DIPLOMARBEIT

Modularisierung des Arbeitsablaufs zur Erstellung ordnungsreduzierter thermischer Modelle von Werkzeugmaschinen

Technische Universität Dresden

Fakultät Maschinenwesen

Institut für Mechatronischen Maschinenbau

Professur für Werkzeugmaschinenentwicklung und adaptive Steuerungen

Michael Bauer

Betreuender Hochschullehrer:	Prof. Dr.-Ing. Steffen Ihlenfeldt
Betreuer:	Dipl.-Ing. Steffen Schroeder
Termin der Ausgabe:	Dresden, 01.12.2019
Termin der Abgabe:	Dresden, 01.06.2019

TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Maschinenwesen Institut für Mechatronischen Maschinenbau
Professur für Werkzeugmaschinenentwicklung und adaptive Steuerungen

Diplomarbeit Nr.: 1501

für Bauer, Michael

Matrikelnummer: 3713118

Studiengang: Maschinenbau | Studienrichtung: Kraftfahrzeug- und Schienenfahrzeugtechnik

Thema: **Modularisierung des Arbeitsablaufs zur Erstellung ordnungsreduzierter thermischer Modelle von Werkzeugmaschinen**

Topic: Modularization of the workflow for the creation of order-reduced thermal models of machine tools

Am IMD ist eine Modellierungs- und Simulationstechnologie zur Analyse des thermo-elastischen Verhaltens von Werkzeugmaschinen entwickelt worden. Bei dieser werden FEM-Modelle der Strukturbaugruppen erstellt. Die Modelle werden dann in der Anzahl ihrer Freiheitsgrade reduziert und in ein Netzwerkmodell der bewegten Gesamtmaschine integriert. Die resultierenden Modelle bilden das thermo-elastische Verhalten der Maschine unter Berücksichtigung der Strukturvariabilität ab und können sehr schnell berechnet werden.
Der entwickelte Arbeitsablauf nutzt derzeit kommerzielle CAD- und FEM-Werkzeuge und ist auf deren proprietäre Schnittstellen zugeschnitten. Dies schränkt die Weiterentwicklung, den Anwendungsbereich und die Automatisierbarkeit der Modellerstellung ein. Zudem wird der Einsatz alternativer Werkzeuge erschwert, obwohl gerade Open-Source CAE-Werkzeuge aufgrund ihres offenen und modularen Aufbaus die genannten Einschränkungen nicht aufweisen. Im Rahmen der Diplomarbeit sollen diese Beschränkungen durch eine Modularisierung des Arbeitsablaufes in verallgemeinerbare Teilaufgaben und Schnittstellen aufgehoben werden. Dabei sollen folgende Teilziele erreicht werden:

- Aufteilung des Arbeitsablaufes mit der derzeitigen Werkzeugkette hinsichtlich notwendiger Teilaufgaben und genutzter Funktionalität und Schnittstellen,
- Analyse alternativer CAD- und FEM-Werkzeuge hinsichtlich ihrer Funktionalität und Schnittstellen,
- Erstellen eines Arbeitsablaufes mit modular gruppierten Teilaufgaben und Schnittstellen, um eine Austauschbarkeit einzelner Werkzeuge und eine durchgehende Automatisierbarkeit zu ermöglichen und
- beispielhafte Demonstration des Arbeitsablaufes anhand einfacher Geometriebauteile.

Betreuer:	Steffen Schroeder
Ausgegeben am:	01.12.2019
Einzureichen am:	01.04.2020

Die Diplomprüfungsordnung der Fakultät Maschinenwesen sowie die von der Professur für Werkzeugmaschinenentwicklung und adaptive Steuerungen erlassenen Hinweise zur Anfertigung von Studien- und Diplomarbeiten sind zu beachten.

Prof. Dr.-Ing. Günther Prokop
Studienrichtungsleiter Kraftfahrzeug- und Schienenfahrzeugtechnik

Prof. Dr.-Ing. S. Ihlenfeldt
Betreuender Hochschullehrer

Selbstständigkeitserklärung

Hiermit versichere ich, Michael Bauer, die vorliegende Arbeit selbstständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet.Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, 01.06.2020

Michael Bauer

Zusammenfassung

In dieser Diplomarbeit wird ein bestehender Arbeitsablauf für die Erstellung ordnungsreduzierter thermischer Modelle von Werkzeugmaschinen, der eine schnelle realitätsnahe Simulation ermöglicht, modularisiert und anschließend mit freien Werkzeugen implementiert. Der modularisierte Arbeitsablauf ist flexibel, kann mit verschiedenen Werkzeugen umgesetzt werden und in bestehende Werkzeugketten integriert werden. Der entwickelte Arbeitsablauf unterstützt den Anwender bei der Auswahl und Aufteilung der Schnittstellen des reduzierten Modells und leitet daraus zugleich eine Geometrievereinfachung ab, um kleinteilige und für das thermische Verhalten irrelevante Strukturen zu entfernen. Anschließend erstellt er das reduzierte Modell voll automatisch. Er ist dabei robust und verarbeitet auch fehlerhafte Geometrien. Dies wird anhand einer komplexen Werkzeugmaschinenbaugruppe demonstriert. Zudem wird gezeigt, wie auch das volle FE-Modell im Kontext des reduzierten Modells simuliert werden kann, wodurch dessen Ergebnisse abgesichert werden können.

Abstract

This diploma thesis modularizes an existing workflow for creating reduced thermal models for machine tools, which allows a quick and realistic simulation. The workflow is then implemented using free software tools. The modularized workflow is flexible, can be implemented with different tools and integrated into existing tool chains. The developed workflow supports the user with defining and partitioning the interfaces for the reduced model and at the same time simplifies the geometry to remove small structures that are irrelevant for the thermal behavior. Afterwards, the reduced model is created automatically. The workflow is robust and can deal with imperfect geometries. This is demonstrated with a complex machine tool assembly. In addition, it is shown that the full FE-model can also be simulated in the context of the reduced model, which can be used to check the results.

Inhaltsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AP	Applikationsprotokoll
APDL	Ansys Parametric Design Language
BG	Baugruppe
CAD	Computer-Aided Design/ Rechnerunterstütztes Konstruieren
DGL	Differentialgleichung
FE	Finite Elemente
FEM	Finite-Elemente-Methode
FLOSS	Free/Libre Open-Source-Software
ICN	Inverse Condition Number
IGE	Inverse Gradient Error
MOR	Modellordnungsreduktion/ Modellordnungsreduziert
SFB	Sonderforschungsbereich
STEP	Standard for the Exchange of Product model data
STL	Stereolithographie
TCP	Tool Center Point
UFL	Unified Form Language
VTK	Visualization Toolkit
WZM	Werkzeugmaschine

Symbolverzeichnis

Symbol	Benennung	Einheit
α	Wärmeübergangskoeffizient	$\mathrm{W\,m^{-2}\,K^{-1}}$
$\boldsymbol{A}$	Systemmatrix	
A_K	Koppelflächeninhalt	$\mathrm{m^2}$
$\boldsymbol{B}$	Steuermatrix	
c	spezifische Wärmekapazität	$\mathrm{J\,kg^{-1}\,K^{-1}}$
$\boldsymbol{C}$	Messmatrix	
$\boldsymbol{C}_T$	Wärmekapazitätsmatrix	
Δ	Laplace-Operator	
Δt	Diskreter Zeitschritt	s
$\boldsymbol{f}$	thermischer Lastvektor	
$\Gamma = \partial\Omega$	Rand des Gebiets Ω	
Γ_1	Rand mit Randbedingung 1. Art: Vorgegebene Temperatur (Dirichlet-Randbedingung)	
Γ_2	Rand mit Randbedingung 2. Art: Vorgegebene Wärmestromdichte (Neumann-Randbedingung)	
Γ_3	Rand mit Randbedingung 3. Art: Vorgegebener Wärmeaustausch (Robin-Randbedingung)	
$H^1(\Omega)$	Raum der Funktionen, deren erste verallgemeinerte Ableitung existiert und die ebenfalls Teil des Raumes $L_2(\Omega)$ sind	
j	Anzahl Wärmestrom-Eingänge: Koppelflächen mit Randbedingung 2. Art	
k	Anzahl Temperatur-Eingänge: Koppelflächen mit Randbedingung 3. Art	
$\boldsymbol{K}_T$	Wärmeleitfähigkeitsmatrix	

$\mathcal{K}$	Krylov-Unterraum	
λ	Wärmeleitfähigkeit	$\mathrm{W\,K^{-1}\,m^{-1}}$
$L_2(\Omega)$	Raum der über Ω quadratisch integrierbaren Funktionen	
$\dot{m}$	Massenstrom	$\mathrm{kg\,s^{-1}}$
n	Dimension des FE-MOR-Modells	
N	Anzahl der Knoten in $\Omega \cup \Gamma_2 \cup \Gamma_3$	
∇	Nabla-Operator	
Ω	Dreidimensionales Gebiet	
p_i	FE-Ansatzfunktion	
$\dot{q}$	Wärmestromdichte	$\mathrm{W\,m^{-2}}$
$\dot{Q}$	Wärmestrom	W
ρ	Dichte	$\mathrm{kg\,m^{-3}}$
Σ	Zustandsraummodell	
t	Zeit	s
T	Temperatur	K
$\dot{T} = \frac{\partial T}{\partial t}$	1. Ableitung von T nach der Zeit t	
$\boldsymbol{u}$	Eingangssignale	
$\boldsymbol{u}_\alpha$	Koeffizientenvektor	
v	Testfunktion	
V_0	Raum der Testfunktionen	
V_0	Raum der für die Lösung der Variationsformulierung zulässigen Funktionen	
V_n	Transformationsmatrix	
$\boldsymbol{y}$	Ausgangssignale	

Akzent	Benennung
$\hat{}$	Reduziert

Index	Benennung
0	Startwert
α	Anteil durch Wärmeleitung am Rand Γ
f	Endwert
Γ	bezogen auf den Rand Γ
h	diskrete Näherung
λ	Anteil durch interne Wärmeleitung im Gebiet Ω
Ω	bezogen auf das Gebiet Ω
U	Umgebung

:END:

1 Einleitung

1.1 Motivation

Thermo-elastische Strukturverformungen verursachen bis zu 75 % der am Werkstück auftretenden Bearbeitungsungenauigkeiten (Mayr et al., 2012). Sie können durch konstruktive Maßnahmen oder eine aktive Korrektur während des Betriebs der Werkzeugmaschine (WZM) gemindert werden. Ersteres wird durch die Simulation des thermo-elastischen Verhaltens der WZM unterstützt und zweiteres dadurch erst ermöglicht.

Für die thermische Simulation von WZM ergeben sich zwei Herausforderungen: Um das thermische Verhalten zu erfassen, muss die ganze WZM und ihre Umgebung beachtet werden. Zudem muss dies für den Zeitraum eines ganzen Produktionstags geschehen. Als Antwort auf diese Herausforderungen wurde im Rahmen des SFB/Transregio 96 ein innovatives Verfahren zur thermo-elastischen Simulation von WZM entwickelt. Es ermöglicht, das für die thermische Strukturverformung maßgebliche thermische Verhalten der Strukturbauteile mit der Finite-Elemente-Methode (FEM) zu modellieren und zugleich weit schneller bei gleichbleibender Genauigkeit zu simulieren. Dafür werden zunächst FE-Modelle von der CAD-Geometrie abgeleitet, die dann durch eine Modellordnungsreduktion (MOR) zum FE-MOR-Modell reduziert werden. Die reduzierten Modelle werden mit weiteren Teilmodellen für die übrigen Teilsysteme in ein Gesamtmodell der bewegten WZM integriert. (Galant et al., 2014)

Der entwickelte Arbeitsablauf zum Erstellen der reduzierten Strukturbaugruppenmodelle und der darauf aufbauenden digitalen Blocksimulation der Gesamtmaschine wurde mit bewährten, proprietären FEM- und numerischen Berechnungswerkzeugen umgesetzt (Galant et al., 2014). Bei ersterem war der Export der FE-Matrizen, aufgrund der proprietären Schnittstellen, nur sehr aufwendig realisierbar. Zugleich wird nur ein kleiner Teil des kommerziellen Softwarepakets genutzt.

Die gewachsene Implementierung soll in dieser Diplomarbeit modularisiert werden, um sie anschließend für eine breitere Anwendung, wie beispielsweise einem Industrietransfer, zugänglich zu machen. Durch die Modularisierung kann der Arbeitsablauf anschließend flexibel angewendet und weiterentwickelt werden. Zudem kann er auch in bestehende Werkzeugketten integriert werden.

Die derzeit für den Arbeitsablauf verwendeten Werkzeuge schränken die Weiterentwicklung, die Automatisierbarkeit und den Anwendungsbereich des Verfahrens ein. Es soll deshalb eine alternative Werkzeugkette für das Verfahren untersucht und entwickelt werden. Diese soll auf der Grundlage von freier Software realisiert werden, die die zuvor benannten Beschränkungen nicht aufweist und zugleich modularer und anpassungsfähiger ist. Freie Software, auch Open-Source-Software oder zusammenfassend Free/Libre Open-Source-Software (FLOSS) genannt, gilt als wichtiger Baustein und Innovationsmöglichkeit im Zusammenhang mit der Industrie 4.0 (Aceto et al., 2019). Sie

ist darüber hinaus ein wichtiger Innovationstreiber der IT und genießt beständig wachsende Bedeutung sowohl in der Forschung als auch in der Industrie.

1.2 Zielstellung

Das Ziel dieser Diplomarbeit ist es, den in Galant et al. (2014) beschriebenen Arbeitsablauf zu untersuchen und in seine Teilfunktionen und Schnittstellen zu modularisieren. Dadurch soll erreicht werden, dass der Arbeitsablauf anschließend flexibel und mit austauschbaren Werkzeugen angewendet werden kann. Da sich der Arbeitsablauf für eine automatische Anwendung anbietet, soll er zudem, um diese zu ermöglichen, mit verschiedensten Eingangsdaten robust und fehlertolerant umgehen können.

Der Arbeitsablauf ermöglicht die thermo-elastische Simulation. In dieser Arbeit wird davon nur die thermische Simulation betrachtet, da sie den Kern des Arbeitsablaufs darstellt und die Berechnung der thermischen Verformung auf ihren Ergebnissen als zusätzliches Modul aufbaut.

Um die gestellten Ziele zu erreichen und den Transfer des Arbeitsablaufs zu vereinfachen, bietet sich die Verwendung von freier Software an. Es soll deswegen ausgehend von der Modularisierung untersucht werden, wie die Module und Schnittstellen mit freier Software unter Beachtung der Ziele umgesetzt werden können.

Anhand der Rechercheergebnisse soll der modularisierte Arbeitsablauf dann beispielhaft implementiert werden. Mit der entwickelten Umsetzung soll zudem untersucht werden, wie gut die gesetzten Ziele erfüllt werden.

1.3 Struktur der Arbeit

Die vorliegende Diplomarbeit ist in sechs Kapitel unterteilt.

In Kapitel 2 werden die Grundlagen des Verfahrens aus (Galant et al., 2014) erläutert. Es wird zunächst ausgehend von der Wärmeleitungsgleichung dargelegt, wie das FE-Modell erstellt wird. Anschließend werden dessen Umformung zur Zustandsraumdarstellung und deren Modellordnungsreduktion erläutert. Abschließend wird dargestellt, wie das FE-MOR-Modell in die Gesamtsimulation der bewegten WZM integriert wird.

In Kapitel 3 wird der Arbeitsablauf aufbauend auf den Grundlagen analysiert. Dafür wird zunächst die Anwendersicht auf den Arbeitsablauf untersucht. Anschließend wird der Arbeitsablauf modularisiert, indem er in seine Teilfunktionen und dazugehörige Daten, die Schnittstellen, zerlegt wird. Davon ausgehend wird dann untersucht, wie diese Module in der Referenzimplementierung aus Galant et al. (2014) umgesetzt sind.

In Kapitel 4 wird untersucht, wie die Module mit freier Software umgesetzt werden können. Dafür werden zunächst Kriterien und Schwerpunkte definiert und anschließend die Rechercheergebnisse zu den Modulen präsentiert.

In Kapitel 5 wird der modularisierte Arbeitsablauf beispielhaft mit freier Software implementiert. Die entwickelte Umsetzung wird anhand von zwei Beispielen demonstriert. Das erste ermöglicht die realistische Einschätzung der Fehlertoleranz und Robustheit des Vorgehens. Mit dem zweiten wird die Umsetzung im direkten Vergleich zur Referenzimplementierung auf Korrektheit geprüft. Die Schwerpunkte der Umsetzung sind die Verarbeitung von komplexer WZM-Geometrie und das Aufstellen der FE-Matrizen für diese.

In Kapitel 6 werden die erzielten Ergebnisse zusammengefasst und mögliche darauf aufbauende Forschungsziele diskutiert.

2 Grundlagen

Dieses Kapitel beschreibt das in Galant et al. (2014) vorgestellte Verfahren zum Erstellen ordnungsreduzierter Modelle von WZM-Strukturbaugruppen.

Es beschränkt sich dabei auf die Methodik zum Erstellen ordnungsreduzierter thermischer Modelle, den Kern der Methode. Die Ergebnisse der thermischen Simulation können dann direkt für anschließende Analysen verwendet werden, aber auch für die Berechnung der thermischen Verformung. Diese erfolgt im Arbeitsablauf als eigenes Submodul, welches am Ende dieses Kapitels in Abschnitt 2.3 in das Gesamtmodell eingeordnet wird.

Es wird zunächst ausgehend von der Wärmeleitungsgleichung beschrieben, wie diese mit Hilfe der FEM im Ort diskretisiert wird. Das dadurch erhaltene FE-Modell wird anschließend in seiner Modellordnung reduziert. Dafür wird es zuerst in eine Zustandsraumdarstellung überführt, die dann durch eine Transformation in einen Krylov-Unterraum zum FE-MOR-Modell reduziert wird. Abschließend wird beschrieben, wie die reduzierten Modelle in ein thermisches Gesamtmodell der bewegten WZM integriert werden und welche weiteren Teilmodelle dafür benötigt werden.

2.1 Wärmeleitungsgleichung und FE-Diskretisierung

Die Wärmeleitungsgleichung beschreibt die zeit- und ortsabhängige Temperaturverteilung in einem Medium. Mit ihr lassen sich noch andere Diffusionsprozesse modellieren und sie wird daher auch Diffusionsgleichung genannt. Die instationäre Wärmeleitung in einem Körper (Gebiet) wird durch eine parabolische Differentialgleichung zweiter Ordnung beschrieben. Da die Temperaturverteilung in einem Bereich abhängig ist von den Bedingungen am Rand des Bereichs und von den Bedingungen zum Startzeitpunkt, ergibt sich die Anfangsrandwertaufgabe:

$$c\rho\frac{\partial T}{\partial t} - \lambda\Delta T = \dot{q}_\Omega \qquad \text{auf } \Omega \times (t_0, t_f], \tag{2.1}$$

$$T = T_\Gamma \qquad \text{auf } \Gamma_1 \times (t_0, t_f], \tag{2.2}$$

$$\lambda\frac{\partial T}{\partial N} = \dot{q}_U \qquad \text{auf } \Gamma_2 \times (t_0, t_f], \tag{2.3}$$

$$-\lambda\frac{\partial T}{\partial N} = \alpha(T - T_U) \qquad \text{auf } \Gamma_3 \times (t_0, t_f], \tag{2.4}$$

$$T = T_0 \qquad \text{wenn } t = t_0. \tag{2.5}$$

Gleichung 2.1 beschreibt die instationäre Temperaturverteilung T im Gebiet Ω für das Zeitintervall $(t_0, t_f]$. c ist die spezifische Wärmekapazität, ρ die Dichte, λ die Wärmeleitzahl und $\dot{q}_\Omega$ interne

Wärmequellen. Δ ist der Laplace-Operator. Für ein räumliches, dreidimensionales Temperaturfeld entspricht er der Summe der zweiten Ableitung in jeder Raumrichtung [1]. Dies beschreibt, dass die zeitliche Änderung der Temperatur an einem Ort durch den Temperaturunterschied zu seiner Umgebung bewirkt wird.

Am Rand des Gebiets werden drei verschiedene Arten von Randbedingungen unterschieden:

1. Gleichung 2.2 gibt eine Temperatur T_Γ am Rand Γ_1 des Gebiets Ω fest vor (1. Art: Dirichlet-Randbedingung)
2. Gleichung 2.3 gibt eine Wärmestromdichte $\dot{q}_U$ in Richtung der Flächennormalen am Rand Γ_2 vor (2. Art: Neumann-Randbedingung)
3. Gleichung 2.4 beschreibt den Wärmeaustausch mit der Umgebung am Rand Γ_3 (3. Art: Gemischte oder Robin-Randbedingung). Der Wärmeaustausch ist abhängig von dem Wärmeaustauschkoeffizienten α und der Temperaturdifferenz zur Umgebungstemperatur T_U. Diese Randbedingung kann sowohl Konvektion als auch Wärmeleitung beschreiben.

$\frac{\partial}{\partial N}$ ist die Richtungsableitung in Richtung der äußeren Einheitsnormalen am Rand Γ.

Die Temperaturverteilung T_0 zum Startzeitpunkt t_0 wird durch die Anfangsbedingung in Gleichung 2.5 vorgegeben.

(Jung et al., 2013, S. 10 f.)

FE-Diskretisierung

Um die parabolische Anfangsrandwertaufgabe zweiter Ordnung zu lösen, muss eine partielle Differentialgleichung zweiter Ordnung über dem Gebiet Ω während des Zeitintervalls $(t_0, t_f]$ gelöst werden. Dabei muss die Lösung der Anfangsbedingung und den Randbedingungen genügen. Die Lösung kann nur in wenigen Spezialfällen analytisch erfolgen. Stattdessen wird das durch die DGL beschriebene kontinuierliche Feldproblem in ein endlichdimensionales Ersatzproblem überführt, was Diskretisierung genannt wird. Gemäß dem Stand der Technik geschieht das für die Wärmeleitgleichung mit der FEM. Das Ersatzproblem wird anschließend mittels spezieller Lösungsverfahren rechnergestützt gelöst. (Jung et al., 2013, S. 1 ff.)

Als Ausgangspunkt der FE-Diskretisierung wird die Anfangsrandwertaufgabe zunächst in die Variationsformulierung, auch schwache Formulierung genannt, überführt. Dafür wird sie mit einer Testfunktion $v \in V_0$ multipliziert und anschließend über das Gebiet Ω partiell integriert. Die Testfunktion muss so gewählt werden, dass sie über Ω quadratisch integrierbar ist und ihre erste verallgemeinerte Ableitung existiert. Der Raum L_2 enthält alle Funktionen, die das erste Kriterium erfüllen, und der Raum H^1 enthält alle Funktionen aus L_2, die außerdem das zweite Kriterium erfüllen. Weiterhin muss die Testfunktion auf dem Randstück Γ_1 gleich Null sein, da hier die Temperatur T_Γ bekannt ist. Der Raum der Testfunktion ist demnach:

[1] $\Delta T = \nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2}$

$$V_0 = \{v \in H_1(\Omega) : v = 0 \text{ auf } \Gamma_1\} \tag{2.6}$$

Wird Gleichung 2.1 mit v multipliziert und anschließend partiell integriert, erhält man die Variationsformulierung:

$$\int_\Omega c\rho \frac{\partial T}{\partial t} v \, \mathrm{d}\Omega + \int_\Omega \lambda \nabla T \nabla v \, \mathrm{d}\Omega - \int_\Gamma \frac{\partial T}{\partial N} v \, \mathrm{d}\Gamma = \int_\Omega \dot{q}_\Omega v \, \mathrm{d}\Omega \tag{2.7}$$

In den durch die partielle Integration entstandenen Term werden nun noch die Randbedingungen zweiter (Gleichung 2.3) und dritter Art (Gleichung 2.4) eingesetzt. Anschließend wird die entstandene Gleichung so sortiert, dass auf der linken Seite nur Terme mit der Unbekannten T stehen, während alle anderen auf der rechten Seite stehen. Die linke Seite wird nun auch Bilinearform und die rechte Seite Linearform genannt. Die Variationsformulierung des Randwertproblems ist nun vollständig:

$$\int_\Omega c\rho \frac{\partial T}{\partial t} v \, \mathrm{d}\Omega + \int_\Omega \lambda \nabla T \nabla v \, \mathrm{d}\Omega + \int_{\Gamma_3} \alpha T v \, \mathrm{d}\Gamma_3 = \int_\Omega \dot{q}_\Omega v \, \mathrm{d}\Omega + \int_{\Gamma_2} \dot{q}_U v \, \mathrm{d}\Gamma_2 + \int_{\Gamma_3} \alpha T_U v \, \mathrm{d}\Gamma_3 \tag{2.8}$$

Abschließend wird unter Beachtung der Randbedingungen erster Art der Raum der zulässigen Funktionen, innerhalb derer die Lösung T gesucht wird, festgelegt:

$$V_D = \{T \in H^1(\Omega) : T = T_\Gamma \text{ auf } \Gamma_1\} \tag{2.9}$$

(Jung et al., 2013, S. 205 ff.)

Die Variationsformulierung 2.8 wird nun räumlich diskretisiert und damit in ein System gewöhnlicher Differentialgleichungen, das FE-Gleichungssystem, welches die Variationsformulierung annähert, überführt. Dafür wird das Gebiet Ω in nicht-überlappende Teilgebiete, die finiten Elemente, beispielsweise Tetraeder oder Hexaeder, zerlegt. Dieses Vorgehen wird Vernetzen genannt und ergibt das FE-Netz. Jedem Element werden nun finite Polynom-Funktionen zugewiesen, die nur in einzelnen Punkten des Elementes, den Knoten des FE-Netzes, von Null verschieden ist. Dies sind die Ansatzfunktionen $\boldsymbol{p}$. Für den betrachteten Arbeitsablauf werden Tetraeder-Elemente und lineare Ansatzfunktionen verwendet (Großmann et al., 2011). Die linearen Ansatzfunktionen unterscheiden sich nur in den Eckpunkten der Tetraeder von Null. Die Eckpunkte sind somit die Knoten des FE-Netzes und ihre Anzahl N bestimmt die Größe des FE-Gleichungssystems.

Mit dem FE-Netz und den Ansatzfunktionen wird die Variationsformulierung 2.8 in das ortsdiskrete FE-Gleichungssystems überführt. Dies geschieht, indem die unendlichdimensionalen Räume V_D und V_0 (vgl. Gl. 2.9 u. 2.6) durch endlichdimensionale Räume V_{Dh} und V_{0h} der Dimension N ersetzt werden. Dafür wird zunächst T durch eine Linearkombination der Ansatzfunktionen angenähert:

$$T \approx \sum_{i=1}^{N} T_{hi} p_i \tag{2.10}$$

T_h ist die N-dimensionale, diskrete Näherungslösung für T. Die Testfunktion v wird ebenfalls diskretisiert, indem für sie auch die finiten Ansatzfunktionen verwendet werden: $v_{hi} = p_i$. Da T und v ab jetzt nur noch diskret verwendet werden, wird das Subskript $_h$ im Folgenden der Übersicht halber weggelassen: $\boldsymbol{T}$ entspricht ab hier der diskreten Temperaturverteilung und $\boldsymbol{v}$ dem Vektor der finiten Testfunktionen[2]. Das FE-Gleichungssystems in Matrixform der instationären Wärmeleitung ergibt sich zu:

$$\boldsymbol{C}_T \dot{\boldsymbol{T}} + \boldsymbol{K}_T \boldsymbol{T} = \boldsymbol{f} \tag{2.11}$$

$$C_{Tij} = \int_\Omega c\rho p_i v_j \,\mathrm{d}\Omega \tag{2.12}$$

$$K_{Tij} = \int_\Omega \lambda \nabla p_i \nabla v_j \,\mathrm{d}\Omega + \int_{\Gamma_3} \alpha p_i v_j \,\mathrm{d}\Gamma_3 \tag{2.13}$$

$$f_i = \int_\Omega \dot{q}_\Omega v_i \,\mathrm{d}\Omega + \int_{\Gamma_2} \dot{q}_U v_i \,\mathrm{d}\Gamma_2 + \int_{\Gamma_3} \alpha T_U v_i \,\mathrm{d}\Gamma_3 \tag{2.14}$$

$\boldsymbol{C}_T$ ist die Wärmekapazitätsmatrix und $\boldsymbol{K}_T$ die Wärmeleitfähigkeitsmatrix. Beide werden in Anlehnung an die Strukturmechanik zuweilen auch thermische Massen- und Steifigkeitsmatrix genannt. $\boldsymbol{f}$ ist der thermische Lastvektor und fasst die Wärmestromdichten der rechten Gleichungsseite zusammen. Gleichungen 2.12, 2.13 und 2.14 ensprechen größtenteils noch der Variationsformulierung in Gleichung 2.8. Jedoch sind $\boldsymbol{T}$ und $\boldsymbol{v}$ nun diskrekt und die Ansatzfunktionen sind einmultipliziert.

Die Anzahl N der Ansatzfunktionen und damit der Knoten des FE-Netzes bestimmt die Form der Matrizen. Sie ergibt sich nach dem Ausmultiplizieren der Faktoren und die Werte der Matrizen durch die anschließende Integration. $\boldsymbol{C}_T$ und $\boldsymbol{K}_T$ sind quadratische Matrizen mit N Zeilen. Sie sind dünn und nur im Bereich der Diagonalen besetzt, denn nur Kombinationen von Ansatzfunktionen, die im Netz nahe beieinander liegen, führen zu einem Wert ungleich Null in der Matrix.

Um mit dem ortsdiskreten FE-Gleichungssystem 2.11 eine FE-Simulation durchführen zu können, muss es noch in der Zeit diskretisiert werden. Das resultierende lineare Gleichungssystem kann dann iterativ gelöst werden. Außerdem wird das FE-Netz der zu simulierenden Geometrie, die dazugehörigen Materialparameter c, ρ und λ und passende Ansätze für die Randbedingungen benötigt. Das durch die FE-Matrizen beschriebene Modell wird im Folgenden FE-Modell genannt.

(Jung et al., 2013, S. 230 ff.)

[2] Bei $\boldsymbol{T}$ handelt es sich um einen Vektor, obwohl der Großbuchstabe sonst eine Matrix bezeichnet. Dies wurde so gewählt, um die Temperatur gemäß der etablierten Konvention T zu nennen.

2.2 Modellordnungsreduktion

Bei der thermischen Modellerierung von WZM-Strukturbaugruppen entstehen komplexe FE-Modelle. Ihre Berechnung ist aufgrund der großen Zahl an Freiheitsgraden sehr aufwendig. Um eine weit schnellere Simulation zu ermöglichen, wird das FE-Modell zum MOR-FE-Modell reduziert. MOR steht für Modellordnungsreduktion und beschreibt ein Verfahren, mit dem die Freiheitsgrade des (numerischen) Modells reduziert werden. Das MOR-Verfahren aus Galant et al. (2014) verwendet die Krylov-MOR-Methode, die sich gut für thermische Modelle eignet. Angewendet auf die Simulation von WZM-Strukturbaugruppen werden damit Rechenzeitersparnisse von bis zu drei Größenordnungen erzielt, während die simulierten Temperaturen nur geringfügig von denen des vollen FE-Modells abweichen. Dies ist möglich, da das thermische Verhalten dieser Modelle als linear angenommen werden kann.

Um von dem FE-Modell aus Gleichung 2.11 ein MOR-FE-Modell zu erstellen sind drei Schritte notwendig:

1. Das FE-Modell wird zunächst in ein Zustandsraummodell überführt. Hierbei werden seine Aus- und Eingänge $\boldsymbol{u}$ und $\boldsymbol{y}$ festgelegt. Die Eingänge sind Wärmeströme oder Temperaturen und dazugehörige Wärmeübergangskoeffizienten, die auf Koppelflächen bezogen werden und als Randbedingungen wirken. Die Ausgänge sind die gemittelten Temperaturen der Koppelflächen.
2. Die Dimension n des reduzierten Modells wird festgelegt und die Transformationsmatrix $\boldsymbol{V}_n$ aufgestellt. An dieser Stelle muss das FE-Modell mit ρ, c und λ parametriert werden. Denn nur so kann das reduzierte Modell das Verhalten des FE-Modells abbilden.
3. Das FE-Modell wird nun zum MOR-FE-Modell reduziert. Dieses besteht aus $j + k$ reduzierten Modellen. Je einem pro Eingang.

(Galant et al., 2014)

Zustandsraumdarstellung

Die Zustandsraumdarstellung wird zur Analyse und Simulation von dynamischen Systemen verwendet. Sie wird auch Eingangs-Ausgangs-Darstellung genannt. Diese Bezeichnung ist auch im Kontext des betrachteten Arbeitsablaufs beschreibend: Denn hier bekommt das FE-Modell und damit auch das MOR-FE-Modell seine Ein- und Ausgänge.

Als erstes wird Gleichung 2.11 präzisiert, indem angenommen wird, dass weder innere Wärmequellen wirken noch ein Wärmeaustausch über Strahlung stattfindet. Es werden auch keine Temperaturen am Rand als Randbedingungen 1. Art fest vorgegeben. Außerdem wird $\boldsymbol{K}_T$ so aufgeteilt, dass der Wärmeübergangskoeffizient α als Eingangsgröße für das Modell verwendet werden kann. Es ergibt sich eine Wärmeleitfähigkeitsmatrix $\boldsymbol{K}_{T\lambda}$ für die Wärmeleitung im Modell und eine zweite

Matrix $\boldsymbol{K}_{T\alpha}$ für die Wärmeleitung am Rand des Modells. Der thermische Lastvektor $\boldsymbol{f}$ wird ebenfalls aufgeteilt, so dass beide Arten von Randbedingungen seperat behandelt werden können.

$$\boldsymbol{C}_T\dot{\boldsymbol{T}} + \underbrace{(\boldsymbol{K}_{T\lambda} + \boldsymbol{K}_{T\alpha}\boldsymbol{\alpha})}_{\boldsymbol{K}_T}\boldsymbol{T} = \boldsymbol{f}_2\dot{\boldsymbol{q}}_U + \boldsymbol{f}_3\boldsymbol{\alpha}\boldsymbol{T}_U \tag{2.15}$$

Gleichung 2.15 wird nun in die explizite Form gebracht, indem von rechts und links mit $\sqrt{\boldsymbol{C}_T}^{-1}$ multipliziert wird. $\boldsymbol{C}_T$ muss zuvor diagonalisiert werden, denn sonst würde die dünn besetzte und potentiell sehr große Matrix $\boldsymbol{C}_T$ durch die Inversion vollbesetzt werden. (Galant, 2014)

Dadurch entsteht die Systemmatrix $\boldsymbol{A}$, die analog zu $\boldsymbol{K}_T$ in zwei Matrizen aufgeteilt wird. $\boldsymbol{A}$ enthält nun die vollständige Beschreibung der Geometrie und der Materialparameter.

$$\dot{\boldsymbol{T}} + \underbrace{(\boldsymbol{A}_\lambda + \bar{\boldsymbol{A}}_\alpha\boldsymbol{\alpha})}_{\boldsymbol{A}}\boldsymbol{T} = \bar{\boldsymbol{f}}_2\dot{\boldsymbol{q}}_U + \bar{\boldsymbol{f}}_3\boldsymbol{\alpha}\boldsymbol{T}_U \tag{2.16}$$

Als nächstes werden die Eingangsgrößen $\dot{\boldsymbol{q}}_U$ und $\boldsymbol{\alpha}\boldsymbol{T}_U$ auf Koppelflächen bezogen und damit kondensiert. Die Koppelflächen sind zusammenhängende, nicht-überlappende Flächen am Rand des Modells, auf die die Eingangsgrößen wirken. Ihre Trennstellen sind durch die Knoten des FE-Netzes vorgegeben.

Dies geschieht aufgrund der Annahme, dass in der Simulationspraxis die Lasten zumeist nicht auf einzelne Knoten des FE-Modells bezogen werden, sondern flächenbezogen auf Knotenmengen. Dadurch ist die Anzahl der verschiedenen Lasten n wesentlich kleiner als die Dimension N des FE-Modells.

Die Lasten werden dafür im Eingangssignalvektor $\boldsymbol{u}$ zusammengefasst. $u_1, \ldots, u_j$ sind die auf die Koppelflächeninhalte $A_{K1}, \ldots, A_{Kj}$ bezogenen Wärmeflüsse $\dot{Q}_1, \ldots, \dot{Q}_j$, die als Randbedingungen 2. Art auf je eine Koppelfläche wirken. $u_j, \ldots, u_{j+k}$ sind die Umgebungstemperaturen $T_{Uj}, \ldots, T_{Uj+k}$ mit den dazugehörigen Wärmeübergangskoeffizienten $\alpha_j, \ldots, \alpha_{j+k}$, die als Randbedingungen 3. Art ebenfalls auf je eine Koppelfläche wirken. Die Lastvektoren $\dot{\boldsymbol{q}}_U$ und $\boldsymbol{\alpha}\boldsymbol{T}_U$ werden folgendermaßen durch $\boldsymbol{u}$ ausgedrückt:

$$\dot{\boldsymbol{q}}_U = \underbrace{\begin{bmatrix} \boldsymbol{b}_1 & \cdots & \boldsymbol{b}_j \end{bmatrix}}_{\boldsymbol{B}_2} \cdot \begin{bmatrix} u_1 = \frac{\dot{Q}_1}{A_{Ki}} \\ \vdots \\ u_j \end{bmatrix} \tag{2.17}$$

$$\boldsymbol{\alpha}\boldsymbol{T}_U = \underbrace{\begin{bmatrix} \boldsymbol{b}_{j+1} & \cdots & \boldsymbol{b}_{j+k} \end{bmatrix}}_{\boldsymbol{B}_3} \cdot \begin{bmatrix} u_{j+1} = \alpha_1 T_{U1} \\ \vdots \\ u_{j+k} \end{bmatrix} \tag{2.18}$$

Die Vektoren $b_1, \ldots, b_k$ sind so definiert, dass sie auf dem Knoten i gleich 1 sind, falls u_i auf diesen Knoten wirkt, und sonst gleich 0 sind. Sie werden zu den Matrizen $\boldsymbol{B}_2$ und $\boldsymbol{B}_3$ zusammengefasst. In die rechte Seite von Gleichung 2.16 eingesetzt ergibt das die Steuermatrix $\boldsymbol{B}$ des Zustandsraummodells:

$$\bar{\boldsymbol{f}}_2\dot{\boldsymbol{q}}_U + \bar{\boldsymbol{f}}_3\boldsymbol{\alpha}\boldsymbol{T}_U = \begin{bmatrix} \bar{\boldsymbol{f}}_2\boldsymbol{B}_2 & \bar{\boldsymbol{f}}_3\boldsymbol{B}_3 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ \vdots \\ u_{j+k} \end{bmatrix} = \boldsymbol{B}\boldsymbol{u} \tag{2.19}$$

Da $\boldsymbol{\alpha}$ auch auf der linken Seite von Gleichung 2.16 vorkommt, werden k Matrizen $\boldsymbol{A}_\alpha$ benötigt, je eine pro Eingang mit Umgebungstemperatur. Die zu den Eingängen $u_j, \ldots, u_{j+k}$ gehörenden Wärmeübergangskoeffizienten $\alpha_j, \ldots, \alpha_{j+k}$ werden im zweiten Eingangsvektor, dem Koeffizientenvektor $\boldsymbol{u}_\alpha$, zusammengefasst:

$$\bar{\boldsymbol{A}}_\alpha\boldsymbol{\alpha} = \sum_{i=1}^{k} \boldsymbol{A}_{\alpha i}u_{\alpha i} = \boldsymbol{A}_{\alpha i}u_{\alpha i} \qquad \text{mit} \quad \boldsymbol{A}_\alpha = \begin{bmatrix} \boldsymbol{A}_{\alpha 1} & \cdots & \boldsymbol{A}_{\alpha k} \end{bmatrix} \tag{2.20}$$

$$\text{und} \quad \boldsymbol{u}_\alpha = \begin{bmatrix} \alpha_1 & \cdots & \alpha_k \end{bmatrix} \tag{2.21}$$

$\boldsymbol{A}_\alpha$ ist ein Tensor dritter Stufe, der die einzelnen $\boldsymbol{A}_{\alpha i}$ pro Koppelfläche enthält.

Nun werden auch die Ausgänge $\boldsymbol{y}$ definiert und ebenfalls auf Koppelflächen bezogen. Dabei wird eine ähnliche Annahme wie bei den Eingängen getroffen: In der Simulationspraxis sind nicht die einzelnen Temperaturen auf den Knoten relevant, sondern Mittelwerte der Temperaturen auf den Koppelflächen. Diese können die selben wie für die Ausgänge sein oder es werden andere gewählt. Es kann also die Anzahl der Eingänge gleich der Anzahl der Ausgänge $j + k$ gesetzt werden oder eine davon verschiedene Anzahl gewählt werden.

Die Verbindung der Ausgänge mit den FE-Knoten wird in der Messmatrix $\boldsymbol{C}$ festgelegt. $\boldsymbol{C}$ hat die selbe Anzahl an Zeilen N wie $\boldsymbol{B}$ und ihre Spaltenzahl entspricht der Anzahl an Eingängen. Falls $\boldsymbol{C}$ eine Einheitsmatrix ist, dann entspricht $\boldsymbol{y}$ dem Temperaturvektor $\boldsymbol{T}$.

Damit ergibt sich die Zustandsraumdarstellung von Gleichung 2.15:

$$\begin{bmatrix} u_1 \\ \vdots \\ u_j \\ \vdots \\ u_{j+k} \\ u_{\alpha 1} \\ \vdots \\ u_{\alpha k} \end{bmatrix} \longrightarrow \quad \Sigma : \dot{T} + \overbrace{(A_\lambda + A_{\alpha i}\, u_{\alpha i})}^{A} T = Bu \qquad y = C^T T \quad \longrightarrow \begin{bmatrix} y_1 \\ \vdots \\ y_{j+k} \end{bmatrix} \tag{2.22}$$

Das Zustandsraummodell Σ besitzt $j + 2k$ Eingänge, wovon die ersten j Wärmestrom-Eingänge und die letzten $2k$ Temperatur-Eingänge sind. Von diesen sind nur die ersten k Eingänge ebenfalls Eingänge des Zustandsraummodells. Die restlichen k Eingänge fließen in die Systemmatrix $\boldsymbol{A}$ ein und sind in der Definition der Zustandsraumdarstellung nicht vorgesehen. Für die betrachteten thermischen Modelle muss jedoch ein mit der Zeit veränderlicher Wärmeübergangskoeffizient realisiert werden, woraus sich der Koeffizientenvektor $\boldsymbol{u}_\alpha$ ergibt. Das Zustandsraummodell Σ hat weiterhin $j + k$ Ausgänge, einen für jeden Ausgang. Die Koppelflächen der Ein- und Ausgänge müssen, wie zuvor dargelegt, nicht übereinstimmen. Es wird aber im Folgenden angenommen, dass sie gleich sind.

$\boldsymbol{A}$ ist die Systemmatrix, $\boldsymbol{B}$ die Steuermatrix und $\boldsymbol{C}$ die Messmatrix des Zustandsraummodells Σ. Die Systemmatrix $\boldsymbol{A}$ ist aufgeteilt in einen Anteil für die Wärmeleitfähigkeit: die Matrix $\boldsymbol{A}_\lambda$, und einen Anteil für die Wärmeübergangskoeffizienten: den Tensor dritter Stufe $\boldsymbol{A}_\alpha$. Der Temperaturvektor $\boldsymbol{T}$ ist der Zustandsvektor des Systems und die Dimension des Systems ist N, die Anzahl der FE-Knoten.

Das System Σ in Gleichung 2.22 ist das FE-Modell mit auf die Koppelflächen reduzierten Ein- und Ausgängen. Dieses wird als nächstes zum FE-MOR-Modell reduziert.

(Galant et al., 2014; Großmann et al., 2012)

MOR

Die MOR-Methode nähert den Zustandsvektor $\boldsymbol{T}$ in einem Krylov-Unterraum an, indem $\boldsymbol{T}$ in diesen Unterraum projiziert wird. Der Unterraum sei $\mathcal{K}_n \subset \mathrm{R}^N$ mit $n << N$ und besitzt n orthonormale Basisvektoren. Diese sind die Spaltenvektoren der Transformationsmatrix $\boldsymbol{V}_n$. Die Approximation von $\boldsymbol{T}$ ergibt sich dann zu:

$$\boldsymbol{T} = \boldsymbol{V}_n \hat{\boldsymbol{T}} \tag{2.23}$$

Die Transformationsmatrix V_n projiziert den reduzierten Zustandsvektor $\hat{T}$ also auf den orginalen Zustandsvektor T.

Die restlichen Matrizen aus Gleichung 2.22 werden wie folgt reduziert:

$$\begin{aligned} \hat{A}_\lambda &= V_n^T A_\lambda V_n \; ; \qquad \hat{A}_\alpha = \begin{bmatrix} V_n^T A_{\alpha 1} V_n & \cdots & V_n^T A_{\alpha k} V_n \end{bmatrix} \\ \hat{B} &= V_n^T B \; ; \qquad \hat{C} = V_n^T C \end{aligned} \tag{2.24}$$

Auch hier kann das Produkt mit V_n wieder als Projektion aus dem Unterraum $\mathcal{K}$ und das Produkt mit V_n^T als Projektion in diesen Unterraum aufgefasst werden. Der Akzent ^bezeichnet die reduzierten Matrizen und Vektoren.

Nachdem nun die Transformationsvorschrift gegeben ist, wird als nächstes das Verfahren zum Berechnen der Transformationsmatrix V_n vorgestellt. Der Ansatz für die MOR ist, dass das System Σ als Kombination linearer Teilmodelle ausgedrückt werden kann. Σ wird in ein Teilmodell pro Eingangsgröße in u aufgeteilt. Es ergeben sich somit $j+k$ Teilmodelle. Diese werden einzeln gelöst und zur Lösung y superpositioniert. Hierfür wird die Steuermatrix B in ihre $j+k$ Spaltenvektoren zerlegt, von denen einer je Teilmodell verwendet wird. Der zweite Eingangsvektor u_α wird nicht aufgeteilt und fließt ganz in jedes Teilmodell ein.

$$\Sigma : \begin{cases} \Sigma_1 : \dot{T}_1 + (A_\lambda + A_{\alpha i}\, u_{\alpha i})\, T_1 = b_1 u_1 \; ; & y_1 = C^T T_1 \\ \qquad\qquad \cdots & \cdots \\ \Sigma_{j+k} : \dot{T}_{j+k} + \underbrace{(A_\lambda + A_{\alpha i}\, u_{\alpha i})}_{A}\, T_{j+k} = b_{j+k} u_{j+k} \; ; & y_{j+k} = C^T T_{j+k} \\ y = y_1 + \cdots + y_{j+k} \end{cases} \tag{2.25}$$

Die $j+k$ Gleichungen 2.25 sind Teilsyteme vom Typ Mono-Eingangssignal. Für Systeme von diesem Typ kann die Transformationsmatrix V_n und damit der Krylov-Unterraum $\mathcal{K}$ mit dem klassischen Arnoldiprozess ermittelt werden. Dieser arbeitet iterativ und benötigt die invertierte Systemmatrix A^{-1}, den Spaltenvektor b_i des jeweiligen Teilsystems und die Dimension n des reduzierten Modells als Eingangsgrößen. Mit den Transformationsvorschriften aus Gleichung 2.24 werden anschließend die reduzierten Matrizen $\hat{A}_{\lambda i}$, $\hat{A}_{\alpha i}$, $\hat{C}_i$ und die reduzierten Vektoren $\hat{b}_i$ berechnet:

$$\begin{aligned} A^{-1}, b_1, n \quad &\xrightarrow{\text{Arnoldi}} \quad V_{n1} \quad \xrightarrow{2.24} \quad \hat{A}_{\lambda 1}, \hat{A}_{\alpha 1}, \hat{b}_1, \hat{C}_1 \\ &\cdots \qquad\qquad \cdots \qquad\qquad \cdots \\ A^{-1}, b_{j+k}, n \quad &\xrightarrow{\text{Arnoldi}} \quad V_{n,j+k} \quad \xrightarrow{2.24} \quad \hat{A}_{\lambda,j+k}, \hat{A}_{\alpha,j+k}, \hat{b}_{j+k}, \hat{C}_{j+k} \end{aligned} \tag{2.26}$$

Es enstehen also $j + k$ reduzierte Teilmodelle. Diese sind das reduzierte FE-MOR-Modell $\hat{\Sigma}$. Der Ausgangsvektor y ergibt sich aus der Summe der Ausgangsvektoren y_i der Teilmodelle.

$$\begin{bmatrix} u_1 \\ \vdots \\ u_j \\ \vdots \\ u_{j+k} \\ u_{\alpha 1} \\ \vdots \\ u_{\alpha k} \end{bmatrix} \longrightarrow \quad \hat{\Sigma} : \begin{cases} \hat{\Sigma}_1 : \dot{\hat{T}}_1 + (\hat{A}_\lambda + \hat{A}_{\alpha i}\, u_{\alpha i})\, \hat{T}_1 = \hat{b}_1 u_1 \; ; & \hat{y}_1 = \hat{C}^T \hat{T}_1 \\ \qquad \cdots & \cdots \\ \hat{\Sigma}_{j+k} : \dot{\hat{T}}_{j+k} + (\hat{A}_\lambda + \hat{A}_{\alpha i}\, u_{\alpha i}) \hat{T}_{j+k} = \hat{b}_{j+k} u_{j+k} \; ; & \hat{y}_{j+k} = \hat{C}^T \hat{T}_{j+k} \\ \hat{y} = \hat{y}_1 + \cdots + \hat{y}_{j+k} \end{cases} \longrightarrow \begin{bmatrix} y_1 \\ \vdots \\ y_{j+k} \end{bmatrix} \tag{2.27}$$

(Galant et al., 2014; Großmann et al., 2012)

Um den reduzierten Temperaturvektor $\hat{T}$ zum vollen Temperaturvektor T zu transformieren und die Anfangsbedingung, die Temperaturverteilung zum Startzeitpunkt, T_0 zu reduzieren, werden zwei weitere Transformationsmatrizen benötigt. Dafür kann V_n nicht direkt verwendet werden. Denn in beiden Fällen muss die beidseitige Multiplikation mit $\sqrt{C_T}^{-1}$, durch die Gleichung 2.15 in die explizite Form 2.16 gebracht wurde, beachtet werden. Die beiden Transformationsmatrizen und die dazugehörigen Transformationsvorschriften ergeben sich zu:

$$V_{n0} = V_n^T \sqrt{C_T}\,; \qquad \hat{T}_0 = V_{n0}\, T_0 \tag{2.28}$$

$$V_{n1} = \sqrt{C_T}^{-1} V_n\,; \qquad T = V_{n1}\, \hat{T} \tag{2.29}$$

(Galant, 2014)

Die Genauigkeit des reduzierten Modells kann nicht ohne Versuch bestimmt werden. Nach Galant et al. (2014) konnte sie jedoch an verschiedenen Versuchen mit WZM-Modellen erfolgreich überprüft werden. Die Autoren kommen zu folgendem Fazit:

- Das reduzierte FE-MOR-Modell $\hat{\Sigma}$ hat eine wesentlich kleinere Dimension als das FE-Modell, wenn $n << N$ und kann deswegen weit schneller berechnet werden.
- Das reduzierte FE-MOR-Modell $\hat{\Sigma}$ gibt die Struktur des Ausgangssystems Σ wieder.
- Das reduzierte FE-MOR-Modell $\hat{\Sigma}$ wird unabhängig von den Eingangssignalen $\boldsymbol{u}$ und $\boldsymbol{\alpha}$ erstellt.

2.3 Thermisches Gesamtmodell

Da das FE-MOR-Modell nun in einer Eingangs-Ausgangs-Darstellung vorliegt, kann es mit weiteren Modellen dieser Art zu einem thermischen Gesamtmodell der WZM kombiniert werden. Die Teil-

modelle sind über ihre Ein- und Ausgänge miteinander verbunden. Sie können verschiedene physikalische Phänomene simulieren und dafür auch verschiedene Modellierverfahren nutzen. Es ist damit möglich, ein Gesamtmodell des komplexen thermischen Verhaltens einer WZM zu erhalten und es aufgrund der Modularität der Teilmodelle während des Entwicklungsprozesses anzupassen und Varianten zu bilden.

Modelle dieser Art, in denen die einzelnen Teilmodelle als Blöcke abstrahiert werden können, werden auch Blocksimulation genannt. Um sie als digitale Blocksimulation implementieren zu können, müssen die einzelnen Teilmodelle oder auch Blöcke in der Zeit diskretisiert sein. Sie können dann auch mit unterschiedlichen Zeitschritten Δt und somit unterschiedlichen Auflösungen simuliert werden. Damit kann dem unterschiedlichen zeitlichen Verhalten der verschiedenen physikalischen Phänomene entsprochen werden, während zugleich der Rechenaufwand daran angepasst wird. Um Blöcke zu verbinden, die mit unterschiedlichen Zeitschritten berechnet werden, wird zwischen diesen interpoliert. Es besteht weiterhin die Möglichkeit, eine automatische Zeitschrittsteuerung zu verwenden.

(Galant et al., 2014; Schroeder; Galant et al., 2018)

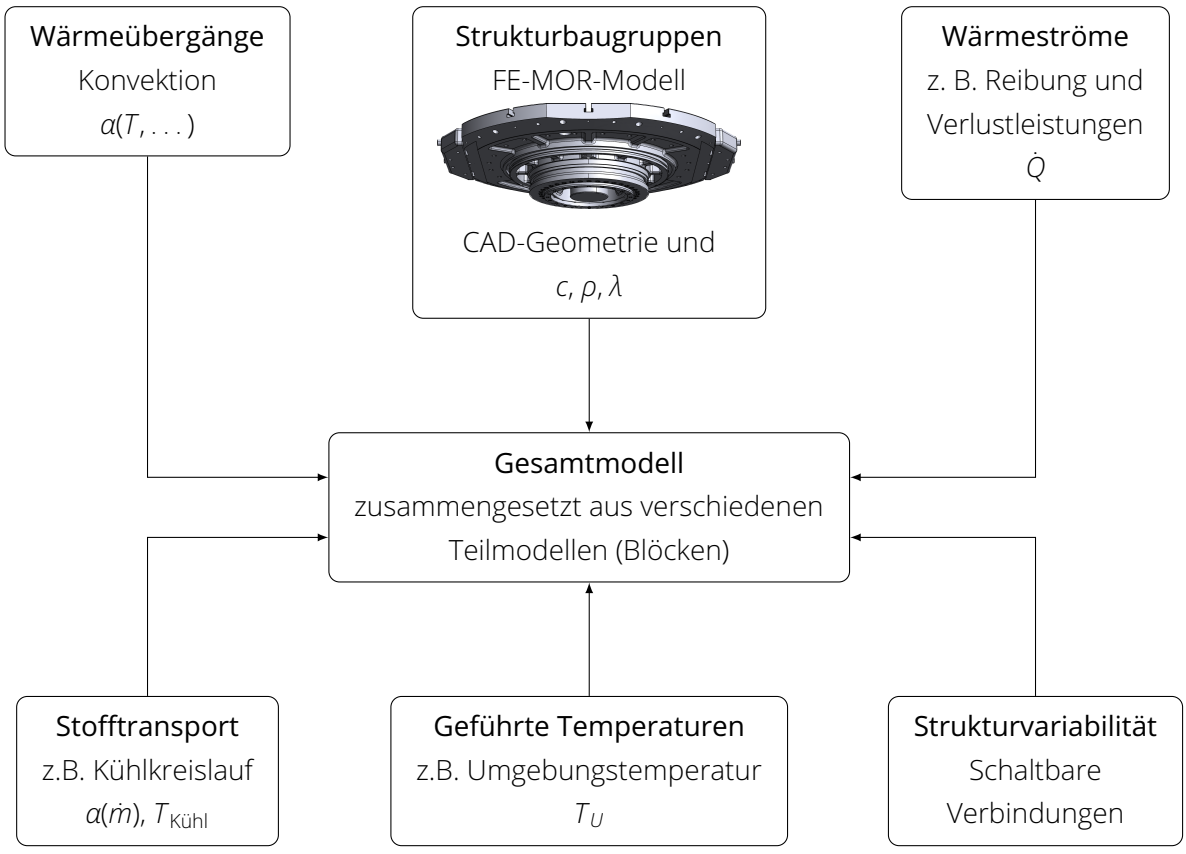

Abbildung 2.1: Thermisches Gesamtmodell als Blocksimulation. Die außen dargestellten Teilmodelle sind Bausteine der thermischen Simulation.

Abbildung 2.1 zeigt verschiedene Teilmodelle, die zu einem thermischen **Gesamtmodell** kombiniert werden. Mit ihnen kann das thermische Verhalten eines Teils der WZM oder auch das der gesamten WZM modelliert werden. Der Vorteil an dieser Art der Modellierung und Darstellung ist

die Möglichkeit, damit zu abstrahieren. Einzelne Teilmodelle können zu größeren Teilmodellen zusammengesetzt werden, die dann wiederum als einzelner Block aufgefasst werden. Dies erlaubt es, die Teilmodelle und auch ihre Parameter zu kondensieren (vgl. hierzu Schroeder; Kauschinger et al. (2019)). Aufgrund der Blockdarstellung können Teilmodelle in verschiedenen Zusammenhängen wiederverwendet und neu kombiniert werden.

Die **Strukturbaugruppen** werden als FE-MOR-Modelle modelliert. Diese geben das thermische Verhalten getreu wieder und erlauben zugleich eine weit schnellere Simulation als die entsprechenden hochaufgelösten FE-Modelle, wie im letzten Abschnitt (2.2) dargelegt. Zum Erstellen der Teilmodelle werden die Geometrie, in Form eines CAD-Modells, und die Materialparameter ρ, c, λ benötigt. Zudem werden die Ein- und Ausgänge des Modells festgelegt. Die Eingänge sind entweder ein Wärmestrom $\dot{Q}$ oder eine Umgebungstemperatur T_U mit dazugehörigem Wärmeübergangskoeffizienten α. Die Ausgänge sind Temperaturen T.

Die restlichen Teilmodelle modellieren die Randbedingungen und werden mit den Ein- und Ausgängen der Strukturbaugruppen verbunden. **Wärmeströme** $\dot{Q}$ werden beispielsweise durch Reibung oder der Abwärme elektrischer Antriebe verursacht. Weitere thermische Lasten werden als **Wärmeübergänge** abgebildet. Sie werden durch den Wärmeübergangskoeffizienten α modelliert. Auf diese Weise wird zum Beispiel die Konvektion modelliert. Erzwungene, gerichtete Wärmeübergänge wie sie beispielsweise durch einen Kühlkreislauf bewirkt werden, werden als Wärmeübergänge durch **Stofftransport** modelliert. Dieser wird wiederum durch α ausgedrückt und ist in diesem Fall abhängig vom Massenstrom $\dot{m}$. Nicht alle Eingänge müssen oder können simuliert werden. Sie können stattdessen auch einem vorgegebenen Verlauf folgen. Auf diese Weise werden **geführte Temperaturen** realisiert, womit beispielsweise die Umgebungstemperatur T_U, die einen maßgeblichen Einfluss auf die thermische Simulation ausübt, vorgegeben werden. Aber auch die relativen Bewegungen der Strukturbauteile während des Arbeitszyklus werden so vorgegeben.

(Schroeder; Galant et al., 2018; Schroeder; Kauschinger et al., 2019)

Die relative Bewegung der Strukturbauteile wird **Strukturvariabilität** genannt. Sie wird durch schaltbare Verbindungen der Ein- und Ausgänge abgebildet. Dafür werden die in der WZM gegenüberliegenden und relativ bewegten Koppelflächen in Bewegungsrichtung in gleich große Teilflächen aufgeteilt. Die Anzahl der Teilflächen wird in Abhängigkeit zur geforderten Auflösung festgelegt. Durch das Aufteilen werden auch die Ein- und Ausgänge aufgeteilt. Sie können positionsabhängig umgeschaltet werden und sind damit entsprechend ihrer aktuellen Lage verbunden.

(Galant et al., 2014)

Die unterschiedlichen Teilmodelle werden durch verschiedene Ansätze charakterisiert, welche ihrerseits mit unterschiedlichen Verfahren umgesetzt werden können. Die Ansätze für die Randbedingungen sind wie auch das FE-Modell Näherungsansätze. Bei ihrer Formulierung muss zwischen benötigter Genauigkeit und möglicher Auflösung, auch im Hinblick auf die Rechenzeit, abgewägt werden. Bei der thermischen Simulation werden neben physikalischen Modellen auch empirische Ansätze verwendet. Dies ist für manche der betrachteten Phänomene aufgrund ihrer Komplexität nicht anders möglich. Dazu zählt der **Wärmeübergang** durch Konvektion. Die dabei stattfindenden Strömungsphänomene sind teils chaotisch und nur schwer physikalisch zu simulieren. Die

durch die empirischen Ansätze enstehenden Ungenauigkeiten sollten für die Gesamtsimulation beachtet werden, damit die Genauigkeit der Teilsysteme zueinender abgestimmt ist und nicht mit einzelnen unnötig detaillierten Modellen die Rechenzeit erhöht wird.

Die in dieser Arbeit betrachteten FE-MOR-Modelle sind physikalische Modelle und können im Gegensatz zu den empirischen Ansätzen mit geringer Unsicherheit berechnet werden.

(Schroeder; Kauschinger et al., 2019)

Die Simulation der thermischen Verformung, wie sie in Galant et al. (2014) beschrieben ist, kann ebenfalls als Teilmodell des Gesamtmodells, das dann die Thermo-Elastik abbildet, modelliert werden. Dies ist aufgrund der schwachen Kopplung zwischen den beiden Teilsystemen möglich. Für die Simulation werden ebenfalls FE-Modelle verwendet. Da bei der Strukturmechanik die Strukturvariabilität in die FE-Matrizen einfließt, kann sie nicht durch schaltbare Verbindungen abgebildet werden. Stattdessen werden für jede Position seperate FE-Modelle verwendet. Die thermische Verformung kann alternativ, falls nur die Verschiebung des Tool Center Point (TCP) von Interesse ist, auch direkt als Ausgang des FE-MOR-Modells berechnet werden (vgl. hierzu Galant et al. (2014)).

3 Analyse des Arbeitsablaufs

In diesem Kapitel wird der Arbeitsablauf zum Erstellen von ordnungsreduzierten thermischen Modellen von WZM analysiert und modularisiert. Im ersten Abschnitt 3.1 wird der Arbeitsablauf aus der Anwendersicht vorgestellt. Dabei werden die einzelnen Arbeitsschritte und dafür benötigten Eingangsgrößen betrachtet. Anschließend wird der Arbeitsablauf in Abschnitt 3.2 modularisiert. Er wird dafür in seine Funktionsmodule und die Schnittstellen zwischen diesen aufgeteilt. Im letzten Abschnitt 3.3 wird anhand des modularisierten Arbeitsablaufs untersucht, wie die Module in der Referenzimplementierung (Galant et al., 2014) und einer weiteren Implementierung umgesetzt wurden.

3.1 Der Arbeitsablauf aus Anwendersicht

Die erste Analyse erfolgt anhand der während des Arbeitsablaufs durchgeführten Arbeitsschritte und dafür benötigten Eingangsgrößen. Dies ist die Anwenderperspektive auf das Verfahren. Der Anwender will mit dem Arbeitsablauf eine Simulation durchführen und deren Ergebnisse nutzen. Seine Sicht beschränkt sich auf die zur Erfüllung dieser Aufgabe notwendigen Schritte und für die Simulation relevanten Parameter. Der Arbeitsablauf aus Anwendersicht ist in Abbildung 3.1 dargestellt; er lässt sich grob in drei Bereiche einteilen.

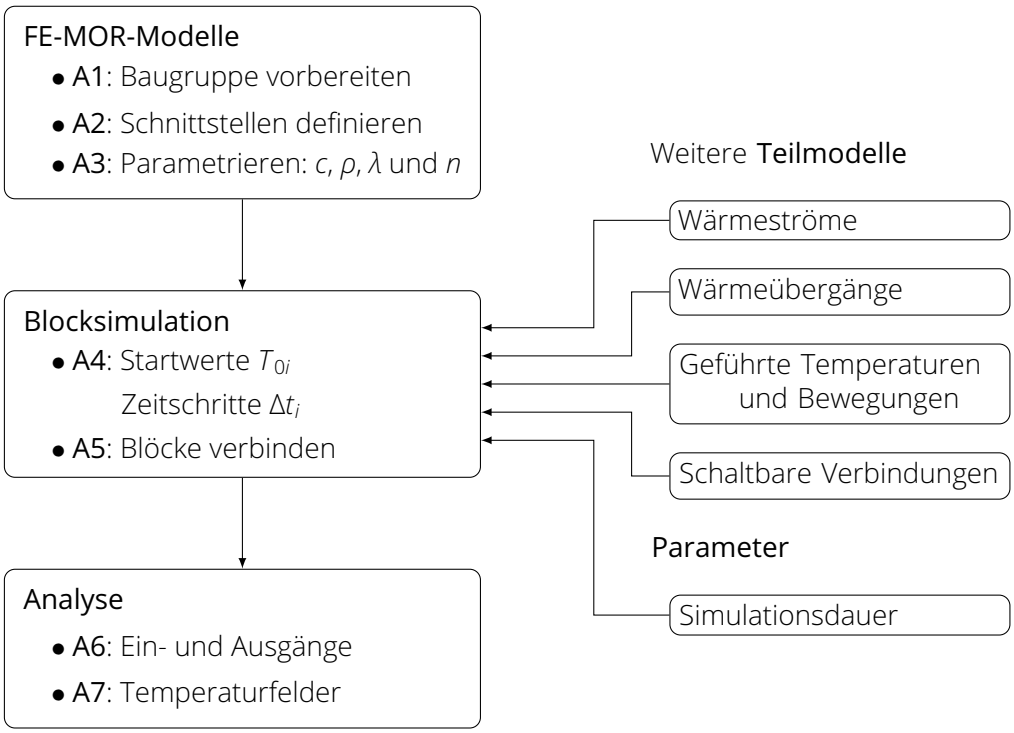

Abbildung 3.1: Der Arbeitsablauf aus Anwendersicht. A1-7 bezeichnen die dabei ausgeführten Arbeitsschritte.

FE-MOR-Modelle

Der erste Bereich ist das Erstellen der FE-MOR-Modelle. Es wird je eines pro strukturfester Baugruppe erstellt. Um in der Blocksimulation die Strukturvariabilität abbilden zu können, wird die WZM so aufgeteilt, dass innerhalb der reduzierten Teilmodelle keine relativen Bewegungen stattfinden. Von jeder Baugruppe wird in drei Arbeitsschritten ein FE-MOR-Modell abgeleitet. Die Baugruppen liegen zu Beginn als CAD-Geometrie-Dateien vor und werden im ersten Arbeitsschritt (A1) vorbereitet. Falls die Baugruppe aus mehreren Teilen zusammengesetzt ist, wird sie zunächst zu einem Bauteil zusammengefügt. Dieses wird anschließend, falls nötig, repariert und vereinfacht. Abschließend werden die Außenflächen der Baugruppe so aufgeteilt, dass im nächsten Arbeitsschritt alle benötigten Koppelflächen als einzelne Flächen vorliegen.

Im zweiten Arbeitsschritt (A2) werden die Schnittstellen des FE-MOR-Modells festgelegt. Bei der Wahl der Ein- und Ausgänge müssen alle Wärmeströme, Wärmeübergänge und Temperaturen, die im Kontext des Gesamtmodells benötigt werden, bedacht werden. Die Schnittstellen werden definiert, indem einzelne Flächen oder Flächengruppen der Baugruppe ausgewählt und benannt werden. An dieser Stelle wird weiterhin die Strukturvariabilität berücksichtigt. Die Flächen, an denen eine Relativbewegung abgebildet werden soll, wurden bereits im vorherigen Schritt aufgeteilt und werden nun ausgewählt und zusammenhängend benannt. Die Namen ermöglichen eine Zuordnung der Ein- und Ausgänge im Blockmodell zu den Koppelflächen der CAD-Geometrie.

Im dritten Arbeitsschritt (A3) wird das FE-MOR-Modell parametriert. Dafür werden zuerst die Materialparameter c, ρ und λ festgelegt und anschließend die Dimension des reduzierten Modells n. Es existiert noch kein systematisches Vorgehen, um letztere zu bestimmen. Sie wird stattdessen unter Abwägung der zulässigen Simulationsdauer und der benötigten Genauigkeit des Modells festgelegt.

Das Erstellen des FE-Netzes wird nicht als eigener Arbeitsschritt aufgeführt, da es beim betrachteten Verfahren komplett automatisiert geschieht. Voraussetzung hierfür ist allerdings ein Vernetzer, der dies auch für komplexe Geometrien gewährleisten kann (Galant et al., 2014).

Blocksimulation

Das FE-MOR-Modell ist nun vollständig bestimmt und kann mit weiteren FE-MOR-Modellen und anderen Teilmodellen zur Blocksimulation der WZM, dem zweiten Bereich des Arbeitsablaufs, kombiniert werden. Die übrigen Teilmodelle sind in Abbildung 3.1 auf der rechten Seite dargestellt. Sie wurden bereits in Abschnitt 2.3 behandelt und werden außerhalb des hier behandelten Arbeitsablaufs erstellt. Als ersten Arbeitsschritt (A4) werden für die Teilmodelle Startwerte festgelegt. Für die FE-MOR-Modelle sind das Temperaturfelder. Zudem wird für jedes Teilmodell festgelegt, mit welchem Zeitschritt es simuliert wird. Dieser wird passend zum Zeitverhalten des Teilmodells gewählt und könnte zukünftig auch durch eine automatische Zeitschrittsteuerung erfolgen.

Im zweiten Arbeitsschritt (A5) werden die Teilmodelle über ihre Ein- und Ausgänge miteinander verbunden. Wird der Eingang eines Teilmodells mit dem Ausgang eines weiteren Teilmodells verbunden, gibt er während der Simulation den Wert des Ausgangs vor. An dieser Stelle müssen sowohl die Einheiten als auch die Vorzeichen der Signale bedacht werden.

Analyse

Das Blockmodell ist nun fertiggestellt und kann nach Festlegen der Simulationsdauer simuliert werden. Dies führt zum dritten Bereich des Arbeitsablaufs: der Analyse der Ergebnisse. Diese kann auf zwei Detailstufen durchgeführt werden. Auf der ersten Detailstufe (A6) werden die an den Verbindungen des Blockmodells gemessenen Werte betrachtet. Das durch das Blockmodell beschriebene Netzwerk bildet die thermischen Zusammenhänge der modellierten WZM ab und kondensiert damit die Zustände des Systems. Diese Kondensation der betrachteten Größen ermöglicht es, das thermische Verhalten während der Simulation nachzuvollziehen und Einflüsse darauf zu ermitteln.

Die zweite Detailstufe (A7) ist die Analyse der Temperaturfelder in den Strukturbauteilen. Diese liegen nach der Rücktransformation aus dem reduzierten Raum (vgl. Abschnitt 2.2) für alle Knoten des FE-Netzes vor. Während sich die vorherige Detailstufe also mit kondensierten Größen zwischen den Teilmodellen beschäfigt hat, werden nun detaillierte Temperaturfelder in den Strukturbauteilen analysiert.

3.2 Modularisierung durch funktionale Zerlegung

Der erste Bereich des Arbeitsablaufs, die Methode zum Erstellen der FE-MOR-Modelle, wird nun genauer analysiert und modularisiert. Dafür wird die Methode in einzelne Funktionen aufgeteilt. Jede von diesen verarbeitet Daten und gibt sie in abgewandelter Form zurück. Die Funktionen werden so aufgeteilt, dass sie alle für den Arbeitsablauf relevanten Daten ausgeben. Die Methode wird somit in ihre kleinsten, sinnvoll zusammenhängenden Funktionsmodule aufgeteilt, mit den Daten als Schnittstellen zwischen diesen.

Die Daten können in Zwischenergebnisse, die durch die Funktionen generiert werden, und Datenquellen, die in die Methode einfließen, unterteilt werden. Für die Reihenfolge der Daten wird festgelegt, dass die Datenquellen, beispielsweise Parameter, zum spätest möglichen Zeitpunkt in die Methode einfließen. Dies ist notwendig, um die größtmögliche Flexibiliät der Module zu erreichen und um die Schnittstellen klar abzugrenzen.

Abbildung 3.2 visualisiert das Ergebnis der Analyse. In der linken Spalte sind die Daten in chronologischer Reihenfolge dargestellt. Datenquellen sind durch ▷ und Zwischenergebnisse durch ● gekennzeichnet. In der rechten Spalte sind die Funktionen ihrem Ablauf entsprechend angeordnet.

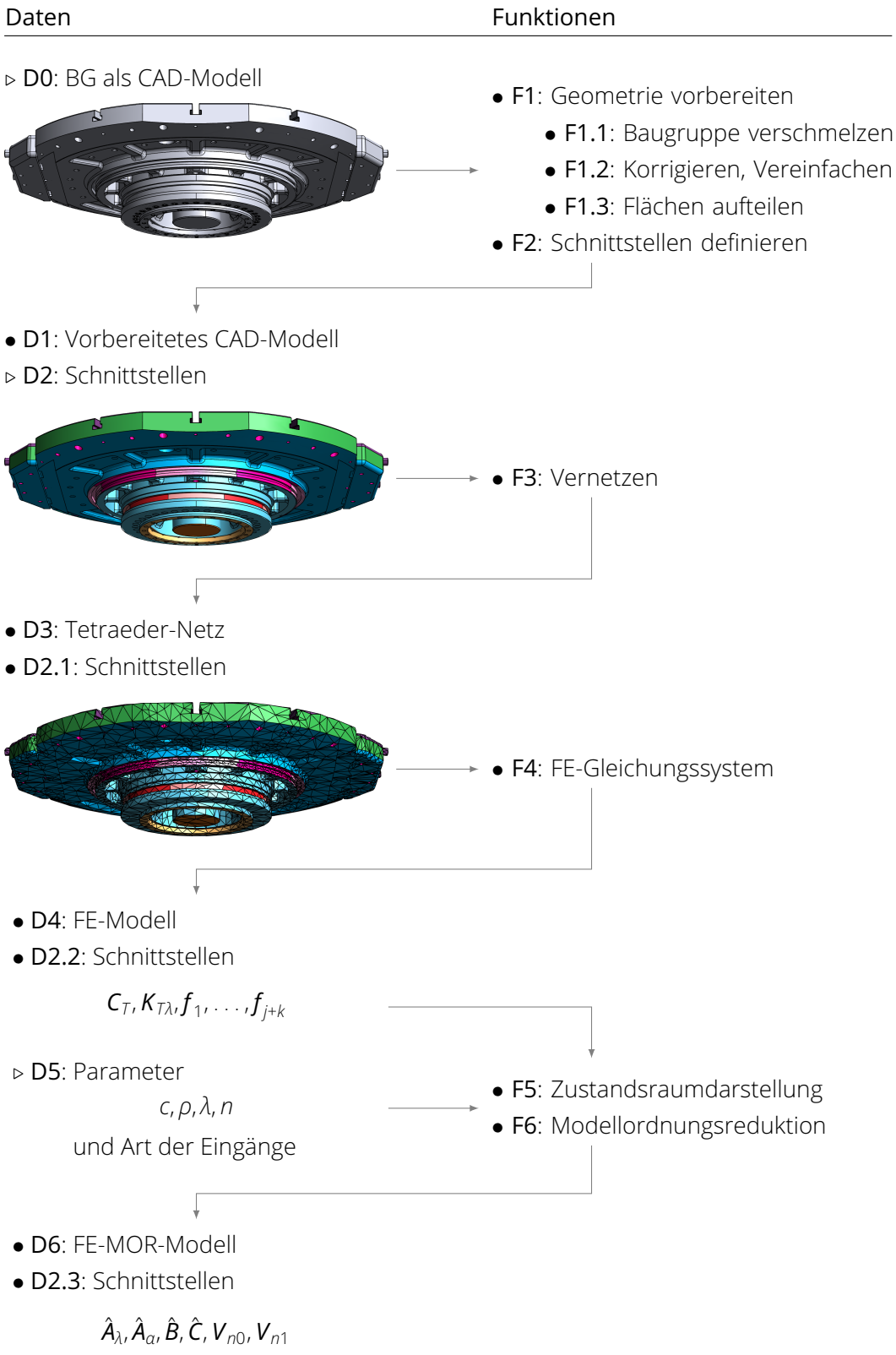

Abbildung 3.2: Daten (D0-6) und Funktionen (F1-F6) des modularisierten Arbeitsablaufs

Den Ausgangspunkt bildet auch hier wieder das CAD-Modell der Baugruppe (D0). Dieses wird durch die erste Funktion, die drei Teilfunktionen umfasst, vorbereitet (F1). Zunächst wird die strukturfeste Baugruppe zu einem Volumen „verschmolzen" (F1.1), damit das später durchgeführte Vernetzen zu einem einheitlichen Netz der gesamten Baugruppe führt. Daraufhin wird die Geometrie nötigenfalls korrigiert und vereinfacht (F1.2). Beides geschieht, um die Geometrie anschließend vernetzen zu können. Komplexe CAD-Geometrien enthalten häufig zahlreiche Fehler, wie Überschneidungen und schmale Öffnungen. Verstärkt werden die Fehler, falls die CAD-Datei zwischen verschiedenen Formaten konvertiert wurde. Auch enthalten komplexe CAD-Geometrien oft kleine Flächen, die zu kleinteiligen Netzstrukturen und ungünstig geformten finiten Elementen führen. All dies macht eine Korrektur häufig notwendig. (Hu; Zhou et al., 2018; Guo et al., 2019; Sadowski et al., 2007)

Vor dem Vernetzen muss die Geometrie meist noch weiter vereinfacht werden, um ein weniger komplexes und damit mit der FEM schneller berechenbares Netz zu erhalten. In diesem Schritt werden für das thermische Verhalten irrelevante Geometriedetails entfernt. Das Vereinfachen der Geometrie kann nur eingeschränkt automatisiert werden und bedeutet meist hohe manuelle Aufwände. Beim betrachteten Verfahren kann jedoch auf eine Vereinfachung der Geometrie verzichtet werden, solange ein Netz für die Geometrie erstellt werden kann, da das FE-Modell selbst reduziert und damit vereinfacht wird. (Galant et al., 2014)

Anschließend werden die Flächen der Geometrie so aufgeteilt, dass alle Koppelflächen als eigene Flächen vorliegen (F1.3). In dieser Teilfunktion werden auch die strukturvariablen Koppelflächen in gleich große Teilflächen unterteilt. Das CAD-Modell liegt nun als vereinfachtes CAD-Modell (D1) vor.

Auf diesem werden nun die Schnittstellen des FE-MOR-Modells definiert (F2). Dafür werden die zu einem Ein- oder Ausgang gehörenden Außenflächen ausgewählt, zu einer Koppelfläche zusammengefasst und dem Ein- oder Ausgang zugewiesen. Hier wird also festgelegt, wie die Schnittstellen (D2) des FE-MOR-Modells auf die Koppelflächen der CAD-Geometrie abgebildet werden.

Das vereinfachte CAD-Modell (D1) wird nun vernetzt (F3). Dadurch wird ein Tetraeder-Netz (D3) für die Geometrie erstellt. Dieses besteht aus einer Liste der FE-Knoten und einer zweiten die die Zugehörigkeit der Knoten zu einem Element enthält. Die auf das vereinfachte CAD-Modell bezogenen Schnittstellen (D2) werden so angepasst, dass sie nun nicht mehr auf Flächen, sondern auf den äußeren Knoten des Netzes definiert sind (D2.1).

Mit dem Tetraeder-Netz (D3) kann nun das FE-Gleichungssystem (F4) erstellt werden. Dieses erzeugt das FE-Modell (D4), wie in Abschnitt 2.1 beschrieben. Es besteht aus der Wärmekapazitätsmatrix $\boldsymbol{C}_T$, der Wärmeleitfähigkeitsmatrix $\boldsymbol{K}_{T\lambda}$ und einem thermischen Lastvektor $\boldsymbol{f}_i$ pro Schnittstelle. Dieser wird für $\boldsymbol{f}_{2i}$, $\boldsymbol{f}_{3i}$ und die Diagonale von $\boldsymbol{K}_{T\alpha i}$ verwendet. Denn diese sind ohne die zu ihnen gehörenden Lastwerte alle gleich aufgebaut (vgl. Gleichungen 2.13, 2.14 und 2.15,). $\boldsymbol{K}_{T\lambda}$ liegt als dünn besetzte Matrix vor, $\boldsymbol{C}_T$ und $\boldsymbol{f}_i$ als Vektoren. Denn $\boldsymbol{C}_T$ musste für die weitere Berechnung diagonalisiert werden und es muss nun lediglich der Diagonalenvektor gespeichert werden. Die Schnittstellen (D2.1) werden wiederum mitgenommen und diesmal auf die Zeilen des FE-Matrixsystems bezogen (D2.2).

Das FE-Modell wird als nächstes in die Zustandsraumdarstellung überführt (F5), wie in Abschnitt 2.2 beschrieben. Dafür werden die Materialparameter c, ρ und λ (D5) benötigt. Zudem wird hier festgelegt, welcher Art, Wärmestrom oder -übergang, die durch die Schnittstellen definierten Eingänge des Modells sind. Ausgehend von der Zustandsraumdarstellung wird dann die Modellordnungsreduktion (F6) durchgeführt. Dafür wird die Dimension des reduzierten Modells n als weiterer Parameter (D5) benötigt.

Schließlich ergibt sich das FE-MOR-Modell (D6). Es besteht aus je $j + k$ Matrizen $\hat{\boldsymbol{A}}_\lambda$, $\hat{\boldsymbol{A}}_\alpha$, $\hat{\boldsymbol{B}}$ und $\hat{\boldsymbol{C}}$; einer für jeden Eingang des Modells (vgl. Gleichung 2.27). Zudem werden die beiden Transformationsmatrizen $\boldsymbol{V}_{n0}$ und $\boldsymbol{V}_{n1}$ benötigt, um das Temperaturfeld zu Beginn der Simulation $\boldsymbol{T}_0$ zu reduzieren und die berechneten reduzierten Temperaturfelder zurückzutransformieren. In der Zustandsraumdarstellung (F5) werden außerdem die Schnittstellen (D2.2) angepasst und nun auf die Ein- und Ausgänge des FE-MOR-Modells bezogen (D2.3).

Um das FE-MOR-Modell zu erstellen, werden also sechs Funktionsmodule (F1-6) benötigt. Es gehen dabei an drei Stellen Daten in den Ablauf ein (D0, D2 und D5). Aus diesen werden vier Ergebnisse (D1, D3, D4, D6) berechnet, von denen das letzte das gesuchte FE-MOR-Modell ist.

3.3 Analyse der Referenzimplementierung

In diesem Abschnitt wird untersucht, wie die Module und Schnittstellen des Arbeitsablaufs in der Referenzimplementierung Galant (2014) umgesetzt sind. In Abbildung 3.3 sind die im letzten Abschnitt ermittelten Funktionen und Daten des Arbeitsablaufs dargestellt. Zusätzlich sind nun die in der Referenzimplementierung verwendeten Werkzeuge eingezeichnet. Sie umfassen die mit ihnen umgesetzten Funktionen und Daten. An den Übergängen zwischen den Werkzeugen ergeben sich die Schnittstellen.

Das CAD-Modell (D0) kann, je nach Präferenz und Verfügbarkeit, mit unterschiedlichen CAD-Programmen erstellt werden, beispielsweise SolidWorks (Dassault Systèmes, 2020) oder Creo (PTC, 2020). Statt der nativen CAD-Formate kann alternativ auch ein standardisiertes Austauschformat, beispielsweise das STEP-Format (ISO-10303-1, 1994), verwendet werden.

Die ersten vier Funktionen (F1-4) wurden mit der proprietären FE-Software Ansys (Ansys Inc., 2020) umgesetzt, wobei die Vorbereitung der Geometrie (F1) wahlweise bereits im CAD-Programm erfolgen kann. Ansys besteht aus verschiedenen Programmen unter einer gemeinsamen Oberfläche (Workbench). Es enthält zwei CAD-Modellierer: DesignModeler und SpaceClaim. Beide unterstützen verschiedene Funktionen zur Korrektur und Vereinfachung der Geometrie, wobei dies zum Teil auch automatisiert durchgeführt werden kann. Außerdem können beide alle gängigen proprietären und zahlreiche offene CAD-Formate lesen, was die Schnittstelle (D0) zur Geometrie äußerst flexibel macht.

Das Definieren der Schnittstellen (F2), das Vernetzen (F3) und das Erstellen des FE-Gleichungssystems (F4) werden mit Ansys Mechanical, dem Strukturmechanikmodul des Programms, durchgeführt. Die Definition der Schnittstellen (F2), welche in Ansys Komponenten genannt werden, erfolgt

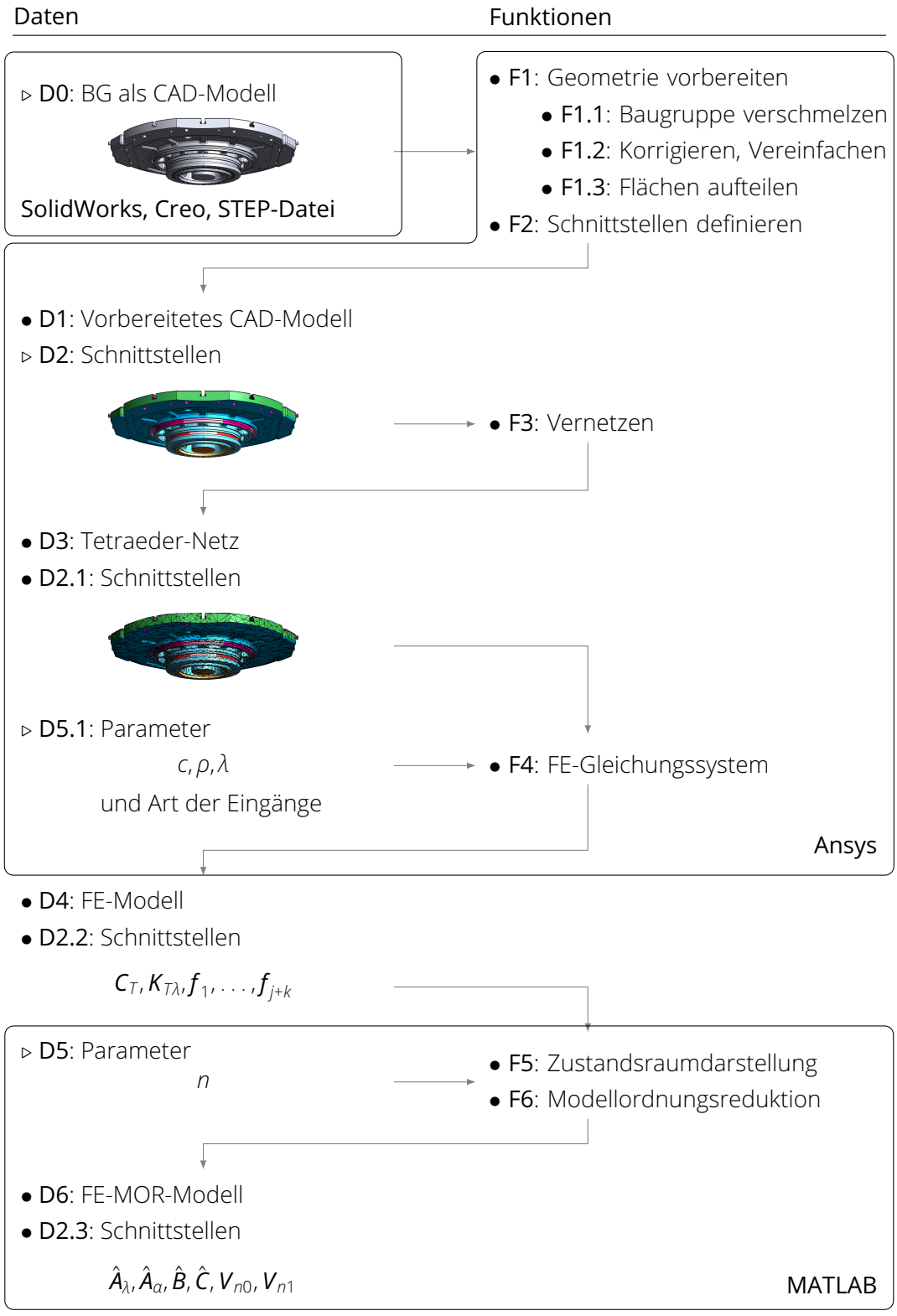

Abbildung 3.3: Implementierung der Module in Galant et al. (2014). Das zur Implementierung genutzte Werkzeug umfasst die mit ihm umgesetzten Funktionen und Daten.

durch interaktives Auswählen der Koppelflächen und benennen als Komponente. Jede Schnittstelle ist somit durch einen Namen eindeutig unterscheidbar.

Um die Geometrie zu vernetzen (F3), wird als Elementtyp lineare Tetraederelemente gewählt und die Elementgröße auf adaptiv gestellt. Das Vernetzen erfolgt dann automatisch, außer die Geometrie war noch fehlerhaft. Für komplexe Geometrien mit kleinen Flächen erstellt Ansys, wie auch andere Vernetzer, eher fein aufgelöste FE-Netze. Diese führen zwar zu großen FE-Netzen. Das hat jedoch auf die Dimension des FE-MOR-Modells keinen Einfluss.

Ein Teil der Parameter (D5), die Materialparameter (D5.1), werden bereits jetzt in Ansys gesetzt. Auch die Art der Eingänge (ebenfalls D5.1) wird hier schon definiert, indem an die Namen der Schnittstellen ein Kürzel, je nach Art, angehängt wird. Die Materialparameter müssen, wie im letzten Abschnitt (3.2) dargelegt, nicht bereits an dieser Stelle des Arbeitsablaufs eingebracht werden. In Verbindung mit Ansys kann durch die vorgezogene Parametrierung jedoch die integrierte Materialdatenbank genutzt werden.

Das FE-Gleichungssystem aufzustellen und zu exportieren (F4) ist mit Ansys äußerst aufwendig umzusetzen, da das Programm dies nicht vorsieht. Beides erfolgt mit einem Skript in der Skriptsprache von Ansys: APDL. Es wird zunächst eine Analyse erstellt und gestartet. Dadurch werden von Ansys die Matrizen für die Analyse aufgestellt. Dann wird Ansys so umgestellt, dass es das FE-Matrixsystem als eine dünnbesetzte Matrix im Harwell-Boeing-Format ausgibt. Es handelt sich dabei um ein Textformat, das gewählt wurde, da es die einzige Möglichkeit für den Matrixexport mit Ansys ist. Dieses Vorgehen wird mit jeweils anderen Einstellungen für $\boldsymbol{C}_T$, $\boldsymbol{K}_{T\lambda}$ und alle $\boldsymbol{f}_i$ je einmal durchgeführt:

- $\boldsymbol{C}_T$ wird über eine transiente, thermische Analyse mit $T_0 = 0\,°\mathrm{C}$ und $\Delta t = t_f = 1\,\mathrm{s}$ exportiert. $\boldsymbol{C}_T$ ergibt sich anschließend, indem $\boldsymbol{K}_{T\alpha}$ von der exportierten Matrix abgezogen wird.
- $\boldsymbol{K}_{T\alpha}$ wird über eine statische, thermische Analyse exportiert.
- $\boldsymbol{f}_i$ wird ebenfalls über eine statische, thermische Analyse exportiert. Zuvor werden die Knoten der Schnittstelle i ausgewählt und geprüft, ob sie zu den Außenflächen gehören. Dann wird auf den Knoten eine Konvektions-Randbedingung mit $\alpha = 1$ definiert, die Analyse durchgeführt und die Matrix exportiert. $\boldsymbol{f}_i$ entspricht der Diagonalen der exportierten Matrix.

(vgl. hierzu Großmann et al. (2011))

Außerdem werden noch je eine Knotenliste mit allen Knoten pro Schnittstelle und die Knotenindizes für die exportierten Matrizen ausgegeben. All dies ergibt das FE-Modell (D4) mit Schnittstellen (D2.2). Obwohl der Export aufwendig umzusetzen ist, läuft er, da skriptbasiert, vollautomatisch ab.

Die Zustandsraumdarstellung (F5) und Modellordnungsreduktion (F6) wurden, wie auch die anschließende Blocksimulation, mit der numerischen Berechnungssoftware Matlab (The MathWorks, Inc., 2020) umgesetzt. Diese bietet sich dafür an, da sie für die schnelle Matrixrechnung konzipiert wurde, vielfältig im Ingenieurwesen angewendet wird und mit Simulink eine eingebaute grafische Oberfläche für Blocksimulationen besitzt.

Das Einlesen der FE-Matrizen ist nicht mit in Matlab integrierten Werkzeugen möglich. Stattdessen wurde freie Software verwendet (Burkardt, 2019). Die anschließende Umformung in den Zustandsraum (F5) und die MOR (F6) werden wie in Abschnitt 2.2 beschrieben durchgeführt.

(Galant, 2014)

Die Referenzimplementierung kann also grob in zwei Module unterteilt werden. Eines ist mit Ansys und das andere mit Matlab implementiert. Die Analyse in 3.2 hat jedoch gezeigt, dass der Arbeitsablauf in sechs Module und die dazugehörigen Schnittstellen aufgeteilt werden kann. Die Schnittstellen gewährleisten eine Austauschbarkeit der verwendeten Werkzeuge.

Dies ist bei der Referenzimplementierung nicht gegeben. Zwar ist die Schnittstelle zur Geometrie flexibel, aber die Schnittstelle zum FE-MOR-Modell ist es nicht. Es sollte hier jedoch noch festgehalten werden, dass Ansys durch seinen Geometrieimport auch noch die Schnittstelle zum vorbereiteten CAD-Modell (D1/2) abbilden kann. Zudem kann es verschiedene Netzformate lesen, wodurch die Schnittstelle zum Tetraeder-Netz (D3/2.1) abgebildet werden kann.

Die in der Referenzimplementierung tatsächlich genutzte Schnittstelle zum FE-Modell (D4/D2.2) wird jedoch von Ansys stark eingeschränkt und ist nicht flexibel. Dies ist so, da diese Schnittstelle in Ansys, das für die Multiphysik-Gesamtsimulation ausgelegt ist, nicht vorgesehen ist. Der Anwender kann in diesem Fall, falls ihm kein cleverer Trick einfällt, nur hoffen, dass sich das in zukünftigen Ansysversionen ändert.

Es existiert noch eine zweite, nicht fertiggestellte Implementierung des Arbeitsablaufs. Sie konnte im Rahmen dieser Arbeit nicht untersucht werden und wird deswegen an dieser Stelle nur kurz behandelt.

Die zweite Implementierung verwendet statt einer Modularisierung des Arbeitsablaufs, wie sie in dieser Arbeit entwickelt und demonstriert wird, eine einheitliche Metasprache um die Gesamtsimulation zu beschreiben. Diese ist als strukturierte, maschinenlesbare Textdatei umgesetzt und beschreibt die FE-MOR-Modelle, andere Teilmodelle und deren Interaktionen. Erstellt wird das Beschreibungsformat mit Ansys, das hierfür über die integrierte IronPython-Schnittstelle[1] zur grafischen Oberfläche für die FE-MOR-Modell-Erstellung erweitert wird.

Das FE-MOR-Blockmodell mit von der Implementierung unabhängigen und von unterschiedlichen Werkzeugen lesbaren Daten zu beschreiben, ist ein sinnvoller Ansatz. Denn im Vergleich zur eher starren Beschreibung im Matlab-Code der Referenzimplementierung ist das die universellere Schnittstelle. Ansys als Oberfläche und Umgebung für den Arbeitsablauf ist sinnvoll, falls Ansys bereits zur Simulation eingesetzt wird oder weitere Funktionaliäten von Ansys benötigt werden. Diese Lösung ist jedoch durch die Funktionalitäten der IronPython-Schnittstelle begrenzt und muss zugleich auf Änderungen dieser reagieren. Ein starres, grafisches Interface für den Arbeitsablauf schränkt zudem dessen Automatisierbarkeit ein.

[1] IronPython ist eine freie Implementierung der Python Programmiersprache auf Basis des Microsoft .NET Frameworks (.NET Foundation, 2020).

4 Recherche zur Umsetzung der Module

In diesem Kapitel wird untersucht, wie die Module und Schnittstellen des Arbeitsablaufs mit freier Software umgesetzt werden können. Es wird recherchiert, welche freie Software für die Umsetzung der Module und Schnittstellen zur Verfügung steht und ob sich mit ihr das gestellte Ziel erfüllen lässt. Dafür werden im ersten Abschnitt 4.1 zunächst Kriterien für die Recherche, deren Bewertung und die anschließende Umsetzung festgelegt. Anschließend werden die Ergebnisse für die einzelnen Module vorgestellt und anhand der Kriterien bewertet.

Zunächst werden in Abschnitt 4.2 CAD-Schnittstellen, -Werkzeuge und die Auswahl der Koppelflächen behandelt. Dann in Abschnitt 4.3 Netz-Formate und Vernetzer. Der nächste Abschnitt 4.4 beschäfigt sich mit FEM-Software. Anschließend werden in Abschnitt 4.5 Werkzeuge zur numerischen Berechnung untersucht, mit denen die MOR und die digitale Blocksimulationen realisiert werden. Abschließend werden in Abschnitt 4.6 Programme für die Visualisierung von Temperaturfeldern und der dazugehörigen CAD-Geometrie betrachtet.

4.1 Kriterien und Schwerpunkte

Das zentrale Kriterium ist die Verwendung freier Software. Es soll gezeigt werden, dass damit der gesamte Arbeitsablauf robust und modular umgesetzt werden kann.

Für die Schnittstellen folgt, dass diese möglichst allgemeingültig sind und damit durch verschiedene Werkzeuge genutzt werden können. Wenn also für eine Schnittstelle ein offizieller oder auch inoffieller freier Standard existiert, dann soll dieser genutzt werden. Falls die Schnittstelle nicht allgemeingültig umgesetzt werden kann, wird stattdessen auf robuste Konvertierung zwischen verschiedenen Formaten gesetzt.

Dies bedeutet im Umkehrschluss, dass auch die Werkzeuge für die Module möglichst allgemeingültige Schnittstellen aufweisen sollen. Das wichtigste Kriterium für die Werkzeuge ist aber zunächst, dass mit ihnen ein Modul des Arbeitsablaufs oder ein Teil davon umgesetzt werden kann. Dabei soll das Modul so umgesetzt werden, dass es zukünftig gut erweitert oder abgewandelt werden kann.

Aus der Zielstellung (vgl. Kapitel 1.2) ergeben sich weitere Kriterien: Die Werkzeuge sollen robust und gerade im Hinblick auf die CAD-Geometrie fehlertolerant sein. Beide Kriterien ermöglichen und erhöhen die Automatisierbarkeit des Arbeitsablaufs. Dieser soll flexibel und mit austauschbaren Werkzeugen umgesetzt werden. Dies wurde bereits durch die Modularisierung erreicht und findet sich in den Kriterien für die Schnittstellen wieder.

Für die Recherche und Implementierung werden zudem zwei Schwerpunkte gesetzt, die sich bei der Analyse des Arbeitsablaufs und seiner Referenzimplementierung (vgl. Kapitel 3) als wesentliche Herausforderungen ergaben:

1. Das Vorbereiten der CAD-Geometrie ist oft nur durch hohen manuellen Aufwand realisierbar. Dies steht im Widerspruch zu der ansonsten guten Automatisierbarkeit des Arbeitsablaufs. Es soll deshalb untersucht werden, ob sich hier durch den Einsatz freier Software eine höhere Automatisierung erreichen lässt.
2. Das Aufstellen der FE-Matrizen erfordert ein proprietäres Simulationspaket, das dafür nicht vorgesehen ist und nur schwer in den restlichen Arbeitsablauf integrierbar ist. Es soll deshalb untersucht werden, ob dieser Arbeitsschritt mit freier Software einfacher umgesetzt werden kann.

Um die Module zu kombinieren und den Arbeitsablauf zu steuern, bietet sich eine Skriptsprache an. Werden die Module dann auch direkt in der Sprache implementiert, ist ihre Funktion eindeutig festgehalten und kann zugleich einfach erweitert und angepasst werden. Der Arbeitsablauf wird dadurch flexibel und ist zugleich automatisiert. Seine Funktionen haben vollen Zugriff auf die Infrastruktur der Skriptsprache, wie unter anderem ihre Schnittstellen und ihr interaktiver Entwicklungsprozess. Um die Module jedoch in einer Skriptsprache implementieren zu können, muss diese die benötigten Funktionalitäten entweder enthalten oder Schnittstellen zu diesen bereitstellen.

Eine naheliegende Wahl ist die Allzweck-Skriptsprache Python. Sie wird auf die unterschiedlichsten Problemstellungen im Ingenieurwesen angewendet und findet auch breite Anwendung über die Wissenschaft hinaus. Ebenso wichtig im betrachteten Kontext ist, dass es sich um freie Software handelt und es zahlreiche Bibliotheken gibt, die ihre Funktionalität erweitern. Die Python Werkzeugkette versucht nicht, bestehende Software zu ersetzen, sondern Schnittstellen zu dieser bereit zu stellen und eine verständliche Anwendung auf einer der Problemstellung angepassten Abstraktionsebene anzubieten. Die gute Anbindung an die Programmiersprache C macht darüber schnelle und optimierte Funktionen in Python möglich. Auf diese Weise bietet Python mit NumPy beispielsweise eine exzellente Bibliothek zur Matrixrechnung, aber auch zu CAD- und FEM-Funktionen, wie im restlichen Kapitel gezeigt wird. (Perez et al., 2011)

Wegen dieser Eigenschaften und seiner an die gegebene Problemstellung angepassten Bibliotheken wird Python für die Umsetzung des Arbeitsablaufs gewählt. Dies gewährleistet dessen Automatisierbarkeit und ermöglicht ihn oder einzelne Module direkt in andere Arbeitsabläufe oder bestehende Software zu integrieren. Es ergibt sich somit Skriptbarkeit, bestenfalls mit Python, als letztes Kriterium.

Bereits die Referenzimplementierung nutzt die Skriptsprache von Matlab für die MOR und die Blocksimulationen. Dieser Ansatz wird somit auch auf den ersten Teil des Arbeitsablaufs, der sich mit der Geometrie und der FEM beschäftigt, erweitert. Möglich wird dies durch die größere Vielfältigkeit der freien Pythonwerkzeugkette.

Zusammengefasst ergeben sich folgende Kriterien für die Recherche und die anschließende Implementierung:

1. Schnittstellen möglichst allgemeingültig
2. Werkzeuge robust und fehlertolerant
3. Werkzeuge skriptbar, vorzugsweise mit Python

4.2 Geometrie

4.2.1 CAD-Formate

Fast jedes CAD-Programm hat ein eigenes Dateiformat und kann zudem in ein standardisiertes Austauschformat exportieren. Da freie CAD-Programme keine proprietären Dateiformate lesen können, kann für die Geometrie-Schnittstelle also nur auf einen freien Standard zurück gegriffen werden.

Diese Rolle erfüllt das in der Norm ISO-10303-1 (1994) beschriebene STEP-Format[1]. Es speichert, wie andere CAD-Formate auch, die Begrenzungsflächen der Geometrie zusammen mit Metadaten. Es verwendet dazu eine parametrische Beschreibung der Punkte, Linien, Flächen und Volumen aus denen sich die Geometrie zusammensetzt. ISO-10303-1 (1994) beschreibt verschiedene Applikationsprotokolle für das STEP-Format. Für Konstruktionsdaten werden das Applikationsprotokoll (AP) 203 und 214 verwendet. Beide speichern Baugruppen und dazugehörige Bauteile. AP 214 kann darüber hinaus zusätzliche Informationen, wie die Farben der Bauteile und ihrer Flächen oder Materialinformationen speichern. Dies ermöglicht es, die Koppelflächen in STEP AP 214 zu speichern. AP 242 ist eine Weiterentwicklung, die AP 203 und 214 in einem Applikationsprotokoll vereint. Sie wird jedoch von keinem im Rahmen dieser Arbeit verwendeten Programm derzeit unterstützt.

Da Begrenzungsflächenformate die Geometrie mit Splines modellieren, ist diese nicht geschlossen. Zum Ausgleich wird im Format eine Toleranz angegeben. Mit ihr werden kleine Öffnungen und Überschneidungen in der Geometrie ausgeglichen. Dieser Umstand macht das Format anfällig für numerische Fehler, die durch Konvertieren von einem anderen Format zusätzlich verstärkt werden. Das Konvertieren zwischen verschiedenen Begrenzungsflächenformaten führt auch aufgrund verschiedener Implementierung häufig zu fehlerhaften Geometrien. Verstärkt wird das Problem durch die Verwendung eines allgemeingültigen Formats, wie STEP, da dieses nicht alle Details, die CAD-Werkzeuge intern verwenden, speichern kann. Die deswegen notwendigen Vereinfachungen machen häufig ein aufwendiges Reparieren vor dem Vernetzen notwendig. (Hu; Zhou et al., 2018; Geuzaine et al., 2009)

Das erste Funktionsmodul, welches die Geometrie vorbereitet, muss somit auch mit fehlerhaften Eingangsdaten umgehen können, um robust zu sein.

[1] Standard for the Exchange of Product model data

Während verschiedene kommerzielle Modellierkerne existieren, gibt es nur einen freien, der einen ähnlichen Funktionsumfang unterstützt: Open CASCADE Technology verwendet intern das BRep-Format und kann auch mit STEP-Daten umgehen. Das BRep-Format beschreibt nur die Geometrie und enthält keine weiteren Metadaten. (OPEN CASCADE, 2020a)

Die Schnittstelle zum CAD-Modell kann mit dem STEP-Format allgemeingültig umgesetzt werden. Es ist aufgrund seiner genormten Spezifikation und weiten Verbreitung auch robust. Jedoch muss mit Fehlern gerechnet werden.

4.2.2 CAD-Werkzeuge

Es gibt drei ausgereifte CAD-Werkzeuge, die auf Open CASCADE Technology aufbauen. Alle drei bieten eine Pythonschnittstelle und sind modular konzipiert. Sie werden nun vorgestellt:

FreeCAD wird für die Konstruktion entwickelt. Es ermöglicht einen modernen parametrischen Konstruktionsablauf mit Skizzen und verschiedensten darauf aufbauenden Funktionen. Obwohl es zahlreiche Funktionen aufweist, kann es vom Umfang noch nicht mit kommerziellen CAD-Programmen mithalten. Die Stärke von FreeCAD ist sein freier und modularer Aufbau. Der Anwender kann zwischen verschiedenen Oberflächen, die für unterschiedliche Aufgaben konzipiert sind, wählen. Alle Funktionen und Oberflächen können durch den Nutzer angepasst und erweitert werden. Dies geschieht mit der integrierten Skriptsprache Python. Auf diese Weise integriert FreeCAD auch ein Macrosystem, mit dem sich Arbeitsabläufe einfach automatisieren lassen. (FreeCAD, 2020)

SALOME ist als Plattform für die Simulation konzipiert und am ehesten mit Ansys vergleichbar. Es ist ähnlich modular und anpassbar wie FreeCAD und verwendet dafür ebenfalls Python. Es unterstützt seit Version 9.3 auch einen parametrischen Konstruktionsablauf. Der Fokus von SALOME ist, statt auf die Konstruktion, deutlich auf die Geometrievorbereitung für die anschließende Simulation, unter anderm mit der FEM, ausgerichtet. SALOME bietet verschiedene Werkzeuge, mit denen Geometrie überprüft, untersucht, repariert und vereinfacht werden kann. Aufgrund der Umstellung auf den moderneren Konstruktionsablauf ist die Funktionalität derzeit auf zwei Module aufgeteilt. Zwischen diesen können die Geometrien übertragen werden. Dies geschieht jedoch manuell. Salome läuft nicht absolut stabil und stürzt zuweilen ab. Manche Funktionen sind zudem nicht optimiert und somit recht langsam. Insgesamt bietet Salome jedoch einen großen Funktionsumfang und eine durchdachte Benutzeroberfläche. (OPEN CASCADE, 2020b)

Gmsh verfolgt ein ähnliches Ziel wie SALOME. Es ist jedoch älter und hat eine andere Herangehensweise, mit seinem Fokus auf Robustheit und geringem Ressourcenverbrauch. Gmsh bietet in seiner grafischen Oberfläche weit weniger Funktionen zur Geometriebearbeitung. Es hat jedoch ähnliche, auf Open CASCADE Technology basierende, Reperaturfunktionen wie Salome und stellt verschiedene Open CASCADE Technology Funktionen über seine Programmierschnittstellen bereit. Die Oberfläche von Gmsh kann im Gegensatz zu den beiden anderen CAD-Werkzeugen nur sehr eingeschränkt angepasst werden und verwendet seine eigene Skriptsprache. Es hat jedoch unter anderen auch eine Pythonschnittstelle und kann damit vollständig in Python eingebunden werden. (Geuzaine et al., 2009)

Alle drei beschriebenen Werkzeuge können mit fehlerhaften Geometrien umgehen. Je nach Schwere der Fehler kann eine automatische Fehlerkorrektur jedoch fehlschlagen und sie muss manuell erfolgen. Das ist aufgrund der damit verbundenen Komplexität unvermeidbar. Gmsh besitzt als einziges der Programme keine Werkzeuge zur manuellen Fehlerkorrektur. Dafür ist es robuster, verbraucht weniger Ressourcen und ist einfacher zu installieren. Alle drei Werkzeuge sind mit Python skriptbar und damit voll automatisierbar. FreeCAD und Gmsh können sogar komplett in Python eingebunden werden und als Bibliothek verwendet werden.

4.2.3 Auswahl der Koppelflächen

FreeCAD kann zwar Flächen benennen und einfärben. Dabei können jedoch derzeit keine Flächengruppen erfasst werden. Auch unterstützt FreeCAD noch keinen Export mit benannten Flächen. Dafür liest es im Gegensatz zu SALOME STEP-Dateien und unterstützt auch die in AP 214 enthaltenen Farben und Benennungen der Bauteile.

SALOME unterstützt benannte Flächengruppen. Diese werden interaktiv ausgewählt, mit Unterstützung durch verschiedene Auswahlwerkzeuge, darunter eine umfangreiche Filterfunktionalität. Mit dieser können beispielsweise koplanare Flächen oder alle Flächen auf einer Geometrie automatisch in die Flächengruppe aufgenommen werden. Falls das Netz ebenfalls mit SALOME erstellt wird, können die Flächengruppe auf dieses übertragen werden. Sonst bietet SALOME für den Export der Geometrie zusätzlich eine eigene erweiterte Variante des BRep-Formats an. Das XAO-Format kann zusätzlich benannte Flächengruppen speichern.

Gmsh unterstützt ebenfalls die interaktive Auswahl benannter Flächengruppen. Es bietet aber keine Hilfsfunktionen darüber hinaus. Wie auch bei den beiden anderen Werkzeugen können automatische Auswahlwerkzeuge jedoch über die Programmierschnittstelle umgesetzt werden. Gmsh definiert Schnittstellen immer auf der Geometrie und überträgt sie automatisch auf das Netz. Es kann zudem wie auch FreeCAD Namen und Farben aus STEP AP 214 lesen.

4.3 FE-Netz

4.3.1 Netzformate

FE-Netze bestehen aus Knoten und durch diese definierten Elementen. Im Netzformat werden die Positionen der Knoten und die Zuordnung der Knoten zu den Elementen gespeichert. Für Tetraeder-Netze werden vier Knoten pro Tetraeder-Element gespeichert. Um die Koppelflächen auf dem Tetraeder-Netz definieren zu können, wird das Dreiecksnetz, das dessen Außenfläche bildet, benötigt. Die Dreiecke bilden somit auch die Koppelflächen. Um die Schnittstellen für das Netz zu speichern, wird also auch das Dreiecksnetz benötigt und für jede Schnittstelle die Information, welche Dreiecke zu ihr gehören.

Ein für den Arbeitsablauf geeignetes Netzformat muss also Knoten, Tetraeder, Schnittstellen und dazugehörige Dreiecke speichern. Das ist nicht bei allen Netzformaten möglich, jedoch bei den meisten für die FEM verwendeten.

Es existiert kein verbreiteter Standard für Netzformate. Sattdessen verwenden die meisten Netz-Werkzeuge ihr eigenes Format und können zudem oft noch weitere Formate lesen und schreiben. Da es zahlreiche Netzformate gibt, kann für die Schnittstelle (D3 und D2.1) kein allgemeingültiges Format festgelegt werden.

Stattdessen wird für diese Schnittstelle auf Konvertierung gesetzt. Es gibt zahlreiche Konvertierer für Netzformate, von denen jeder nur in eine eingeschränkte Anzahl von Formaten konvertieren kann. Meshio ist ein junges und wachsendes Projekt, das sich dieser Problemstellung angenommen hat. Es kann derzeit schon zwischen 24 verschiedenen Formaten konvertieren. Bei vielen dieser Formate kann es, falls vorhanden, sowohl die Binär- als auch die Text-Variante konvertieren. Meshio ist zudem komplett in Python implementiert. Es kann sowohl als eigenständiges Werkzeug als auch als Bibliothek verwendet werden. (Schlömer, 2020)

4.3.2 Vernetzer

Vernetzer erstellen aus einer dreidimensionalen Geometrie ein dreidimensionales Netz. Die gängigen Vernetzer für Tetraeder-Netze gehen dabei von außen nach innen vor. Sie teilen zuerst die Linien der Geometrie auf, anschließend teilen sie die Außenflächen in Dreiecke auf. Das so erhaltene Oberflächennetz dient als Startpunkt für das Volumennetz. Es wird erstellt, indem das Volumen von außen nach innen mit Tetraedern aufgefüllt wird, wobei die äußeren Dreiecke die Startpunkte sind.

Um das Oberflächennetz zu erstellen verwenden die Vernetzer entweder direkt die Geometrie als Grundlage oder ein bereits vorhandenes Oberflächennetz. Zweiteres liegt oft in Form einer STL-Datei vor und wird beispielsweise in der Fertigung oder zur Visualisierung der Geometrie verwendet. Es bildet auch gekrümmte Oberflächen näherungsweise genau ab und verwendet hierfür Dreiecke mit teils sehr spitzen Winkeln. Es kann deswegen nicht direkt für die Simulation verwendet werden. Vernetzer, die das Oberflächennetz direkt von der parametrischen Geometrie ableiten, können auf mehr Details der Geometrie zugreifen und sind dewegen robuster.

Sobald das Oberflächennetz erstellt wurde, wird die Geometrie nicht mehr benötigt und das Volumennetz kann automatisch aus dem Oberflächennetz generiert werden. Damit das Netz für die Simulation geeignet ist, sollte es nur wohlgeformte Elemente aufweisen. Diese haben eher stumpfe Winkel und es gibt verschiedenen Maße, um ihre Qualität je nach Anwendung zu bewerten. Solche Qualitätsmaße verwenden die Netzgeneratoren, um damit die Netzform zu optimieren.

(Geuzaine et al., 2009)

In der Wissenschaft wurden verschiedene freie und innovative Vernetzer entwickelt (Jung et al., 2013, S. 8). Eine Auswahl, die auf Robustheit setzt, ist in Tabelle 4.1 aufgelistet. Bei den ersten drei

Vernetzern handelt es sich um erprobte und zuverlässige Vernetzer. Der letzte Vernetzer ist neuer und aufgrund seines innovativen Vorgehens sehr robust.

Tabelle 4.1: Eine Auswahl robuster Open-Source-Vernetzer

Vernetzer	Oberflächennetz	Volumennetz	Quelle
Gmsh	X	X	Geuzaine et al. (2009)
Netgen	X	X	Schöberl (1997)
TetGen		X	Si (2015)
TetWild		X	Hu; Zhou et al. (2018)

Gmsh und Netgen können sowohl ausgehend von der Geometrie ein Oberflächennetz erstellen, als auch daraus ein Volumennetz. Gmsh ist in allen drei in Abschnitt 4.2 eingeführten CAD-Werkzeugen als Vernetzer integriert. SALOME unterstützt zudem Netgen und Gmsh nutzt optional Netgen, um das Volumennetz zu optimieren. Der Vernetzer von Gmsh eignet sich auch besonders für überdurchschnittlich große Netze. Er kann nach dem Stand von 2009 eine Million Tetraeder pro Minute und pro 150 MB Arbeitsspeicher auf damals geläufiger Hardware erstellen. Auf heutige Hardware übertragen kann der Speicherbedarf als konstant und die Rechenzeit geringer angenommen werden. Es besitzt zudem eine hohe Toleranz gegenüber fehlerhaften Eingangsdaten. (Geuzaine et al., 2009)

TetGen und TetWild sind beide auf Tetraeder-Netze spezialisiert. TetGen erzeugt qualitativ hochwertige Netze für die Simulation. Es kann dabei auch mit komplexer Geometrie adaptiv umgehen. Jedoch braucht es als Voraussetzung ein fehlerfreies Oberflächennetz.

Dieses kann aufgrund der in CAD-Daten enthaltenen Ungenauigkeiten und Fehlern (vgl. Abschnitt 4.2.1) auch mit den soeben beschriebenen Vernetzern oft nur durch hohen manuellen Aufwand erzeugt werden. Das selbe trifft auch auf moderne, kommerzielle Vernetzer, wie beispielsweise Ansys oder Hypermesh, zu. Ein fehlerfreies Oberflächennetz kann also nur erstellt werden, falls die Geometrie bereits fehlerfrei ist oder etwaige Fehler durch den Vernetzer ausgeglichen werden können. (Guo et al., 2019)

Gmsh und Netgen können auch für fehlerhafte Geometrien ein Oberflächennetz erstellen. Jedoch weist dieses dann Überschneidungen auf, weswegen daraus kein Volumennetz erstellt werden kann. Vernetzer, die von außen nach innen vorgehen, erzeugen zudem für kleine Flächen, wie sie zum Beispiel durch Verrundungen und Fasen entstehen, auch ein kleinteiliges Oberflächennetz.

Diesem Problem zu begegnen erfordert eine andere Herangehensweise, bei der zuerst ein gleichmäßiges Netz erstellt wird, das anschließend an die Geometrie angeglichen wird. TetWild verwendet ein innovatives Verfahren, das auf diese Weise vorgeht. Mit ihm können beliebige Geometrien vernetzt werden. Es macht dabei keinen Unterschied, ob diese Überschneidungen oder sogar größere Löcher aufweisen. TetWild startet mit einem Oberflächennetz, das unpräzise und fehlerhaft sein kann und an das keine besonderen Ansprüche gestellt werden. Um dieses möglichst robust zu vernetzen, hat TetWild das Ziel, für beliebige Oberflächennetze ein qualitativ hochwertiges Netz zu erstellen, das möglichst ohne weitere Anpassungen zur Simulation verwendet werden kann.

Dies wurde von den Autoren anhand von 10 000 verschiedenen CAD-Geometrien, darunter auch 3D-Scans, demonstriert. (Hu; Zhou et al., 2018)

TetWild produziert äußerst gleichmäßige Netze mit gut geformten Elementen. Kleinteilige Strukturen werden während der Netzgenerierung lediglich angenähert und somit vereinfacht. Der Anwender kann die Toleranz, mit der der originalen Oberfläche gefolgt wird, die gewünschte Elementgröße und -qualität vorgeben. Dies ist optional, da schon die Standardeinstellungen zu guten Ergebnissen führen. Da TetWild das Oberflächennetz annähert und nicht exakt übernimmt, muss entweder die Definition der Schnittstellen erst auf dem Volumennetz erfolgen oder ein Verfahren zum Übertragen der Schnittstellen entwickelt werden. TetWild folgt mit den Rändern der Dreiecke auf der Oberfläche zwar, wie die zuvor beschriebenen Vernetzer, den Kanten auf der Geometrie. Jedoch werden Flächen, die nicht durch Kanten, sondern nur durch Linien getrennt sind, nicht erhalten. TetWild kann deswegen eine Änderung der Flächeninhalte von Koppelflächen bewirken, falls diese nicht durch Kanten abgegrenzt sind, beispielsweise bei aufgeteilten strukturvariablen Koppelflächen.

TetWild ist gerade beim Vernetzen von einfachen Geometrien deutlich langsamer als andere Netzgeneratoren. Es wurde bereits eine schnellere Variante, fTetWild genannt, veröffentlicht (Hu; Schneider et al., 2019). Sie beschleunigt das Verfahren deutlich, ist jedoch immer noch vergleichsweise langsam. Diesen Nachteil macht TetWild jedoch für komplexe Geometrien wieder wett, indem es diese robust und mit hoher Qualität vernetzt, während es sie zugleich vereinfacht und eine aufwendige Fehlerkorrektur unnötig macht. TetWild ist zwar ein relativ junges Projekt, setzt jedoch zur Umsetzung seiner Funktion auf bewährte und schnelle Geometriebibliotheken.

4.4 FEM

Es gibt zahlreiche freie FEM-Programme (Schneider et al., 2019). Ein umfangreicher Vergleich von diesen und kommerziellen FEM-Programmen findet sich in Ladutenko (2020). Für die Umsetzung des Arbeitsablaufs wurden zwei FEM-Programme genauer untersucht.

Code_aster ist ein umfangreiches und robustes FEM-Programm. Es wird vor jeder Veröffentlichung streng auf Korrektheit geprüft. Code_aster verfügt über eine grafische Simulationsumgebung: Salome-Meca. Diese verwendet SALOME, um ein komplettes FEM-Werkzeug bereitzustellen. Sie ist vom Funktionsumfang mit kommerziellen FEM-Programmen, wie Ansys, vergleichbar. Code_aster wird von einer französischen Elektrizitätsgesellschaft entwickelt. Seine umfangreiche Dokumentation und auch seine Skriptsprache sind deshalb auf französisch. Die Dokumentation wird aber auch in einer maschinell übersetzten, englischen Version angeboten. Die französische Skriptsprache verfügt über eine Pythonschnittstelle. (Electricité de France, 2020)

FeniCS verfolgt einen anderen Ansatz. Es stellt eine Berechnungsplattform für partielle Differentialgleichungen bereit. Dafür verwendet es, statt einer eigenen für die FEM ausgelegten Skriptsprache, eine in Python eingebettete mathematische Formulierung, die Unified Form Language (UFL). Sie

wird kompiliert, um sie anschließend schnell ausführen zu können. Mit ihr können auch komplexe Simulationsprobleme verständlich ausgedrückt werden. Um eine partielle Differentialgleichung mit FeniCS zu simulieren, müssen lediglich zwei Schritte durchgeführt werden: (Logg et al., 2010)

1. Die finiten-Elementräume V_D und V_0 werden durch das FE-Netz definiert und die Art der Ansatzfunktionen wird festgelegt.
2. Die partielle Differentialgleichung wird in der Variationsformulierung mit der UFL ausgedrückt.

(vgl. Abschnitt 2.1)

FeniCS besitzt eine ausführliche Anleitung in Buchform (Langtangen et al., 2017). Die Dokumentation seiner Implementierung ist allerdings lückenhaft und teils schwer verständlich. FeniCS wird jedoch aktuell überarbeitet, um es nach 15 Jahren dem aktuellen Stand der Technik anzupassen. Mit der neuen Version sollen neue Funktionen, eine noch bessere Leistungsfähigkeit und eine verbesserte Dokumentation erreicht werden. (FeniCS Project, 2020)

Für die Umsetzung des Arbeitsablaufs muss das FEM-Werkzeug einerseits die thermische Simulation unterstützen und andererseits den Export der FE-Matrizen ermöglichen. Dabei soll es robust sein und auch große Matrizen unterstützen, was auch auf beide Werkzeuge zutrifft.

Code_aster unterstützt nicht nur die thermische Simulation, sondern auch die Strukturmechanik und weitere Phänomene. Der Export der FE-Matrizen ist im Gegensatz zu Ansys mit der integrierten Pythonschnittstelle möglich. Jedoch können so nur die Matrizen exportiert werden, nicht aber die Knotenindizes. Ihr Export ist wie auch bei Ansys nicht vorgesehen. Um auf die Knotenindizes zuzugreifen, muss auf die internen, in Fortran implementierten Datenstrukturen zugegriffen werden.

FeniCS erlaubt es, alle Phänomene, die mit der FEM und partiellen Differentialgleichungen beschrieben werden können, zu simulieren. Der Export der FE-Matrizen ist trivial umzusetzen, da alle Datenstrukturen direkt in Python zur Verfügung stehen. FeniCS ist zudem im Gegensatz zu Code_aster vollständig in Python integriert.

4.5 MOR und Blocksimulation

Die MOR und die Blocksimulation werden hier zusammengenommen, da für beide ein numerisches Berechnungswerkzeug benötigt wird. Um die lineare Algebra in Python zu implementieren, wird NumPy verwendet. Für die darüberhinaus benötigten Funktionalitäten, wie dünnbesetzte Matrizen und lineare Gleichungslöser, wird SciPy verwendet. SciPy ist eine Python-Bibliothek, die Funktionen für wissenschaftliche Berechnungen bereitstellt. (Perez et al., 2011)

Die in Matlab umgesetzte Referenzimplementierung kann mit Hilfe dieser beiden Bibliotheken direkt in Python umgesetzt werden.

Blocksimulation

Die Blocksimulation kann, analog zur Referenzimplementierung, komplett mit den zuvor genannten Bibliotheken umgesetzt werden. Jedoch müssen dann alle benötigten Funktionalitäten, wie beispielweise das Interpolieren zwischen verschiedenen Zeitschritten, auch implementiert werden. Wird stattdessen ein für die Blocksimulation vorgesehenes Programm verwendet, sind diese Funktionen bereits vorhanden.

OpenModelica ist ein ausgereiftes, freies Werkzeug zur digitalen Blocksimulation. Es verwendet die Modelliersprache Modelica, um die Blöcke und ihre Verbindungen zu beschreiben. Das Verhalten der Blöcke wird durch Gleichungen modelliert, die hierfür nicht extra nach den Ein- und Ausgängen aufgelöst werden müssen. Ihnen können zudem physikalische Einheiten zugewiesen und diese können durch das Programm geprüft werden. OpenModelica hat eine grafische Oberfläche zur Blocksimulation und kann somit auch als freie Simulink Alternative angesehen werden. Mit OMPython existiert zudem eine Pythonschnittstelle. (Fritzson et al., 2019)

SimuPy ist weniger umfangreich als OpenModelica, dafür aber komplett in Python implementiert. Es handelt sich um ein minimales Blocksimulations-Framework, das direkt auf robusten Python-Bibliotheken aufbaut. Es stellt Funktionen bereit, mit denen die verschiedenen Blöcke, wie dynamische Systeme, und ihre Verbindungen beschrieben werden können. Die Systeme können entweder direkt zeitdiskret implementiert werden oder durch eine symbolische Gleichung beschrieben werden. SimuPy ist zwar noch ein recht junges Projekt mit nur einem Entwickler, seine Grundlagen können aber als robust angenommen werden, da sie direkt auf bewährte Python-Bibliotheken aufbauen. Es verwendet NumPy und SciPy für die numerische Simulation und das Computeralgebrasystem SymPy zum Lösen symbolischer Gleichungen. Durch die Verwendung von SimuPy ist es möglich, für die Blocksimulation auf das gesamte Python-Ökosystem zurückzugreifen. Auf diese Weise können beispielweise komplexe, empirische Ansätze für die Randbedingungen implementiert oder Optimierungsaufgaben gelöst werden. In Kombination mit FeniCS ist es so auch möglich, FE-Modelle direkt in der Blocksimulation, auch in Verbindung mit FE-MOR-Modellen, zu berechnen. (Margolis, 2020)

4.6 Visualisierung

Dieser Abschnitt behandelt Werkzeuge, mit denen Temperaturfelder oder auch beliebige andere Daten, die auf FE-Netzen definiert sind, zusammen mit diesen visualisiert werden können.

Viele solcher Werkzeuge verwenden das Visualization Toolkit (VTK). Es ist eine Bibliothek, die verschiedenste Werkzeuge für die wissenschaftliche Datenverarbeitung, Visualisierung und Datenanalyse bereitstellt. Sie besitzt also weit mehr Funktionen als für die hier betrachtete Visualisierung notwendig. VTK ist erprobt und vielfach angewendet. Es hat auch eine Pythonschnittstelle. Jedoch ist diese teils unnötig komplex, da sie sich eng an den zugrundeliegenden C++ Funktionen orientiert (Sullivan et al., 2019).

ParaView baut auf VTK auf und erweitert dessen Funktionen. Dabei bietet es vor allem eine grafische Oberfläche zur Visualisierung und Analyse mit Python. Diese, wie auch ParaView selbst, bietet ebenfalls eine Pythonschnittstelle. ParaView ist zudem in SALOME integriert. Es heißt dort ParaVis, ist mit den restlichen Modulen von SALOME verbunden und wird für die Analyse der Simulationsergebnisse verwendet. (Quammen, 2015; OPEN CASCADE, 2020b)

Mayavi baut ebensfalls auf VTK auf und ist eine Python-Bibliothek. Mit ihr können direkt aus Python heraus mit nutzerfreundlichen Funktionen Visualisierungen erstellt werden. Mit Mayavi können zudem auch eigenständige interaktive Visualisierungsprogramme realisert werden. (Ramachandran et al., 2011)

PyVista ist eine weitere Python-Bibliothek, die VTK verwendet. Sie stellt ebenfalls nutzerfreundliche Funktionen bereit, mit denen sowohl schnell Prototypen, als auch aufwendige Visualisierungen mit großen Datensätzen direkt aus Python realisert werden können. (Sullivan et al., 2019)

Auch Gmsh ermöglicht die interaktive Visualisierung und Analyse von FE-Simulationsergebnissen. Es verwendet als einziges vorgestelltes Werkzeug nicht VTK, sondern implementiert die Visualisierung selbst. Bei Gmsh geschieht der komplette Simulationsablauf von der Geometrievorbereitung, über die Netzerstellung, bis zur Analyse unter einer Oberfläche. Mit der Pythonschnittstelle von Gmsh kann auch die Visualisierung, ebenso wie der Rest von Gmsh, gesteuert werden. (Geuzaine et al., 2009)

5 Implementierung des Arbeitsablaufs

In diesem Kapitel wird gezeigt, wie der Arbeitsablauf unter Ausnutzung der Modularisierung mit freier Software umgesetzt werden kann.

Der entwickelte Arbeitsablauf unterstützt den Anwender bei der Auswahl der Schnittstellen und leitet daraus zugleich eine Geometrievereinfachung ab, um kleinteilige und für das thermische Verhalten irrelevante Strukturen zu entfernen. Nun erstellt er das FE-MOR-Modell voll automatisch. Dabei ist er robust und verarbeitet auch fehlerhafte Geometrien. Er teilt die strukturvariablen Koppelflächen zudem automatisch auf.

Die Implementierung umfasst den ganzen Arbeitsablauf: von der Erstellung der FE-MOR-Modelle bis zur Blocksimulation. Es wird zudem ein Ansatz vorgestellt, um die Dimension n des FE-MOR-Modells durch einen Abgleich mit dem FE-Modell zu bestimmen.

Der implementierte Arbeitsablauf wird anhand von zwei Beispielen demonstriert:

1. Komplexe und fehlerhafte Dreh-Schwenk-Tisch-Baugruppe einer 5-Achs-WZM, um das Vorgehen zum Erstellen der FE-MOR-Modelle realistisch bewerten zu können.
2. Einfache Beispielgeometrie, in Form eines Würfels, um das FE-MOR-Modell und die Blocksimulation durch einen Abgleich mit der Referenzimplementierung auf Korrektheit zu prüfen.

In Abschnitt 5.1 wird die Umsetzung des Arbeitsablaufs und seiner Komponenten vorgestellt. Dann wird in Abschnitt 5.2 zunächst das Vorgehen zur Geometrievorbereitung und Definition der Schnittstellen behandelt. In Abschnitt 5.3 wird die Netzerstellung und in Abschnitt 5.4 das Erstellen des FE-Modells und des FE-MOR-Modells erläutert. Anschließend werden in Abschnitt 5.5 die Blocksimulation beschrieben. Hierbei erfolgt ein Abgleich mit der Referenzimplementierung und ein Vergleich des FE-Modells mit dem FE-MOR-Modell. Abschließend geht Abschnitt 5.6 auf die Visualisierung der Ergebnisse ein.

5.1 Überblick

Für die Implementierung werden die selben Kriterien und Schwerpunkte gesetzt, die für die Recherche in Abschnitt 4.1 entwickelt wurden. Die Kriterien sind, dass die Schnittstellen möglichst allgemeingültig sind und der Arbeitsablauf flexibel, robust, fehlertolerant und skriptbar ist.

Allgemeingültige Schnittstellen für die Geometrie und das Netz wurden bereits in der Recherche festgelegt. Die restlichen Schnittstellen sind Matrizen, Vektoren und Skalare. Für diese werden durch die Implementierung mit Python verschiedene Formate unterstützt und können je nach Anwendung gewählt werden.

Der Arbeitsablauf ist bereits durch die Modularisierung (vgl. Abschnitt 3.2) flexibel und die verwendeten Werkzeuge können durch die klar abgegrenzten Schnittstellen ausgetauscht werden. Robustheit und Fehlertoleranz werden in der Implementierung umgesetzt. Sie erfolgt für den automatisierten Teil des Arbeitsablaufs mit Python.

Der Arbeitsablauf zum Erstellen der FE-MOR-Modelle soll weitestgehend automatisiert werden. Das Erstellen der Blocksimulation und die Ergebnisanalyse erfolgen anwendungsspezifisch und können nur in diesem Kontext, falls sinnvoll, automatisiert werden.

Um diesen Teil des Arbeitsablaufs zu automatisieren, muss zunächst ermittelt werden, welche Arbeitsschritte nicht automatisierbar sind und deshalb manuell erfolgen müssen. Dies trifft auf einen Arbeitsschritt zu: Das Auswählen und Benennen der Koppelflächen für die Schnittstellendefinition muss, ebenso wie das Erstellen der Blocksimulation, manuell erfolgen.

Alle weiteren Arbeitsschritte werden automatisiert. Sie benötigen dann nur ihre jeweiligen Eingangsdaten. Da die Geometrievereinfachung nur eingeschränkt automatisch erfolgen kann, wird sie automatisiert, indem sie ebenfalls das Ergebnis der manuellen Schnittstellendefinition mitnutzt. Der einzige zwingend manuelle Arbeitsschritt wird somit zum Automatisieren eines zweiten, nur teilweise automatisierbaren Arbeitsschrittes, mitgenutzt. Dieses Vorgehen ermöglicht zugleich eine Unterstützung bei der Auswahl der Koppelflächen. So müssen dann nicht mehr alle Koppelflächen ausgewählt werden, sondern nur einige wesentliche, was in Anbetracht komplexer WZM-Baugruppen eine deutliche Arbeitserleichterung darstellt.

Abbildung 5.1 zeigt die Implementierung des Arbeitsablaufs. Sie ist modular gehalten. Jedes Modul umfasst eine oder mehrere Funktionen, welche ihrerseits mit einem oder mehreren Werkzeugen umgesetzt werden. Die allgemeingültigen Schnittstellen zwischen den Modulen ermöglichen eine äußerst flexible Verwendung des entwickelten Arbeitsablaufs. Je nach vorhandenen Werkzeugen und Erfahrung des Anwenders kann der ganze Arbeitsablauf oder auch nur einzelne Module übernommen werden. Der Arbeitsablauf kann so flexibel in eine vorhandene Werkzeugkette integriert werden und an neue Problemstellungen angepasst werden.

Geometrie vorbereiten und Schnittstellen definieren

Der Arbeitsablauf wird in einen manuellen und einen automatischen Teil aufgeteilt. Dafür werden die Module Geometrie vorbereiten und Schnittstellen definieren ebenfalls aufgeteilt (vgl. Abschnitt 3.2). Nur der erste Schritt wird manuell durchgeführt. Er wird mit SALOME umgesetzt, da es von allen in Abschnitt 4.2.2 vorgestellten CAD-Werkzeugen die besten Funktionalitäten zur interaktiven Geometrievereinfachung, -korrektur und vor allem Schnittstellendefinition bietet.

Die Schnittstellen werden mit SALOME nur grob definiert. Dafür werden einige wesentliche Flächen ausgewählt, durch die die Koppelflächen klar abgegrenzt sind. Anschließend muss der Anwender noch entscheiden, was mit den restlichen Flächengruppen geschehen soll. In diesem Schritt können diese auch gelöscht oder aufgefüllt werden, wodurch die Geometrie für die Simulation vereinfacht wird.

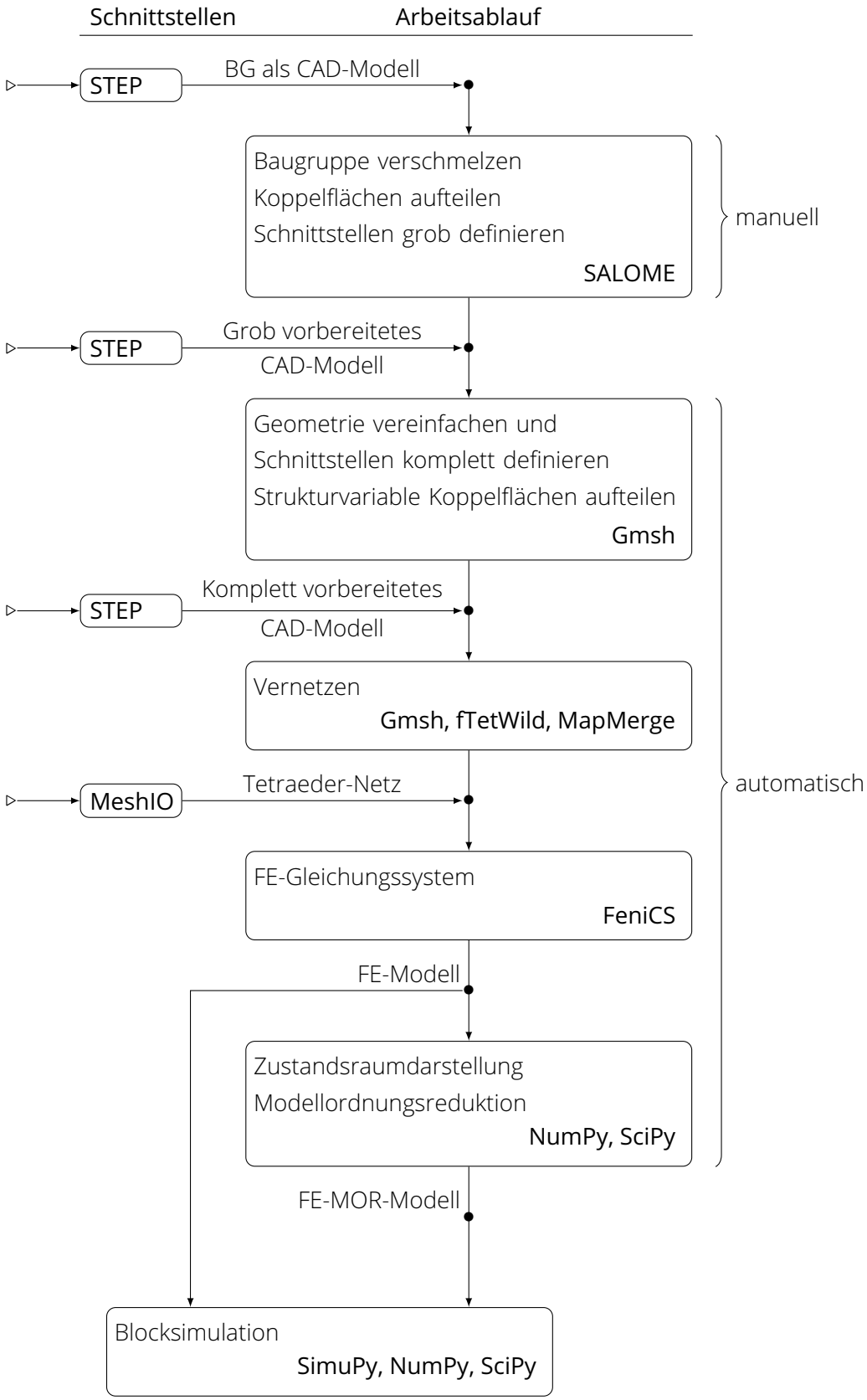

Abbildung 5.1: Implementierung des Arbeitsablaufs. Die Module enthalten die zur Implementierung genutzten Werkzeuge. Zwischen den Modulen sind die Schnittstellen ● eingetragen. Durch die allgemeingültigen Schnittstellen ▷ kann der entwickelte Arbeitsablauf flexibel angewendet werden.

Der automatische Teil wird mit Gmsh umgesetzt, da es im Gegensatz zu SALOME direkt in Python integriert werden kann, weniger Ressourcen benötigt und einfacher zu installieren ist. Diese Entscheidung wurde auch getroffen, um eine Integration in einen bestehenden CAD-Arbeitsablauf in Kombination mit der automatischen Geometrievorbereitung zu ermöglichen.

Gmsh vereinfacht die Geometrie zunächst aufgrund der zuvor getroffenen Entscheidungen und definiert die Schnittstellen anschließend vollständig. Strukturvariable Koppelflächen werden nun automatisch in gleich große Teilflächen aufgeteilt. Der Anwender muss hierzu lediglich deren Anzahl und, falls nötig, deren Anfang festlegen. Dieser Schritt kann ebenso wie das Aufteilen der restlichen Koppelflächen interaktiv in SALOME geschehen. Das automatische Vorgehen ermöglicht es jedoch, die Anzahl der Teilflächen und die Auflösung der strukturvariablen Komponenten zu variieren und damit ihren Einfluss auf die Simulation zu untersuchen.

Vernetzen

Die Vernetzung kann aufgrund der Verwendung von TetWild äußerst robust und fehlertolerant durchgeführt werden. Es sind somit auch für komplexe Baugruppen keine aufwendigen Vorarbeiten notwendig, um das Netz zu erstellen.

Zunächst wird mit Gmsh ein Oberflächennetz erzeugt. Gmsh ist für diesen Zweck sehr gut geeignet, da es aufgrund von Open CASCADE Technology robust STEP-Dateien lesen kann und daraus auch bei fehlerhafter Geometrie ein Oberflächennetz erstellen kann. Dieses ist qualitativ hochwertig, kann aber ebenso Fehler enthalten.

Dies ist jedoch irrelevant, da TetWild selbst für komplexe und fehlerhafte Geometrien ein einwandfreies Tetraeder-Netz erzeugt. TetWild ist hierbei so fehlertolerant, dass sogar größere Löcher ausgeglichen werden. Dies wird für die Geometrievereinfachung noch weiter ausgenutzt, indem für die Simulation irrelevante Geometriedetails einfach gelöscht werden. Für die Implementierung wird die neuere, schnellere Variante von TetWild, fTetWild, genutzt.

Gmsh überträgt die Koppelflächen automatisch auf das Oberflächennetz. Da TetWild jedoch nur dessen Form übernimmt, können die Koppelflächen nicht direkt übertragen werden. Es wird deswegen ein Werkzeug für diese Aufgabe entwickelt: MapMerge leitet vom Tetraeder-Netz ein Oberflächennetz ab und überträgt darauf die Koppelflächen.

Sowohl TetWild als auch MapMerge bewirken, dass sich der Flächeninhalt der Koppelflächen ändern kann. Die Änderungen werden deshalb während des Arbeitsablaufs ausgegeben. So kann geprüft werden, ob diese zu groß werden und die Simulation verfälschen könnten. Denn die Koppelflächen sind ebenso wie das Volumen der Geometrie ein wesentlicher Einflussfaktor für die Simulation, da über sie die Wärme übertragen wird.

FE-Gleichungssystem, MOR und Blocksimulation

Das FE-Gleichungssystem wird direkt in Python mit FeniCS aufgestellt und kann anschließend exportiert oder direkt weiterverwendet werden. Die MOR erfolgt ebenfalls in Python unter Nutzung von NumPy und SciPy.

Auch die Blocksimulation wird mit Python durchgeführt. Sie verwendet SimuPy, um die benötigten Funktionen bereitzustellen und die Blocksimulation flexibel zu gestalten. Da das FE-Modell, ebenso wie das FE-MOR-Modell, in Python vorliegt, kann es auch in der Blocksimulation verwendet werden. Seine iterative Berechnung während der Simulation erfolgt mit FeniCS. FeniCS kann, wie auch beliebige andere Pythonfunktionen, als Teilmodell in die SimuPy-Blocksimulation eingebunden werden.

Somit ist es auch möglich, einzelne Strukturbauteile statt modellordnungsreduziert mit der vollen FEM in die Blocksimulation zu integrieren, falls die zusätzliche Auflösung benötigt wird. Dieses Vorgehen ermöglicht es zudem, die Dimension n des FE-MOR-Modells durch einen Abgleich mit dem FE-Modell zu bestimmen.

Falls das FE-Modell in der Blocksimulation verwendet wird, ändern sich die Anforderungen an das FE-Netz. Dieses sollte nun nicht zu viele Knoten besitzen, um weiterhin berechenbar zu sein und eine simulationsgerechte Qualität aufzuweisen. Beide Anforderungen werden vom zuvor beschriebenen Vernetzungsmodul bereits erfüllt, was zugleich das hier entwickelte Vorgehen auch allgemein für die FEM interessant macht.

5.2 Geometrie

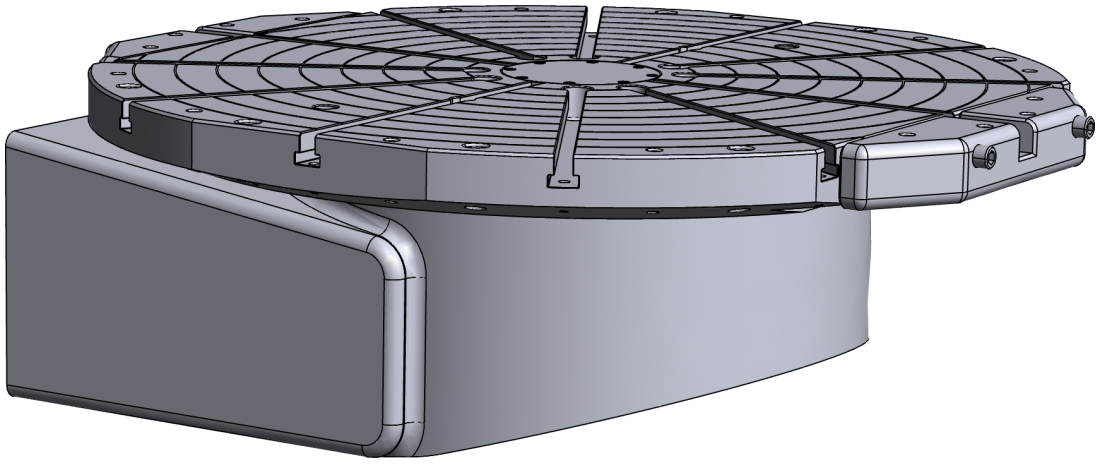

Abbildung 5.2: Reale Beispielgeometrie: Baugruppe eines Dreh-Schwenk-Tisches einer 5-Achs-WZM

Abbildung 5.2 zeigt die Dreh-Schwenk-Tisch-Baugruppe, anhand der der Arbeitsablauf zum Erstellen ordnungsreduzierter thermischer Modelle von Werkzeugmaschinen, der FE-MOR-Modelle, demonstriert wird. Der Drehtisch stammt von einer 5-Achs-WZM. Er wurde in Creo erstellt und als Solidworks-Datei exportiert. In dieser Form lag er für die Verwendung in dieser Arbeit vor. Aufgrund der häufig bei einem Export dieser Art (vgl. Abschnitt 4.2.1) auftretenden Probleme, weist

der Drehtisch an seinen vielen Fasen, Nuten und Rillen zahlreiche Fehler auf. Diese führen in diesem Fall im CAD-Werkzeug noch nicht zu Problemen. Sie verhindern jedoch ein Vernetzen mit den meisten in Abschnitt 4.3.2 vorgestellten Vernetzern.

Um den Arbeitsablauf nicht nur anhand eines Bauteils, dem Drehtisch, zu demonstrieren, wurde er zu einer realistischen WZM-Baugruppe erweitert. Dafür wurden weitere Bauteile hinzugefügt: Seitliche Backen, Schrauben, ein vereinfachter Antrieb und ein Lager. Zudem wurde ein vereinfachter Schwenkarm als Gegenstück konstruiert. Dies wurde mit Solidworks durchgeführt und die komplette Baugruppe als STEP AP 214 exportiert.

Die Baugruppe weist aufgrund der Konvertierung und des verwendeten Austauschformats verschiedene Fehler auf (vgl. Abschnitt 4.2.1). Zudem besitzt sie zahlreiche kleinteilige Strukturen, wie beispielsweise die Bohrungen des Drehtisches, die Ölkanäle des Lagers und die zahlreichen Schraubenköpfe. All das macht einen realistischen Test des entwickelten Arbeitsablaufs möglich. Die Geometriefehler erschwerten die Entwicklung des Arbeitsablaufs. Da solche Fehler real, aufgrund von Konvertierung und der Verwendung von Austauschformaten, auftreten können, ermöglichen auch sie eine realistische Bewertung. Darüber hinaus machten sie eine innovative Lösung notwendig. Diese stellt eine Verbesserung des Stands der Technik dar und wurde erst durch die Verwendung freier Software möglich.

5.2.1 Manuelle Vorbereitung und Schnittstellendefinition

Die Baugruppen im STEP-Format werden in SALOME nach dem Import zunächst in ihre beiden strukturfesten Teilbaugruppen aufgeteilt. Beide werden dann im nativen BRep-Format wieder exportiert und anschließend seperat für die Simulation vorbereitet.

Die manuelle Vorbereitung erfolgt mit SALOME Shaper, dem neuen parametrischen Modellierer von SALOME. Jede Baugruppe wird zunächst importiert und daraufhin partitioniert. Dies ermöglicht es, anschließend die Bauteile zu einem einzigen Volumen mit der Funktion Union zu „verschmelzen"[1]. Der Drehtisch ist nun bereits fertig und die Schnittstellen können definiert werden. Die Koppelflächen des Armes müssen noch aufgeteilt werden, damit die Schnittstelle zum Antrieb definiert werden kann. Dafür wird auf der Baugruppenebene eine Skizze erstellt. Mit dieser werden die Abmaße des Antriebs vom Drehtisch auf den Schwenkarm übertragen und dort die Flächen passend aufgeteilt.

Die Schnittstellen werden nun auf beiden Baugruppen grob definiert. Das Ergebnis ist in Abbildung 5.3 dargestellt. Es werden nur die farbig markierten Flächen ausgewählt und je Farbe zu einer Flächengruppe zusammengefasst. Jede Flächengruppe erhält den Namen der entsprechenden Schnittstelle des FE-MOR-Modells. Die Oberseite des Tisches wurde mit dem Filter für koplanare Flächen ausgewählt, wodurch jeweils nur eine Teilfläche angewählt werden muss. Bei der groben Definition der Schnittstellen werden nur größeren Flächen so ausgewählt, dass durch sie

[1] Das „Verschmelzen" könnte auch im automatischen Teil des Arbeitsablaufs geschehen, jedoch kann Gmsh die Baugruppe mit dem Frästisch nicht fehlerfrei „verschmelzen". Zudem ändern sich hierbei die Außenflächen der Geometrie, was Einfluss auf die Definition der Koppelflächen hat. Aus diesem Grund sollten die Koppelflächen möglichst nach der Geometrievorbereitung definiert werden.

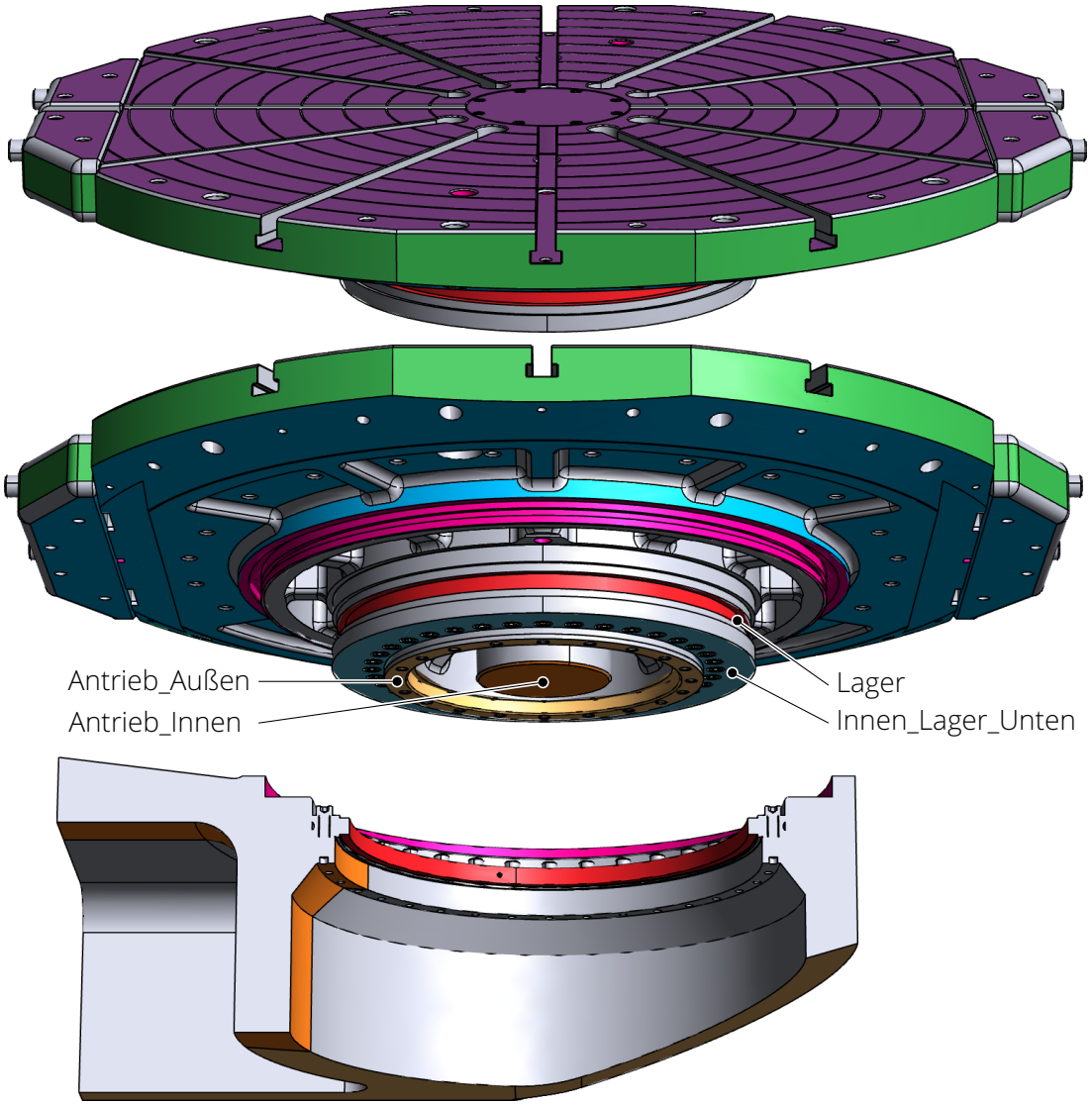

Abbildung 5.3: Drehtisch und Schwenkarm mit grob definierten Schnittstellen. Diese sind mit einer Farbe je Schnittstelle markiert.

die Schnittstellen eindeutig abgegrenzt sind. Aus diesem Grund müssen die meisten Bohrungen nicht berücksichtigt werden; durchgehende Bohrungen, die Ober- und Unterseite verbinden, jedoch schon. Auch muss beispielsweise der Rand des Drehtisches durchgehend ausgewählt werden, um Ober- und Unterseite zu trennen. Kleine Flächen und die Taschen und Rippen auf der Unterseite des Tisches werden nicht ausgewählt und werden anschließend automatisiert behandelt.

Die beiden Baugruppen sind nun fertig manuell vorbereitet und werden im SALOME-eigenen XAO-Format exportiert, da dieses die Schnittstellen mit abspeichern kann.

5.2.2 Automatische Vorbereitung und Schnittstellendefinition

Die restlichen Flächen, die noch keiner Schnittstelle zugewiesen wurden, werden nun mit einem hierfür entwickelten Algorithmus analysiert:

Der Algorithmus

1. ermittelt alle Flächen, die keiner Schnittstelle zugeordnet wurden
2. gruppiert diese Flächen in zusammenhängende Flächengruppen
3. ermittelt die zu jeder Flächengruppe benachbarten Schnittstellen
4. ermittelt, wie viele Linien die Flächengruppen mit der jeweiligen Schnittstelle verbinden
5. untersucht, welche der Flächengruppen eine geschlossene Begrenzung haben
6. gruppiert gleichartige Flächengruppen

Angewendet auf den in Abbildung 5.3 dargestellten Drehtisch mit grob definierten Schnittstellen werden 250 zusammenhängende Flächengruppen gefunden. Diese können in 28 gleichartige Flächengruppen zusammengefasst werden. Die ersten 5 davon sind in Tabelle 5.1 aufgelistet. Die jeweils benachbarten Schnittstellen sind in Abbildung 5.3 eingezeichnet..

Tabelle 5.1: Gleichartige Flächengruppen, die manuell keiner Schnittstelle zugeordnet wurden. Dargestellt sind die ersten 5 von 28, die für den Drehtisch ermittelt wurden. Die Spalten enthalten die Anzahl der gleichartigen Flächengruppen, die jeweils benachbarten Schnittstellen, die Anzahl an Verbindungen zu diesen, ob die Verbindung geschlossen ist und die für die Gruppe gewählte Aktion.

	Anzahl	Benachbarte Schnittstellen	Verb.	Geschl.	Aktion
1	102		0	Ja	Löschen
2	20	Antrieb_Außen	4	Ja	Füllen
3	1	Antrieb_Außen, Antrieb_Innen	4, 2	Ja	Innen_Antrieb
4	1	Antrieb_Außen, Innen_Lager_Unten	4, 4	Ja	Innen_Lager_Unten
5	36	Innen_Lager_Unten	2	Ja	Füllen
	…	und 23 weitere			

Der Anwender muss nun entscheiden was mit den 28 gleichartigen Flächengruppen geschehen soll. Dafür wird für jede Gruppe eine von drei Aktionen gewählt:

1. Schnittstelle: Die Flächengruppen werden einer bestehenden oder neuen Schnittstelle zugeordnet, indem als Aktion der Name der Schnittstelle angegeben wird.
2. Füllen: Die Flächengruppe wird gelöscht und das entstandene Loch aufgefüllt. Dies ist nur möglich, falls die Begrenzung der Flächengruppe geschlossen ist.

3. Löschen: Die Flächengruppe wird gelöscht und das dadurch möglicherweise entstandene Loch wird beim Vernetzen von TetWild anschließend geschlossen und geglättet.
4. Ignorieren: Die Flächengruppe wird nicht mitvernetzt. Dies hat die selbe Wirkung auf das Netz wie Löschen, die Flächen werden aber erhalten. So stehen diese anschließend noch für die Visualisierung zur Verfügung.

Die in Zeile 1 der Tabelle 5.1 aufgeführten Flächengruppen haben keine benachbarten Schnittstellen. Es handelt sich bei den 102 gleichartigen Flächengruppen um interne Volumen. Sie ergeben sich beispielsweise durch Hohlräume hinter den Schrauben oder zu den Anbauteilen. Sie werden gelöscht, da sie vergleichsweise klein und somit für das thermische Verhalten unbedeutend sind. Um die Größen der internen Volumen einzuschätzen, werden diese vom Analyseskript ausgegeben.

Die nächste Gruppe (Zeile 2) besteht aus 20 Flächengruppen und grenzt an den äußeren Antriebsflansch. Es handelt sich um die dort vorhandenen Gewindebohrungen. Sie werden aufgefüllt, da sie dies bei montiertem Antrieb auch wären. Die nächsten beiden Gruppen (Zeile 3 und 4) sind größere zusammenhängende Bereiche auf der Unterseite des Drehtisches. Sie wurden zuvor wegen ihrer Kleinteiligkeit nicht manuell ausgewählt. Jetzt können sie anhand der benachbarten Schnittstellen einfach neuen und bestehenden Schnittstellen zugeordnet werden. Bei der in Zeile 5 aufgeführten Gruppe handelt es sich um versenkte Schraubenköpfe. Sie werden aufgefüllt, um die Geometrie zu vereinfachen. Die Schnittstelle Innen_Lager_Unten wurde bei der manuellen Definition extra für diesen Zweck ausgewählt; diese wäre sonst auch der Gruppe in Zeile 4 zugeordnet worden.

Mit diesem Vorgehen ist es möglich, sowohl die Schnittstellen komplett zu definieren ohne zuvor alle Flächen oder auch alle Schnittstellen auswählen zu müssen; zugleich kann auf diese Weise auch eine Geometrievereinfachung durchgeführt werden. Der Analysealgorithmus ermöglicht eine effiziente und anschauliche Bearbeitung der nicht zugewiesenen Flächen.

Dieser Arbeitsschritt kann wie hier dargestellt als Tabelle implementiert werden. Auf diese Weise kann er beispielsweise auch direkt in das CAD-Werkzeug integriert werden. So könnte er den Anwender schon bei der interaktiven Auswahl der Schnittstellen unterstützen. Die ermittelten nicht zugewiesenen Gruppen können dann auch direkt farblich auf der Geometrie hervorgehoben werden, was das Vorgehen interaktiver und verständlicher macht.

Die gewählten Aktionen werden von einem zweiten Algorithmus automatisiert angewandt. Das Ergebnis für den Drehtisch und den dazugehörigen Schwenkarm ist in Abbildung 5.4 dargestellt.

Die Schnittstellen sind nun vollständig definiert. Die vielen kleinen Flächen auf der Unterseite werden so alle erfasst. Ebenso werden auch alle im Drehtisch vorhandenen Löcher einer Schnittstelle zugeordnet. Die Baugruppen sind nun auch direkt vereinfacht. Sie weisen keine internen Volumen mehr auf und Gewindebohrungen und Schraubenköpfe wurden aufgefüllt. Die Schraubenköpfe am Schwenkarm wurden nicht vereinfacht, da sie nicht eingelassen sind, und so bewertet werden kann, wie der Vernetzer mit solch kleinen Strukturen umgeht.

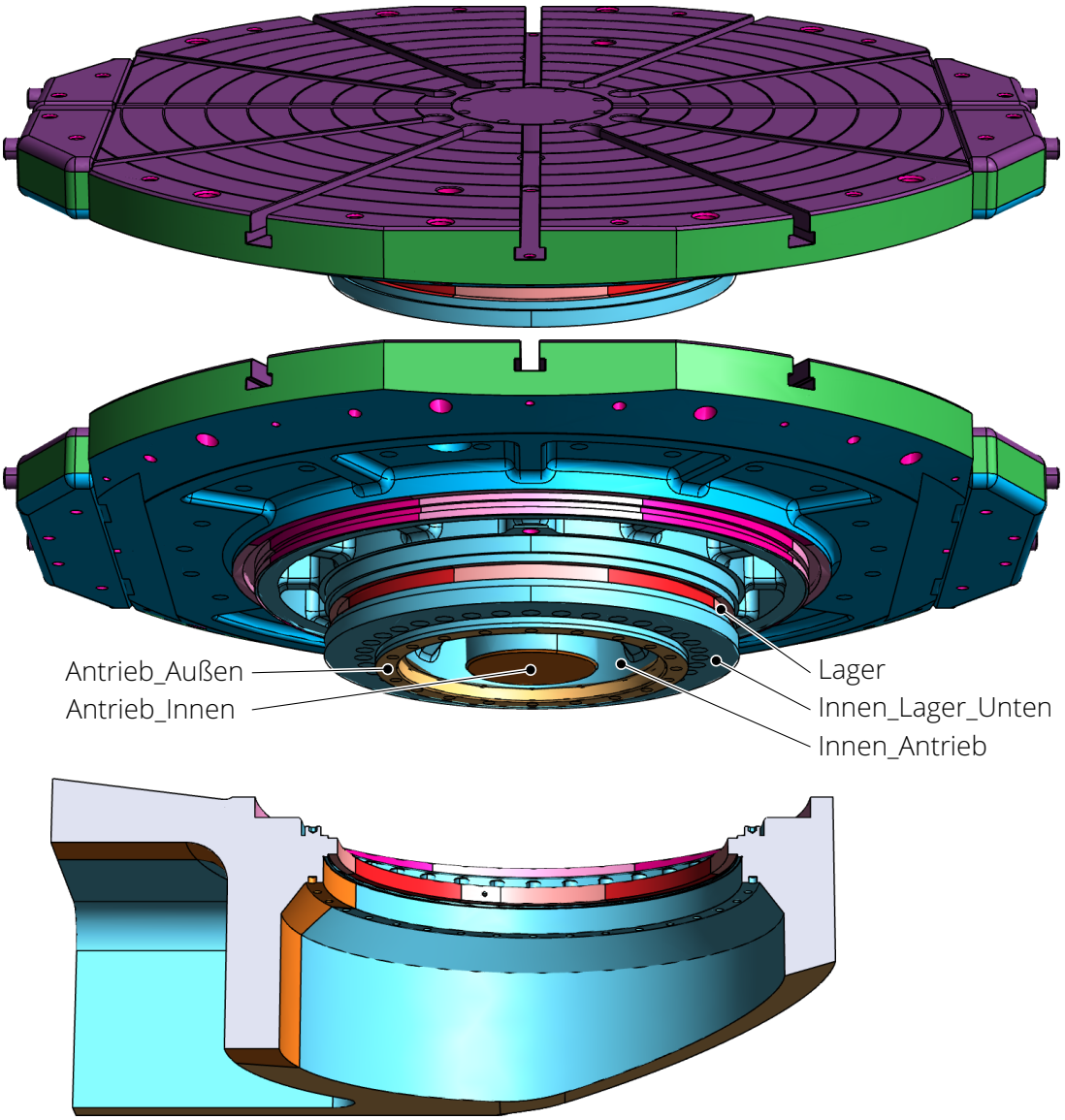

Abbildung 5.4: Drehtisch und Schwenkarm mit komplett definierten Schnittstellen, vereinfachter Geometrie und aufgeteilten strukturvariablen Koppelflächen. Die Schnittstellen sind mit unterschiedlichen Farben hervorgehoben.

5.2.3 Automatisches Aufteilen der strukturvariablen Koppelflächen

In Abbildung 5.4 wurden die strukturvariablen Koppelflächen am Lager und an der Dichtung bereits aufgeteilt. Dafür muss vom Anwender angegeben werden, ob die Bewegung und damit die Aufteilung linear oder rotatorisch erfolgt. Zudem werden die Anzahl der Teilflächen und bei rotatorischer Bewegung der Startwinkel mit Bezugsachse angegeben.

Anhand dieser Angaben und der zur Schnittstelle gehörenden Flächen erfolgt das Aufteilen automatisiert. Der hierfür entwickelte Algorithmus wird nun für den linearen Fall vorgestellt:

1. Der die Flächen umschließende Quader wird bestimmt. Seine längste Kante gibt die Aufteilungsrichtung und die Länge der Teilflächen vor.
2. Die kleinste Fläche des Quaders wird nun in diese Richtung und um diese Länge extrudiert.
3. Mit dem daraus neu entstandenen Quader werden die Flächen aufgeteilt und es ergibt sich die erste Teilfläche.
4. Schritt 3 wird für jede weitere Teilfläche wiederholt, wobei die zu extrudierende Fläche zuvor um die Länge einer Teilfläche verschoben wird.
5. Die Teilflächen werden benannt, indem jeweils an den Namen der Schnittstelle die Nummer der Teilfläche angehängt wird.

Für den rotatorischen Fall funktioniert der Algorithmus ähnlich. Zunächst wird eine Fläche durch den Mittelpunkt zum Rand bestimmt. Ihre Richtung ergibt sich durch den Startwinkel. Diese Fläche wird anschließend kreisförmig extrudiert und der restliche Ablauf ist analog zum linearen Fall.

5.3 FE-Netz

Das FE-Netz wird erstellt, indem zuerst mit Gmsh ein Dreiecksoberflächennetz (2D-Netz) von der CAD-Geometrie abgeleitet wird. Aus diesem wird anschließend mit TetWild ein Tetraeder-Netz (3D-Netz) erstellt.

Gmsh legt die Größe der Tetraeder-Elemente in Abhängigkeit von der Geometrie fest. Sie kann dann durch verschiedene Parameter und Funktionen genauer angepasst werden. Die Hauptstellschraube ist der Elementgrößenfaktor, mit dem alle Tetraeder-Elemente zugleich angepasst werden. Für diesen wird 0,2 verwendet. Das ist die einzige Stelle, an der von der Standardeinstellung von Gmsh abgewichen wird. Denn Gmsh nähert runde Geometrien normalerweise recht grob und bietet aus diesem Grund optional eine adaptive Anpassung an Rundungen. Diese ist jedoch vergleichsweise langsam. Deswegen wird stattdessen ein feineres Netz über den Elementgrößenfaktor erstellt. Dieses gibt die Geometrien gut wieder und wird anschließend mit TetWild sowieso vereinfacht. Das Dreiecksoberflächennetz für den Drehtisch ist in Abbildung 5.5 abgebildet. Es bildet die Geometrie gut ab und ist dabei wie erwartet recht kleinteilig. Kleinere Zylinder werden jedoch immer noch etwas grob genähert.

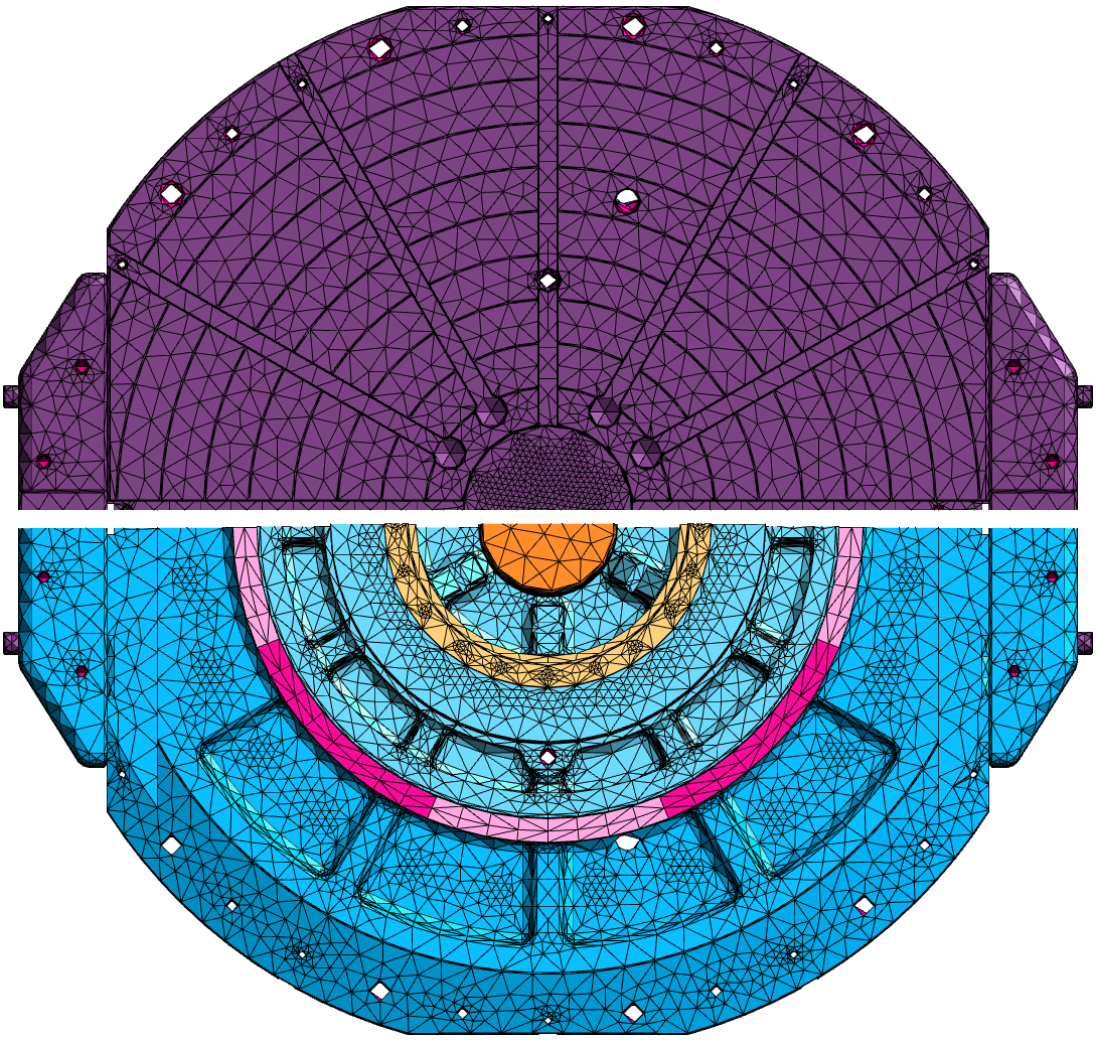

Abbildung 5.5: Dreiecksoberflächennetz des Drehtisches. Es dient als Grundlage für das Tetraeder-Netz und besteht aus 43 392 Dreiecken. In der oberen Hälfte der Abbildung ist die Oberseite des Tisches dargestellt, in der unteren Hälfte die Unterseite.

Bei der Verwendung von TetWild kann die Toleranz, mit der dem Oberflächennetz gefolgt wird, die gewünschte Elementgröße und -qualität eingestellt werden. Die ersten beiden werden als Bruchteil der Diagonalen des die Geometrie begrenzenden Quaders angegeben. Die Qualität wird im intern von TetWild verwendeten Qualitätsmaß angegeben (Conformal AMIPS energy; vgl. Hu; Zhou et al. (2018)). TetWild wendet die drei Parameter wie folgt an: Es erzeugt zunächst ein gleichmäßiges Hintergrundnetz mit der vorgegebenen Elementgröße, in das die Geometrie in Form ihres Oberflächennetzes eingebettet wird. Anschließend gleicht es das Hintergrundnetz gemäß der geforderten Toleranz an das Oberflächennetz der Geometrie an. Das so erhaltene Tetraeder-Netz wird abschließend iterativ optimiert, bis alle Elemente mindestens die geforderte Elementqualität erfüllen.

TetWild erzeugt mit Standardeinstellungen ein qualitativ hochwertiges Netz, das jedoch auch recht fein ist. Um einschätzen zu können, ob auch ein gröberes Netz die Geometrie noch ausreichend genau wiedergibt, wird außer dem feinen auch ein grobes Netz erstellt. Die Einstellungen für das feine und das grobe Netz sind in Tabelle 5.2 aufgelistet. Die Einstellungen für das feine Netz ent-

sprechen den TetWild Standardeinstellungen.

Tabelle 5.2: Einstellungen von TetWild für das grobe und das feine Tetraeder-Netz

	Toleranz	Elementgröße	Elementqualität
Grob	$2 \cdot 10^{-3}$	0,1	20
Fein	$1 \cdot 10^{-3}$	0,05	10

Das grobe Netz für die Beispielgeometrie ist in Abbildung 5.6 und das feine Netz in Abbildung 5.7 abgebildet. Beide vereinfachen das Oberflächennetz in Abbildung 5.5, aus dem sie abgeleitet wurden. Sie bilden beide die komplexe Geometrie gut ab. Das grobe Netz benötigt dafür nur etwa ein Fünftel der Tetraeder des feinen Netzes. Es weist jedoch auch weniger gleichmäßige Elemente auf. Beide Netze bestehen überwiegend aus sehr gleichmäßigen Elementen und geben die Geometrie mit minimalem Volumenfehler wieder (unter 1 %). Außerdem ist gut erkennbar, wie TetWild die Geometrie vereinfacht. Kleine Strukturen, wie beispielsweise die Rillen auf der Oberfläche des Tisches, werden einfach entfernt.

In Tabelle 5.3 ist die Dauer der Netzerstellung und die Anzahl der Dreiecke und Tetraeder der Netze für den Drehtisch und den Schwenkarm aufgelistet. Bei den Tetraeder-Netzen (3D) entsprechen die Dreiecke (2D) der Oberfläche.

Tabelle 5.3: Dauer der Netzerstellung und Anzahl der Dreiecke und Tetraeder für die Beispielgeometrien

	Zeit in s	Dreiecke	Tetraeder
Tisch			
Gmsh 2D	16	43 392	-
TetWild 3D grob	38	12 738	20 744
TetWild 3D fein	77	33 976	111 243
Arm			
Gmsh 2D	2,4	20 189	-
TetWild 3D grob	14	5774	9340
TetWild 3D fein	27	12 128	36 328

Für den Drehtisch werden für das feine Netz insgesamt gut eineinhalb Minuten und für das grobe Netz knapp eine Minute benötigt. Das grobe Netz hat sowohl für den Tisch als auch für den Arm weit weniger Elemente.

Tabelle 5.4 enthält zwei Qualitätsmaße, um die Qualität der Netze zu vergleichen. Die Maße werden von Gmsh berechnet. Das erste Maß, der Inverse Gradient Error (IGE), misst den Fehler des Gradienten der FE-Lösung. Das zweite Maß, die Inverse Condition Number (ICN), misst die Kondition der Steifigkeitsmatrix, also wie sehr deren Lösung von Störungen der Eingangsdaten beeinflusst wird. Die Maße werden in Johnen et al. (2016) vorgestellt. Beide Maße können zwischen 0 und 1 liegen, wobei ein höherer Wert für eine höhere Qualität steht.

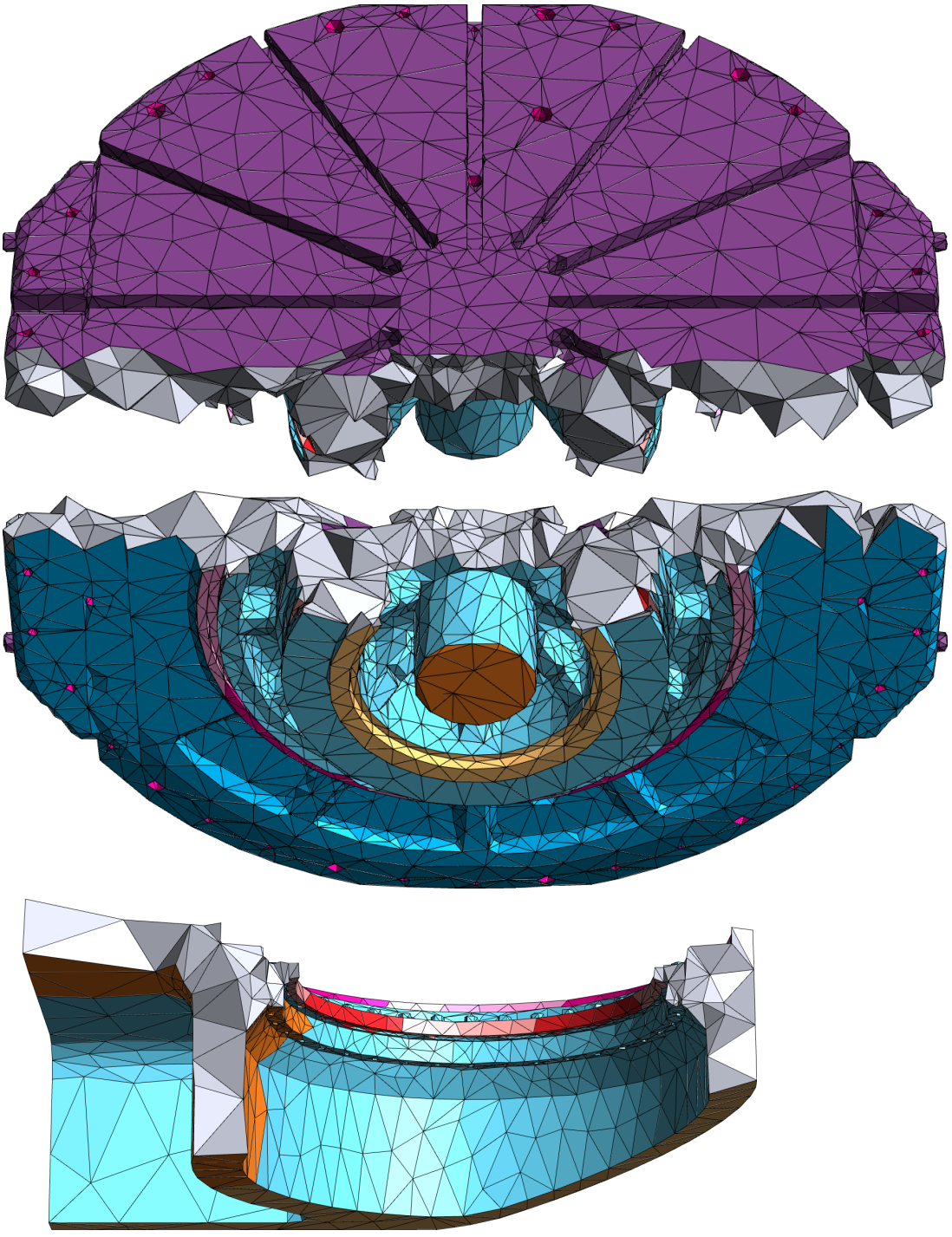

Abbildung 5.6: Grobes Tetraeder-Netz: Der Drehtisch wurde mit 20 744 Tetraedern und der Schwenkarm mit 9340 Tetraedern vernetzt.

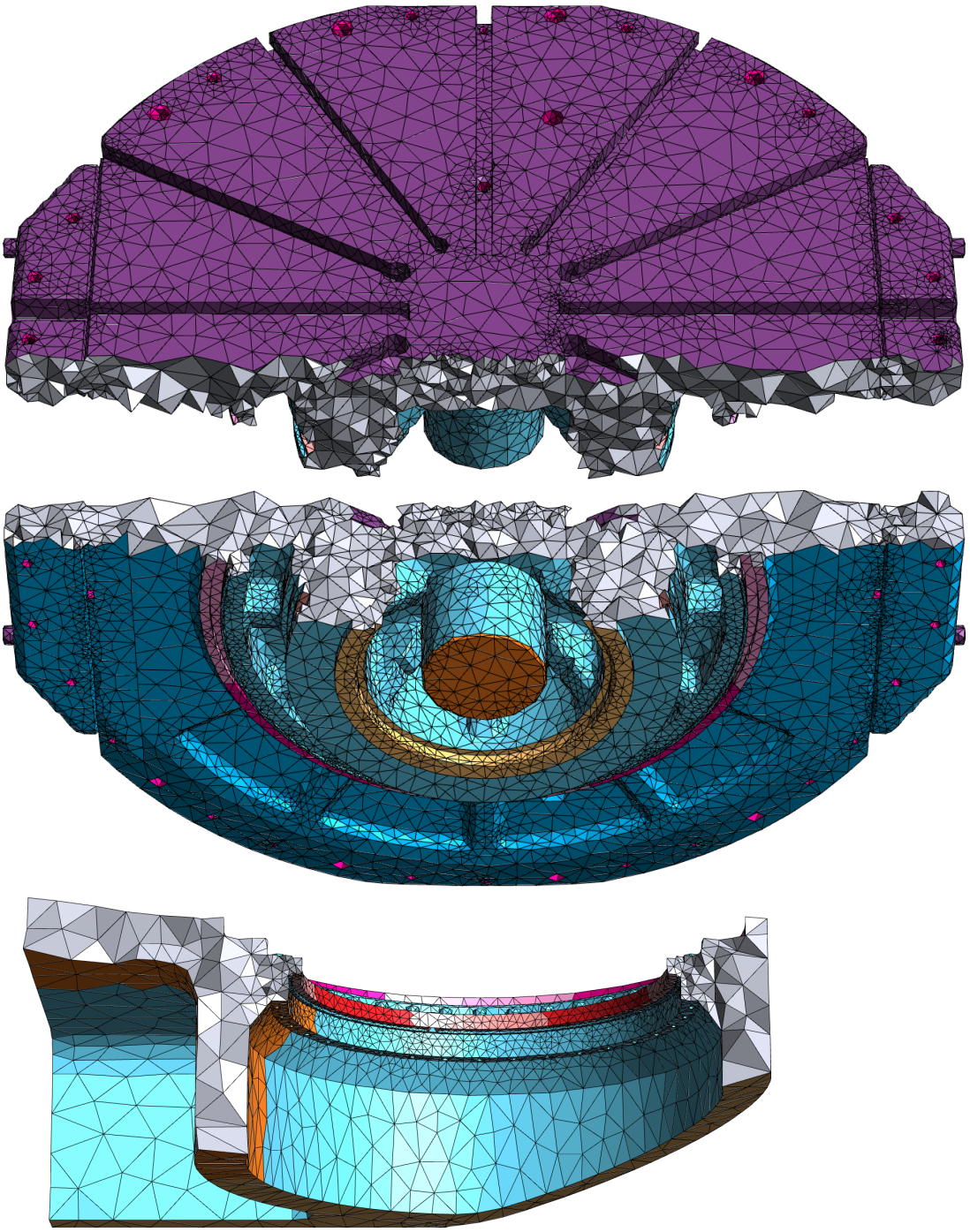

Abbildung 5.7: Feines Tetraeder-Netz: Der Drehtisch wurde mit 111 243 Tetraedern und der Schwenkarm mit 36 328 Tetraedern vernetzt.

Tabelle 5.4: Die beiden Qualitätsmaße IGE und ICN für alle Netze. Für jedes Qualitätsmaß ist das Minimum, der Mittelwert und das Maximum aller Tetraeder-Elemente angegeben.

	IGE			ICN		
	Min	Mittel	Max	Min	Mittel	Max
Tisch						
Gmsh 2D	0	0,8	1	0	0,6	1
TetWild 3D grob	0,1	0,7	1	0,2	0,6	1
TetWild 3D fein	0,3	0,8	1	0,4	0,8	1
Arm						
Gmsh 2D	0	0,9	1	0	0,8	1
TetWild 3D grob	0,1	0,7	1	0,2	0,7	1
TetWild 3D fein	0,3	0,8	1	0,4	0,8	1

Die mit TetWild erstellten Tetraeder-Netze weisen im Vegleich zu den Oberflächennetzen alle eine höhere Mindestqualität und damit weniger ungünstig geformte Elemente auf. Zudem besitzt selbst das grobe Netz für komplexe Geometrie eine zum sehr feinen Oberflächennetz vergleichbare Qualität. Das feine Netz übertrifft das Oberflächennetz sogar meist.

MapMerge und Flächenfehler durch das Vernetzen

Bevor abschließend die Flächenfehler der Koppelflächen durch das Vernetzen diskutiert werden, wird zunächst das Verfahren vorgestellt, mit dem die Schnittstellen auf das Tetraeder-Netz übertragen werden. Es wird MapMerge genannt, da es zunächst die Koppelflächen überträgt und anschließend das Tetraeder-Netz zusammen mit einem Oberflächennetz, auf dem die Schnittstellen definiert sind, abspeichert.

MapMerge verwendet einen sogenannten AABB-Baum, um schnell zu bestimmen, welches Dreieck des originalen Oberflächennetzes einem beliebigen Punkt am nächsten liegt. MapMerge wurde, wie auch TetWild, in C++ als Kommandozeilenwerkzeug implementiert. Auf diese Weise kann es auf die selbe Geometriebibliothek, Geogram, wie TetWild zugreifen. Mit dieser kann die benötigte Funktionaliät direkt umgesetzt werden und MapMerge ist auch für große Netze sehr schnell.

MapMerge erstellt zunächst einen AABB-Baum für das originale Oberflächennetz, das inklusive Schnittstellen zuvor mit Gmsh erstellt wird. Anschließend erstellt MapMerge für das Tetraeder-Netz ein neues Oberflächennetz. Nun prüft MapMerge für jedes Dreieck davon, welches Originaldreieck ihm am nächsten liegt. Es fragt dafür jeweils den Mittelpunkt über den AABB-Baum ab. Das neue Dreieck bekommt dann die selbe Schnittstelle zugewiesen wie das Original. Schließlich wird das Tetraeder-Netz zusammen mit dem dazu passenden Oberflächennetz mit Schnittstellen abgespeichert.

In Tabelle 5.5 sind die Flächenfehler in % für einige Schnittstellen des Drehtisches angegeben. Darunter sind auch die jeweils acht Teilflächen der strukturvariablen Schnittstellen.

Tabelle 5.5: Flächenfehler in % für einige beispielhafte Schnittstellen des Drehtisches

Schnittstelle	Gmsh 2D	TetWild 3D grob	TetWild 3D fein
Antrieb_Innen	3,3	8,8	3
Antrieb_Außen	0,2	1,7	1,3
Innen_Antrieb	1	2	1,9
Oberseite	0,7	5	4,3
Bohrungen_Durch	10	12,9	11,2
Lager (gesamt)	0,1	2,9	0,4
Lager (aufgeteilt): 1-8	0,1	0 bis 10,8	0,6 bis 5,9
Dichtung (gesamt)	0,1	4,1	3,1
Dichtung (aufgeteilt): 1-8	0,1	0,8 bis 6,7	2,4 bis 4

Die größten Flächenfehler ergeben sich bei den durchgehenden Bohrungen, da deren teils feine Strukturen grob vom Oberflächennetz angenähert werden. Solche Details werden für die thermische Analyse mit der FEM jedoch sowieso häufig vereinfacht und sollten somit unproblematisch sein. Die nächst höchsten Fehler ergeben sich bei den Teilflächen der strukturvariablen Koppelflächen. Ihr Flächeninhalt variiert deutlich, was aufgrund des Vorgehens zu erwarten war. Die Flächenfehler der zugrundeliegenden Koppelflächen sind jedoch gering. Die Schnittstelle Antrieb_Innen weist aufgrund der ungenauen Näherung durch das Oberflächennetz ebenfalls einen erhöhten Flächenfehler auf. Sollen solche Fehler vermieden werden, muss ein noch feineres Oberflächennetz mit Gmsh erstellt werden.

Die restlichen Flächenfehler sind als gering einzuschätzen und im Rahmen der für die thermische Simulation vertretbaren Abweichung. Der leicht höhere Fehler auf der Oberseite ergibt sich aus den hier erfolgten Geometrievereinfachungen. Ein Teil davon ist jedoch einem Zehntel-Millimeter Spalt zwischen Antrieb und Drehtisch geschuldet. Beim manuellen Grobdefinieren der Schnittstellen wurde die Außenfläche des Antriebs ausgewählt. Diese befindet sich teilweise im Spalt und wird von TetWild vereinfacht.

Der implementierte Vernetzungsablauf zeigt dem Anwender die Oberflächenfehler, die Volumenfehler und die Qualitätsmaße an und ermöglicht so eine Einschätzung der Netzqualität.

Beide erstellten Netze eignen sich gut für die thermische Simulation und ihre Oberflächen- und Volumenfehler sind im Rahmen der hierfür zulässigen Abweichungen. Aufgrund der geringeren Elementeanzahl wird das grobe Tetraeder-Netz für die weitere Simulation empfohlen, da es schneller berechnet werden kann und darüber hinaus auch eine Simulation des FE-Modells ermöglicht. Das feine Netz kann gewählt werden, falls ein geringerer Flächenfehler notwendig ist.

Mit dem beschriebenen Verfahren und anderen Einstellungen für TetWild können auch noch einfachere Netze mit weniger Elementen erzeugt werden. Diese werden dann jedoch stark vereinfacht und weisen große Flächenfehler auf.

5.4 FE-Modell und FE-MOR-Modell

Das FE-Modell wird mit FeniCS erstellt. Dafür wird die Variationsformulierung der instationären Wärmeleitungsgleichung benötigt. Diese wurde in Abschnitt 2.1 hergeleitet. Für das FE-Modell werden, wie in Abschnitt 3.2 beschrieben, die folgenden FE-Matrizen benötigt:

$$C_T, K_{T\lambda}, f_1, \ldots, f_{j+k} \tag{5.1}$$

Ihre Variationsformulierung ergibt sich zu (vgl. Gleichung 2.12, 2.13 und 2.14):

$$C_T = \int_\Omega pv \, \mathrm{d}\Omega \tag{5.2}$$

$$K_{T\lambda} = \int_\Omega \nabla p \nabla v \, \mathrm{d}\Omega \tag{5.3}$$

$$f_i = \int_{\Gamma i} v \, \mathrm{d}\Gamma_i \tag{5.4}$$

Die Materialparameter c, ρ und λ kommen hier noch nicht vor. Mit ihnen werden die Matrizen erst danach multipliziert, wodurch es möglich ist, die Materialparameter zu variieren ohne neue FE-Matrizen zu berechnen. Die Lasten werden auch noch nicht mitbeachtet. Sie werden erst während der Simulation iterativ aktualisiert.

Die FE-Matrizen und die dazugehörigen Knotenindizes für die Schnittstellendefinition werden mit FeniCS und der UFL in Python implementiert (vgl. Kapitel 4.4):

```
1  V = FunctionSpace(Netz, 'P', 1)

2  p = TrialFunction(V)
3  v = TestFunction(V)

4  UFL_CT = p*v*dx
5  UFL_KT = dot(grad(p), grad(v))*dx
6  UFL_f  = [ v*ds(i) for i in range(Schnittstellen)]

7  CT = assemble(UFL_CT)
8  KT = assemble(UFL_KT)
9  fi  = [ assemble(f) for f in UFL_f]

10 Knotenindizes =  vertex_to_dof_map(V)
```

Die Implementierung mit Python ist kurz und auf das Wesentliche beschränkt. In Zeile 1 wird der Funktionsraum durch das FE-Netz und die Art der Ansatzfunktionen definiert. Für die Ansatzfunktionen werden lineare, finite Elemente verwendet. In den Zeilen 2 und 3 werden die Ansatzfunktionen p und die Testfunktionen v zusammen mit ihren finiten Elementräumen V_D und V_0 durch den Funktionsraum V festgelegt.

In den Zeilen 4 bis 6 werden die beiden Matrizen $\boldsymbol{C}_T$, $\boldsymbol{K}_{T\lambda}$ und die Vektoren $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{j+k}$ durch die UFL ausgedrückt. dx steht für das Integral über das gesamte Gebiet Ω. ds(i) wird so definiert, dass es pro Schnittstelle i das Integral über den jeweiligen Rand Γ_i bildet. Da ein $\boldsymbol{f}_i$ pro Eingang des FE-Modells benötigt wird, ist UFL_fi in Zeile 6 eine Liste von Vektoren mit je einem Vektor $\boldsymbol{f}_i$ pro Schnittstelle und ds(i).[2]

In den Zeilen 7 bis 9 werden die Matrizen und Vektoren zusammengebaut. Anschließend liegen die Matrizen dünnbesetzt und die Vektoren in einer Liste vor. Die Knotenindizes werden bei FeniCS mit der Funktion vertex_to_dof_map für den Funktionsraum V ausgegeben. Dies geschieht in Zeile 10.

Die Wärmekapazitätsmatrix $\boldsymbol{C}_T$ wird anschließend noch diagonalisiert, indem sie mit dem Einheitsvektor $\boldsymbol{e}$ multipliziert wird. Dadurch wird die Matrix spaltenweise summiert. Das Verfahren wird „mass-lumping" genannt und ergibt für lineare Ansatzfunktionen die selbe Approximationsgenauigkeit wie ohne „mass-lumping" (Jung et al., 2013, S. 562). Dieser Schritt ist notwendig, damit die Wärmekapazitätsmatrix für die Zustandsraumdarstellung invertiert werden kann (vgl. Abschnitt 2.2). Um die Wärmestrom-Eingänge auf ihre Koppelfläche zu beziehen, werden während der Simulation die Koppelflächeninhalte A_{Ki} benötigt. Diese können bereits mit Gmsh ermittelt werden. Um den Arbeitsablauf flexibel zu halten, werden sie jedoch direkt mit FeniCS berechnet. Dafür wird ein Wert von 1 über den jeweiligen Rand Γ_i integriert.

Die Implementierung zeigt, dass das FE-Gleichungssystem weit einfacher mit FeniCS als mit Ansys oder Code_aster erstellt und exportiert werden kann. Auch die Knotenindizes können so einfach ausgegeben werden und erfordern keine umfangreiche Exportroutine.

FE-MOR-Modell

Das FE-MOR-Modell wird mit NumPy und SciPy erstellt. Dabei wird wie in Kapitel 2.2 beschrieben vorgegangen. Der Arnoldiprozess zum Ermitteln der Transformationsmatrix wird ebenfalls direkt in Python implementiert.

Um ein FE-Modell des Drehtisches unter Verwendung des groben FE-Netzes zu erstellen, wird, sobald die UFL-Ausdrücke einmalig kompiliert wurden, nur ein Bruchteil einer Sekunde benötigt. Das dazugehörige FE-MOR-Modell wird anschließend in etwa 8 Sekunden generiert.

Der entwickelte Arbeitsablauf zum Erstellen der FE-MOR-Modelle ist also nicht nur robust und fehlertolerant, sondern auch leistungsfähig genug, um selbst komplexe Geometrien schnell zu

[2] Der verwendete Ausdruck mit den eckigen Klammern ist eine Python List Comprehension. Mit ihr wird beschrieben, wie eine Liste durch eine for Schleife erstellt wird.

verarbeiten. Möglich machen das die verwendeten Pythonbibliotheken mit ihren schnellen Kernfunktionen in C++, C oder Fortran. Gleichzeitig ist die Pythonimplementierung verständlich und leicht anpassbar.

5.5 Blocksimulation und Abgleich mit der Referenzimplementierung

Die Blocksimulation wird mit SimuPy umgesetzt. Um das FE-MOR-Modell zu simulieren, muss es zuvor in eine zeitdiskrete Form gebracht werden. Hierfür wird, wie auch in der Referenzimplementierung, das implizite Eulerverfahren verwendet.

Als Grundlage für die zeitdiskrete Form dient das FE-MOR-Modell aus Gleichung 2.27 (vgl. Kapitel 2.2). Die zeitdiskrete Zustandsgleichung ergibt sich zu:

$$\hat{\boldsymbol{T}}_i = (\hat{\boldsymbol{T}}_{i,-1} + \Delta t\, \hat{\boldsymbol{b}}_i\, u_i)\, (\boldsymbol{E} + \Delta t\, \hat{\boldsymbol{A}}_\lambda + \Delta t \sum_{i=1}^{k} \hat{\boldsymbol{A}}_{\alpha i}\, u_{\alpha i})^{-1} \tag{5.5}$$

Δt gibt die Länge eines Zeitschritts vor. $\hat{\boldsymbol{T}}_{i,-1}$ ist der Temperaturvektor aus dem letzten Zeitschritt t_{-1} für das jeweilige reduzierte Teilmodell i. Mit der Ausgangsgleichung ergibt sich der superpositionierte Ausgangsvektor $\hat{\boldsymbol{y}}$:

$$\hat{\boldsymbol{y}} = \sum_{i=1}^{j+k} \hat{\boldsymbol{T}}_i\, \hat{\boldsymbol{C}}_i \tag{5.6}$$

Die beiden Gleichungen 5.5 und 5.6 werden in SimuPy als dynamisches System implementiert und können anschließend mit weiteren Blöcken zur Blocksimulation verbunden werden.

Um auch das FE-Modell in der Blocksimulation berechnen zu können, muss es ebenfalls zeitdiskret vorliegen. Hierfür wird ebenso das implizite Eulerverfahren genutzt. Aus Gleichung 2.15 (vgl. Abschnitt 2.2) ergibt sich damit die Zustandsgleichung zu:

$$c\rho \boldsymbol{C}_T \boldsymbol{T} + \Delta t\, (\lambda \boldsymbol{K}_{T\lambda} + \sum_{i=1}^{k} \alpha_i \boldsymbol{K}_{T\alpha i})\, \boldsymbol{T} = c\rho \boldsymbol{T}_{-1} \int_\Omega \boldsymbol{v}\, \mathrm{d}\Omega + \sum_{i=1}^{j} \Delta t\, \dot{q}_i \boldsymbol{f}_i + \sum_{i=1}^{k} \Delta t\, \alpha_i T_{Ui}\, \boldsymbol{f}_i \tag{5.7}$$

$\boldsymbol{T}_{-1}$ ist wiederum der Temperaturvektor aus dem letzten Zeitschritt t_{-1}. Damit die Ausgänge des FE-Modells gleich den Ausgängen des FE-MOR-Modells sind, wird die Ausgangsgleichung so definiert, dass sie die mittlere Temperatur für alle Schnittstellen i des FE-MOR-Modells im Ausgangsvektor $\boldsymbol{y}$ ausgibt. Dafür wird $\boldsymbol{T}$ über dem jeweiligen Rand Γ_i integriert und anschließend auf den Koppelflächeninhalt A_{Ki} bezogen:

$$\boldsymbol{y}_i = \frac{\int_{\Gamma i} \boldsymbol{T}\, \mathrm{d}\Gamma_i}{A_{Ki}} \tag{5.8}$$

Das FE-Modell wird auch mit SimuPy als dynamisches System modelliert. Die Zustandsgleichung 5.7 und die Ausgangsgleichung 5.8 werden mit FeniCS in der UFL umgesetzt. Somit werden für die iterative Berechnung die schnellen und optimierten Funktionen von FeniCS genutzt.

Das FE-Modell und das FE-MOR-Modell können nun beide in der Blocksimulation verwendet werden. Damit ist es möglich eine Gesamtsimulation sowohl mit dem FE-Modell als auch mit dem FE-MOR-Modell zu rechnen. So kann abgeschätzt werden, ob die Dimension n des FE-MOR-Modells für die Gesamtsimulation passend gewählt wurde. Es zeigt sich dabei, ob das FE-MOR-Modell im Vergleich zum FE-Modell und im Kontext der Gesamtsimulation die betrachteten thermischen Phänomene ausreichend genau wiedergibt oder ob die Dimension n angepasst werden muss.

5.5.1 Vergleich mit der Referenzimplementierung

Die Korrektheit des implementierten Arbeitsablaufs zum Erstellen der FE-MOR-Modelle und deren anschließender Blocksimulation wird anhand einer einfachen Beispielgeometrie und darauf aufbauenden Blocksimulation überprüft. Das Beispiel wurde gewählt, da es auch in der Referenzimplementierung umgesetzt vorlag.

Die Beispielgeometrie ist ein 1 m^3 großer Würfel aus Baustahl. Er hat folgende Materialparameter: $c = 434\,J\,kg^{-1}\,K^{-1}$, $\rho = 7850\,kg\,m^{-3}$, $\lambda = 60{,}5\,W\,K^{-1}\,m^{-1}$. Die Starttemperatur des Würfels zu Beginn der Simulation entspricht der Umgebungstemperatur: $T_0 = T_U = 20\,°C$. Die Umgebungstemperatur bewirkt an allen Außenflächen mit Ausnahme der Unterseite einen Wärmeübergang. Dieser wird über den Wärmeübergangskoeffizienten mit einem Ansatz, der Konvektion und Strahlung abbildet, modelliert:

$$\alpha = 2,7 + 2,2\,|\,T_W - T_U\,|^{0,33} \tag{5.9}$$

T_W ist die mittlere Temperatur auf den fünf Außenflächen des Würfels. Diese bilden die Koppelfläche der ersten Schnittstelle (Außenseite). Die Unterseite ist die Koppelfläche der zweiten Schnittstelle. Hier wird der Würfel konstant mit einem Wärmestrom $\dot{Q} = 1\,kW$ erwärmt. Die Simulationsdauer t_f beträgt 10 Stunden und der Würfel wird mit dem Zeitschritt $\Delta t = 10\,s$ simuliert. Das FE-MOR-Modell des Würfels wird auf die Dimension $n = 10$ reduziert.

Für den Vergleich des FE-Modells mit der Referenzimplementierung wurde das dort mit Ansys erstellte Netz exportiert und anschließend mit MeshIO eingelesen. Der Vergleich ergab, dass beide Modelle bis auf einzelne numerische Fehler, durch die manche Matrixelemente etwas abweichen, übereinstimmen.

Die FE-MOR-Modelle konnten nicht direkt verglichen werden, da Ansys und FeniCS die FE-Matrizen jeweils anders sortieren. Stattdessen wurden direkt die Ergebnisse der Blocksimulation verglichen. Die Ergebnisse der Blocksimulation stimmen mit der Referenzimplementierung überein. Die entwickelte Implementierung ist somit korrekt.

5.5.2 Vergleich des FE-Modells mit dem FE-MOR-Modell

Anhand des selben Beispiels wurde zudem ein Vergleich des FE-Modells mit dem FE-MOR-Modell durchgeführt. Die Simulation mit dem FE-Modell dauert 62 s und mit dem FE-MOR-Modell 2,5 s. Selbst für dieses einfache Beispiel ergibt sich durch die MOR eine signifikante Reduzierung der Simulationsdauer. Das Ergebnis der beiden Simulationen ist in Abbildung 5.8 visualisiert.

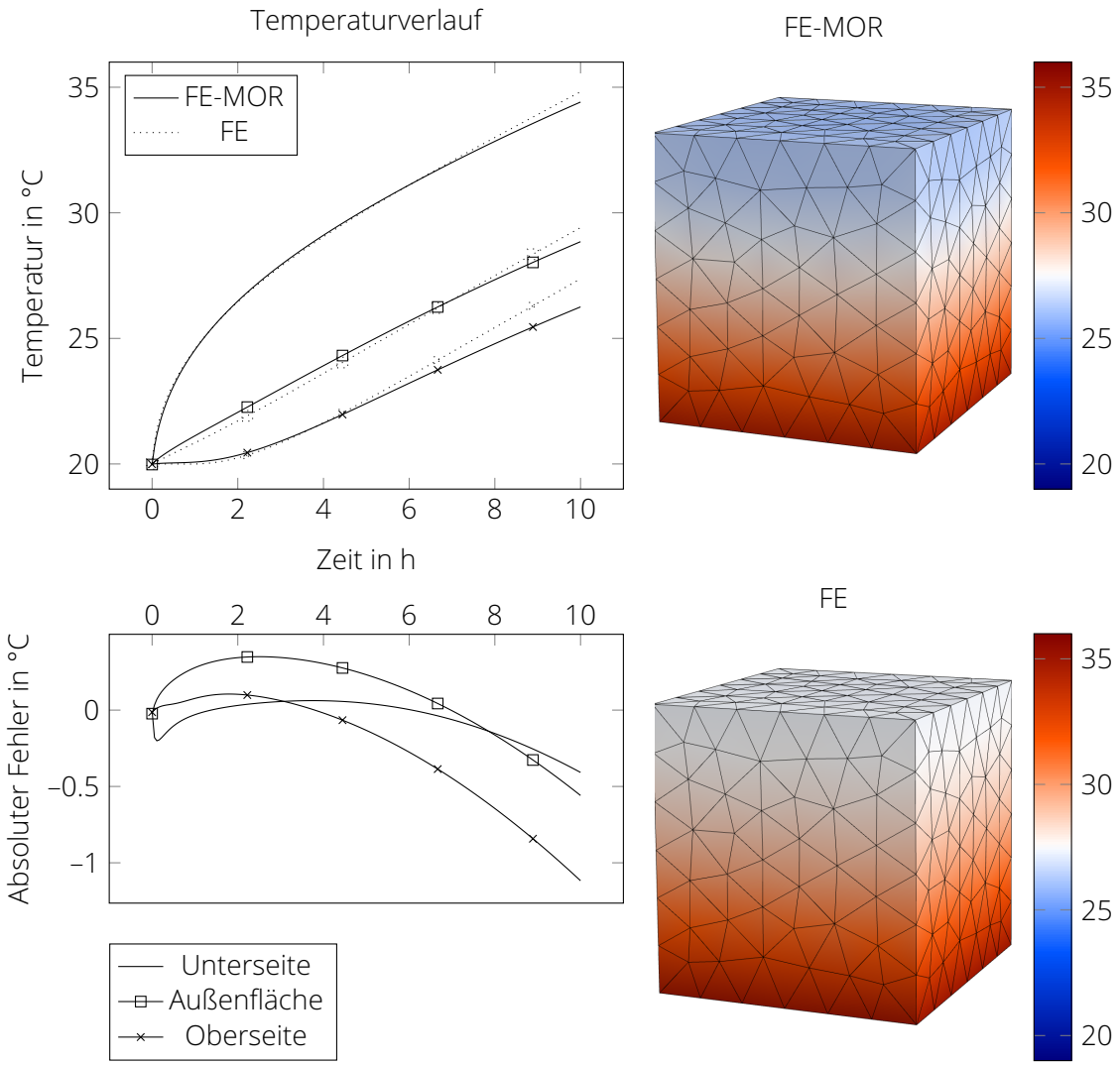

Abbildung 5.8: Verlauf der Temperaturen während der Simulation mit dem FE-MOR-Modell und dem FE-Modell für die Unterseite, die Außenfläche ohne Unterseite und die Oberseite des Würfels. Auf der rechten Seite ist die Temperaturverteilung am Ende der Simulation für beide Modelle visualisiert.

Die Verläufe der beiden Modelle sind zunächst ähnlich und driften über den Simulationszeitraum leicht auseinander. Die deutlichsten Abweichungen ergeben sich für die Oberseite des Würfels. Denn diese ist keine eigene Schnittstelle des Blockmodells, sondern wurde aus dem rücktransformierten Temperaturfeld berechnet. Die Abweichungen ergeben sich, da die Außenflächen des Würfels zu einer Schnittstelle zusammengefasst wurden, obwohl sie aufgrund der einseitigen Erwärmung unterschiedliche Temperaturniveaus aufweist.

Dadurch fällt die Temperaturverteilung am Ende der Simulation bei den beiden Modellen unterschiedlich aus und das FE-Modell weist eine gleichmäßigere Erwärmung auf. Die mittlere Temperatur der Außenfläche stimmt jedoch zwischen beiden Modellen größtenteils überein. Die Reduktion auf zwei Schnittstellen beeinflusst also für dieses Beispiel die resultierende Temperaturverteilung.

Dieser Effekt muss bei der Definition der Schnittstellen des FE-MOR-Modells beachtet werden, da sonst eine möglicherweise zu große Vereinfachung vorgenommen wird, die dann zu ungenauen Simulationsergebnissen führt. Das FE-Modell in der Blocksimulation zum Vergleich zu verwenden, macht also auch für diesen Fall eine Überprüfung der Simulationsergebnisse möglich. Es stellt somit eine sinnvolle Erweiterung des Verfahrens dar, um dessen Ergebnisse abzusichern.

5.6 Analyse und Visualisierung

Aufgrund der flexiblen Implementierung in Python können die Ergebnisse der Blocksimulation je nach Anwendungsfall in unterschiedliche Formate exportiert werden. Python bietet beispielsweise Exportfunktionen in das HDF5-Format oder in das von Matlab verwendete mat-Format. Die Ergebnisse können auch direkt mit Python analysiert werden, was hierfür zahlreiche Bibliotheken bietet.

Falls die Temperaturfelder analysiert werden sollen, müssen diese auf die Knoten des FE-Netzes bezogen werden. Dies kann mit MeshIO für verschiedenste Netzformate erfolgen.

Von den in Kapitel 4.6 vorgestellten Visualisierungs-Werkzeugen wurden ParaVis und Gmsh getestet. Mit beiden können auch große Geometrien interaktiv visualisiert und untersucht werden. Sie können zudem durch verschiedene Schnittebenen nur Teile der Geometrie darstellen. Beim Schneiden kann angepasst werden, ob die Tetraeder-Elemente an der Schnittkante mit geschnitten oder ganz belassen werden sollen.

ParaVis, ebenso wie das zugrundeliegende ParaView, bietet zahlreiche weitere Filterfunktionalitäten an, um die Visualisierung individuell anzupassen. Während ParaVis als eigene Oberfläche in SALOME umgesetzt ist, ist die Visualisierung in Gmsh mit den CAD- und Netz-Funktionen unter einer Oberfläche kombiniert.

Beide Werkzeuge bieten unterschiedliche grafische Exportformate an und können auch Animationen des zeitlichen Verlaufs der Temperaturverteilung erstellen. Sie können zudem auch die durch die Thermo-Elastik hervorgerufenen Verschiebungen visualisieren. Aufgrund ihrer Pythonschnittstelle können beide Werkzeuge auch automatisiert werden. Gmsh hat hierbei den Vorteil, dass es auch vollständig in Python integriert ist. Für diesen Anwendungsfall bieten sich aber auch die anderen in Kapitel 4.6 vorgestellten Werkzeuge an.

Die Visualisierungen in dieser Arbeit wurden mit Python und Gmsh erstellt.

6 Zusammenfassung und Diskussion

In dieser Diplomarbeit wurde der in Galant et al. (2014) beschriebene Arbeitsablauf zum Erstellen ordnungsreduzierter thermischer Modelle von Werkzeugmaschinen untersucht und modularisiert sowie anschließend implementiert und demonstriert. Dabei wurde nur die thermische Simulation betrachtet, da sie den Kern des Verfahrens bildet.

Der Arbeitsablauf wurde zunächst in seine Teilfunktionen und Schnittstellen zerlegt. Der nun modulare Arbeitsablauf ist flexibel gestaltet und die für seine Teilfunktionen verwendeten Werkzeuge können aufgrund der klar abgegrenzten Schnittstellen ausgetauscht werden.

Um den Arbeitsablauf robust, fehlertolerant und auf verschiedenste Anwendungskontexte übertragbar zu gestalten, sollten die Werkzeuge mit freier Software umgesetzt werden. Zudem sollten die Schnittstellen möglichst allgemeingültig gestaltet werden, um die Austauschbarkeit der Werkzeuge zu ermöglichen. Es konnte gezeigt werden, dass für alle identifizierten Teilfunktionen ein oder mehrere freie Werkzeuge verwendet werden können. Auch die Schnittstellen konnten allgemeingültig gestaltet werden. Für die CAD-Schnittstelle wurde hierfür ein freies Standardformat und für die Netz-Schnittstelle ein Konvertierer gewählt.

Ausgehend von der Modularisierung und der Analyse der Referenzimplementierung wurde eine flexible und robuste Implementierung mit freien Werkzeugen realisiert. Diese verfügt über eine innovative Geometrievorbereitung und ist vollständig mit Python automatisierbar. Der entwickelte Arbeitsablauf unterstützt den Anwender bei der Auswahl der Schnittstellen des FE-MOR-Modells und leitet daraus zugleich eine Geometrievereinfachung ab, um kleinteilige und für das thermische Verhalten irrelevante Strukturen zu entfernen. Nun erstellt er das FE-MOR-Modell voll automatisch. Dabei ist er robust und verarbeitet auch fehlerhafte Geometrien. Er benötigt somit keine eventuell aufwendige, manuelle Vorbereitung der Geometrie, die über die Schnittstellendefinition hinausgeht. Es wurde zudem ein Verfahren entwickelt, mit dem die strukturvariablen Koppelflächen automatisch aufgeteilt werden.

Für die Blocksimulation wurde gezeigt, wie diese, wie auch der restliche Arbeitsablauf, mit Python umgesetzt werden kann. Dies ermöglicht es, für die Erstellung der Teilmodelle auf alle Pythonfunktionen zugreifen zu können. So konnte zusätzlich zum FE-MOR-Modell auch das FE-Modell in die Blocksimulation integriert werden. Damit kann ein Abgleich der Dimension des reduzierten Modells mit dem FE-Modell erfolgen.

Der Arbeitsablauf zum Erstellen der FE-MOR-Modelle wurde anhand einer komplexen und fehlerhaften Dreh-Schwenk-Tisch-Baugruppe einer 5-Achs-WZM demonstriert. Er kann diese robust und automatisch vernetzen. Der erste der beiden in Kapitel 4.1 gesetzten Schwerpunkte konnte somit durch die Verwendung freier Software erfüllt werden. Der zweite Schwerpunkt konnte ebenso erfolgreich umgesetzt werden, da die FE-Matrizen im Arbeitsablauf einfach und verständlich umgesetzt werden können und zudem ihr Export in unterschiedliche Formate problemlos möglich ist.

Die Korrektheit des entwickelten Arbeitsablaufs und der Blocksimulation wurden anhand eines einfachen Beispiels im Vergleich mit der Referenzimplementierung geprüft.

Insgesamt ermöglicht es der entwickelte Arbeitsablauf, schnell und robust FE-MOR-Modelle von komplexen Strukturbauteilen abzuleiten und diese anschließend zu simulieren. Indem der Anwender durch die Automatisierung bei der Modellerstellung unterstützt wird, kann er sich voll auf die Gestaltung der Simulation und die Ergebnisanalyse konzentrieren. Die Unterstützung bei der Schnittstellendefinition, die automatische Geometrievereinfachung und Flächenaufteilung ermöglichen es, die FE-MOR-Modelle schnell zu variieren und verschiedene Varianten im thermischen Entwurf der WZM zu berücksichtigen. Der Vergleich des FE-MOR-Modells mit dem FE-Modell direkt in der Blocksimulation ermöglicht darüber hinaus, die Genauigkeit der Ergebnisse einzuschätzen.

Da der Arbeitsablauf nun vollständig mit freien Werkzeugen und Schnittstellen umgesetzt ist, kann er problemlos in verschiedenen Anwendungsbereichen verwendet und weiterentwickelt werden. Auch ein Industrietransfer des Arbeitsablaufs ist so direkt möglich. Hierfür sind vor allem die allgemeingültigen Schnittstellen relevant, da sie es ermöglichen den Arbeitsablauf flexibel in eine bestehende Werkzeugkette einzubinden.

Ausblick

Die Blocksimulation wurde noch nicht mit komplexen WZM-Baugruppen getestet. Dies sollte jedoch problemlos möglich sein, da für die Implementierung robuste und bewährte Werkzeuge verwendet wurden. Durch die neuen Möglichkeiten können anschließend an einen solchen Test direkt derzeit offene Forschungsfragen zum Verfahren getestet werden. Es sollte beispielsweise untersucht werden, welchen Einfluss die Schnittstellendefinition und die Dimension des reduzierten Modells auf die Simulationsergebnisse haben. Als Fragestellung kann hierbei untersucht werden, ob sich allgemeinere Richtwerte ableiten lassen. Mit diesen kann der Arbeitsablauf weitergehend automatisiert werden.

Statt allgemeiner Richtlinien kann die Dimension des reduzierten Modells auch durch einen automatisierten Abgleich des FE-MOR-Modells mit dem FE-Modell erfolgen. Da die Simulation mit den vollen FE-Matrizen für die benötigte Simulationsdauer weit länger als mit den reduzierten Modellen dauert, kann der Abgleich auch nur über einen kürzeren Zeitraum erfolgen.

Insgesamt bietet der entwickelte Arbeitsablauf aufgrund seiner flexiblen Module und allgemeingültigen Schnittstellen eine solide Grundlage, um darauf weitere Simulationsabläufe aufzubauen. Es können sowohl Teile, beispielsweise die innovative Geometrievorbereitung oder Netzerstellung, in andere Abläufe integriert werden als auch ein umfassenderes Simulationswerkzeug daraus weiterentwickelt werden. Ein erster Schritt ist sicherlich, die in Galant et al. (2014) ebenfalls vorgestellte Erweiterung des Verfahrens zur thermo-elastischen Simulation in den entwickelten Arbeitsablauf zu integrieren. Dies wird von den verwendeten Werkzeugen unterstützt. Der Vernetzungsablauf erlaubt zudem eine schnelle und automatische Ableitung der dafür benötigten strukturvariablen FE-Netze. Weiterhin kann darauf aufbauend untersucht werden, ob die für die Strukturmechanik genutzten FE-Modelle ebenfalls reduziert werden können.

7 Digitaler Anhang

- Übersicht.txt
- Quellcode
  - geo: Funktionen für Geometrie und Netz
  - map-merge: MapMerge
  - mor: Funktionen für FEM, MOR und Blocksimulation
  - salome: Dockerfile für SALOME
- Experimente
  - geo-tbl: Vorbereitung der Geometrie und Vernetzen der Dreh-Schwenk-Tisch-Baugruppe
  - geo-test: Erster Test der Funktionen zum Vorbereiten der Geometrie und zum Vernetzen
  - mor-min: FEM, MOR und Blocksimulation des Würfels
  - mor-tbl: FEM, MOR der Dreh-Schwenk-Tisch-Baugruppe
  - referenz: Referenzimplementierung des Würfels
  - code-aster: Versuch mit Code_Aster
- Diplomarbeit
  - diplom.tex
  - diplom.pdf
  - nomencl.ist
  - figs: Bilder