TECHNISCHE UNIVERSITÄT DRESDEN

Michael Bauer

Modularisierung des Arbeitsablaufs zur Erstellung ordnungsreduzierter thermischer Modelle von Werkzeugmaschinen

Diplomverteidigung // Dresden, 26.06.2020

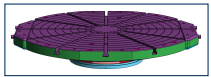
Inhalt

Stand der Technik

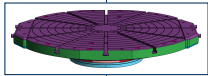
- Bearbeitungsungenauigkeit durch thermische Verformung bei Werkzeugmaschinen
- Kompensation oder Korrektur mittels Simulation
- Am LWM wurde Verfahren zur schnellen und realistischen thermischen Simulation von Werkzeugmaschinen entwickelt
- Strukturbaugruppen werden mit FEM und anschließender MOR modelliert
  $\Rightarrow$ FE-MOR-Modelle

Simulationsverfahren

FE-MOR-Modell



Blocksimulation



Analyse

- Verbindungen
- Temperaturfelder

Weitere **Teilmodelle**

- Wärmeströme $\dot{Q}$
- Wärmeübergänge α
- Geführte Temperaturen und Bewegungen T
- Schaltbare Verbindungen

Zielstellung

- Modularisierung: austauschbare Werkzeuge und allgemeingültige Schnittstellen
- Arbeitsablauf flexibel, robust und automatisiert
- Erweiterter Anwendungsbereich durch Verwendung freier Software

Modularisierung

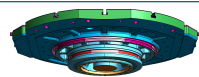
Schnittstellen *Module*

▷ CAD-Modell

→ • Geometrie vorbereiten

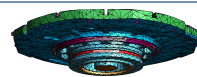
• Koppelflächen definieren

• Vorbereitetes CAD-Modell
▷ Koppelflächen



→ • Vernetzen

• Tetraeder-Netz



→ • FE-Gleichungssystem

FEM

• FE-Modell
▷ Parameter und Art der Eingänge

→ • Zustandsraumdarstellung
• Modellordnungsreduktion

MOR

• FE-MOR-Modell

Referenzimplementierung

FEM

Ansys

MOR

MATLAB

Referenzimplementierung

FEM

Ansys

MOR

MATLAB

Defizite

- Modularisierung
- Geometrieentfeinerung
- Koppelflächenauswahl
- Vernetzen
- Export des FE-Gleichungssystems

Freie Werkzeuge

FEM

MOR

Ansys

MATLAB

SALOME

FreeCAD

Gmsh

Netgen

TetGen

fTetWild

Code_aster

FeniCS

NumPy

SciPy

Python

Schnittstellen

CAD

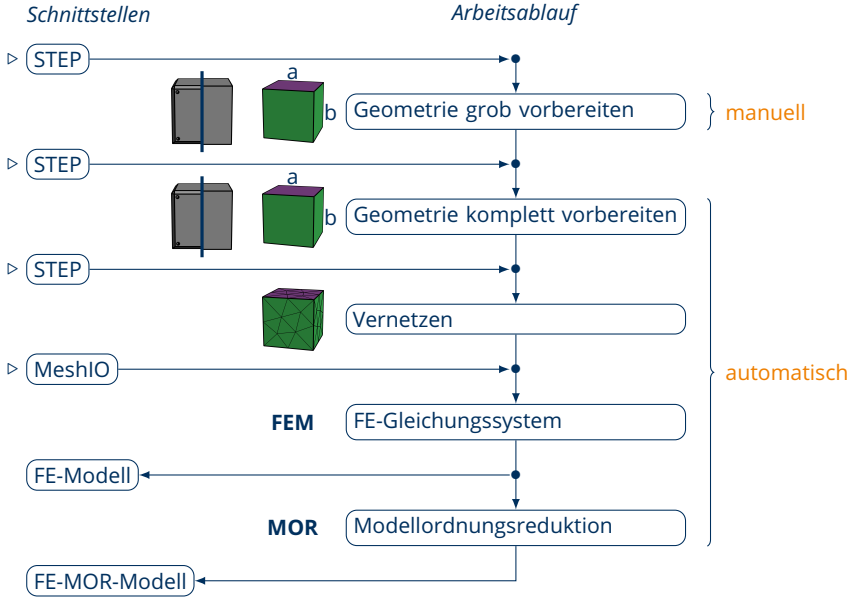
- Zahlreiche, meist proprietäre Formate
- Normiertes, weit verbreitetes STEP-Format
- Fehler durch Konvertieren

Netz

- Zahlreiche, ähnlich aufgebaute Formate
- Kein Standard
- Konvertierer: MeshIO

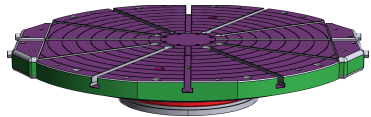
⇒ **Allgemeingültige Schnittstellen**

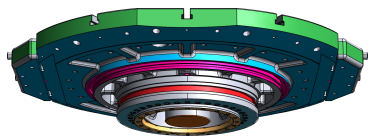
Implementierung des Arbeitsablaufs



Geometrie vorbereiten

manuell





- Abgrenzung der Koppelflächen
  ⇒ 251 Flächen (5 %)
  45 min ⇒ 10 min
- Automatisierte Behandlung der restlichen Flächen
- Zugleich Geometrieentfeinerung
- Automatische Auf teilung der strukturvariablen Koppelflächen

4487 restliche Flächen ⇒ 250 zusammenhängende Flächen ⇓ 28 Gruppen

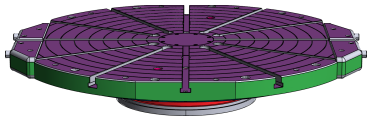
FEM

MOR

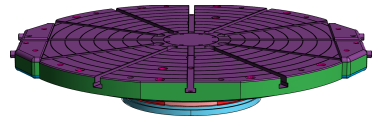
Anzahl	Benachbarte Schnittstellen	Verb.	Geschl.	Aktion
1	Oberseite	4	Nein	Oberseite
1	Lager, Dichtung, Bohrung	2, 4, 8	Nein	Innen_Lager
20	Antrieb_Außen	4	Ja	Füllen
…	und 25 weitere			

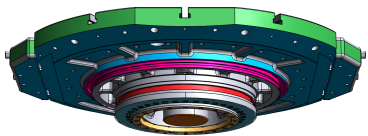
Geometrie vorbereiten

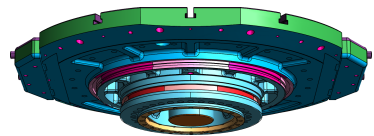
manuell



automatisch







4487 restliche Flächen ⇒ 250 zusammenhängende Flächen ⇓ 28 Gruppen

FEM

MOR

Anzahl	Benachbarte Schnittstellen	Verb.	Geschl.	Aktion
1	Oberseite	4	Nein	Oberseite
1	Lager, Dichtung, Bohrung	2, 4, 8	Nein	Innen_Lager
20	Antrieb_Außen	4	Ja	Füllen
…	und 25 weitere			

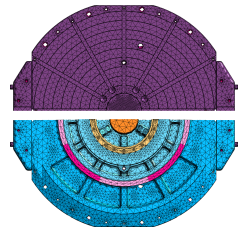
FE-Netz

Dreiecks-Netz

- Schnell und robust aus STEP

Tetraeder-Netz

- Innovatives Vorgehen mit fTetWild
- Äußerst robust und qualitativ hochwertiges Netz
- Intuitive Parameter:
  Toleranz, Elementgröße und -qualität
- Koppelflächen übertragen mit MapMerge



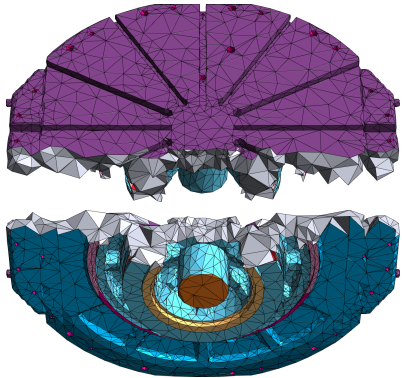
FEM

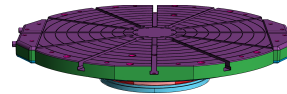
MOR

Vernetzervergleich anhand von 10 000 Geometrien
(Hu et al., 2019)

Vernetzer	Korrekt vernetzt	Zeit
fTetWild	100 %	50 s
TetGen	50 %	32 s
4 weitere	37 % bis 87 %	12 s bis 2182 s

FE-Netz





- 20 744 Tetraeder
- 1 min
- Minimaler Volumenfehler (0,4 %)
- Flächeninhaltsfehler

FEM

MOR

Koppelfläche	Flächeninhaltsfehler	111 243 Tetraeder
Oberseite	5,0 %	4,3 %
Unterseite	1,6 %	1,0 %
Innen_Lager	3,7 %	1,9 %
Dichtung	4,1 %	3,1 %
Dichtung aufgeteilt	0,8 % bis 6,7 %	2,4 % bis 4 %

FE-MOR-Modell

- Direkt in Python flexibel umgesetzt

FEM

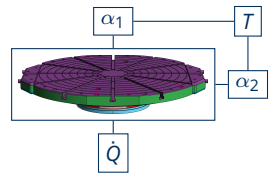
- FeniCS
- Mathematische Notation
- Matrizen und Knotenindizes in Python

Blocksimulation

- Python-Simulationsframework
- Alle Pythonfunktionalitäten:
  Empirische oder statistische Teilmodelle
- Auch FE-Modell
  $\Rightarrow$ Abgleich MOR und Koppelflächen
- Korrektheit der Implementierung anhand
  Referenzimplementierung geprüft

MOR

- Numpy, SciPy



Zusammenfassung

- Modularisierter Arbeitsablauf mit allgemeingültigen Schnittstellen kann flexibel eingesetzt und weiterentwickelt werden
- Ein manueller Arbeitsschritt, in dem die Koppelflächen definiert werden und zugleich die Geometrie entfeinert wird
- Automatisches Aufteilen der strukturvariablen Koppelflächen
- Robuste und intuitive Netzerstellung, die zudem zuverlässig die Geometrie entfeinert
- Flexibel integrierbare FEM
- Komplett mit freien Werkzeugen umgesetzt

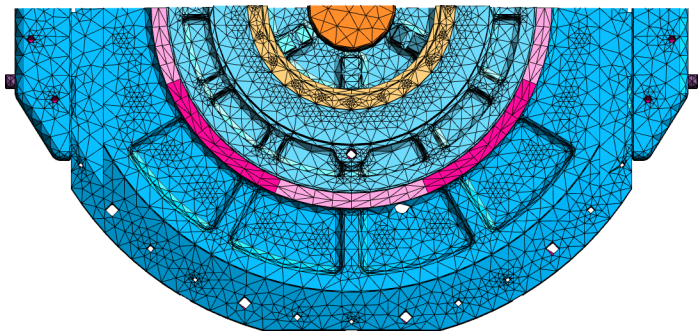
$\Rightarrow$ Ziele wurden erreicht

$\Rightarrow$ Ermöglicht schnelles Iterieren der thermischen Modelle

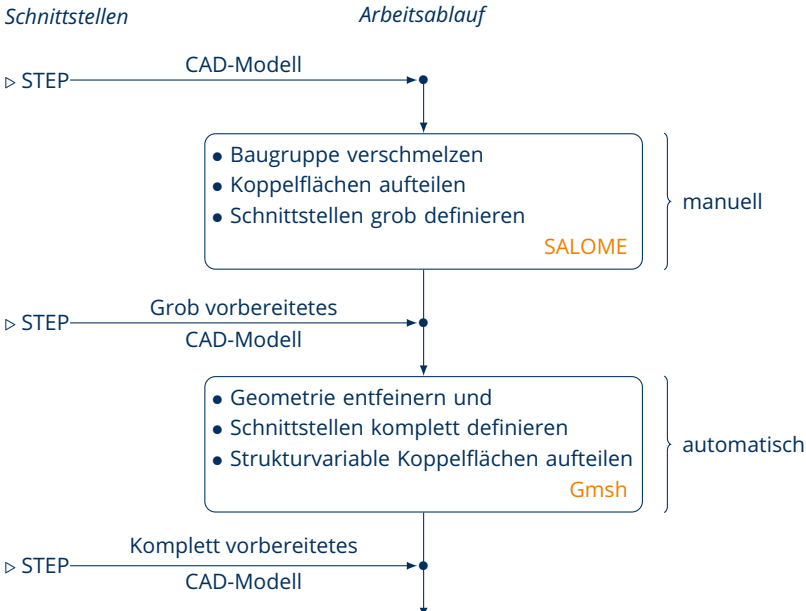
Ausblick

- Integration in bestehende Werkzeugketten
- Geometrievorbereitung und Netzerstellung breiter anwendbar
- Erweiterung zur thermo-elastischen Simulation
- Abgleich der reduzierten Dimension mit dem FE-Modell

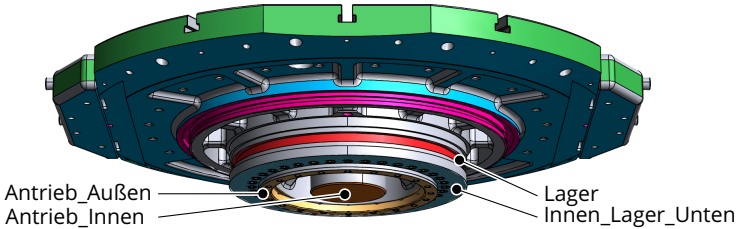
Vielen Dank für Ihre Aufmerksamkeit!



Geometrie vorbereiten

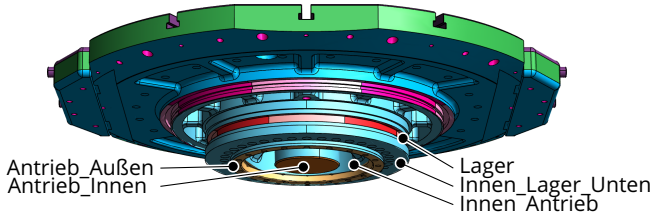


Restliche Flächengruppen



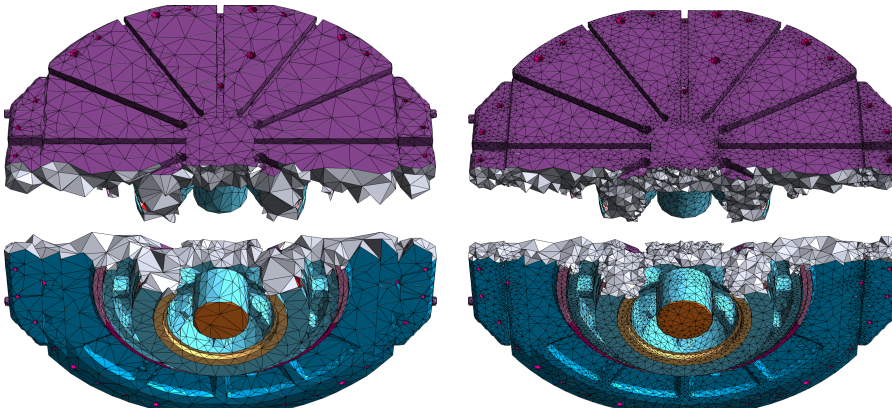
	Anzahl	Benachbarte Schnittstellen	Verb.	Geschl.	Aktion
1	102		0	Ja	Löschen
2	20	Antrieb_Außen	4	Ja	Füllen
3	1	Antrieb_Außen, Antrieb_Innen	4, 2	Ja	Innen_Antrieb
4	1	Antrieb_Außen, Innen_Lager_Unten	4, 4	Ja	Innen_Lager_Unten
5	36	Innen_Lager_Unten	2	Ja	Füllen
	…	und 23 weitere			

Flächenfehler



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

FE-Netze



FE-Gleichungssystem

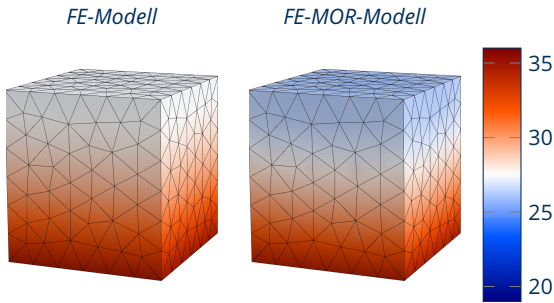
$$\boldsymbol{C}_T = \int_\Omega \boldsymbol{p}\boldsymbol{v}\,\mathrm{d}\Omega \tag{1}$$

$$\boldsymbol{K}_{T\lambda} = \int_\Omega \nabla\boldsymbol{p}\nabla\boldsymbol{v}\,\mathrm{d}\Omega \tag{2}$$

$$\boldsymbol{f}_i = \int_{\Gamma i} \boldsymbol{v}\,\mathrm{d}\Gamma_i \tag{3}$$

```
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)
```

Blocksimulation: Würfel



Blocksimulation: Verlauf