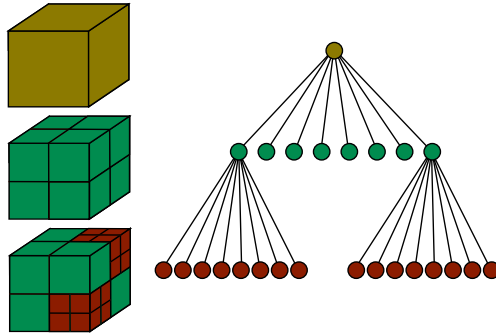


fig. 3.15: Example of an octree

3.2.3 Equation System Solvers

Linear equation systems are generally written in the form $\underline{\underline{A}}x = \underline{b}$. In the context of Finite Element software $\underline{\underline{A}}$ is a sparse matrix. This has implications for efficient storage and solver algorithms.

While a dense matrix stores all entries of the matrix in order, sparse matrix storage only stores nonzero values. For those, their position within the matrix and the value are saved. Sparse storage is advantageous when more then 2/3 of all values are zeros, as row and column positions also have to be saved in sparse storage.³ Executing arithmetic operations on sparse matrices requires adapted algorithms to avoid unnecessary index searches. (Pissanetzky, 1984)

Linear equation systems can be solved with direct or iterative methods. Direct methods strive to compute \underline{x} by rearranging terms. Iterative methods improve a

³This ratio neglects potentially different sizes between index and value data types in memory.

given solution \underline{x}_n to obtain a better solution \underline{x}_{n+1} , thus converging to \underline{x} over multiple iterations. The leading direct method is LU decomposition. When used with state of the art matrix multiplication algorithms it has a time complexity of about $\mathcal{O}(n^{2.4})$ (Demmel et al., 1999). It can be massively parallelized in modern computer architectures as shown by Demmel et al. (1999) in their implementation of the superLU library.

Multigrid algorithms are the state of the art technology for iterative equation system solvers. They exhibit $\mathcal{O}(n)$ complexity while the multiplicative factor of their run time is driven by the spectral radius of \underline{A} . To reduce the required computation time to solve a given problem the spectral radius of \underline{A} has to be minimized. The spectral radius of \underline{A} is its maximum eigenvalue. Due to the better complexity order, multigrid algorithms are generally preferred over direct solvers. (Briggs et al., 2000) Nonlinear equation systems in the form $f(\underline{x}) = \underline{b}$ can be solved using for example the Newton-Raphson algorithm which internally also solves multiple linear equation systems with one of the methods described above. (Ortega and Reinbolt, 2000)

3.2.4 Graph Theory

Graphs as they are subject to graph theory are sets of nodes which are connected by edges. Graphs can be used to model a wide range of problems. Since they are a well researched domain of computer science, modeling a problem in terms of a graph often improves the understanding of the given problem and makes a wide range of algorithms available to use on it. Graph nodes do not have a position, thus a graph can be arranged at will.

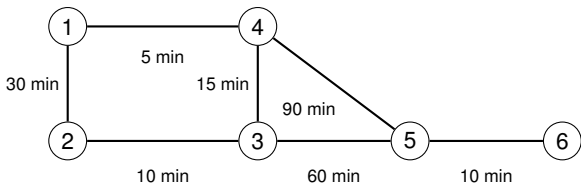


fig. 3.16: Example graph representing a road map

A typical application of graphs is road map navigation (see fig. 3.16). All cities on

the map are modeled as nodes, and their connecting roads are modeled as edges. The number of edges on a particular node is its valency. The time required to travel one of these roads is assigned as an attribute to each edge. The total travel time from 1 to 5 is either 100 minutes when traveling via 2, 80 minutes when traveling via 4 and 3, or 95 minutes via 4. The fastest way can be computed by using, for example, Dijkstra's weighted shortest path algorithm. (Krumke and Noltemeier, 2012) A graph is called planar if it is possible to draw it in 2D space without its edges crossing each other. A road map might not be planar, as bridges may cause edges to cross. Many theorems only hold for planar graphs. For a given graph, tasks such as finding disconnected groups, finding cycles, or finding the shortest way between two nodes with weighted edges can be solved by well-known algorithms. For this thesis, the graph theory library `networkx` by the NetworkX developers (2022) was used.

Within this dissertation, multiple problems are solved by applying answer set programming (ASP) to graph problems. Specifically, ASP is used to find subgraphs of a given graph which satisfy given constraints and are optimal in terms of a given criterion. An ASP program consists of atoms and rules linking the atoms. An atom may be true or false. In the context of finding a subgraph, each node of the parent graph is represented as an atom, denoting whether the node is selected to be part of the subgraph. Within this dissertation, the ASP solver Clingo (Gebser et al., 2019) is used. Clingo uses concepts from satisfiability solving (SAT).

The basic function of a SAT solver is to determine if there is an assignment of values for a set of Boolean variables, for which a provided Boolean formula is true. The SAT solver exhaustively explores the search space of possible value assignments for the variables. In a worst-case situation, this results in testing every solution in the search space. As the search space for n variables is 2^n , the runtime of this algorithm would be exponential in the number of variables. The practical runtime of SAT solvers is usually much better, as they can use heuristics to exclude large areas of the search space by reasoning about few tested solutions. The actual runtime of a SAT solver mostly depends on how well the implemented heuristics of the solver work for the structure of the given problem. In the context of this dissertation, the main advantage that ASP holds over directly using SAT is that ASP allows implementing a reachability constraint easier than SAT. It is needed to ensure that the solution is one cohesive graph. Generally speaking, using a solver which provides capabilities not needed for the problem to solve can be disadvantageous, as it can-

not use heuristics that might be applicable in the simpler given problem. (Knuth, 2015; Biere et al., 2021)

3.3 Mesh quality

3.3.1 Element Quality

In the literature many element quality metrics are described. They differ in applicability for element types and prediction of numerical mesh performance. For hexahedral elements Gao et al. (2017) have studied the correlation of 19 quality metrics with solution accuracy and stability in several applications.

tbl. 3.1: Quality metrics examined by Gao et al. (2017)

Metric	Range	Range*	Trend
diagonal	[0, 1]	[1, 1]	↑
dimension	[0, +∞]	[0, +∞]	↑
distortion	[−∞, +∞]	[0, 1]	↑
edge ratio	[1, +∞]	[1, +∞]	↓
Jacobian	[−∞, +∞]	[0, +∞]	↑
maximum edge ratio	[1, +∞]	[1, +∞]	↓
aspect Frobenius	[1, +∞]	[1, +∞]	↓
mean aspect Frobenius	[1, +∞]	[1, +∞]	↓
Oddy	[0, +∞]	[0, +∞]	↓
relative size squared	[0, 1]	[0, 1]	↑
Scaled Jacobian	[−1, 1]	[0, 1]	↑
shape	[0, 1]	[0, 1]	↑
shape size	[0, 1]	[0, 1]	↑
shear	[0, 1]	[0, 1]	↑
shear size	[0, 1]	[0, 1]	↑
skew	[0, 1]	[0, 1]	↓
stretch	[0, 1]	[0, 1]	↑
taper	[0, +∞]	[0, +∞]	↓
volume	[−∞, +∞]	[0, +∞]	↑

Tbl. 3.1 is a list of all examined mesh quality metrics. Range indicates possible values of the metric while Range* are the maximum boundaries for numerical use of elements. Trend marks whether high or low values indicate a good element. Accuracy is measured with the L_2 norm of all local errors. Accuracy and stability

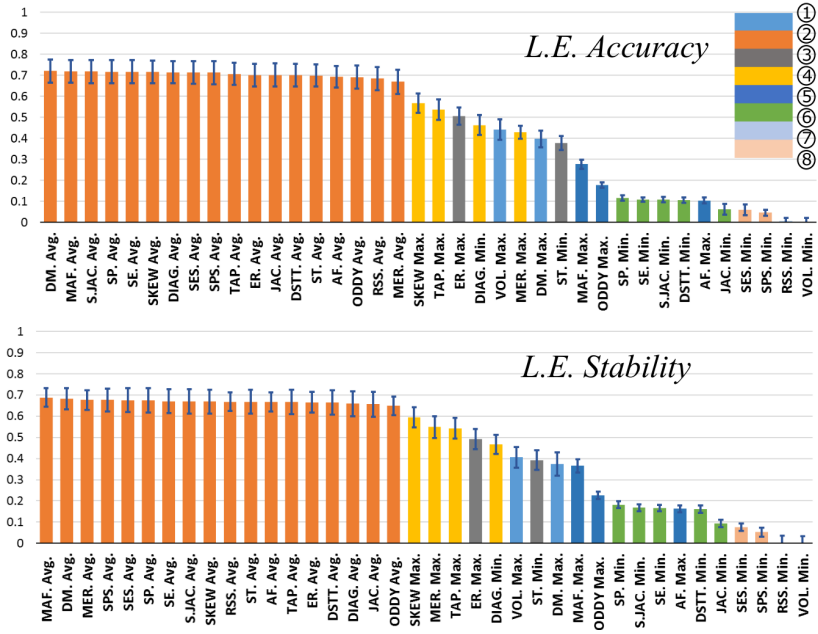


fig. 3.17: Measured correlation of mesh quality metrics with accuracy and stability for solving a linear elasticity problem (Gao et al., 2017)

are evaluated for the primary solution – node deflection. These results are shown in figure 3.17.

For accuracy and stability about half of the metrics are equally well suited to predict mesh performance. The main limitations of the work by Gao et al. (2017) for mechanical FEM are that L_2 norms of errors were evaluated and that only the error of the primary solution is evaluated. The most common problem mechanical engineers solve with FEM is to evaluate the maximum stress of a given part. Stricklin et al. (1977) and others have found the quality of the B matrix to be very sensitive towards element quality. As high stresses correlate with sharp radii which in turn correlate with bad element quality, elements with bad quality often experience high stresses. Thus the maximum stress over the whole part may exhibit an error above the norm of all errors. A followup study evaluating the maximum error for stresses would be very helpful for the field.

The same issues renders the findings of Schneider et al. (2019) on the superiority

of tetrahedral elements with quadratic basis functions over hexahedrons irrelevant for FEM applications in mechanical engineering.

For purely hexahedral meshes the current consensus in the community is to use Scaled Jacobian (SJ) as the main measurement of element quality. The term is slightly misleading as it must not be confused with the Jacobian matrix used in FEM theory (see (3.10)). The Scaled Jacobian is computed for all nodes of a given element. Then the minimum value encountered is used. The Jacobian matrix J_i for a node i is constructed by arranging vectors from the node to its neighbours in a matrix. (Shepherd, 2007)

$$J_0 = \begin{bmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{bmatrix} \quad (3.28)$$

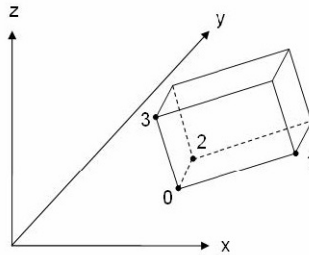


fig. 3.18: Jacobian formulation (Shepherd, 2007)

The order of rows in (3.28) must be chosen in such a way that a hexahedron with identical \vec{x} and $\vec{\xi}$ coordinates returns a positive determinant of J_0 . This has to be done for all 8 nodes. The presented example shows the matrix for node 0.

$|J|$ is a signed volume metric. In order to obtain a universally comparable metric from $|J|$ it has to be scaled with the volume of an optimal element. This can either be done by computing the product of the length of the three longest edges of the element (see (3.29)) or by taking the maximum edge length to the third power (3.30).

$$J_{s0} = \frac{\begin{vmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{vmatrix}}{|\vec{x}_1 - \vec{x}_0| |\vec{x}_2 - \vec{x}_0| |\vec{x}_3 - \vec{x}_0|} \quad (3.29)$$

$$J_{s0} = \frac{\begin{vmatrix} x_1 - x_0 & x_2 - x_0 & x_3 - x_0 \\ y_1 - y_0 & y_2 - y_0 & y_3 - y_0 \\ z_1 - z_0 & z_2 - z_0 & z_3 - z_0 \end{vmatrix}}{(\max(l_i))^3} \quad (3.30)$$

These approaches are not sufficiently distinguished in literature. It must be noted, that (3.29) neglects the elements aspect ratio. Any cuboid element – even with an excessive aspect ratio – is rated with $J_s = 1$. The second option (3.30) punishes high aspect ratios; thus use of the latter is recommended.

Scaled Jacobians has a range of -1 through 1 . A negative Scaled Jacobian shows that the element is self-intersecting. As such elements are not usable for FEM applications a Scaled Jacobian threshold of 0.2 is typical. (Shepherd, 2007)

Lobos (2015) formulates the idea of a unified metric for the quality of elements of different types. Most research is focused on purely hexahedral or purely tetrahedral meshes. A classical mixed element setup of hexahedrons, terahedrons, wedges and pyramids is considered. The quality of hexahedrons is evaluated using the SJ metric described above.

For tetrahedrons, Lobos (2015) uses the inverse of the Aspect Ratio Gamma (ARG) metric (3.31)⁴ which was originally described by Parthasarathy et al. (1994). ARG returns values between 1 and ∞ with 1 being optimal. Consequently ARG^{-1} returns values from -1 to 1 for tetrahedrons with signed volumes.

$$ARG = \left(\sum_{i=0}^5 l_i^2 \right)^{\frac{3}{2}} \frac{\sqrt{3}}{216 V} \quad (3.31)$$

The Scaled Jacobian metric for tetrahedrons is constructed analogously to hexahedrons with a SJ for each node. In the case of a tetrahedron, however, it is

⁴original equation simplified

impossible for all nodes to reach a Scaled Jacobian of 1 simultaneously. The best attainable value for the minimum SJ is $J_{S,opt} = \frac{\sqrt{2}}{2}$ for an equilateral tetrahedron. With this limit known, the Element Normalized Scaled Jacobian J_{ENS} is defined. (Lobos, 2015)

$$J_{ENS} = \begin{cases} (1 + J_{S,opt}) - J_S & \text{if } J_S > J_{S,opt} \\ J_S / J_{S,opt} & \text{if } J_S \leq J_{S,opt} \\ -(1 + J_{S,opt}) - J_S & \text{if } J_S < -J_{S,opt} \end{cases} \quad (3.32)$$

The resulting metric J_{ENS} produces values very similar to ARG^{-1} for typical and defective elements as shown by Lobos (2015).

This principal is now extended onto pyramids and wedges. The optimum SJ of a pyramid is also $J_{S,opt} = \frac{\sqrt{2}}{2}$, so (3.32) can be applied here as well. The top node of a pyramid, however, needs to be treated specifically as it has four neighbouring nodes. For this node the minimum Scaled Jacobian of all four possible combinations with other nodes is chosen. Lobos (2015) observes that inverted base quadrangles (see fig. 3.19) are not detected.



fig. 3.19: Defective element with positive Normalized Scaled Jacobian (Lobos, 2015)

This issue can be overcome by extending the ideas of Lobos (2015). For the whole element the normalized SJ is computed as the minimum of all nodes and a fictive hexahedron on the base quadrangle. The height of the hexahedron should be chosen to be the average of the edge lengths in the quadrangle in order to make a perfect element with $J_S = 1$ possible.

For wedge elements, $J_{S,opt} = \frac{\sqrt{3}}{2}$ as this is the optimum SJ for a wedge. (Lobos, 2015)

Extending the definition of SJ to mixed meshes improves their optimizability as

the SJ metric is a node specific characteristic which allows an optimizer to identify the most problematic nodes within a mesh. (Lobos, 2015) Such an approach is limited to situations in which the identified nodes are moveable. If they are fixed on geometric entities, other nodes of an element with a low SJ have to be moved. The concept is a very useful tool to increase the applicability and quality of mixed meshes.

3.3.2 Mesh Structure

Important tools for the evaluation of mesh structure are derived from the mathematical field of topology. Topology is concerned with the shape of objects while neglecting their precise geometry. Two objects are – topologically speaking – identical if they can be deformed into each other without cutting or gluing. They are homeomorphic. It is necessary, but not sufficient for two bodies to have the same number of holes in order to be homeomorphic. The number of holes of an object is called its genus.

The most regular hexahedral mesh that can be constructed is a Cartesian grid. Within such a mesh each element has 6 adjacent elements, each edge has 4 adjacent elements and each node has 8 adjacent elements. Each node has a valence of 6. For surface nodes these numbers reduce accordingly. As shown by Tautges (2004) structured meshes can be stored more efficiently than unstructured meshes. Furthermore the time and memory efficiency of solving the linear equation system created with FEM is better for lower valence nodes as the number of non zero entries in the stiffness matrix is reduced.

Such a mesh can only be obtained for geometries which are homeomorphic to a cube, since the Cartesian mesh of such a cube can be deformed into the desired geometry. Since a cube is homeomorphic to a sphere only geometries which are also homeomorphic to a sphere can be deformed into a cube according to the classification theorem for surfaces. (Gallier and Xu, 2012) The extension of this problem to higher dimensions is the famous Poincaré conjecture proven by Perelman. (Perelman, 2002, 2003a,b)

Even if a geometry is homeomorphic to a cube the obtainable mesh might be undesirable due to highly distorted elements. This is, for example, the case for geometries with high aspect ratio – needle like – protrusions. Meshes created in the

described manner are called structured. (Pietroni et al., 2022)

Any node of the mesh within the volume that does not have a valence of 6 or a surface node which does not have a valence of 5 is considered to be singular. Edges within the volume which do not have 4 adjacent elements or surface edges which do not have 2 adjacent elements are also singular. In a purely hexahedral mesh both ends of a singular edge are also singular nodes. Consequently singular edges form singular arcs which start and terminate on the geometry's surface (see fig. 3.20). The fraction of nodes which are singular is denoted as r_V . The minimum number of singular nodes in any fully hexahedral mesh is 8. (Pietroni et al., 2022)

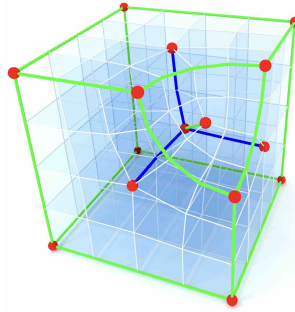


fig. 3.20: Graph of singular edges (Pietroni et al., 2022)

Any hexahedral mesh can be decomposed into structured blocks with 8 singular nodes each. This decomposition is not distinct. However, there is a distinct decomposition with a minimum number of blocks B . The minimum number of blocks divided by the number of elements in the mesh is denoted as r_B . The quotients r_V and r_B can now be used to categorize the structure of meshes (see tbl. 3.2). Examples are shown in fig. 3.21 with the graph of singularities over minimum block decompositions.

tbl. 3.2: Evaluation of mesh structure according to Pietroni et al. (2022)

Structure type	r_V	r_B
Structured	↓	↓ ($B = 1$)
Semi structured	↓	↓
Valence semi-structured	↓	↑
Unstructured	↑	↑

If r_V is small and the mesh is composed of one block it is denoted as structured. Semi structured meshes exhibit low values for both. They can be decomposed into a few structured blocks (see fig. 3.21). Valence semi-structured meshes have more blocks and hence a larger r_B . They are still structured in terms of valence. Unstructured meshes show high values for r_B and r_V . They are neither structured in terms of blocks nor in terms of singular nodes. (Pietroni et al., 2022)

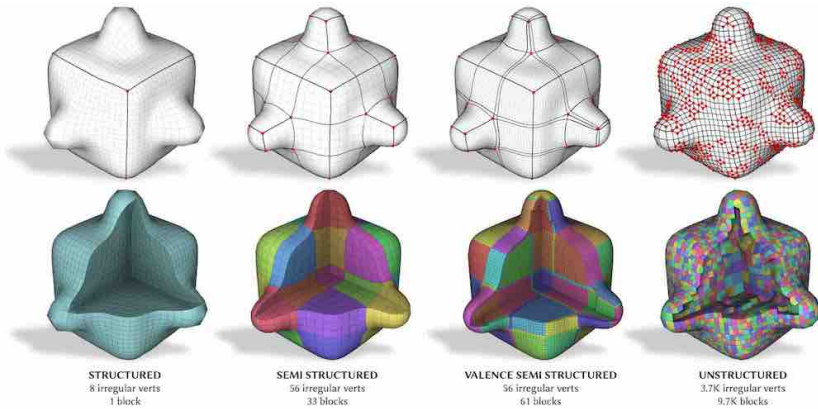


fig. 3.21: examples of mesh structure (Pietroni et al., 2022)

The structure of meshes generated by Sweeping, which is the state of the art for meshing in mechanical FEM depends largely on the structure of the underlying surface mesh. As the volume mesh Sweeping only mildly increases the number of blocks and singular nodes; the resulting meshes are mostly semi structured.

3.4 Meshing

In general meshes for FEM can be grouped into three categories: fully tetrahedral, fully hexahedral and mixed. In order to obtain volumetric meshes a number of methods has been conceived.

3.4.1 Tetrahedral Mesh Generation

Triangular meshes of surfaces are polygons (see. fig. 3.7). Delaunay (1934) is the first systematic approach to the triangulation of surfaces. For the described algorithm, points need to be pre-generated. This may be done by generating random points or with a systematic technique. The algorithm presented by Delaunay (1934) offers an approach to generate and improve a triangular mesh in an iterative procedure. First all points at the boundary of the domain are connected with edges. Then for each edge one point in the interior is selected to construct a triangle. This process finishes with a triangular mesh of poor quality.

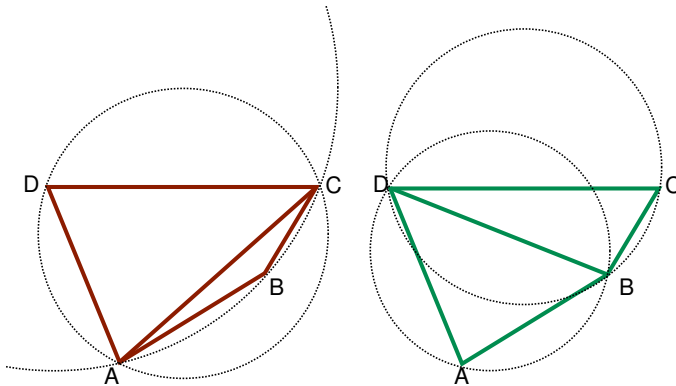


fig. 3.22: Element improvement in Delaunay triangulation

It can be enhanced by iterating over all internal edges of the mesh. Each of these edges is part of two triangles which together have four nodes. If the circumcircle of at least one of these triangles contains the fourth point, the quality of the two triangles can be improved by flipping the diagonal edge. In the example in fig. 3.22 this can be achieved by flipping the diagonal of the quadrangle from \overline{AC} to \overline{BD} .

This algorithm cannot only be performed on planar domains but also on NURBS surfaces. A surface mesh for 3D geometry is obtained according to the geometric hierarchy shown in fig. 3.10 by first creating seed nodes on all curves and then triangulating the surfaces from there as shown, for example, by Cignoni et al. (1993).

Tetrahedral meshers are observed to be less preformant in terms of nodes generated per second compared to hexahedral mesh generators (see fig. 3.13 in section 3.2.2). This however is rather a characteristic of unstructured meshers as they iterate over a generated element multiple times and cannot preallocate memory. Current hexahedral mesh generating algorithms are in fact considered to be slower than tetrahedral meshers when leveling the playing field in terms of structural awareness. (Shepherd, 2007)

3.4.2 Hexahedral Mesh Generation

As shown in section 3.4.1 tetrahedral meshers revolve around locally changing the mesh in an iterative fashion. For hexahedral meshes this approach does not work as local changes propagate globally through the mesh. (Kremer et al., 2014) Consequently, hexahedral meshing is significantly more challenging than tetrahedral meshing.

Hexahedral meshing algorithms are categorized as a direct type if they generate a mesh from the geometry representation. They are called indirect if a tetrahedral mesh is used as an intermediate step. (Shepherd, 2008; Pietroni et al., 2022) display the capabilities as well as the state of current research for all prevalent hexahedral meshing techniques. The processing related traits of meshing algorithms are listed in tbl. 3.3 lists while traits of the resulting meshes are listed in tbl. 3.4.

Ansys mechanical, the leading FEM software in mechanical engineering (see chapter 2) implements the Sweeping and Hexdominant algorithms. Successful generation of meshes from complex parts with Sweeping requires a lot of effort (Harwick et al., 2005), as well as a deep insight into the inner workings of the mesher. Hexdominant on the other hand tends to deliver results of poor quality, has excessive meshing time or may terminate unsuccessfully. As is shown in tbl. 3.5 most research is accumulated in these two algorithms.

Even though there is much current research on Frame Field algorithms, they still

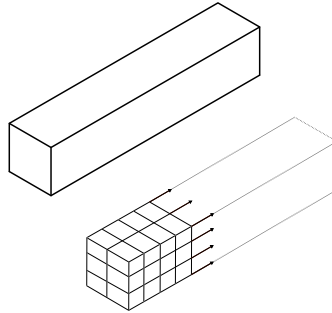


fig. 3.23: Hexahedral meshing with sweeping approach

offer poor robustness and are not fully automatic. Sweeping algorithms create elements of good quality. The element quality of Hexdominant algorithms is described to be "often good". (Pietroni et al., 2022) However this statement is slightly misleading as ill shaped elements often occur in areas with high stresses, thereby jeopardizing the simulation's quality as a whole as shown in section 3.3.1. Besides the required user input, a major downside of Sweeping algorithms in comparison to the Hexdominant method is their inability to offer local element size control (see. tbl. 3.4).

The main area of current research for Sweeping based meshing is to automatically decompose bodies into sweepable sections. e.g. (Sonthi' et al., 1997; Lu et al., 2001)). A section is sweepable if a 2D mesh can be constructed on a surface which is then dragged along a spline to form the part (see fig. 3.23). Obtaining these decompositions is the main part of the required user input for Sweeping methods. The state of the art in this field is described by Pietroni et al. (2022) in detail.

Mesh structure as defined in section 3.3.2 is also heavily influenced by the used meshing algorithm. Hexdominant algorithms are attributed with producing semi-structured meshes by Pietroni et al. (2022) which is for example backed up by current works by Reberol et al. (2021). However the structure in critical areas of parts tends to be worse due to sharp radii. Pietroni et al. (2022) also describe meshes generated with Sweeping methods to be semi-structured. In most cases Sweeping methods produce meshes of superior structure compared to Hexdominant (see section 3.3.2).

The stated disadvantages of Hexdominant and Sweeping methods have not

changed substantially compared to the amount of research that went into improving them. In this situation the improvement of a method with less current research coverage may be advantageous. For this dissertation grid methods were chosen as the basis of the further work as they can be fully automatic and excel in robustness which are important traits for the application in FEM for mechanical engineering as shown in chapter 2.

Further information on the other methods from tbl. 3.3 can be found in the literature reviews by Sarrate et al. (2014) and Pietroni et al. (2022).

tbl. 3.3: Application perspective on meshing methods (Pietroni et al., 2022)

Method	Type	User Interaction	Shape class	Robustness
Advancing front	Direct	Automatic	CAD oriented	Poor
Dual Methods	Both	Automatic, semi automatic	CAD oriented	Poor
Sweeping methods	Both	Semiautomatic	CAD oriented	Good (manual)
Grid based	Indirect	Automatic	Any shape	Great (commercial, product, demonstrated on many datasets)
Polycube maps	Indirect	Automatic, semiautomatic	Any shape	Good (demonstrated on medium datasets)
Frame fields	Indirect	Automatic, manual fixing	Any shape	Poor
Hexdominant	Both	Automatic	Any shape	Good (demonstrated on medium datasets)

tbl. 3.4: Qualities of meshes generated with different methods (Pietroni et al., 2022)

Method	Feature preserving	Size control	Mesh structure	Element quality	Orientation sensitive
Advancing front	Surface features only	No	Unstructured	Good at border, poorer inside	No
Dual Methods	Surface features only	No	Unstructured, semi-structured	Good at border, poorer inside	No
Sweeping methods	Surface features only	No	Unstructured, semi-structured	Good	No
Grid based	Yes, (limited valence)	Yes	Severely Unstructured	Poor at border, optimal inside	Yes
Polycube maps	Yes, (limited valence)	Yes	valence semi-structured	Good (depends on map)	Yes
Frame fields	Yes	Yes	valence semi-structured	Good (depends on map)	No
Hexdominant	Yes	Yes	valence semi-structured, hybrid	Often good	No

tbl. 3.5: Current research on meshing methods (Pietroni et al., 2022)

Method	Total works	Recent works	Open problems
Advancing front	7	0	Improve handling, colliding fronts, complex topologies
Dual Methods	13	2	Robust handling of self-intersecting sheets
Sweeping methods	35	11	Automatic definition of sweepable sub-volumes
Grid based	18	3	Feature preservation, mesh size. mapping
Polycube maps	18	8	Polycube topology, mapping, feature preservation
Frame fields	17	12	Generation of hexable fields, field aligned mapping
Hexdominant	20	11	Hybrid elements (topological control, amount, quality)

3.4.3 Mesh Optimization

Once generated, meshes can be subject to further optimization. Mesh optimization strives to improve quality metrics of the mesh. This can either be achieved by altering the structure of the mesh or by moving nodes.

Local remeshing can be used to improve the local quality of tetrahedral meshes.

It can either be used on surface meshes (Pellerin et al., 2011) or on volumetric meshes (Anderson et al., 2005; Zheng et al., 2005). In both cases elements of low quality and their immediate surrounding are replaced with a new mesh. Such approaches are only known for tetrahedral meshes as their structure can be changed and improved locally.

Moving nodes to improve the quality of a mesh is more versatile as it can be used on any mesh but more limited in its effect. The improvement of mesh quality by moving nodes can be achieved with force directed graph layout or with nonlinear optimization. For this method the optimization of the mesh is perceived as a non-linear equation system (3.33). In this context f computes the desired quality metric of the mesh based on the position \underline{x} of all nodes. When computing a nodal metric such as the SJ version suggested by Lobos (2015) \underline{Q} and \underline{x} have the same length. When an elementwise metric is used, sizes may vary. The quality metric can now be improved by using nonlinear optimization techniques. Such an approach is presented by Nealen et al. (2006).

$$\underline{Q} = \underline{f}(\underline{x}) \tag{3.33}$$

The main alternative to this approach are force directed graph layout techniques such as the one presented by Bhowmick and Shontz (2010). The main idea behind force directed graph layout is to model all edges of the mesh as springs pulling at the nodes. For each node the net force vector is computed and it is then moved in the direction of the net force by a distance proportional to the forces magnitude. This process has to be performed iteratively as the angles between nodes change in each iteration making the problem nonlinear.

3.4.4 Grid Based Meshing

Grid based meshing can be seen as a mesh first, geometry second approach. In contrast to Sweeping or Hexdominant methods, in which a mesh is created within a preexisting geometry, a mesh is created and then adapted to the target geometry. Such algorithms can be fully automated and have been found to produce high quality meshes with convergence characteristics similar to meshes generated with Sweeping methods for mechanical problems. Grid based methods are more

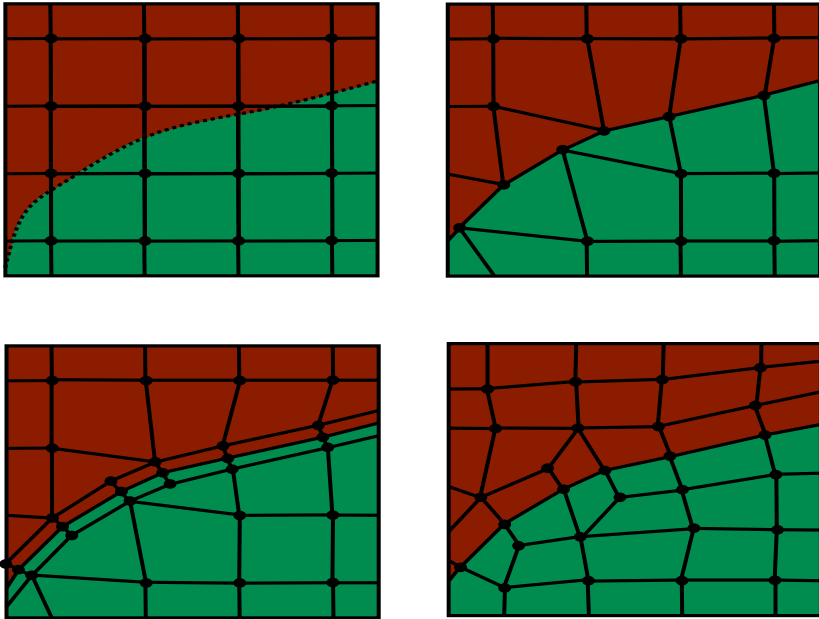


fig. 3.24: Sculpt algorithm (Owen and Shelton, 2015)

tolerant towards CAD model errors such as gaps and non watertight BREPs. By nature they produce purely hexahedral meshes. However, some elements may degenerate to unusable shapes with low Scaled Jacobians. In such cases they can be replaced with other element types such as tetrahedrons. (Owen and Shelton, 2015)

Owen and Shelton (2015) describe their sculpt algorithm which is a specific implementation of the grid based meshing concept. Sculpt creates a mesh in a pipeline with 5 stages (see fig. 3.24). First a grid mesh is created which encloses the whole geometry to be meshed. Then nodes close to the boundary of the geometry or the boundaries of zones with different materials are moved onto the boundary. In order to reduce distortion a padding layer of elements is introduced into the boundary zone. Lastly the mesh is smoothed to improve element quality. (Owen and Shelton, 2015)

The grid based method described above uses an equidistant polycube grid. This

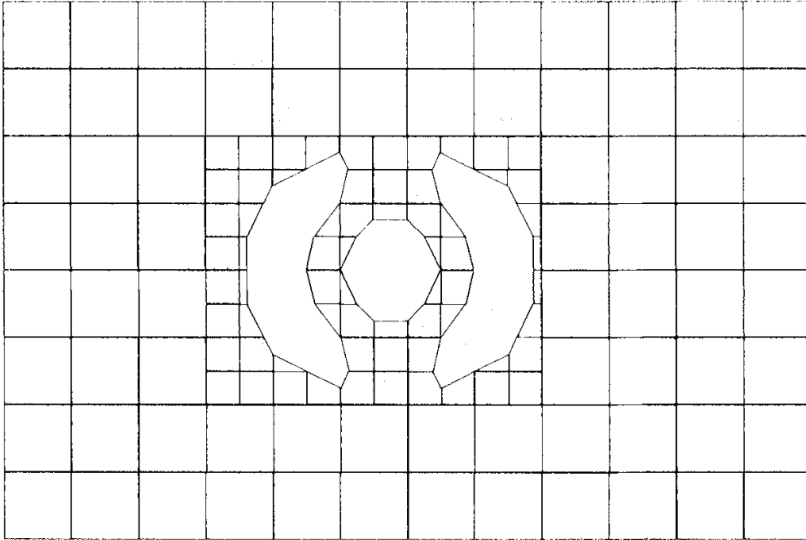


fig. 3.25: 2D octree of a geometry (Yerry and Shephard, 1984)

guarantees optimal mesh quality, but impedes local mesh refinement. Octree based methods mitigate this limitation by using an octree as their base grid. As the maximum depth of the tree may change locally, varying local mesh densities can be achieved. Fig. 3.25 shows a geometry represented by an octree. This structure cannot be used as a mesh for FEM as it is non conforming (see section 3.1.2). In order to obtain a usable mesh from an octree, rules to fill the cells of the tree with elements have to be formulated. In the 2D case $2^4 = 16$ different cases with their transformations exist. Mesh templates for all these cases have to be formulated to be combined into a mesh. (Yerry and Shephard, 1984; Yamakawa et al., 2011) Octree meshers may use different element types depending on their templates. They can be used in 2D or 3D applications. (Maréchal, 2009)

Practical implementations of grid based meshers such as snappyHexMesh by OpenCFD Ltd (2022) tend to have difficulties with exactly approximating sharp boundary geometry. (Pietroni et al., 2022) Enabling the representation of sharp features may require user input to assign edges to curves.

In order to improve the applicability of grid based meshers for mechanical FEM

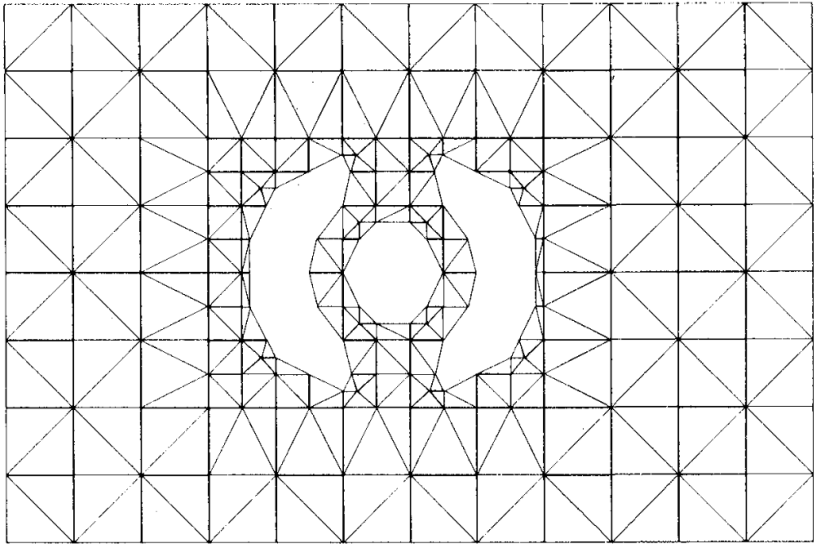


fig. 3.26: 2D mesh created from octree (Yerry and Shephard, 1984)

several issues have to be addressed (see tbl. 3.4 and tbl. 3.5):

- orientation sensitivity
- no automatic geometry mapping
- bad mesh structure
- no local mesh sizing

Orientation sensitivity of a mesh denotes whether the mesh will change when the represented part is rotated in the coordinate system. Such behaviour is not desirable as a suboptimal orientation of a part will result in a suboptimal mesh. This issue is even worse for parts with multiple rotation symmetric features such as gear teeth since they will all vary in mesh quality and consequently computed stresses. Geometry mapping refers to identifying which edges of the mesh map to which curves of the geometry. Local mesh sizing is only possible when the sizing is inherent to the underlying grid. This may be achieved with the formally introduced octree technique.

4 Combinatorial Meshing

Within this dissertation the method of Combinatorial Meshing is developed. It is meant to augment known methods of grid based meshing. This chapter gives an overview on the proposed algorithm and its motivation. The method is called Combinatorial Meshing as a novel combinatorial interpretation of the meshing problem is one of its cornerstones.

4.1 Consequences from Engineering Needs

In this section user needs and technical requirements for meshing in mechanical FEM are summarized. As shown in chapter 2 users expect more robust automatic meshing with usable mesh quality. Element quality can be measured sufficiently via the SJ metric and its extension to mixed elements by Lobos (2015). Meshes may not exhibit hanging nodes (see sec. 3.1.2). As most part geometries contain curved surfaces and also bend, when subjected to loads, quadratic basis functions are advantageous for most applications (see sec. 3.1.4). In the case of quadratic elements, hanging edges as well as hanging nodes cause singularities. A hanging edge is created when one side of a quadrangular face is occupied by two triangles (see fig. 4.1). Element edges with quadratic shape need a midside node to fully define it. This situation occurs when, for example, two tetrahedrons are put onto a hexahedron instead of a pyramid. In this case the basis functions become discontinuous between the elements as the midside node of the diagonalizing edge is now a hanging node. Such configurations are created by some mesh generators (Ansys Inc., 2022) but must be avoided.

Grid based meshing algorithms generally excel in satisfying these needs. (Owen and Shelton, 2015) Their main drawbacks for use in FEM meshing for mechanical engineering is that the resulting meshes are orientation sensitive and lack automatic detection of geometric edges. (Pietroni et al., 2022) Orientation sensitivity

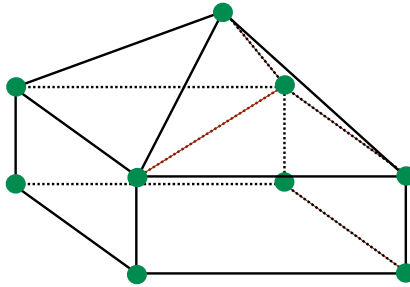


fig. 4.1: Hanging edge between a hexahedron and two tetrahedrons marked in red

means, that the quality of the mesh changes when moving the geometry in the global coordinate system. Furthermore they work well on box like parts but deliver inferior results on cylindrical objects such as shafts or gears. In consequence improving on grid based meshing techniques to resolve these issues is a worthy goal to pursue.

4.2 The Concept of Combinatorial Meshing

The common method of grid based meshing fills the bounding box of the target geometry with a grid mesh and then tries to form a valid mesh, representing the geometry, by carving it out of this grid mesh. From the perspective of a mechanical engineer this process is similar to milling the part from a block of material. Even though arbitrary parts can in theory be machined from a bounding box block, parts are usually made from bar stock with appropriate geometry. For example a shaft is typically manufactured from a cylinder and not from a block.

This concept is utilized to replace the basis grid of conventional grid based meshing with a problem adapted mesh. This mesh is called Precursor Mesh. As bar stocks are manufactured in a continuous rolling process they always have a sweepable geometry. Hence it can be meshed with ease. Figure 4.2 shows how a complex non-sweepable part is machined from an I-beam and meshed accordingly.

Since sweeping paths can be circular, any geometry which can be produced with a lathe can also be swept and thus be used as a Precursor Mesh. The Precursor

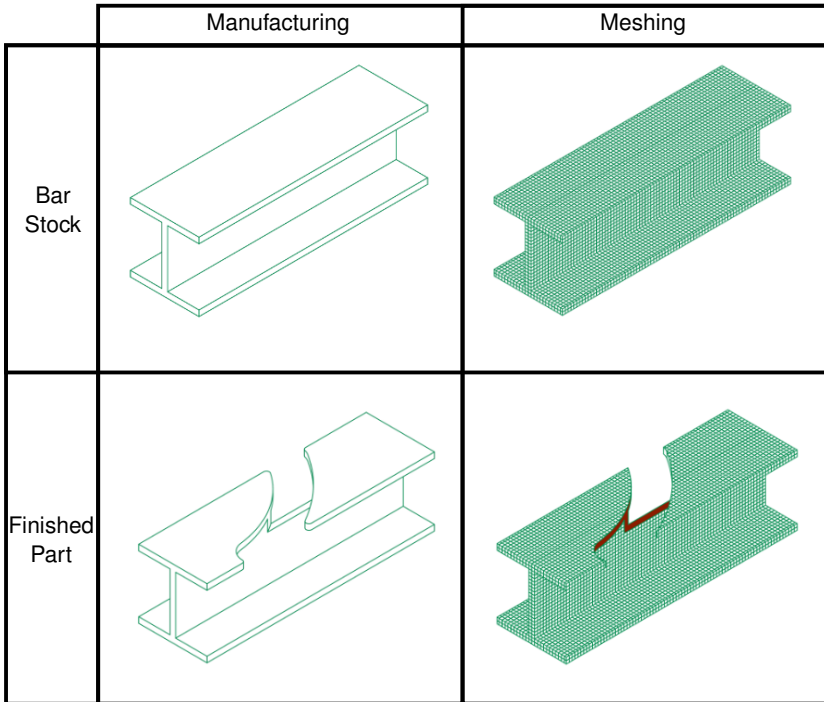


fig. 4.2: Manufacturing and meshing of an I-beam with a cut groove

Mesh may also be locally refined to allow local variations in mesh sizing.

Since it is good engineering practice to follow the manufacturing process in the setup of a CAD model, information on the bar stock geometry can be retrieved from the structure of the CAD model. Such data is not part of the current implementation of exchange file formats such as STEP specified by ISO 10303. (ISO, 2020)

As users advocate for including this information in novel exchange file formats for portability reasons (see chapter 2), it should be implemented to aid in meshing processes. For the presented results of this dissertation, augmented STEP files were created using a custom made CAD plugin. Such efforts should be standardized in future.

The most obvious way of cutting a mesh would be to either remove all elements which contain nodes outside of the target geometry or only remove an element if

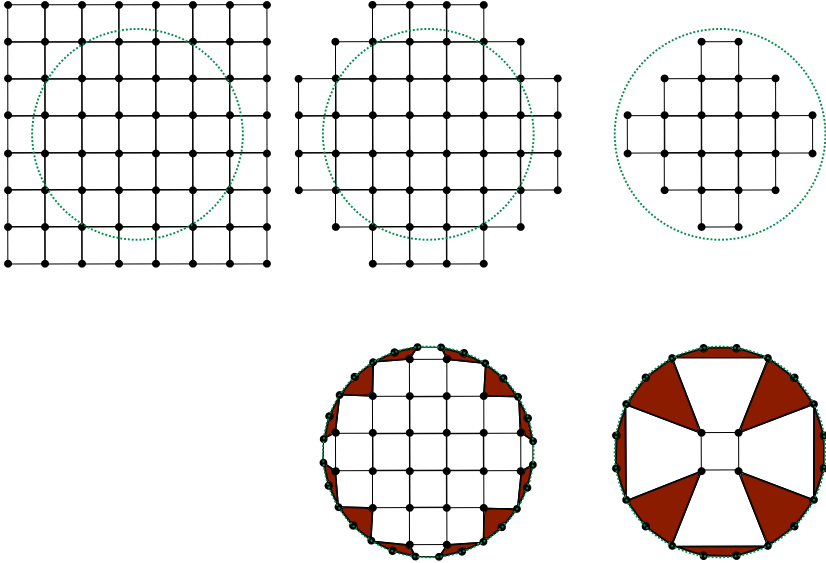


fig. 4.3: Results of trivial nodes removal for a circular geometry with unrecoverable elements are marked in red

all of its nodes lay outside of the target geometry. Neither of these is practical as both approaches cause a staircase effect when applied to curved geometry (see fig. 4.3). Such a mesh will have boundary elements with very low SJ which cannot be fixed by moving nodes, once all nodes at the perimeter are drawn onto the target geometry.

A practical method to enable node deletion is to generate a Precursor Mesh, which is more coarse than the target mesh, and insert small mesh templates into each cell of the Precursor Mesh depending on which nodes of the cell are to be kept. Within this dissertation these templates are called Super Elements. Practically speaking the Super Elements act as bricks shaped in different ways which are assembled to represent the target geometry. The problem of choosing the right Super Elements is a combinatorial problem.

The Super Elements only contain very few nodes and can be computed independently from the target geometry in advance. This opens up the possibility to sacrifice speed of generation for quality by using unorthodox methods for their compu-

tation. Such a method is logical programming (LP).

LP is chosen, as it allows to solve a problem with proven optimality. In general the runtime of the current solving technology is not guaranteed to be polynomial in respect to the number of variables to solve. To enable meshing to be solved with LP, the meshing problem has to be transformed into a suitable formulation. Such a formulation is a graph selection problem.

The graph representation of a mesh is discussed in chapter 4.3 while the solution to the problem of mesh generation based on this representation is shown in chapter 5. In chapter 6 two approaches for solving the Super Element selection problem are presented and discussed. In order to be able to fit the resulting mesh to a target geometry, its nodes and faces have to be assigned to geometric entities on which they are drawn. In this dissertation this process is called Entity Mapping. Two approaches for solving this problem are shown and compared in chapter 7.

4.3 Graph representation of a mesh

In order to make meshing a graph selection problem, an analogous graph for a mesh has to be defined. Furthermore the previously defined criteria for a well defined mesh have to be expressed in terms of such a graph.

An appropriate graph of a mesh for this purpose is a graph in which all elements of the mesh are graph nodes and all element faces are graph edges. Such a graph is shown in fig. 4.4. Triangular faces are marked in red and quadrangular faces are marked in green.

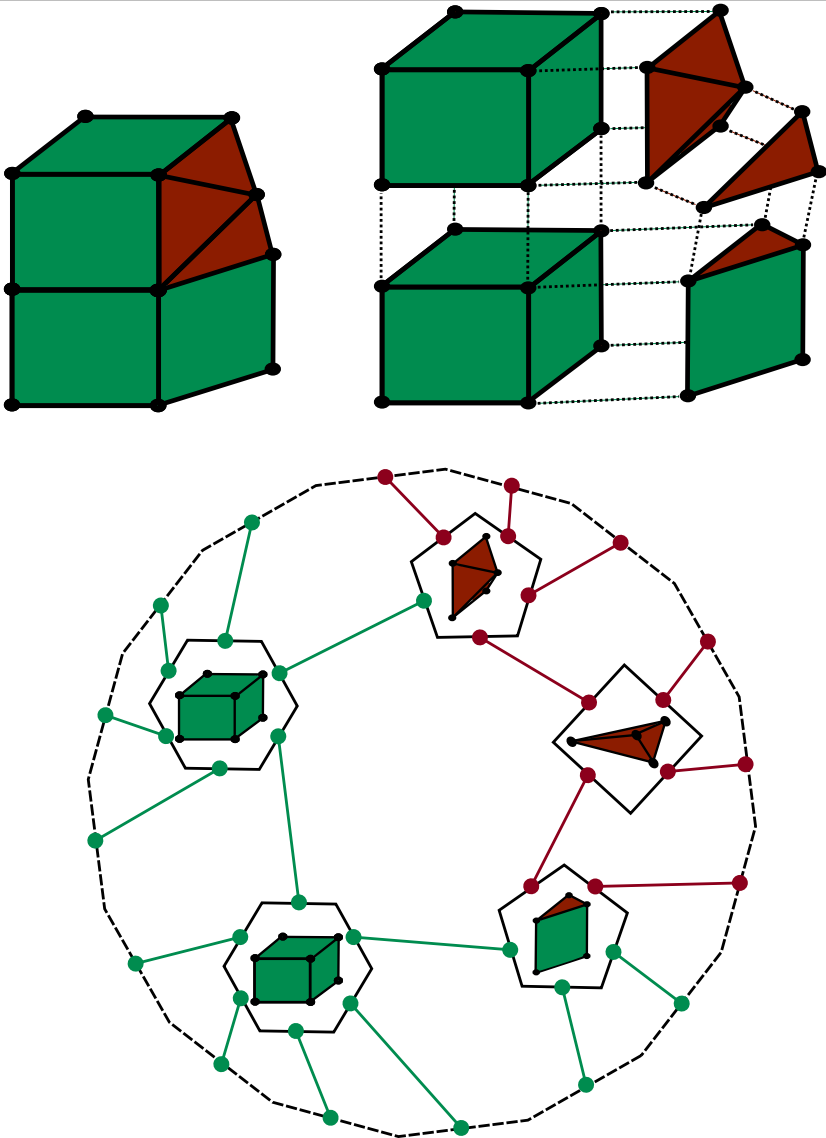


fig. 4.4: Graph representation of a mesh

Each element face must belong to one or two elements. If it only belongs to one element it is considered to be part of the mesh's surface. Furthermore no elements present in the mesh may intersect other elements. Scaled Jacobian of the elements

and node valence can be checked using its graph representation. The approximation quality of the mesh for the target geometry can be evaluated by comparing the surface mesh to the geometry BREP.

5 Generation of Super Elements

The method described in this chapter has been developed together with Valentin Mayer-Eichberger and is subject of a joint publication. (Stromberg et al., 2023) In this joint work Valentin Mayer-Eichberger has contributed the solver code.

5.1 Problem Definition

Super Elements are intended to be used as building blocks (see 4) to construct a mesh from. These blocks have to be designed in such a way that the assembled mesh does not have hanging nodes (see fig. 3.3). This is achieved by defining for each node of the Precursor Mesh if it is present in the final mesh. These nodes are called Outer Nodes. Furthermore, rules for touching faces of Super Elements have to be established in order to enforce their compatibility with each other.

All faces of Super Elements touching other Super Elements have to comply with the faces shown in 5.1 and 5.2. Outer Nodes which are defined as present are marked green and non present Outer Nodes are marked red. The black nodes are inserted into the mesh to allow for the removal of Outer Nodes while maintaining a valid mesh. Consequently the mesh on these faces is given by the presence configuration of the Outer Nodes of the Super Element.

Precursor Meshes discussed in this dissertation may consist of hexahedrons, tetrahedrons, prisms and pyramids. Thus Super Elements for these four element types have to be generated. For each element type each node may or may not be present within the mesh, so 2^n different Super Elements are required for an element type with n nodes. Some of these configurations can be derived trivially by transforming other configurations. Since this can also be done with the solution, the number of Super Elements for which solutions have to be found is reduced as shown in tbl. 5.1.

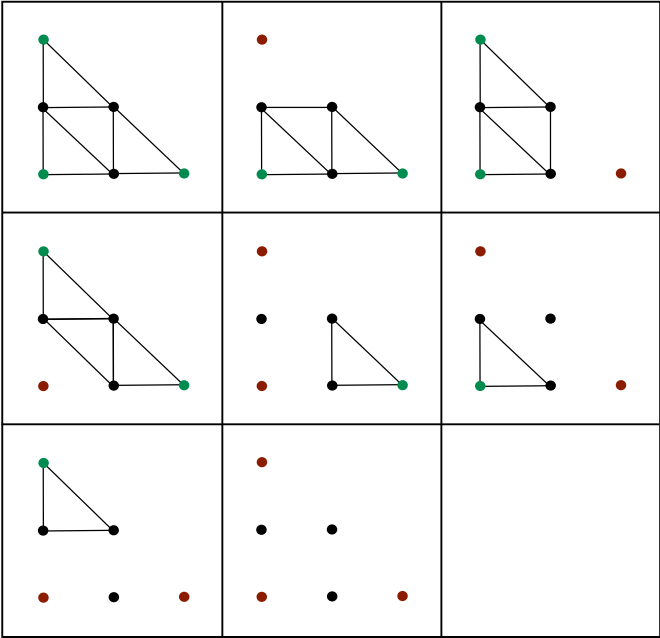


fig. 5.1: Possible reduced triangular faces on Super Elements

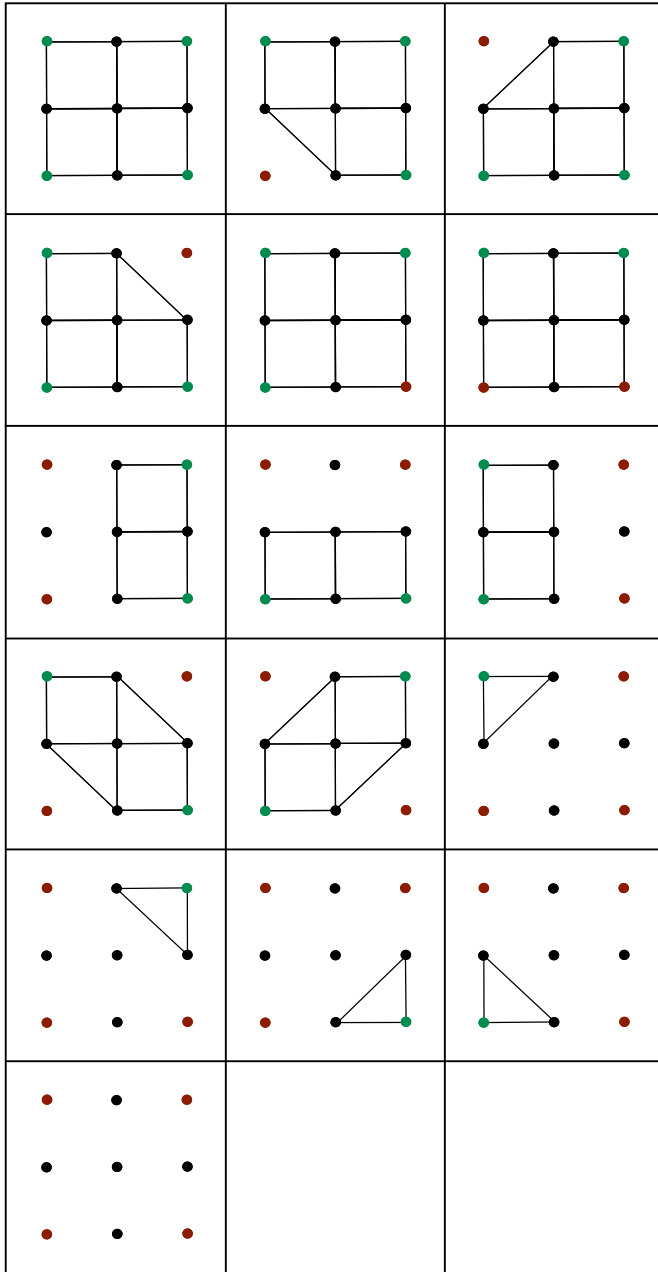


fig. 5.2: Possible reduced quadrangular faces on Super Elements

tbl. 5.1: Relevant cases for Super Element types

Element	Node Count	Super Element Count	Reduced Super Element Count
Hexahedron	8	256	23
Tetrahedron	4	16	5
Prism	6	64	13
Pyramid	5	32	12

For each of these cases a mesh has to be generated in order to provide a complete set of building blocks for arbitrary meshes. These Super Element meshes are independent of the mesh assembled from them. Therefore they can be computed in advance. Thus only a limited amount of meshes with low node count has to be computed once in order to generate the Super Elements. This enables the use of rather costly meshing approaches with superior quality. In this dissertation Super Elements are generated using ASP. This chapter focuses on hexahedral Super Elements. The described process however can be adapted for all other types of Super Elements. For the results presented in this dissertation Super Element meshes for tetrahedrons, prisms and pyramids were created manually.

In order to enable the application of ASP for generating meshes the problem of generating a mesh is reformulated into a graph selection problem as shown in section 4.3.

As arbitrary node locations have to be considered, the space from which graphs are chosen is a priori not finite. This problem can be made finite by selecting a set of Considered Node locations (see fig. 5.3, fig. 5.4, fig. 5.5 and fig. 5.6). The size of this set drives the computation time required for the ASP based meshing progress.

5.2 Considered Node Locations

The number of possible elements with k nodes each which can be constructed from n nodes, is generally $\frac{n!}{(n-k)!}$. For each set of nodes only one order has to be considered as only one order has a positive SJ, or in the case of tetrahedral elements, all orders are equal in quality. Thus the count of considered elements is

reduced to $\binom{n}{k}$.

As $\frac{n!}{(n-k)!}$ is a very steep function and computation time is proportional to it, only very few nodes can be considered for constructing meshes with this method. In figures 5.3, 5.4, 5.5 and 5.6 the Considered Node locations for all Precursor Mesh element types are shown. The Outer Nodes are marked in green. Black nodes are Considered Nodes on the surface and red circles are nodes within the Precursor Mesh element. The internal lines show the internal mesh of the Super Elements in the case of conserving all Outer Nodes.

The Considered Node locations are chosen to be compatible over all four element types of the Precursor Mesh. The most common Super Elements are the ones which conserve all Outer Nodes. Thus node positions for the Super Elements are advantageous which only contain elements of the same quality as the parent element from the Precursor Mesh. This criterion holds for the proposed positioning of internal nodes for all Super Element types but pyramids. A pyramid with all nodes retained is split into six pyramids and four tetrahedrons. This solution was chosen in order to avoid badly shaped pyramids inside the Super Elements. The internal nodes – visualized as red circles – were chosen to allow the placement of well shaped pyramids in the quadrangular outer faces. They are needed in order to allow transitions from quadrangular faces to triangular faces within a Super Element. In the chosen schema hexahedrons have 35 Considered Nodes, prisms 24, pyramids 14 and tetrahedrons 10. Tables with the coordinates of the Considered Nodes can be found in appendix B.

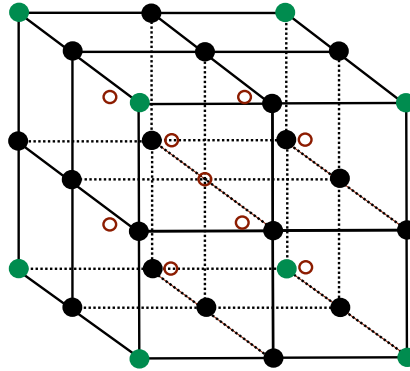


fig. 5.3: Considered Node Locations for hexahedral Super Elements

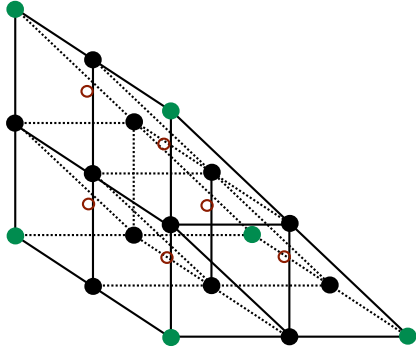


fig. 5.4: Considered Node Locations for prismatic Super Elements

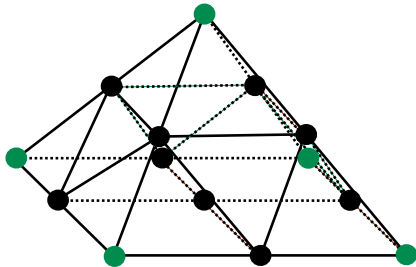


fig. 5.5: Considered Node Locations for pyramid Super Elements

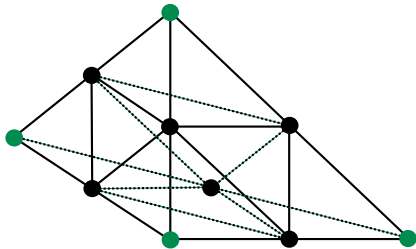


fig. 5.6: Considered Node Locations for tetrahedral Super Elements

The reminder of this chapter focuses on computing Super Element meshes for hexahedral Super Elements based on the 35 Considered Nodes from fig. 5.3. For these 35 nodes about 18 million elements with positive SJ exist (see tbl. 5.2).

In order to exclude elements with inferior quality and to allow the generation of meshes in feasible time only elements with SJ over the thresholds specified in tbl. 5.2 are used in the further process.

tbl. 5.2: Considered Sub Elements for Hexahedrons (Stromberg et al., 2023)

Element type	All elem. (SJ > 0)	Min. SJ	Count
Tetrahedrons	44850	0.16	22750
Hexahedrons	16333575	0.3	3809
Prisms	1351685	0.45	2751
Pyramids	264501	0.35	1626
Total	17994611	-	32061

All 18 million elements are generated and their SJ according to Lobos (2015) is evaluated in order to generate the data shown in tbl. 5.2.

5.3 Meshing ASP Model

Generating a mesh for a given geometry can be seen as selecting a set of elements from the elements generated in section 5.2. This selection must be valid in terms of the criteria established in section 4.3. The ASP model used to find these graphs is described in this section.

Fig. 5.7 shows the structure of the used ASP model. Each element is assigned the IDs of the faces it uses. These IDs are signed to account for the side of a face an element uses. Each face must either be used on both sides or is considered part of the Super Element's hull in order to guarantee the resulting mesh to be cohesive. For each possible element, its type and SJ are also part of the ASP model to be used as optimization goals.

Surface triangles are assigned to each face. For a triangular face one triangle is assigned. Quadrangular faces are broken in two triangles along their diagonal. As the quadrangle may be split along any of its two diagonals all four possible triangles are assigned to the face. The relation between faces and triangles is also signed to propagate the direction of normal vectors through the model.

The triangles are used to implement a rule for local convexity. This rule is enforced

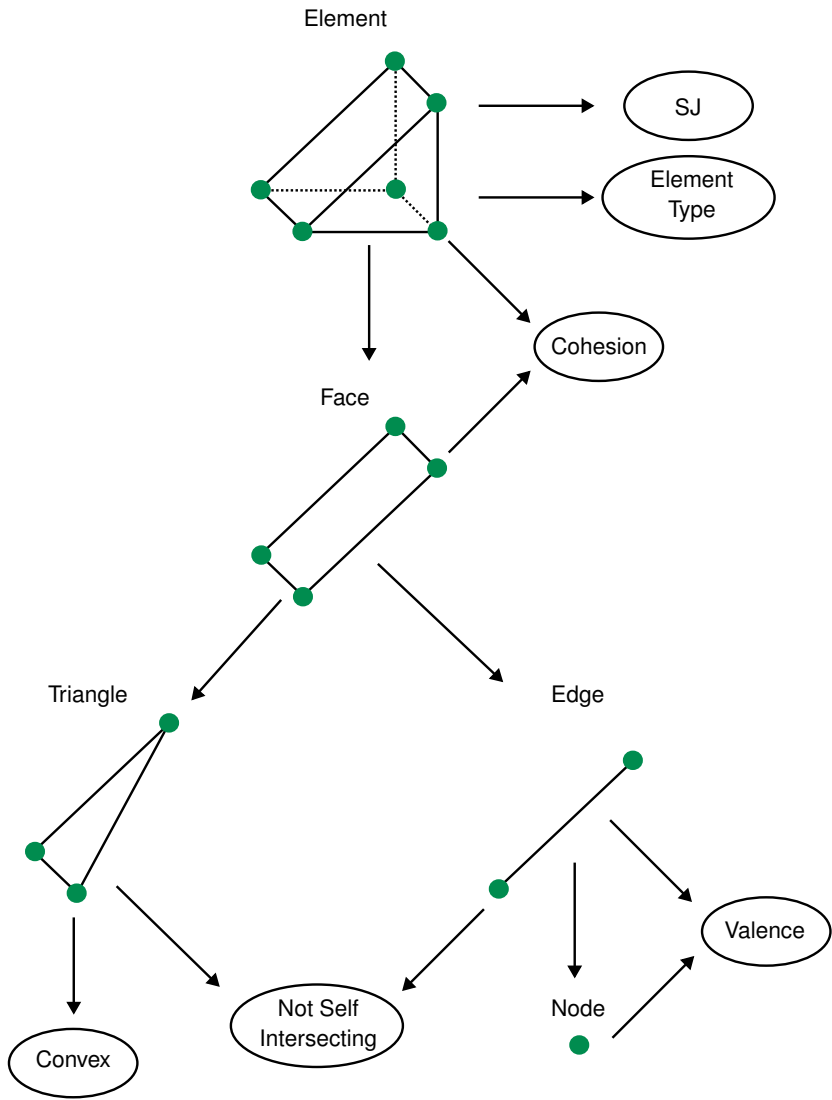


fig. 5.7: Used ASP model

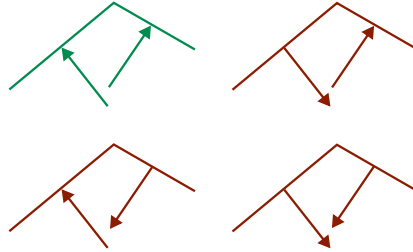


fig. 5.8: Legal and illegal combination of normal vectors

for all triangles stemming from faces which are only used by an element from one side and are as a result part of the surface of the Super Element. If two of those triangles share an edge, their normal vectors can be used to enforce local convexity (see fig. 5.8). This local convexity rule is useful as it prohibits impractical solutions such as meshes with internal voids.

The local convexity rule is found not to safely exclude all non convex solutions as it is possible to construct self-intersecting meshes which satisfy all rules stated so far. This issue is resolved by adding a rule which explicitly excludes self-intersecting solutions.

In order to do so, the model is expanded to store the IDs of all edges which are part of a face. Now a list of edges can be computed for each triangle. Members of this list may not be part of the solution once the triangle is as the edges intersect said triangle.

For all edges, the IDs of their nodes are stored in the model. This information can be used to compute node valence as an optimization target function.

With the SJ cutoffs from tbl. 5.2 a model with 767020 clauses is created. This model is created only once and can be reused for different hexahedral Super Elements and optimization targets.

The Super Element for which a mesh is computed is specified by the Outer Nodes which ought to be present in the mesh. These then dictate the mesh for all side faces of the Super Element as shown in fig. 5.1 and fig. 5.2. In the ASP model these surface meshes are enforced by selecting the faces used by the defined regions of the surface mesh and by prohibiting the use of any other faces there.

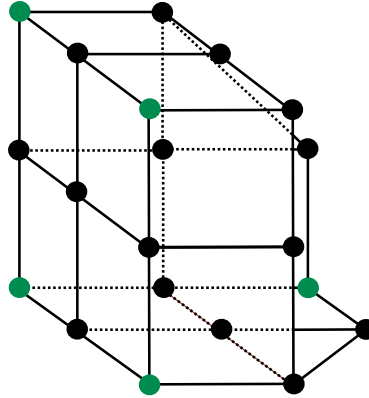


fig. 5.9: Example for a boundary mesh defining the task to be fulfilled by the ASP model

The resulting ASP program tasks the solver with finding a mesh filling a partially defined boundary with a convex mesh as is shown in fig. 5.9.

The described ASP model is implemented using the ASP solver clingo by Gebser et al. (2019). The model is provided in the following form:

```
info(element_id, element_type, sj).
% Assigns type and SJ converted to an integer to an
available elements.
...
filter(tet, min_sj).
filter(hex, min_sj).
filter(pyr, min_sj).
filter(prsm, min_sj).
% Specifies SJ thresholds for all element types.
element2face(element_id, (sgn_face_id; ...)).
% Lists all faces ids for an element with the normal
orientation encoded in the sign.
face2triangle(face_id, sgn_triangle_id).
% Defines a triangle to be part of a face. If the sign is
negative their normal vectors are opposite.
convex(sgn_triangle_id, sgn_triangle_id).
% Defines a the legal combination of signs for two touching
triangles.
face2edge(face_id, (edge_id; ...)).
% Lists all edges belonging to a face.
triangleRemoveEdge(triangle_id, (edge_id; ...)).
```



```

% Lists all edges which may not be used if the given
   triangle is used.
edge2node(edge_id, (node_id; node_id)).
% Assignes nodes to edges.
boundary_in(sgn_face_id; ...).
% Lists all faces which must be used for the boundary with
   their orientation encoded via their sign.
boundary_out(sgn_face_id; ...).
% Lists all faces which must not be used for the boundary
   with their orientation encoded via their sign.

```

In order to compute this input file, all possible elements (see tbl. 5.2) are computed and evaluated for SJ. Implemented in a mixture of Python and C++, this takes about 12 hours on a state of the art workstation.

This input file with the resulting 32061 elements is passed into clingo together with the solver code listed below. This solver code is contributed by Valentin Mayer-Eichberger. (Stromberg et al., 2023)

```

% Contributed by Valentin Mayer-Eichberger
% Only use elements that have Scaled Jacobian above Min from
   filter
element2faceF(E,F) :- element2face(E,F), info(E,T,SC),
    filter(T,Min), SC >= Min.

elementIn(E) :- element2faceF(E,_).
faceIn(F) :- element2faceF(_,F).
faceIn(F) :- boundary_in(F).
faceIn(F) :- boundary_out(F).
firstFace(F) :- F = #min { X : boundary_in(X) }.

{ element(E) } :- elementIn(E).
{ face(F) } :- faceIn(F).

% the faces part of the boundary of the shape need to be in
   the model
:- boundary_in(F), not face(F).

% the faces that cannot be part of the boundary of the shape
   need to be excluded
:- boundary_out(F), face(F).

% if F is chosen Face then exactly one element needs to be

```

```
    chosen of which contains that face
:- face(F), not 1 { element(E) : element2faceF(E,F) } 1.

% if an element is chosen, then all faces need to be in the model
:- element(E), element2faceF(E,F), not face(F).

% if a face contacts a void (= not face) then it is on the surface
face2triangle(-F,-T) :- face2triangle(F,T).
surfaceTriangle(T) :- face(F), not face(-F), face2triangle(F,T).

strict_concave(-T1,T2) :- convex(T1,T2).
strict_concave(T1,-T2) :- convex(T1,T2).
strict_concave(-T1,-T2) :- convex(T1,T2), not convex(-T1,-T2).

% concave combination of faces on the surface are forbidden
:- strict_concave(T1,T2), surfaceTriangle(T1),
   surfaceTriangle(T2).

% Infer edges
edge(E) :- face(F), face2edge(F,E).

% Infer inner triangles
triangle(|T|) :- face(F), face2triangle(F,T).

% each chosen triangle removes all edges it violates (intersection)
:- triangleRemoveEdge(T,E), triangle(T), edge(E).

% the first face of the boundary is the starting point of the connectivity analysis
reachedFace(F) :- firstFace(F).

% contact faces pass reachability forward
reachedFace(-F) :- reachedFace(F), face(-F).

% if a face is reached, then also the chosen element that it connects to
reachedElement(E) :- reachedFace(F), element2faceF(E,F),
   element(E).

% if an element is reached, then also all faces it has
```

```

    contact with are reached
reachedFace(F) :- reachedElement(E), element2faceF(E,F).

% all chosen elements need to be reached!
:- element(E), not reachedElement(E).

% N is a SJ in info
num(N) :- info(_, _, N).

% J is successor of I
next(I,J) :- num(I), num(J), I<J, not I<T : T<J, num(T),num(
    I),num(J).

% choose to remove all elements with SJ with value I
{ remove(I,J) } :- next(I,J).

% if we remove all SJ with J, then we also need to remove I
:- not remove(I,J), next(I,J), remove(J,K).

% if remove(I,_) is true, then remove all elements with SJ I
:- remove(SJ,_), info(E,_,SJ), element(E).

#maximize { 1@2,I : remove(I,_) }.

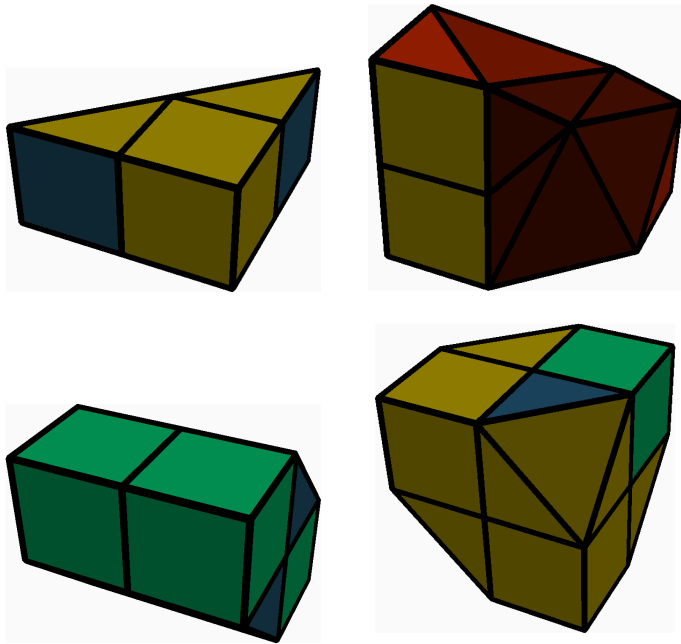
% minimize number of elements on second level
#minimize{ 1@1,E : element(E) }.

```

5.4 Results

Executing this model for all unique 23 hexahedral Super Elements on a state of the art workstation takes approximately 4 hours.

The described ASP model is capable of optimizing the resulting mesh for various qualities. It is possible to maximize SJ, to minimize node valence and to minimize element count. Combinations of these criteria may also be implemented. Tbl. 5.4 summarizes the results for these three criteria on all unique hexahedral Super Elements. The generated meshes can be found in appendix C. All these results are proven to be optimal by the solver. A range of results is shown in tbl. 5.3. Hexahedrons are printed green, tetrahedrons red, prisms blue and pyramids yellow.



tbl. 5.3: Multiple examples of generated Super Elements optimized for maximum SJ

tbl. 5.4: Optimization results (Stromberg et al., 2023)

Case	Min. Valence	Max SJ	Min Element Count
0	0	n.a.	0
1	3	0.5	1
2	4	0.577	2
3	6	0.408	6
4	8	0.5	10
5	5	1.0	4
6	5	0.385	4
7	11	0.302	20
8	10	0.302	24
9	7	0.5	12
10	9	0.5	15
11	10	0.302	24
12	8	0.348	13

Continued on the following page

Case	Min. Valence	Max. Min. SJ	Min Element Count
13	9	0.348	18
14	6	0.577	8
15	8	0.5	15
16	6	0.577	8
17	8	0.5	22
18	8	0.5	16
19	8	0.5	17
20	8	0.5	22
21	8	0.5	15
22	6	1.0	8

As any mesh constructed from Super Elements only contains elements from these Super Elements, the minimum element quality is guaranteed to be the minimum element quality from the used Super Element set.

It is also possible to generate Super Elements which can function as refinement adapters between two mesh regions with different node distance. An example boundary mesh for such an element is shown in fig. 5.10. In order to compute such meshes the list of possible faces (see fig. 5.2) would have to be extended for faces with some missing midside nodes.

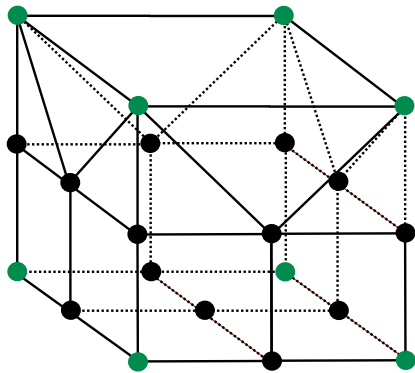


fig. 5.10: Boundary mesh for computing a refinement Super Element

6 Combinatorial Mesh Assembly

6.1 Problem Description

Super Elements can be combined to form arbitrary geometry. In order to obtain a legal mesh the faces of the combined Super Elements have to be compatible. The faces of Super Elements generated with the method shown in chapter 5 conform with the established rules for face meshes in fig. 5.1 and fig. 5.2. Hence Super Elements which are inserted into elements of the Precursor Mesh have conforming faces if they coincide in the existence of shared Outer Nodes in the final mesh. Therefore, selecting a valid combination of Super Elements can be achieved by declaring each node of the Precursor Mesh (which are the Outer Nodes of the Super Elements) to be included or excluded from the final mesh. Super Elements are then selected accordingly.

Meshes are thus formed based on a Precursor Mesh by coloring its nodes in red or black, in which a black node is kept and a red node is removed. Each coloring of the nodes of the Precursor Mesh is correlated with one resulting mesh. The quality of this coloring can be assessed by computing the geometric deviation between mesh and target geometry. The deviation can be expressed as a Residual Volume (RV), which is the negated intersection of both bodies. A good coloring should minimize this volume.

6.2 Residual Volume Integration

This volume can be computed by means of numerical integration. In this approach the elements of the Precursor Mesh are divided into integration subdomains. Now the center of each subdomain is checked that it is contained in the RV. RV is then computed as the sum of volume of all subdomains which are in the RV.

This straightforward process turns out to produce inferior results for cases in which the geometry cuts through the Precursor Mesh parallel to the element faces. As minor variations in the mesh or geometry will cause the preference for coloring a node red or black to flip, a zigzag pattern of kept and removed nodes is created (see fig. 6.1). To combat this issue RV which makes the resulting mesh too small is raised to the power of two. This target function leads to more smooth results. The integration process is summarized in fig. 6.2.

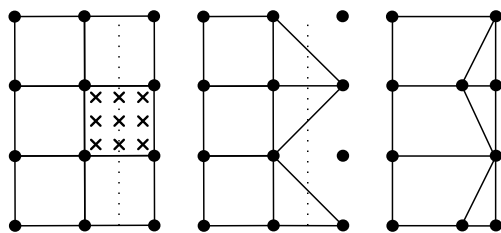


fig. 6.1: Bad mesh quality due to instability of trivial RV function

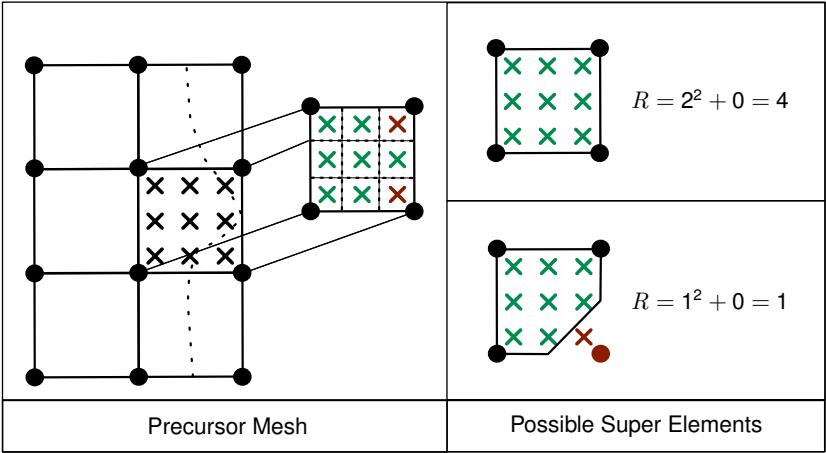


fig. 6.2: Target function for node coloring optimization

For the whole mesh the sum of RV has to be minimized. This process can be made easier regardless of the selected optimization method by computing whether integration subdomains are inside or outside of a Super Element with respect to

ξ -space in advance and independently from the geometry. The same computation can now be performed for the subdomains of the Precursor Mesh with respect to the target geometry in x -space. For the division of elements into subdomains the lowest number of divisions for each dimension was chosen which ended up creating a unique integration result for each Super Element. The resulting grids are shown in fig. 6.3, fig. 6.4, fig. 6.5 and fig. 6.6.

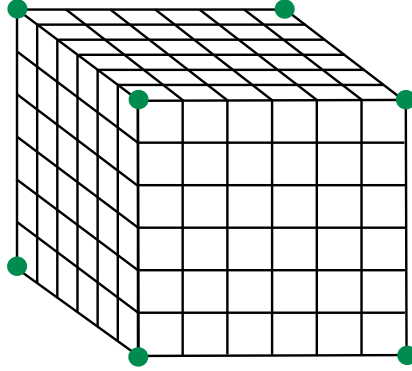


fig. 6.3: Chosen integration subdomains of a hexahedron

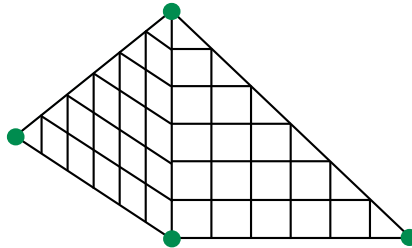


fig. 6.4: Chosen integration subdomains of a tetrahedron

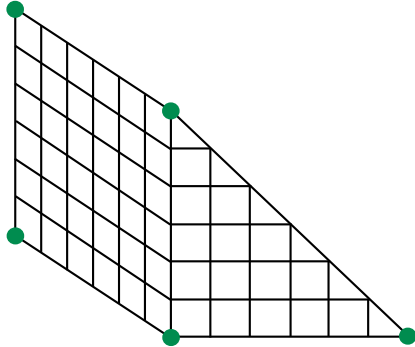


fig. 6.5: Chosen integration subdomains of a prism

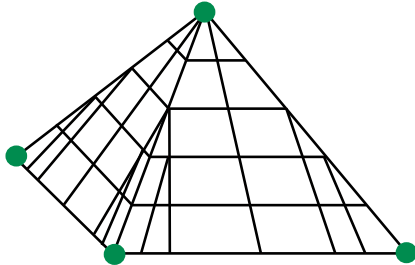


fig. 6.6: Chosen integration subdomains of a pyramid

6.3 ASP Approach

Finding the node coloring which minimizes RV is an optimization problem. In a first attempt an ASP encoding for this problem is created to obtain solutions with known optimality. Then the scalability of this approach is tested and a faster heuristic is developed.

The ASP model is outlined in fig. 6.7. For each node of the Precursor Mesh a color is chosen. Then for each Precursor Element a Super Element is assigned. The ASP model uses three types of clauses. The “decomposition” clause defines the RV caused by using a given Super Element for a given element of the Precursor Mesh. The “in” clause lists all nodes of the Precursor Mesh which are Outer Nodes of the noted Super Element when it is used for the given Precursor Element while

the “out” clause does the same for all nodes which are not Outer Nodes.

```

    decomposition(precursor_element_id, super_element_id, rv
    ).
    ...
    in(precursor_element_id, super_element_id, (node, node,
    ...)).
    ...
    out(precursor_element_id, super_element_id, (node, node,
    ...)).
    ...

```

This model is constructed geometry-specific and has to be solved in order to obtain a mesh for the geometry. For solving the model, a solver code for Clingo contributed by Mayer-Eichberger. (Stromberg et al., 2023)

```

% Contributed by Valentin Mayer - Eichberger
#show decompose/2.

% find min cost for each element
decomp_cost(E,C) :- decomposition(E,_,C).
has_less(E,C2) :- decomp_cost(E,C1), decomp_cost(E,C2),
    C1 < C2.
min_cost(E,C1) :- decomp_cost(E,C1), not has_less(E,C1).

element(E) :- decomposition(E,_,_).
node(N) :- out(_,_,N).
node(N) :- in(_,_,N).

1 { decompose(E, D) : decomposition(E,D,_) } 1 :-
    element(E).

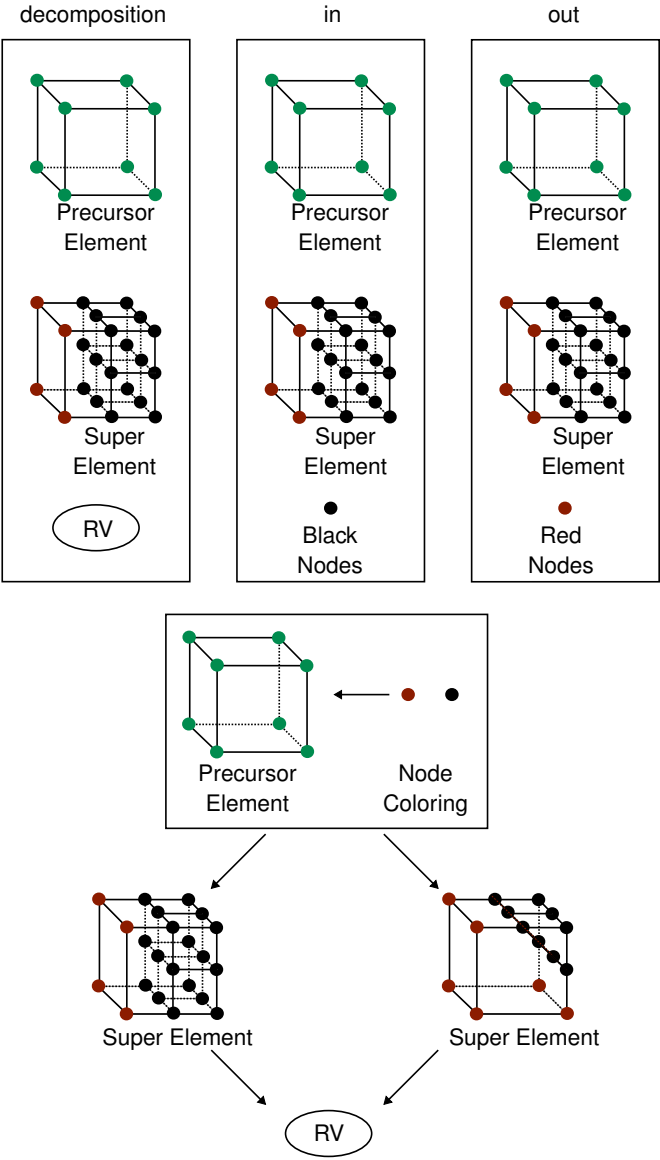
is_in(N) :- decompose(E,D), in(E,D,N).

:- decompose(E,D), out(E,D,N), is_in(N).

#minimize{ C-X@1,D,E : decompose(E,D), decomposition(E,D
    ,C), min_cost(E,X) }.
#minimize{ X@1,E: min_cost(E,X) }.

```

While computing the model input data has linear time complexity in terms of size of the Precursor Mesh, solving the model with clingo requires exponential time as expected and measured. The data shown in fig. 6.8 was obtained by meshing a cube



with a cylindrical hole based on a cube grid Precursor Mesh (see tbl. 6.1). The element count denoted on the x-axis refers to the Precursor Mesh. The performance may be improved upon by optimizing the model and clingo options. However, it is not likely to be feasible for the generating meshes of practical use. The computation time in relation to the element count of the Precursor Mesh is highly nonlinear. The next largest test case with 1000 elements did not finish in 24 hours. Since the node count of meshes for practical application is orders of magnitude greater, a heuristic algorithm is developed.

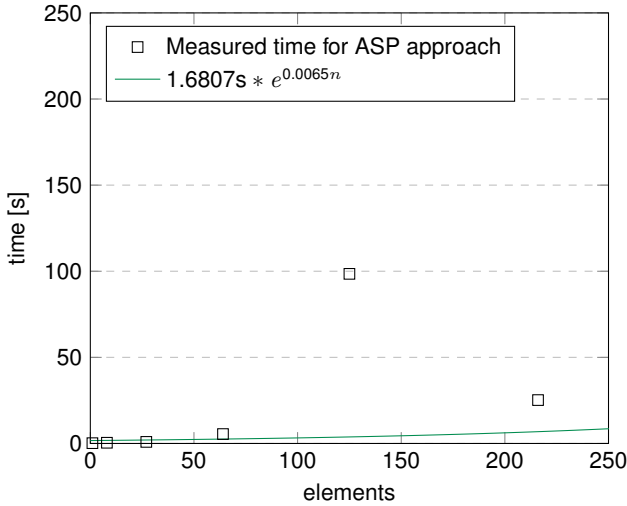


fig. 6.8: Computation time for node coloring with ASP

6.4 Heuristic Approach

The proposed heuristic is based on local greedy optimization. First the best possible Super Element for each Precursor Element is chosen. Then these Super Elements vote on the node coloring. Each Super Element votes +1 for its black nodes and -1 for its red nodes. Now all nodes with a positive score are colored black and all others red. Finally, the assigned Super Elements are chosen so that they satisfy this coloring.

The heuristic has linear time complexity as shown in equations (6.1), (6.2) and

(6.3). Experimental time complexity fulfills this prediction (see fig. 6.9). Linear time complexity is achieved by sacrificing the ability to accept locally disadvantageous Super Elements in order to profit from positive propagation effects.

$$n_{nodes} = \mathcal{O}(n_{elem}) \quad (6.1)$$

$$C_t(n_{elem}) = n_{elem} * k_1 + n_{nodes} * k_2 \quad (6.2)$$

$$C_t(n_{elem}) = \mathcal{O}(n_{elem}) \quad (6.3)$$

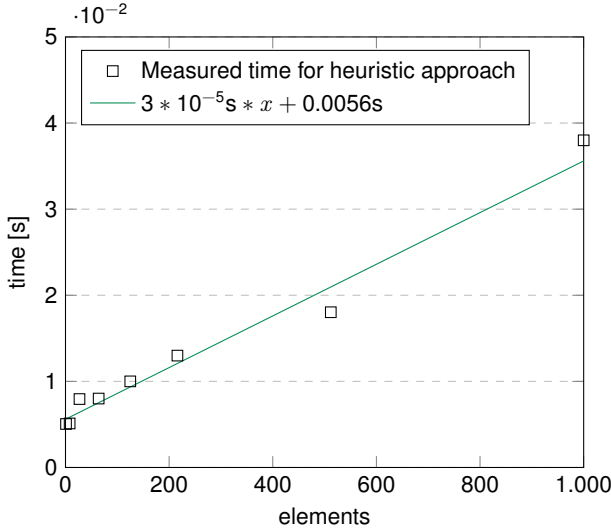


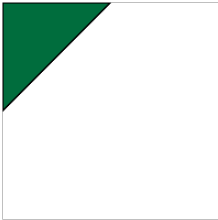
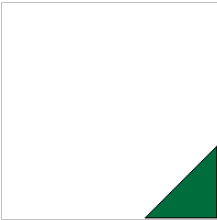
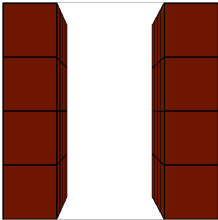
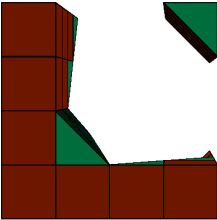
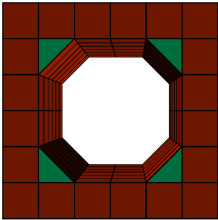
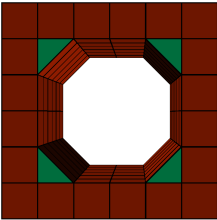
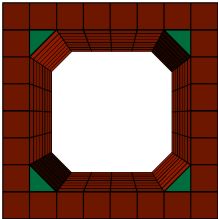
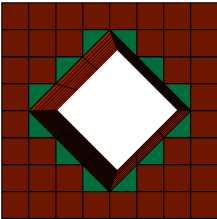
fig. 6.9: Computation time for node coloring with heuristic

The presented performance was achieved on a typical workstation running a Python implementation on a single core. Considering room for improvement of the implementation the required computation time for typical meshes with 10^6 elements can be neglected.

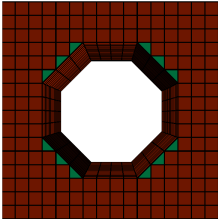
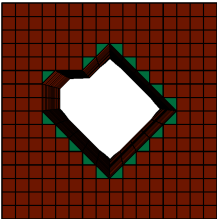
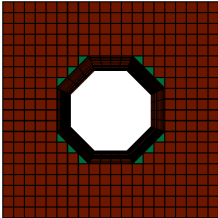
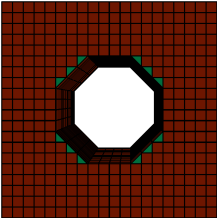
Results for both algorithms on an example geometry are shown in tbl. 6.1. The objective in this test case is to subtract a cylinder from a grid meshed cube. The element count of the Precursor Mesh cube is varied. Both algorithms struggle on very coarse meshes. The results for 1 element and 8 elements from both are not fit for practical use.

The other cases show high similarities of the results of both algorithms. However the solutions selected by the heuristic approach for 64 and 512 elements are inferior to the results of the ASP approach. Such issues can be resolved by slightly adjusting the number of elements in the Precursor Mesh. When implementing such a measure, the heuristic approach is deemed to be practically viable.

tbl. 6.1: Comparison of results for voting heuristic

Precursor count	Element	ASP result	Heuristic Result
1			
8			
27			
64			

Continued on the following page

Precursor count	Element	ASP result	Heuristic Result
512			
			

7 Entity Mapping

Established mesh generation techniques such as sweeping or hex-dominant recombination of tetrahedrons start with a surface mesh which is developed into a volume mesh. Since this volume mesh is generated from the geometric entities of the target BREP, (see fig. 3.10) the assignment of mesh entities to geometric entities is known. This relation (see tbl. 7.1) is important to allow for the adaptive refinement of the mesh, the elevation of the element order and the assignment of boundary conditions. Typical implementations of grid based meshers rely on user interaction to map mesh entities to geometric entities. Since users prefer a higher degree of automation (see chapter 2) a better method for the Entity Mapping process is required.

tbl. 7.1: Required geometry mapping

Geometric entity	Mesh entity
corner	node
curve	edge
surface	face

The proposed algorithm is a top down approach which first maps mesh faces to geometric surfaces and then deduces a mapping for curves and corners as outlined in fig. 7.2. For each element face of the surface mesh, a ray is cast from its center. For this ray the nearest intersection with a geometric surface is searched. The element face is now mapped to this surface. Each surface element face has to be mapped to exactly one geometric surface.

Computing the intersection of a ray with a NURBS surface is a nonlinear root search problem. Since the known methods for finding roots such as the Newton–Raphson method cannot guarantee finding the closest root to the chosen starting point (see fig. 7.1) it is not possible to perform the ray casting with NURBS BREP geometry. Instead a surface triangulation is computed using Delaunay’s method. (Delaunay, 1934)

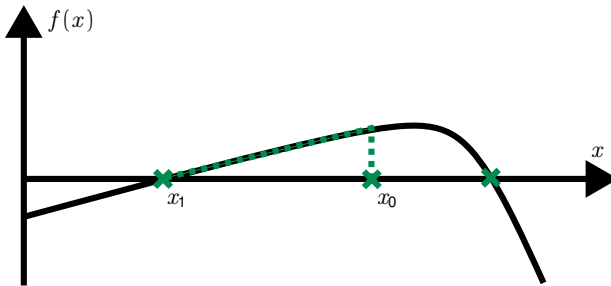


fig. 7.1: Convergence of a Newton iteration to a far away root

The intersection of a ray with a polygon is a linear problem and can be solved exactly. The triangulation is computed with a target edge length of half the minimum edge length of the mesh which needs to be mapped. By doing so, small surfaces are automatically defeatured. The term defeaturing describes the removal of geometric features which are small and do not impact simulation results but hinder meshing. Since the underlying surfaces for each simplex of the triangulation are known from the triangulation process, the mapping of faces to surfaces is now completed.

Finding the intersection of a ray with n triangles can be achieved in logarithmic time for an average case when implemented with an k-d tree or octree (see section 3.2.2). The worst case complexity however is linear in n . (Szirmay-Kalos and Márton, 1998) As this has to be performed for each face and the number of faces is linearly proportional to the number of triangles (see above), the total time complexity of computing the mapping is $\mathcal{O}(n \cdot \log n)$.

$$C_{t,oct}(n) = \mathcal{O}(\log n) \quad (7.1)$$

$$n_{faces} \propto n_{trgs} \quad (7.2)$$

$$C_{t,map}(n) = \mathcal{O}(n \cdot \log n) \quad (7.3)$$

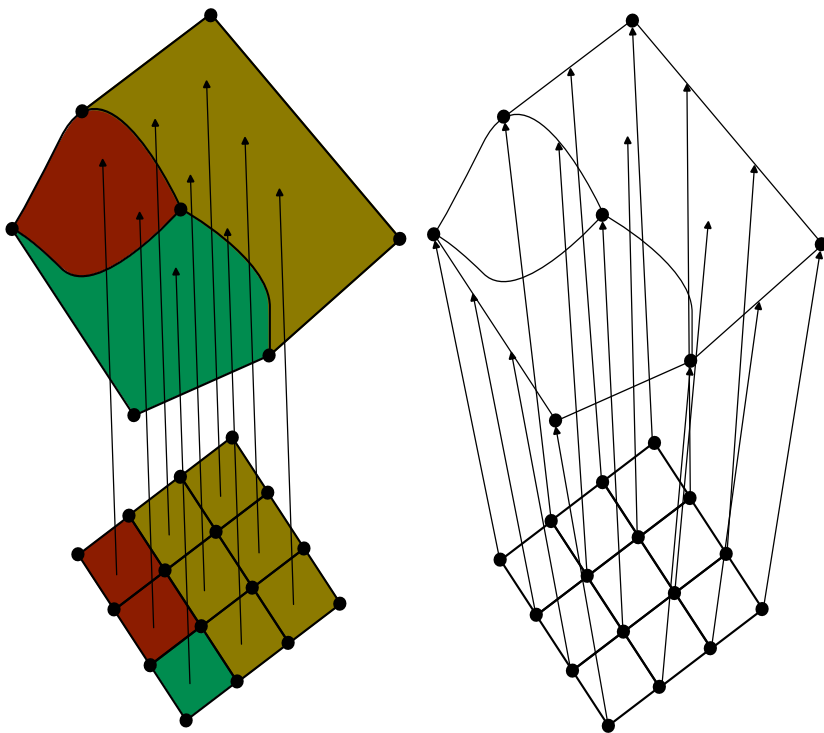


fig. 7.2: Entity Mapping process

Each curve has one or two adjacent surfaces. Since the mapping must also represent this neighbourhood constraints it can be exploited for deducing the mapping of curves to edges from the mapping of faces to surfaces. An edge is mapped to a curve if the adjacent faces of the edge are mapped to the surfaces adjacent to the

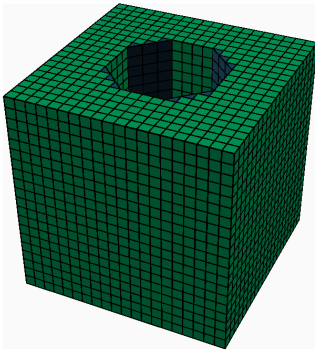
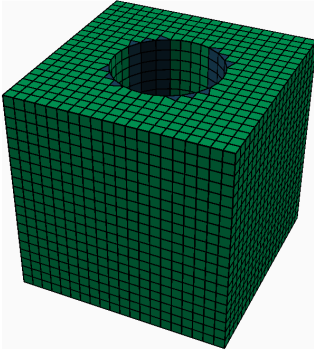
curve. Edges may end up not mapped to a curve. If no continuous path of edges is mapped to a curve it is defeatured. All mappings to defeatured curves are dropped.

Nodes are mapped to corners if all edges connected to the node within the surface mesh are mapped to curves connected to the corner.

Once this mapping is computed it can also be used for mesh optimization. For doing so, each node of the surface mesh is assigned to an entity of the geometry (see fig. 7.2). In this process nodes assigned to corners keep this assignment. Nodes are assigned to curves if two of the surface mesh edges connected to the node are assigned to the curve. A node is assigned to a surface if all surface mesh faces adjacent to the node are mapped to this geometric surface. Now mesh quality is improved by moving surface nodes on their assigned geometric entities. Nodes within the volume may be moved freely. This method may be used to improve Laplacian smoothing as shown by Owen and Shelton (2015) while keeping nodes on the target geometry and preserving sharp edges. It can also be used in conjunction with force directed layout techniques.

In tbl. 7.2 the fitting of several meshes to their respective geometry is shown. Hexahedrons are printed in green, prisms in blue, pyramids in yellow and tetrahedrons in red.

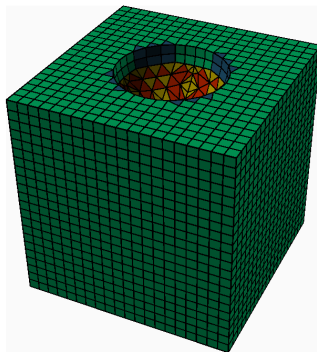
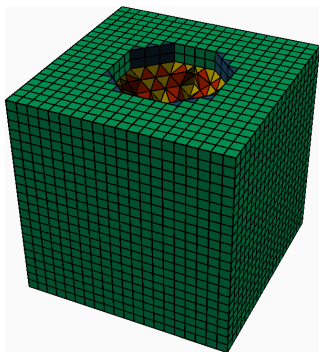
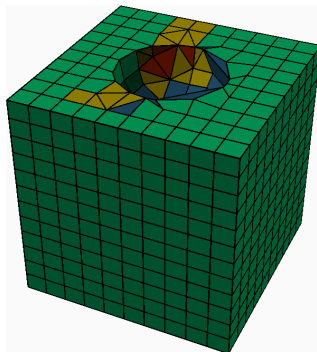
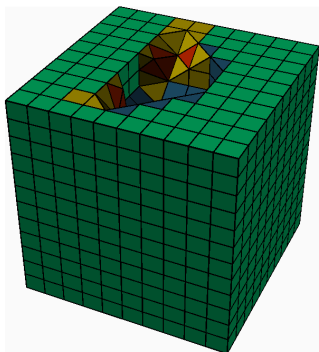
tbl. 7.2: Results of Entity Mapping process

Raw mesh	Mapped mesh
	

Continued on the following page

Raw mesh

Mapped mesh



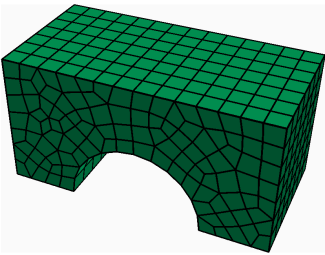
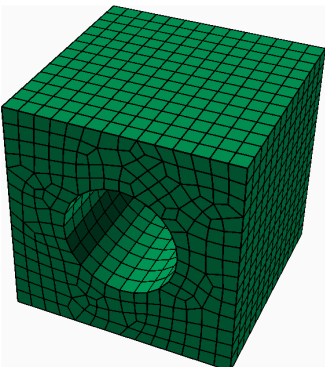
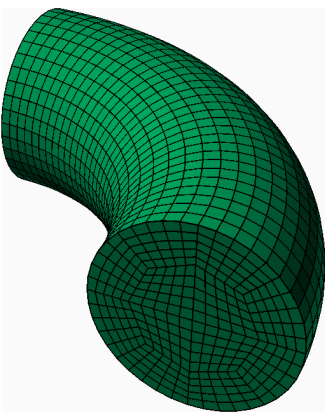
8 Demonstration Results

In table 8.1 results of Combinatorial Meshing for some geometries from the MAMBO data set (Ledoux, 2022) are presented. It is a collection of geometries for evaluating the performance of meshing algorithms. For all geometries, type wise element counts and Scaled Jacobian are given. For these examples the Entity Mapping process was not executed in order to preserve geometry defects for inspection and due to excessive time consumption of the current implementation of the Entity Mapping (see chapter 7).

In some cases (for example B0) no cutting of the Precursor Mesh is required. In cases where the Precursor Mesh has rotational symmetry and cannot be swepted in axial the direction (see B42), the results are superior to the industry standard tools for automated meshing such as Ansys. (Ansys Inc., 2022) This advantage is achieved by exploiting information provided through the CAD tree of the geometry. Furthermore the structure information can be utilized to use 2D mesh templates for recognized sketch geometries as shown in B11.

Successful cutting operations by using Super Elements are, for example, shown with B6, B19 or B21. For test geometry B2, a suboptimal mesh is generated. The algorithm fails to generate an adequate mesh for the sharp transition from the box to the cylinder. The cause for this behaviour of the algorithm is that the used Super Elements are enforced to be convex (see section 5.3). Such issues may be circumvented by using a wider spectrum of Super Elements.

tbl. 8.1: Results of Combinatorial Meshing on geometries from the MAMBO dataset

Meta Data					Generated Mesh	
Geometry: MAMBO B0						
	Hex.	Tet.	Prsm.	Pyr.		
Count	756	0	0	0		
Min. SJ	0.24	0.0	0.0	0.0		
Mean SJ	0.51	0.0	0.0	0.0		
Max. SJ	0.88	0.0	0.0	0.0		
Geometry: MAMBO B10						
	Hex.	Tet.	Prsm.	Pyr.		
Count	2665	0	0	0		
Min. SJ	0.24	0.0	0.0	0.0		
Mean SJ	0.54	0.0	0.0	0.0		
Max. SJ	0.82	0.0	0.0	0.0		
Geometry: MAMBO B11						
	Hex.	Tet.	Prsm.	Pyr.		
Count	9600	0	0	0		
Min. SJ	0.21	0.0	0.0	0.0		
Mean SJ	0.41	0.0	0.0	0.0		
Max. SJ	0.63	0.0	0.0	0.0		

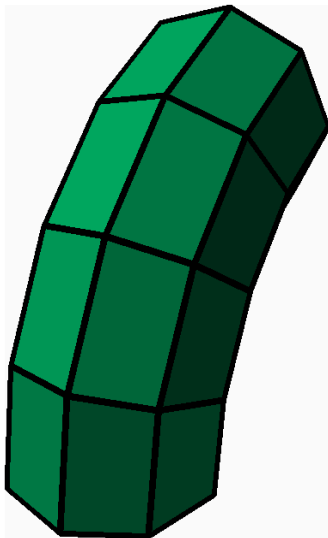
Continued on the following page

Meta Data

Generated Mesh

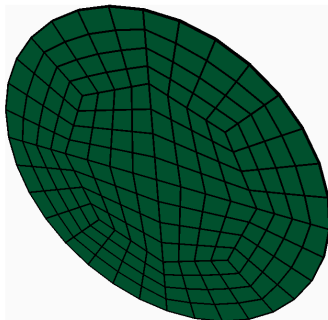
Geometry: MAMBO B12

	Hex.	Tet.	Prsm.	Pyr.
Count	48	0	0	0
Min. SJ	0.09	0.0	0.0	0.0
Mean SJ	0.15	0.0	0.0	0.0
Max. SJ	0.26	0.0	0.0	0.0

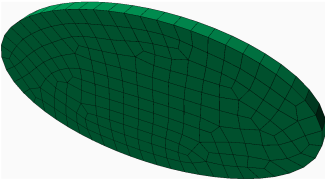
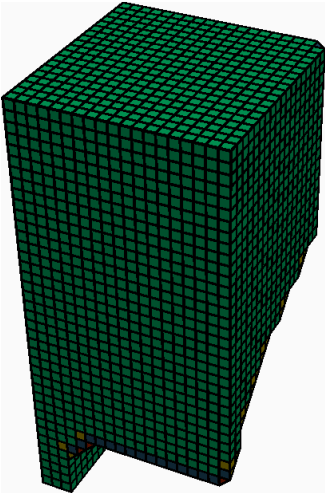
A 3D visualization of a green hexahedral mesh for the MAMBO B12 geometry. The mesh is composed of several large hexahedra arranged in a curved, elongated structure. The edges of the hexahedra are highlighted with black lines, and the overall shape is a smooth, curved surface.

Geometry: MAMBO B14

	Hex.	Tet.	Prsm.	Pyr.
Count	192	0	0	0
Min. SJ	0.03	0.0	0.0	0.0
Mean SJ	0.04	0.0	0.0	0.0
Max. SJ	0.06	0.0	0.0	0.0

A 3D visualization of a green hexahedral mesh for the MAMBO B14 geometry. The mesh is composed of many small hexahedra arranged in a curved, elongated structure. The edges of the hexahedra are highlighted with black lines, and the overall shape is a smooth, curved surface.

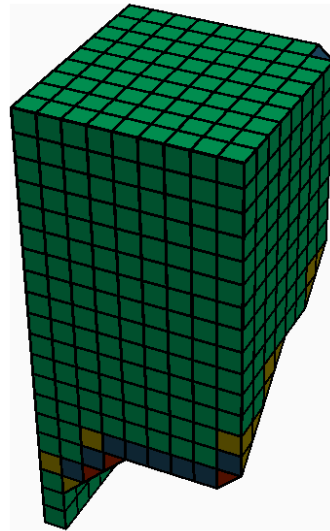
Continued on the following page

Meta Data					Generated Mesh
Geometry: MAMBO B15					
	Hex.	Tet.	Prsm.	Pyr.	
Count	217	0	0	0	
Min. SJ	0.26	0.0	0.0	0.0	
Mean SJ	0.61	0.0	0.0	0.0	
Max. SJ	0.87	0.0	0.0	0.0	
Geometry: MAMBO B17					
	Hex.	Tet.	Prsm.	Pyr.	
Count	9330	140	264	204	
Min. SJ	0.35	0.35	0.35	0.35	
Mean SJ	0.99	0.49	0.73	0.65	
Max. SJ	1.0	0.55	1.0	1.0	

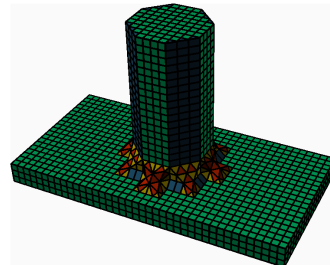
Continued on the following page

Meta Data**Generated Mesh**

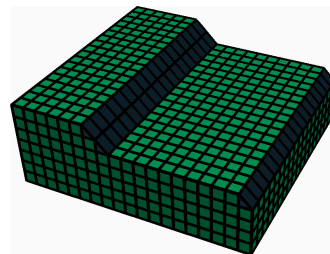
Geometry: MAMBO B19				
	Hex.	Tet.	Prsm.	Pyr.
Count	950	88	76	120
Min. SJ	0.35	0.35	0.35	0.35
Mean SJ	0.95	0.49	0.49	0.62
Max. SJ	1.0	0.55	0.55	1.0



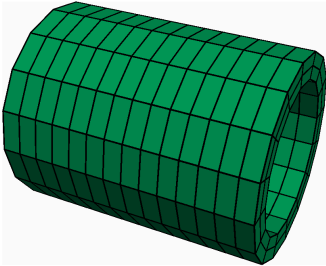
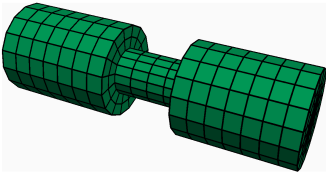
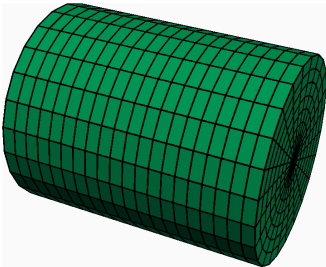
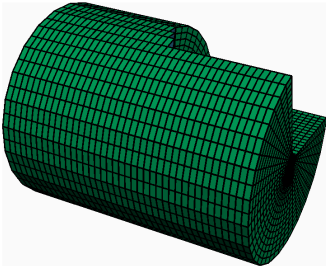
Geometry: MAMBO B2				
	Hex.	Tet.	Prsm.	Pyr.
Count	3996	176	280	300
Min. SJ	0.35	0.35	0.35	0.35
Mean SJ	0.98	0.47	0.67	0.69
Max. SJ	1.0	0.55	1.0	1.0



Geometry: MAMBO B21				
	Hex.	Tet.	Prsm.	Pyr.
Count	2010	7	76	6
Min. SJ	0.5	0.5	0.5	0.5
Mean SJ	0.98	0.5	0.94	0.5
Max. SJ	0.98	0.5	0.98	0.5



Continued on the following page

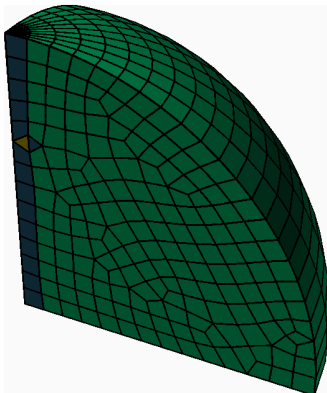
Meta Data					Generated Mesh
Geometry: MAMBO B4					
	Hex.	Tet.	Prsm.	Pyr.	
Count	576	0	32	0	
Min.	0.1	0.0	0.1	0.0	
SJ					
Mean	0.18	0.0	0.18	0.0	
SJ					
Max.	0.47	0.0	0.47	0.0	
SJ					
Geometry: MAMBO B42					
	Hex.	Tet.	Prsm.	Pyr.	
Count	1456	0	0	0	
Min.	-0.0	0.0	0.0	0.0	
SJ					
Mean	0.17	0.0	0.0	0.0	
SJ					
Max.	0.53	0.0	0.0	0.0	
SJ					
Geometry: MAMBO B5					
	Hex.	Tet.	Prsm.	Pyr.	
Count	3360	0	480	0	
Min.	0.23	0.0	0.23	0.0	
SJ					
Mean	0.43	0.0	0.44	0.0	
SJ					
Max.	0.71	0.0	0.71	0.0	
SJ					
Geometry: MAMBO B6					
	Hex.	Tet.	Prsm.	Pyr.	
Count	22848	24	3511	14	
Min.	0.08	0.08	0.08	0.08	
SJ					
Mean	0.48	0.17	0.48	0.17	
SJ					
Max.	0.83	0.23	0.83	0.22	
SJ					

Continued on the following page

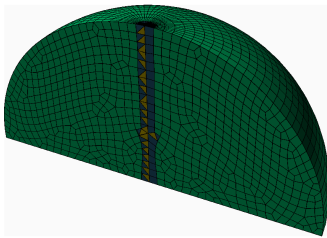
Meta Data

Geometry: MAMBO B7				
	Hex.	Tet.	Prsm.	Pyr.
Count	1710	0	149	23
Min. SJ	0.05	0.0	0.05	0.09
Mean SJ	0.35	0.0	0.36	0.33
Max. SJ	0.71	0.0	0.71	0.71

Generated Mesh

A 3D perspective view of a curved, wedge-shaped mesh. The mesh is composed of green hexahedral elements. A small, dark blue triangular feature is visible on the left vertical face of the mesh.

Geometry: MAMBO B9				
	Hex.	Tet.	Prsm.	Pyr.
Count	11264	0	671	51
Min. SJ	0.05	0.0	0.05	0.05
Mean SJ	0.38	0.0	0.39	0.34
Max. SJ	0.84	0.0	0.84	0.81

A 3D perspective view of a curved, semi-circular mesh. The mesh is composed of green hexahedral elements. A vertical line of dark blue elements runs along the center of the curved surface.

9 Summary

Within this dissertation the concept of generating meshes by solving a graph selection problem with an ASP solver is introduced. This method allows for the generation of small meshes with proven optimality. The underlying modeling enables researchers to solve meshing problems with a wide range of methods from other fields of computer science which are currently not in use for meshing.

The presented approach for utilizing this method for meshing larger geometries is not the only combinatorial algorithm worth considering. Many other combinations with other grid based meshing techniques are possible and may be investigated in the future. Furthermore the graph representation of the meshing problem may aid the better theoretical understanding of the complexity of the problem.

The discussed approach of combinatorial meshing brings several advancements to the field of grid based meshing:

1. It enables the generation of meshes with guaranteed minimum element quality by using Super Elements.
2. It generates meshes independent from the orientation of the geometry and is better suited for meshing shaft like geometries by using Precursor Meshes.
3. It is capable of automatically capturing sharp and smooth features.
4. It allows for local refinements with refined Precursor Meshes and in the future also with refinement Super Elements.

The parts of the algorithm which are executed when meshing a geometry are the generation of the Precursor Mesh, the computation of the Residual Volume for each element of the Precursor Mesh, Super Element Assignment and Entity Mapping. In tbl. 9.1 the time complexity with respect to the number of elements of these stages are summarized with reference to the argument for these complexities.

tbl. 9.1: Time complexity of all components with respect to element count of the algorithm

Step	Time complexity	Reference	Page
Generate Precursor Mesh	$\mathcal{O}(n)$	figure 3.13	29
Compute Residual Volume	$\mathcal{O}(n \log n)$	section 6.4	83
Super Element Assignment	$\mathcal{O}(n)$	section 6.2	77
Entity Mapping	$\mathcal{O}(n \log n)$	chapter 7	87

For the complete process, $\mathcal{O}(n \log n)$ is dominant. Compared to established unstructured methods which have linear time complexity (see fig. 3.13), this is slightly slower but can be tolerated as for typical problems $n < 10^7$. The implementation used to produce the results shown in this dissertation does not implement a spacial tree (for example an octree) for Entity Mapping. In turn the current implementation has a complexity of $\mathcal{O}(n^2)$ and is limited to small meshes. However, this issue only relates to the implementation, not to the method. (Szirmay-Kalos and Márton, 1998)

The meshes produced by the current implementation of the method are good for some examples but need further improvements. The ability to create sharp concave geometries should be enhanced by widening the pallet of provided Super Elements.

10 Outlook

The results of this dissertation open two main paths for future research. The generation of meshes with ASP based on a graph model of the problem is a Non Polynomial (NP) algorithm. An open question is whether the problem itself is NP hard. This seems likely and would prove optimal generation of large meshes to be futile. The formulation of meshing as a graph selection problem is strict enough to enable its time complexity analysis. Thus it will aid the complexity analyses of hexahedral meshing.

The second path to tread is to develop the algorithm presented in this dissertation – or a combination of it and the algorithm by Owen and Shelton (2015) – into an industry viable software. In this process the implementation of the Entity Mapping algorithm has to be enhanced and more Super Elements have to be added in order to improve results for sharp concave geometries. Doing so would make performing FEM simulations in mechanical engineering a much more effective tool.

Bibliography

Anderson, A., Zheng, X. and Cristini, V. (2005), 'Adaptive unstructured volume remeshing – I: The method', *Journal of Computational Physics* **208**(2), 616–625.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0021999105001221>

Ansys Inc. (2022), 'Ansys Mechanical 2022 R2'.

Bathe, K.-J. (2006), *Finite Element Procedures*, 2nd edn, Englewood Cliffs Prentice Hall.

Bentley, J. L. (1975), 'Multidimensional binary search trees used for associative searching', *Communications of the ACM* **18**(9), 509–517.

URL: <https://doi.org/10.1145/361002.361007>

Bhowmick, S. and Shontz, S. M. (2010), 'Towards high-quality, untangled meshes via a force-directed graph embedding approach', *Procedia Computer Science* **1**(1), 357–366.

URL: <https://www.sciencedirect.com/science/article/pii/S1877050910000402>

Biere, A., Heule, M., van Maaren, H. and Walsh, T., eds (2021), *Handbook of Satisfiability - Second Edition*, Vol. 336 of *Frontiers in Artificial Intelligence and Applications*, IOS Press.

URL: <https://doi.org/10.3233/FAIA336>

Briggs, W. L., Van Emden, H. and McCormick, S. F. (2000), *A Multigrid Tutorial*, 2 edn, SIAM, Philadelphia.

Cignoni, P., Montani, C., Perego, R. and Scopigno, R. (1993), 'Parallel 3D Delaunay Triangulation', *Computer Graphics Forum* **12**(3), 129–142.

URL: <https://onlinelibrary.wiley.com/doi/10.1111/1467-8659.1230129>

Cormen, T. H., Leiserson, C. E., Rivest, R. L. and Stein, C. (2001), *Introduction To Algorithms*, 2 edn, The MIT Press, London.

Delaunay, B. N. (1934), 'Sur la sphère vide (on an empty sphere)', **7**(6), 793–800.

Demmel, J. W., Gilbert, J. R. and Li, X. S. (1999), SuperLU users' guide, Technical Report LBNL–44289, 751785, Lawrence Berkeley National Lab (LBNL).

URL: <http://www.osti.gov/servlets/purl/751785-85h6H0/webviewable/>

European Union (2003), 'Commission recommendation of 6 may 2003 concerning the definition of micro, small and medium-sized enterprises', (2003/361/EC).

Galerkin, B. G. (1915), 'СТЕРЖНИ И ПЛАСТИНКИ РЯДЫ В НЕКОТОРЫХ ВОПРОСАХ УПРУТОГО РАВНОВЕСИЯ СТЕРЖНЕЙ И ПЛАСТИНОК (beams and plates. Series in some questions of elastic equilibrium of beams and plates)', *Vestnik Ingenerov* **19**, 897–908.

Gallier, J. and Xu, D. (2012), 'A Guide to the Classification Theorem for Compact Surfaces'.

URL: <https://www.cis.upenn.edu/~jean/surfclass-n.pdf>

Gao, X., Huang, J., Xu, K., Pan, Z., Deng, Z. and Chen, G. (2017), 'Evaluating Hex-mesh Quality Metrics via Correlation Analysis', *Computer Graphics Forum* **36**(5), 105–116.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. (2019), 'Multi-shot ASP solving with clingo', **19**. arXiv:1705.09811 [cs] type: article.

URL: <http://arxiv.org/abs/1705.09811>

Geuzaine, C. and Remacle, J.-F. (2009), 'Gmsh: A 3-D Finite Element Mesh Generator with Built-in Pre- and Post-Processing Facilities', *International Journal for Numerical Methods in Engineering* **79**, 1309–1331.

Halpern, M. (1997), Industrial requirements and practices in finite element meshing: a survey of trends, in 'Proceedings 6th International Meshing Roundtable', pp. 399–411.

Harwick, M., Clay, R. L., Boggs, P. T., Wahlsh, E. J., Larzelere, A. R. and Altshuler, A. (2005), DART System Analysis, Technical report, Sandia National Laborato-

ries (SNL).

URL: <https://www.osti.gov/servlets/purl/876325>

Hormann, K. and Agathos, A. (2001), 'The point in polygon problem for arbitrary polygons', *Computational Geometry* **20**, 131–144.

ISO (2020), Standard for the exchange of product model data, Technical Report ISO 10303, International Organization for Standardization.

Kang, P. and Youn, S.-K. (2015), 'Isogeometric analysis of topologically complex shell structures', *Finite Elements in Analysis and Design* **99**, 68–81.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0168874X15000141>

Knuth, D. E. (2015), *The art of computer programming*, Vol. 4, 2 edn, Addison-Wesley, Upper Saddle River NJ.

Kremer, M., Bommers, D., Lim, I. and Kobbelt, L. (2014), Advanced Automatic Hexahedral Mesh Generation from Surface Quad Meshes, in J. Sarrate and M. Staten, eds, 'Proceedings of the 22nd International Meshing Roundtable', Springer International Publishing, Cham, pp. 147–164.

Krumke, S. O. and Noltemeier, H. (2012), *Graphentheoretische Konzepte und Algorithmen*, 3 edn, Vieweg+Teubner, Wiesbaden.

Ledoux, F. (2022), 'MAMBO'.

URL: <https://gitlab.com/franck.ledoux/mambo>

Lobos, C. (2015), Towards a unified measurement of quality for mixed-elements, Technical Report 2015/01, Departamento de Informática, UTFSM.

URL: <http://www.inf.utfsm.cl/clobos/tech.html>

Lu, Y., Gadh, R. and Tautges, T. J. (2001), 'Feature based hex meshing methodology: feature recognition and volume decompositionq', *Computer-Aided Design* **33**(3), 221–232.

Maréchal, L. (2009), Advances in Octree-Based All-Hexahedral Mesh Generation: Handling Sharp Features, in B. W. Clark, ed., 'Proceedings of the 18th International Meshing Roundtable', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 65–84.

Meagher, D. (1982), 'Geometric modeling using octree encoding', *Computer Graphics and Image Processing* **19**(2), 129–147.

URL: <https://www.sciencedirect.com/science/article/pii/0146664X82901046>

Nealen, A., Igarashi, T., Sorkine, O. and Alexa, M. (2006), Laplacian mesh optimization, in 'Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia', GRAPHITE '06, Association for Computing Machinery, New York, NY, USA, pp. 381–389.

URL: <https://doi.org/10.1145/1174429.1174494>

NetworkX developers (2022), 'NetworkX — NetworkX documentation'.

URL: <https://networkx.org/>

Open Cascade (2022).

URL: <https://www.opencascade.com/>

OpenCFD Ltd (2022), 'OpenFOAM'.

URL: <https://www.openfoam.com/>

Ortega, J. and Reinbolt, W. (2000), *Iterative Solution of Nonlinear Equations in Several Variables*, Society for Industrial & Applied Mathematics, Philadelphia.

Owen, S. J. and Shelton, T. R. (2015), 'Evaluation of grid-based hex meshes for solid mechanics', *Engineering with Computers* **31**(3), 529–543.

URL: <https://doi.org/10.1007/s00366-014-0368-8>

Parthasarathy, V. N., Graichen, C. M. and Hathaway, A. F. (1994), 'A comparison of tetrahedron quality measures', *Finite Elements in Analysis and Design* **15**(3), 255–261.

URL: <https://www.sciencedirect.com/science/article/pii/0168874X94900337>

Pellerin, J., Levy, B. and Caumon, G. (2011), Topological control for isotropic remeshing of non-manifold surfaces with varying resolution: application to 3D structural models, in 'IAMG 2011 - Mathematical Geosciences at the Crossroads of Theory and Practice', Salzburg.

Perelman, G. (2002), 'The entropy formula for the Ricci flow and its geometric applications'. arXiv:math/0211159.

URL: <http://arxiv.org/abs/math/0211159>

Perelman, G. (2003a), ‘Finite extinction time for the solutions to the Ricci flow on certain three-manifolds’.

URL: <http://arxiv.org/abs/math/0307245>

Perelman, G. (2003b), ‘Ricci flow with surgery on three-manifolds’. arXiv:math/0303109.

URL: <http://arxiv.org/abs/math/0303109>

Piegl, L. and Tiller, W. (1997), *The NURBS Book*, 2 edn, Springer-Verlag Berlin, Berlin, Heidelberg.

Pietroni, N., Campen, M., Sheffer, A., Cherchi, G., Bommes, D., Gao, X., Scateni, R., Ledoux, F., Remacle, J.-F. and Livesu, M. (2022), ‘Hex-Mesh Generation and Processing: a Survey’.

URL: <http://arxiv.org/abs/2202.12670>

Pissanetzky, S. (1984), *Sparse Matrix Technology*, Academic Press, London.

Reberol, M., Georgiadis, C. and Remacle, J.-F. (2021), ‘Quasi-structured quadrilateral meshing in Gmsh – a robust pipeline for complex CAD models’.

URL: <http://arxiv.org/abs/2103.04652>

Rivest, R. L., Shamir, A. and Adleman, L. (1978), ‘A Method for Obtaining Digital Signatures and Public-Key Cryptosystems’, *Communications of the ACM* **21**(2), 120–126.

Roca Navarro, X. (2009), Paving the path towards automatic hexahedral mesh generation, Ph.D. Thesis, Universitat Politècnica de Catalunya.

URL: <http://www.tdx.cat/handle/10803/5858>

Sarrate, J., Ruiz, E. and Roca, X. (2014), ‘Unstructured and semi-structured hexahedral mesh generation methods’, *Computational Technology Reviews* **10**, 35–64.

Schneider, T., Hu, Y., Gao, X., Dumas, J., Zorin, D. and Panozzo, D. (2019), ‘A Large Scale Comparison of Tetrahedral and Hexahedral Elements for Finite Element Analysis’.

URL: <http://arxiv.org/abs/1903.09332>

- Shepherd, J. F. (2007), *Topological and geometric constraint-based hexahedral mesh generation*, University of Utah. PhD Thesis, University of Utah, USA.
- Shepherd, J. F. (2008), 'Hexahedral mesh generation constraints', *Engineering with Computers* **24**(3), 195–213.
- Sonthi', R., Kunjur, G. and Gadh', R. (1997), 'Shape Feature Determination using the Curvature Region Representation', *Proceedings of the fourth ACM symposium on Solid modeling and applications* pp. 285–296.
- Stricklin, J. A., Ho, W. S., Richardson, E. Q. and Haisler, W. E. (1977), 'On isoparametric vs linear strain triangular elements', *International Journal for Numerical Methods in Engineering* **11**(6), 1041–1043.
URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620110610>
- Stromberg, H., Hochberger, I. and Lohrengel, A. (2021), 'Anwenderperspektive auf FEM Software', *Institut für Maschinenwesen Institutsmitteilung* **46**.
- Stromberg, H., Mayer-Eichberger, V. and Lohrengel, A. (2023), 'Combinatorial local mesh generation'.
- Szirmay-Kalos, L. and Márton, G. (1998), 'Worst-case versus average case complexity of ray-shooting', *Computing* **61**(2), 103–131.
URL: <https://doi.org/10.1007/BF02684409>
- Tautges, T. J. (2004), 'MOAB-SD: integrated structured and unstructured mesh representation', *Engineering with Computers* **20**(3), 286–293.
URL: <https://doi.org/10.1007/s00366-004-0296-0>
- Wriggers, P. (2001), *Nichtlineare Finite-Elemente-Methoden*, Springer, Berlin.
- Yamakawa, S., Gentilini, I. and Shimada, K. (2011), 'Subdivision templates for converting a non-conformal hex-dominant mesh to a conformal hex-dominant mesh without pyramid elements', *Engineering with Computers* **27**(1), 51–65.
URL: <https://doi.org/10.1007/s00366-010-0178-6>
- Yerry, M. A. and Shephard, M. S. (1984), 'Automatic three-dimensional mesh generation by the modified-octree technique', *International Journal for Numerical Methods in Engineering* **20**(11), 1965–1990.

Zheng, X., Lowengrub, J., Anderson, A. and Cristini, V. (2005), 'Adaptive un-structured volume remeshing – II: Application to two- and three-dimensional level-set simulations of multiphase flow', *Journal of Computational Physics* **208**(2), 626–650.

URL: <https://linkinghub.elsevier.com/retrieve/pii/S0021999105001233>

A Questionnaire



Vielen Dank, dass Sie sich die Zeit nehmen meine Forschung durch die Teilnahme an dieser Umfrage zu unterstützen,

Henrik Stromberg

Teil A: Einordnung des Unternehmens

Zur statistischen Einordnung Ihrer Antworten möchte ich mit einigen Fragen zu dem Unternehmen, in dem Sie tätig sind, beginnen. Falls Sie in einem Konzern arbeiten, betrachten Sie bitte nur ihre autonome Organisation (z.B. MAN statt VW Konzern).

A1. In welchem Sektor ist Ihr Unternehmen vorwiegend tätig?

Produktion ☐

Dienstleistung ☐

A2. In welcher Branche ist das Unternehmen hauptsächlich tätig?

A3. Wie viele Mitarbeiter beschäftigt das Unternehmen insgesamt?

Bis zu 50 ☐

Bis zu 250 ☐

Bis 1000 ☐

Mehr als 10000 ☐

A4. Wie viele Umsatz generiert das Unternehmen jährlich?

Bis 10 Mio. € ☐

Bis 50 Mio. € ☐

Bis 500 Mio. € ☐

Mehr als 1 Mrd. € ☐

A5. Wo hat das Unternehmen Standorte? (Mehrfachnennung möglich)

Ozeanien ☐

Asien ☐

Afrika ☐

Südamerika ☐

--	--

--	--

--	--

[illegible]

--	--

--	--

--	--

1

--	--

1

7

1

1

--	--

7

1

1

1



B4. Welche FEM-Software benutzen Sie oder andere Mitarbeiter des Unternehmens aktuell vorwiegend?

Ansys ☐

NX Nastran ☐

Abaqus ☐

Comsol Multiphysics ☐

Hyperworks ☐

Marc ☐

Sonstiges ☐

Sonstiges

B5. Wie zufrieden sind Sie mit der Benutzerfreundlichkeit dieser Software? (5 entspricht maximaler Zufriedenheit)

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

B6. Wie zufrieden sind Sie mit dem Funktionsumfang dieser Software? (5 entspricht maximaler Zufriedenheit)

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

B7. Haben Sie Verbesserungswünsche für diese Software?



B8. Würden Sie lieber mit einer anderen Software als der aktuell genutzten arbeiten wollen?

Ja ☐

Nein ☐

B9. Welchen Stellwert misst das Unternehmen, in dem Sie tätig sind, Simulationssoftware bei? (5 entspricht maximalem Stellenwert)

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

B10. Welchen Stellenwert würden Sie persönlich Simulationssoftware beimesen (unabhängig von Ihrer aktuellen Nutzung)? (5 entspricht maximalem Stellenwert)

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

B11. Sehen Sie einen Bedarf für die Nutzung von FEM-Software in Ihrem Unternehmen?

Ja ☐

Nein ☐

B12. Haben Sie bereits vor Ihrer aktuellen Tätigkeit Erfahrung mit FEM-Software gesammelt?

Ja, in wissenschaftlichen Tätigkeiten (z.B. Studium oder Promotion) ☐

Ja, an einem früheren Arbeitsplatz ☐

Ja, beides ☐

Nein ☐



B13. Welche FEM-Software haben Sie selbst in der Vergangenheit genutzt? (Mehrfachnennung möglich)

- Ansys ☐
- NX Nastran ☐
- Abaqus ☐
- Comsol Multiphysics ☐
- Hyperworks ☐
- Marc ☐
- Sonstiges ☐

Sonstiges

B14. Könnten Sie sich vorstellen eine FEM-Software als Onlineservice zu nutzen?

- Ja ☐
- Nein, aus Datenschutzgründen ☐
- Nein, aus organisatorischen Gründen ☐
- Nein, aus anderen Gründen ☐

B15. Welche CAD-Software wird im Unternehmen genutzt? (Mehrfachnennung möglich)

- Inventor ☐
- AutoCAD ☐
- SolidWorks ☐
- CATIA ☐
- NX ☐
- Creo ☐
- Sonstiges ☐

Sonstiges



B16. Besteht aus Ihrer Sicht Bedarf nach einem neutralen CAD-Format, das bearbeitbare Feature enthält?

Ja ☐

Nein ☐

Teil C: Angaben zur Person

C1. Bitte nennen Sie Ihre Position bzw. Ihr Tätigkeitsfeld im Unternehmen.

C2. Wie viele Jahre Berufserfahrung haben Sie bereits?

Bis zu 5 Jahre ☐

Bis zu 10 Jahre ☐

Bis zu 20 Jahre ☐

Mehr als 20 Jahre ☐

Vielen Dank für Ihre Teilnahme!

B Considered Node locations

The following tables list the Considered Node locations as shown in 5.3, 5.4, 5.5, 5.6. Outer nodes are printed in bold.

Table B.1: Considered Node locations for hexahedrons

Node index	ξ	η	ζ
0	-1.0	-1.0	-1.0
1	1.0	-1.0	-1.0
2	1.0	1.0	-1.0
3	-1.0	1.0	-1.0
4	-1.0	-1.0	1.0
5	1.0	-1.0	1.0
6	1.0	1.0	1.0
7	-1.0	1.0	1.0
8	0.0	-1.0	-1.0
9	-1.0	0.0	-1.0
10	-1.0	-1.0	0.0
11	1.0	0.0	-1.0
12	1.0	-1.0	0.0
13	1.0	0.0	1.0
14	0.0	-1.0	1.0
15	-1.0	0.0	1.0
16	0.0	1.0	-1.0
17	1.0	1.0	0.0
18	0.0	1.0	1.0
19	-1.0	1.0	0.0
20	0.0	0.0	-1.0
21	1.0	0.0	0.0
22	0.0	0.0	1.0
23	-1.0	0.0	0.0
24	0.0	-1.0	0.0
25	0.0	1.0	0.0
26	0.0	0.0	0.0
27	0.5	-0.5	-0.5
28	0.5	0.5	-0.5
29	-0.5	0.5	-0.5

Continued on the following page

Node index	ξ	η	ζ
30	-0.5	-0.5	-0.5
31	0.5	-0.5	0.5
32	0.5	0.5	0.5
33	-0.5	0.5	0.5
34	-0.5	-0.5	0.5

Table B.2: Considered Node locations for tetrahedrons

Node index	ξ	η	ζ
0	0.0	0.0	0.0
1	1.0	0.0	0.0
2	0.0	1.0	0.0
3	0.0	0.0	1.0
4	0.5	0.0	0.0
5	0.0	0.5	0.0
6	0.0	0.0	0.5
7	0.5	0.5	0.0
8	0.5	0.0	0.5
9	0.0	0.5	0.5

Table B.3: Considered Node locations for prisms

Node index	ξ	η	ζ
0	0.0	0.0	-1.0
1	1.0	0.0	-1.0
2	0.0	1.0	-1.0
3	0.0	0.0	1.0
4	1.0	0.0	1.0
5	0.0	1.0	1.0
6	0.5	0.0	-1.0
7	0.0	0.0	0.0
8	0.5	0.5	-1.0
9	1.0	0.0	0.0
10	0.0	0.5	-1.0
11	0.0	1.0	0.0
12	0.5	0.0	1.0
13	0.5	0.5	1.0
14	0.0	0.5	1.0

Continued on the following page

Node index	ξ	η	ζ
15	0.0	0.5	0.0
16	0.5	0.5	0.0
17	0.5	0.0	0.0
18	$\frac{1}{3}$	$\frac{1}{3}$	-1.0
19	$\frac{1}{3}$	$\frac{1}{3}$	1.0
20	$\frac{1}{6}$	$\frac{1}{6}$	-0.5
21	$\frac{2}{3}$	$\frac{1}{6}$	-0.5
22	$\frac{1}{6}$	$\frac{2}{3}$	-0.5
23	$\frac{1}{6}$	$\frac{1}{6}$	0.5
24	$\frac{2}{3}$	$\frac{1}{6}$	0.5
25	$\frac{1}{6}$	$\frac{2}{3}$	0.5

Table B.4: Considered Node locations for pyramids

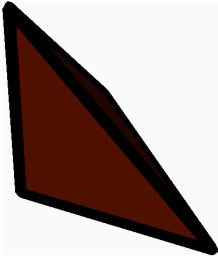
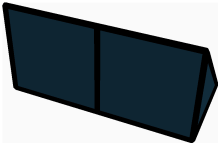
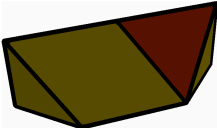
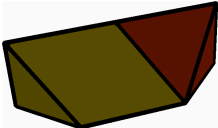
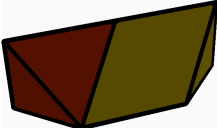
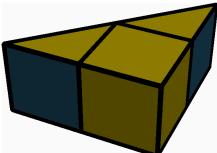
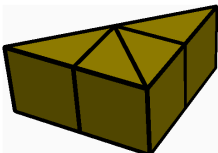
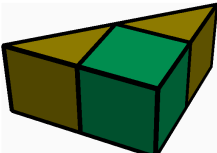
Node index	ξ	η	ζ
0	-1.0	-1.0	0.0
1	1.0	-1.0	0.0
2	1.0	1.0	0.0
3	-1.0	1.0	0.0
4	0.0	0.0	1.0
5	0.0	-1.0	0.0
6	1.0	0.0	0.0
7	0.0	1.0	0.0
8	-1.0	0.0	0.0
9	-0.5	-0.5	0.5
10	0.5	-0.5	0.5
11	0.5	0.5	0.5
12	-0.5	0.5	0.5
13	0.0	0.0	0.0

C Super Elements

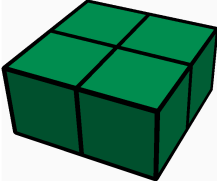
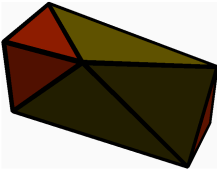
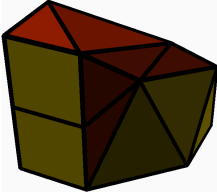
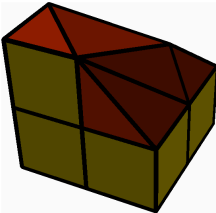
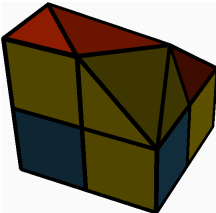
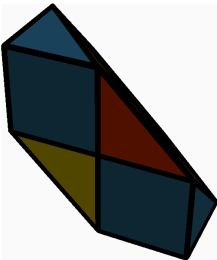
In the following tables the generated Super Elements are visualized. Hexahedrons are printed in green, tetrahedrons in red, prisms in blue and pyramids in yellow.

In many cases the different optimizations converge into the same mesh.

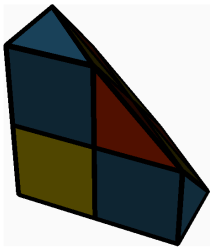
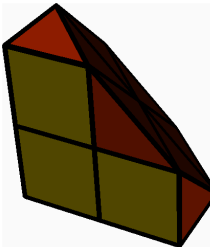
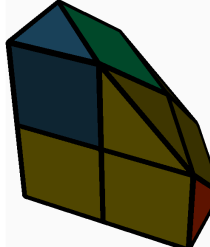
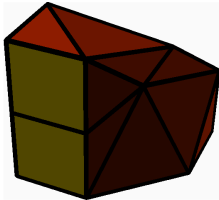
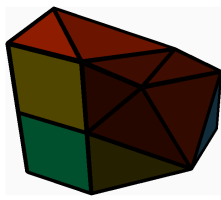
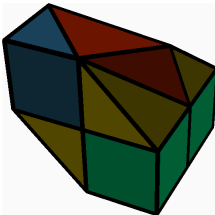
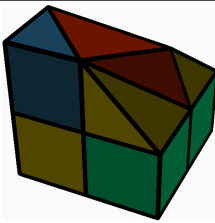
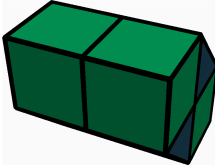
Table C.1: Hexahedral Super Elements with different optimization goals

Index	Max. SJ	Min. Element Count	Min. Valence
0			
1			
2			
3			

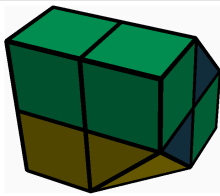
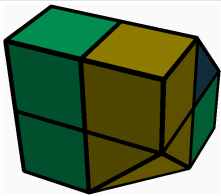
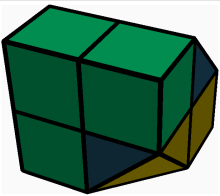
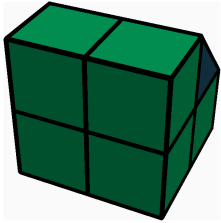
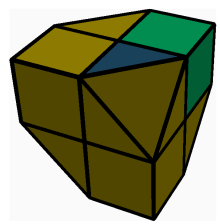
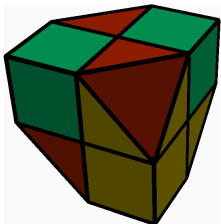
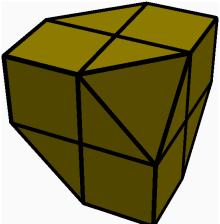
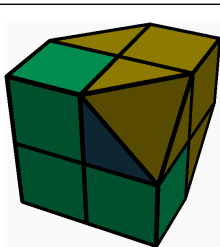
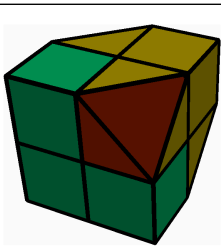
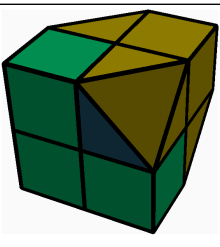
Continued on the following page

Index	Max. SJ	Min. Element Count	Min. Valence
4			
5			
6			
7			
8			

Continued on the following page

Index	Max. SJ	Min. Element Count	Min. Valence
9			
10			
11			
12			
13			

Continued on the following page

Index	Max. SJ	Min. Element Count	Min. Valence
14			
15			
16			
17			

1

Continued on the following page

¹In this case the resulting meshes for Max. SJ and Min.Valence are identical

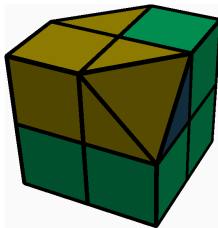
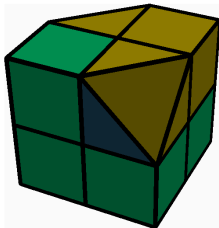
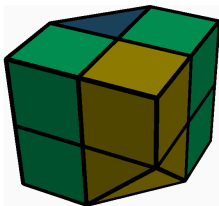
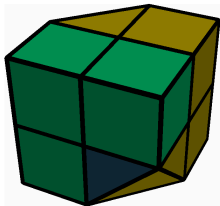
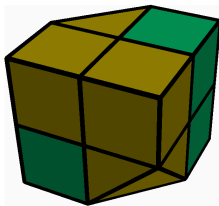
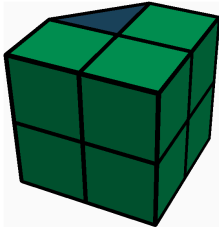
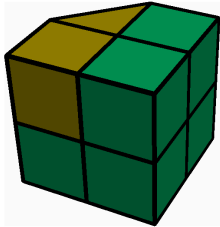
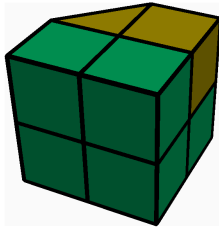
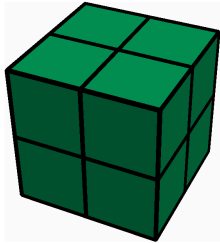
Index	Max. SJ	Min. Element Count	Min. Valence
18			
19			
20			
21			

Table C.2: Incidence matrices for hexahedral Super Elements optimized for maximized SJ

Super Element	Incidence Matrix							
0								
1	0	8	9	10				
2	0	9	10	8	20	24		
	1	11	12	8	20	24		
3	11	17	20	26				
	16	17	20	26				
	0	8	20	9	10			
	2	11	20	16	17			
	8	10	26	11	20			
	9	10	26	16	20			
4	8	20	24	27				
	9	16	20	26				
	11	20	21	27				
	20	21	26	27				
	20	24	26	27				
	0	9	10	8	20	24		
	2	16	17	11	20	21		
	1	8	20	11	27			
	1	8	24	12	27			
	1	11	21	12	27			
	9	10	24	20	26			
	12	21	26	24	27			
5	16	17	21	20	26			
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
6	3	9	20	16	19	23	26	25
	2	11	16	17				
	4	10	14	15				
	10	14	17	16	11			
	10	14	17	16	15			

Continued on the following page

Super Element		Incidence Matrix				
7	4	14	15	26		
	8	24	26	31		
	9	23	26	33		
	14	15	26	32		
	14	24	26	31		
	14	26	31	32		
	15	23	26	33		
	15	26	32	33		
	17	26	31	32		
	17	26	32	33		
	0	8	20	9	26	
	0	8	24	10	26	
	0	9	23	10	26	
	2	11	20	16	17	
	4	10	24	14	26	
	4	10	23	15	26	
	8	11	17	26	20	
	8	11	17	26	31	
	9	16	17	26	20	
	9	16	17	26	33	
8	2	16	17	26		
	4	14	15	26		
	9	16	20	26		
	9	16	26	33		
	9	23	26	33		
	12	14	21	26		
	12	14	24	26		
	14	15	26	32		
	14	21	26	32		
	15	23	26	33		
	15	26	32	33		
	16	17	26	33		
	17	21	26	32		
	17	26	32	33		
	0	8	20	9	26	
	0	8	24	10	26	
	0	9	23	10	26	
	1	8	20	11	26	
	1	8	24	12	26	
	1	11	21	12	26	
	2	11	20	16	26	
	2	11	21	17	26	
	4	10	24	14	26	
	4	10	23	15	26	

Continued on the following page

Super Element	Incidence Matrix					
9	11	16	20	26		
	12	14	24	26		
	15	19	23	26		
	20	23	24	26		
	1	11	12	8	20	24
	3	16	19	9	20	23
	4	14	15	10	24	23
	8	9	23	24	10	
	8	9	23	24	20	
	11	12	24	20	26	
	14	15	23	24	26	
	16	19	23	20	26	
10	8	20	24	30		
	9	20	23	30		
	10	23	24	30		
	11	16	20	26		
	12	14	24	26		
	15	19	23	26		
	20	23	26	30		
	20	24	26	30		
	23	24	26	30		
	1	11	12	8	20	24
	3	16	19	9	20	23
	4	14	15	10	24	23
	0	8	20	9	30	
	0	8	24	10	30	
	0	9	23	10	30	
	11	12	24	20	26	
	14	15	23	24	26	
	16	19	23	20	26	

Continued on the following page

Super Element		Incidence Matrix							
11	2	11	17	26					
	4	14	15	26					
	8	11	20	26					
	8	11	26	31					
	8	24	26	31					
	11	17	26	31					
	14	15	26	32					
	14	24	26	31					
	14	26	31	32					
	15	19	23	26					
	15	19	26	32					
	17	25	26	32					
	17	26	31	32					
	19	25	26	32					
	0	8	20	9	26				
	0	8	24	10	26				
	0	9	23	10	26				
	2	11	20	16	26				
	2	16	25	17	26				
	3	9	20	16	26				
	3	9	23	19	26				
	3	16	25	19	26				
	4	10	24	14	26				
	4	10	23	15	26				
12	14	15	26	32					
	14	21	26	32					
	15	25	26	32					
	1	8	24	12	11	20	26	21	
	2	11	21	17	16	20	26	25	
	3	9	20	16	19	23	26	25	
	4	14	15	10	24	23			
	8	9	20	24	23	26			
	8	9	23	24	10				
	12	21	26	24	14				
	14	15	23	24	26				
	26	23	19	25	15				
	17	21	26	25	32				

Continued on the following page

Super Element	Incidence Matrix							
13	10	23	24	30				
	14	15	26	32				
	14	21	26	32				
	15	25	26	32				
	23	24	26	30				
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	14	15	10	24	23		
	0	8	20	9	30			
	0	8	24	10	30			
	0	9	23	10	30			
	8	20	26	24	30			
	9	20	26	23	30			
	12	21	26	24	14			
	14	15	23	24	26			
	26	23	19	25	15			
	17	21	26	25	32			
14	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	9	10	23	20	24	26		
	11	12	21	20	24	26		
	13	17	21	22	25	26		
	15	19	23	22	25	26		
15	8	20	24	30				
	20	24	26	30				
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	11	12	21	20	24	26		
	13	17	21	22	25	26		
	15	19	23	22	25	26		
	0	8	20	9	30			
	0	8	24	10	30			
	0	9	23	10	30			
	24	12	11	20	8			
	9	20	26	23	30			
	10	23	26	24	30			

Continued on the following page

Super Element	Incidence Matrix							
16	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	13	17	21	22	25	26		
	15	19	23	22	25	26		
17	10	23	24	34				
	11	20	21	27				
	12	21	24	27				
	15	22	23	34				
	20	21	26	27				
	21	24	26	27				
	22	23	26	34				
	23	24	26	34				
	3	9	20	16	19	23	26	25
	6	13	22	18	17	21	26	25
	8	9	20	24	23	26		
	13	14	22	21	24	26		
	16	17	25	20	21	26		
	18	19	25	22	23	26		
	1	8	20	11	27			
	1	8	24	12	27			
	1	11	21	12	27			
	4	10	24	14	34			
	4	10	23	15	34			
	4	14	22	15	34			
	8	9	23	24	10			
	8	20	26	24	27			
	21	17	16	20	11			
	24	14	13	21	12			
	14	22	26	24	34			
	23	19	18	22	15			

Continued on the following page

Super Element		Incidence Matrix							
18	13	21	22	32					
	17	21	25	32					
	18	22	25	32					
	21	22	26	32					
	21	25	26	32					
	22	25	26	32					
	0	8	20	9	10	24	26	23	
	1	8	24	12	11	20	26	21	
	3	9	20	16	19	23	26	25	
	4	10	23	15	14	24	26	22	
	11	16	20	21	25	26			
	12	14	24	21	22	26			
	15	19	23	22	25	26			
	6	13	21	17	32				
	6	13	22	18	32				
	6	17	25	18	32				
	11	16	25	21	17				
	12	14	22	21	13				
15	19	25	22	18					
19	14	22	24	34					
	15	22	23	34					
	22	23	26	34					
	22	24	26	34					
	0	8	20	9	10	24	26	23	
	1	8	24	12	11	20	26	21	
	2	11	21	17	16	20	26	25	
	3	9	20	16	19	23	26	25	
	6	13	22	18	17	21	26	25	
	12	13	21	24	22	26			
	18	19	25	22	23	26			
	4	10	24	14	34				
	4	10	23	15	34				
	4	14	22	15	34				
	10	23	26	24	34				
	12	13	22	24	14				
	23	19	18	22	15				

Continued on the following page

Super Element	Incidence Matrix							
20	12	21	24	31				
	19	23	25	29				
	21	24	26	31				
	23	25	26	29				
	0	8	20	9	10	24	26	23
	2	11	21	17	16	20	26	25
	4	10	23	15	14	24	26	22
	6	13	22	18	17	21	26	25
	8	11	20	24	21	26		
	15	18	22	23	25	26		
	3	9	20	16	29			
	3	9	23	19	29			
	3	16	25	19	29			
	5	12	21	13	31			
	5	12	24	14	31			
	5	13	22	14	31			
	8	11	21	24	12			
	9	20	26	23	29			
	13	21	26	22	31			
	14	22	26	24	31			
	15	18	25	23	19			
	16	20	26	25	29			
21	19	23	25	29				
	23	25	26	29				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	6	13	22	18	17	21	26	25
	15	18	22	23	25	26		
	3	9	20	16	29			
	3	9	23	19	29			
	3	16	25	19	29			
	9	20	26	23	29			
	15	18	25	23	19			
	16	20	26	25	29			
22	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	6	13	22	18	17	21	26	25
	7	15	23	19	18	22	26	25

Table C.3: Incidence matrices for hexahedral Super Elements optimized for minimized element count

Super Element	Incidence Matrix							
0								
1	0	8	9	10				
2	0	9	10	8	20	24		
	1	11	12	8	20	24		
3	2	11	17	26				
	2	16	17	26				
	0	8	20	9	10			
	2	11	20	16	26			
	8	10	26	11	20			
	9	10	26	16	20			
4	0	9	10	26				
	2	16	17	26				
	9	16	20	26				
	0	8	20	9	26			
	0	8	24	10	26			
	1	8	20	11	26			
	1	8	24	12	26			
	1	11	21	12	26			
	2	11	20	16	26			
5	2	11	21	17	26			
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
6	3	9	20	16	19	23	26	25
	2	11	16	17				
	4	10	14	15				
	10	14	17	16	11			
	10	14	17	16	15			
Continued on the following page								

Super Element	Incidence Matrix				
7	4	14	15	26	
	8	24	26	31	
	9	23	26	33	
	14	15	26	32	
	14	24	26	31	
	14	26	31	32	
	15	23	26	33	
	15	26	32	33	
	17	26	31	32	
	17	26	32	33	
	0	8	20	9	26
	0	8	24	10	26
	0	9	23	10	26
	2	11	20	16	17
	4	10	24	14	26
	4	10	23	15	26
	8	11	17	26	20
	8	11	17	26	31
	9	16	17	26	20
	9	16	17	26	33
8	2	16	17	26	
	4	14	15	26	
	9	16	20	26	
	9	16	26	33	
	9	23	26	33	
	12	14	21	26	
	12	14	24	26	
	14	15	26	32	
	14	21	26	32	
	15	23	26	33	
	15	26	32	33	
	16	17	26	33	
	17	21	26	32	
	17	26	32	33	
	0	8	20	9	26
	0	8	24	10	26
	0	9	23	10	26
	1	8	20	11	26
	1	8	24	12	26
	1	11	21	12	26
	2	11	20	16	26
	2	11	21	17	26
	4	10	24	14	26
	4	10	23	15	26

Continued on the following page

Super Element	Incidence Matrix					
9	11	16	20	33		
	12	14	24	33		
	15	19	23	33		
	20	23	24	33		
	1	11	12	8	20	24
	3	16	19	9	20	23
	4	14	15	10	24	23
	8	9	23	24	10	
	8	9	23	24	20	
	11	12	24	20	33	
	14	15	23	24	33	
	16	19	23	20	33	
10	1	11	12	26		
	3	16	19	26		
	4	14	15	26		
	11	16	20	26		
	12	14	24	26		
	15	19	23	26		
	0	8	20	9	26	
	0	8	24	10	26	
	0	9	23	10	26	
	1	8	20	11	26	
	1	8	24	12	26	
	3	9	20	16	26	
	3	9	23	19	26	
	4	10	24	14	26	
	4	10	23	15	26	

Continued on the following page

Super Element	Incidence Matrix							
11	2	11	17	26				
	4	14	15	26				
	8	11	20	26				
	8	11	26	31				
	8	24	26	31				
	11	17	26	31				
	14	15	26	32				
	14	24	26	31				
	14	26	31	32				
	15	19	23	26				
	15	19	26	32				
	17	25	26	32				
	17	26	31	32				
	19	25	26	32				
	0	8	20	9	26			
	0	8	24	10	26			
	0	9	23	10	26			
	2	11	20	16	26			
	2	16	25	17	26			
	3	9	20	16	26			
	3	9	23	19	26			
	3	16	25	19	26			
	4	10	24	14	26			
	4	10	23	15	26			
12	14	15	26	32				
	14	21	26	32				
	15	25	26	32				
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	14	15	10	24	23		
	8	9	20	24	23	26		
	8	9	23	24	10			
	12	21	26	24	14			
	14	15	23	24	26			
	26	23	19	25	15			
	17	21	26	25	32			

Continued on the following page

Super Element	Incidence Matrix							
13	10	23	24	30				
	14	15	26	32				
	14	21	26	32				
	15	25	26	32				
	23	24	26	30				
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	14	15	10	24	23		
	0	8	20	9	30			
	0	8	24	10	30			
	0	9	23	10	30			
	8	20	26	24	30			
	9	20	26	23	30			
	12	21	26	24	14			
	14	15	23	24	26			
	26	23	19	25	15			
	17	21	26	25	32			
14	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	9	10	23	20	24	26		
	11	12	21	20	24	26		
	13	17	21	22	25	26		
	15	19	23	22	25	26		
15	12	21	24	31				
	21	24	26	31				
	0	8	20	9	10	24	26	23
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	8	11	20	24	21	26		
	13	17	21	22	25	26		
	15	19	23	22	25	26		
	5	12	21	13	31			
	5	12	24	14	31			
	5	13	22	14	31			
	8	11	21	24	12			
	13	21	26	22	31			
	14	22	26	24	31			

Continued on the following page

Super Element	Incidence Matrix							
16	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	13	17	21	22	25	26		
	15	19	23	22	25	26		
17	8	9	10	24				
	8	9	20	26				
	8	9	24	26				
	11	16	17	25				
	11	16	20	26				
	11	16	25	26				
	12	13	14	22				
	15	18	19	22				
	4	10	23	15	14	24	26	22
	6	13	22	18	17	21	26	25
	1	8	20	11	26			
	1	8	24	12	26			
	1	11	21	12	26			
	3	9	20	16	26			
	3	9	23	19	26			
	3	16	25	19	26			
	26	23	10	24	9			
	26	21	17	25	11			
	26	21	13	22	12			
	26	22	14	24	12			
	15	22	26	23	19			
	18	22	26	25	19			

Continued on the following page

Super Element	Incidence Matrix							
18	11	16	17	20				
	12	13	14	24				
	15	18	19	23				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	6	13	21	17	26			
	6	13	22	18	26			
	6	17	25	18	26			
	11	20	26	21	17			
	12	21	26	24	13			
	26	22	14	24	13			
	15	22	26	23	18			
	16	20	26	25	17			
	26	23	19	25	18			
19	14	22	24	34				
	15	22	23	34				
	22	23	26	34				
	22	24	26	34				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	6	13	22	18	17	21	26	25
	12	13	21	24	22	26		
	18	19	25	22	23	26		
	4	10	24	14	34			
	4	10	23	15	34			
	4	14	22	15	34			
	10	23	26	24	34			
	12	13	22	24	14			
	23	19	18	22	15			

Continued on the following page

Super Element	Incidence Matrix							
20	11	20	21	28				
	18	22	25	32				
	20	21	26	28				
	22	25	26	32				
	0	8	20	9	10	24	26	23
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	8	12	24	20	21	26		
	15	19	23	22	25	26		
	2	11	20	16	28			
	2	11	21	17	28			
	2	16	25	17	28			
	6	13	21	17	32			
	6	13	22	18	32			
	6	17	25	18	32			
	8	12	21	20	11			
	13	21	26	22	32			
	15	19	25	22	18			
	16	20	26	25	28			
	17	21	26	25	28			
	17	21	26	25	32			
21	15	22	23	34				
	22	23	26	34				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	5	12	24	14	13	21	26	22
	6	13	22	18	17	21	26	25
	18	19	25	22	23	26		
	4	10	24	14	34			
	4	10	23	15	34			
	4	14	22	15	34			
	10	23	26	24	34			
	14	22	26	24	34			
	23	19	18	22	15			
22	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	6	13	22	18	17	21	26	25
	7	15	23	19	18	22	26	25

Table C.4: Incidence matrices for hexahedral Super Elements optimized for minimized node valence

Super Element	Incidence Matrix							
0								
1	0	8	9	10				
2	0	9	10	8	20	24		
	1	11	12	8	20	24		
3	0	8	10	26				
	0	9	10	26				
	0	8	20	9	26			
	2	11	20	16	17			
	8	11	17	26	20			
	9	16	17	26	20			
4	9	16	20	29				
	1	8	24	12	11	20	26	21
	0	8	30	9	20	29		
	2	11	28	16	20	29		
	8	24	30	20	26	29		
	11	21	28	20	26	29		
	0	8	24	10	30			
	0	9	29	30	10			
	2	11	21	17	28			
	2	16	29	28	17			
	30	24	26	29	10			
	29	26	21	28	17			
5	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
6	2	11	16	17				
	4	10	14	15				
	10	14	17	16	11			
	10	14	17	16	15			

Continued on the following page

Super Element	Incidence Matrix					
8	2	16	17	28		
	4	14	15	34		
	14	15	32	34		
	15	23	33	34		
	15	32	33	34		
	16	17	28	33		
	17	21	28	32		
	17	28	32	33		
	0	8	30	10	24	34
	0	9	30	10	23	34
	1	8	27	11	20	28
	1	12	27	11	21	28
	0	8	20	9	30	
	1	8	24	12	27	
	2	11	20	16	28	
	2	11	21	17	28	
	4	10	24	14	34	
	4	10	23	15	34	
	8	20	28	27	30	
	8	24	34	30	27	
	9	16	28	30	20	
	9	16	28	30	33	
	9	23	34	30	33	
	12	14	34	27	24	
	12	14	34	27	32	
	12	21	28	27	32	
	27	28	33	34	30	
27	28	33	34	32		
9	11	16	20	28		
	12	14	24	31		
	15	19	23	33		
	1	11	12	8	20	24
	3	16	19	9	20	23
	4	14	15	10	24	23
	11	20	28	12	24	31
	14	24	31	15	23	33
	16	20	28	19	23	33
	20	23	24	28	33	31
	8	9	23	24	10	
	8	9	23	24	20	

Continued on the following page

Super Element	Incidence Matrix							
11	4	14	15	34				
	11	17	26	31				
	11	24	26	31				
	14	15	32	34				
	14	24	31	34				
	14	31	32	34				
	15	19	23	25				
	16	20	25	29				
	17	25	26	32				
	17	26	31	32				
	19	23	25	29				
	20	25	26	29				
	23	25	26	29				
	24	26	31	34				
	26	31	32	34				
	0	8	20	9	10	24	26	23
	2	11	17	16	20	25		
	3	9	20	16	29			
	3	9	23	19	29			
	3	16	25	19	29			
	4	10	24	14	34			
	4	10	23	15	34			
	8	20	26	24	11			
	9	20	26	23	29			
10	23	26	24	34				
11	17	25	20	26				
34	23	25	32	15				
23	25	32	34	26				
12	14	15	26	32				
	14	21	26	32				
	15	25	26	32				
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	14	15	10	24	23		
	8	9	20	24	23	26		
	8	9	23	24	10			
	12	21	26	24	14			
	14	15	23	24	26			
	26	23	19	25	15			
	17	21	26	25	32			

Continued on the following page

Super Element	Incidence Matrix							
13	10	23	24	30				
	14	15	26	32				
	14	21	26	32				
	15	25	26	32				
	23	24	26	30				
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	14	15	10	24	23		
	0	8	20	9	30			
	0	8	24	10	30			
	0	9	23	10	30			
	8	20	26	24	30			
	9	20	26	23	30			
	12	21	26	24	14			
	14	15	23	24	26			
	26	23	19	25	15			
	17	21	26	25	32			
14	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	9	10	23	20	24	26		
	11	12	21	20	24	26		
	13	17	21	22	25	26		
	15	19	23	22	25	26		
15	11	20	21	28				
	20	21	26	28				
	0	8	20	9	10	24	26	23
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	8	12	24	20	21	26		
	13	17	21	22	25	26		
	15	19	23	22	25	26		
	2	11	20	16	28			
	2	11	21	17	28			
	2	16	25	17	28			
	8	12	21	20	11			
	16	20	26	25	28			
	17	21	26	25	28			

Continued on the following page

Super Element	Incidence Matrix							
16	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	13	17	21	22	25	26		
	15	19	23	22	25	26		
17	9	20	23	29				
	11	20	21	27				
	13	21	22	32				
	15	22	23	34				
	8	20	27	10	23	34		
	12	21	27	14	22	34		
	16	20	29	17	21	32		
	18	22	32	19	23	29		
	20	21	27	23	22	34		
	20	23	29	21	22	32		
	1	8	20	11	27			
	1	8	24	12	27			
	1	11	21	12	27			
	3	9	20	16	29			
	3	9	23	19	29			
	3	16	25	19	29			
	4	10	24	14	34			
	4	10	23	15	34			
	4	14	22	15	34			
	6	13	21	17	32			
	6	13	22	18	32			
	6	17	25	18	32			
	8	10	23	20	9			
	8	10	34	27	24			
	21	17	16	20	11			
	12	14	22	21	13			
	12	14	34	27	24			
	23	19	18	22	15			
	16	17	32	29	25			
	18	19	29	32	25			

Continued on the following page

Super Element	Incidence Matrix							
18	13	21	22	32				
	17	21	25	32				
	18	22	25	32				
	21	22	26	32				
	21	25	26	32				
	22	25	26	32				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	11	16	20	21	25	26		
	12	14	24	21	22	26		
	15	19	23	22	25	26		
	6	13	21	17	32			
	6	13	22	18	32			
	6	17	25	18	32			
	11	16	25	21	17			
	12	14	22	21	13			
	15	19	25	22	18			
19	13	21	22	32				
	18	22	25	32				
	21	22	26	32				
	22	25	26	32				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	12	14	24	21	22	26		
	15	19	23	22	25	26		
	6	13	21	17	32			
	6	13	22	18	32			
	6	17	25	18	32			
	12	14	22	21	13			
	15	19	25	22	18			
	17	21	26	25	32			

Continued on the following page

Super Element	Incidence Matrix							
20	12	21	24	31				
	15	22	23	34				
	21	24	26	31				
	22	23	26	34				
	0	8	20	9	10	24	26	23
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	6	13	22	18	17	21	26	25
	8	11	20	24	21	26		
	18	19	25	22	23	26		
	4	10	24	14	34			
	4	10	23	15	34			
	4	14	22	15	34			
	5	12	21	13	31			
	5	12	24	14	31			
	5	13	22	14	31			
	8	11	21	24	12			
	10	23	26	24	34			
	13	21	26	22	31			
	14	22	26	24	31			
	14	22	26	24	34			
	23	19	18	22	15			
21	18	22	25	32				
	22	25	26	32				
	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	15	19	23	22	25	26		
	6	13	21	17	32			
	6	13	22	18	32			
	6	17	25	18	32			
	13	21	26	22	32			
	15	19	25	22	18			
	17	21	26	25	32			
22	0	8	20	9	10	24	26	23
	1	8	24	12	11	20	26	21
	2	11	21	17	16	20	26	25
	3	9	20	16	19	23	26	25
	4	10	23	15	14	24	26	22
	5	12	24	14	13	21	26	22
	6	13	22	18	17	21	26	25
	7	15	23	19	18	22	26	25

Table C.5: Incidence matrices for manually created tetrahedral Super Elements

Super Element	Incidence Matrix			
0				
1	0	4	5	6
2	0	4	5	6
	1	4	8	7
	6	4	8	7
	6	7	4	5
3	0	4	5	6
	1	4	8	7
	2	5	7	9
	6	5	9	7
	6	5	7	4
	6	7	8	4
	6	7	8	9
4	0	4	5	6
	1	4	8	7
	2	5	7	9
	3	6	8	9
	6	5	9	7
	6	5	7	4
	6	7	8	4
	6	7	8	9

Table C.6: Incidence matrices for manually created pyramidal Super Elements

Super Element	Incidence Matrix			
0				
1	0	5	8	9
2	5	13	9	10
	0	5	13	8
	1	5	13	6
3	0	5	13	8
	2	6	13	7
	5	6	11	9
	7	8	9	11

Continued on the following page

Super Element	Incidence Matrix				
4	5	13	9	10	
	6	13	10	11	
	9	10	11	13	
	0	5	13	8	9
	1	5	13	6	10
	2	6	13	7	11
	9	11	7	8	13
5	5	13	9	10	
	6	13	10	11	
	7	13	11	12	
	8	13	9	12	
	5	8	13	9	
	2	6	13	7	11
	1	5	13	6	10
	3	7	13	8	12
	9	10	11	12	13
6	9	10	11	12	4
	9	10	11	12	4
7	5	9	8	0	
	10	12	9	4	
	10	12	11	4	
	5	10	12	8	9
8	5	13	10	9	
	6	13	10	11	
	13	8	9	12	
	9	10	11	12	4
	0	5	13	8	9
	1	5	13	6	10
	9	10	11	12	13
9	5	13	9	10	
	6	13	10	11	
	7	13	11	12	
	8	13	9	12	
	5	6	13	10	
	8	7	13	12	
	0	5	13	8	9
	2	6	13	7	11
	9	10	11	12	13
	9	10	11	12	4
Continued on the following page					

Super Element	Incidence Matrix				
10	5	13	9	10	
	6	13	10	11	
	7	13	11	12	
	8	13	9	12	
	8	7	13	12	
	0	5	13	8	9
	1	5	13	6	10
	2	6	13	7	11
	9	10	11	12	13
	9	10	11	12	4
11	13	7	12	11	
	13	8	9	12	
	13	6	10	11	
	5	13	9	10	
	9	10	11	12	4
	9	10	11	12	13
	0	5	13	8	9
	1	5	13	6	10
	2	6	13	7	11
	3	7	13	8	12

Table C.7: Incidence matrices for manually created prismatic Super Elements

Super Element	Incidence Matrix				
0					
1	0	10	6	7	
2	6	8	10	17	
	0	6	17	7	10
	6	1	9	17	8
3	0	10	6	7	15 17
	1	6	8	9	17 16
	2	8	10	11	16 15
	6	8	10	17	16 15
4	0	10	6	7	15 17
	7	15	17	3	14 12
5	12	9	17	8	
	7	17	6	8	
	14	12	8	17	
	7	17	14	8	
	1	6	17	9	8
	3	7	17	12	14

Continued on the following page

Super Element	Incidence Matrix					
6	9	12	17	22		
	19	22	9	17		
	22	19	17	15		
	0	6	10	7	17	15
	3	14	12	7	15	17
	1	6	17	9	19	
	6	10	15	17	19	
	12	14	15	17	22	
7	9	16	17	12		
	11	15	16	14		
	1	6	8	9	17	16
	6	8	10	17	16	15
	2	8	10	11	16	15
	3	12	14	7	17	15
	6	10	15	17	7	
	12	14	15	17	16	
8	0	10	6	7	15	17
	2	10	8	11	15	16
	10	6	8	15	17	16
	7	15	17	3	14	12
	9	16	17	4	13	12
	15	16	17	14	13	12
	6	8	16	17	9	
	13	14	15	16	11	
9	0	6	10	7	17	15
	7	17	15	3	12	14
	1	6	8	9	17	16
	9	17	16	4	12	13
	6	8	10	17	16	15
	17	16	15	12	13	14
10	17	16	9	12		
	11	15	16	14		
	0	6	10	7	17	15
	6	1	8	17	9	16
	6	8	10	17	16	15
	8	2	10	16	11	15
	7	15	17	3	14	12
	12	14	15	17	16	

Continued on the following page

Super Element		Incidence Matrix					
11	0	10	6	7	15	17	
	1	6	8	9	17	16	
	2	10	8	11	15	16	
	10	6	8	15	17	16	
	7	15	17	3	14	12	
	9	16	17	4	13	12	
	15	16	17	14	13	12	
	13	14	15	16	11		
12	0	10	6	7	15	17	
	1	6	8	9	17	16	
	2	10	8	11	15	16	
	10	6	8	15	17	16	
	7	15	17	3	14	12	
	9	16	17	4	13	12	
	15	16	17	14	13	12	
	5	13	14	11	16	15	