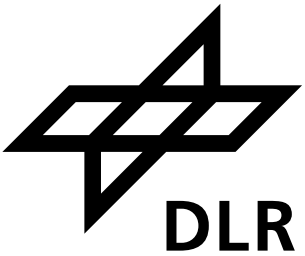
SEMESTERARBEIT

Prozess-Implementierung einer automatisierten globalen Turbomaschinen-Analyse

von

Thomas Kaller

Beginn: 5.1.2011
Abgabe: 16.6.2011



Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Antriebstechnik, Köln
Institutsleiter: Prof. Dr.-Ing. Reinhard Mönig
Abteilung für Numerische Methoden
Abteilungsleiter: Dr.-Ing. Edmund Kügeler
Betreuer: Dipl.-Ing. Jens Wellner

Technische Universität München
Institut für Luft- und Raumfahrt
Lehrstuhl für Flugantriebe
Lehrstuhlleiter: Prof. Dr.-Ing. Hans-Peter Kau
Betreuer: Dipl.-Ing. Florian Danner

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Nomenklatur

Trotz der Breite des behandelten Themas wird versucht, möglichst eindeutige Variablennamen zu vergeben. Wenn die Möglichkeit einer Verwechslung jedoch ausgeschlossen ist, können zwei verschiedene Größen auch identisch bezeichnet werden (zum Beispiel ω)

Koordinaten

η	dimensionslose Normalkoordinate
ξ	dimensionslose Meridionalkoordinate
$xr\theta$	zylindrische Koordinaten
xyz	kartesische Koordinaten

Indizes

abs	absolut
IJK	Indizes der Netzknoten
ijk	Indizes der Einsteinschen Summenkonvention
is	isentrop
kin	kinetisch
m	meridional
max	maximal
min	minimal
n	normal
$poly$	polytrop
pot	potentiell
rel	relativ
t	Totalgröße
z	zentrifugal

Symbole

Δ	allgemeine Differenz
δ	Kronecker-Delta
C	allgemeine Konstante

Physikalische Größen

$\dot{m}$	Massenstrom
$\dot{Q}$	Wärmestrom
η_{is}	isentroper Wirkungsgrad
η_{poly}	polytroper Wirkungsgrad

Γ	Zirkulation
γ	Isentropenkoeffizient
λ	Wärmeleitfähigkeit
μ	dynamische Viskosität
μ_t	turbulente Viskosität
ν	kinematische Viskosität
ν_{poly}	Polytropenverhältnis
ω	Winkelgeschwindigkeit
ω	spezifische Dissipationsrate
Φ	Dissipationsfunktion
ρ	Dichte
$\tau_{t,ij}$	Reynolds'scher Schubspannungstensor
$\underline{\phi}$	Lösungsvektor
$\underline{\underline{A}}$	Koeffizientenmatrix
$\underline{Q}$	Quellterm
$\underline{R}$	Residuum
$\vec{v}$	Geschwindigkeitsvektor
$\vec{x}$	Ortsvektor
c_p	Druckbeiwert
c_p	isobare Wärmekapazität
c_v	isochore Wärmekapazität
e	spezifische Energie
F	Kraft
F_{thrust}	Triebwerksschub
g	Erdbeschleunigung
h	Enthalpie
h_t	Totalenthalpie
h_{rot}	Rothalpie
I	Impuls
j	Dissipation (spezifisch)
k	spezifische turbulente kinetische Energie
L	Drall
l	Länge
M	Moment
Ma	Mach-Zahl
N	Anzahl
n_{poly}	Polytropenexponent
p	Druck
p_t	Totaldruck

R	spezifische Gaskonstante
Re	Reynolds-Zahl
s	Entropie
s	Sehnenlänge
S_{ij}	Scherratentensor
T	Temperatur
T	Zeitmaß
t	Schaufelteilung
t	Zeit
T_t	Totaltemperatur
T_{ij}	Spannungstensor
u	Umfangsgeschwindigkeit
u	innere Energie
v	Geschwindigkeit (Betrag)
v_{spec}	spezifisches Volumen
P	Leistung
q	Wärmemenge (spezifisch)
w_{eff}	Nutzarbeit (spezifisch)
$w_{fluid,t}$	totale Druckänderungsarbeit (spezifisch)
w_{fluid}	Druckänderungsarbeit (spezifisch)
w_{tech}	technische Arbeit (spezifisch)

Kapitel 1

Einleitung

Stetig wachsende Anforderungen aus den Bereichen der Ökonomie, der Ökologie sowie seitens der gesetzlichen Rahmenbedingungen erfordern eine zunehmende Optimierung von Turbomaschinen. Insbesondere bei Flugantrieben ist aufgrund des beständig steigenden Kerosinpreises einerseits und des erwarteten international zunehmenden Flugverkehrsaufkommens, besonders in den wirtschaftlich aufsteigenden Ländern des asiatischen Raums, andererseits eine weitere Steigerung der Effizienz nötig. Gleichzeitig ist auch die Thematik der Nachhaltigkeit zu berücksichtigen. Ergo gilt es die Schadstoffemissionen in Zeiten des Klimawandels sowie der andauernden Verschmutzung und Zerstörung der Umwelt auf ein Minimum zu reduzieren.
Im Entwurfsprozess von Strömungsmaschinen in den beiden Hauptanwendungsbereichen, derjenige der Flugantriebe und der der stationären Gasturbinen zur Stromgewinnung, spielt heutzutage die CFD eine entscheidende Rolle. Denn aufgrund der über die letzten Jahre enorm gewachsenen Rechenleistung von Computern können thermofluiddynamische Problemstellungen in einer annehmbaren Zeit und mit einer relativ hohen Genauigkeit simuliert werden. Die erhaltenen Simulationsresultate sind schließlich so aufzubereiten, dass eine optimale Analyse der untersuchten Konfiguration möglich ist. Optimal bedeutet dabei in erster Linie zeitoptimal, jedoch darf keine essentielle Information verloren gehen bzw. verfälscht werden. Dieser Aufgabenbereich stellt den Kern des Postprocessings dar.
Im Rahmen der vorliegenden Arbeit wird die Implementierung einer globalen Turbomaschinen-Analyse in das am Deutschen Zentrum für Luft- und Raumfahrt (DLR) von der Abteilung Antriebstechnik-Numerische Methoden entwickelte Postprocessing-Tool *POST* beschrieben. Dieses Programm dient zur Auswertung der vom Strömungslöser *TRACE* gelieferten resultierenden Strömungsfelder. *TRACE* steht dabei für *Turbomachinery Research Aerodynamic Computational Environment*. Für weiterführende Informationen wird auf [NESZ01], [BHK10] und [AHK10] verwiesen.
Das Ziel ist die Implementierung einer Turbomaschinen-Analyse, welche unabhängig vom Typ der untersuchten Turbomaschine ist und ausschließlich auf der von *TRACE* gelieferten dreidimensionalen CGNS-Datei basiert. Zusätzlich zur eigentlichen Turbomaschinen-Auswertung werden weitere Funktionalitäten benötigt, auf welchen Erstere basiert. Deren Implementierung wird ebenfalls vorgestellt und abschließend anhand ver-

schiedener Testfälle validiert. Die Aufgabenstellung setzt sich dabei im Einzelnen wie folgt zusammen:

- Erstellung von Schnittflächen normal zur Meridianrichtung basierend auf einer temporären S2M-Fläche. Diese stellen die Grundlage der zu generierenden S2M-Fläche dar.
  - Ermittlung des Schattenrisses bzw. der Minimal- und Maximalkontur einer gegebenen 3D-Konfiguration
  - Definition eines temporären S2M-Netzes zur Verschneidung mit der gegebenen 3D-Konfiguration basierend auf der ermittelten Kontur
  - Einlesen eines vorhandenen S2M-Netzes zur Verschneidung mit der gegebenen 3D-Konfiguration
- Implementierung von Fluss-, Massen- und Flächenmittelung (Mittelung in Umfangs- und Normalrichtung)
- Aufbau der resultierenden S2M-Fläche der Konfiguration mit umfangsgemittelten Strömungsgrößen
- Erstellung von Schaufelschnitten und S1-Stromflächen für eine konstante relative Kanalhöhe oder einen konstanten relativen Massenstrom
- Berechnung typischer 0D-Kenngrößen sowie radialer Verteilungen zur Analyse einer Turbomaschinen-Konfiguration
- Validierung der Implementierung anhand verschiedener Testfälle

Kapitel 2

Aero-thermodynamische Grundlagen der Turbomaschinen

2.1 Turbomaschinen - Arbeitsprinzip und Definition

Der erste Hauptsatz der Thermodynamik besagt, dass Energie weder erzeugt noch vernichtet werden kann, sie kann nur von einer Form in eine andere transformiert werden. Nach [Brä09] handelt es sich bei Turbomaschinen um Energiewandler. In diesen findet hauptsächlich eine Umwandlung von mechanischer Arbeit in Strömungsenergie und umgekehrt statt. Gleichzeitig tritt als weitere Energieform die Dissipation auf, welche denjenigen Anteil der nutzbaren Energie darstellt, der aufgrund von Reibung in innere Energie des Fluids übergeht und somit für den weiteren Arbeitsprozess verloren ist. Turbomaschinen basieren also grundsätzlich darauf, dass sich durch den Austausch mechanischer Arbeit zwischen dem offen durchströmten System und der Umgebung das Niveau der in der Strömung enthaltenen Energie über eine Komponente ändert. Die Strömungsenergie setzt sich dabei aus der Enthalpie und der kinetischen Energie des Fluids zusammen und beschreibt somit die Arbeitsfähigkeit des Mediums.
Der Name „Turbomaschine“ leitet sich vom lateinischen *turbare* (sich drehen) ab und weist darauf hin, dass die Energiewandlung über Laufräder bzw. Rotoren, welche sich auf einer drehenden Welle befinden, erfolgt. Ein Synonym stellt der Begriff „Strömungsmaschine“ dar. Dieser dient zur Abgrenzung gegenüber Kolbenmaschinen, wie zum Beispiel Dieselmotoren. Denn im Gegensatz zu Letzteren findet bei Strömungsmaschinen die Energieübertragung nicht in einem intermittierten Prozess, sondern in Form eines kontinuierlichen Prozesses statt. Dieser Unterschied zwischen Kolben- und Strömungsmaschinen wird von *Abb. 2.1* verdeutlicht. Die Abbildung zeigt, dass grundsätzlich die gleichen Arbeitsvorgänge, also Ansaugung, Verdichtung, Verbrennung, Arbeitstakt sowie Ausstoßung und zwar in identischer Reihenfolge ablaufen. Der Unterschied besteht darin, dass es sich bei Turbomaschinen um offene, durchströmte Systeme handelt und bei Kolbenmaschinen um abgeschlossene Systeme. Diese Tatsache hat zur Folge, dass die einzelnen Schritte des Arbeitsprozesses bei Strömungsmaschinen auf mehrere, örtlich voneinander getrennte Komponenten verteilt sind, bei Kolbenmaschinen jedoch in einem Bauelement ablaufen.

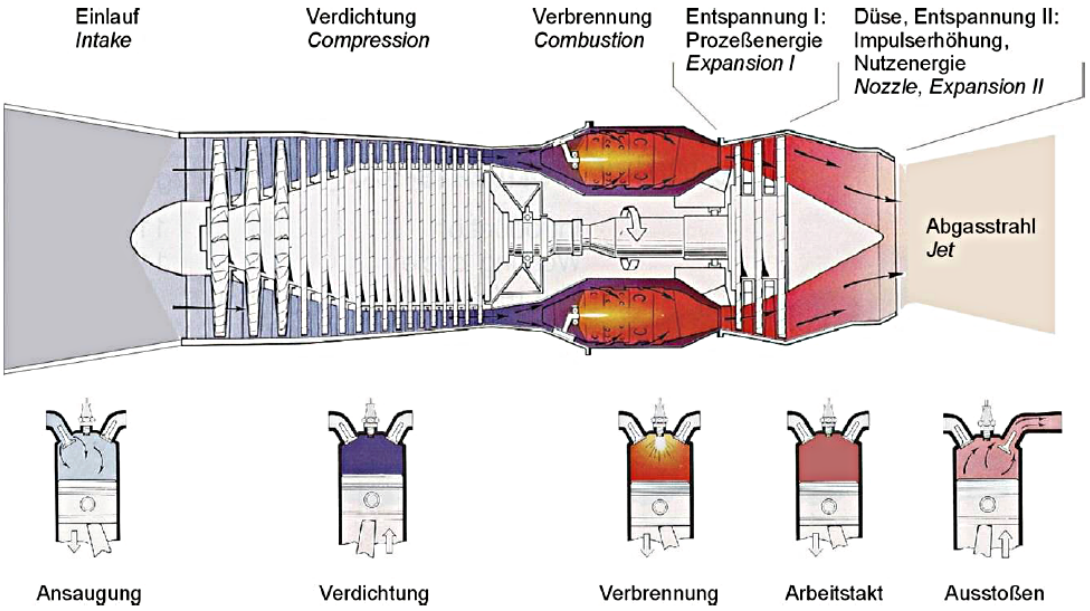


Abb. 2.1: Vergleich der Arbeitsprozesse von Turbomaschinen und Kolbenmaschinen [K$^+$09a]

Die kontinuierliche Durchströmung und damit auch die kontinuierliche Energieumsetzung hat nach [K$^+$09b] zur Folge, dass bezogen auf die Baugröße bei Turbomaschinen eine deutlich höhere Anlagenleistung erzielt werden kann als bei Kolbenmotoren. Gleichzeitig vermindert die offene Bauweise der Brennkammer die strukturelle Belastung durch die Verbrennung des Kraftstoffs, was Gewichtseinsparungen ermöglicht. Beide Vorteile sind verantwortlich für die heute prädestinierte Stellung der Strömungsmaschinen im Bereich der Flugzeugantriebe.
Turbomaschinen lassen sich nach diversen Kriterien untergliedern. Eine erste Kategorisierungsmöglichkeit stellt die Kompressibilität des Arbeitsmediums dar. Bei Verwendung von inkompressiblen Medien wie zum Beispiel Wasser spricht man von hydraulischen und bei Verwendung von kompressiblen Medien wie Luft von thermischen Turbomaschinen.
Die hauptsächliche Einteilung der Strömungsmaschinen erfolgt hinsichtlich der Richtung der Energiewandlung. Auf der einen Seite stehen die Turbokraftmaschinen bzw. antreibenden Turbomaschinen, wie zum Beispiel Turbinen, und auf der anderen die Turboarbeitsmaschinen bzw. angetriebenen Turbomaschinen, wie zum Beispiel Verdichter. Bei Ersteren wird dem Arbeitsfluid über die Rotoren Energie entzogen, wodurch das Niveau der Strömungsenergie vom Eintritt zum Austritt abgesenkt wird und die gewonnene Nutzarbeit über die Welle weiteren Prozessen zugeführt werden kann. Bei Turboarbeitsmaschinen hingegen wird durch die Zufuhr von mechanischer Arbeit das Energieniveau der Strömung angehoben. Wie *Abb. 2.1* verdeutlicht, stellen Flugantriebe eine Kombination von Turboarbeits- und Turbokraftmaschine dar. Die Turbine entzieht dem Arbeitsmedium Luft Strömungsenergie, welche dazu genutzt wird den Verdichter anzutreiben. Die Leistungsübertragung erfolgt dabei über die gemeinsame Welle.
Eine weitere Einteilung erfolgt bezüglich der Durchströmungsrichtung der Maschine. Axialmaschinen sind dadurch gekennzeichnet, dass sich der mittlere Radius der Strö-

mung nicht signifikant ändert. Bei Radialmaschinen dagegen erfolgt bei Verdichtern die Anströmung axial, die Abströmung jedoch in radialer Richtung beziehungsweise für Turbinen umgekehrt. Zwischen diesen beiden Extrema befinden sich die Diagonalmaschinen. Diese sind im Fall eines Verdichters ebenfalls durch eine axiale Anströmung gekennzeichnet; die Abströmung erfolgt dann unter einem Winkel von kleiner 90°.

2.2 Grundlagen der Thermodynamik

Dieser Abschnitt befasst sich mit den thermodynamischen Grundlagen einer Turbomaschinen-Strömung. Dabei beschränkt sich die Betrachtung auf das Arbeitsmedium Luft, welche zudem als ideales Gas behandelt wird.
Zunächst wird auf die Beschreibung des thermodynamischen Zustands einer Strömung eingegangen. In diesem Zusammenhang wird auch der Unterschied zwischen statischen und totalen Größen betrachtet. Anschließend erfolgt die Herleitung des ersten Hauptsatzes der Thermodynamik für ein offenes, stationär durchströmtes System. Das abschließende Unterkapitel beschäftigt sich mit der Beschreibung von Zustandsänderungen sowie der Definition von isentropem und polytropem Wirkungsgrad.

2.2.1 Beschreibung des thermodynamischen Zustands

Der thermodynamische Zustand eines Systems wird über dessen Zustandsgrößen beschrieben. Eine Veränderung des Systemzustands bedingt somit eine Änderung der Zustandsgrößen. Zur Beschreibung der Wechselwirkung zwischen diesen Größen dienen Zustandsgleichungen, welche einen funktionalen Zusammenhang zwischen den einzelnen Zustandsgrößen eines Systems darstellen.
Für den Bereich der Turbomaschinen sind insbesondere die thermische und kalorische Zustandsgleichung von Bedeutung. Erstere stellt eine Verknüpfung zwischen der Temperatur T, dem Druck p und der Dichte ρ bzw. dem spezifischen Volumen v_{spec} her. Nach [K$^+$09b] lautet die thermische Zustandsgleichung in der differentiellen Form:

$$\frac{dT}{T} = \frac{1}{\alpha_v} \cdot \frac{dp}{p} + \frac{1}{\alpha_p} \cdot \frac{dv_{spec}}{v_{spec}} \tag{2.1}$$

Bei α_p und α_v handelt es sich um den isobaren bzw. isochoren Ausdehnungskoeffizienten, welche für ein ideales Gas zu $\alpha_p = \alpha_v = 1$ bestimmt sind. Damit ergibt sich nach Integration von *Gl. (2.1)* die thermische Zustandsgleichung idealer Gase:

$$p\, v_{spec} = R\, T \quad \text{bzw.} \quad p = \rho\, R\, T \tag{2.2}$$

Die zweite essentielle Korrelation, die kalorische Zustandsgleichung bzw. Energiegleichung, stellt einen funktionalen Zusammenhang zwischen der inneren Energie u bzw. der Enthalpie h und den thermischen Zustandsgrößen T, p und ρ dar. In der allgemeinen Form lautet diese:

$$\begin{aligned} dh &= c_p(T) \cdot dT + (1 - \alpha_p)\, v_{spec} \cdot dp \\ du &= c_v(T) \cdot dT - (1 - \alpha_v)\, p \cdot dv_{spec} \end{aligned} \tag{2.3}$$

Für ideale Gase gilt folglich:

$$\begin{aligned} dh &= c_p(T) \cdot dT \\ du &= c_v(T) \cdot dT \end{aligned} \tag{2.4}$$

Für die zumeist gerechtfertigte Annahme eines idealen Gases ergibt sich folglich, dass sich die Energie- bzw. Enthalpieänderung einer Strömung oder allgemein des betrachteten thermodynamischen Systems direkt proportional zur Änderung der Temperatur verhält.
Die beiden Zustandsgleichungen wurden zunächst für statische thermodynamische Zustandsgrößen betrachtet. Insbesondere zur Analyse des Arbeitsumsatzes oder der Strömungsverluste werden Totalgrößen wie der Totaldruck p_t sowie die Totalenthalpie h_t verwendet. Die totalen Größen ergeben sich dabei aus den statischen durch eine (gedachte) isentrope, also verlustfreie Verzögerung auf die Geschwindigkeit $v = 0$.
Nach [Brä09] lässt sich die Leistungsfähigkeit einer Triebwerkskomponente über das Verhältnis der Totaldrücke am Ein- und Auslass beurteilen. Der Totaldruck stellt dabei die Summe aller Druckenergien in einer Strömung dar und setzt sich aus dem statischen Druck p sowie einem kinetischen Anteil zusammen. Durch Leistungszufuhr ergibt sich ein Anstieg des Totaldruck-Niveaus sowie ein Abfall bei Leistungsabgabe. Findet kein Arbeitsaustausch des betrachteten Systems mit der Umgebung statt, sinkt der Totaldruck dennoch aufgrund von Reibungseffekten. Somit stellt der Totaldruck ein Maß zur Bestimmung der Strömungsverluste dar. Für kompressible Medien lässt er sich nach [K$^+$09a] wie folgt ermitteln:

$$p_t = p \cdot \left(1 + \frac{\gamma - 1}{2}\, Ma^2\right)^{\frac{\gamma}{\gamma - 1}} \tag{2.5}$$

Für die Totalenthalpie, welche den Energieinhalt des Arbeitsfluids beschreibt, gilt:

$$h_t = h + \frac{v^2}{2} \tag{2.6}$$

Schließlich ergibt sich für die Totaltemperatur, welche sich für ein ideales Gas direkt proportional zur Enthalpie verhält:

$$T_t = T + \frac{v^2}{2 \cdot c_p(T)} = T\left(1 + \frac{\gamma - 1}{2} Ma^2\right) \tag{2.7}$$

2.2.2 Thermodynamische Zustandsänderungen

Wie bereits im vorigen Abschnitt erwähnt, bedeutet eine Änderung des thermodynamischen Zustands eines Systems eine Änderung der beschreibenden thermodynamischen Zustandsgrößen. Die Grundlage der Zustandsänderungen bilden neben der Kontinuitätsgleichung sowie den Impulserhaltungsgleichungen der erste und zweite Hauptsatz der Thermodynamik.
Nach [K$^+$09a] lautet die allgemeine integrale Formulierung des ersten Hauptsatzes:

$$\sum \dot{Q} + \sum P_{tech} + \sum \dot{m}\left(u + \frac{p}{\rho} + \frac{v^2}{2} + gz\right) = \frac{d}{dt} \sum \dot{m}\left(\bar{u} + \bar{e}_{kin} + \bar{e}_{pot}\right) \tag{2.8}$$

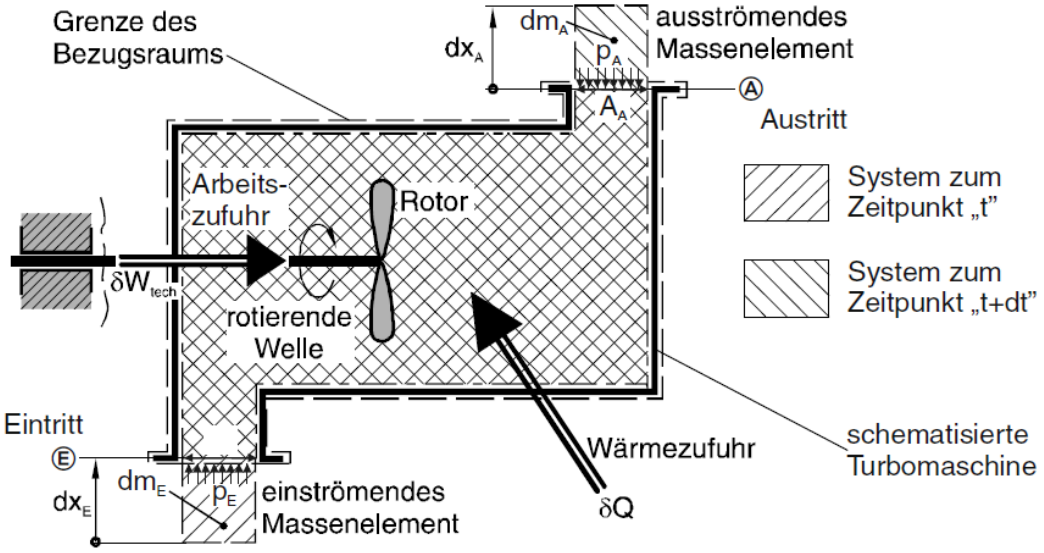


Abb. 2.2: Schematisierte Turbomaschinen-Darstellung [Brä09]

Für ein offenes stationär durchströmtes System inklusive Wärme- und Arbeitsaustausch mit der Umgebung (siehe *Abb. 2.2*) ergeben sich folgende Vereinfachungen:

- stationäres System $\Rightarrow \frac{d}{dt} = 0$
- Massenerhaltung $\Rightarrow \dot{m} =$ konstant

Zusammen mit der Totalenthalpie der Strömung

$$h_t = \left(u + \frac{p}{\rho}\right) + \frac{v^2}{2} + gz \tag{2.9}$$

ergibt sich nach Normierung durch den konstanten Massenstrom $\dot{m}$ folgender Zusammenhang:

$$q + w_{tech} = \Delta h_t \tag{2.10}$$

Bei Betrachtung des Arbeitsmediums Luft kann zur Berechnung der Totalenthalpie h_t, welche, wie bereits erwähnt, den gesamten Energieinhalt eines Fluids beschreibt, der geodätische Anteil gz meist vernachlässigt werden. Bezüglich der Vorzeichenkonvention ist anzumerken, dass nach *Bräunling* ein übertragener Wärmestrom bzw. die übertragene Leistung positiv ist, wenn sich aufgrund dieser Übertragung die Totalenthalpie des Fluids erhöht.
Die aus dem ersten Hauptsatz hergeleitete *Gl. (2.10)* ist für alle Komponenten einer Turbomaschine gültig, wobei Gitter- und Kanalströmungen meist adiabat behandelt werden. Es erfolgt also keinerlei Wärmeaustausch mit der Umgebung ($q = 0$). Existieren in der zu untersuchenden Konfiguration jedoch Nebenströme, wie zum Beispiel infolge von Luftentnahme im Verdichter oder Kühlung in der Turbine, so kann zur Bilanzierung über den Kontrollraum nicht die spezifische Form des ersten Hauptsatzes herangezogen werden. Stattdessen wird unter Annahme einer adiabaten Komponente eine Leistungsbilanz aufgestellt:

$$P_{tech} = \sum \dot{m} \cdot h_t \tag{2.11}$$

Neben der Untersuchung des Leistungsvermögens einer Turbomaschine ist eine Analyse der Strömungsverluste von entscheidender Bedeutung. Verluste sind dabei allgemein durch einen Anstieg der Entropie gekennzeichnet und lassen sich durch eine Analyse des Totaldruckverhältnisses und des Wirkungsgrads einer Komponente charakterisieren. Die Entropie wird durch die Gibbs'sche Fundamentalgleichung beschrieben, welche einen funktionalen Zusammenhang mit den thermischen und kalorischen Zustandsgrößen darstellt. In der differentiellen Form lautet diese nach [K$^+$09b] folgendermaßen:

$$dh = \frac{1}{\rho}\, dp + Tds \tag{2.12}$$

Nach Integration entlang eines Stromfadens vom Eintritt zum Austritt ergibt sich:

$$\Delta h = \underbrace{\int_{in}^{out} \frac{1}{\rho}\, dp}_{w_{fluid}} \quad + \quad \underbrace{\int_{in}^{out} Tds}_{\int_{in}^{out}(Tds)_{rev}+\int_{in}^{out}(Tds)_{irr} = q+j} \tag{2.13}$$

Beim ersten Term handelt es sich um die spezifische Strömungsarbeit w_{fluid}. Der zweite Term, $\int Tds$, lässt sich in einen reversiblen Anteil, die übertragenen spezifische Wärmemenge q, und einen irreversiblen Anteil, die Dissipation j, aufspalten. Letztere beschreibt diejenige Energie, welche durch Reibungsvorgänge in innere Energie umgewandelt wird. Der irreversible Anteil j ist für reale, ergo nicht isentrope Zustandsänderungen grundsätzlich größer als Null, wohingegen das Vorzeichen von q von der Richtung des Wärmeaustausches abhängt.
Wird *Gl. (2.13)* in der Form $\Delta h = w_{fluid} + q + j$ in *Gl. (2.10)* eingesetzt, so ergibt sich:

$$w_{tech} = w_{fluid,t} + j \tag{2.14}$$

mit

$$w_{fluid,t} = w_{fluid} + \frac{\Delta v^2}{2} \tag{2.15}$$

In *Gl. (2.14)* stellt w_{tech} die von den Schaufeln übertragene bzw. entzogene Arbeit dar. Dagegen repräsentiert $w_{fluid,t}$ denjenigen Anteil der technischen Arbeit, welcher für den Fall einer Verdichterströmung letztendlich auf das Fluid übertragen wird. Die Differenz bildet somit die Dissipation j.
Wird die integrale Form der Gibbs'schen Fundamentalgleichung nach *Gl. (2.13)* betrachtet, gilt, dass zwar die Integration der Enthalpie wegunabhängig ist, jedoch nicht diejenige der Dissipation, der Wärmemenge sowie der Strömungsarbeit. Zur Ermittlung der Größen j, q und w_{fluid} werden also Informationen über den Verlauf der zu integrierenden Funktionen $T(s)$ und $\rho(p)$ zwischen Ein- und Austritt des betrachteten Kontrollvolumens benötigt. Dadurch, dass der reale Verlauf einer Zustandsänderung mathematisch schwierig zu erfassen ist, wird stattdessen eine Ersatzzustandsänderung mit zum Realfall identischen Eintritts- und Austrittbedingungen angenommen. Bei dieser handelt es sich um eine sogenannte polytrope Zustandsänderung. Diese ist dadurch gekennzeichnet, dass für jedes infinitesimale Element entlang des Integrationswegs das Verhältnis von Enthalpieänderung zur Änderung der spezifischen Strömungsarbeit konstant ist. Diese Konstante wird als Polytropenverhältnis ν_{poly} bezeichnet und lässt sich nach [K$^+$09b] wie folgt schreiben:

$$\nu_{poly} = \frac{dh}{dw_{fluid}} = 1 + \frac{dq + dj}{\frac{1}{\rho}dp} = 1 + \frac{q + j}{w_{fluid}} \tag{2.16}$$

Neben dem Polytropenverhältnis spielt der Polytropenexponent n_{poly} eine wichtige Rolle bei der Beschreibung dieser Ersatzzustandsänderung. Nach [Brä09] handelt es sich bei diesem um die Steigung der Polytrope, welche bei doppelt-logarithmischer Auftragung im p-v_{spec}-Diagramm eine Gerade darstellt. Mit Hilfe des Polytropenexponenten lässt sich eine polytrope Zustandsänderung allgemein wie folgt definieren (Polytropenbeziehung):

$$\frac{p}{\rho^{n_{poly}}} = konstant \tag{2.17}$$

Zur Beurteilung der aero-thermodynamischen Güte einer Komponente wird deren Wirkungsgrad η betrachtet, wobei für Gitterströmungen meist der isentrope und der polytrope Wirkungsgrad, η_{is} bzw. η_{poly}, zur Auswertung herangezogen werden. Zur Bestimmung von η_{is} wird die integrale Totalenthalpiedifferenz zwischen Ein- und Austritt der Komponente (Δh_t) mit derjenigen einer isentropen (s = konstant), also verlustfreien Zustandsänderung ($\Delta h_{t,is}$) verglichen. Zur Berechnung von η_{poly} hingegen dient die lokale isentrope Zustandsänderung als Vergleichsprozess. Es wird also ein infinitesimaler Enthalpieanstieg entlang der Polytrope, $dh_{t,poly}$ mit der lokalen isentropen Enthalpieänderung $dh_{t,is,poly}$ verglichen und damit die Divergenz der Isobaren im h-s-Diagramm berücksichtigt. *Abb. 2.3* verdeutlicht hierbei die Unterschiede in der Ermittlung der beiden Wirkungsgrade.

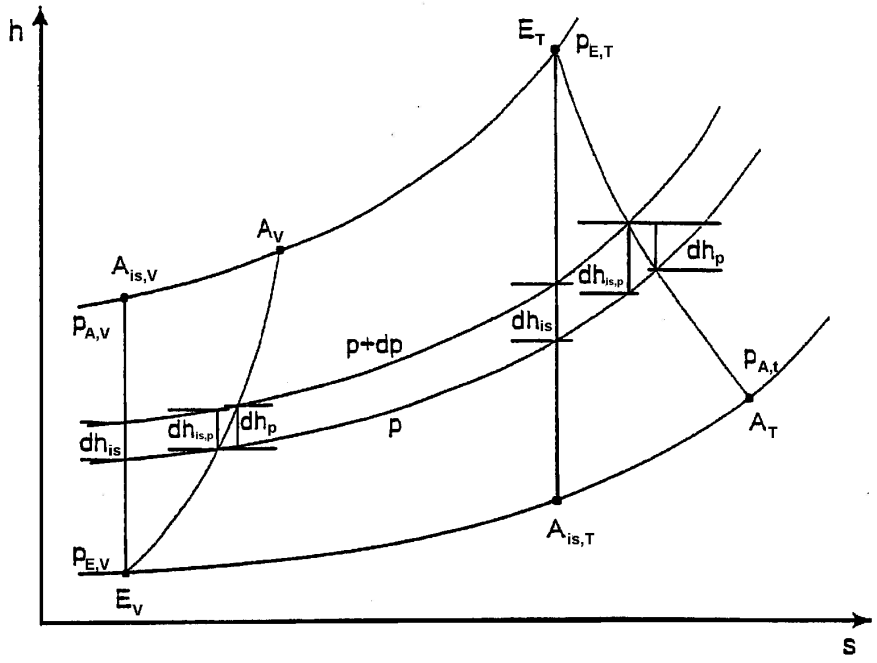


Abb. 2.3: Ermittlung des isentropen und polytropen Wirkungsgrads im h-s-Diagramm [K$^+$09a]

Für Verdichter ergeben sich folgende Definitionen für die beiden Wirkungsgrade:

$$\eta_{is} = \frac{\Delta h_{t,is}}{\Delta h_t}$$
$$\eta_{poly} = \frac{dh_{t,is,poly}}{dh_{t,poly}} \tag{2.18}$$

Aufgrund der Divergenz der Isobaren ist für Verdichter der polytrope Wirkungsgrad grundsätzlich höher als der isentrope, da die aufintegrierte Gesamtlänge $\sum dh_{t,is,poly}$ größer ist als $\sum dh_{t,is}$. Bei Turbinen hingegen ist η_{poly} stets kleiner. Für diese gilt:

$$\eta_{is} = \frac{\Delta h_t}{\Delta h_{t,is}}$$
$$\eta_{poly} = \frac{dh_{t,poly}}{dh_{t,is,poly}} \tag{2.19}$$

2.3 Betrachtung des Joule-Brayton-Prozesses

Im Rahmen dieses Unterkapitels wird der Joule-Brayton-Kreisprozess, welcher auch als Gasturbinen-Vergleichsprozess bekannt ist, vorgestellt. Durch die idealisierte Beschreibung können die thermodynamischen Zustandsänderungen, welche das Arbeitsmedium Luft beim Durchströmen einer Turbomaschine, beispielsweise einem Flugtriebwerk, durchläuft, qualitativ erfasst und einer vereinfachten analytischen Betrachtung unterzogen werden. Zur Veranschaulichung ist in *Abb. 2.4* der Joule-Brayton-Prozess im *h*-*s*-Diagramm dargestellt, wobei die verwendete Indizierung auf der international gebräuchlichen Gasturbinen-Notation beruht. Diese ist beispielhaft für ein einwelliges Turbojettriebwerk *Abb. 2.5* zu entnehmen. Für dieses Triebwerk wird auch die folgende Kreisprozess-Betrachtung durchgeführt.

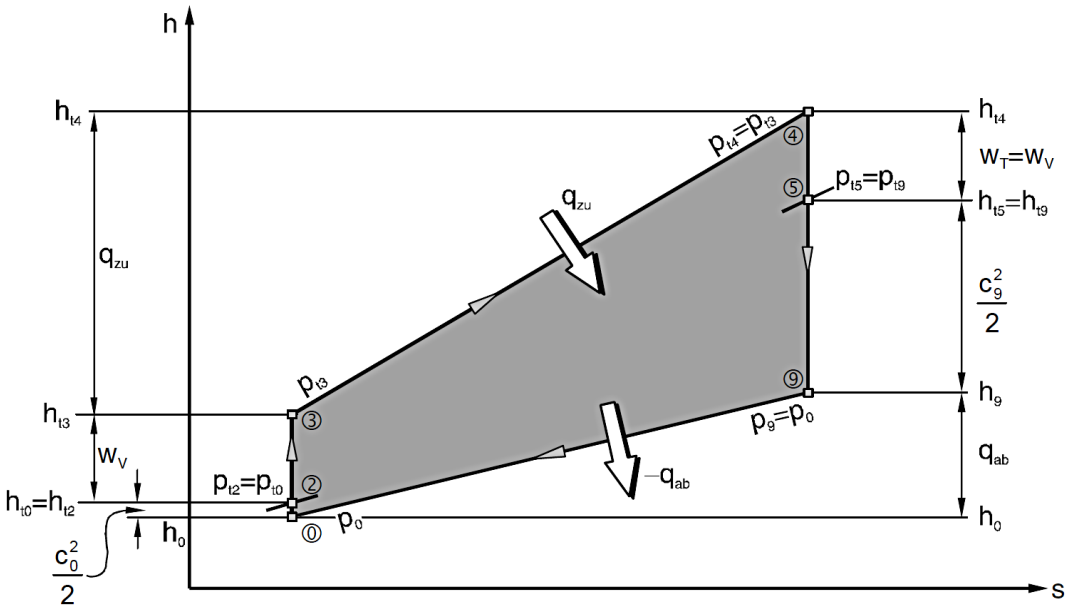


Abb. 2.4: Joule-Brayton-Kreisprozess im *h*-*s*-Diagramm [Brä09]

Für den Joule-Brayton-Prozess gelten nach [Brä09] folgende vereinfachenden Annahmen:

- isentrope Verdichtung und Expansion
- isobare Verbrennung

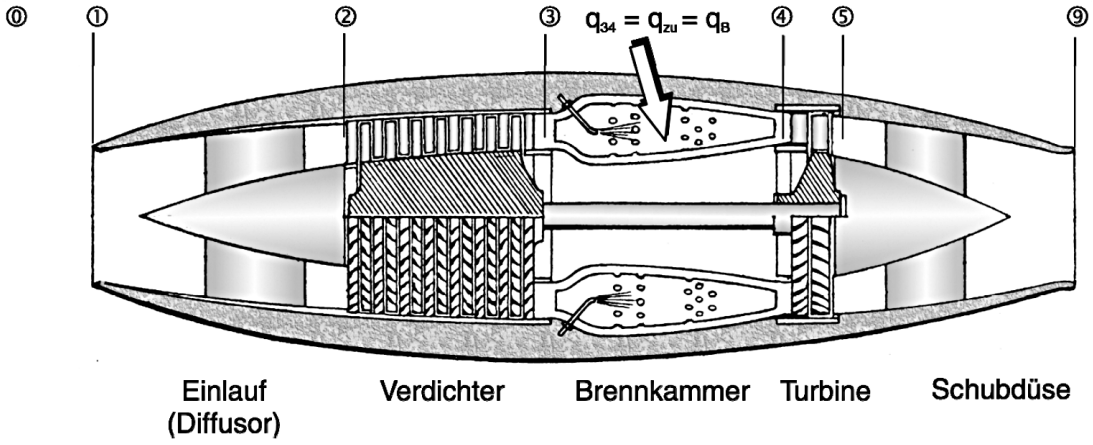


Abb. 2.5: Einwelliges Turbojettriebwerk mit Gasturbinen-Notation [Brä09]

- Massenstrom $\dot{m}$ ist konstant $\Rightarrow$ Treibstoff-Massenstrom $\dot{m}_{fuel}$ wird nicht berücksichtigt
- konstante Gaseigenschaften
- isobare Wärmeabfuhr zur (gedachten) Schließung des Kreisprozesses

Der Gasturbinen-Vergleichsprozess setzt sich aus insgesamt zwei Kompressionen und zwei Expansionen zusammen. Zunächst erfolgt durch den aerodynamischen Aufstau ein Anstieg des Druckniveaus vom Umgebungsdruck in der jeweiligen Flughöhe $p_0 = p_\infty$ auf den Totaldruck des Triebwerkseinlaufs p_{t1}. Über den Einlauf sinkt der Totaldruck aufgrund von Strömungsverlusten leicht ab. Anschließend erfolgt eine zweite, deutlich stärkere Kompression über den mehrstufigen Verdichter. In diesem wird der durchströmenden Luft über die Laufschaufeln Arbeit zugeführt, sodass sich der Energieinhalt der Strömung und damit auch der Totaldruck auf p_{t3} erhöht.
Die Nutzarbeit des Kreisprozesses w_{eff} ergibt sich aus:

$$w_{eff} = \oint Tds \tag{2.20}$$

Die Nutzarbeit entspricht also der von den beiden Isentropen und den beiden Isobaren eingeschlossenen Fläche im T-s-Diagramm (siehe *Abb. 2.6*). Aufgrund der Divergenz der Isobaren kann diese Fläche durch eine Zunahme des Verdichter-Kompressionsgrads überproportional gesteigert werden. Aus diesem Grund ist zur Optimierung der Nutzarbeit eine möglichst hohe Verdichtung bzw. ein möglichst hoher Verdichter-Enddruck p_{t3} anzustreben.
Im Anschluss an den Verdichter erfolgt in der Brennkammer die Energiezufuhr in Form einer Gleichdruckverbrennung. Wie bereits erwähnt, ändern sich beim Joule-Brayton-Prozess dabei weder der Massenstrom noch die Gaszusammensetzung. Im Gegensatz zur verlustfreien Betrachtung des Vergleichsprozesses sinkt in der Realität aufgrund von Strömungsverlusten das Totaldruckniveau über der Brennkammer. Das Absinken des Drucks ist gleichzeitig eine notwendige Voraussetzung, um eine ausreichende Kühlung der Turbinenschaufeln zu ermöglichen.

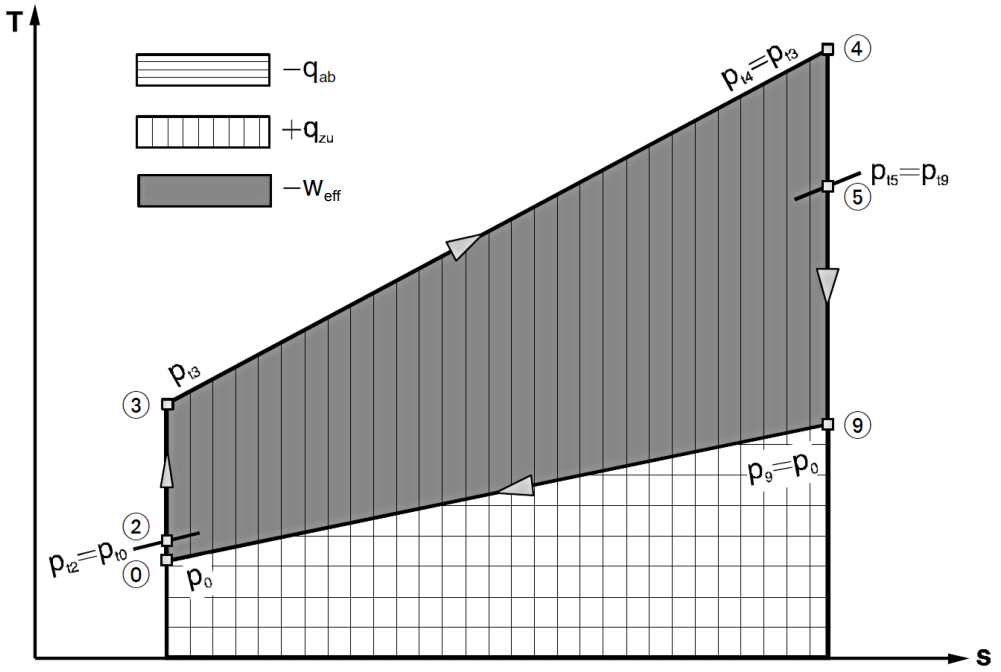


Abb. 2.6: Nutzarbeit w_{eff} des Joule-Brayton-Prozesses im T-s-Diagramm [Brä09]

In der Turbine schließlich findet die erste Phase der Expansion des Arbeitsmediums statt. Die Strömung überträgt mittels der Rotoren Energie an die Welle, welche wiederum den Verdichter antreibt. Unter Vernachlässigung mechanischer Verluste, zum Beispiel bedingt durch Lagerreibung, ergibt sich ein Leistungsgleichgewicht zwischen der Turboarbeits- und Turbokraftmaschinenkomponente:

$$P_{compressor} = |-P_{turbine}| \tag{2.21}$$

Wird der erste Hauptsatz für das Gesamttriebwerk aufgestellt, so ergibt sich nach [Brä09] unter Benutzung von *Gl. (2.21)* :

$$|w_{eff}| = q_{zu} - |q_{ab}| = \frac{v_9^2 - v_0^2}{2} \tag{2.22}$$

Die Nutzarbeit stellt diejenige Arbeit dar, welche je nach Einsatzzweck der Gasturbine dazu verwendet wird, um beispielsweise einen Hubschrauberrotor anzutreiben oder um, wie im Fall des Turbojettriebwerks, den geforderten Triebwerksschub zu erzeugen. Wie *Gl. (2.22)* dabei verdeutlicht, entspricht w_{eff} der Differenz der kinetischen Energie des Arbeitsmediums vor und hinter dem Flugtriebwerk.
Nach der ersten Expansion in der Turbine, welche beim Flugtriebwerk hauptsächlich dazu dient, die benötigte Leistung für den Verdichter sowie weiterer, hier vernachlässigter Abnehmer bereitzustellen, erfolgt die zweite Expansion über die Triebwerksdüse. Bei Flugtriebwerken ist es hierbei das Ziel, einen möglichst hohen Schub, welcher sich proportional zur Geschwindigkeitsdifferenz über das Triebwerk verhält, zu erreichen. Die Schubgleichung für das betrachtete Turbojettriebwerk lässt sich folgendermaßen schreiben:

$$F_{thrust} = \dot{m}_0 \cdot (v_9 - v_0) \tag{2.23}$$

Im Fall einer angepassten Düse verlässt der Luftstrahl schließlich das Triebwerk mit einem statischen Druck von $p_9 = p_0$. Dabei geht die noch im Abgasstrahl enthaltene Wärmeenergie verloren. Das entspricht der gedachten isobaren Wärmeabfuhr q_{ab} zur Schließung des Kreisprozesses.

2.4 Analyse von Gitterströmungen

Im Folgenden wird sich dieses Kapitel mit der Analyse von Gitterströmungen befassen. Zunächst werden dabei die aerodynamischen und kinematischen Grundlagen der Leistungsübertragung zwischen rotierender Schaufel und dem strömenden Fluid betrachtet. Dies bildet die Basis der anschließenden Herleitung der Eulerschen Hauptgleichung für Turbomaschinen. Daraufhin wird auf die Systematik der Geschwindigkeitsdreiecke zur Beschreibung der Strömungskinematik eingegangen. Schließlich wird im letzten Abschnitt ausgehend von den Ergebnissen der vorangegangenen Unterkapitel die erweiterte Form der Eulerschen Hauptgleichung sowie die Rothalpie abgeleitet und deren Eigenschaften diskutiert.
Anzumerken ist zunächst, dass aufgrund abweichender Notationen für die Geschwindigkeiten in den Abbildungen die einzelnen Geschwindigkeitskomponenten anders bezeichnet sind als im Text. Es gilt dabei:

- $\vec{c} \mathrel{\widehat{=}} \vec{v}_{abs}$
- $\vec{w} \mathrel{\widehat{=}} \vec{v}_{rel}$
- Index $u \mathrel{\widehat{=}}$ Index θ

2.4.1 Leistungsübertragung in Schaufelgittern - Eulersche Hauptgleichung der Turbomaschinen

In diesem Abschnitt wird die Leistungsübertragung in Schaufelgittern behandelt. Hierfür wird die Eulersche Hauptgleichung der Turbomaschinen sowohl anhand des Drallsatzes als auch mit Hilfe des Satzes von Kutta-Joukowski hergeleitet. Zunächst erfolgt dabei eine allgemeine Betrachtung der Energieumsetzung.
Wendet man die grundlegenden Gesetze und Erkenntnisse aus der Tragflügeltheorie auf die Beschaufelung von Turbomaschinen an, so wird aus *Abb. 2.7* ersichtlich, dass sich aufgrund der Ausbildung einer Profilsaug- und einer Profildruckseite eine Auftriebskraft auf die Schaufeloberfläche ergibt. Diese überlagert sich mit den tangential zur Oberfläche wirkenden Schubspannungen infolge Reibung zu einer resultierenden Gesamtkraft $\vec{F}$. Da Arbeit das Produkt aus Kraft und Weg darstellt, ist zur Leistungsübertragung folglich der Anteil der Gesamtkraft in Umfangs- bzw. Drehrichtung relevant. Ebenso wird ersichtlich, dass nur in bewegten Gittern (Rotoren) Arbeit übertragen werden kann, nicht aber in feststehenden Gittern (Statoren). Um das letztendlich für die Übertragung der Leistung verantwortliche Schaufelmoment bestimmen zu können, existieren mehrere unterschiedliche Ansätze. Zum einen ist es möglich dieses mittels einer Integration über die gesamte Schaufeloberfläche zu bestimmen und zum anderen kann hierfür

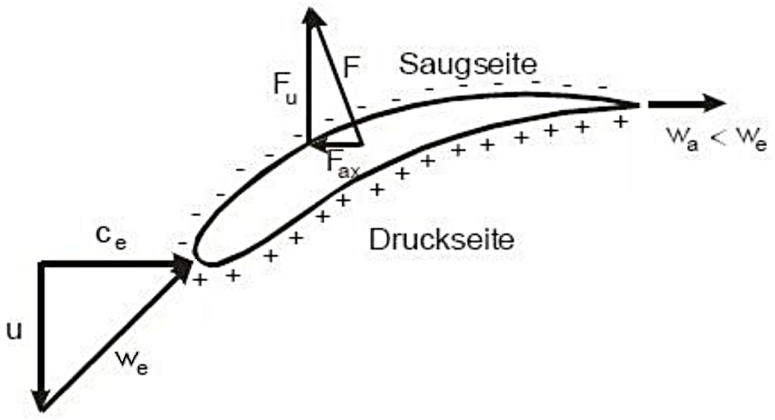


Abb. 2.7: Schaufelumströmung mit resultierender Strömungskraft [K$^+$09b]

eine integrale Betrachtung über die jeweilige Schaufelreihe durchgeführt werden. Letzterer Ansatz wird zur Herleitung der Eulerschen Hauptgleichung verwendet.
Die Eulersche Hauptgleichung der Turbomaschinen basiert grundsätzlich auf dem Drallsatz, also der Erhaltung des Strömungsdralls. Dieser lässt sich nach [K$^+$09b] wie folgt schreiben:

$$\frac{d}{dt}\vec{L}_0 = \sum \vec{M}_0 \tag{2.24}$$

Die totale zeitliche Ableitung des Dralls bezogen auf einen Punkt 0 entspricht also der Summe der Momente, ebenfalls auf den Punkt 0 bezogen. Dieser Punkt wird in Turbomaschinen so gewählt, dass er auf der Maschinenachse liegt. Der Drall L ergibt sich dabei allgemein als Kreuzprodukt zwischen Abstandsvektor $\vec{r}$ und Impulsvektor $\vec{I}$. hier betrachtet im Absolutsystem.

$$\vec{L} = \vec{r} \times \vec{I} = \vec{r} \times (m \cdot \vec{v}_{abs}) \tag{2.25}$$

Bezogen auf die Maschinenachse bzw. den Punkt 0 folgt für die Drallkomponente in Axialrichtung:

$$L_0 = r \cdot m\, v_{\theta,abs} \tag{2.26}$$

Unter Voraussetzung eines stationären Strömungszustands sowie einer unveränderlicher Geometrie des betrachteten Kontrollraums resultiert für die totale zeitliche Ableitung des Dralls:

$$\frac{d}{dt}\vec{L}_0 = \frac{\partial r}{\partial t}\, m\, v_{\theta,abs} + r\, \frac{\partial m}{\partial t}\, v_{\theta,abs} + r\, m\, \frac{\partial v_{\theta,abs}}{\partial t} = r\, \dot{m}\, v_{\theta,abs} \tag{2.27}$$

Wird als Kontrollvolumen ein Rotorgitter gewählt und auf dieses der Drallsatz angewandt, so ergibt sich unter Vernachlässigung der Momente am Ein- und Austritt für die relevante Axialkomponente des Dralls:

$$(\dot{m}\, r\, v_{\theta,abs})_2 - (\dot{m}\, r\, v_{\theta,abs})_1 = \sum M_{blade} + M_{hub} + M_{tip} \tag{2.28}$$

Nach [K$^+$09a] ist das Gehäusemoment M_{tip} infolge Reibung meist in guter Näherung vernachlässigbar. Das für die Leistungsübertragung relevante Rotordrehmoment stellt somit die Summe aus dem aufsummierten Schaufeldrehmoment $\sum M_{blade}$ und dem Nabendrehmoment M_{hub} dar.

$$M_{rotor} = \sum M_{blade} + M_{hub} \tag{2.29}$$

Hiermit lässt sich die übertragene Leistung folgendermaßen ermitteln:

$$P = M_{rotor} \cdot \omega \tag{2.30}$$

Insgesamt ergibt sich für einen konstanten Massenstrom $\dot{m}_1 = \dot{m}_2 = \dot{m}$ mit *Gl. (2.11)* und *Gl. (2.28)* :

$$\frac{P}{\dot{m}} = \Delta h_t = v_{\theta,abs,2} u_2 - v_{\theta,abs,1} u_1 \tag{2.31}$$

Bei dieser Gleichung handelt es sich um die Eulersche Hauptgleichung der Turbomaschinen. Mit deren Hilfe ist es möglich eine Verbindung zwischen thermodynamischer und kinematischer Betrachtungsweise von Gitterströmungen herzustellen, denn mit $\Delta h_t = c_p \cdot \Delta T_t$ gilt:

$$v_{\theta,abs,2} u_2 - v_{\theta,abs,1} u_1 = c_p \cdot \Delta T_t \tag{2.32}$$

Nach [Brä09] ist die Eulersche Hauptgleichung allgemeingültig, da sie sowohl zur Analyse reibungsfreier als auch reibungsbehafteter sowie kompressibler als auch inkompressibler Strömungen herangezogen werden kann. Eingeschränkt ist sie jedoch durch die Voraussetzung einer stationären Strömung sowie durch die Verwendung gemittelter Strömungsgrößen. Daraus folgend wird die Rotationssymetrie der untersuchten Konfiguration angenommen. *Gl. (2.32)* setzt zudem eine adiabate Strömung voraus.

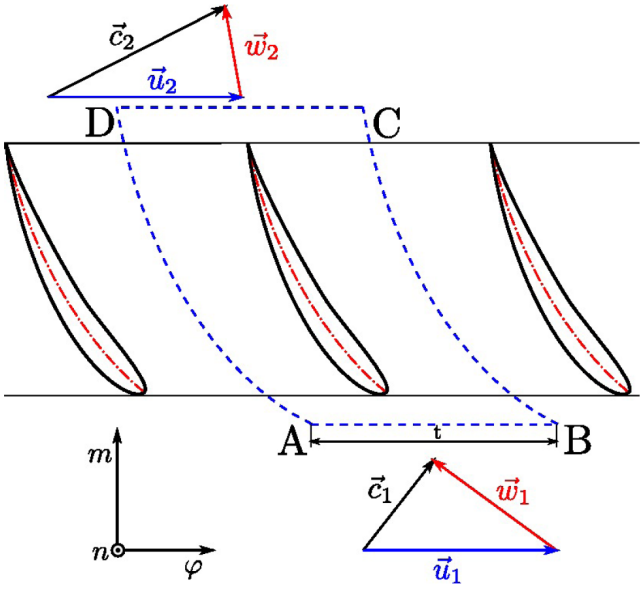


Abb. 2.8: Schaufelgitter mit Kurvenintegral zur Ermittlung der Zirkulation [K+09]

Nach [K+09b] lässt sich die Euler-Gleichung ebenso unter Verwendung der Zirkulation und des Kutta-Joukowski Gesetzes ermitteln. Für die Zirkulation gilt:

$$\Gamma = \oint \vec{v}\, d\vec{l} \tag{2.33}$$

Die Auswertung dieses Kurvenintegrals für eine Schaufel (siehe *Abb. 2.8*) ergibt:

$$\Gamma_{blade} = \oint_{ABCD} \vec{v}\, d\vec{l} = \int_A^B \vec{v}\, d\vec{l} + \oint_C^D \vec{v}\, d\vec{l} \tag{2.34}$$

Die Beiträge der Linienintegrale von B nach C und von D nach A sind betragsmäßig gleich groß, jedoch aufgrund der Laufrichtung von $d\vec{l}$ entgegengesetzt gerichtet und heben sich somit auf. An Ein- und Austritt ist aufgrund der Kollinearität zu $d\vec{l}$ jeweils nur die Umfangskomponente der Geschwindigkeit zu berücksichtigen. Es ergibt sich somit:

$$\Gamma_{blade} = \frac{2\pi}{N_{blades}} \left(r_2 \, v_{\theta,abs,2} - r_1 \, v_{\theta,abs,1}\right) \tag{2.35}$$

Das Rotordrehmoment lässt sich aus der Zirkulation Γ folgendermaßen ableiten:

$$M_{rotor} = \frac{N_{blades}}{2\pi} \Gamma_{blade} \cdot \dot{m} \tag{2.36}$$

Somit resultiert schließlich die aus *Gl. (2.31)* bereits bekannte Eulersche Hauptgleichung der Turbomaschinen:

$$\frac{P}{\dot{m}} = \Delta h_t = \omega \cdot \frac{N_{blades}}{2\pi} \Gamma_{blade} = c_{u2} u_2 - c_{u1} u_1 \tag{2.37}$$

Die Herleitung der erweiterten Euler-Gleichung sowie deren Diskussion erfolgt im übernächsten Kapitel, nachdem die dafür benötigten Grundlagen der Geschwindigkeitsdreiecke behandelt worden sind.

2.4.2 Kinematische Verhältnisse in Schaufelgittern - Systematik der Geschwindigkeitsdreiecke

Zur Analyse von Gitterströmungen in Turbomaschinen werden zwei unterschiedliche Bezugssysteme herangezogen, das Absolutsystem und das Relativsystem. Das absolute System ist dadurch gekennzeichnet, dass die Bewegung bzw. die Kinematik der Strömung aus der Sicht eines feststehenden Beobachters beschrieben wird. Es wird daher auch als gehäusefestes Bezugssystem bezeichnet. Im Relativsystem hingegen bewegt sich der Beobachter mit den rotierenden Schaufeln (schaufelfestes Bezugssystem). Nach [Brä09] erfolgt die Betrachtung der Strömungsverhältnisse bzw. der Schaufelumströmung eines Rotorgitters (rotierend) im Relativsystem und diejenige eines Statorgitters (feststehend) im Absolutsystem.
Im relativen Bezugssystem gelten dieselben Gesetze der Mechanik wie im Absolutsystem, da es sich um ein gleichförmig bewegtes Inertialsystem handelt. Zur Konvertierung der Gesetze muss die Galilei-Transformation auf das Relativsystem angewandt werden. Für die Geschwindigkeit folgt damit:

$$\vec{v}_{abs} = \vec{u} + \vec{v}_{rel} \tag{2.38}$$

Dabei bezeichnet $\vec{u}$ die Eigengeschwindigkeit des bewegten Relativsystems, welche in Turbomaschinen der Umfangsgeschwindigkeit entspricht. Diese ergibt sich wiederum als Kreuzprodukt aus Winkelgeschwindigkeit und Radius.

$$\vec{u} = \vec{\omega} \times \vec{r} \tag{2.39}$$

Der zweite Geschwindigkeitsanteil in *Gl. (2.38)* , $\vec{v}_{rel}$, stellt die Relativgeschwindigkeit dar, also die Bewegung der Strömung wie sie von einem mitrotierenden Beobachter

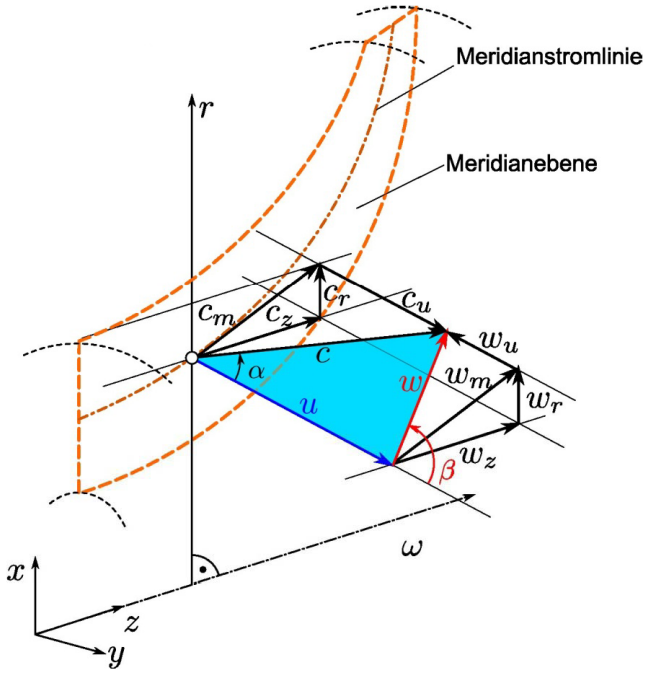


Abb. 2.9: Geschwindigkeitsdreick eines Radialverdichters [K+09]

wahrgenommen wird. Als Summe der beiden Vektoren resultiert der Vektor der Absolutgeschwindigkeit $\vec{v}_{abs}$.
Zur Auswertung der kinematischen Verhältnisse eines Schaufelgitters wird die Systematik der Geschwindigkeitsdreiecke benutzt. In *Abb. 2.9* ist hierfür beispielhaft ein Geschwindigkeitsdreieck eines Radialverdichters dargestellt. Die dreidimensionalen gemittelten Geschwindigkeitsvektoren aus *Gl. (2.38)* werden dabei auf eine zweidimensionale Meridianstromfläche projiziert. Die Meridiankomponente der Geschwindigkeit setzt sich also im Allgemeinen aus einem axialen sowie einem radialen Geschwindigkeitsanteil zusammen.

$$v_{m,abs} = v_{x,abs} + v_{r,abs} \qquad \text{bzw.} \qquad v_{m,rel} = v_{x,rel} + v_{r,rel} \tag{2.40}$$

In einem Geschwindigkeitsdreieck sind somit die Geschwindigkeitskomponenten im absoluten und relativen Bezugssystem in Meridional- und Umfangsrichtung aufgetragen. Die erstgenannte Komponente charakterisiert den Massendurchsatz der Passage und die zweite den Drall der Strömung. Wie aus der Eulerschen Turbinenhauptgleichung ersichtlich, repräsentiert dessen Änderung die Arbeitsumsetzung der Komponente. Es ergibt sich somit:

$$\vec{v}_{abs} = \vec{u} + \vec{v}_{rel} \qquad \Longleftrightarrow \qquad \begin{pmatrix} v_{m,abs} \\ v_{\theta,abs} \end{pmatrix} = \begin{pmatrix} v_{m,rel} \\ v_{\theta,rel} + u \end{pmatrix} \tag{2.41}$$

Zur Analyse der Kinematik werden zudem die Strömungswinkel herangezogen. Nach der Definition von [K+09a] werden diese folgendermaßen bezeichnet:

$$\text{Absolutströmungswinkel} \quad \alpha = \angle(\vec{v}_{abs}, \vec{u}) \tag{2.42}$$

$$\text{Relativströmungswinkel} \quad \beta = \angle(\vec{v}_{rel}, \vec{u}) \tag{2.43}$$

$$\text{Meridianströmungswinkel} \quad \epsilon = \angle(v_{m,abs}, v_{x,abs}) \tag{2.44}$$

Dabei werden α und β relativ zur positiven Umfangsrichtung gemessen.
Abb. 2.10 zeigt die charakteristischen kinematischen Verhältnisse für eine Verdichter-

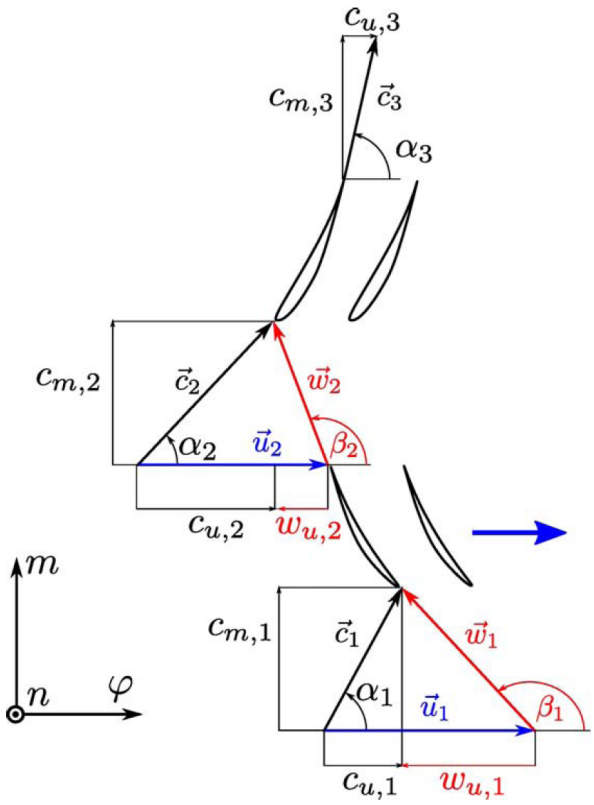


Abb. 2.10: Kinematische Verhältnisse einer Verdichterstufe [K+09]

stufe repräsentiert durch die Geschwindigkeitsdreiecke. Dabei wird zur Vereinfachung eine schaufelkongruente Zu- und Abströmung angenommen. Bei der ersten Schaufelreihe einer Stufe handelt es sich bei Verdichtern um den Rotor. Über diesen erfolgt ein Teil des Druckaufbaus, indem der Strömung über die Schaufeln Arbeit zugeführt wird. Diese Arbeitszufuhr ist mit einer Drallerzeugung im Absolutsystem verbunden ($v_{\theta,abs,2} > v_{\theta,abs,1}$). Im nachfolgenden Leitrad wird dann dieser Drall wieder abgebaut und die gewonnene Energie in einer weiteren Steigerung des statischen Drucks investiert. Hinzuzufügen ist, dass die Verdichterbeschaufelung aus relativ dünnen Profilen mit spitzer Vorderkante besteht und die Strömungsumlenkung und damit einhergehend die Schaufelbelastung im Vergleich zu Turbinen geringer ist.
Bei Verdichter-Schaufelreihen handelt es sich um Verzögerungsgitter. Für den Rotor gilt

somit $|\vec{v}_{rel,2}| < |\vec{v}_{rel,1}|$ und für den Stator analog $|\vec{v}_{abs,3}| < |\vec{v}_{abs,2}|$. Aufgrund der Verzögerung der Strömung in Verdichtergittern besteht die Gefahr der Grenzschicht- bzw. Strömungsablösung und damit verbunden eine Verschlechterung des Wirkungsgrads sowie weitere negative Folgen. Über die in den Geschwindigkeitsdreiecken enthaltenen Informationen lassen sich diverse Kenngrößen ableiten, mit deren Hilfe die Belastung der Schaufel und die Ablösegefahr der Strömung beurteilt werden kann. Nach [K$^+$09a] sind dabei die wichtigsten:

- DeHaller-Zahl:

$$DeHaller = \left(\frac{|\vec{v}_{rel,2}|}{|\vec{v}_{rel,1}|}\right) \quad \text{(Rotoren)}$$
$$DeHaller = \left(\frac{|\vec{v}_{abs,3}|}{|\vec{v}_{abs,2}|}\right) \quad \text{(Statoren)} \tag{2.45}$$

  Die DeHaller-Zahl beurteilt die Gefahr der Strömungsablösung am Gehäuse sowie an der Nabe. Sie sollte sowohl für Rotorgitter als auch Statorgitter $\geq 0{,}7$ sein, um ein Ablösen der Strömung zu vermeiden.

- Diffusionszahl:

$$D = \left(1 - \frac{|\vec{v}_{rel,2}|}{|\vec{v}_{rel,1}|}\right) + \frac{\Delta v_{\theta,rel}}{2\frac{s}{t}|\vec{v}_{rel,1}|} \quad \text{(Rotoren)}$$
$$D = \left(1 - \frac{|\vec{v}_{abs,3}|}{|\vec{v}_{abs,2}|}\right) + \frac{\Delta v_{\theta,abs}}{2\frac{s}{t}|\vec{v}_{abs,2}|} \quad \text{(Statoren)} \tag{2.46}$$

  Die Diffusionszahl charakterisiert die Ablösegefahr an den Seitenwänden und am Schaufelprofil, indem sie Strömungsverzögerung- und umlenkung im jeweiligen Relativsystem des Schaufelgitters betrachtet. Diese Kenngröße sollte $\geq 0{,}6$ sein, um Ablösungen zu vermeiden.

Abb. 2.11 zeigt das charakteristische Verhalten der Strömung durch eine Turbinenstufe. Bei Turbinen folgt der Rotor jeweils auf den Stator, die Anordnung ist also im Vergleich zu Verdichtern umgekehrt. Zunächst wird im Stator Drall erzeugt ($v_{\theta,abs,2} > v_{\theta,abs,1}$), wobei dies aufgrund des unbewegten Gitters nicht mit einem Leistungsumsatz verbunden ist. Der Drall wird im anschließenden Rotor unter Energieabgabe der Strömung wieder abgebaut. Bei Turbinenstufen handelt es sich im Gegensatz zu Verdichterstufen um Beschleunigungsgitter ($|\vec{v}_{abs,1}| > |\vec{v}_{abs,0}|$ bzw. $|\vec{v}_{rel,2}| > |\vec{v}_{rel,1}|$). Deshalb ist in Turbinen die Ablösegefahr geringer, da die langsam strömenden und damit prinzipiell ablösegefährdeten Grenzschichten durch die Beschleunigung der gesamten Strömung stabilisiert werden. Dadurch ist eine höhere Strömungsumlenkung pro Schaufelreihe möglich als beim Verdichter.
Außerdem unterscheiden sich Turbine und Verdichter in der Gestaltung der Schaufelgeometrie. So sind Turbinenschaufeln vergleichsweise füllig und weisen eine runde Vorderkante auf. Die hauptsächlichen Auslegungskriterien sind, ebenso wie beim Verdichter, eine effiziente Strömungsführung sowie zusätzlich die Gewährleistung der erforderlichen Schaufelkühlung.

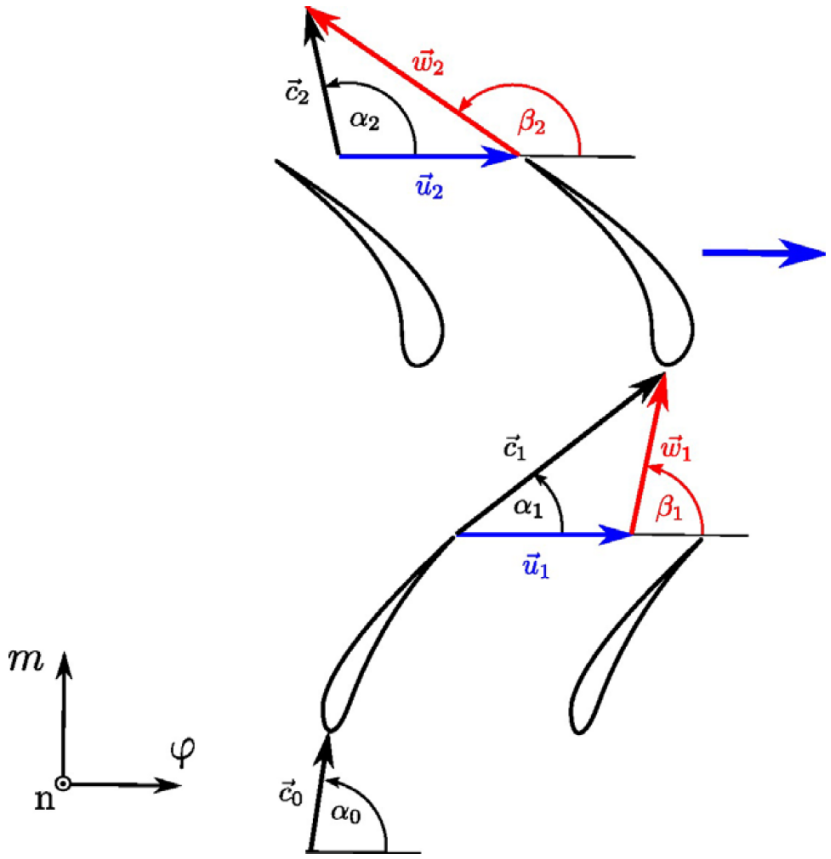


Abb. 2.11: Kinematische Verhältnisse einer Turbinenstufe [K$^+$09]

2.4.3 Diskussion der Eulerschen Hauptgleichung der Turbomaschinen

Im Rahmen dieses Abschnitts wird ausgehend von der Eulerschen Hauptgleichung der Turbomaschinen nach *Gl. (2.31)* die erweiterte Form hergeleitet und anschließend deren Terme diskutiert. Ebenso wird eine weitere Größe, die Rothalpie, abgeleitet und auf deren Eigenschaften eingegangen. Für weiterführende Informationen zu dieser Thematik wird insbesondere auf [Brä09] verwiesen.
Zunächst wird nun die erweitere Form der Eulerschen Hauptgleichung hergeleitet. Mit Hilfe des Kosinussatzes lässt sich für ein beliebiges Geschwindigkeitsdreieck folgende Beziehung ermitteln:

$$v_{rel}^2 = v_{abs}^2 + u^2 - 2 \cdot v_{abs}\ u \cdot \cos\alpha \tag{2.47}$$

mit

$$\cos\alpha = \frac{v_{\theta,abs}}{u} \tag{2.48}$$

ergibt sich:

$$u\, v_{\theta,abs} = \frac{1}{2}\left(v_{abs}^2 + u^2 - v_{rel}^2\right) \tag{2.49}$$

Wird dieser Term in die Eulersche Hauptgleichung nach *Gl. (2.31)* eingesetzt, so folgt daraus deren erweiterte Form:

$$\Delta h_t = \frac{v_{abs,2}^2 - v_{abs,1}^2}{2} - \frac{v_{rel,2}^2 - v_{rel,1}^2}{2} + \frac{u_2^2 - u_1^2}{2} \tag{2.50}$$

Anhand *Gl. (2.50)* wird im Folgenden auf die Bedeutung der einzelnen auftretenden Terme eingegangen.

- $\boxed{\frac{\mathbf{v_{abs,2}^2 - v_{abs,1}^2}}{\mathbf{2}}}$

  Der erste Term beschreibt die Änderung der kinetischen Energie und damit auch diejenige des Dralls bzw. der Komponente der Absolutgeschwindigkeit in Umfangsrichtung $v_{\theta,abs}$ über den Rotor. Aufgrund der Zufuhr von Arbeit ist dieser Term für Verdichter im Normalfall positiv, denn:

  $$w_{compressor} > 0 \quad \Rightarrow \quad v_{\theta,abs,2}\, u_2 - v_{\theta,abs,1}\, u_1 > 0$$

  Für Turbinen hingegen ist der Term negativ, da:

  $$w_{turbine} < 0 \quad \Rightarrow \quad v_{\theta,abs,2}\, u_2 - v_{\theta,abs,1}\, u_1 < 0$$

- $\boxed{\frac{\mathbf{v_{rel,2}^2 - v_{rel,1}^2}}{\mathbf{2}}}$

  Dieser Term beschreibt die Änderung der statischen Zustandsgrößen über den Rotor, wobei diese für das absolute und relative Bezugssystem identisch sind. Denn nach [Brä09] lässt sich zeigen:

  $$\frac{v_{rel,2}^2 - v_{rel,1}^2}{2} = \frac{\gamma}{\gamma - 1} \cdot \frac{p_1 - p_2}{\rho_1 - \rho_2} \tag{2.51}$$

  Für inkompressible Strömungen ist die Änderung des statischen Drucks somit direkt proportional zum betrachteten Term. Da die Hauptaufgabe eines Verdichters die Steigerung des statischen Drucks ist, gilt für diesen:

  $$p_2 > p_1 \quad \rightarrow \quad \frac{v_{rel,2}^2 - v_{rel,1}^2}{2} < 0 \qquad \text{Verzögerungsgitter}$$

  Für Turbinen ergibt sich hingegen folgender Zusammenhang:

  $$p_2 < p_1 \quad \rightarrow \quad \frac{v_{rel,2}^2 - v_{rel,1}^2}{2} > 0 \qquad \text{Beschleunigungsgitter}$$

- $\boxed{\frac{\mathbf{u_2^2 - u_1^2}}{\mathbf{2}}}$

  Dieser Term berücksichtigt den Einfluss der Zentrifugalkraft auf die Arbeitsumsetzung. Falls der mittlere Radius nicht konstant ist, ergibt sich ein zusätzlicher Arbeitsterm infolge der Zentrifugalkraft und der Zentrifugalbewegung (Verdichter) bzw. Zentripetalbewegung (Turbine) der Strömung. Ausgehend von der durch die Drehbewegung hervorgerufenen Zentrifugalkraft auf ein Fluidteilchen

  $$dF_z = r\,\omega^2 \cdot dm \tag{2.52}$$

  ergibt sich der resultierende Arbeitsterm zu:

  $$w_z = \omega^2 \int_{r_1}^{r_2} r dr = \omega^2 \cdot \frac{r_2^2 - r_1^2}{2} = \frac{u_2^2 - u_1^2}{2} \tag{2.53}$$

Die Änderung der Fliehkraft dF_z hat wiederum eine Druckänderung in Radialrichtung zur Folge. Nach [Brä09] lässt sich zeigen, dass gilt:

$$\frac{u_2^2 - u_1^2}{2} = \frac{\Delta p}{\rho} \tag{2.54}$$

Für Verdichter ist der betrachtete Term somit positiv und für Turbinen negativ. Aufgrund seiner Dominanz gegenüber den anderen beiden Termen besitzen Radialmaschinen ein höheres Leistungsvermögen als Axialmaschinen.

Nun wird die Herleitung der Rothalpie h_{rot} vorgestellt sowie auf die Bedeutung dieser Strömungsgröße eingegangen. Die Totalenthalpiedifferenz über eine Schaufelreihe lässt sich folgendermaßen schreiben:

$$\Delta h_t = h_2 - h_1 + \frac{v_{abs,2}^2}{2} - \frac{v_{abs,1}^2}{2}$$

Wird diese der erweiterten Euler-Gleichung nach *Gl. (2.50)* gleichgesetzt, so folgt:

$$\underbrace{h_1 + \frac{v_{rel,1}^2}{2} - \frac{u_1^2}{2}}_{h_{rot,1}} = \underbrace{h_2 + \frac{v_{rel,2}^2}{2} - \frac{u_2^2}{2}}_{h_{rot,2}} = konstant$$

Die sich ergebende, zwischen Ein- und Austritt des Rotorgitters konstante Strömungsgröße wird dabei als Rothalpie h_{rot} bezeichnet.

$$h_{rot} = h + \frac{v_{rel}^2}{2} - \frac{u^2}{2} = h_{t,rel} - \frac{u^2}{2} \tag{2.55}$$

$$h_{rot} = h_t - \left(\frac{v_{abs}^2}{2} + \frac{u^2}{2} - \frac{v_{rel}^2}{2} \right) = h_t - u\, v_{\theta,abs} \tag{2.56}$$

Die erste Schreibweise der Rothalpie nach *Gl. (2.55)* zeigt, dass die Rothalpie die relative Totalenthalpie abzüglich der kinetischen Energie des rotierenden Bezugssystems darstellt. Wie bereits erwähnt, ist das Kennzeichen dieser Strömungsgröße, dass sie über den Rotor entlang eines Stromfadens konstant ist. Nach [Brä09] gilt dies jedoch nicht in Gehäusenähe. Denn dort übt das bei der Herleitung der Eulerschen Hauptgleichung vernachlässigte Gehäusedrehmoment M_{tip} (siehe *Gl. (2.28)*) infolge von Reibungseffekten einen signifikanten Einfluss auf die Strömung aus. In der Realität ist somit die Rothalpiedifferenz niemals Null. Folglich stellt Δh_{rot} eine Messgröße zur Abschätzung der Verluste dar. Aus *Gl. (2.56)* und *Gl. (2.31)* resultiert schließlich folgende Gleichung:

$$\frac{P}{\dot{m}} = u_2\, v_{\theta,abs,2} - u_1\, v_{\theta,abs,1} = (h_{t2} - h_{t1}) - \Delta h_{t,rot} \tag{2.57}$$

Abb. 2.12 zeigt abschließend das *h-s*-Diagramm einer Radialverdichterstufe. Mit dessen Hilfe lässt sich sowohl die Arbeitsumsetzung als auch die Verluste grafisch veranschaulichen. Zudem können charakteristische Kenngrößen der einzelnen Schaufelgitter oder der Gesamtstufe wie zum Beispiel der isentrope Wirkungsgrad oder der Reaktionsgrad basierend auf der statischen Enthalpie (ρ_h) grafisch bestimmt werden. Für Letzteren gilt:

$$\rho_h = \frac{\Delta h_{rotor}}{\Delta h_{stage}} \tag{2.58}$$

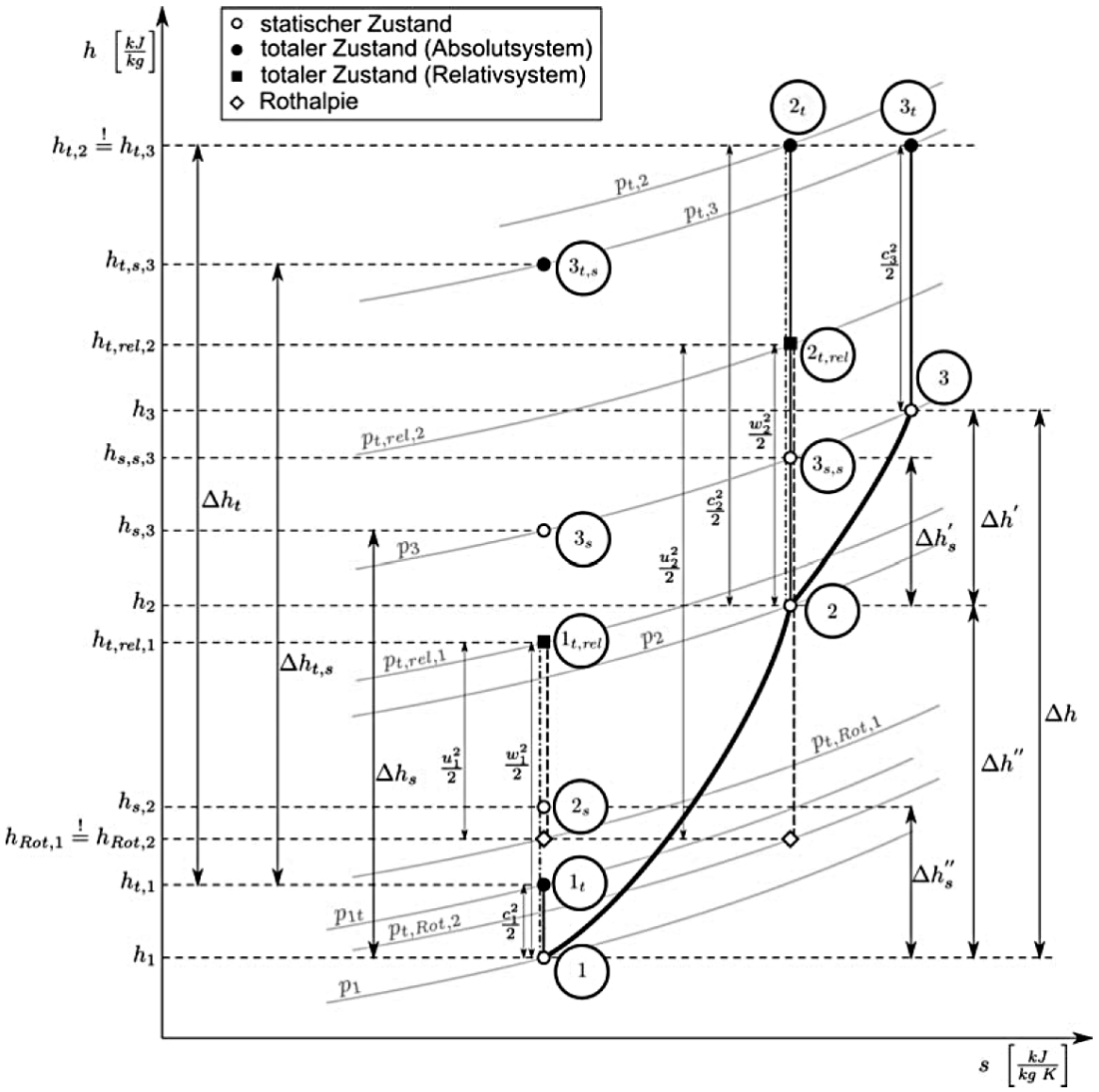


Abb. 2.12: Radialverdichterstufe im h-s-Diagramm [K$^+$09]

Der Reaktionsgrad beschreibt also den Anteil des Rotors an der Gesamtenthalpieerhöhung über der Stufe. Oft wird auch anstelle des enthalpiebasierten ein druckbasierter Reaktionsgrad zur Charakterisierung der Durchströmung der Stufe verwendet. Ebenso wird im h-s-Diagramm der Einfluss der im Rahmen dieses Kapitels diskutierten Terme der erweiterten Eulerschen Hauptgleichung auf die Arbeitsumsetzung deutlich.

2.4.4 Auswertung der Strömungsverhältnisse auf repräsentativen Stromflächen

Zur Auswertung der Strömungsverhältnisse in Schaufelgittern ist es üblich die Strömung auf repräsentativen zweidimensionalen Flächen zu betrachten. Bei diesen kann es sich um Stromflächen handeln oder um geometrische Flächen. Erstere stellen das Ergebnis einer Strömungssimulation dar, wohingegen geometrische Flächen, wie der Name bereits sagt, über die Geometrie der analysierten Konfiguration definiert werden. Nach [K$^+$09b] erfolgt häufig eine Approximation der Stromflächen durch geometrische Flächen. Diese werden somit als geometrische Stromflächen bezeichnet.
Im Einzelnen werden nach [May08] und [Wu52] folgende geometrische Stromflächen unterschieden:

S1-Fläche Geometrische Stromfläche, welche zwischen zwei Schaufeln aufgespannt ist. Der Normalenvektor steht orthogonal zur Meridionalrichtung.

S2-Fläche Geometrische Stromfläche, welche zwischen Nabe und Gehäuse aufgespannt ist. Der Normalenvektor steht orthogonal zur Meridionalrichtung.

S3-Fläche Geometrische Stromfläche, welche zwischen zwei Schaufeln und zwischen Nabe und Gehäuse aufgespannt ist. Der Normalenvektor entspricht der Strömungsrichtung.

Abb. 2.13 veranschaulicht beispielhaft die Lage der S1- und S2-Flächen in einem Schaufelgitter. Anzumerken ist, dass im Rahmen dieser Arbeit eine mittlere S2-Fläche (S2M-Fläche) betrachtet wird, welche jedoch nicht, wie im allgemeinen Fall, dreidimensional, sondern zweidimensional in xr-Koordinaten definiert ist.

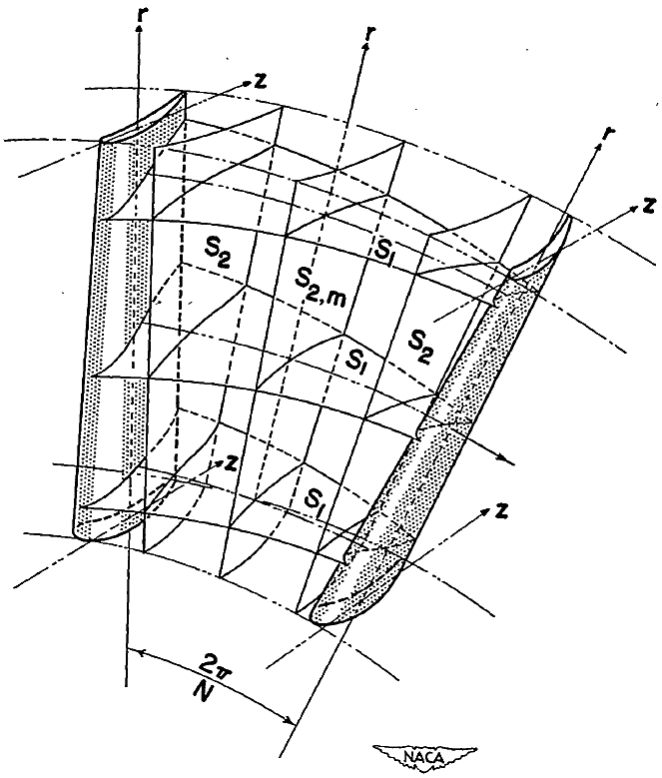


Abb. 2.13: S1- und S2-Stromflächen in einem Schaufelgitter [Wu52]

Kapitel 3

Numerische Strömungsmechanik

Die numerische Strömungsmechanik bzw. *Computational Fluid Dynamics* (CFD) wird seit den 1980er Jahren zur Lösung strömungsmechanischer Problemstellungen verwendet. Ihr Anwendungsbereich hat sich dabei durch die stetige Steigerung der zur Verfügung stehenden Rechnerleistung über die letzten Jahre zunehmend erweitert. Nach [K$^+$09] sind hierfür unter anderem folgende Vorteile gegenüber experimentellen Versuchen verantwortlich:

- Senkung von Projektkosten
- Verkürzung der Entwicklungszeit
- Möglichkeit der automatisierten Optimierung
- Möglichkeit der Analyse von nur unter hohem Aufwand oder gar nicht messbaren strömungsphysikalischen Phänomenen

Zu bedenken ist jedoch, dass die CFD je nach Abstraktionsgrad auf diversen Modellannahmen und Vereinfachungen beruht. Aus diesem Grund folgt die Notwendigkeit der Validierung der Implementierung des jeweiligen Strömungslösers anhand experimentell vermessener Referenzfälle.
Anzumerken ist, dass sich dieses Kapitel im Folgenden am gleichnamigen Kapitel von [Kal10] orientiert.

3.1 Grundgleichungen der Strömungsmechanik

Die Basis der physikalisch-mathematischen Strömungs-Modellierung bilden die sogenannten Grundgleichungen der Strömungsmechanik. Zur Beschreibung des thermo-fluiddynamischen Zustands sind für eine dreidimensionale, instationäre, kompressible, laminare oder turbulente Strömung unter Voraussetzung eines idealen Gases sowie unter Vernachlässigung von Volumenkräften sechs unbekannte Strömungsgrößen zu ermitteln. Bei diesen handelt es sich um die Dichte ρ, den Druck p, die Temperatur T sowie den dreidimensionalen Geschwindigkeitsvektor $\vec{v}$.

Zur Lösung dieser sechs Unbekannten steht das Gleichungssystem der Grundgleichungen, bestehend aus der Kontinuitätsgleichung (*Gl. (3.1)*), den drei Impulserhaltungsgleichungen (*Gl. (3.2)*) sowie der Energieerhaltungsgleichung (*Gl. (3.3)*) zur Verfügung. Zusätzlich wird die Zustandsgleichung für ein kalorisch ideales Gas herangezogen (*Gl. (3.4)*). Nach [O+09] lassen sich die Grundgleichungen der Strömungsmechanik unter Verwendung der Einsteinschen Summenkonvention folgendermaßen schreiben:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v_i)}{\partial x_i} = 0 \tag{3.1}$$

$$\frac{\partial (\rho v_i)}{\partial t} + \frac{\partial (\rho v_i v_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\frac{\partial v_k}{\partial x_k}\right] \tag{3.2}$$

$$\rho c_p\left(\frac{\partial T}{\partial t} + v_j\frac{\partial T}{\partial x_j}\right) = \frac{\partial p}{\partial t} + v_j\frac{\partial p}{\partial x_j} + \frac{\partial}{\partial x_j}\left[\lambda\frac{\partial T}{\partial x_j}\right] + \rho\dot{q}_s + \mu\Phi \tag{3.3}$$

$$e = c_p\, T - \frac{p}{\rho} \tag{3.4}$$

3.2 Strömungsphysikalische Modellbildung - Turbulenzmodellierung

Bei der Analyse technisch relevanter Problemstellungen treten nahezu ausschließlich turbulente Strömungen auf. Diese sind durch ein dreidimensionales, instationäres Strömungsfeld gekennzeichnet. Dabei sind einer makroskopisch erkennbaren Hauptströmung scheinbar regellose Schwankungsbewegungen in allen drei Raumrichtungen überlagert. Die exakte Beschreibung dieses komplexen Strömungszustands, also inklusive der kompletten Abbildung aller Längen- und Zeitskalen der Schwankungen, erfordert eine hohe zeitliche sowie räumliche Auflösung der zu untersuchenden Konfiguration.
Aufgrund der mit der erforderlichen hohen Auflösung verbundenen Anforderungen hinsichtlich der Rechenleistung werden in der Praxis diverse vereinfachende Annahmen zur Beschreibung des Strömungszustands gemacht. Es lassen sich hierbei drei hauptsächliche Modellierungsansätze turbulenter Strömungen unterscheiden:

- DNS (*Direct numerical simulation*)
- LES (*Large eddy simulation*)
- RANS (*Reynolds-averaged Navier–Stokes*)

Der grundlegende Vorteil der DNS besteht darin, dass diese direkt auf den Grundgleichungen beruht und keinerlei vereinfachenden Modellierungsannahmen benötigt. Aufgrund der Tatsache, dass sich der Gesamtaufwand einer DNS proportional zur dritten Potenz der Reynolds-Zahl verhält, ist der numerische und der damit verbundene zeitliche Aufwand für die überwiegende Zahl von Anwendungen deutlich zu hoch. Deshalb beschränkt sich der Einsatz dieser Methode auf den Bereich der Grundlagenforschung.
In der heutigen Praxis, insbesondere für industrielle Anwendungen, ist der RANS-Ansatz weit verbreitet. Dieser beruht darauf, dass die strömungsmechanischen Grundgleichungen für zeitlich gemittelte Strömungsgrößen gelöst werden. Es erfolgt somit

keine Auflösung der instationären turbulenten Wirbelstrukturen. Folglich sind für die turbulenten Schwankungsbewegungen bzw. für deren Einfluss auf das zeitlich gemittelte Strömungsfeld geeignete Modellierungsannahmen erforderlich.
Einen Mittelweg zwischen DNS und RANS stellt die LES dar. Bei dieser Methode werden die großen Wirbelstrukturen vollständig berechnet und ausschließlich die kleinen modelliert. Mit zunehmender verfügbaren Rechenleistung gewinnt dieser Ansatz immer weiter an Bedeutung.
Im Rahmen dieses Kapitels, welches die Modellierung turbulenter Strömungen behandelt, wird zunächst auf die physikalischen Grundlagen der Turbulenz eingegangen. Anschließend beschäftigt sich der folgende Abschnitt mit der Reynolds-Mittelung sowie dem Wirbelviskositätsansatz. Abschließend wird das in *TRACE* implementierte k-ω Turbulenzmodell nach *Wilcox* vorgestellt.

3.2.1 Physikalische Betrachtung der Turbulenz

Die laminar-turbulente Transition einer Strömung erfolgt, wenn die das Strömungsproblem beschreibende Reynolds-Zahl einen charakteristischen Grenzwert überschreitet. Diese Kennzahl beschreibt dabei, ob eine Strömung von Reibungskräften (kleine Re-Zahlen) oder von Trägheitskräften (große Re-Zahlen) dominiert wird. Zur Ermittlung der Reynolds-Zahl des Problems wird das Produkt aus einer für die untersuchte Konfiguration charakteristischen Geschwindigkeit und Länge auf die kinematische Viskosität des Fluids ν bezogen. Es gilt also:

$$Re = \frac{\text{Trägheitskräfte}}{\text{Reibungskräfte}} = \frac{v_{char}\, l_{char}}{\nu} \tag{3.5}$$

Wenn der für die betrachtete Problemstellung charakteristische kritische Wert der Reynolds-Zahl überschritten wird, klingen kleine Störungen nicht mehr ab. Die Strömung wird somit instabil. Diese Instabilität stellt die Grundlage der Turbulenz dar.
Je nach Art des untersuchten Falls bilden sich dann typische primäre Instabilitäten aus, welche die größten Wirbelstrukturen im Strömungsfeld darstellen. So tritt zum Beispiel bei der Analyse von Scherschicht-Strömungen die sogenannte Kelvin-Helmholtz-Instabilität auf, welche die ersten Wirbel erzeugt. Steht hingegen die Strömung in Kontakt mit festen Wänden, zum Beispiel bei Gitterströmungen, so handelt es sich bei den primären Instabilitäten zumeist um Ablösungen.
Die primären Wirbelstrukturen sind ebenso instabil gegenüber kleinen Störungen sowie Wechselwirkungen mit anderen Wirbeln. Folglich zerfallen diese in immer kleinere Strukturen. Dieser Zerfallsprozess wird als Wirbelkaskade bzw. Energiekaskade bezeichnet und ist in *Abb. 3.1* dargestellt. Eine turbulente Strömung ist somit dadurch gekennzeichnet, dass der Wirbelkaskade durch die primären Instabilitäten stetig kinetische Energie aus der Hauptströmung zugeführt wird. Durch den Zerfallsprozess der Wirbelstrukturen wird diese an immer kleinere Wirbel transferiert. Schließlich wird die übertragene kinetische Energie in den kleinsten Skalen, den dissipativen Skalen, in innere Energie umgewandelt ([Hir09]). Auf dieser Ebene dominieren also die viskosen Kräfte.
Zusammengefasst bewirkt die Wirbelkaskade als Grundlage turbulenter Strömungen

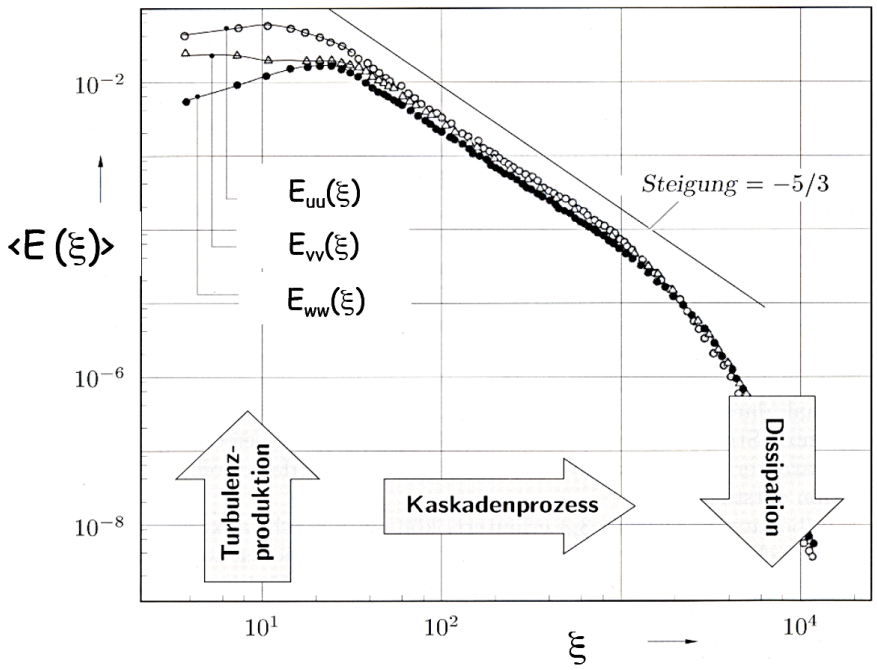


Abb. 3.1: Energiekaskade bzw. Wirbelkaskade [Hic10]

die enorme Vielfalt an unterschiedlichen in der Gesamtströmung auftretenden Längen- und Zeitskalen, deren exakte Auflösung nur mit relativ großem numerischen Aufwand möglich ist.

3.2.2 Reynolds-Mittelung und Schließungsproblem

Wie bereits erwähnt, ist der zeitliche Aufwand einer DNS für die meisten Anwendungen zu hoch. Zudem basieren Auslegungsberechnungen hauptsächlich auf der Analyse von mittleren Strömungsfeldern; Details der turbulenten Schwankungsbewegungen sind demnach von untergeordnetem Interesse. In der heutigen Praxis werden deshalb die Grundgleichungen in der überwiegenden Mehrzahl der Fälle für die zeitlichen Mittelwerte der Strömungsgrößen gelöst. Die Grundlage hierfür bildet die Reynolds-Mittelung. Nach [Wil06] ist die Basis der von *Reynolds* im Jahr 1895 veröffentlichten Theorie die Aussage, dass sich eine beliebige Strömungsgröße wie zum Beispiel die Geschwindigkeit $v(x_i,\, t)$ in einen zeitlichen Mittelwert $V(x_i)$ sowie einen Schwankungswert $v'(x_i, t)$ aufspalten lässt.

$$v(x_i, t) = V(x_i) + v'(x_i, t) \tag{3.6}$$

mit

$$V_i(x_i, t) = \frac{1}{T} \int_t^{t+T} v_i(x_i, t)\, dt' \quad \text{mit } T_1 << T << T_2 \tag{3.7}$$

Um diesen Separationsansatz auch auf instationäre Strömungen anwenden zu können, muss nach [S$^+$10a] das Zeitmaß der turbulenten Fluktuationen (T_1) sehr viel kleiner sein als das Zeitmaß der instationären Strömung (T_2). Der Mittelwert einer Strömungsgröße ist in diesem Fall dann nicht mehr zeitlich konstant. Die Reynolds-Mittelung bzw. die

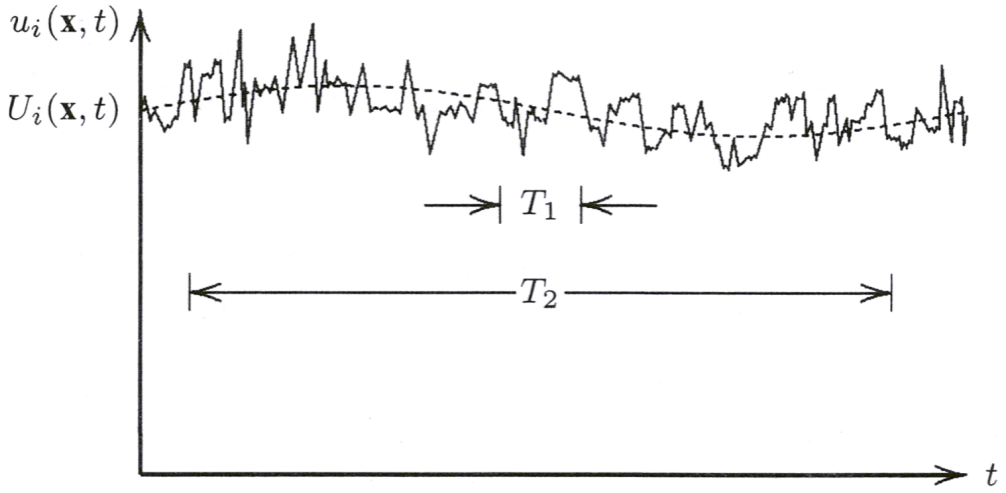


Abb. 3.2: Reynolds-Mittelung des Geschwindigkeitssignals für eine instationäre Strömung [Wil06]

Verläufe des messbaren Geschwindigkeitssignals und des zeitlichen Mittelwerts dieses Signals sind in *Abb. 3.2* für eine instationäre Strömung dargestellt. Der Schwankungswert ergibt sich dabei durch die jeweilige Differenz beider Verläufe.
Werden die Strömungsgrößen in den Navier-Stokes-Gleichungen bzw. Impulserhaltungsgleichungen (*Gl. (3.2)*) durch die jeweilige Summe aus zeitlichem Mittelwert und Schwankungswert ersetzt und anschließend eine zeitliche Mittelung der gesamten Gleichung durchgeführt, so ergeben sich unter Anwendung der Regeln zur Mittelwertsberechnung die Reynolds-gemittelten Navier-Stokes-Gleichungen (*Reynolds-Averaged-Navier-Stokes*, RANS).

$$\frac{\partial(\rho V_i)}{\partial t} + \frac{\partial(\rho V_i V_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial T_{ij}}{\partial x_j} - \frac{\partial(\rho\,\overline{v_i' v_j'})}{\partial x_j} \tag{3.8}$$

Im Folgenden wird zur Vereinfachung eine inkompressible Strömung betrachtet. Die Dichte ρ kann somit vor den Differentialoperator gezogen werden. Unter dieser Voraussetzung ergibt sich zudem aus der Kontinuitätsgleichung die Divergenzfreiheit des Geschwindigkeitsfelds.
Für ein Newtonsches Fluid lässt sich nach [Wil06] der in *Gl. (3.8)* auftretende mittlere Spannungstensor T_{ij} in Abhängigkeit vom Scherratentensor S_{ij} und der dynamischen Viskosität des Fluids μ wie folgt beschreiben:

$$T_{ij} = 2\mu S_{ij} \tag{3.9}$$

mit

$$S_{ij} = \frac{1}{2}\left(\frac{\partial V_i}{\partial x_j} + \frac{\partial V_j}{\partial x_i}\right) \tag{3.10}$$

Gegenüber den ursprünglichen Impulserhaltungsgleichungen tritt bei den RANS-Gleichungen ein zusätzlicher Term auf der rechten Seite auf. Dieser beschreibt den Einfluss der instationären turbulenten Fluktuationen auf das zeitlich mittlere Strömungsfeld. Der Term $-\rho\,\overline{v_i' v_j'}$ resultiert dabei aus dem nichtlinearen konvektiven Term der linken Seite der Navier-Stokes-Gleichungen. Bezeichnet wird dieser als Reynolds'scher Schubspannungstensor bzw. als Tensor der turbulenten Schubspannungen $\tau_{t,ij}$.

Aufgrund dieses Zusatzterms sind für den allgemeinen dreidimensionalen Fall neun weitere Größen, die turbulenten Schubspannungen, zu bestimmen. Unter Berücksichtigung der Symmetrie des Reynoldsspannungstensors reduziert sich diese Zahl auf sechs Unbekannte. Folglich werden also neben den sechs Grundgleichungen weitere sechs Gleichungen zur Schließung des Gleichungssystems benötigt. Der Ansatz, für jede Komponente des turbulenten Schubspannungstensors eine eigene Transportgleichung herzuleiten, führt dabei infolge des nichtlinearen Verhaltens der Navier-Stokes-Gleichungen nicht zur Lösung dieses Problems. Denn in den abgeleiteten Korrelationen treten immer weitere Unbekannte auf, für welche zwar wiederum Transportgleichungen aufgestellt werden können, jedoch erhöht sich hierbei stetig die Ordnung der auftretenden unbekannten Tensoren. Insgesamt stehen also immer mehr zu lösende Variablen als Gleichungen zur Verfügung. Dieses Problem wird als Schließungsproblem bezeichnet.
Die Schließung des Gleichungssystems stellt die Hauptaufgabe der Turbulenzmodellierung dar. Hierfür werden empirisch gestützte Modelle herangezogen, welche die unbekannten turbulenten Schubspannungen als Funktion der mittleren Strömungsgrößen beschreiben. Die beiden wichtigsten Klassen von Turbulenzmodellen sind Reynoldsspannungsmodelle und Wirbelviskositätsmodelle. Nach[Hic10] basieren Reynoldsspannungsmodelle auf der Bildung von Modell-Transportgleichungen für die turbulenten Schubspannungen $\tau_{t,ij}$ und ergeben bei der Analyse komplexer Strömungsvorgänge in der Regel bessere Ergebnisse. Aufgrund der sechs zusätzlichen Transportgleichungen sind diese jedoch aufwendiger in der Berechnung sowie numerisch weniger stabil. Wirbelviskositätsmodelle hingegen sind weniger rechenintensiv sowie numerisch stabiler. Darüber hinaus liefern sie für die meisten Anwendungen ebenfalls Resultate zufriedenstellender Genauigkeit.

3.2.3 Wirbelviskositätsansatz

Wie bereits erwähnt, besteht die Hauptaufgabe der Turbulenzmodellierung in der Schließung des Gleichungssystems, welches durch die Anwendung der Reynolds-Mittelung auf die Grundgleichungen der Strömungsmechanik entstanden ist. Eine Möglichkeit hierfür stellt der Wirbelviskositätsansatz nach *Boussinesq* aus dem Jahr 1877 dar. Dieser Ansatz beruht auf der Annahme, dass sich die unbekannten turbulenten Schubspannungen analog zu den laminaren Schubspannungen in Abhängigkeit der zeitlich gemittelten Strömungsgrößen beschreiben lassen. Es gilt:

$$\tau_{t,ij} = -\rho\,\overline{v_i' v_j'} = 2\mu_t S_{ij} - \frac{2}{3}\rho\, k\, \delta_{ij} \tag{3.11}$$

mit

$$\delta_{ij} = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{für } i \neq j \end{cases} \tag{3.12}$$

und

$$k = \frac{1}{2}(\overline{v_i' v_i'}) \tag{3.13}$$

S_{ij} stellt dabei den Scherratentensor nach *Gl. (3.10)* , δ_{ij} das Kronecker-Delta und k die spezifische turbulente kinetische Energie dar. Als Resultat dieses Ansatzes kann die stoffspezifische molekulare Viskosität μ und die turbulente Wirbelviskosität μ_t zu einer

scheinbaren Gesamtviskosität zusammengefasst werden. Zu beachten ist, dass es sich bei μ_t im Gegensatz zu μ um eine reine Modellierungsgröße handelt, welche eine Funktion des lokalen Strömungszustands ist.
Der Vorteil der vereinfachten Betrachtung durch den Wirbelviskositätsansatz ist, dass zur Schließung des Gleichungssystems nicht mehr sechs zusätzliche partielle Differentialgleichungen zu lösen sind, sondern nur noch eine Gleichung zur Bestimmung der skalaren Wirbelviskosität. Diese wiederum kann mit Hilfe von Nullgleichungsmodellen (algebraische Turbulenzmodelle) sowie mit Ein- oder Zweigleichungsmodellen ermittelt werden.
Das Grundproblem der Wirbelviskositätsmodelle, welche allesamt auf der Annahme von *Boussinesq* beruhen, ist eine stark vereinfachte Beschreibung der realen Physik und der damit verbundene Fehler. Nach [Kai07] und [Hic10] sind die Hauptmängel der Boussinesq-Approximation:

Isotropie der Turbulenz: Der Reynolds-Spannungstensor $\tau_{t,ij}$ ist mit dem Scherratentensor S_{ij} durch den skalaren Faktor der turbulenten Viskosität μ_t verknüpft. Anisotrope Einflüsse zum Beispiel aufgrund starker Stromlinienkrümmung sind somit nicht darstellbar.

Kein Transport der Reynoldsspannungen: Der Transport der turbulenten Schubspannungen $\tau_{t,ij}$ wird nicht erfasst.

Diffusiver Gradientenansatz: Der unbekannte nichtlineare konvektive Term auf der rechten Seite, $-\rho\,\overline{v_i{}'v_j{}'}$, wird als diffusiver Term modelliert.

Obwohl Wirbelviskositätsmodelle einige grundsätzliche Schwächen aufweisen, kommen sie in der Praxis häufig zur Anwendung. Denn dadurch, dass zum Beispiel für ein Zweigleichungsmodell nur zwei zusätzliche partielle Differentialgleichungen gelöst werden müssen, kann die Rechenzeit im Vergleich zu den genaueren Reynoldsspannungsmodellen signifikant verkürzt werden. Zudem lassen sich die einzelnen Turbulenzmodelle mit Hilfe empirischer Konstanten speziell auf das zu untersuchende Strömungsproblem anpassen, sodass das resultierende Strömungsfeld in den meisten Fällen mit zufriedenstellender Genauigkeit vorhergesagt werden kann. Zwei der bekanntesten Vertreter der Zweigleichungsmodelle sind das k-ϵ Modell sowie das k-ω Modell.

3.2.4 Wilcox k-ω Turbulenzmodell

Im Folgenden wird das in *TRACE* implementierte k-ω Turbulenzmodell nach *Wilcox* vorgestellt. Für weiterführende Informationen zum Modell bzw. zur Implementierung wird auf [Wil06] sowie [WF06] verwiesen.
Beim k-ω Modell handelt es sich um ein Zweigleichungsmodell. Es sind also zwei partielle Differentialgleichungen zur Berechnung von $\mu_t\ (x_i,\ t)$ zu lösen. Als Modellierungsgrößen wurden von dem russischen Mathematiker *Kolmogorov* 1942 für dieses Modell, wie der Name bereits sagt, k und ω gewählt. Dabei stellt k $[\frac{m^2}{s^2}]$ die spezifische turbulente kinetische Energie dar und ω $[\frac{1}{s}]$ kann entweder als die charakteristische Frequenz der dissipativen Skalen der Wirbelkaskade oder nach [S$^+$06] auch als die spezifische Dissi-

pationsrate pro Einheit turbulenter kinetischer Energie gesehen werden. Mit Hilfe dieser beiden Turbulenz-Modellierungsgrößen lässt sich die turbulente Viskosität μ_t wie folgt bestimmen:

$$\mu_t = \gamma \frac{\rho k}{\omega}, \qquad \gamma = const. = 1 \tag{3.14}$$

Die beiden Transportgleichungen für k und ω lauten dabei:

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho U_j k)}{\partial x_j} = \tau_{ij}\frac{\partial U_i}{\partial x_j} - \rho\beta_k k\omega + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_k\mu_t)\frac{\partial k}{\partial x_j}\right] \tag{3.15}$$

$$\frac{\partial(\rho \omega)}{\partial t} + \frac{\partial(\rho U_j \omega)}{\partial x_j} = \alpha\frac{\omega}{k}\tau_{ij}\frac{\partial U_i}{\partial x_j} - \rho\beta_\omega \omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_\omega\mu_t)\frac{\partial \omega}{\partial x_j}\right] \tag{3.16}$$

Nach [don10] sind die in diesen beiden Gleichungen auftretenden Schließungskoeffizienten für das Basismodell bestimmt zu:

$$\alpha = \frac{5}{9}, \qquad \beta_k = \frac{9}{100}, \qquad \beta_\omega = \frac{3}{40}, \qquad \sigma_k = \frac{1}{2}, \qquad \sigma_\omega = \frac{1}{2}$$

Zusätzlich sind in *TRACE* verschiedene Modifikationen implementiert, welche zum Beispiel über zusätzliche Quellterme in den beiden Transportgleichungen die Vorhersagegüte für bestimmte Anwendungsfälle verbessern.

3.3 Diskretisierung

Die Voraussetzung Problemstellungen aus dem Bereich der Thermo-Fluiddynamik mit Hilfe numerischer Methoden lösen zu können, ist die Diskretisierung der strömungsbeschreibenden partiellen Differentialgleichungen. Damit erfolgt der Übergang von einer kontinuierlichen auf eine diskrete Betrachtungsweise. Das bedeutet, dass der Strömungszustand, beschrieben durch die physikalischen Strömungsgrößen, nicht mehr an allen räumlichen und zeitlichen Punkten ausgewertet wird, sondern nur noch an bestimmten Positionen und Zeitpunkten. Aus der räumlichen und zeitlichen Diskretisierung der strömungsbeschreibenden Differentialgleichungen resultiert schließlich ein lineares oder nichtlineares algebraisches Gleichungssystem. Dieses lässt sich mit Hilfe numerischer Methoden computergestützt lösen.
Für die räumliche Diskretisierung existieren je nach Anwendungsbereich unterschiedliche Ansätze. Bei Strömungssimulationen dominiert dabei die Finite-Volumen-Methode (FVM). Erwähnenswert ist zudem die Finite-Elemente-Methode (FEM), welche hauptsächlich im Bereich der Strukturmechanik zum Einsatz kommt.
Bei der Finiten-Volumen-Methode wird das gesamte 3D-Rechengebiet zunächst in unterschiedlich große Kontrollvolumina aufgeteilt, welche verschiedene Formen (Hexaeder, Tetraeder, Pyramiden, ...) aufweisen können. Im Schwerpunkt jeder Zelle befindet sich der sogenannte Zellknoten. An diesem werden die diskreten Strömungsgrößen berechnet und gespeichert.
Der grundlegende Vorteil der FVM besteht darin, dass es sich um ein konservatives Verfahren handelt. Das bedeutet, die Erhaltungsgleichungen für Masse, Impuls und Energie werden sowohl für jede Zelle als auch global für das gesamte betrachtete Gebiet erfüllt.

Um dies zu gewährleisten, werden die strömungsbeschreibenden Differentialgleichungen zunächst über die einzelnen Kontrollvolumina integriert. Erst im zweiten Schritt erfolgt dann die Diskretisierung, also eine mit Fehlern verbundene Approximation. Für detailliertere Informationen zur räumlichen und zeitlichen Diskretisierung wird auf [Hic10] sowie [S$^+$10a] verwiesen. Informationen zur Implementierung in *TRACE* sind [WF06] zu entnehmen.
Die für die FVM benötigte Netzgenerierung stellt einen wichtigen Teilbereich des Preprocessings dar. Denn das Rechennetz legt die räumliche Lage der diskreten Punkte fest, an welchen die Strömungsgrößen berechnet und gespeichert werden. Das Rechennetz muss dabei in der Lage sein, beliebig komplexe durchströmte Geometrien abbilden zu können, sodass die anschließend folgende Lösung des Strömungsproblems mit der erforderlichen Genauigkeit möglich ist. Des Weiteren beeinflusst die Art sowie die lokale und globale Gestaltung des Netzes sowohl die zur Lösung des Problems zu implementierenden Algorithmen als auch die Qualität und Stabilität der numerischen Simulation. So sind, abhängig vom jeweils verwendeten Solver, verschiedene Anforderungen an die Netzgüte bzw. an die Gestaltung des Netzes zu berücksichtigen.
Grundsätzlich wird zwischen strukturierten und unstrukturierten Gittern unterschie-

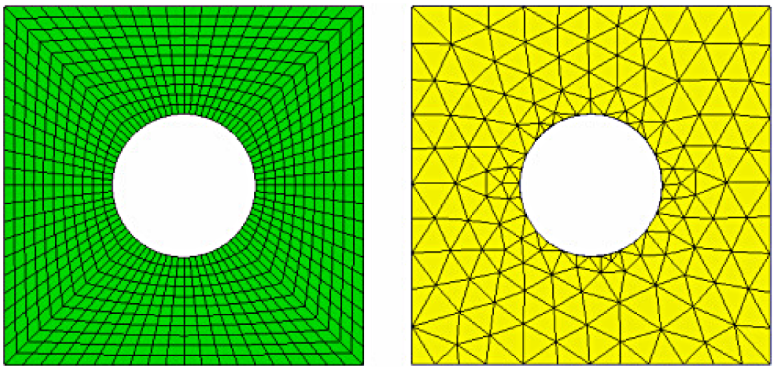


Abb. 3.3: Strukturiertes (links) und unstrukturiertes 2D-Gitter (rechts) [Kla07]

den. *Abb. 3.3* zeigt hierfür beispielhaft zwei 2D-Netze, welche jeweils eine der beiden Varianten repräsentieren.
Die Bilanzvolumina strukturierter Netze sind im Dreidimensionalen Hexaeder. Dadurch hat jede Zelle die gleiche Anzahl an Nachbarn und es ergibt sich somit eine relativ einfache Topologie. Das hat nach [Hic10] zur Folge, dass die Matrizen des numerisch zu lösenden algebraischen Gleichungssystems regulär sind. Deshalb können relativ effiziente Lösungsalgorithmen verwendet werden. Probleme bereitet jedoch die Vernetzung komplexer Geometrien bzw. der damit verbundene Zeitaufwand.
Um den Vernetzungsaufwand zu reduzieren, können unstrukturierte Gitter benutzt werden, welche eine relativ einfach zu implementierende automatische Netzgenerierung erlauben. Da diese jedoch meist aus Tetraedern aufgebaut sind und somit weder über eine globale noch lokale Struktur verfügen, ist die Datenverwaltung in diesem Fall komplizierter. Denn bei unstrukturierten Gittern müssen zusätzlich zu den Daten aller Knoten und Zellen auch Informationen über die jeweiligen Nachbarzellen abgespeichert werden. Das wiederum führt verglichen mit strukturierten Gittern zu einem erhöhten Speicherbedarf und zu längeren Rechenzeiten. Zudem ist die numerische Diffusion bei unstrukturierten

Netzen größer. Das bewirkt, dass für die gleiche Ergebnisqualität eine feinere Gitterauflösung, ergo mehr Zellen, benötigt werden. Nach [Hic10] sind jedoch die Vorteile der unstrukturierten Gitter einerseits die große Flexibilität hinsichtlich der lokalen Netzverfeinerung und andererseits, wie bereits erwähnt, die gute Automatisierbarkeit der Netzgenerierung.
Darüber hinaus existieren auch hybride Gitter, welche eine Mischung von strukturierter und unstrukturierter Vernetzung darstellen. So wird versucht die Vorteile beider Netztopologien zu verknüpfen. Es ist zudem üblich, das gesamte Rechennetz in einzelne Blöcke zu zerlegen, um eine optimale Vernetzung in den verschiedenen Teilbereichen des Strömungsfelds zu erreichen. In den einzelnen Blöcken selbst kann die Vernetzung wiederum strukturiert oder unstrukturiert erfolgen. Werden hierbei strukturierte Netze verwendet, wird das Gitter insgesamt als blockstrukturiertes Netz bezeichnet.

3.4 Lösung des resultierenden Gleichungssystems

Durch die räumliche und zeitliche Diskretisierung ergibt sich ein algebraisches Gleichungssystem, welches mit Hilfe numerischer Methoden gelöst werden kann. Dieses besitzt folgende Form:

$$\underline{\underline{A}} \cdot \underline{\Phi} = \underline{Q} \tag{3.17}$$

Da es sich bei Strömungssimulationen um Anfangsrandwertprobleme handelt, müssen zur Lösung des Gleichungssystems zusätzlich die Anfangsbedingungen zum Zeitschritt t_0 sowie die Randbedingungen an den Grenzen des Rechengebiets definiert und mit entsprechenden Werten belegt werden. In *Gl. (3.17)* stellt $\underline{\underline{A}}$ die Koeffizientenmatrix, $\underline{Q}$ den Quellterm, welcher unter anderem einen Teil der Randbedingungen enthält, und $\underline{\Phi}$ den zu bestimmenden Lösungsvektor dar. Für ein zweidimensionales Gitter unter Benutzung der Kompassnotation (siehe [Hic10]) verfügt dieses Gleichungssystem über folgende Struktur:

$$A_S\phi_S + A_W\phi_W + A_P\phi_P + A_E\phi_E + A_N\phi_N = Q \tag{3.18}$$

Die Koeffizienten A_i können dabei nach [S$^+$10a], je nach betrachteter Differentialgleichung, diffusive, konvektive und zeitliche Anteile enthalten. Charakteristisch für die Koeffizientenmatrix ist ihre relativ schwache Besetzung und ihre Bandstruktur. Beide Eigenschaften werden zur Implementierung effizienter Lösungsalgorithmen ausgenutzt. Meist werden dabei iterative Algorithmen benutzt, da diese deutlich effizienter und damit auch schneller als direkte Verfahren sind. Grundsätzlich wäre das Gleichungssystem jedoch auch direkt lösbar, zum Beispiel mit dem Gaußschen Eliminationsverfahren. Für weitere Informationen zu dieser Thematik wird auf [S$^+$10a] und [Hic10] verwiesen.
Das Grundproblem iterativer Lösungsverfahren besteht darin, ein sinnvolles Abbruchkriterium zu definieren. Der Optimalfall wäre es hierfür einen bestimmten Betrag des numerischen Fehlers vorzugeben. Jedoch kann während der Rechnung der Gesamtfehler zu einem bestimmten Iterationsschritt nicht bestimmt werden, da ja die exakte Lösung a priori nicht bekannt ist. Zur Abschätzung des numerischen Fehlers kann das Residuum $\underline{R}$ herangezogen werden. Für dieses gilt nach [S$^+$10a]:

$$\underline{\underline{A}} \cdot \underline{\Phi}^n = \underline{Q} - \underline{R}^{n-1} \quad \Longrightarrow \quad \underline{R}^{n-1} = \underline{Q} - \underline{\underline{A}} \cdot \underline{\Phi}^n \tag{3.19}$$

$\underline{R}^{n-1}$ ist dabei das Residuum des vorangegangenen Iterationsschritts n-1 und repräsentiert den numerischen Fehler, welcher mit der Lösung zum aktuellen Iterationsschritt $\underline{\Phi}^n$ gemacht wird.
Als Abbruchkriterium eines iterativen Solvers ist es sinnvoll einen skalaren Residuums-Grenzwert zu bestimmen, welcher erreicht werden soll. Ein skalarer Wert lässt sich aus dem Residuumsvektor, welcher die Residuen aller Zellen für einen Iterationsschritt enthält, auf verschiedene Arten gewinnen. Meist wird entweder ein mittleres Residuum nach der L_p-Vektornorm gebildet oder das maximal auftretende Residuum herangezogen. Nach [S+10a] ist die Vorgabe eines Residuums-Grenzwerts als Abbruchkriterium des iterativen Solver sinnvoll, denn es gilt:

$$\lim_{n\to\infty} \underline{R}^n = 0 \quad \Longrightarrow \quad \lim_{n\to\infty} \underline{\Phi}^n = \underline{\Phi}^{exakt} \tag{3.20}$$

Das bedeutet, dass ein iteratives Lösungsverfahren die exakte Lösung erreicht, wenn alle Einträge des Residuumsvektors zu Null werden. Als Abbruchkriterium werden dem Lösungsalgorithmus für das Residuum somit ein ausreichend kleiner, aber endlicher Wert vorgegeben.
Für stationäre Problemstellungen wird häufig auch die Änderung einer oder mehrerer für das Strömungsproblem charakteristischen Größen über einem bestimmten Intervall von Iterationsschritten analysiert. Ist die Veränderung der betrachteten Größen vernachlässigbar klein, ist der stationäre Endzustand erreicht und die Rechnung wird beendet.

Kapitel 4

TRACE-Simulationsprozess

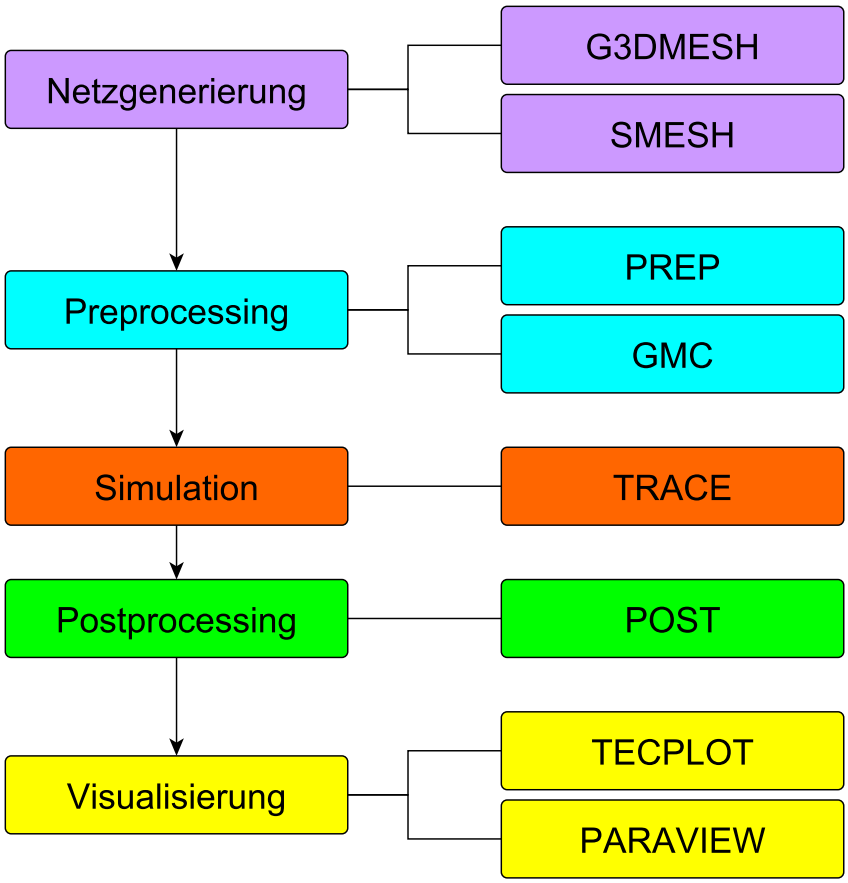


Abb. 4.1: Flussdiagramm des *TRACE*-Simulationsprozesses

Der *TRACE*-Simulationsprozess (siehe [don11a]) setzt sich aus mehreren Einzelschritten beziehungsweise aus mehreren Programmen zusammen (siehe *Abb. 4.1*).
Am Anfang steht die Netzgenerierung. Hierfür wird bei blockstrukturierten Netzen das Programm *G3DMESH* ([Web10a] sowie [Web10b]) verwendet. Für unstrukturierte Gitter hingegen wird auf *SMESH* ([Hof10]) zurückgegriffen. Anschließend folgt das Preprocessing mit *PREP* ([don11c]) und *GMC* ([don11b]). Mit Hilfe von *PREP* erfolgt die Initialisierung aeroelastischer Simulationen und die Definition von zusätzlichen Auswer-

teflächen neben den vorhandenen Interfaces. In *GMC* wird anschließend die zu analysierende Konfiguration aufgebaut und für die Simulation mit *TRACE* ([WF06]) vorbereitet. Hierbei werden unter anderem die Randbedingungen definiert, die Initialisierung der Strömungslösung durchgeführt und die Solver-Einstellungen wie zum Beispiel die Behandlung der Turbulenz vorgegeben.
Dann erfolgt die Simulation der zu analysierenden Konfiguration mit *TRACE*. Diese endet, wenn entweder das definierte Konvergenzkriterium oder die maximale Anzahl an Iterationsschritten erreicht ist.
Schließlich folgt das Postprocessing, also die Aufbereitung der Daten, mit *POST*. Hierbei werden auf Basis des von *TRACE* gelieferten CGNS-Datensatzes unter anderem zusätzliche Strömungsgrößen berechnet und verschiedene Mittelungen durchgeführt. Diese umfassen eine Umfangsmittelung sowie eine zeitliche Mittelung für instationäre Fälle. Zusammengefasst besteht die Aufgabe von *POST* also darin, aussagekräftige 3D-, 2D, 1D- und 0D-Daten zu erzeugen, welche eine möglichst schnelle und genaue Auswertung der untersuchten Konfiguration ermöglichen. Die Ergebnisse werden als CGNS-, PLT-, oder ASCII-Daten geschrieben. Diese Resultate wiederum lassen sich mittels *TECPLOT 360* oder *ParaView* grafisch darstellen und auswerten.

Kapitel 5

Vorstellung der Task-Implementierungen

5.1 Vorstellung des Programms *POST 7*

Im Folgenden wird das Programm *POST 7* vorgestellt, für welches die im Rahmen der Arbeit entstandenen Tasks der globalen Turbomaschinen-Auswertung implementiert worden sind. Beim Postprocessing-Tool *POST 7* handelt es sich im Gegensatz zur aktuellen Version von *TRACE* um eine komplette Neuentwicklung, welche hauptsächlich auf der Programmiersprache C basiert.
POST greift zudem auf die Bibliotheken und diverse Routinen von *TRACE* zu, welches somit eine der beiden Grundlagen darstellt. Das zweite Programm, welches *POST* benötigt, ist *INTERSEC*, welches selbst wiederum verschiedene Funktionen von *VTK* der Version 5.4.3 oder höher verwendet. Die Abkürzung *VTK* steht dabei für „Visualization Toolkit“ und stellt eine Open-Source-C++-Klassenbibliothek dar, welche hauptsächlich für grafische Anwendungen eingesetzt wird. Für weitere Informationen wird auf [S+10b] verwiesen. Im Rahmen von *INTERSEC* besteht die Hauptaufgabe der *VTK*-Bibliothek in der Generierung beliebiger Schnittflächen und Schnittlinien. *INTERSEC* stellt dabei die Schnittstelle zwischen *POST* und der eigentlichen, auf *VTK* beruhenden Verschneidung dar. Das Tool kümmert sich somit vor allem um die Konvertierung von *POST*-Datenstrukturen auf *VTK*-Datenstrukturen und umgekehrt.
Nun wird das *POST* zugrunde liegende Task-Konzept vorgestellt. Die Ausführung einer vom Benutzer gestellten Aufgabe wird in mehrere Unterschritte zerlegt, wobei jeder dieser Schritte von einem darauf spezialisierten Task abgearbeitet wird. Der Vorteil der Auftrennung in abgeschlossene Unteraufgaben besteht darin, dass die Implementierung der einzelnen Tasks relativ allgemein erfolgen kann und sich diese somit nahezu beliebig miteinander kombinieren lassen. Dies führt zu einer deutlichen Steigerung der Flexibilität des Programms. Beispielsweise werden mit Hilfe der implementierten Tasks zur Turbomaschinen-Auswertung momentan nur stationäre Lösungen der Strömungsfelder bestimmter Konfigurationen analysiert, durch die Flexibilität der einzelnen Tasks jedoch ist es mit relativ geringem Änderungsaufwand möglich, diese auch für eine Untersuchung instationärer Problemstellungen einzusetzen.

Ein weiteres Basiskonzept bildet die Tatsache, dass Daten bzw. Zwischenresultate aus dem Datenset entfernt werden, sobald diese für den weiteren Prozess nicht mehr benötigt werden. Dies sorgt für eine Beschleunigung des Programmablaufs sowie eine Reduktion des Speicherbedarfs.
Zu Beginn eines *POST*-Aufrufs steht der *input*-Task, dessen Aufgabe das Einlesen der zu bearbeitenden Daten ist. Im Anschluss an diesen folgt eine beliebige Zahl unterschiedlicher Tasks, welche die vorhandenen Daten manipulieren, neue Daten erzeugen sowie nicht mehr benötigte löschen. Diese Tasks umfassen unter anderem die Ableitung zusätzlicher Strömungsgrößen, die räumliche und zeitliche Mittelung sowie das Duplizieren simulierter Passagen. Am Ende schließlich schreibt der *output*-Task die fertige Ergebnis-Datei. In *POST* wird dabei zwischen vier verschiedene Arten von Tasks unterschieden :

- *peasant*-Tasks
- *supreme*-Tasks
- *combined*-Tasks
- *global*-Tasks

Die einzelnen Task-Arten werden nun etwas genauer vorgestellt. Die *peasant*-Tasks stellen einen großen Vorteil von *POST* dar, weil über diese die Erweiterbarkeit des Programms durch den Benutzer möglich ist. Denn, da diese extra kompiliert werden, können auch Anwender, welche nicht über den Quellcode verfügen, dem Programm eigene *peasant*-Tasks hinzufügen, welche beispielsweise zusätzlich gewünschte Variablen ableiten. Außerdem kann vom Anwender vorgegeben werden, auf welchen Daten der *peasant*-Task arbeiten soll, zum Beispiel nur auf 2D-Blöcken. Diese Einschränkungen werden dann vom *Scheduler* bzw. der Task-Ablaufsteuerung abgefragt und dem jeweiligen Task schließlich die entsprechenden Daten Block für Block zur Bearbeitung übergeben. Zusätzlich können *peasant*-Tasks auf weitere Informationen wie zum Beispiel die Drehzahl einer Schaufelreihe oder Daten zu deren Beschaufelung zurückgreifen. Der Nachteil von *peasant*-Tasks ist jedoch, dass diese aufgrund der blockweisen Datenverarbeitung der jeweiligen Aufgabe für komplexere, blockübergreifende Operationen nicht verwendet werden können.
Für diese Operationen sind *supreme*-Tasks zu benutzen, welche jedoch nicht durch den Benutzer erweiterbar sind. Zudem besitzen diese deutlich weniger Einschränkungen als *peasant*-Tasks. Nahezu alle der im Rahmen der Semesterarbeit implementierten Tasks entsprechen dabei diesem Typ.
Schließlich existieren noch die *combined*- und *global*-Tasks. Erstere stellen, wie der Name bereits sagt, eine Kombination verschiedener *supreme*- und *peasant*-Tasks dar und dienen hauptsächlich zur Verkürzung des Kommandozeilenaufrufs. Die *global*-Tasks wiederum sind Tasks, die auf dem kompletten Datenset arbeiten, jedoch nicht mit anderen Tasks kombinierbar sind und somit auch nicht vom *Scheduler* gesteuert werden. Diese Tasks beinhalten das Einlesen der Daten, deren Verarbeitung sowie die Ausgabe. Im Normalfall jedoch sind die Einzelaufgaben entsprechend des Konzepts von *POST* auf verschiedene Tasks verteilt.
Abschließend wird auf die Funktionsweise des *Schedulers* eingegangen. Dessen Aufgabe

besteht in der Steuerung des Taskablaufs sowie in der Speicherverwaltung. Zunächst wird im Rahmen eines sogenannten „*dry runs*“ überprüft, ob die über die Kommandozeile eingegebenen Befehle jeweils einem Task zugeordnet werden können und ob alle obligatorischen Optionen des jeweiligen Tasks vom Anwender vorgegeben worden sind. Es findet somit zunächst eine Überprüfung der Input-Variablen statt. Während des Programmablaufs besteht die Aufgabe des *Schedulers* dann darin, vor jedem Task zu überprüfen, ob die zu verarbeitenden Daten den Task-Vorgaben bzw. den Task-Einschränkungen entsprechen, welche jeweils vom Task-Programmierer spezifiziert worden sind. Über diese Vorgaben sind auch diejenigen Daten definiert, welche vom Task verarbeitet werden können, beispielsweise alle 2D-Blöcke. Diese Blöcke bzw. Daten werden dann dem Task über ein Interface vom *Scheduler* bereitgestellt.
Des Weiteren kann über die gesetzten Optionen dem *Scheduler* mitgeteilt werden, ob der Task beispielsweise ein Geometrie-Setup des Netzes oder eine MPI-Kommunikation benötigt. Der *Scheduler* ruft dann zusätzliche Routinen auf, welche das bewerkstelligen. Werden zum Beispiel innerhalb eines Tasks neue Blöcke erzeugt, so ist eine anschließende Kommunikation und Synchronisation der neu generierten Daten nötig, was der *Scheduler* über die jeweils gesetzten Task-Attribute erkennt. Dasselbe gilt, wenn ein Geometrie-Setup des Netzes benötigt wird.

5.2 Taskablauf der Turbomaschinen-Auswertung

Im Rahmen des Gesamtkapitels werden die Implementierungen der einzelnen Tasks der Turbomaschinen-Analyse vorgestellt. Da die meisten Tasks direkt oder indirekt voneinander abhängen, wird in diesem Abschnitt zunächst auf den Taskablauf der Turbomaschinen-Auswertung nach *Abb. 5.1* eingegangen.
Die Turbomaschinen-Analyse lässt sich dabei in vier Teilbereiche untergliedern:

- Erstellung eines temporären S2M-Netzes, auf welchem die Verschneidung der 3D-Konfiguration basiert (gelb)
- Verschneidung der 3D-Konfiguration zur Erzeugung von Analyseflächen (blau)
- Mittelung für die Analyseflächen und Boundary-Flächen in Umfangs- und Normalrichtung (grün)
- Analyse der Turbomaschine (orange)

Im ersten Schritt ist ein temporäres S2M-Netz bereitzustellen, auf welchem die anschließende Verschneidung der 3D-Konfiguration basiert. Dieses Netz kann auf zwei unterschiedliche Arten beschafft werden. Entweder wird die Geometrie eines bereits vorhandenes S2M-Netzes geladen (*readS2m*) oder es erfolgt eine automatische Generierung des benötigten Gitters. Für letzteren Fall sind vom Benutzer die entsprechenden Nabenwand- und Gehäusewand-Familien zu spezifizieren, aus welchen mit Hilfe mehrerer θ = konstant Schnitte die Kontur beschreibenden Schnittlinien gewonnen werden (*generateThetaCuts*). Auf deren Basis wird anschließend der Schattenriss der Konfiguration bzw. zwei vom Inlet zum Outlet durchgehende Linien für Nabe und Gehäuse erzeugt (*generateExtremalOutline*). Für diesen Task sind jedoch vorher über den *collectGeome-*

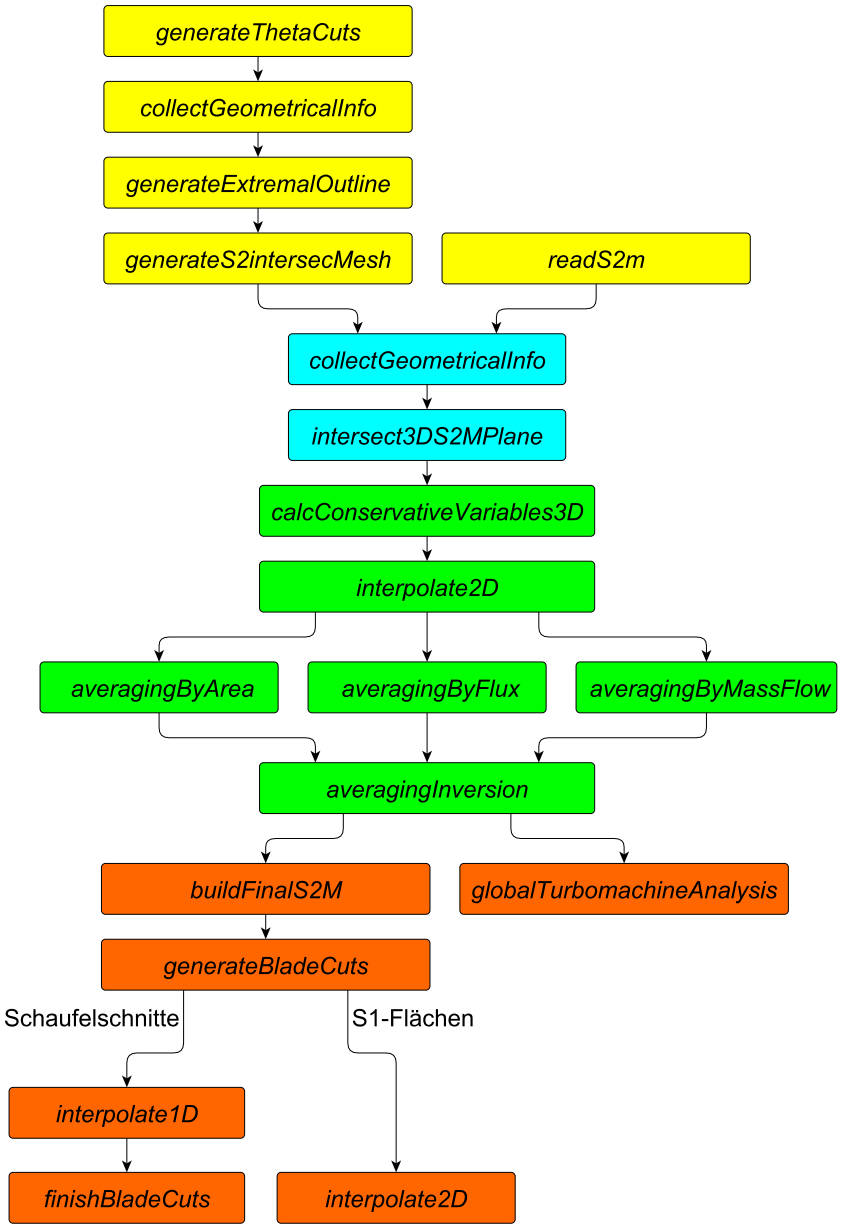


Abb. 5.1: Flussdiagramm des Taskablaufs der Turbomaschinen-Auswertung

tricalInfo-Task zusätzlich benötigte Geometrie-Informationen zu ermitteln. Ausgehend von der Naben- und Gehäuselinie wird der von diesen umschlossene Bereich mit einem strukturierten Gitter vernetzt (*generateS2intersecMesh*-Task). Das resultierende Gitter stellt schließlich das fertige, automatisch generierte S2M-Netz dar.
Zur Verschneidung der 3D-Konfiguration muss für das S2M-Netz zunächst eine ξ- und η-Verteilung, auf welcher die geometrische Lage der Schnitt- und Bandpositionen basiert, definiert werden. Ebenso ist auch für diesen Task der *collectGeometricalInfo*-Task aufzurufen. Durch die Verschneidung des *intersect3DS2MPlane*-Tasks entstehen dann beliebig viele Analyseflächen, auf welche mittels des *interpolate2D*-Tasks die Strömungslösung des 3D-Felds interpoliert wird. Im selben Task bekommen die Boundary-Blöcke die benötigte Strömungslösung übertragen. Bevor die Interpolation jedoch durchgeführt

wird, werden zunächst die konservativen Variablen, auf welchen die Mittelung beruht, für das 3D-Feld berechnet (*calcConservativeVariables3D*) und somit schließlich auf die 2D-Flächen interpoliert.
Bei der implementierten Variante des *interpolate2D*-Tasks handelt es sich um eine lineare Interpolation. Um die Werte des 3D-Felds auf die Analyseflächen zu interpolieren, werden dabei die Gradienten über die Methode der kleinsten Quadrate bestimmt. Damit ergibt sich ein resultierender Interpolationsfehler 2.Ordnung. Um die Strömungsgrößen auf 2D-Zellen von Boundary-Blöcken zu erhalten, wird hingegen der Mittelwert aus den beiden links und rechts der Fläche liegenden Werte des 3D-Felds benutzt. Anzumerken ist, dass der *interpolate2D*-Task nicht im Rahmen der Semesterarbeit programmiert wurde.
Daraufhin folgen die Mittelungs-Kernroutinen *averagingByArea*, *averagingByMassFlow* sowie *averagingByFlux*. Deren Aufgabe ist, abhängig vom Mittelungstyp, eine Summation der gewichteten Zellbeiträge für alle Bänder eines Blocks sowie für dessen Gesamtfläche. Diese Routinen operieren dabei noch auf den lokalen Blöcken eines Prozesses. Mit Hilfe der vom *averagingInversion*-Task bereitgestellten Routinen werden dann die Beiträge der einzelnen Blöcke an ihre jeweilige Familie kommuniziert und aufsummiert. Anschließend werden darauf basierend die resultierenden integralen Strömungsgrößen sowohl für die Bänder als auch für die Gesamtfläche berechnet.
Nach Abschluss der Mittelung kann entweder die globale Turbomaschinen-Analyse (*globalTurbomachineAnalysis*) durchgeführt werden oder das resultierende S2M-Netz aufgebaut werden (*buildFinalS2M*). Erstere berechnet dabei charakteristische 0D-Kenngrößen und 1D-Verteilungen für Interfaces, Schaufelreihen, Stufen und die Komponente. Wird der *buildFinalS2M*-Task ausgeführt, wird die S2M-Fläche, welche auf einer strukturierten Netztopologie basiert, unter Verwendung der vom Benutzer spezifizierten Lösung, gemittelt nach dem vorgegebenen Mittelungstyp, generiert. Die Grundlage bilden die umfangsgemittelten Bandwerte der Analyseflächen. Auf Basis der S2M-Fläche können Schaufelschnitte und S1-Stromflächen (*generateBladeCuts*) für einen konstanten relativen Massenstrom oder eine konstante relative Kanalhöhe definiert werden. Für S1-Stromflächen ist schließlich wiederum der *interpolate2D*-Task aufzurufen, wohingegen für die Schaufelschnitte *interpolate1D* verwendet wird, um die Lösung auf den Schnitten zu erhalten. Zusätzlich ist für Letztere noch der *finishBladeCuts*-Task aufzurufen, welcher entlang des Schaufelprofils den Druckbeiwert c_p sowie die isentrope Mach-Zahl Ma_{is} bestimmt.

5.3 Bereitstellung des temporären S2M-Schnittnetzes

Der erste Schritt des Gesamtprozesses besteht darin, ein temporäres S2M-Netz zur Verschneidung der 3D-Konfiguration bereitzustellen. Dieses wird im Allgemeinen automatisch generiert. Falls der Anwender bereits über ein S2M-Netz verfügt, welches die entsprechenden Anforderungen erfüllt, kann dieses mit Hilfe des *readS2m*-Tasks eingelesen werden. Für die automatische Netzgenerierung sind dagegen folgende vier Tasks aufzurufen:

- *generateThetaCuts*

- *collectGeometricalInfo*
- *generateExtremalOutline*
- *generateS2intersecMesh*

Die bei der Netzgenerierung verwendeten Begriffe bzw. Elemente sind *Abb. 5.2* zu entnehmen. Es gilt dabei:

- Vertex ≙ Knoten
- Edge ≙ Kante
- Face ≙ Fläche
- Cell ≙ Zelle

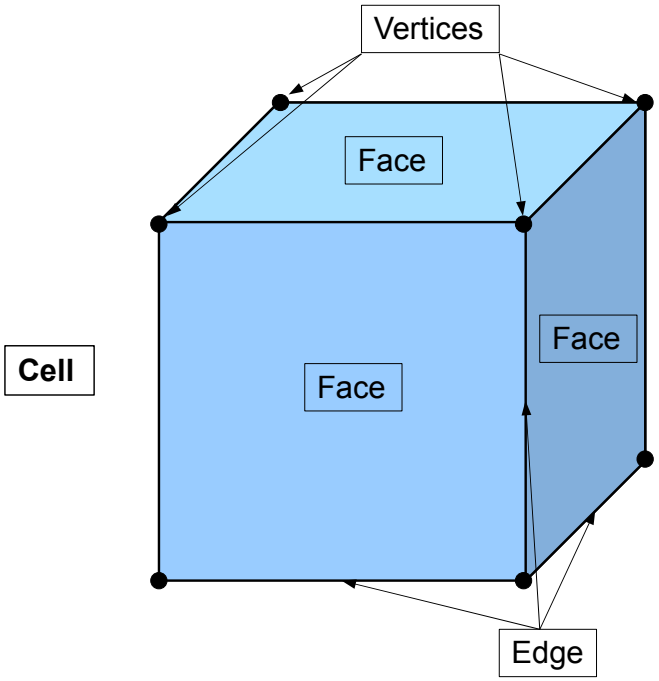


Abb. 5.2: Bezeichnung der Netzelemente am Beispiel einer 3D-Zelle

5.3.1 Erstellung von θ = konstant Linienschnittsegmenten - *generateThetaCuts*

Zu Beginn der Generierung des für die Verschneidung der 3D-Konfiguration benötigten temporären S2M-Netzes steht der *generateThetaCuts*-Task (siehe *Tab. 5.1*). Dessen Aufgabe besteht darin, die Kontur der zu analysierenden Konfiguration in der *xr*-Ebene zu erfassen. Hierfür wird eine Linienverschneidung der Naben- und Gehäuse-Berandungsflächen mit einer bestimmten Anzahl von θ = konstant Schnitten durchgeführt. In den nachfolgenden Tasks werden dann die resultierenden Schnittsegmente zunächst zu einer durchgehenden Gehäuse- bzw. Nabenlinie zusammengefügt und schließlich das von diesen umschlossene Gebiet vernetzt.
Beim *generateThetaCuts*-Task erfolgt zunächst die Abfrage der Benutzereingaben. Über

Name	generateThetaCuts		
Aufruf	-gtc		
Optionen	-nCuts	Anzahl an Schnitten	optional
		18	default
	-hub	Naben-Familiennamen	obligatorisch
	-tip	Gehäuse-Familiennamen	obligatorisch

Tabelle 5.1: Übersicht über Aufruf und Optionen des *generateThetaCuts*-Tasks

die Option „-nCuts“ kann die Anzahl an θ-Schnitten vorgegeben werden. Wird seitens des Anwenders hierzu keine Angabe gemacht, wird der Standardwert von 18 Schnitten verwendet, was sich für die untersuchten, meist umfangssymetrischen Konfigurationen in der Mehrzahl der Fälle als ausreichend erwiesen hat. Über die Option „-hub“ bzw. „-tip“ werden vom Benutzer die Namen der zu verschneidenden Boundary-Familien, jeweils durch ein Leerzeichen getrennt, angegeben. Dabei ist die getrennte Vorgabe von Naben- und Gehäuse-Familien zu beachten. Anhand der beiden Namenslisten wird dann überprüft, ob die spezifizierten Boundary-Familien im Datenset vorhanden sind. Anzumerken ist hierbei, dass es sich bei den vorzugebenden Familien nicht zwangsläufig um Nabenwand- oder Gehäusewand-Familien handeln muss; ebenso kann beispielsweise das Interface zu einer Kavität angegeben werden, falls diese zur Erstellung des S2M-Netzes nicht berücksichtigt werden soll. Es ist jedoch darauf zu achten, möglichst keine Lücken zwischen den vorgegebenen Familien zu lassen, da der anschließende *generateExtremalOutline*-Task (*Kap. 5.3.2*) auf Basis der Vertices der Schnittlinien eine durchgehende Linie vom Inlet zum Outlet erstellt. Falls jedoch Lücken auftreten, existieren in diesem Bereich keine Schnittlinien und keine Vertices. Folglich kann die Kontur der Konfiguration nur unzureichend erfasst werden.
Bezüglich des Konzepts dieses Tasks ist darauf hinzuweisen, dass anfangs die Implementierung einer Flächenverschneidung geplant war, also eine Verschneidung von 3D-Blöcken. Für die resultierenden unstrukturierten 2D-Schnittflächen hätten dann alle inneren Netzknoten entfernt werden müssen, um die Naben- und Gehäuse-Konturlinie zu erhalten. Diese Vorgenhensweise weist jedoch gegenüber der umgesetzten Linienverschneidung diverse Nachteile auf. Zunächst einmal ist nicht exakt definiert, welche Linie die Naben- und welche die Gehäusekontur darstellt. Die Ermittlung der beiden Linien ist zudem zeitlich sehr aufwendig, da die Verschneidung von 3D-Blöcken bzw. deren Konvertierung von *POST*- auf *VTK*-Strukturen numerisch relativ teuer ist. Folglich wurde eine Verschneidung von 2D-Blöcken implementiert.
Im Folgenden wird die eigentliche Kernroutine, welche für Naben- und Gehäusefamilien getrennt aufgerufen wird, vorgestellt. Im ersten Schritt erfolgt dabei die Abfrage der Periodizität der zu untersuchenden Konfiguration. Ebenso wird die Definition der Rotationsachse überprüft. Analog zu *TRACE* muss diese identisch zur x-Achse sein. Die Periodizität entscheidet dann, welche Art Schnittnetz zur Verschneidung der spezifizierten 2D-Blöcke aufgebaut wird. Für rotationsperiodische Konfigurationen werden kreisförmige und für translationsperiodische Konfigurationen viereckige Schnittnetze definiert.

Zur Definition des Schnittnetzes werden für beide Varianten zunächst die Maximalabmessungen der Gruppe der zu verschneidenden 2D-Blöcke ermittelt, welche lokal auf dem jeweiligen Prozess vorhanden sind. Zu verschneidende Blöcke sind dabei diejenigen, welche zu einer der vom Benutzer angegebenen Nabenwand- oder Gehäusewand-Panelfamilien gehören. Bei den ermittelten Abmessungen handelt es sich je nach Periodizität um folgende Größen:

- rotationsperiodisch: r_{max}, θ_{min}, θ_{max}
- translationsperiodisch: y_{min}, y_{max}, z_{min}, z_{max}

Für den Fall einer rotationsperiodischen Konfiguration wird dann mit Hilfe der bestimmten Maximalabmessungen entschieden, ob das Schnittnetz als Vollkreis oder als Kreissegment aufgebaut wird. Dabei gilt:

$$\Delta\theta = \theta_{max} - \theta_{min} < 180^\circ \qquad \Longrightarrow \qquad \text{Kreissegment-Schnittnetz}$$

$$\Delta\theta = \theta_{max} - \theta_{min} \geq 180^\circ \qquad \Longrightarrow \qquad \text{Vollkreis-Schnittnetz}$$

Falls $\Delta\theta$ also kleiner als 180° ist, wird ein zweidimensionales strukturiertes Schnittnetz in zylindrischen Koordinaten, definiert in der $r\theta$-Ebene mit $x = 0$, verwendet. Bei den Zellen um die Kreismitte handelt es sich dabei um entartete Viereckszellen, welche die Form eines Dreiecks aufweisen. Zwei der vier Zellpunkte sind also jeweils identisch zum Kreismittelpunkt. Der Radius des Kreisbogens ist gegenüber den ermittelten Abmessungen der lokalen Blockgruppe um 5% vergrößert. Andernfalls kann aufgrund numerischer Ungenauigkeiten nicht gewährleistet werden, dass die Schnittlinien wie beabsichtigt generiert werden. In Radialrichtung verfügt das Schnittnetz über 10 Netzknoten, wohingegen die Punktzahl in Umfangsrichtung der vorgebbaren Anzahl an Schnitten entspricht, welche äquidistant verteilt werden. Somit stellen die θ = konstant Netzlinien gleichzeitig die Schnittpositionen dar, wobei die Schnittrichtung der x-Achse entspricht. Der Vorteil der Verwendung eines Kreissegments anstelle eines Vollkreises als Schnittnetz liegt darin, dass nur dort Schnitte gesetzt werden, wo sich die zu verschneidenden 2D-Blöcke befinden. Damit ergibt sich bei Vorgabe einer festen Anzahl an Schnitten eine Verbessserung der Auflösung bzw. bei gleichbleibender Ergebnisqualität kann die Schnittzahl und damit der Rechenaufwand verringert werden.
Ist $\Delta\theta$ größer oder gleich 180°, ist es effizienter ein Vollkreis-Schnittnetz zu verwenden. Dieses ist ebenfalls in Zylinderkoordinaten definiert und wird auf die gleiche Weise aufgebaut wie das Kreissegment-Netz. Der Unterschied besteht darin, dass bei Verwendung eines Vollkreises die Netztopologie unstrukturiert ist. Der grundlegende Vorteil stellt dabei die Halbierung der benötigten Zahl an Schnittebenen dar. Denn im Gegensatz zur vorigen Variante, werden die Netzlinien θ = konstant sowie $\theta + 180^\circ$ = konstant jeweils zu einer Schnittlinie zusammengefasst. Die Schnittlinien entsprechen somit jeweils einer Kreisdiagonale. Die Verteilung der Schnitte erfolgt wiederum äquidistant.
Anzumerken ist, dass es aufgrund eines nicht exakt ermittelbaren Fehlers in *INTERSEC* oder *VTK* nicht möglich ist, kreisförmige Schnittnetze zu verwenden. Denn, wie *Abb. 5.3* beispielhaft für den Testfall der LISA-Turbine zeigt, können für diese die resultierenden Schnittlinien nicht korrekt ermittelt werden. Folglich wird für alle Fälle, sowohl rotationsperiodische als auch translationsperiodische, ein viereckiges Schnittnetz

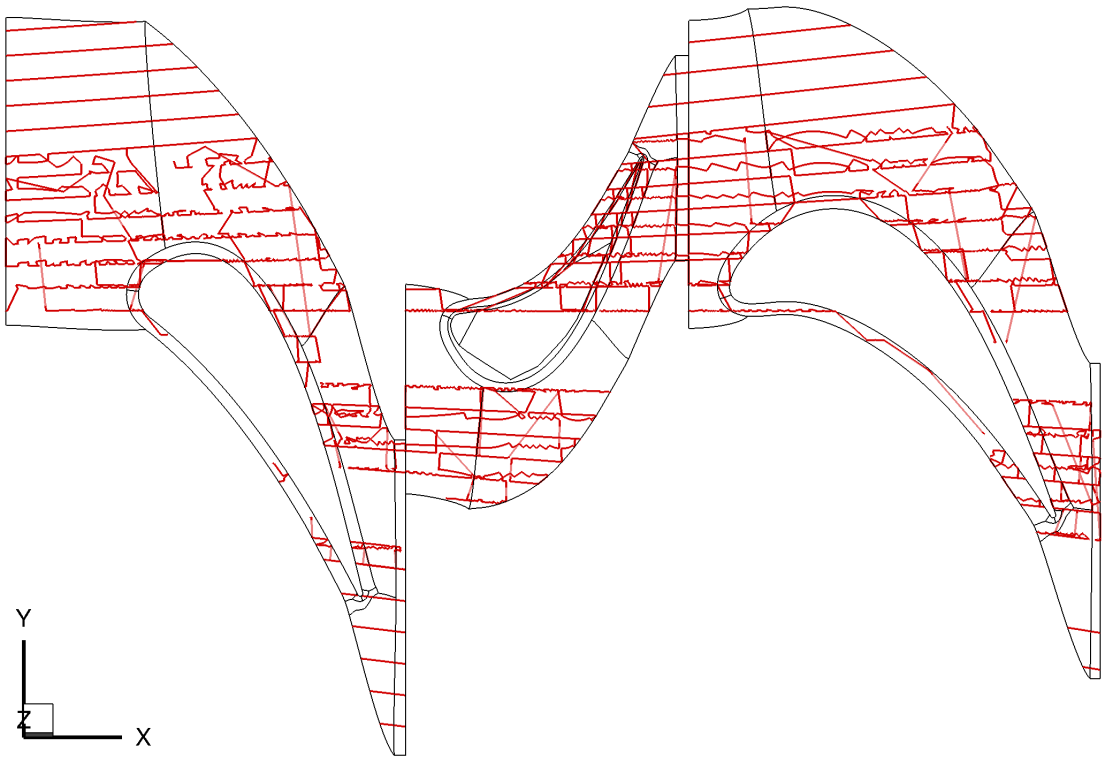


Abb. 5.3: Schnittlinien für ein Kreissegment-Schnittnetzes (Testfall LISA)

aufgebaut, welches ursprünglich translationsperiodischen Konfigurationen vorbehalten war.

Beim Vierecks-Schnittnetz handelt es sich um ein strukturiertes Gitter definiert in kartesischen Koordinaten in der yz-Ebene an der Position $x = 0$. Die ermittelten Maximalabmessungen der zu verschneidenden Blockgruppe werden dabei sowohl in y- als auch in z-Richtung um jeweils 5% in positiver und negativer Richtung verlängert, um, wie bereits erwähnt, eindeutig definierte Schnittlinien zu erhalten. In z-Richtung verfügt das Netz, ebenso wie die anderen Schnittnetze, über 10 Knoten und in y-Richtung über die vorgegebene Schnittanzahl. Die Schnittlinien entsprechen dabei den y = konstant Netzlinien und werden wiederum äquidistant verteilt. Bei der Erstellung des Schnittnetzes wird dabei angenommen, dass die Spannweitenrichtung der z-Koordinatenachse entspricht.

Nach Definition des Schnittnetzes sowie weiterer sekundärer Steuerungsparameter wird die Linienverschneidung der spezifizierten Blöcke in Axialrichtung durchgeführt (siehe *Abb. 5.4*). Diese basiert auf *INTERSEC*, welches wiederum auf die Funktionalitäten von *VTK* zurückgreift. Die Verschneideoperation erfolgt dabei blockweise, das heißt jeder zu verschneidende Block wird von *POST*- auf *VTK*-Datenstrukturen konvertiert und anschließend werden alle definierten Schnitte für diesen Block durchgeführt. Diese Vorgehensweise wurde gewählt, um eine Blockparallelisierung der Verschneidung zu ermöglichen.

Zunächst werden von *VTK* unstrukturierte Schnittlinien in kartesischen Koordinaten geliefert, welche dann in strukturierte 1D-Netze konvertiert werden. Um zu gewährleisten, dass die Indexreihenfolge bzw. die Orientierung der strukturierten Linien korrekt

ist, also geordnet vom inletseitigen zum outletseitigen Linienende, wird anschließend der mittlere Normalenvektor jeder Schnittlinie jeweils mit demjenigen des eben verschnittenen 2D-Blocks, aus welchem diese hervorgegangen ist, verglichen. Falls diese in entgegengesetzte Richtungen zeigen (negatives Skalarprodukt), wird die Indexreihenfolge invertiert.

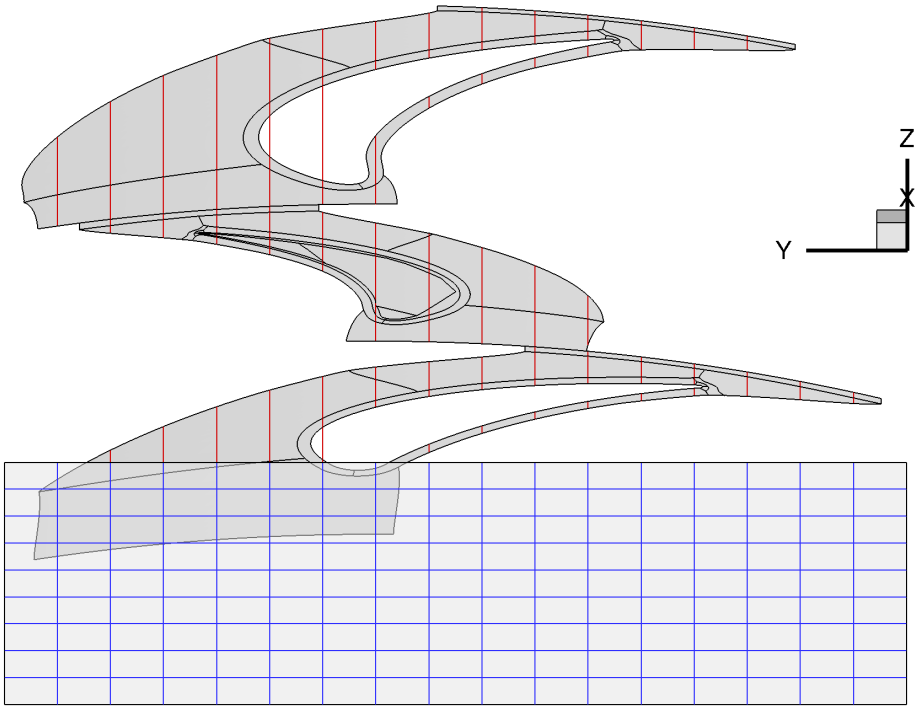


Abb. 5.4: Verschneidung der Gehäuseflächen - Prinzip (Testfall LISA)

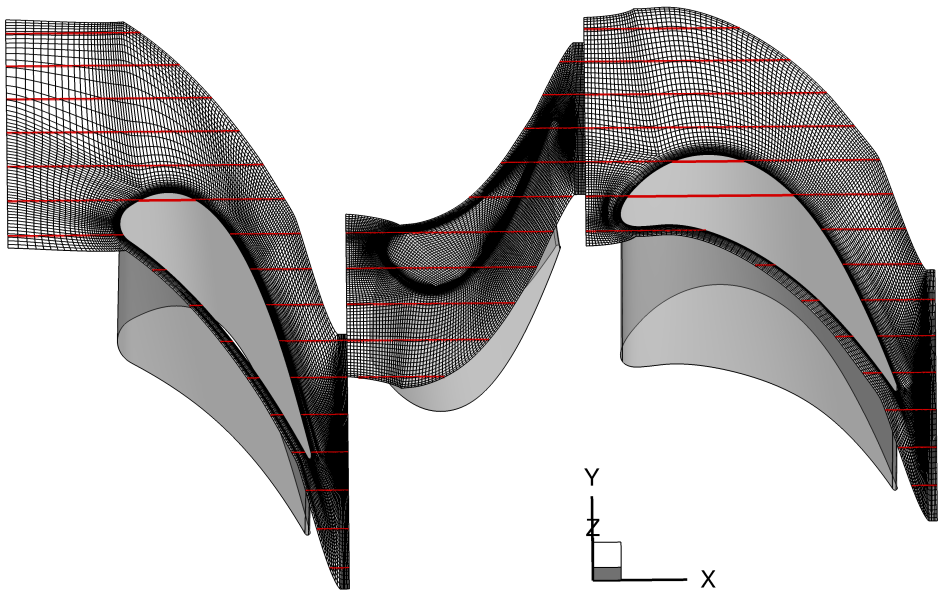


Abb. 5.5: Verschneidung der Gehäuseflächen - Draufsicht (Testfall LISA)

Abschließend werden die einzelnen Liniensegmente als jeweils ein 1D-Block unter dem Namen „IntersecMeshSegmentHub“ bzw. „IntersecMeshSegmentTip“, je nachdem, ob der verschnittene 2D-Elternblock zu einer der vorgegebenen Naben- oder Gehäusefami-

lien gehört, dem Datenset angefügt. Ebenso wird der Blocktyp *CUTMESH_BLOCK_-HUB* bzw. *CUTMESH_BLOCK_TIP* gesetzt, um die spätere Auftrennung der Schnittlinien in Naben- und Gehäuselinien zu ermöglichen.
Im Anschluss an den *generateThetaCuts*-Task werden dann alle 1D-Linien an einen Prozess kommuniziert, welcher auf Basis dieser Daten jeweils eine zusammenhängende Naben- und Gehäuselinie vom Inlet zum Outlet der Konfiguration bestimmt. Das Ergebnis dieses Tasks ist in *Abb. 5.5* anhand des Testfalls der LISA-Turbine für 18 θ-Schnitte dargestellt.

5.3.2 Ermittlung des Verlaufs der Minimal- und Maximalkontur - *generateExtremalOutline*

Name	generateExtremalOutline		
Aufruf	-geo		
Optionen	-type	min/max	obligatorisch

Tabelle 5.2: Übersicht über Aufruf und Optionen des *generateExtremalOutline*-Tasks

Dieser Abschnitt beschäftigt sich mit der Implementierung des *generateExtremalOutline*-Tasks (siehe *Tab. 5.2*). Dieser erstellt aus den Nabenkontur- und Gehäusekontur-Schnitten des vorangegangenen Tasks, *generateThetaCuts*, jeweils eine durchgehende Maximal- oder Minimallinie vom Einlass zum Auslass der analysierten Konfiguration. Es können dabei beliebige Turbomaschinen-Konfigurationen, also Axial-, Diagonal-, und Radialmaschinen, mit und ohne Naben- oder Gehäusekonturierung sowie weiterer Nebengeometrien basierend auf strukturierten, unstrukturierten oder hybriden Ausgangsnetzen behandelt werden. Eine Ausnahme bilden Konfigurationen, welche lokale omega-förmige Geometrien beinhalten, also Geometrien mit einer Winkeländerung von größer als 90°. Diese werden durch den Algorithmus nicht unterstützt, wobei zu beachten ist, dass seitens des *generateExtremalOutline*-Tasks hierfür keine automatische Überprüfung erfolgt.
Zur Generierung der beiden resultierenden Linien müssen alle 1D-Schnittlinien auf einem Prozess vorhanden sein. Zusätzlich ist ein vorheriger Aufruf des *collectGeometricalInfo*-Tasks (*Kap. 5.3.5*) mit der Option „-task geo“ erforderlich. Die beiden Konturlinien, welche sich als Resultat dieses Tasks ergeben, bzw. deren Punkteverteilung dienen anschließend als Grundlage zur Erstellung des temporären S2M-Netzes.
Zur Ausführung des Tasks ist vom Benutzer anzugeben, welches S2M-Netz erstellt werden soll. Entweder wird der Schattenriss der Konfiguration, also die Maximalabmessungen („-type max“), bestimmt oder die Minimalabmessungen („-type min“). Ermittlung der Maximalabmessungen bedeutet dabei minimale Naben-Konturlinie sowie maximale Gehäuse-Konturlinie und Ermittlung der Minimalabmessungen maximale Naben-Konturlinie sowie minimale Gehäuse-Konturlinie. Die Option „min“ entspricht somit einer Auswertung der Hauptströmung, wohingegen bei „max“ der komplette Schattenriss der Konfiguration erstellt wird und anschließend analysiert werden kann.
Im ersten Schritt werden alle Naben- und Gehäuse-Schnittlinien voneinander getrennt

in einem Netz-Pointer-Array abgelegt. Die Trennung erfolgt dabei auf Basis des Blocktyps, also entweder *CUTMESH_BLOCK_TIP* oder *CUTMESH_BLOCK_HUB*. Zusätzlich werden die kartesischen strukturierten 1D-Netze auf zylindrische Koordinaten konvertiert, in die xr-Ebene gedreht (θ-Koordinate = 0) und die Netzkanten- bzw. Edge-Struktur aufgebaut. Dann wird für jedes der beiden Netz-Pointer-Arrays die *generateExtremalOutline*-Kernroutine aufgerufen, um je nach Option den maximalen oder minimalen Konturverlauf zu bestimmen. Die Bestimmung der Naben- und Gehäuselinie erfolgt somit unabhängig voneinander.
Innerhalb der Kernroutine wird für das Array der übergebenen 1D-Netzlinien zunächst ein globaler mittlerer Normalenvektor basierend auf den Normalenvektoren der einzelnen Netzkanten berechnet und normiert. Dieser definiert die Suchrichtung zur Bestimmung der Maximal- oder Minimalkontur. Aufgrund der Anforderung beliebige Konfigurationen zu unterstützen, wird nicht die Radialrichtung als Suchrichtung verwendet, wie für Axialmaschinen üblich.
Im Anschluss daran wird die extremale, je nach Vorgabe entweder minimale oder maximale Start- und End-Netzlinie des übergebenen Linien-Arrays bestimmt. Der erste Vertex der extremalen Startlinie definiert dabei den Startpunkt des Gesamt-Algorithmus zur Konturermittlung und der letzte Vertex der extremalen Endlinie den Endpunkt, mit welchem abschließend der Erfolg des Algorithmus beurteilt wird. Bei Startlinien handelt es sich um Linien, deren erster Netzknoten bzw. Vertex auf der Inlet-Familie liegt und bei Endlinien um Linien, für welche der letzte Vertex auf dem Outlet liegt. Es wird somit eine vom einlassseitigen zum auslassseitigen Ende der Netzlinien geordnete Vertex-Reihenfolge vorausgesetzt.
Zur Bestimmung der Start- und Endlinien werden die im Rahmen des *collectGeometricalInfo*-Tasks ermittelten Koordinaten des jeweiligen Maximal- und Minimalpunkts des Inlets und Outlets herangezogen. Der Minimalpunkt wird dabei zur Generierung der Nabenlinie und der Maximalpunkt zur Generierung der Gehäuselinie verwendet. Wenn der Abstand des ersten Punkts einer Netzlinie vom jeweiligen Extremalpunkt kleiner als 10^{-3} ist, so handelt es sich bei dieser um eine Startlinie. Die Vorgehensweise zur Ermittlung der Endlinien ist analog.
Die nachfolgende Beschreibung beschränkt sich auf die Ermittlung der extremalen Startlinie. Dieselbe Vorgehensweise gilt jedoch auch für Endlinien. Nachdem alle Startlinien gesammelt worden sind, also diejenigen Linien, deren erster Vertex jeweils auf dem Inlet liegt, wird aus dieser Gruppe die extremale bestimmt. Hierfür ist zunächst die Suchrichtung zu definieren. Für die Suche nach der maximalen Linie entspricht diese dem Vektor vom Minimal- zum Maximalpunkt des Inlets und für eine Minimumsuche demjenigen vom Maximal- zum Minimalpunkt. Mittels der orthogonalen Projektion des Verbindungsvektors vom ersten Vertex einer Startlinie zu einem weiteren ersten Vertex einer zweiten Startlinie auf den Suchvektor wird entschieden, welche der beiden Linien höher bzw. niedriger liegt. Für eine Maximumsuche überprüft also die aktuell am höchsten liegende Netzlinie, ob alle anderen Startlinien unterhalb liegen. Die Schleife endet, wenn diejenige Linie gefunden ist, deren erster Vertex über allen anderen ersten Vertices bzw. auf gleicher Höhe mit diesen liegt.
Nach der Initialisierung beginnt der Algorithmus zur Generierung der jeweiligen Konturlinie nach *Abb. 5.6* . Dieser arbeitet vertexbasiert, da das Ziel einerseits die Erfas-

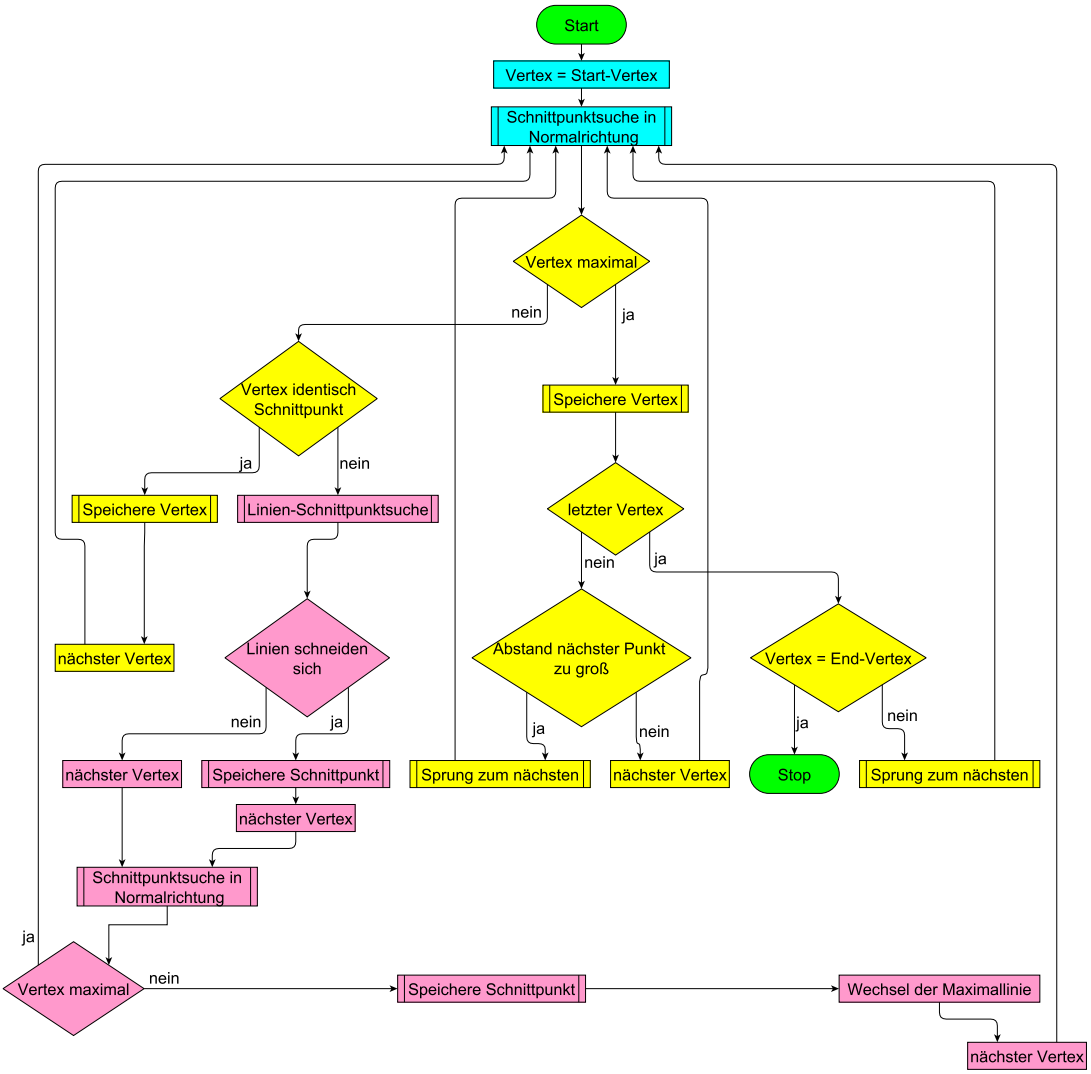


Abb. 5.6: Flussdiagramm des Kernalgorithmus des *generateExtremalOutline*-Tasks

sung des Konturverlaufs ist und andererseits die Vorgabe einer sinnvollen Punktverteilung entlang der resultierenden Linie. Denn auf dieser Punktverteilung basiert diejenige des vom *generateS2intersecMesh*-Task generierten temporären S2M-Netzes, welches zur Verschneidung der 3D-Konfiguration benutzt wird. Die Punktverteilung der resultierenden Linie orientiert sich an derjenigen der θ-Schnittlinien und damit an derjenigen der im Rahmen des *generateThetaCuts*-Tasks verschnittenen Naben- und Gehäuseflächen. Somit wird erreicht, dass Bereiche hoher Punktdichte der 3D-Konfiguration auf Bereiche hoher Punktdichte im temporären S2M-Netz führen.
Die Vorstellung des Taskablaufs erfolgt dabei für den Fall einer Maximumsuche. Der Kernalgorithmus arbeitet also, wie bereits erwähnt, vertexbasiert, was bedeutet, dass ein Vertex der aktuellen Maximallinie jeweils überprüft, ob in Normalrichtung eine höher liegende Netzlinie existiert. Dies erfolgt ihm Rahmen einer Schnittpunktsuche mit allen anderen Netzlinien des übergebenen Arrays. Wird kein gültiger Schnittpunkt oberhalb des aktuell suchenden Punkts gefunden, so wird davon ausgegangen, dass es sich bei diesem um den maximalen Vertex an dieser Position handelt. Unter bestimmten

Bedingungen, auf welche später im Rahmen der *storeNewVertex*-Routine eingegangen wird, erfolgt dann eine Übertragung des Punkts auf das 1D-Ergebnisnetz. Anschließend wird geprüft, ob es sich beim aktuellen Vertex um den letzten Netzpunkt der derzeitigen Maximallinie handelt. Wenn dies der Fall ist, wird der Abstand zum Outlet der Konfiguration überprüft. Beträgt dieser Abstand Null, endet der Algorithmus, da eine durchgehende Linie vom Inlet zum Outlet generiert worden ist. Andernfalls erfolgt der Übergang auf eine neue Maximallinie mittels der Funktion *getNextLineToJump*. Ebenso wird diese Sprungfunktion aufgerufen, wenn der Abstand zum folgenden Vertex zu groß ist. Hierbei gilt folgende Bedingung:

$$\Delta l_{vertices} > C_{max} \cdot \bar{l}_{edge} \qquad \Longrightarrow \qquad \text{Sprung}$$

$$\text{mit} \qquad C_{max} = 5$$

$\bar{l}_{edge}$ bezeichnet dabei die arithmetisch gemittelte Edge-Länge der aktuellen Maximallinie.
Handelt es sich beim aktuellen Vertex allerdings um einen inneren Netzknoten, dessen Distanz zum Nachbarpunkt innerhalb des vorgegebenen Limits liegt, so läuft der Algorithmus zum nächsten Vertex der derzeitigen Maximallinie und überprüft für diesen wieder, ob er das Konturmaximum an der jeweiligen Position darstellt.
Falls für den aktuell suchenden Vertex ein gültiger Schnittpunkt in Normalenrichtung gefunden wird, so bedeutet das, dass die gegenwärtige Maximallinie nicht mehr maximal ist und folglich auf die neue Maximallinie gewechselt wird. Zur Erhöhung der Genauigkeit wird bei jedem Wechsel zudem der Linienschnittpunkt zwischen neuer und alter Maximallinie bestimmt und gespeichert. Hierfür wird die aktuelle Edge der alten Maximallinie mit allen Edges der neuen auf Verschneidungen hin überprüft. Für den Fall mehrfacher Schnitte infolge stark unterschiedlicher Auflösung bzw. Edge-Längen der 1D-Netze, wird derjenige Schnittpunkt verwendet, welcher dem rechten Vertex der untersuchten Edge der gegenwärtigen Maximallinie am nächsten liegt. Der Algorithmus fährt dann mit demjenigen Vertex fort, welcher rechts vom ermittelten Schnittpunkt auf der neuen Maximallinie liegt.
Bevor der Algorithmus für den neuen Vertex wieder von vorne beginnt, wird zuvor überprüft, ob es sich bei diesem um den Maximalpunkt an der jeweiligen Position handelt. Denn es kommt vor, dass die ermittelte neue Maximallinie nur temporär maximal ist. Das bedeutet, dass zwar die aktuell ermittelte Maximallinie an der Suchposition des vorangegangenen Vertex maximal ist, jedoch eventuell nicht mehr an der Position, an welcher der Algorithmus fortgesetzt wird, also am rechts vom Schnittpunkt liegenden Vertex. An dieser Stelle kann bereits wieder eine andere Linie höher liegen. Die Suchrichtung stellt je nach Orientierung der Nachbaredges des suchenden Vertex der globale oder lokale Normalenvektor dar. Diese Abfrage wird dabei zur Verbesserung der Stabilität des Algorithmus durchgeführt. Wird ein gültiger Schnittpunkt mit einer höher liegenden Linie gefunden, fährt der Algorithmus wiederum am rechten Vertex derjenigen Edge fort, auf welcher dieser Schnittpunkt liegt. Um die lokale Netzauflösung durch das Überspringen einzelner Punkte nicht zu verschlechtern, wird der gefundene, interpolierte Schnittpunkt ebenfalls ins Ergebnisnetz übernommen.
Wird kein Linienschnittpunkt gefunden, da die beiden Schnittlinien lokal parallel verlaufen, läuft der Algorithmus ebenso am rechten Vertex der bei der Schnittpunktsu-

che in Normalrichtung getroffenen Edge der neuen Maximallinie weiter. Genauer wäre es natürlich, mit dem linken Vertex fortzufahren, was jedoch unter bestimmten Voraussetzungen zu Endlosschleifen führen kann. Obwohl nicht aktiviert, gehen viele der implementierten Stabilisierungsmaßnahmen auf die Anforderung am linken Vertex weiterzulaufen zurück.
Wenn die Maximallinie wechselt, folgt jedoch nicht in jedem Fall die Suche nach einem Linienschnittpunkt. Zunächst wird überprüft ob der Abstand des gefundenen Schnittpunkts vom suchenden Vertex kleiner als 10^{-6} ist. In diesem Fall liegt der Vertex direkt auf der neuen Maximallinie und wird auf das Ergebnisnetz übertragen. Anschließend erfolgt ein Vergleich der beiden rechts von diesem Schnittpunkt liegenden Vertices, um die neue Maximallinie und den nächsten Vertex zu definieren, an welchem der Algorithmus fortgesetzt wird.
Nun wird etwas detaillierter auf die wichtigsten Unterroutinen eingegangen:

Schnittpunktsuche (*intersectionCore*):

Die Aufgabe der Schnittpunktsuche besteht in der Überprüfung, ob ein gegebener Vertex an dessen Position das Konturmaximum darstellt. Ausgehend von diesem Vertex wird hierzu ein Suchvektor definiert und die somit bestimmte Gerade mit allen Edges jeder Netzlinie mit Ausnahme derjenigen, welche zum selben θ-Schnitt wie der aktuell suchende Punkt gehören, verschnitten.
Beim Suchvektor handelt es sich dabei in der Regel um den globalen mittleren Normalenvektor. Falls der suchende Vertex jedoch so liegt, dass der Winkel zwischen dessen lokalen und dem globalen Normalenvektor mehr als 75° beträgt, dann wird die lokal ermittelte Normale zur Maximumsuche verwendet. Die Berechnung der lokalen Normale basiert dabei auf den Normalenvektoren der Nachbaredges des jeweiligen Vertex. Auf diese Weise können auch u-förmige Radialverdichter ausgewertet werden. Der lokale Normalenvektor wird aber nur in Ausnahmefällen benutzt, da zu erwarten ist, dass dieser für bestimmte Linien-Konfigurationen zu ungenaueren Ergebnissen führt. In *Abb. 5.7* können beispielsweise die Vertices der blauen Linie mit dem lokalen Normalenvektor als Suchrichtung die eigentlich höher liegende schwarze Linie nicht detektieren. Bei Verwendung des globalen Normalenvektors stellt dies jedoch kein Problem dar.
Wie bereits erwähnt, wird für alle Edges überprüft, ob diese sich oberhalb bzw. für eine Minimumsuche unterhalb des aktuellen Punkts befinden. Im letzteren Fall, also wenn nicht die Maximalkontur, sondern die Minimalkontur gesucht wird, wird der Suchvektor einfach umgedreht. Hinsichtlich der Schnittpunktsuche werden Netzkanten nicht berücksichtigt, wenn

- der Suchvektor und die Edge-Normale in entgegengesetzte Richtungen zeigen (z.B. bei u-förmigen Konfigurationen)
- die Edge-Länge zu groß ist, ergo $> 5 \cdot \bar{l}_{edge}$
- der suchende Vertex der erste Vertex und die zu überprüfende Edge die letzte Edge ihrer jeweiligen Linie sind

In allen anderen Fällen wird zunächst ein Geradenschnittpunkt bestimmt und anschließend überprüft, ob dieser innerhalb der aktuell betrachteten Edge, also

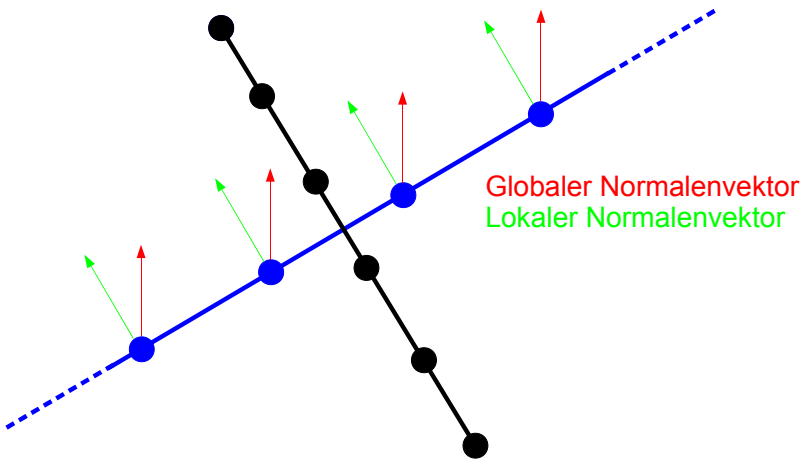


Abb. 5.7: Grafischer Vergleich der Schnittpunktsuche unter Verwendung des globalen und des lokalen Normalenvektors

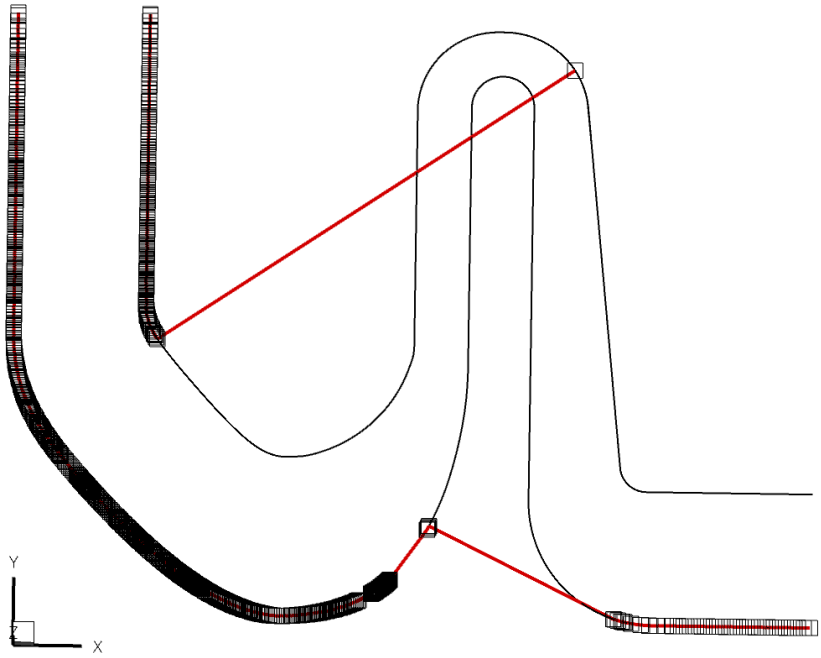


Abb. 5.8: Fehlschlagen des Algorithmus aufgrund falsches Schnittpunkt-Ermittlung

zwischen deren beiden Begrenzungsvertices, liegt. Ebenso wird die Distanz zum suchenden Vertex berechnet. Beträgt diese mehr als 10% des Abstands vom Inlet-Mittelpunkt zum Outlet-Mittelpunkt der Konfiguration, handelt es sich um keinen gültigen Schnittpunkt. Diese Einschränkung ist zur Analyse von Radialverdichtern nötig, da ansonsten die Möglichkeit besteht, dass falsche Schnittpunkte gefunden werden und der Algorithmus zur Konturermittlung fehlschlägt (siehe *Abb. 5.8*). Daraus jedoch können für Konfigurationen Probleme entstehen, bei welchen die Maximalabmessungen ermittelt werden sollen und größere nicht umfangssymetrische Spalte vorliegen. Wenn jedoch der Schnittpunkt innerhalb der untersuchten Edge und der Abstand zum suchenden Vertex innerhalb des vorgegeben Limits

liegt, handelt es sich um einen gültigen Schnittpunkt.
Da es im Allgemeinen vorkommt, dass mehrere Linien über dem aktuell suchenden Vertex liegen, wird der am höchsten liegende valide Schnittpunkt und damit die Maximallinie an dieser Position gesucht. Der am höchsten liegende Schnittpunkt ist gleichzeitig derjenige, welcher vom suchenden Vertex die maximale Distanz aufweist.

Speicherroutine (*storeNewVertex*):

Die Speicherroutine führt zunächst anhand mehrerer Kriterien eine Überprüfung des übergebenen Punkts durch und speichert dessen Koordinaten nur dann in das Ergebnisnetz, falls er die folgenden Voraussetzungen erfüllt:

- Punkt ist noch nicht gespeichert
- Punkt liegt nicht zu nahe am zuletzt gespeicherten Vertex. Es gilt:

$$\Delta l_{vertices} > C_{min} \cdot \bar{l}_{edge}$$

$$\text{mit} \qquad C_{min} = 0,075$$

- Punkt liegt „rechts“ vom zuletzt gespeicherten Vertex

Die Überprüfung, ob der übergebene Punkt „rechts“ vom letzten Vertex liegt, geschieht mittels einer orthogonalen Projektion des Verbindungsvektors vom zuletzt gespeicherten Vertex zum aktuell zu speichernden Punkt auf den Richtungsvektor, welcher durch die beiden letzten Vertices des Ergebnisnetzes definiert wird. Ist das Skalarprodukt größer Null, wird der Punkt ins resultierende Netz übernommen. Diese Abfrage ist natürlich erst ab dem dritten zu speichernden Punkt möglich. Sie gewährleistet die kontinuierliche Ordnung bzw. Nummerierung der Vertices vom einlassseitigen zum auslassseitigen Ende der resultierenden Konturlinie. Die Netzlinien des fertigen S2M-Netzes überkreuzen sich somit nicht mehr aufgrund falscher Indizierung der Netzknoten.

Sprungroutine (*getNextLineToJump*):

Dadurch, dass der *generateThetaCuts*-Task strukturierte Schnittlinien erzeugt, können sich bei der Verschneidung von Schaufel-Grenzschichtblöcken relativ große Zwischenräume in der resultierenden Netzlinie ergeben (siehe *Abb. 5.9*). Zur Optimierung der Netzqualität bzw. der Punktverteilung des Ergebnisnetzes erfolgt ein Sprung zu einer der anderen makroskopisch parallel verlaufenden θ-Schnittlinien, welche an dieser Position keine Schaufel schneidet und folglich keine Lücke aufweist. Die optimale dieser parallel verlaufenden Linien zu finden ist die Aufgabe der Sprungroutine. Eine Lücke im Netz ist dabei, wie bereits erwähnt, wie folgt definiert:

$$\Delta l_{vertices} > C_{max} \cdot \bar{l}_{edge} \qquad \Longrightarrow \qquad \text{Sprung}$$

$$\text{mit} \qquad C_{max} = 5$$

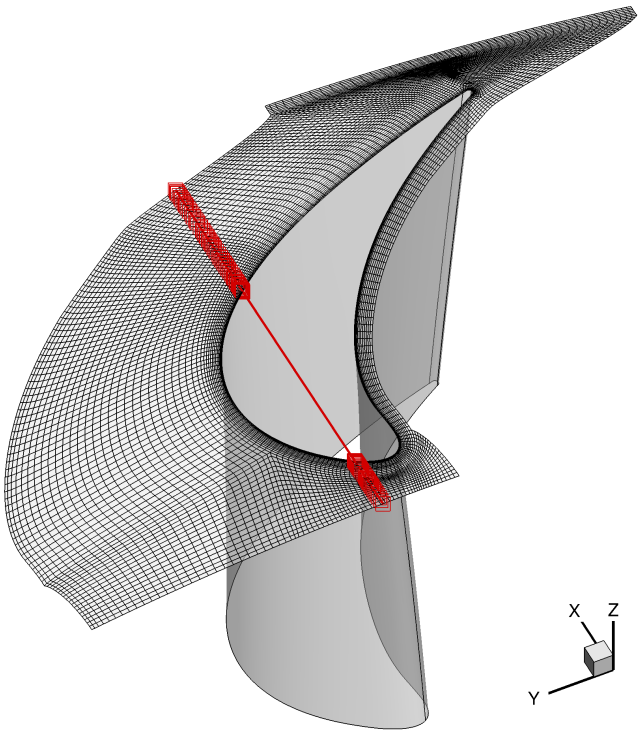


Abb. 5.9: Schnittlinien-Lücke infolge Verschneidung eines Schaufel-Grenzschichtblocks

Des Weiteren wird diese Funktion verwendet, wenn der Algorithmus an einem letzten Vertex einer Maximallinie angelangt ist. In diesem Fall wird zunächst versucht auf eine unmittelbar anschließende Linie des gleichen θ-Schnitts überzuwechseln. Die Koordinaten des letzten Vertex sind in diesem Fall identisch zu denen des ersten Vertex der folgenden Linie. Wird keine sich unmittelbar anschließende Linie gefunden, wird zur Bestimmung der neuen Linie und des nächsten Vertex die gleiche Vorgehensweise wie für Sprünge bei einer Lücke im Netz angewandt.
Zunächst werden hierfür in der Routine *getNextLineToJump* die Koordinaten des optimalen nächsten Punkts definiert. Bei diesem handelt es sich um das Resultat einer Punktspiegelung des linken Nachbar-Vertex am der Sprungfunktion übergebenen Vertex. Auf diese Weise wird eine lokale Punktverdichtung vermieden, das heißt die Auflösung im Bereich um die Sprungstelle herum bleibt ungefähr konstant. Im Rahmen einer Schleife über alle Vertices aller Netzlinien wird dann derjenige Punkt gesucht, welcher den minimalen Abstand zum Optimalpunkt aufweist. Als zweite Bedingung muss dieser außerdem „rechts“ vom übergebenen Vertex liegen, was mittels einer orthogonalen Projektion auf den Richtungsvektor vom linken Nachbarpunkt zum übergebenen Punkt überprüft wird. Folglich kann die Sprungfunktion für einen ersten Vertex einer Linie nicht aufgerufen werden, was jedoch in der Regel auch nicht nötig ist.
Zur Erhöhung der Stabilität wird für den ermittelten Punkt anschließend mittels einer Schnittpunktsuche in Normalrichtung überprüft, ob dieser an der jeweiligen Stelle das Maximum der Kontur darstellt. Falls ein gültiger Schnittpunkt gefunden wird, ergo der gefundene Optimalpunkt nicht maximal ist, so erfolgt ein Wechsel auf den rechtsseitigen Vertex derjenigen Edge der Maximallinie, welche über dem

suchenden Punkt liegt. Der gefundene Schnittpunkt wird zudem ins Ergebnisnetz gespeichert, sodass die angestrebte, möglichst identische Zellgröße rechts und links des Sprungpunktes erreicht wird.

Anzumerken ist, dass sich die angeführten Stabilisierungsmaßnahmen und Einschränkungen teilweise überschneiden. Diese senken zwar die Genauigkeit des Ergebnisses, jedoch wird es als wichtiger erachtet, einen numerisch stabilen Algorithmus zu implementieren und hierfür auf einige Vertices, welche Probleme bereiten könnten, zu verzichten. Aus diesem Grund läuft der Algorithmus grundsätzlich nach „rechts“, wenn eine neue Maximallinie gefunden wird.
Die meisten Stabilitätsprobleme, genauer gesagt Endlosschleifen, verursachen makroskopisch parallel liegende Linien. Durch die Verschneideroutinen, welche von *INTERSEC* und *VTK* bereitgestellt werden, liegen diese aufgrund von numerischen Ungenauigkeiten nicht mehr exakt parallel, sodass es auch bei umfangssymetrischen Testfällen zu mehreren Verschneidungen der einzelnen Linien kommt. Um im Fall einer Endlosschleife den Prozess zu beenden, wird die Anzahl der suchenden Vertices mitgezählt. Überschreitet diese Zahl die Gesamtanzahl an vorhandenen Vertices, wird das Programm beendet. Zur Stabilisierung des Algorithmus kann anschließend versucht werden die Anzahl an Schnitten im *generateThetaCuts*-Task zu ändern.
In diesem Zusammenhang ist zu erwähnen, dass mehr θ-Schnitte das Ergebnis nicht unbedingt verbessern, sie können es je nach Verlauf der 1D-Netzlinien sogar verschlechtern. Denn die verschiedenen Stabilisierungsmaßnahmen sorgen dafür, dass einzelne Punkte übergangen werden bzw. der Anteil an interpolierten Punkten im Ergebnisnetz gegenüber den „realen“ Punkten zunimmt. Je mehr Linien vorhanden sind, desto häufigere Überschneidungen gibt es zwischen den eigentlich parallelen Linien und bei häufigen Linienwechseln nehmen wiederum die Eingriffe durch Stabilisierungsmaßnahmen zu. Ein weiterführender Ansatz wäre natürlich die Schnittpunktsuche dahingehend anzupassen, dass parallele Linien auch als solche behandelt werden. Hierfür müsste jedoch ein möglichst allgemeingültiger Faktor definiert werden, über welchen „falsche“ Schnittpunkte, ergo Schnittpunkte, welche zu nahe an der alten Maximallinie liegen, automatisch erkannt werden. Dennoch bleibt die Gefahr von Endlosschleifen weiterhin bestehen.
Der Algorithmus stoppt, wenn er am Outlet der Konfiguration angelangt ist. Der letzte Punkt des Ergebnisnetzes wird dann mit dem vorab ermittelten extremalen Outlet-Punkt verglichen. Stimmen beide überein, so wird die jeweilige Konturlinie als neuer 1D-Block dem Datenset angefügt. Das Netz besitzt dabei eine strukturierte Topologie und ist in zylindrischen Koordinaten in der xr-Ebene definiert. Der Blocktyp ist entweder *OUTLINE_TIP* oder *OUTLINE_HUB* und der Zonenname einer der folgenden:

- Tip_Minimum_Contour
- Tip_Maximum_Contour
- Hub_Minimum_Contour
- Hub_Maximum_Contour

Im Anschluss an den *generateExtremalOutline*-Task können dann die nicht mehr benötigten 1D-Linien, welche zur Erzeugung der Extremal-Konturlinien benutzt wurden,

mit dem *deleteBlock*-Task gelöscht werden. Die beiden Konturlinien müssen schließlich noch an alle Prozesse kommuniziert werden. Somit ist dann jeder Prozess in der Lage das für die Verschneidung der 3D-Konfiguration benötigte S2M-Netz zu generieren. Das Ergebnis dieses Tasks sowohl für die Option „min“ als auch die Option „max“ zeigt *Abb. 5.10* am Beispiel der Naben-Netzlinie des Testfalls Rig250 mit Nabenkonturierung.

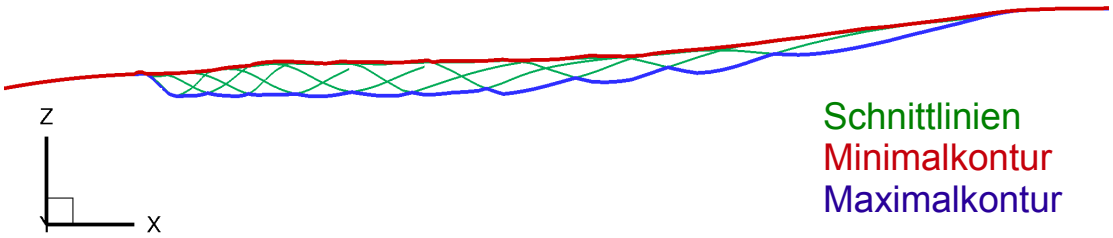


Abb. 5.10: Resultat des *generateExtremalOutline*-Tasks für die Optionen Minimal- und Maximalkontur anhand eines Ausschnitts der Naben-Netzlinie des Testfalls Rig250 mit Nabenkonturierung

Zum Schluss wird noch auf einen weiteren Ansatz zur Ermittlung der Minimal- und Maximalkontur aus einer beliebigen Anzahl von 1D-Schnittlinien eingegangen. Der Ablauf dieses Algorithmus könnte folgendermaßen aussehen:

- Ermittlung und Markierung aller auftretenden Edge-Schnittpunkte
- An jedem Schnittpunkt den lokal maximalen Sub-Polygonzug, welcher bis zum nächsten Schnitt- oder Endpunkt definiert ist, behalten und den minimalen löschen
- Nach Entfernung aller Schnittpunkte endet der Algorithmus und liefert abhängig von der untersuchten Konfiguration je eine oder mehrere Naben- und Gehäuselinien, welche jeweils keine gemeinsamen Schnittpunkte aufweisen
- Die Maximalkontur entspricht dann demjenigen Linienpaar, welches die maximale Fläche einschließt
- Die Minimalkontur entspricht dann demjenigen Linienpaar, welches die minimale Fläche einschließt

Dieser Algorithmus wurde nicht umgesetzt, da lokal abseits liegende Netzlinien, welche keinerlei Schnittpunkte mit den übrigen Linien aufweisen, nicht in die Bestimmung der Kontur mit einbezogen werden können. Diese entstehen beispielsweise bei der Verschneidung von Spalt-Randflächen. Bei der implementierten Schnittpunktsuche in Normalenrichtung stellt dies jedoch keinerlei Problem dar.

5.3.3 Erstellung des temporären S2M-Netzes - *generateS2intersecMesh*

Die Aufgabe des Tasks *generateS2intersecMesh* (siehe *Tab. 5.3*) besteht in der Generierung des strukturierten temporären S2M-Netzes in der *xr*-Ebene zur anschließenden Verschneidung der 3D-Konfiguration. Der Task basiert dabei auf den beiden im Rahmen

Name	generateS2intersecMesh		
Aufruf	-genS2		
Optionen	-reference	hub/tip	optional
		max. Vertices	default
	-xieta	index/length	obligatorisch

Tabelle 5.3: Übersicht über Aufruf und Optionen des *generateS2intersecMesh*-Tasks

des *generateExtremalOutline*-Tasks ermittelten Konturlinien für Nabe und Gehäuse.
Zuerst werden die zwei eindimensionalen Berandungskurven mit gegebener Punktverteilung aus der Gruppe der lokalen Blöcke herausgesucht. Diese stehen dabei allen Prozessen zur Verfügung. Anschließend wird entschieden, welche der beiden Netzlinien als Referenzkurve dient und für welche eine Punktumverteilung bzw. ein Remapping durchgeführt wird. Durch den Benutzer kann gesteuert werden, ob die Naben- („-reference hub“) oder Gehäuselinie („-reference tip“) als Referenzlinie zu verwenden ist. Wenn keine Vorgabe seitens des Anwenders erfolgt, dann wird diejenige Kurve als Referenzlinie herangezogen, welche über eine bessere Auflösung, ergo über mehr Netzpunkte, verfügt. Probleme können hierbei auftreten, wenn sich die lokale Punktverteilungen von Naben- und Gehäuselinie stark unterscheiden. Dementsprechend sollte beispielsweise zur Analyse einer Nabenkonturierung die Naben-Netzlinie als Referenznetz vorgegeben werden, um deren Verlauf identisch auf das resultierenden S2M-Schnittnetz zu übertragen.
Im Anschluss folgt die Remapping-Routine. Deren Ziel besteht zunächst darin, die jeweilige Punktverteilung der beiden Berandungskurven aufeinander anzupassen. Aufgrund der strukturierten Netztopologie verfügen die beiden Randkurven dabei über eine identische Punktanzahl, welche durch die Referenzlinie vorgegeben wird. Das Remapping erfolgt dann mit der Vorgabe, dass die zwei Punkte der Naben- und der Gehäuselinie, welche denselben Index haben auch dieselbe relative Lauflänge aufweisen. Die Normierung erfolgt dabei mit der jeweiligen Netzlinien-Gesamtlänge. Anzumerken ist, dass die Ausgangsnetze unverändert bleiben, da die Positionen der umverteilten Punkte direkt ins Ergebnisnetz gespeichert werden.
Zunächst werden also die beiden Gesamtlängen in Meridionalrichtung, l_{ref} für das Referenznetz sowie l_{rem} für das Remapnetz, nach folgender Formel ermittelt:

$$l = \sum_{nEdges} l_{edge} \tag{5.1}$$

Auf Basis des Referenznetzes wird dann das Array aller relativen Liniensegmentlängen bestimmt, welches ebenso für die neue Punktverteilung der gegenüberliegenden Konturlinie gelten muss. Die Vorgabe für die Implementierung des Algorithmus, welcher die Umverteilung der inneren Netzpunkte der Konturlinie vornimmt, ist, dass alle neuen Punkte auf der Kontur des Originalnetzes liegen müssen. Anzumerken ist hierbei, dass die neue Berandungslinie je nach Wahl des Referenznetzes über mehr oder weniger Punkte verfügen kann als das Ausgangsnetz. Im Folgenden wird nun die Umsetzung des Remap-Algorithmus für die Berandungslinie vorgestellt. Dieser setzt sich aus zwei Einzelschritten zusammen.

1. Schritt: Ermittlung des benötigten Liniensegments des Originalnetzes

Zur Ermittlung des Liniensegments des Originalnetzes, in welchem der neu zu generierende Punkt liegt, wird zunächst mit Hilfe des Längen-Arrays dessen Abstand vom Linienanfang, also vom Inlet der Konfiguration, in Lauflängenrichtung bestimmt. Anschließend werden die Längen der Liniensegmente des Originalnetzes ausgehend vom Inlet solange aufsummiert bis der berechnete Sollabstand des neuen Punktes erreicht oder überschritten wird. Damit ist dasjenige Liniensegment des Originalnetzes gefunden, in welchem der neue Punkt liegt.

2. Schritt: Ermittlung der Koordinaten des neuen Punkts

Hierbei gibt es zwei verschiedene Lösungswege, je nachdem, ob der aktuelle und der zuletzt generierte Punkt im selben oder in unterschiedlichen Liniensegmenten liegen.

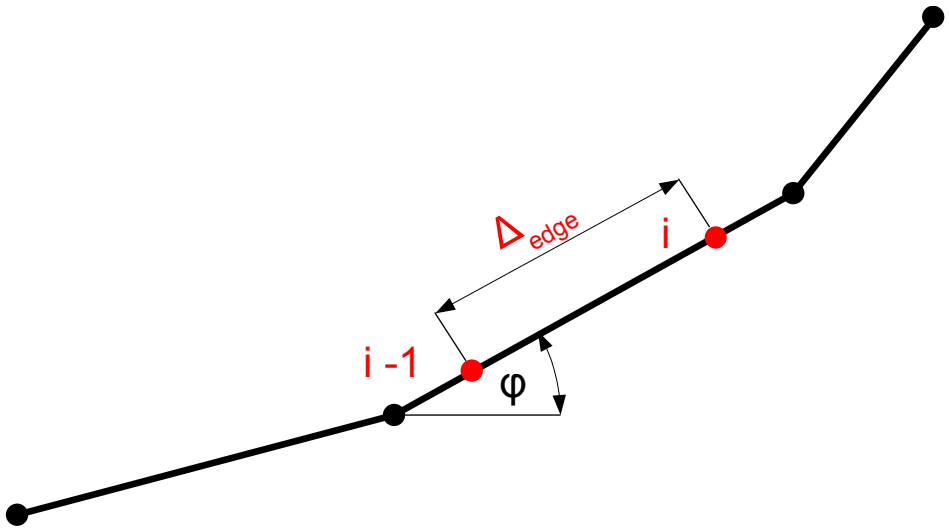


Abb. 5.11: Fall 1: Gleiches Liniensegment des Originalnetzes (schwarz)

Für den Fall nach *Abb. 5.11* , der aktuelle Punkt i und der zuletzt generierte Punkt i-1 liegen also im selben Liniensegment des Originalnetzes, werden die Koordinaten des neuen Punkts folgendermaßen berechnet:

$$\begin{pmatrix} x \\ r \\ 0 \end{pmatrix}_i = \begin{pmatrix} x \\ r \\ 0 \end{pmatrix}_{i-1} + \begin{pmatrix} cos\varphi \\ sin\varphi \\ 0 \end{pmatrix} \cdot \Delta_{edge} \tag{5.2}$$

Dabei ergibt sich Δ_{edge} aus der Netzlinien-Gesamtlänge l_{rem} und dem jeweiligen Eintrag des Liniensegmentlängen-Arrays.

Im zweiten Fall, welcher in *Abb. 5.12* dargestellt ist, liegen der aktuelle und der zuletzt gespeicherte Punkt auf unterschiedlichen Liniensegmenten des Ausgangsnetzes, wobei die beiden Punkte nicht notwendigerweise in direkt aufeinanderfolgenden Segmenten liegen müssen. Zunächst wird der Winkel α_{v1v2} zwischen den Vektoren $\vec{v}_1$ und $\vec{v}_2$ bestimmt. Bei $\vec{v}_1$ handelt es sich dabei um den Vektor vom zuletzt generierten Punkt zum Anfangspunkt des Liniensegments, in welchem der neue Punkt liegt, und bei $\vec{v}_2$ um den Vektor vom Anfangs- zum Endpunkt dieses Liniensegments. Unter Verwendung des Winkels sowie der Vektorlänge v_1 und

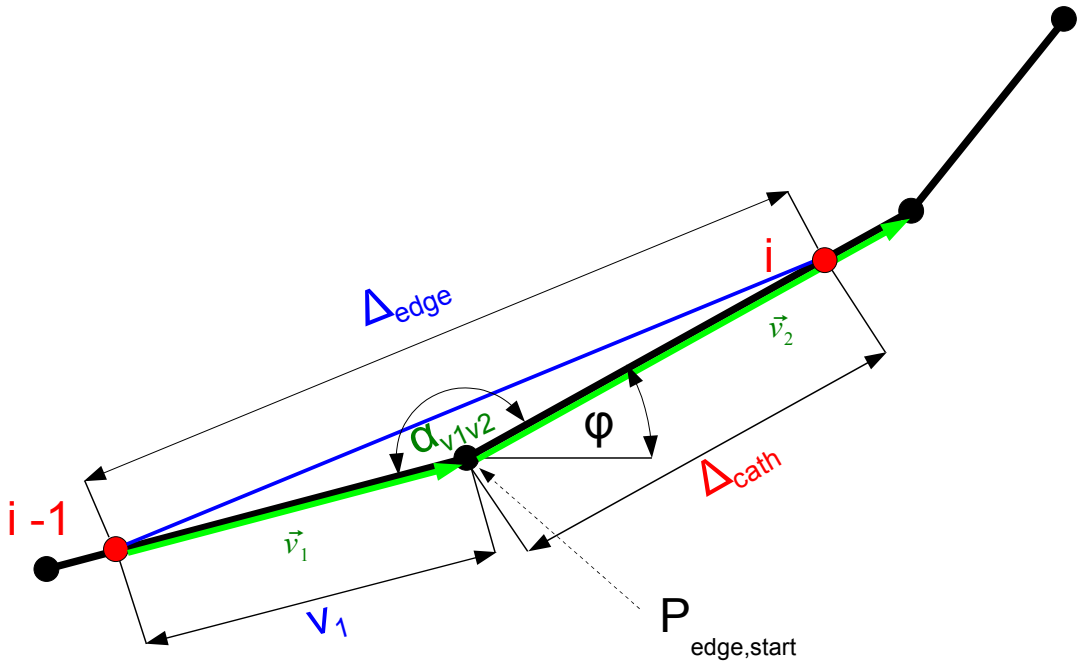


Abb. 5.12: Fall 2: Unterschiedliche Liniensegmente des Originalnetzes (schwarz)

dem vorgegebenen Abstand Δ_{edge} lässt sich mit Hilfe des Kosinussatzes Δ_{cath}, der Abstand des neuen Punktes zum Anfangspunkt des zugehörigen Liniensegments $P_{edge,start}$, bestimmen. So ergibt sich:

$$\begin{pmatrix} x \\ r \\ 0 \end{pmatrix}_i = \begin{pmatrix} x \\ r \\ 0 \end{pmatrix}_{P_{edge,start}} + \begin{pmatrix} cos\varphi \\ sin\varphi \\ 0 \end{pmatrix} \cdot \Delta_{cath} \tag{5.3}$$

Nach Abschluss der Punktumverteilung erfolgt schließlich die Vernetzung des von den beiden Berandungskurven eingeschlossenen 2D-Gebiets. Hierfür werden die jeweils zueinander gehörigen Punkte der Naben- und Gehäuselinie miteinander verbunden. Diese in Normalrichtung verlaufenden Verbindungsstrecken werden anschließend mit Ausnahme des ersten und letzten Liniensegments in äquidistante Teilstücke zerlegt. Die Randliniensegmente verfügen dabei jeweils über die halbe Länge wie die übrigen Segmente. In Normalrichtung verfügt das temporäre S2M-Netz über 72 Punkte, wohingegen sich die Punktzahl in Meridionalrichtung aus dem vorangegangenen *generateExtremalOutline*-Task und damit aus der Auflösung der zu untersuchenden Konfiguration ergibt. Diese Verteilung ist analog zur Standard-Banddefinition des nachfolgenden *intersect3DS2MPlane*-Tasks (*Kap. 5.4*).

Im Anschluss an die eigentliche Netzgenerierung werden für jeden Netzknoten des S2M-Netzes dessen ξ- und η-Koordinate bestimmt. Dabei stellt ξ die normierte Koordinate in meridionaler und η in normaler Richtung dar. Beide sind auf dem Intervall $[0; 100]$ definiert. Auf Grundlage der ξ- und η-Verteilung des S2M-Netzes wird nachfolgend die Verschneidung der 3D-Konfiguration durchgeführt. Da die einzelnen Schnitt- und Bandpositionen als ξ- und η-Werte definiert sind, wird die geometrische Lage der Bänder und

der Gesamtschnittfläche vom S2M-Netz bestimmt. Denn dieses liefert den Zusammenhang zwischen ξ- und η-Koordinaten der Schnitte und Bänder und der geometrischen Position, also der x- und r-Koordinaten der einzelnen Punkte der Schnittfläche.
Zur Berechnung der dimensionslosen Koordinaten und damit auch zur Steuerung der Lage der Schnittflächen, existieren zwei vom Benutzer vorgebbare Möglichkeiten:

- indexbasierte ξ- und η-Definition („-xieta index“)
- längenbasierte ξ- und η-Definition („-xieta length“)

Bei der indexbasierten Variante entsprechen die ξ- und η-Werte den durch die Gesamtzahl an Vertices der jeweiligen Netzlinie dividierten I- und J-Indizes, also der Indizes der Netzknoten in Meridional- bzw. Normalrichtung. Der Vorteil hierbei ist, dass Bereiche hoher Punktdichte stärker gewichtet werden. Für die bezüglich ξ äquidistante Verteilung der Schnittebenen im folgenden *intersect3DS2MPlane*-Task ergibt sich somit, dass in Bereichen mit hoher Auflösung mehr Schnittflächen liegen als in den übrigen Bereichen. Diese Option gewährleistet zudem, dass die Netzlinien des resultierenden S2M-Netzes, welche vom *buildFinalS2M*-Task (*Kap. 5.7*) aufgebaut wird, ähnlich liegen wie diejenigen des 2D-Ausgangsnetzes. Dies ist zum Beispiel dann von besonderer Bedeutung, wenn die Netzlinien auch für komplexere Geometrien orthogonal zur Naben- und Gehäusewand verlaufen sollen.
Nachteilig jedoch ist, dass mit dieser Variante für die folgende Verschneidung der 3D-Konfiguration keine geometrisch äquidistante Verteilung der Schnittflächen vorgegeben werden kann. Zudem ist es nicht ohne Weiteres möglich, Schnittflächen an bestimmten Lauflängenpositionen zu definieren. Aus diesem Grund ist als zweite Option die längenbasierte Variante implementiert. Die ξ- und η-Koordinaten entsprechen somit der relativen Position in Meridional- bzw. Normalrichtung bezogen auf die Gesamtlänge der jeweiligen Netzlinie.

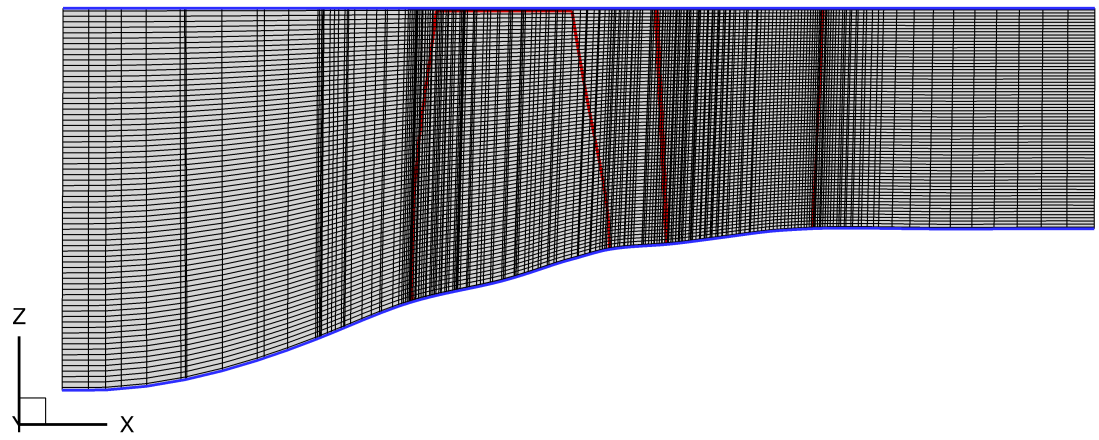


Abb. 5.13: Temporäres S2M-Schnittnetz (Testfall THD Rotor 1)

Das fertige S2M-Netz sowie die „XiEta“-Lösung werden dann als neuer Block mit dem Zonennamen „S2MSurface“ gespeichert. Das Resultat dieses Tasks, beispielhaft für den THD Rotor 1, zeigt *Abb. 5.13*. Die auf diesem Netz basierende ξ-Verteilung ist in *Abb. 5.14* für die Option „length“ und in *Abb. 5.15* für die Option „index“ dargestellt. Anhand der jeweils identischen, äquidistanten Stufung der ξ-Isolinien ist zudem der Einfluss

der ξ-Verteilung auf die geometrische Lage der Schnittflächen, welche im folgenden Task generiert werden, zu erkennen.

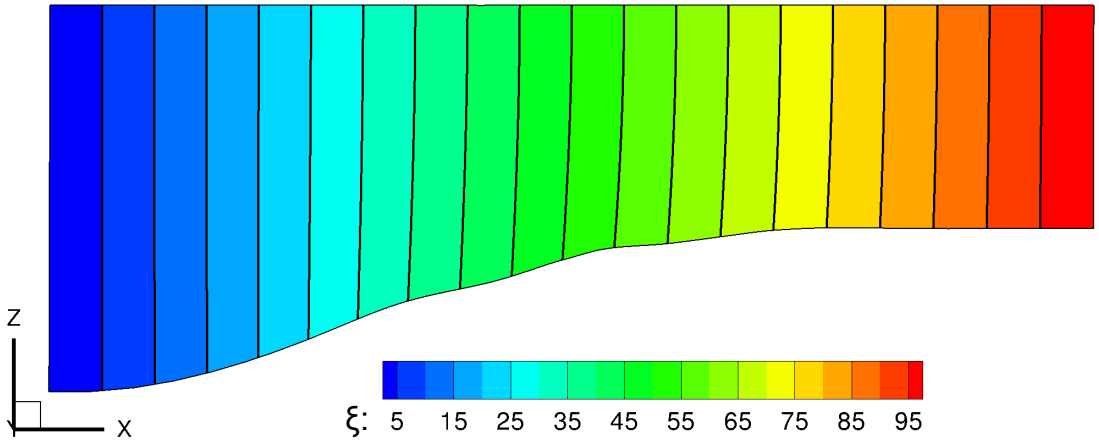


Abb. 5.14: Temporäre S2M-Fläche mit ξ-Verteilung und ξ-Isolinien (äquidistante Verteilung, Stufung $\Delta\xi = 5$) für Option „length“

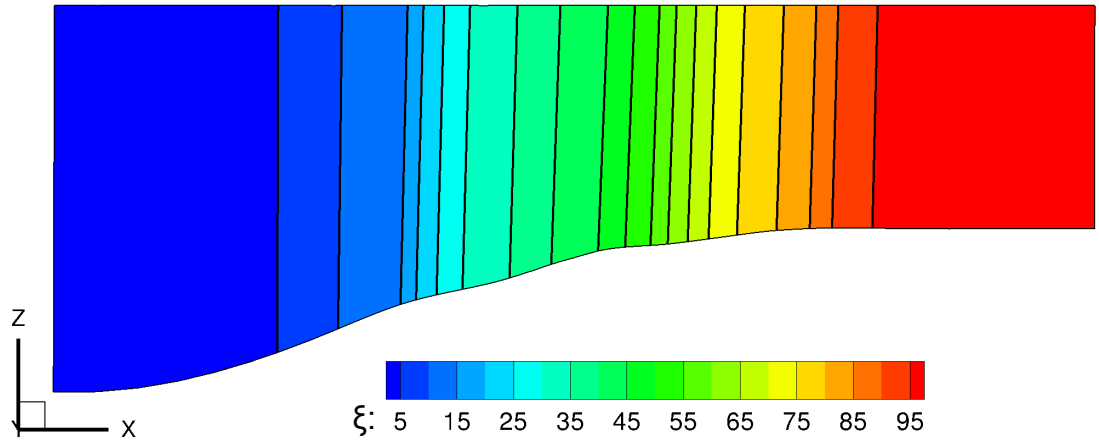


Abb. 5.15: Temporäre S2M-Fläche mit ξ-Verteilung und ξ-Isolinien (äquidistante Verteilung, Stufung $\Delta\xi = 5$) für Option „index“

Abschließend wird nun auf bekannte bzw. erwartete Probleme sowie eventuelle Lösungsmöglichkeiten eingegangen. Zunächst war geplant das temporäre S2M-Netz unstrukturiert zu generieren, um beliebige Geometrien, inklusive Kavitäten, Spalte oder Dichtungen, erfassen zu können. Das Problem bei der Verwendung eines unstrukturierten Gitters ist jedoch die ξ- und η-Definition. Denn aufgrund der nicht vorhandene Struktur können diese weder index- noch längenbasiert ermittelt werden.
Aus diesem Grund fiel schließlich die Wahl auf ein strukturiertes Netz. An Positionen von Kavitäten nimmt somit die mittlere Banddicke zu, da die η-Koordinate weiterhin von der Nabe bis zum Gehäuse mit einem Wertebereich von 0 bis 100 definiert ist. Dabei wird angenommen, dass die Abmessungen der Sekundärströmungsbereiche im Vergleich zu denjenigen der Hauptströmung relativ klein sind und somit keine übermäßige Aufdickung der Bänder erfolgen wird. Sollte dies nicht der Fall sein, so ist eventuell eine getrennte Auswertung von Hauptströmung und Kavität durchzuführen.
Ein weiteres Problem stellt die Vernetzung von Spalten und damit auch Kavitäten dar,

da sich bei diesen der Naben- bzw. Gehäusekonturverlauf in der Regel sprunghaft ändert. Dadurch, dass das Remapping auf der relativen Lauflänge beruht und folglich für kontinuierliche Konturverläufe optimiert ist, wird erwartet, dass sich einzelne Netzlinien überschneiden. Ebenso ergeben sich Probleme, wenn Naben- und Gehäusekontur lokal signifikant unterschiedliche Lauflängen aufweisen (beispielsweise in *Kap. 6.5*). Eine Lösungsmöglichkeit für diese Probleme stellt die Implementierung einer transfiniten Interpolation unter Vorgabe mehrerer Stützlinien zur automatischen Netzgenerierung dar.

5.3.4 Einlesen eines vorhandenen S2M-Netzes - *readS2m*

Name	readS2m		
Aufruf	-rS2m		
Optionen	-lS2	Name der Tecplot-Datei	obligatorisch
	-xieta	index/length	obligatorisch

Tabelle 5.4: Übersicht über Aufruf und Optionen des *readS2m*-Tasks

Wie bereits erwähnt, steht am Anfang des Gesamtprozesses die Aufgabe ein temporäres S2M-Netz für die anschließende Verschneidung der 3D-Konfiguration in *POST*-Strukturen bereitzustellen. Eine mögliche Vorgehensweise ist hierbei die in den vorangegangen Abschnitten vorgestellte automatische Generierung. Falls jedoch bereits ein S2M-Netz mit der entsprechenden Auflösung zur Verfügung steht, kann dieses eingelesen und auf *POST*-Datenstrukturen konvertiert werden. Diese Funktionalität wird durch den *readS2m*-Task (siehe *Tab. 5.4*) bereitgestellt.
Zunächst ist vom Benutzer unter Verwendung der Option „-lS2“ der Name der punktbasierten ASCII-Datei anzugeben, welche das S2M-Netz enthält. Dieses wird dann mit Hilfe der Routine *readDataSetFromTec* von allen Prozessen eingelesen. Über die Option „-xieta“ wird vorgegeben, ob die zur anschließenden Schnitterstellung benötigte ξ- und η-Definition index- oder längenbasiert erfolgen soll (siehe auch *Kap. 5.3.3*).
Im Anschluss an das Einlesen erfolgt dann die Überprüfung des gegebenen Netzes. Folgende Anforderungen sind von diesem zu erfüllen :

- Netztopologie strukturiert
- Definition als ein Block bzw. eine Zone
- 2D-Netz mit IJ-Indizierung
- I-Index entspricht Meridionalrichtung (aufsteigend vom Einlass zum Auslass)
- J-Index entspricht Normalrichtung (aufsteigend von der Nabe zum Gehäuse)
- Netzabmessungen in Meridionalrichtung größer oder gleich derjenigen der 3D-Konfiguration (bei Verschneidung der Gesamtkonfiguration)
- Netzabmessungen in Normalrichtung gleich derjenigen der 3D-Konfiguration (bei Verschneidung der Gesamtkonfiguration)

Zu bemerken ist, dass nicht alle Einschränkungen abgefragt werden. Da der folgende *intersect3DS2MPlane*-Task auf einem zylindrisch definierten Schnittnetz basiert, wird für kartesische Netze zudem eine Koordinatentransformation durchgeführt.
Die Anforderung, dass der I-Index der Meridionalrichtung entspricht, wird für die Verschneidung des Gesamtnetzes folgendermaßen überprüft: Jeder Prozess, auf welchem sich ein Inlet-Block der 3D-Konfiguration befindet, berechnet dessen mittleren Normalenvektor in *xr*-Koordinaten. Ebenso wird für das gegebene S2M-Netz der mittlere „Inlet-Normalenvektor“ bzw. die mittlere Startrichtung der Netzlinien ermittelt. Dabei wird davon ausgegangen, dass der I-Index der Meridional- und der J-Index der Normalrichtung entspricht sowie der minimale I-Wert am Inlet vorliegt. Für den Normalenvektor ergibt sich somit:

$$\vec{n}_{S2M-Inlet} = \begin{pmatrix} \sum_J (x_{I=1,J} - x_{I=0,J}) \\ \sum_J (r_{I=1,J} - r_{I=0,J}) \\ 0 \end{pmatrix}$$

Es wird davon ausgegangen, dass die Netzlinien in Meridionalrichtung ungefähr orthogonal zum Inlet starten. Nach Normierung der Normalenvektoren wird also überprüft, ob der Winkel zwischen den beiden kleiner als 20° ist. Wenn dies nicht erfüllt ist, sendet der prüfende Prozess eine Warnung an den Benutzer, dass eventuell die Indexrichtungen des geladenen Netzes vertauscht sind bzw. nicht den Anforderungen entsprechen.
Eine automatische Korrektur des Netzes kann nicht durchgeführt werden, da der Benutzer nicht zwangsläufig das S2M-Netz der Gesamtkonfiguration bereitstellt, sondern auch die Vernetzung von Teilbereichen vorgeben kann, beispielsweise einer Kavität. Für diese ist der Abgleich mit dem Inlet-Normalenvektor der 3D-Konfiguration nicht sinnvoll. Auch für solche Teilnetze gilt grundsätzlich, dass die I-Indexrichtung der Strömungsrichtung entspricht und die J-Indexrichtung der Richtung von der Nabe zum Gehäuse. Die Option der Vernetzung von Teilbereichen wird jedoch von der momentanen Implementierung der Turbomaschinen-Auswertung nicht unterstützt.
Falls das Netz die gestellten Anforderungen erfüllt, werden die ξ- und η-Werte der Netzknoten bestimmt. Dies erfolgt dabei auf dieselbe Weise wie für den *generateS2intersecMesh*-Task. Schließlich wird das S2M-Netz, inklusive ξ- und η-Belegung, als neuer Block mit dem Zonennamen „S2MSurface“ gespeichert. Beim Blocktyp handelt es sich dabei um *S2MPLANE_BLOCK*.

5.3.5 Bereitstellung geometrischer Informationen - *collectGeometricalInfo*

Name	collectGeometricalInfo		
Aufruf	-cGeomI		
Optionen	-task	geo/i3DS2	obligatorisch

Tabelle 5.5: Übersicht über Aufruf und Optionen des *collectGeometricalInfo*-Tasks

Im Folgenden wird der *collectGeometricalInfo*-Task (siehe *Tab. 5.5*) vorgestellt. Bei diesem handelt es sich um einen Hilfstask, welcher die von den Tasks *generateExtremalOutline* (*Kap. 5.3.2*) sowie *intersect3DS2MPlane* (*Kap. 5.4*) benötigten geometrischen Größen mittels einer MPI-Kommunikation bereitstellt.
Für den *generateExtremalOutline*-Task (Option „-geo“) werden hierbei die Koordinaten des Minimal- und Maximalpunkts am In- und Outlet der Konfiguration in der *xr*-Ebene ermittelt. Die Berechnung erfolgt dabei zunächst blockweise für die lokal auf einem Prozess vorhandenen In- und Outlet-Blöcke. In der anschließenden Kommunikation wird dann der Minimal- und der Maximalpunkt der In- bzw. Outlet-Familie ermittelt. Diese setzen sich im Allgemeinen aus mehreren Blöcken zusammen. Die resultierenden Koordinaten werden schließlich in der IntegralValues-Struktur der jeweiligen Familie mit dem „Mittelungstyp“ *S2MGEN_GEOMETRY_QUANTITIES* abgelegt.
Die IntegralValues-Struktur existiert dabei sowohl für Blöcke als auch für Familien. Sie enthält, wie der Name bereits sagt, integrale Werte beliebiger Variablen für die jeweilige Struktur. Um welche Art von Größen es sich handelt, ist über den „Mittelungstyp“ definiert. Daraus wird ersichtlich, dass diese Struktur in erster Linie zur Speicherung gemittelter Strömungsgrößen verwendet wird. Daneben existiert zusätzlich die BandIntegralValues-Struktur. Diese Struktur enthält wiederum die IntegralValues-Strukturen der einzelnen Bänder eines Blocks oder einer Familie, sofern diese über eine Bandstruktur verfügen.
Für die Option „-i3DS2“ hingegen werden die ξ-Werte des In- und Outlets der Konfiguration berechnet. ξ stellt dabei die dimensionslose Meridionalkoordinate des S2M-Schnittnetzes dar (siehe *Kap. 5.3.3*). Dadurch, dass der Anwender bereits vorhandene S2M-Netze zur Verschneidung laden kann, deren Abmessungen in Meridionalrichtung in der Regel größer oder gleich groß wie diejenigen der 3D-Konfiguration sind, muss ξ_{inlet} und ξ_{outlet} bestimmt werden, um zu gewährleisten, dass die erste bzw. letzte Schnittebene unmittelbar an der Eintritts- bzw. Austrittsfläche liegt. ξ_{inlet} bezeichnet dabei beispielsweise die Lage der in die Ebene des S2M-Schnittnetzes gedrehten Inlet-Panelfamilie bezüglich des Inlets des S2M-Netzes. Um sicherzustellen, dass die Schnitte komplett innerhalb des 3D-Rechengebiets liegen, wird zunächst der maximale Abstand zwischen dem Inlet der geladenen S2M-Fläche sowie demjenigen der 3D-Konfiguration bestimmt. Der maximale Abstand wird dabei in Richtung der Normalen der S2M-Inletfläche gemessen. Folglich wird der Maximalwert der orthogonalen Projektion der Verbindungsvektoren aller S2M-Inlet-Netzpunkte zu allen 3D-Inlet-Netzpunkte gesucht. Anschließend wird mit Hilfe dieses Abstands die ξ-Koordinate des 3D-Inlets, ξ_{inlet}, interpoliert. Dieselbe Methode wird zur Bestimmung von ξ_{outlet} angewandt. Nachdem die Berechnung von ξ_{inlet} und ξ_{outlet} für alle in Frage kommenden Blöcke des jeweiligen Prozesses durchgeführt worden ist, erfolgt im Anschluss eine Kommunikation zur Ermittlung des maximalen Wertes am In- und Outlet. Das Resultat wird wiederum in der IntegralValues-Struktur der entsprechenden Familie mit dem Typ *S2MGEN_GEOMETRY_QUANTITIES* gespeichert.
Die Bereitstellung der geometrischen Informationen erfolgt dabei nicht innerhalb der Tasks, welche diese benötigen, sondern wird von einem zusätzlichen Task durchgeführt, da die einzelnen Prozesse im Allgemeinen nicht auf alle Inlet- und Outlet-Blöcke des Datensets bzw. die benötigten Netzinformationen, zugreifen können. Zwar ist die S2M-

Fläche für alle Prozesse verfügbar, jedoch sind die entsprechenden In- und Outlet-Blöcke im Normalfall unterschiedlichen Prozessen zugewiesen. Zu beachten ist zudem, dass die Inlet- und Outlet- Panelfamilien in mehrere Einzelblöcke aufgespalten sein können, welche die benötigten Netzinformationen beinhalten.

5.4 Verschneidung der 3D-Konfiguration mit dem S2M-Netz - *intersect3DS2MPlane*

Name	intersect3DS2MPlane		
Aufruf	-i3DS2		
Optionen	-nCuts	Anzahl ξ-Schnitte	optional
		101	default
	-nBands	Anzahl Bänder	optional
		71	default

Tabelle 5.6: Übersicht über Aufruf und Optionen des *intersect3DS2MPlane*-Tasks

Die Aufgabe des *intersect3DS2MPlane*-Tasks (siehe *Tab. 5.6*) besteht darin, eine Verschneidung der zu analysierenden 3D-Konfiguration mit dem temporären S2M-Netz durchzuführen, welches von den vorangegangenen Tasks bereitgestellt wurde. Für die resultierenden zweidimensionalen ξ-Schnittebenen, die sogenannten Analyseflächen, wird anschließend eine Umfangsmittelung durchgeführt und auf Basis dieser Ergebnisse das fertige S2M-Netz aufgebaut.
Die Voraussetzungen dieses Tasks sind:

- strukturiertes S2M-Schnittnetz mit gegebener ξ- und η-Verteilung
- Abstand Inlet_{S2M} - Inlet_{3D} als ξ-Wert (ξ_{inlet})
- Abstand Outlet_{S2M} - Outlet_{3D} als ξ-Wert (ξ_{outlet})

Da das temporäre S2M-Netz von allen Prozessen generiert bzw. eingelesen wird, steht dieses allen zur Verfügung. Die ebenfalls benötigten Werte ξ_{inlet} und ξ_{outlet} werden vom *collectGeometricalInfo*-Task mit der Option „-task i3DS2“ bereitgestellt und befinden sich in den IntegralValues-Strukturen der Inlet- bzw. Outlet-Familie (*Kap. 5.3.5*). Auf diese Werte kann somit ebenfalls jeder Prozess zugreifen. Denn *POST* arbeitet blockparallel, das heißt ein Prozess kann zwar nicht auf die vollständigen Informationen aller Blöcke, aber auf diejenigen aller Familien des Datensets zugreifen.
Im ersten Schritt wird aus der Gruppe der lokalen Blöcke des jeweiligen Prozesses derjenige gesucht, welcher das S2M-Netz mit der benötigten ξ- und η-Verteilung enthält. Die Koordinate ξ entspricht dabei der auf den Bereich von 0 - 100 normierten dimensionslosen Meridionalkoordinate und η der auf den selben Bereich normierten dimensionslosen Normalkoordinate. Nachdem der S2M-Block gefunden worden ist, werden anschließend alle benötigten Informationen kopiert. Dies ist nötig, weil während des

Verschneidungs-Prozesses zusätzliche Blöcke erzeugt und abgespeichert werden. Somit wird das ursprüngliche Block-Array des Datensets mit Hilfe der *realloc*-Funktion dynamisch erweitert. Dies kann dazu führen, dass der S2M-Block an eine andere Stelle im Speicher verschoben wird und der ursprünglich gesetzte Zeiger somit auf eine falsche Adresse zeigt.
Im nächsten Schritt werden die Schnitt- und Bandpositionen bestimmt. Falls der Benutzer über „-nCuts“ bzw. „-nBands“ keine Vorgaben gemacht hat, gelten hierbei folgende Defaultwerte:

- Anzahl der Bänder: 71
- Anzahl der Schnittpositionen: 101

Die Meridionalschnitte (ξ = konstant) sind bezüglich der ξ-Koordinate äquidistant verteilt, wobei die erste und letzte Position über ξ_{inlet} und ξ_{outlet} festgelegt wird. Zu diesen beiden Werten wird jeweils ein Sicherheitsfaktor addiert bzw. abgezogen, wodurch gewährleistet wird, dass die erste und die letzte Schnittfläche unmittelbar am Inlet bzw. Outlet liegen, jedoch vollständig innerhalb des Rechengebiets der 3D-Konfiguration. Andernfalls ist aufgrund numerischer Ungenauigkeiten der Verschneidung damit zu rechnen, dass ein Schnitt, welcher eigentlich exakt mit der jeweiligen Berandung zusammenfällt, keine durchgängige Schnittebene ergibt und einige Bänder eventuell nicht existieren. Für den Sicherheitsfaktor gilt:

$$\text{Sicherheitsfaktor} = \frac{\xi_{outlet} - \xi_{inlet}}{1000}$$

Für die Bandpositionen (η = konstant) ist ebenfalls eine äquidistante Verteilung bezüglich der η-Koordinate implementiert mit Ausnahme des ersten und des letzten Bands. Diese sind nur halb so breit wie die übrigen Bänder. Zukünftig wird es als weitere Option möglich sein, über eine Input-Datei die Anzahl sowie die Verteilung der Bänder und der Schnittpositionen vorzugeben.
Anzumerken ist, dass durch eine äquidistante Verteilung der Schnitte nicht notwendigerweise geometrisch äquidistant verteilte Schnittflächen generiert werden. Denn deren Verteilung ist neben den ξ-Schnittpositionen in erster Linie über die ξ-Definition des S2M-Netzes (längen- oder indexbasiert) bestimmt (siehe *Kap. 5.3.3*).
Nach Bereitstellung der benötigten Informationen, erfolgt die Verschneidung aller 3D-Blöcke der Konfiguration. Diese wird parallel auf mehreren Prozessen im Rahmen einer Blockschleife über alle lokalen 3D-Blöcke ausgeführt. Die resultierenden, unstrukturiert vernetzten Schnitt- bzw. Analyseflächen werden dann als jeweils ein neuer Block unter dem Namen „AnalysisSurface“ mit dem Typ *ANALYSIS_BLOCK* gespeichert. Zudem wird für jeden Meridionalschnitt eine eigene Analysefamilie mit einem Familiennamen der Form „ISO_XI_CUT_50.000000“ generiert. Zu dieser Familie gehören alle 2D-Blöcke einer fixen ξ-Schnittposition, welche aus der Verschneidung ihres jeweiligen 3D-Elternblocks für diesen ξ-Wert entstanden sind. Aus der Verschneidung eines 3D-Blocks kann dabei für eine ξ-Schnittebene nur ein 2D-Block entstehen.
Probleme treten auf, wenn ein Schnitt über ein Interface läuft und die einzelnen Blöcke der Familie folglich in verschiedenen Blockgruppen liegen, da die Analysefamilie nur zu jeweils einer Blockgruppenfamilie gehören kann. Einzelne 2D-Blöcke der Analysefamilie gehören somit nicht mehr zur selben Blockgruppe wie ihr 3D-Elternblock. Im Rahmen

der Mittelung wird dann für einige Blöcke auf die Daten der falschen Blockgruppenfamilie zurückgegriffen. Dies kann in unphysikalischen Geschwindigkeitssprüngen über Interfaces resultieren. Diese Problematik wurde dadurch behoben, dass jedem neu erstellten 2D-Block die Blockgruppenfamilie seines 3D-Elternblocks zugewiesen wird. Jedoch ist auf diese Weise das Problem nicht vollständig gelöst. Dies wird im Rahmen der Validierung in *Kap. 6.1* behandelt.
Nach Beendigung des Tasks werden im Anschluss die neu erstellten Blöcke und Familien kommuniziert und synchronisiert. Das Resultat des *intersect3DS2MPlane*-Tasks zeigen *Abb. 5.16* , beispielhaft für zehn äquidistant verteilte Schnittebenen, sowie *Abb. 5.17* . Abschließend wird noch auf vorhandene Probleme des Tasks eingegangen. Es ist möglich, dass für bestimmte ξ-Schnittpositionen das S2M-Netz und das 3D-Netz des zu verschneidenden Blocks ungünstig zueinander liegen. Aufgrund der Interpolation des ξ-Schnittverlaufs auf die 3D-Konfiguration können dann Schwingungen in den einzelnen Schnittflächen auftreten (*Abb. 5.18*). Dieses Phänomen scheint ausschließlich für automatisch generierte S2M-Netze aufzutreten, da sich bei diesen die Abstände der einzelnen Netzlinien lokal stark unterscheiden. Bei Verwendung ungefähr äquidistant vernetzter S2M-Gitter verschwindet das Problem jedoch. Ebenfalls tritt es nur bei Verwendung der indexbasierten ξ-Definition auf und nicht bei der längenbasierten Variante. Es wird deshalb davon ausgegangen, dass eventuell eine „falsche Gewichtung“ der Netzknoten die Schwingungen hervorruft. Denn egal wie groß der geometrische Abstand zweier Vertices ist, der ξ-Abstand ist bei der indexbasierten Variante äquidistant.
Solange diese Schwingungen jedoch nicht allzu stark sind, sollten diese keine weiteren negativen Auswirkungen auf die Ergebnisqualität haben. Eine Möglichkeit die Schwingungen zu vermeiden ist die Konvertierung des automatisch generierten S2M-Schnittnetzes auf ein äquidistantes Gitter. Diese Idee wurde jedoch nicht umgesetzt, da somit die Information über die ursprüngliche Punktverteilung der Naben- und Gehäuse-Panelfamilien verloren gehen würde. Über diese Punktverteilung jedoch wird die geometrische Lage der Schnittflächen gesteuert. Zudem soll ja mit der indexbasierten ξ-Variante erreicht werden, dass Bereiche höherer Punktdichte stärker gewichtet werden, ergo, dass dort mehr Schnittflächen liegen. Für ein äquidistantes S2M-Schnittnetz aber ist ausschließlich eine äquidistante Verteilung der Schnittebenen möglich.

5.5 Implementierung der S2-Mittelung

Eine der Kernaufgaben der globalen Turbomaschinen-Auswertung stellt die Implementierung der S2-Mittelung dar, also der Mittelung in Umfangsrichtung. Angeboten werden hierbei drei verschiedene Varianten, nämlich die Fluss-, die Massen- und die Flächenmittelung. Die Routinen zur Ermittlung der panelintegralen und bandintegralen Werte sind dabei in der Lage sowohl auf strukturierten als auch auf unstrukturierten Gittern zu arbeiten. Es wird also keine Unterscheidung zwischen Interfaces, welche meist eine strukturierte Topologie aufweisen, und Analyseflächen, welche grundsätzlich unstrukturiert vernetzt sind, gemacht. Die Implementierung der Mittelung setzt sich aus den fünf folgenden, teilweise aufeinander aufbauenden Tasks zusammen:

- calcConservativeVariables3D (*Kap. 5.5.1*)

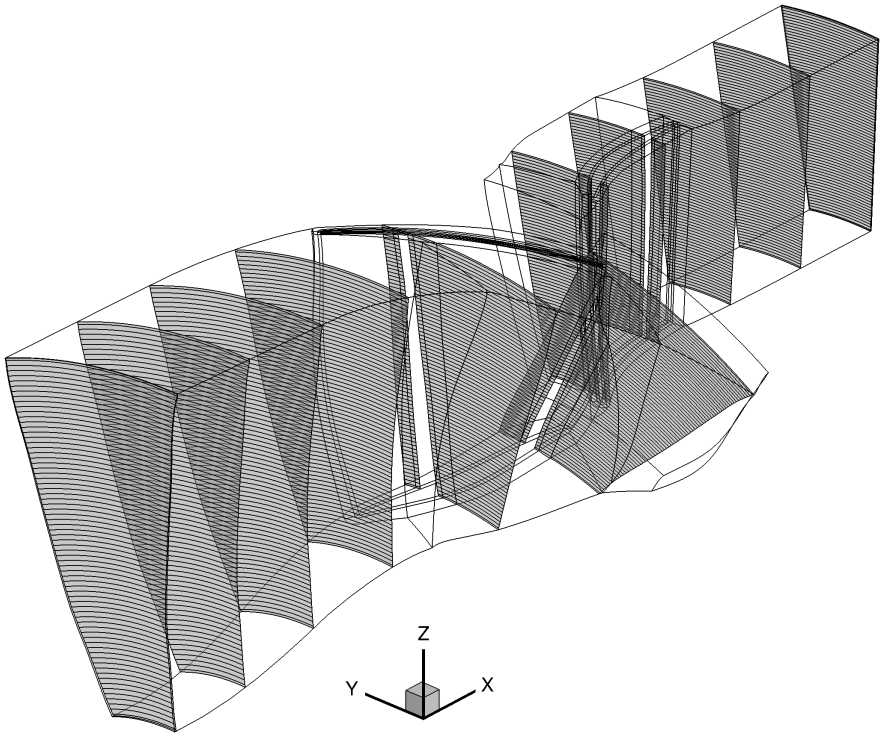


Abb. 5.16: Verschneidung der 3D-Konfiguration mit geometrisch äquidistanter Verteilung der ξ-Schnittebenen (Testfall THD R1)

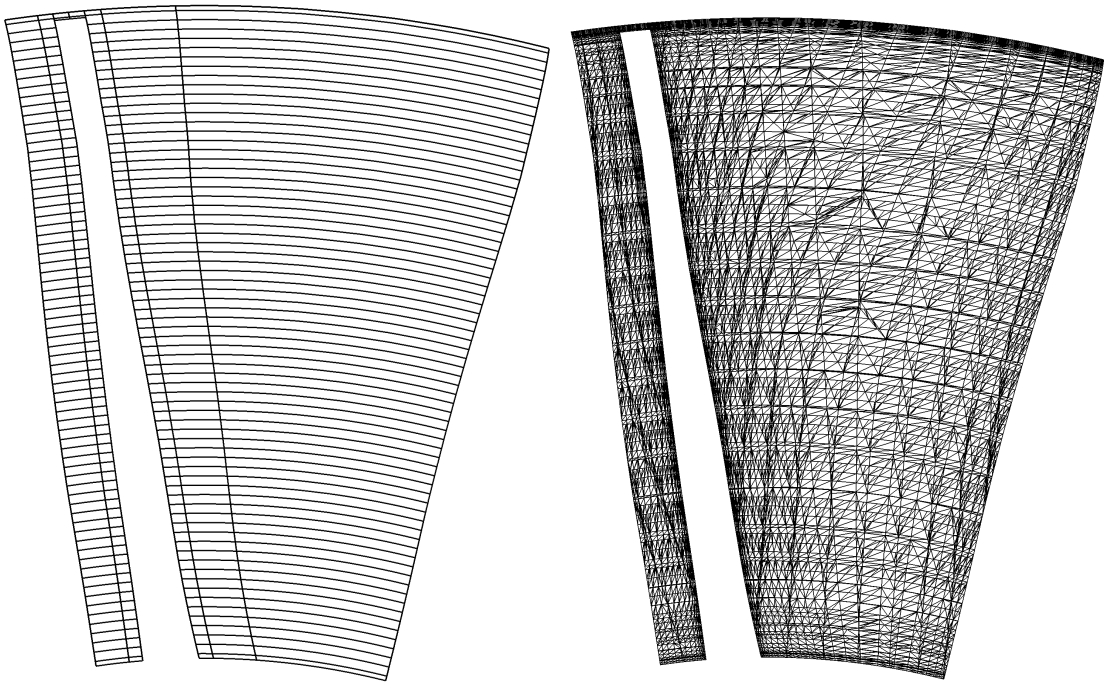


Abb. 5.17: Bandstruktur und Vernetzung einer Analysefamilie (fünfte von vorn aus *Abb. 5.16*)

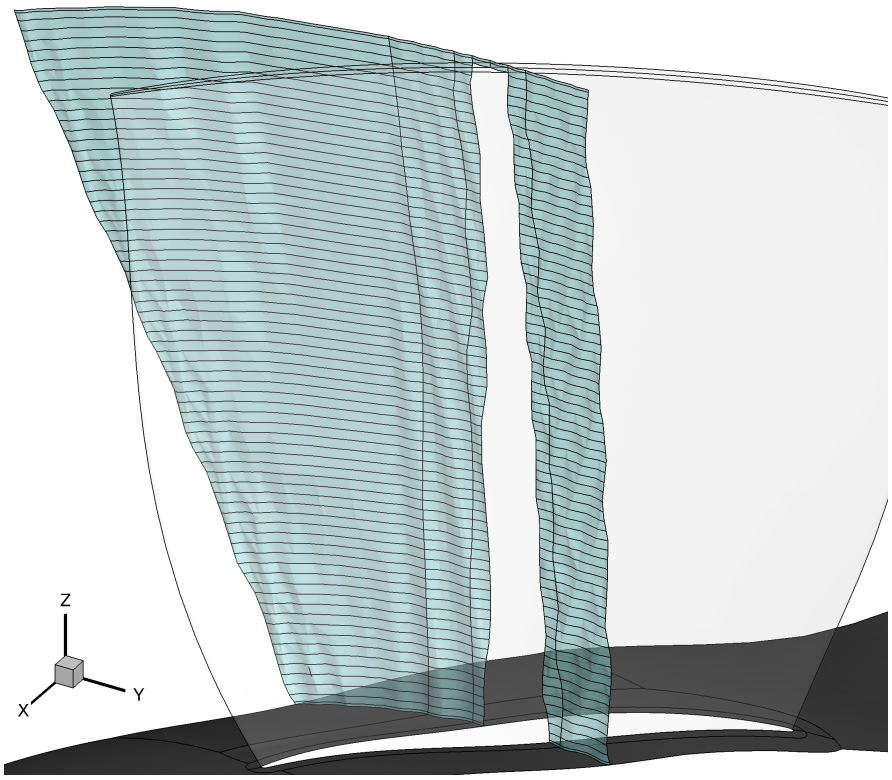


Abb. 5.18: Schwingungen im Verlauf der Analysefamilie bei $\xi = 41.22$ - indexbasierte ξ-Definition (Testfall THD R1)

- averagingByArea (*Kap. 5.5.2*)
- averagingByFlux (*Kap. 5.5.2*)
- averagingByMassFlow (*Kap. 5.5.2*)
- averagingInversion (*Kap. 5.5.3*)

5.5.1 Berechnung der konservativen Variablen für das 3D-Feld - *calcConservativeVariables3D*

Name	calcConservativeVariables3D
Aufruf	-cconv
Optionen	-

Tabelle 5.7: Übersicht über Aufruf und Optionen des *calcConservative-Variables3D*-Tasks

Vor der Implementierung der Mittelungsroutinen war zunächst die Frage zu klären, auf welchem Satz von Variablen die Mittelung basieren soll, um den mittleren Strömungszustand physikalisch möglichst korrekt zu beschreiben. Dieser Ausgangssatz von Variablen wird dann, falls nicht direkt von *TRACE* geliefert, für jede Zelle der 3D-Blöcke berechnet und anschließend mittels des *interpolate2D*-Tasks auf die 2D-Flächen interpoliert.

Alle weiteren zu ermittelnden Strömungsgrößen werden dann in den folgenden Tasks ausgehend von den interpolierten Basisvariablen für die jeweilige 2D-Fläche abgeleitet. Die verschiedenen diskutierten Ausgangsvariablensätze sind dabei folgende:

- primitive Variablen
- Fluss aller benötigten Variablen
- konservative Variablen im Absolut- und Relativsystem
- konservative Variablen nur im Relativsystem

Die erste Möglichkeit, die primitiven Variablen ρ, p und $\vec{v}$ als Basis zu benutzen, hat den Vorteil, dass diese bereits von *TRACE* bereitgestellt werden und somit keine Ableitung weiterer Strömungsgrößen für das 3D-Feld notwendig ist. Diese Vorgehensweise wurde jedoch verworfen, da sie den Erhalt von Masse, Impuls und Energie nicht gewährleisten kann. Die Mittelung wäre somit nicht konservativ.
Eine weitere Option für die Massenmittelung stellt die Berechnung der Flüsse aller benötigten physikalischen Größen für das 3D-Feld nach folgendem Schema dar:

$$\rho\,\vec{v}\cdot \mathit{Variable} = \begin{pmatrix} \rho\, v_x \cdot \mathit{Variable} \\ \rho\, v_r \cdot \mathit{Variable} \\ \rho\, v_{\theta,abs/rel} \cdot \mathit{Variable} \end{pmatrix} \tag{5.4}$$

Diese Flüsse von beliebigen Strömungsgrößen werden anschließend auf die 2D-Flächen interpoliert und jeder Beitrag mit der jeweiligen Zellfläche $\vec{A}$ multipliziert, um letztendlich $\dot{m} \cdot \mathit{Variable}$ zu erhalten. Zur Mittelwertbildung werden dann die Beiträge der einzelnen Zellen aufsummiert und durch den Gesamtmassenstrom geteilt. Der Vorteil dieser Methode liegt in einer Verringerung des Interpolationsfehlers. Denn anstatt ρ sowie die Geschwindigkeitskomponenten $\vec{v}$ einzeln auf die Flächen zu interpolieren, darauf basierend anschließend den Massenstrom sowie die zu gewichtenden Strömungsgrößen abzuleiten und somit die einzelnen Interpolationsfehler aufzusummieren, wird der komplette Term $\rho\,\vec{v}\cdot \mathit{Variable}$ interpoliert und der Fehler folglich verringert.
Die Nachteile jedoch sind, dass die Mittelung erstens nicht konservativ ist und sich zweitens ein inkonsistenter Strömungszustand ergeben kann. Darüber hinaus kann diese Methode aufgrund der Druckterme im konvektiven Flussvektor nicht für die Flussmittelung benutzt werden, da in den Impulsflusstermen die Zellnormale $\vec{A}$ nicht ausgeklammert werden kann. Somit ist es mit diesem Ansatz nicht möglich den aufsummierten konvektiven Flussvektor zu ermitteln.
Um also eine einheitliche Vorgehensweise für Fluss-, Massen- und Flächenmittelung zu ermöglichen, wurde letztendlich entschieden die konservativen Variablen im Relativsystem als Basis für die S2-Mittelung zu verwenden. Die Vorteile hierbei sind, dass einerseits die Vorgehensweise von *TRACE* nachgeahmt wird, was die Vergleichbarkeit erhöht sowie den Wartungsaufwand reduziert, und andererseits eine konservative Mittelung gewährleistet ist. Zur Vermeidung eines inkonsistenten Strömungszustands werden die konservativen Variablen nur im Relativsystem berechnet. Ausgehend von diesen werden dann alle weiteren Größen für die 2D-Fläche bzw. deren Bänder abgeleitet.
Die gemeinsame Basis aller drei implementierten Mittelungsvarianten bilden somit die konservativen Variablen, bei welchen es sich nach [Bla01] um die Komponenten des

Vektors $(\rho, \rho v_x, \rho v_r, \rho v_\theta, \rho e_t)^T$ handelt. Diese werden vom *calcConservativeVariables3D*-Task (siehe *Tab. 5.7*) bereitgestellt. Der Task ist als Peasant-Task implementiert und berechnet für jede Zelle aller 3D-Blöcke, inklusive der Geisterzellen, die konservativen Variablen. In Anlehnung an *TRACE* werden diese in Zylinderkoordinaten im Relativsystem ermittelt.
Dadurch, dass die von *TRACE* gelieferten Simulationsergebnisse in kartesischen Koordinaten im Relativsystem vorliegen, muss zur Ermittlung der Geschwindigkeitskomponenten im Zylinderkoordinatensystem zunächst folgende Transformation durchgeführt werden:

$$\begin{pmatrix} v_x \\ v_r \\ v_{\theta,rel} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & sin\theta & cos\theta \\ 0 & cos\theta & -sin\theta \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} \tag{5.5}$$

Der jeweilige Drehwinkel θ lässt sich hierbei ebenso wie der Radius aus der Zellgeometrie ermitteln. Für innere Zellen werden die entsprechenden geometrischen Größen als arithmetisches Mittel der Werte an den Zell-Eckpunkten bestimmt, für Geisterzellen hingegen werden sie basierend auf den Koordinaten des Zell-Mittelpunkts abgeleitet.
Die fünfte Komponente des Vektors der konservativen Variablen, das Produkt aus Dichte und Totalenergie bzw. der Energiefluss, wird nach folgender Gleichung berechnet, welche auf der Annahme eines perfekten Gases basiert ([Bla01]):

$$\rho e_t = \frac{p}{\gamma - 1} + \frac{1}{2}\rho \left(v_x^2 + v_r^2 + v_{\theta,rel}^2 - (\omega \cdot r)^2 \right) \tag{5.6}$$

Nach Berechnung der konservativen Variablen auf allen 3D-Blöcken wird der *interpolate2D*-Task verwendet, um diese beziehungsweise alle vorhandenen Variablen des 3D-Felds auf jede der Boundary-Flächen sowie auf jede der Analyseflächen der Konfiguration, also auf alle 2D-Blöcke, zu interpolieren.

5.5.2 Vorstellung der Kernroutinen der Flussmittelung, der Massenmittelung und der Flächenmittelung

Im Folgenden wird die Implementierung der drei Tasks *averagingByFlux* (*Tab. 5.8*) zur Fluss-, *averagingByMassFlow* (*Tab. 5.9*) zur Massen- sowie *averagingByArea* (*Tab. 5.10*) zur Flächenmittelung vorgestellt. Zwar basieren alle drei Mittelungsvarianten auf einer ähnlichen Funktionsstruktur, jedoch werden dem Benutzer jeweils unterschiedliche Steuerungsoptionen angeboten. Aus diesem Grund wurde die Aufspaltung in drei voneinander unabhängige Tasks gewählt und umgesetzt. Zudem kann vom Benutzer so genau bestimmt werden welche Mittelungsart durchzuführen ist.
Anzumerken ist, dass für einen bestimmten Mittelungstyp alle physikalischen Größen, soweit möglich, nach der vom Anwender gewählten Variante gemittelt werden, auch wenn diese Vorgehensweise nicht immer physikalisch sinnvoll ist. So ist nach [CH06] für extensive Zustandsgrößen sowie für den Totaldruck eine Massenmittelung sinnvoll bzw. üblich, für den statischen Druck jedoch wird eine Flächenmittelung empfohlen.
In den Mittelungs-Kernroutinen erfolgt zunächst die blockweise Aufsummierung der Beiträge der 2D-Zellen für die Bänder und die Gesamtfläche. Dies erfolgt jeweils für die lokal vorhandenen Blöcke parallel auf mehreren Prozessen. Die Summen werden im

anschließenden *averagingInversion*-Task an die zugehörigen Familien kommuniziert, für welche dann die Mittelwerte der Strömungsgrößen berechnet werden.

Name	averagingByFlux
Aufruf	-avgf
Optionen	-

Tabelle 5.8: Übersicht über Aufruf und Optionen des *averagingByFlux*-Tasks

Name	averagingByMassFlow		
Aufruf	-avgm		
Optionen	-option	fields/gta	optional
		fields + gta kombiniert	default
	-signed	-	optional
		Gewichtung über $\|\dot{m}\|$	default

Tabelle 5.9: Übersicht über Aufruf und Optionen des *averagingByMassFlow*-Tasks

Name	averagingByArea		
Aufruf	-avga		
Optionen	-option	fields/gta	optional
		fields + gta kombiniert	default

Tabelle 5.10: Übersicht über Aufruf und Optionen des *averagingByArea*-Tasks

Zunächst werden die jeweils vorgebbaren Optionen vorgestellt. Bei der Massen- und Flächenmittelung kann der Anwender über das Schlüsselwort „-option“ angeben, welcher Satz von Variablen gemittelt werden soll. Zur Auswahl stehen hierbei:

- „fields“ → alle auf den 2D-Strömungsflächen vorhandenen Variablen
- „gta“ → alle für die globale Turbomaschinen-Auswertung benötigten Variablen
- keine Angabe → Option „fields“ und „gta“ kombiniert

Für die gta-Variante werden ausschließlich die den Strömungszustand beschreibenden Variablen, welche für die globale Turbomaschinen-Auswertung benötigt werden, bereitgestellt (siehe *Tab. 5.12*). Falls seitens des Benutzers keine Angaben gemacht werden, sind standardmäßig beide Optionen aktiviert; das heißt, es werden integrale Werte sowohl für die gta-Größen als auch für alle vorhanden Variablenfelder ermittelt. Bei dieser Variante wird mit Hilfe einer enumerator-Liste sichergestellt, dass keine Variable doppelt gemittelt wird. Für die Flussmittelung ist anzumerken, dass diese auf die gta-Funktionalität beschränkt bleibt, da für diese keine Mittelung beliebiger Felder möglich ist.
Als zusätzliche Option für den *averagingByMassFlow*-Task kann der Benutzer wählen, ob die Gewichtung der einzelnen Strömungsgrößen über einen vorzeichenbehafteten

Massenstrom $\dot{m}$ („-signed“) oder standardmäßig über den Betrag des Massenstroms $|\dot{m}|$ erfolgen soll.
Wie bereits erwähnt, basieren alle drei Mittelungsarten auf dem Vektor der konservativen Variablen im Relativsystem $(\rho, \rho v_x, \rho v_r, \rho v_\theta, \rho e_t)^T$, um einerseits die Erhaltung von Masse, Impuls und Energie zu gewährleisten und andererseits einen konsistenten Strömungszustand zu erhalten. Folglich wird zunächst überprüft, ob für alle zu behandelnden 2D-Blöcke diese Größen sowie im Fall der gta-Option die zusätzlich benötigten Turbulenzgrößen vorhanden sind. Da eine Flächenmittelung beliebiger Variablen ausschließlich auf der Zellgeometrie beruht, werden die konservativen Variablen für diese Variante nicht benötigt. Sind alle Basisvariablen gefunden, beginnt die eigentliche Kernroutine *averagingGeneral*, welche die Steuerungsfunktion der Mittelung darstellt und von allen Varianten verwendet wird. Bei dieser handelt es sich um eine Schleife über alle lokalen Blöcke des jeweiligen Prozesses. Innerhalb dieser werden für die Mittelung alle zweidimensionalen Strömungsflächen berücksichtigt, also sowohl Analyseflächen als auch alle Boundary-Blöcke, welche keine Wand-Randbedingung aufweisen, ergo durchströmte Flächen sind.
Für die einzelnen 2D-Blöcke erfolgt dann die Mittelung zunächst für die Bänder und anschließend für die Gesamtfläche. Anzumerken ist, dass der Begriff „Mittelung“ in diesem Zusammenhang zunächst nur die Ableitung und Summation der gewichteten Variablen bzw. Komponenten des konvektiven Flussvektors bezeichnet. Die letztendlich resultierenden integralen Werte werden, wie bereits erwähnt, im nachfolgenden *averagingInversion*-Task ermittelt.
Im ersten Schritt erfolgt die Speicherallokierung für die zu berechnenden panel- und bandintegralen Summen. Dabei wird für jede der auf dem 2D-Block existierenden Lösungen, mit Ausnahme von harmonischen, eine eigene Struktur für die integralen Werte angelegt. Diese IntegralValues-Struktur enthält neben den aufsummierten gewichteten Variablen den Mittelungstyp sowie den Index der jeweiligen Lösung. Für die integralen Bandwerte wird zudem der Index des zugehörigen Bands sowie geometrische Bandinformationen gespeichert, welche zum Aufbau des resultierenden S2M-Netzes im *buildFinalS2M*-Task benötigt werden (*Kap. 5.7*). Diese befinden sich in einer eigenen IntegralValues-Struktur mit dem „Mittelungstyp“ *BAND_DIMENSIONS*.
Nach Aufbau der Struktur für die integralen Werte beginnt im Anschluss die eigentliche Mittelung, das heißt die Aufsummierung der Beiträge aller 2D-Zellen eines Bands bzw. der gesamten Fläche des jeweiligen Blocks. Für diese Aufgabe existieren zwei Routinen, eine für die Flussmittelung und die andere sowohl für die Massen- als auch die Flächenmittelung. Zunächst soll die Vorgehensweise für Letztere betrachtet werden.
Falls die gta-Option aktiviert ist, werden ausgehend von den konservativen Variablen alle weiteren, für die später folgende Turbomaschinen-Analyse benötigten physikalischen Strömungsgrößen für jede 2D-Zelle abgeleitet sowie mit dem Gewicht derselben, also mit $\dot{m}$, $|\dot{m}|$ oder $|\vec{A}|$, multipliziert und aufsummiert. Hierbei werden zunächst die Geschwindigkeitskomponenten v_x, v_r und $v_{\theta,rel}$ sowie der statische Druck p, die Dichte ρ und die Totalenthalpie im Relativsystem $h_{t,rel}$ ermittelt. Ergo werden im ersten Schritt die primitiven Variablen sowie $h_{t,rel}$ aus den konservativen abgeleitet.

$$(\rho, \rho v_x, \rho v_r, \rho v_{\theta,rel}, \rho e_t)^T \quad \Longrightarrow \quad (p, \rho, v_x, v_r, v_{\theta,rel}, h_{t,rel})^T$$

Ausgehend von den primitiven Variablen werden dann alle weiteren Strömungsgrößen für die jeweilige Zelle berechnet und mit dem entsprechenden Gewicht multipliziert. Anzumerken ist, dass die implementierten Formeln auf der Annahme eines idealen Gases beruhen. Die für die gta-Variante bereitgestellten Größen können *Tab. 5.12* entnommen werden.
Ähnlich ist die Vorgehensweise zur Bestimmung integraler Werte für beliebige gegebene Variablen. Diese sind vom Benutzer zunächst für das 3D-Feld zu berechnen, beispielsweise im Rahmen eines *peasant*-Tasks, und mit dem *interpolate2D*-Task auf alle 2D-Flächen zu interpolieren, auf welchen dann die Flächen- oder Massenmittelung erfolgt. Der hauptsächliche Unterschied gegenüber der gta-Variante besteht darin, dass zur Mittelung beliebiger Variablen neben den Gewichtungsfaktoren, also $\dot{m}$ bzw. $|\dot{m}|$ oder $|\vec{A}|$, keine weiteren Strömungsgrößen abgeleitet werden müssen.
Abschließend werden die aufsummierten gewichteten Werte in die bereits vorher allokierten Strukturen des zugehörigen 2D-Blocks geschrieben. Das aufsummierte Gewicht wird dabei als eigene Variable behandelt.
Die zweite Mittelungs-Kernfunktion behandelt die Flussmittelung. Innerhalb dieser erfolgt in Anlehnung an *TRACE* die Ermittlung des aufsummierten konvektiven Flussvektors, welche wiederum auf dem Vektor der konservativen Variablen beruht. Der Flussvektor für eine Zelle lässt sich nach [Bla01] wie folgt schreiben:

$$\begin{pmatrix} \rho v_x A_x + \rho v_\theta A_\theta + \rho v_r A_r \\ (\rho v_x v_x + p) A_x + (\rho v_x v_\theta) A_\theta + (\rho v_x v_r) A_r \\ (\rho v_\theta v_x) A_x + (\rho v_\theta v_\theta + p) A_\theta + (\rho v_\theta v_r) A_r \\ (\rho v_r v_x) A_x + (\rho v_r v_\theta) A_\theta + (\rho v_r v_r + p) A_r \\ v_x(\rho e_t + p) A_x + v_\theta(\rho e_t + p) A_\theta + v_r(\rho e_t + p) A_r \end{pmatrix} =$$

$$\begin{pmatrix} \dot{m} \\ \dot{m} v_x + p A_x \\ \dot{m} v_\theta + p A_\theta \\ \dot{m} v_r + p A_r \\ \dot{m} \left(\frac{\gamma}{\gamma - 1} \frac{p}{\rho} + \frac{1}{2} \left(v_x^2 + v_\theta^2 + v_r^2 - (\omega r)^2 \right) \right) \end{pmatrix} \tag{5.7}$$

Anzumerken ist, dass die Verwendung der $x\theta r$-Reihenfolge anstelle der normalerweise gebräuchlichen $xr\theta$-Schreibweise *TRACE*-Konventionen folgt und der konvektive Flussvektor im Relativsystem ermittelt wird.
Für die Bänder bzw. die Gesamtfläche des aktuellen Blocks werden die einzelnen Beiträge zum konvektiven Flussvektor aufsummiert und ebenso die Flächennormalenvektoren in zylindrischen Koordinaten. Da sich aus dem konvektiven Flussvektor nicht alle benötigten Variablen ableiten lassen, werden zusätzlich benötigte geometrische Größen sowie Turbulenzgrößen flächengemittelt.
Zur Überprüfung, ob für das auszuwertende Band bzw. die auszuwertende Fläche Rückströmungen, zum Beispiel infolge einer Strömungsablösung, auftreten, welche die Resultate eventuell verfälschen, wird am Ende der Summationsroutinen der Betrag der Summe des vorzeichenbehafteten Massenstroms mit der Summe des Betrags des Massenstroms verglichen. Weichen die beiden Werte um mehr als 10% voneinander ab, so

wird dies dem Benutzer mitgeteilt. Diesem obliegt dann die Entscheidung, wie stark der Einfluss der Rückströmung ist bzw. ob das Ergebnis im Rahmen einer gewissen Toleranz noch physikalisch sinnvoll ist. Zum Vergleich können hierbei die flächengemittelten Werte herangezogen werden, da diese von Rückströmungen nicht beeinflusst werden. Zusammengefasst gilt somit für den Rückströmungsdetektor:

$$\left| \frac{\sum |\dot{m}| - |\sum \dot{m}|}{\sum |\dot{m}|} \right| > 10\,\% \qquad \Longrightarrow \qquad \text{signifikante Rückströmung}$$

5.5.3 Berechnung der resultierenden integralen Strömungsgrößen - *averagingInversion*-Task

Im Anschluss an die drei Tasks *averagingByFlux*, *averagingByMassFlow* sowie *averagingByArea* folgt der *averagingInversion*-Task (siehe *Tab. 5.11*). Falls mehrere verschiedene Mittelungsarten durchgeführt werden sollen, ist es sinnvoll zuerst alle Mittelungs-Kernroutinen auszuführen, bevor der *averagingInversion*-Task aufgerufen wird. Der Task setzt sich dabei aus zwei Kernaufgaben zusammen. Zunächst erfolgt im Rahmen einer MPI-Kommunikation die Übertragung der aufsummierten Werte von den Blöcken auf die zugehörigen Familien. Anschließend werden die gemittelten Strömungsgrößen sowohl für die einzelnen Bänder als auch die Gesamtfläche der Familie ermittelt.

Name	averagingInversion
Aufruf	-avgi
Optionen	-

Tabelle 5.11: Übersicht über Aufruf und Optionen des *averagingInversion*-Tasks

Zu Beginn des *averagingInversion*-Tasks erfolgt, wie bereits erwähnt, eine MPI-Kommunikation, um die gewichteten und aufsummierten zu mittelnden Variablen oder die aufsummierten Beiträge zum konvektiven Flussvektor von den IntegralValues-Strukturen der Blöcke auf diejenigen der zugehörigen Familien zu übertragen. Berücksichtigt werden dabei alle Boundary-Familien und alle Analysefamilien. Die Kommunikation der integralen Werte ist nötig, da die einzelnen Blöcke, welche zum Beispiel zu einer Analysefamilie bzw. einer ξ-Schnittebene gehören, im Allgemeinen auf verschiedenen Prozessen liegen. Bevor also die umfangsgemittelten Werte der Variablen berechnet werden können, müssen zuerst alle Beiträge der einzelnen Blöcke aufsummiert und an die jeweilige Familie übertragen werden.
Im ersten Schritt der Kommunikationsroutine werden die Strukturen für die integralen und bandintegralen Werte der Familien aufgebaut, auf welche jeder Prozess zugreifen kann. Diese Initialisierung der Familien geschieht mittels einer Blockschleife über alle lokalen Blöcke. Jede Familie wird dann jeweils durch den ersten ihrer zugehörigen Blöcke initialisiert. Initialisierung bedeutet dabei, dass die Informationen der IntegralValues-Strukturen des ersten Blocks auf die Familie übertragen werden. Ergo kennt jede Familie anschließend die Anzahl ihrer Bänder, welche Lösungen nach welcher Mittelungsart vorhanden sind und welche integralen Variablen berechnet werden.

Im Anschluss an die Blockschleife zur Initialisierung folgt eine Schleife über alle zu behandelnden Familien. Zunächst wird ermittelt, wie viele lokalen Blöcke des jeweiligen Prozesses zur aktuellen Familie gehören, also die Zahl der Beiträge, welche der Prozess liefern wird. Falls auf diesem kein Block der Familie existiert, muss der Prozess dennoch einen Beitrag senden, da die MPI-Kommunikation Informationen von allen Prozessen verlangt. In diesem Fall wird ein Beitragsvektor generiert, dessen Zahlenwerte alle mit dem operationsspezifischen neutralen Element belegt werden, beispielsweise Null für eine Summation oder Eins für eine Multiplikation.
Die Grundlage der implementierten Kommunikation bildet die Gleichsetzung des MPI-Reduktionsvektors mit einer IntegralValues-Struktur. Die aktuelle Familie definiert dabei welche integralen Werte jeweils zu kommunizieren sind. Die zur Familie gehörigen Blöcke schreiben dann ihren Beitrag für die aktuelle IntegralValues-Struktur in einen der Reduktionsvektoren. Dieser Beitrag entspricht einem Float-Array, welches die Werte aller in der Struktur enthaltenen zu mittelnden Variablen enthält.
Nach dem Sammeln der Beiträge des jeweiligen Prozesses erfolgt dann die eigentliche MPI-Kommunikation mit der Option *MPI_SUM*, also eine Summation der jeweiligen Beiträge. Anschließend werden die aufsummierten Variablen auf die aktuelle IntegralValues-Struktur der Familie übertragen.
Die Kommunikation der bandintegralen Werte erfolgt analog für alle IntegralValues-Strukturen jedes Bands. Zusätzlich werden die Geometriedaten des Bandes kommuniziert, also dessen Koordinaten der Ober- und Untergrenze in der xr-Ebene. Diese Kommunikation erfolgt mittels der Optionen *MPI_MIN* für die Unter- und *MPI_MAX* für die Obergrenze.
Bei Bändern ist zusätzlich zu beachten, dass zwar für jeden Block Speicher für die gleiche Anzahl an Bändern allokiert ist, jedoch nicht alle dieser Bänder existieren müssen. Denn in unvernetzten Bereichen der 3D-Konfiguration werden keine Bänder aufgebaut. Ein Beispiel hierfür liefert *Kap. 6.4* . Für nicht vorhandene Bänder werden in den vorangegangenen Mittelungs-Tasks identische IntegralValues-Strukturen aufgebaut wie für existierende Bänder, jedoch werden alle Variablen mit NaN belegt. Bevor ein lokaler Block also seine Werte auf den Reduktionsvektor überträgt, wird zuerst abgefragt ob diese NaN sind. In diesem Fall wird anstelle des jeweiligen Wertes das operationsspezifische neutrale Element übergeben.
Im Anschluss an die Kommunikation folgt eine zweite Familienschleife. Innerhalb dieser werden die resultierenden integralen Werte für jede der IntegralValues-Strukturen sowohl für die Bänder als auch die Gesamtfläche der Familie ermittelt. Dafür wird zunächst der jeweilige Mittelungstyp abgefragt. Falls flächen- oder massengemittelte Werte vorliegen, werden alle in dieser Struktur enthaltenen Variablen mit Ausnahme der Fläche, des Massenstroms sowie des Enthalpieflusses durch das ebenfalls abgespeicherte integrale Gesamtgewicht geteilt, um den resultierenden familienintegralen oder bandintegralen Wert der jeweiligen Strömungsgröße zu erhalten. Der reduzierte Massenstrom $\dot{m}_{red}$ wird daraufhin basierend auf den integralen Werten abgeleitet.
Falls flussgemittelte Werte berechnet werden, ist zur Ermittlung der resultierenden Größen eine Flussinvertierung erforderlich. Deren Implementierung orientiert sich dabei an der Vorgehensweise von *TRACE*. Im ersten Schritt werden die für die Flussinvertierung benötigten aufsummierten Werte aus der jeweiligen IntegralValues-Struktur ent-

nommen. Dabei handelt es sich um die fünf Komponenten des konvektiven Flussvektors, die flächengewichteten Geometrieparameter und Turbulenzgrößen, sowie die drei Komponenten des aufsummierten Flächennormalenvektors. Zur Ermittlung des mittleren Strömungszustands bzw. der diesen beschreibenden Variablen wird das Gleichungssystem, welches auf den aufsummierten Erhaltungsgleichungen für Masse, Impuls und Energie beruht, gelöst. Den Ausgangspunkt stellt der aufsummierte konvektive Flussvektor und der Normalenvektor der Fläche dar:

$$\vec{F} = (F_0, F_1, F_2, F_3, F_4)^T \qquad \text{und} \qquad \vec{A} = (A_x, A_\theta, A_r)^T \tag{5.8}$$

Folgende Abkürzungen werden verwendet:

$$A_{norm,sqr} = A_x^2 + A_\theta^2 + A_r^2 \tag{5.9}$$

$$FA_I = F_1 A_x + F_2 A_\theta + F_3 A_r \tag{5.10}$$

Nach Umformung ergeben sich schließlich die Formeln zur Bestimmung der primitiven Variablen:

$$p = \frac{FA_I + \sqrt{FA_I^2 + (\gamma^2 - 1) \cdot A_{norm,sqr} \cdot (F_1^2 + F_2^2 + F_3^2 - 2F_0 \cdot (F_4 + \frac{1}{2}(\omega r)^2 F_0))}}{(\gamma + 1) \cdot A_{norm,sqr}} \tag{5.11}$$

$$\rho = \frac{F_0^2}{FA_I - p \cdot A_{norm,sqr}} \tag{5.12}$$

$$v_x = \frac{F_1 - p \cdot A_x}{F_0} \tag{5.13}$$

$$v_{\theta,rel} = \frac{F_2 - p \cdot A_\theta}{F_0} \tag{5.14}$$

$$v_r = \frac{F_3 - p \cdot A_r}{F_0} \tag{5.15}$$

Ausgehend von den fünf primitiven Variablen p, ρ, v_x, $v_{\theta,rel}$ und v_r werden anschließend alle weiteren benötigten Strömungsgrößen nach der gta-Option ermittelt (siehe *Tab. 5.12*). Die Flussmittelung unterscheidet sich also von der Massen- und der Flächenmittelung grundsätzlich dadurch, dass für die beiden Letzteren die Strömungsvariablen zunächst für jede Zelle ermittelt, gewichtet und aufsummiert werden, wohingegen bei der Flussmittelung alle zusätzlichen Strömungsgrößen auf Basis der integralen primitiven Variablen berechnet werden.

Nach Ermittlung der Variablen wird die *success-flag* der zugehörigen IntegralValues-Struktur auf *AVERAGING_SUCCESSFUL* gesetzt. Falls jedoch eine der gewählten Mittelungsvarianten für ein Band oder für die Gesamtfläche nicht erfolgreich durchgeführt werden kann, wird *AVERAGING_FAILED* eingetragen. Die Mittelungsergebnisse werden jedoch nicht gelöscht. So bleibt anhand der Ergebnisse nachvollziehbar, aus welchen Gründen die spezifizierte Variante nicht erfolgreich war. Als Kriterium für das Fehlschlagen einer Flussmittelung gelten analog zu *TRACE* folgende Kriterien:

- $|Ma|$ normal zur Fläche $< 0,025$
- $\rho < 0$

Variable	**Symbol**	**avgf**	**avgm**	**avga**
Dichte	ρ	F	M	A
Druck	p	F	M	A
Temperatur	T	F, I	M	A
Enthalpie	h	F, I	M	A
Entropie	s	F, I	M	A
x-Geschwindigkeit	v_x	F	M	A
r-Geschwindigkeit	v_r	F	M	A
θ-Geschwindigkeit (rel)	$v_{\theta,rel}$	F	M	A
θ-Geschwindigkeit (abs)	$v_{\theta,abs}$	F, I	M	A
Schallgeschwindigkeit	a	F, I	M	A
Mach-Zahl (abs)	Ma_{abs}	F, I	M	A
Mach-Zahl (meridional)	Ma_m	F, I	M	A
Mach-Zahl (rel)	Ma_{rel}	F, I	M	A
Strömungswinkel (abs)	α_{tan}	F, I	M	A
Strömungswinkel (rel)	β_{tan}	F, I	M	A
Strömungswinkel (abs)	α_z	F, I	M	A
Strömungswinkel (rel)	β_z	F, I	M	A
Meridianwinkel	ϵ	F, I	M	A
Totalenthalpie (abs)	h_t	F, I	M	A
Totalenthalpie (rel)	$h_{t,rel}$	F, I	M	A
Totaltemperatur (abs)	T_t	F, I	M	A
Totaltemperatur (rel)	$T_{t,rel}$	F, I	M	A
Totaldruck (abs)	p_t	F, I	M	A
Totaldruck (rel)	$p_{t,rel}$	F, I	M	A
Turbulente Dissipation	ω	A	M	A
Turbulente kinetische Energie	k	A	M	A
Turbulentes Längenmaß	l_t	A	M	A
Wirbelviskosität	ν_t	A	M	A
Massenstrom	$\dot{m}$	S	S	S
Massenstrom (reduziert)	$\dot{m}_{red}$	F, I	M, I	A, I
Enthalpiefluss	-	-	S	S
Drall	L	F/A, I	M	A
Reynolds-Zahl	Re	F, I	M	A
Fläche	A	S	S	S
Radius	r_m	A	M	A
x-Koordinate	x	A	M	A

Tabelle 5.12: Integrale und bandintegrale umfangsgemittelte Variablen für die gta-Option, Legende siehe *Tab. 5.13*

Abkürzung	Bedeutung
F	flussgemittelter Wert
A	flächengemittelter Wert
M	massengemittelter Wert
S	aufsummierter, ungewichteter Wert
I	aus integralen Werten abgeleitet

Tabelle 5.13: Legende zu *Tab. 5.12*

- $|\dot{m}| = |\mathrm{F}_0| = 0$
- Diskriminante der Gleichung zur Ermittlung des statischen Drucks (*Gl. (5.11)*) kleiner gleich Null

Für die Massenmittelung gilt hingegen:

- $|\dot{m}| = 0$
- $p <= 0$

Das letzte Kriterium gilt ebenso für die Flächenmittelung, wobei dieses nur bei aktivierter gta-Option abgefragt wird.
Durch die Markierung mittels der *success-flag* kann in folgenden Tasks, welche zum Beispiel auf flussgemittelten Größen basieren, entschieden werden, ob für den Fall des Fehlschlagens auf eventuell vorhandene massen- oder flächengemittelte Werte zurückgegriffen werden soll. Die ursprüngliche Idee alle Bänder einer Familie auf *AVERAGING_FAILED* zu setzen, falls die Mittelung für ein einzelnes Band fehlschlägt, wurde nicht umgesetzt. Denn die Flussmittelung kann in Wandnähe aufgrund des dort vorliegenden niedrigen Geschwindigkeitsniveaus manchmal fehlschlagen.
Es wäre zudem vorteilhaft, dass beim Fehlschlagen des vom Benutzer vorgegebenen Mittelungstyps ein automatisches Umschalten erfolgt, also von einer Fluss- auf eine Massen- und von der auf eine Flächenmittelung. Dieses Umschalten müsste jedoch bereits in den Kernroutinen der Mittelung erfolgen und für alle Teilstücke eines Bands, welche zu verschiedenen Blöcken gehören, realisiert werden. Dies ist jedoch nicht möglich, da sich zum Beispiel Analyseflächen aus mehreren Teilblöcken zusammensetzen, die in der Regel auf mehrere Prozesse verteilt sind, welche untereinander während dieser Routinen nicht kommunizieren können. Folglich ist es für hinsichtlich der Mittelung instabile Konfigurationen zu empfehlen, alle drei Mittelungsvarianten auszuführen. Denn falls beispielsweise die Flussmittelung fehlschlagen sollte, kann ein nachfolgender Task immer noch auf die Ergebnisse der Massen- oder Flächenmittelung zurückgreifen.
Im Fall des Fehlschlagens der Flussmittelung ist es dabei zu empfehlen auf die Massenmittelung zu wechseln, da nach [CH06] die flussgemittelte Totalenthalpie der massengemittelten entspricht. Für ein ideales Gas gilt dieselbe Aussage auch für die Totaltemperatur. Insgesamt wird somit erreicht, dass sich für die Analyse der Arbeitsumsetzung keinerlei Unterschiede ergeben. Eine flächengemittelte Totalenthalpie ist dagegen physikalisch weniger sinnvoll, da es sich bei h_t um eine extensive Zustandsgröße handelt. Für diese Größen ist es am sinnvollsten entweder eine Fluss- oder eine Massenmittelung

durchzuführen.
Die Ergebnisse der Mittelungsroutinen zeigen *Abb. 5.19* sowie *Abb. 5.20*. Erstere veranschaulicht die Abweichungen der Mittelung von *POST* gegenüber derjenigen von *TRACE* am Beispiel der radialen Verteilung des Totaldrucks $p_{t,abs}$ am Outlet des Testfalls THD R1. Dabei werden jeweils die verschiedenen implementierten Mittelungsarten folgendermaßen miteinander verglichen:

$$\Delta p_{t,abs} = 100 \cdot \frac{p_{t,abs}|_{POST} - p_{t,abs}|_{TRACE}}{p_{t,abs}|_{TRACE}}$$

Die zweite Abbildung vergleicht die Resultate der in *POST* implementierten Mittelungsarten anhand der radialen Verteilung der Totaltemperatur $T_{t,abs}$, ebenso am Outlet des Testfalls THD R1. Die Werte der Flächen- und Massenmittelung sind dabei auf diejenigen der Flussmittelung bezogen.

$$\Delta T_{t,abs} = 100 \cdot \frac{T_{t,abs}|_{Masse/Fläche} - T_{t,abs}|_{Fluss}}{T_{t,abs}|_{Fluss}}$$

Insgesamt zeigt sich aus *Abb. 5.19*, dass die Unterschiede zwischen der Mittelung von *TRACE* und *POST*, beispielhaft gezeigt für die radiale Verteilung des Totaldrucks, vernachlässigbar gering sind. Die größten Abweichungen treten dabei im Randbereich auf, vermutlich aufgrund der im Gegensatz zu den *TRACE*-Resultaten fehlenden Grenzschichtauflösung bzw. der äquidistanten Verteilung der Bänder in *POST* (siehe *Kap. 5.4*). Anzumerken ist, dass die radialen Verteilungen von *POST* der Analysefamilie entnommen sind, welche dem Outlet am nächsten liegt, jedoch nicht exakt mit diesem übereinstimmt. Die *TRACE*-Resultate hingegen gelten für die Outlet-Panelfamilie. Da sich zudem die Bandanzahl und Bandverteilung von *TRACE* und *POST* unterscheiden, werden zum Vergleich die *TRACE*-Werte auf die von *POST* definierten radialen Positionen interpoliert. Dies erfolgt über die in *MATLAB R2009a* implementierte Spline-Interpolation.
Die zweite Abbildung, *Abb. 5.20*, erbringt den Nachweis, dass die Mittelung in *POST* korrekt implementiert ist, da die Ergebnisse für die massen- und die flussgemittelte Verteilung der Totaltemperatur im Absolutsystem identisch sind ([CH06]). Neben der Totaltemperatur $T_{t,abs}$ und dem Totaldruck $p_{t,abs}$ erfolgte ebenso eine Überprüfung der integralen Werte für alle weiteren Strömungsgrößen, welche von *TRACE* berechnet werden.

5.6 Globale Analyse der Turbomaschinen-Konfiguration - *globalTurbomachineAnalysis*

Im Folgenden wird die Implementierung des *globalTurbomachineAnalysis*-Tasks (*Tab. 5.14*) vorgestellt. Dessen Aufgabe besteht darin, charakteristische 0D-Kenngrößen zur Beurteilung der zu analysierenden Turbomaschinen-Konfiguration zu berechnen und auszugeben. Ebenso werden 1D-Verteilungen beziehungsweise für den Fall einer Axialmaschine radiale Verteilungen ermittelt. Dieser Task ersetzt somit das Postprocessing des MTU-Verzeichnisses von *TRACE*.

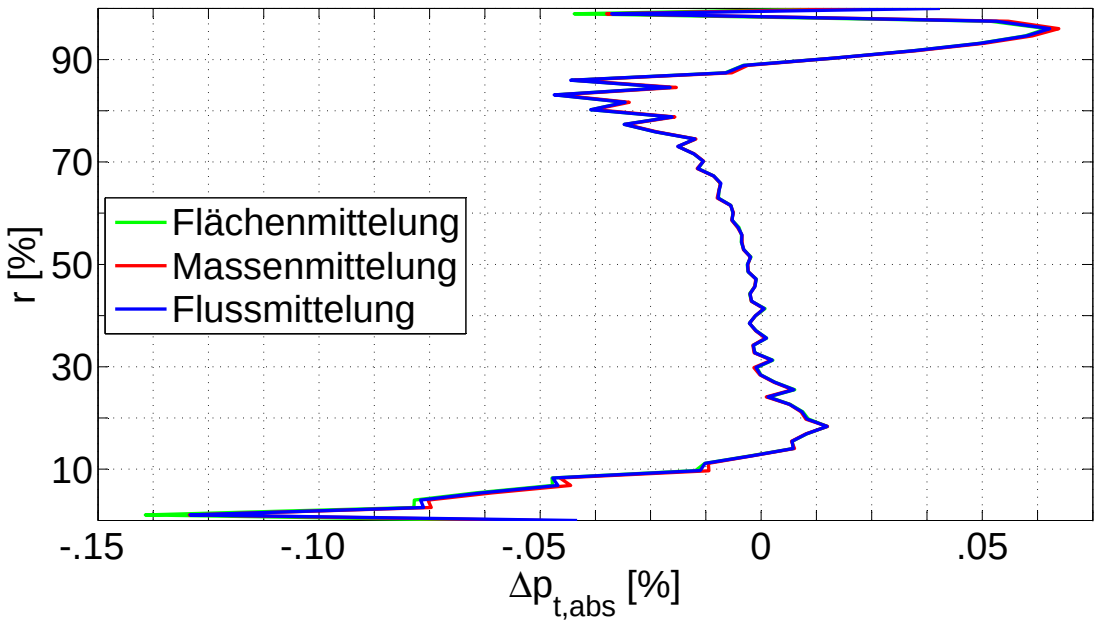


Abb. 5.19: Radiale Verteilungen der Abweichungen im Totaldruck-Niveau von *POST* bezüglich *TRACE* am Outlet des Testfalls THD R1

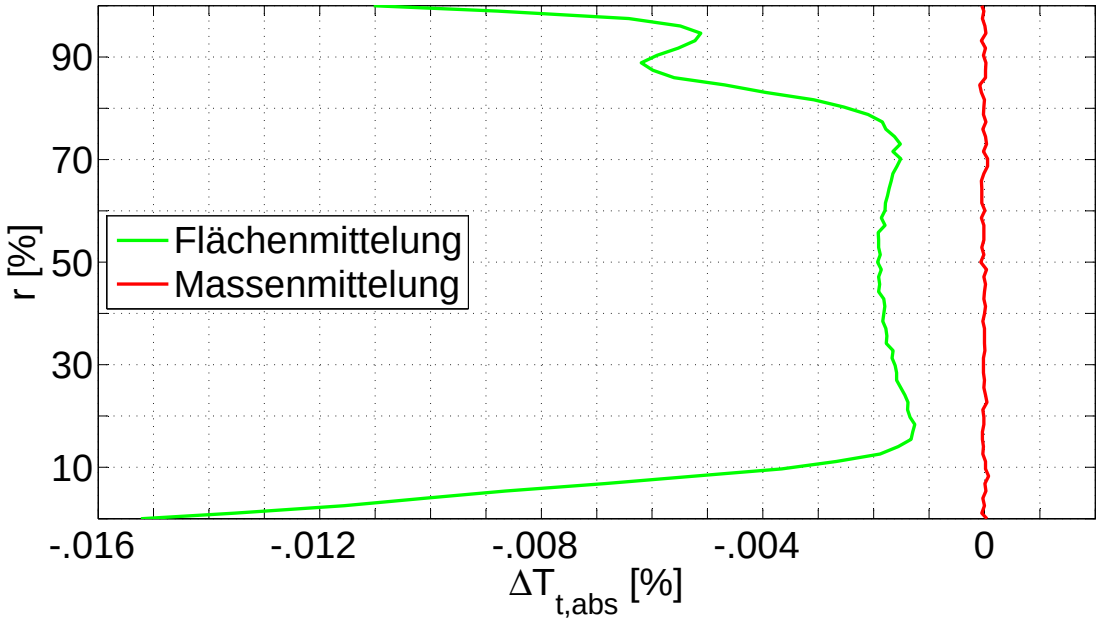


Abb. 5.20: Radiale Verteilungen der Abweichungen der massen- und flächengemittelten Totaltemperatur bezüglich der flussgemittelten am Outlet des Testfalls THD R1

Name	globalTurbomachineAnalysis
Aufruf	-gta75
Optionen	-

Tabelle 5.14: Übersicht über Aufruf und Optionen des *globalTurbomachine-Analysis*-Tasks

Der *globalTurbomachineAnalysis*-Task oder kurz *GTA*-Task ist bereits für *POST 7.3* implementiert und validiert worden. Die 7.3er Variante basiert jedoch auf einem *TRACE*-Initialisierungsschritt. Von diesem wird unter anderem die Umfangsmittelung durchgeführt sowie der Aufbau der Turbomaschine bereitgestellt. Der *GTA*-Task in *POST 7.5* bietet dem Anwender dieselbe Funktionalität, basiert jedoch vollständig auf *POST*-Strukturen. Bevor dieser ausgeführt wird sind also die Tasks der S2-Mittelung aufzurufen. Im Folgenden beschränkt sich die Beschreibung auf die in *POST 7.5* implementierte Variante des *GTA*-Tasks. Der grundsätzliche Programmablauf der 7.3er Variante ist jedoch ähnlich strukturiert.

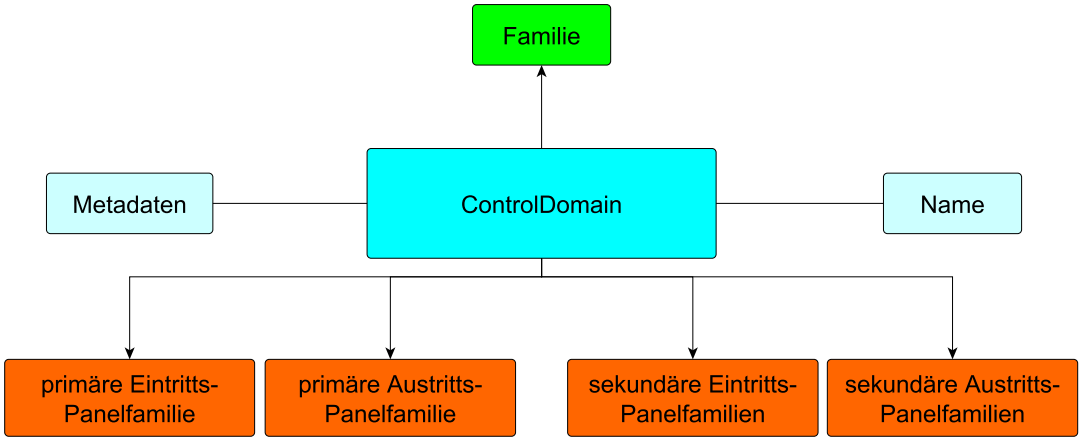


Abb. 5.21: *ControlDomain*-Struktur des *globalTurbomachineAnalysis*-Tasks

Der *GTA*-Task basiert auf der sogenannten *ControlDomain*-Struktur (*Abb. 5.21*). Dabei stellt jede auszuwertende Blockgruppe, beispielsweise eine Schaufelreihe oder eine Stufe, ein eigenes Kontrollvolumen bzw. *ControlDomain* dar, welches über seine jeweiligen Randflächen definiert ist. Für die Auswertung der aero-thermodynamischen Verhältnisse sind ausschließlich durchströmte Flächen von Interesse, welche als primäre oder sekundäre In- oder Outlets der *ControlDomains* behandelt werden.
In *POST 7.3* erfolgt diese Auswertung stets für Kontrollvolumina, welche durch Interfaces, In- und Outlets eingeschlossen, begrenzt sind. Für *POST 7.5* hingegen ist geplant, die Auswertung dahingehend zu erweitern, dass auch *ControlDomains* basierend auf Analyseflächen, welche vom Benutzer nahezu beliebig vorgegeben werden können, behandelt werden können. Dazu ist es nötig, dass der Benutzer die entsprechenden Randflächen der Kontrollvolumina über *GMC* spezifiziert. Diese Option war jedoch bei Fertigstellung der Arbeit noch nicht vollständig implementiert. Die Auswertung beschränkt sich somit momentan auf durch Interfaces begrenzte Volumina. Dazu zählen:

- Schaufelreihen
- Stufen
- Komponenten
- Analysevolumina

Die Analysevolumina stellen dabei eine vom Benutzer definierbare zusammenhängende Gruppe von Schaufelreihen bzw. Blockgruppen dar.
Der *GTA*-Task setzt sich aus folgenden drei Programmteilen zusammen:

- Aufbau der Turbomaschinen-Konfiguration
- Berechnung der Kenngrößen für die Kontrollvolumina
- Ausgabe der Ergebnisse

Wenn in *GMC* die Option implementiert ist, dass der Benutzer die Stromflächen der *ControlDomains* vorgeben kann, ist der nachfolgend erläuterte Aufbau der Turbomaschine deutlich einfacher zu gestalten.
Der Input-Task von *POST* liest die in der von *TRACE* gelieferten CGNS-Datei abgelegten Informationen ein und erstellt für jede Stufe, jede Schaufelreihe, jede Panelfamilie, usw. eine eigene Familie, welche im Familienarray des Datensets gespeichert wird. Basierend auf diesen Familien und den eingelesenen Informationen wird dann für den *GTA*-Task die Struktur der untersuchten Turbomaschine lokal aufgebaut.
Im ersten Schritt werden dabei die vorhandenen durchströmten Flächen des Familienarrays bearbeitet. Dies erfolgt im Rahmen einer Schleife über alle Familien des Datensets. Hat eine Familie den Typ *FAMILY_TYPE_BOUNDARY* und handelt es sich bei der Fläche um einen Eintritt, einen Austritt, ein Interface, eine Kavität oder eine Leckage, dann wird für die Turbomaschine eine entsprechende Panelfamilien-Struktur aufgebaut. Diese enthält auch einen Pointer auf die zugehörige Familie des Datensets, um auf die dort abgelegten Daten zugreifen zu können. Anschließend erfolgt für alle aufgebauten Panelfamilien der Turbomaschine eine lokale Neu-Definition ihres Randbedingungs-Typs, da die Analyse der Konfiguration bzw. der einzelnen Blockgruppen nur auf den ein- und austretenden Fluidströmen der Kontrollvolumina beruht. Ergo gibt es für den *GTA*-Task nur noch In- und Outlets.
Ebenso ist zu definieren, welche Ein- und welche Austritte eines Kontrollvolumens die primären sind, also welche die Hauptströmung beschreiben, und welche die sekundären sind. Denn die Änderung des Strömungszustands über eine Komponente wird für die Hauptströmung betrachtet. Ein Beispiel wäre das Druckverhältnis oder die Änderung von Strömungswinkeln zwischen Haupteintritt und Hauptaustritt einer *ControlDomain*. Bei der Berechnung der Leistungsumsetzung mit Hilfe des ersten Hauptsatzes (siehe *Kap. 2.2.2*) liefern aber auch Nebenströme, also sekundäre Stromflächen, einen zu berücksichtigenden Beitrag. Die Entscheidung, ob es sich bei einer durchströmten Fläche um eine primäre oder sekundäre Stromfläche handelt, wird anhand ihres *TRACE*-Boundary-Typs bestimmt. Dabei sind In- und Outlets immer primäre Stromflächen und Interfaces meist. Alle anderen Typen werden als sekundäre Stromflächen behandelt. Gust-Randbedingungen werden aufgrund fehlender Informationen in *POST* nicht unterstützt, da nicht bestimmt werden kann, ob es sich um eine primäre oder sekundäre Stromfläche handelt.
Für ein Interface erfolgt eine Umdefinition zu einem primären Ein- oder Austritt, sofern dieses zwischen zwei Schaufelreihen liegt. Falls eine der beiden Interface-Seiten zu einer Kavität gehört, wird diese durchströmte Fläche, also beide Seiten des Interfaces, als sekundär betrachtet, da Kavitäten selbst nicht ausgewertet werden, jedoch ihr Beitrag

zur Leistungsumsetzung zu berücksichtigen ist. Zur Entscheidung, ob es sich bei der betrachteten Fläche, egal, ob es sich um eine primäre oder sekundäre Stromfläche handelt, um einen Ein- oder einen Austritt handelt, wird der integrale Wert des Totalenthalpieflusses $\int \dot{m}\, h_{t,abs}$ über diese Fläche betrachtet. Ist dieser negativ, die Strömung tritt also in das Kontrollvolumen ein, handelt es sich um einen Einlass und ist dieser positiv, die Strömung verlässt also das Kontrollvolumen, handelt es sich um einen Auslass. Falls $\int \dot{m}\, h_{t,abs} \approx 0$ wird die Fläche nicht durchströmt und folglich für die Auswertung nicht berücksichtigt.
Nach der Definition aller Panelfamilien der Turbomaschine werden die Kontrollvolumina bzw. die *ControlDomain*-Strukturen beginnend mit den Komponenten aufgebaut. Hierfür werden die eingelesenen Komponenten-Familien des Datensets benötigt. Anzumerken ist, dass die Implementierung grundsätzlich mehrere Komponenten zulässt, an manchen Stellen des *GTA*-Tasks wird momentan aber nur eine Komponente berücksichtigt. Die Komponenteninformationen werden dabei auf zwei unterschiedliche Strukturen verteilt. Informationen, welche den Aufbau der Komponente betreffen, also die Anzahl an Schaufelreihen und Stufen sowie ein Array der zugehörigen Schaufelreihen und Stufen, beinhaltet die Struktur *gtaComponentStructure*. Gleichzeitig wird die Komponente auch als auszuwertendes Kontrollvolumen betrachtet. Die Kontrollvolumen werden immer als *gtaControlDomain* definiert und enthalten unter anderem, ebenso wie die Panelfamilien, einen Pointer auf die ursprüngliche Familie des Datensets. Die später berechneten Kenngrößen werden schließlich in die IntegralValues-Strukturen dieser Familien geschrieben.
Daraufhin werden für jede Komponente der Maschine die zugehörigen Schaufelreihen aufgebaut und in der jeweiligen *gtaComponentStructure* gespeichert. Diese sind dabei vom Inlet zum Outlet der Konfiguration zu ordnen. Zunächst wird hierzu der Inlet und der Outlet einer Komponente ermittelt bzw. die Schaufelreihen, zu welchen diese beiden Panelfamilien gehören. Ausgehend von der Inlet-Schaufelreihe läuft dann der Algorithmus von Schaufelreihe zu Schaufelreihe, indem er vom jeweiligen Outlet-Interface der aktuellen Schaufelreihe zum Inlet-Interface der folgenden Schaufelreihe springt. Der Algorithmus endet, wenn die Outlet-Schaufelreihe erreicht ist.
Die Ordnung der Schaufelreihen ist deshalb von Bedeutung, da der Inlet der ersten Schaufelreihe ebenso der Inlet der zugehörigen Komponente ist. Folglich wird der Komponente der Inlet-Pointer ihrer ersten Schaufelreihe übergeben. Dasselbe gilt für den Outlet. Des Weiteren ist über diese Ordnung eine strukturierte Ausgabe der Ergebnisse möglich. Das bedeutet, dass die Schaufelreihen-Datenblöcke später in vom Inlet zum Outlet geordneter Reihenfolge ausgegeben werden können. Jedoch gelten somit folgende Einschränkungen:

- Jede Komponente beginnt mit einer Panelfamilie mit dem Typ *ENTRY* und endet mit einer Panelfamilie mit dem Typ *EXIT*
- Es folgt immer genau eine Schaufelreihe auf eine vorhergehende (keine Splitter-Konfigurationen). Bei Kavitäten handelt es sich zwar ebenfalls um Blockgruppen, sie werden jedoch nicht als Volumen berücksichtigt.

Diese Einschränkungen bedeuten, dass jede Schaufelreihe und damit auch jede *ControlDomain* je eine primäre Eintritts- und eine primäre Austritts-Panelfamilie sowie eine

beliebige Zahl sekundärer Panelfamilien besitzen kann. Die Beschränkung auf jeweils einen primären In- und Outlet ist jedoch auch sinnvoll, da somit eindeutig definiert ist zwischen welchen beiden Ebenen die integralen Kenngrößen zu berechnen sind. Nach dem Aufbau einer Schaufelreihe werden dieser die zugehörigen primären und sekundären Panelfamilien, welche bereits für die Turbomaschine aufgebaut worden sind, zugewiesen.
Auf Basis der Schaufelreihen werden schließlich die Stufen und die Analysevolumina aufgebaut. Eine Stufe besteht dabei aus einer Stator- sowie einer Rotor-Schaufelreihe und ein Analysevolumen aus einer beliebigen Zahl aufeinander folgender Schaufelreihen. Die Zuweisung der Randflächen erfolgt über die Informationen der Schaufelreihen, das heißt die erste Schaufelreihe liefert jeweils den Inlet und die letzte den Outlet. Alle dazwischen liegenden übergeben nur ihre sekundären Strömungsflächen. Dieselbe Vorgehensweise wird angewandt, um die Randflächen der Komponenten zu ermitteln. Aus diesem Grund ist, wie bereits erwähnt, die korrekte Reihenfolge der Schaufelreihen wichtig, also geordnet von der Eintritts-Panelfamilie der Komponente zur Austritts-Panelfamilie.
An den Aufbau der Turbomaschine schließt sich die Berechnung der integralen Kenngrößen für die Kontrollvolumina an. Welche Größen hierbei berechnet werden, richtet sich nach dem Typ der *ControlDomain*. In der 7.3er Variante des *GTA*-Tasks werden zunächst für die Panelfamilien weitere Strömungsgrößen basierend auf den primitiven Variablen abgeleitet. Dies wird in *POST 7.5* bereits im Rahmen der Mittelung durch die GTA-Option bewerkstelligt (*Kap. 5.5*). Die Größen, welche für die primären Panelfamilien benötigt bzw. berechnet werden, sind *Tab. 5.12* zu entnehmen.
Die Auswertung der Kontrollvolumina erfolgt ausschließlich für die Lösung mit dem Namen „FlowSolution“. Die berechneten integralen 0D-Größen umfassen vor allem Kenngrößen, welche die Leistungs- und Arbeitsumsetzung, Wirkungsgrade sowie Strömungsverluste charakterisieren. Diese beschreiben die Änderung des Strömungszustands über das betrachtete Kontrollvolumen. Zur Ermittlung der radialen Verteilungen werden dieselben Formeln verwendet wie zur Berechnung der 0D-Kenngrößen. Die abgeleiteten integralen Werte der 1D-Verteilung basieren dabei jeweils auf den beiden zueinander gehörenden Bändern der primären Inlet- und der primären Outlet-Panelfamilie einer *ControlDomain*. Die Ergebnisse sind jedoch nicht immer sinnvoll. Denn, wenn beispielsweise eine Leistungsbilanz für ein Bandpaar aufgestellt wird, bekommt der Einfluss sekundärer Panelfamilien durch den bezogen auf die Bänder vergleichsweise großen Massenstrom ein überproportional hohes Gewicht, da für sekundäre Stromflächen jeweils panelintegrale Werte in die Formeln einfließen.
Sowohl die integrale als auch die bandintegrale Auswertung erfolgt für alle im Rahmen der S2-Mittelung aufgerufenen Mittelungsarten, also im Allgemeinen für die Fluss-, die Massen- und die Flächenmittelung. Falls eine bestimmte Mittelungsart für einzelne Bänder oder die Gesamtfläche nicht erfolgreich durchgeführt worden ist, erfolgt analog zum *buildFinalS2M*-Task (*Kap. 5.7*) ein Umschalten der Mittelungsart von Fluss- auf Massen- und von dieser auf Flächenmittelung. Deshalb ist es zu empfehlen im Rahmen der S2-Mittelung alle Mittelungsarten und zwar für die GTA-Variante aufzurufen. Wie bereits erwähnt, besteht der Vorteil des Umschaltens von Fluss- auf Massenmittelung darin, dass nach [CH06] die fluss- und massengemittelte spezifische Totalenthalpie iden-

tisch sind. Da viele der Kenngrößen auf dieser basieren, ändern sich die Ergebnisse der Auswertung durch ein Umschalten nur geringfügig.
Abschließend erfolgt die Ausgabe der Resultate. Die hierfür verwendeten Routinen sind aus der 7.3er Variante des *GTA*-Tasks übernommen und entsprechend angepasst. Die Ausgabe des *globalTurbomachineAnalysis*-Tasks ist dabei zweigeteilt. Zunächst wird eine menschenlesbare Output-Textdatei erstellt, welche eine Übersicht ausgewählter 0D-Kenngrößen aller Kontrollvolumina enthält. Diese Datei wird für jeden ausgewerteten Mittelungstyp einzeln generiert. Im Allgemeinen werden somit folgende drei Dateien geschrieben:

- GTA_0D_fluxAveraged
- GTA_0D_massAveraged
- GTA_0D_areaAveraged

Des Weiteren wird die Datei „GTA.ulst“ erzeugt. Bei dieser handelt es sich um eine maschinenlesbare Datei basierend auf dem MTU-spezifischen ulst-Format (*unified lists*), welchem wiederum das JSON-Format (*JavaScript Object Notation*) zugrunde liegt. Die Verwendung dieses Formats bietet den Vorteil komplexe Datenstrukturen plattformunabhängig und dauerhaft lesbar abspeichern zu können. Diese Datei enthält alle berechneten integralen und bandintegralen Kenngrößen jedes Kontrollvolumens und jeder primären Stromfläche. Gegliedert ist die ulst-Datei zunächst nach den Elementtypen, also nach Panelfamilien, Schaufelreihen, Stufen, Komponenten und Analysevolumina. Unter jedem Elementtyp-Knoten befinden sich dann die einzelnen Elemente, also beispielsweise alle Schaufelreihen. Diese wiederum enthalten jeweils Datenblöcke, welche die berechneten integralen und bandintegralen Kenngrößen beinhalten. Auf der gleichen Ebene wie die Werte sind auch die Variablennamen sowie der jeweilige Mittelungstyp gespeichert.
Auch werden für jedes Element zusätzliche Metadaten gespeichert, beispielsweise die Eintritts- und Austritts-Panelfamilie jedes Kontrollvolumens oder für den Fall einer Stufe die Rotor- und Stator-Schaufelreihe. Am Ende befindet sich ein weiterer Datenblock, welcher Informationen zum Rechnungslauf enthält, zum Beispiel die verwendete *POST*-Version und das Erstellungsdatum der Datei. Mit Hilfe des sogenannten ulst-Readers, welcher jedoch bei Abschluss der Arbeit noch nicht zur Verfügung stand, kann diese Liste schließlich ebenfalls in ein menschenlesbares Format konvertiert werden.

5.7 Aufbau des finalen S2M-Netzes - *buildFinalS2M*

Im Folgenden wird die Implementierung des *buildFinalS2M*-Tasks (siehe *Tab. 5.15*) vorgestellt. Dessen Aufgabe besteht in der Erstellung des resultierenden S2M-Netzes der untersuchten Konfiguration. Neben dem Aufbau der Netzgeometrie werden auf die S2M-Fläche ebenso die im Rahmen der vorangegangenen Mittelungstasks berechneten umfangsgemittelten Strömungsvariablen übertragen. Da der Aufbau der S2M-Fläche auf den integralen Werten der Analysefamilien, auf welche jeder Prozess zugreifen kann, ba-

Name	buildFinalS2M		
Aufruf	-bS2M		
Optionen	-sol	Name der Solution	obligatorisch
	-avg	area/mass/flux	obligatorisch
	-switch	band/line	obligatorisch

Tabelle 5.15: Übersicht über Aufruf und Optionen des *buildFinalS2M*-Tasks

siert, kann dieser Task auf allen Prozessen ausgeführt werden. Das resultierende Gitter steht somit für nachfolgende Tasks allen Prozessen zur Verfügung. Die grundlegende Idee des Tasks ist, dass jede existierende Analyseflächen-Familie, welche im Rahmen des *intersect3DS2MPlane*-Tasks (*Kap. 5.4*) erstellt werden, eine Netzlinie I = konstant, also eine Netzlinie vom Nabe zum Gehäuse, darstellt.
Zunächst erfolgt die Abfrage der Benutzereingaben. Für diesen Task ist erstens der Name derjenigen Lösung anzugeben, welche auf der S2M-Fläche dargestellt werden soll („-sol“). In den meisten Fällen handelt es sich dabei um die Strömungslösung mit dem Namen „FlowSolution“. Da jede vorhandene Lösung auf unterschiedliche Arten gemittelt werden kann, ist des Weiteren vorzugeben, welche umfangsgemittelten Werte der spezifizierten Lösung verwendet werden sollen („-avg area/mass/flux“).
Zudem ist das Verhalten des Tasks für den Fall zu definieren, dass für einzelne Bänder die Mittelung nach der gewählten Variante fehlgeschlagen ist. Dies kann insbesondere für die Flussmittelung in den Grenzschicht-Bändern auftreten. Die Information über den Mittelungserfolg ist dabei in den IntegralValues-Strukturen jedes Bands abgelegt und wird durch den *averagingInversion*-Task definiert (*Kap. 5.5.3*). Über die Option „-switch“ kann vom Benutzer festgelegt werden, ob im Fall des Fehlschlagens der gewählten Mittelungsvariante, der Mittelungstyp nur für das jeweilige Band („-switch band“) geändert wird oder für die komplette Analysefamilie („-switch line“), zu welcher dieses Band gehört. Um ein Umschalten zu ermöglichen, sollten für die spezifizierte Lösung mehrere Mittelungsvarianten, zumindest eine zusätzliche Flächenmittelung, durchgeführt werden. Das automatische Umschalten erfolgt dabei von Fluss- auf Massen- und von dieser auf Flächenmittelung.
Die ermittelten integralen Werte der einzelnen Bänder werden also nicht auf das S2M-Netz übertragen, wenn die Mittelung fehlgeschlagen ist. Zwar wird dem Benutzer mitgeteilt, für welches Band ein Umschalten des Mittelungstyps erfolgt, jedoch kann er im fertigen S2M-Netz nicht mehr erkennen, nach welcher Methode die einzelnen Bandwerte bestimmt worden sind.
Bevor der eigentliche Aufbau des strukturierten S2M-Netzes beginnt, wird zunächst die Funktion *generateDummyBands* aufgerufen, welche sicherstellt, dass jede Analysefamilie über dieselbe Anzahl an Bändern verfügt. Diese Überprüfung kann erst nach der im *averagingInversion*-Task erfolgten Kommunikation durchgeführt werden, nachdem die Beiträge und Informationen aller Bänder einer Schnittfamilie gesammelt worden sind. Die Funktion überprüft zunächst für jede Analysefamilie, ob deren erstes und letztes Band existieren, also diejenigen Bänder, welche unmittelbar am Gehäuse bzw. an der Nabe liegen. Denn für ein vom Benutzer generiertes und eingelesenes S2M-Netz

kann es bei der Verschneidung des *intersect3DS2MPlane*-Tasks vorkommen, dass diese Bänder für einzelne Analysefamilien fehlen. Dies tritt jedoch nur dann auf, wenn das geladene S2M-Netz in Normalrichtung größere Abmessungen aufweist als diejenigen der 3D-Konfiguration. Falls eines der Bänder fehlt, wird der *buildFinalS2M*-Task nicht fortgesetzt, sondern mit einer Fehlermeldung abgebrochen.
Des Weiteren werden im Rahmen der Funktion *generateDummyBands*, wie der Name bereits andeutet, fehlende innere Bänder als Pseudobänder für die Analysefamilien aufgebaut. Denn dadurch, dass ein strukturiertes S2M-Netz auf Basis der Analysefamilien aufgebaut wird, benötigt jede von diesen dieselbe Anzahl an Bändern. Für Splitter-Konfigurationen beispielsweise fehlen jedoch einzelne innere Bänder im unvernetzten Bereich des 3D-Modells, welcher Haupt- und Nebenströmung voneinander trennt. Für diese Lücke werden die fehlenden Bänder nachträglich definiert. Hierzu werden für jede Schnittfamilie jeweils die fehlende Anzahl an Bändern sowie die geometrischen Abmessungen der Lücke in der *xr*-Ebene ermittelt. Diese Lücke wird anschließend mit äquidistant verteilten Bändern aufgefüllt, was bedeutet, dass Geometriedaten der fehlenden Bänder berechnet und in deren IntegralValues-Struktur mit dem „Mittelungstyp“ *BAND_DIMENSIONS* abgelegt werden. Alle Variablen der Strömungslösung dieser Pseudobänder werden dabei zu Null gesetzt.
Der Aufbau des S2M-Netzes und der zugehörigen Lösung erfolgt mittels einer Schleife über alle vorhandenen Analyseflächen-Familien. Dadurch, dass diese in der Regel nicht in der benötigten Reihenfolge, also nach aufsteigender ξ-Schnittposition, im Datenset abgelegt sind, werden sie zunächst geordnet. Hierfür wird ein Familienpointer-Array erstellt, welches Zeiger auf alle Analyseflächen des Datensets enthält. Anhand des Familiennamens einer Schnittfläche, welcher jeweils die Form „ISO_XI_CUT_50.000000“ aufweist, kann deren Schnittposition bestimmt und somit die Zeiger des Arrays in die richtige Reihenfolge gebracht werden.
Der Aufbau des Netzes erfolgt sukzessive, also Netzebene für Netzebene bzw. Familie für Familie, weshalb zur Vereinfachung der Speicherung der Koordinaten der Vertices und der Lösungsvariablen die I- und J-Indexrichtung zunächst vertauscht werden. I entspricht somit temporär der Normalrichtung und J der Meridionalrichtung. Diese Zuordnung wird im Anschluss an die Analysefamilien-Schleife wieder invertiert, um die übliche Notation zu erhalten.
Innerhalb der Schleife über alle Analysefamilien wird, falls die Option „-switch line“ gewählt wurde, zuerst für jede Familie der zu verwendende Mittelungstyp für die spezifizierte Lösung festgelegt. Für diesen muss auf allen Bändern der Familie die Mittelung erfolgreich durchgeführt worden sein, das heißt im Optimalfall handelt es sich um den vom Anwender vorgegebenen Typ. Falls jedoch die Option „-switch band“ aktiviert ist, geschieht das automatische Umschalten des Mittelungstyps für jedes Band einzeln, falls die Mittelung für dieses fehlgeschlagen ist.
Daraufhin schließt sich die Initialisierung des Netzes sowie der Lösung der S2M-Fläche an. Hierfür wird auf die Informationen der IntegralValues-Strukturen des ersten Bandes der ersten Analysefläche zurückgegriffen. Beim Mittelungstyp der spezifizierten Lösung handelt es sich dabei um den jeweils vom Benutzer vorgegebenen. Ob für diesen die Mittelung erfolgreich durchgeführt worden ist, spielt keine Rolle, da ausschließlich die Information benötigt wird, welche Variablen vorhanden sind. Jedoch werden nicht alle

auf der IntegralValues-Struktur des ersten Bandes sich befindenden Strömungsgrößen auf das S2M-Netz übertragen bzw. die entsprechenden PostField-Strukturen für diese aufgebaut. Ausnahmen bilden beispielsweise die Gewichtungsfaktoren der Mittelung.
Anzumerken ist, dass die Lösung der S2M-Fläche nicht knoten- oder zellbasiert gespeichert wird, sondern im Schwerpunkt der ehemaligen Bänder (*IFaceCenter*). Der Vorteil dieser Variante besteht darin, dass die Werte dort gespeichert sind, wo sie auch ermittelt werden. Somit können Fehler aufgrund einer Interpolation der Strömungsvariablen vermieden werden. Jedoch wird diese Option von *Tecplot 2011 Release 1* nicht unterstützt, obwohl sie nach [Ian08] der *CGNS*-Norm entspricht.
Das S2M-Netz verfügt über eine strukturierte Topologie und wird in der *xr*-Ebene, also in zylindrischen Koordinaten, aufgebaut. Die Anzahl der Netzknoten in Normal- und Meridianrichtung ergibt sich wie folgt:

- Anzahl an Bändern pro Familie + 1 ergibt die Anzahl der Netzknoten in Normalrichtung
- Anzahl an Familien ergibt die Anzahl der Netzknoten in Meridianrichtung

Im Anschluss an die Initialisierung wird das zu erstellende Netz für jede Analysefamilie um jeweils eine Netzschicht erweitert. Die zum Aufbau des Gitters benötigten Band-Geometriedaten sind für jedes Band in dessen IntegralValues-Struktur mit dem „Mittelungstyp“ *BAND_DIMENSIONS* hinterlegt. Diese Struktur enthält die Minimal- und Maximalwerte der *x*- und *r*-Koordinate des jeweiligen Bandes, welche bereits im Rahmen der Mittelungsroutinen automatisch für jeden Block berechnet (*Kap. 5.5.2*), anschließend kommuniziert (Operation *MPI_MIN* bzw. *MPI_MAX*) und auf die zugehörigen Familien übertragen werden. Da sich die Oberseite eines Bandes, repräsentiert durch die beiden Maximalwerte, aufgrund numerischer Ungenauigkeiten insignifikant von der Unterseite des folgenden Bandes, repräsentiert durch dessen beiden Minimalwerte, unterscheiden kann, werden die Band-Geometriewerte zunächst korrigiert, bevor auf deren Basis die neue Netzlinie aufgebaut wird. Hierfür werden jeweils die beiden Oberseitenwerte eines Bandes durch die beiden Unterseitenwerte des sich darüber anschließenden Bandes ersetzt.
Nach dem Aufbau der Netzschicht erfolgt die Übertragung der gemittelten Strömungsvariablen, welche bereits im Rahmen der Initialisierung des S2M-Netzes festgelegt worden sind. So wird gewährleistet, dass für die gesamte Fläche der gleiche Satz von Variablen vorliegt, auch wenn ein automatisches Umschalten des Mittelungstyps erfolgt ist. Denn je nach gewählter Option für die einzelnen Mittelungsvarianten in *Kap. 5.5.2* , können für diese unterschiedliche Variablen bereitgestellt worden sein. Zusätzlich werden zwei weitere Variablen abgeleitet, der relative Massenstrom und eine Art relative Kanalhöhe, welche beide auf den Bereich von 0 bis 1 normiert sind. Die Ermittlung der relativen Kanalhöhe basiert dabei auf der Geometrie der eben erstellten Netzlinie. Die Gesamtlänge dieser Linie, also die Summe der Längen aller Bänder in der *xr*-Ebene, wird als Kanalhöhe an dieser Position definiert. Die relative Kanalhöhe eines Bandes stellt somit die Summe der Bandlängen bis zum Mittelpunkt des jeweils aktuellen Bands normiert auf die Gesamtlänge dar. Sie entspricht somit der relativen Lauflänge entlang der zugehörigen Netzlinie. Beim relativen Massenstrom eines Bandes handelt es sich um den integrierten Massenstrom von der Nabe bis zum Mittelpunkt dieses Bandes bezogen

auf die Beiträge aller Bänder einer Schnittfamilie, also bezogen auf den gesamten durch die betrachtete Analysefamilie durchströmenden Massenstrom. Die relative Kanalhöhe sowie der relative Massenstrom werden ebenfalls *IFaceCentered* gespeichert. Folglich treten für diese die beiden Extrema, 0 und 1, nicht auf.
Nachdem die Beiträge aller Analyseflächen zum S2M-Netz und zur S2M-Lösung berücksichtigt worden sind, erfolgt die Invertierung der Indexrichtung. Denn, wie oben bereits erwähnt, werden zur Generierung des S2M-Netzes die Indexrichtungen vertauscht. Diese Invertierung entspricht einer Umordnung der Speicherreihenfolge der Netzknoten sowie der Lösungsvektoren der einzelnen Variablen. Somit gilt letztendlich die übliche Indizierung:

- I-Index $\widehat{=}$ Meridionalrichtung
- J-Index $\widehat{=}$ Normalrichtung

Anschließend wird der Speicherort der S2M-Lösung temporär von *IFaceCenter* auf *Vertex* geändert werden, um die S2M-Fläche in *Tecplot 360* laden zu können. Hierfür werden die jeweils am Schwerpunkt eines Bandes gespeicherten Variablen auf dessen nabenseitigen Vertex verschoben (siehe *Abb. 5.22*). Für die zusätzliche Gehäuse-Netzlinie, für welche keine umfangsgemittelten Werte existieren, werden diejenigen der Strömungsgrößen der darunter liegenden Netzlinie kopiert. Abschließend wird die S2M-Fläche als neuer Block unter dem Namen „S2MareaAveraged“, „S2MmassAveraged“ oder „S2MfluxAveraged“, je nach der vom Benutzer vorgegeben Mittelungsvariante, mit dem Blocktyp *S2MPLANE_BLOCK* dem Datenset angefügt. Der neue Block gehört dabei zur Komponentenfamilie. Das Resultat dieses Tasks zeigt *Abb. 5.23* .

5.8 Generierung von Schaufelschnitten und S1-Stromflächen - *generateBladeCuts* und *finishBladeCuts*

Name	generateBladeCuts		
Aufruf	-genBlC		
Optionen	-option	blade/flow	obligatorisch
	-family	Namen zu verschneidender Familien	obligatorisch
	-load	*Tecplot*-Schnittdefinitionsdatei	obligatorisch

Tabelle 5.16: Übersicht über Aufruf und Optionen des *generateBladeCuts*-Tasks

Nach der Erstellung der S2M-Fläche der untersuchten Konfiguration kann auf deren Basis die Generierung von Schaufelschnitten und S1-Stromflächen erfolgen. Im Rahmen des *generateBladeCuts*-Tasks (siehe *Tab. 5.16*) lassen sich sowohl Schaufeloberflächen als auch die 3D-Blockgruppen des Strömungsgebiets verschneiden. Aus der Verschneidung resultieren somit entweder Schaufel-Schnittlinien oder S1-Stromflächen. Die Schnittebenen sind dabei über einen konstanten relativen Massenstrom oder eine konstante relative Kanalhöhe definiert.
Zunächst werden die Benutzereingaben eingelesen und entsprechend konvertiert. Vom

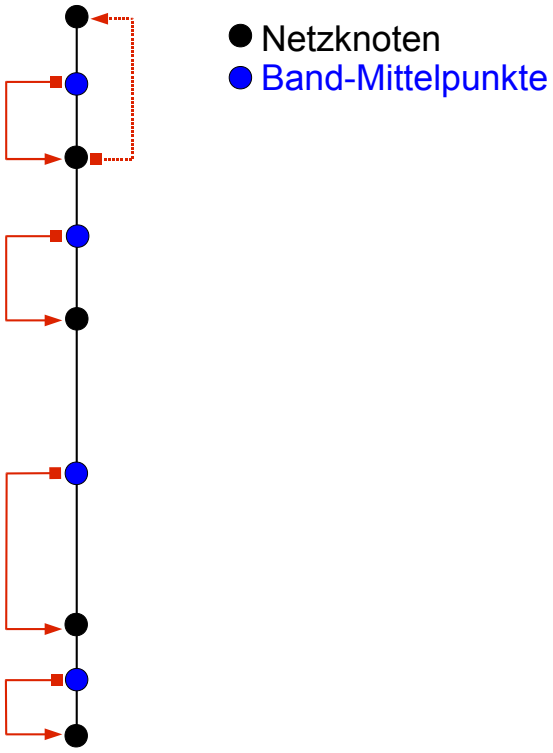


Abb. 5.22: Schema zur Änderung des Speicherorts der Lösung von *IFaceCenter* auf *Vertex* für eine Netzlinie

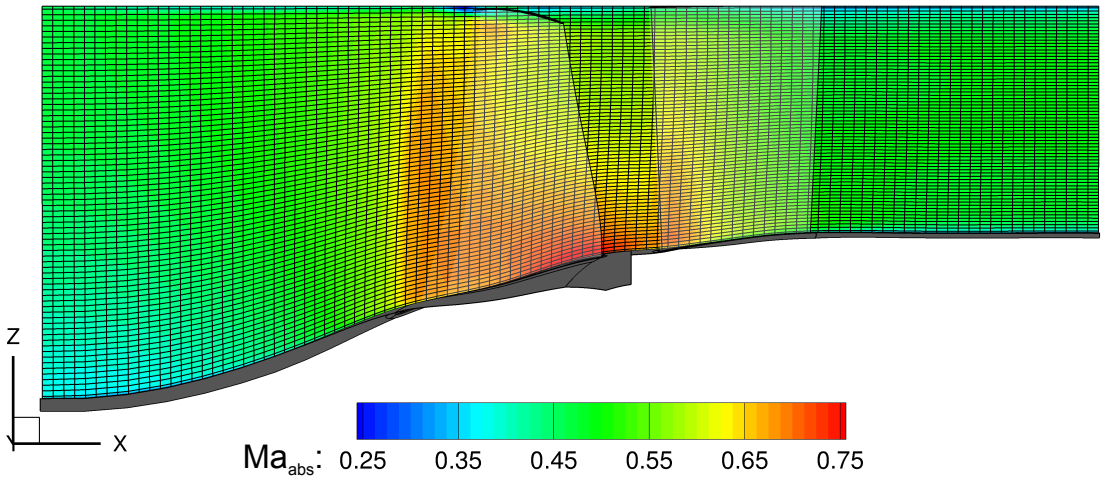


Abb. 5.23: Finale S2M-Fläche mit äquidistanter Vernetzung und Mach-Zahl-Verteilung (Testfall THD R1)

Anwender kann vorgegeben werden, ob entweder Schaufelschnitte („-option blade“) oder S1-Stromflächen („-option flow“) zu erstellen sind. Abhängig von dieser Vorgabe sind vom Anwender die zu verschneiden Blockfamilien, also entweder die 2D-Schaufeloberflächen bzw. Panelfamilien oder die 3D-Schaufelreihen bzw. Blockgruppen, über deren Namen zu spezifizieren („-family“). Einzugeben sind die Namen jeweils durch ein Leerzeichen getrennt. Für die übergebene Namensliste wird dann überprüft, ob die angegebenen Boundary- oder Blockgruppenfamilien im Datenset existieren. Des Weiteren ist über „-load“ der Name der *Tecplot*-Datei anzugeben, welche die Schnittdefinitionen enthält. Die Namensliste der zu verschneidenden Familien kann dabei nicht über diese Datei eingelesen werden, da die benutzte *readDataSetFromTec*-Funktion aus *TRACE* nur einen String von maximal 32 Zeichen zulässt. Dies beruht auf der *CGNS*-Spezifikation nach [Ian08]. Den prinzipiellen Aufbau einer Schnittdefinitionsdatei zeigt *Tab. 5.17*. Diese Datei ist für die Option „flow“ und „blade“ identisch und erzeugt für das dargestellte Beispiel insgesamt drei Schnitte bei 50, 60 und 70% relativer Kanalhöhe.

```
TITLE = "THD-cut-definition"
VARIABLES = "ChannelHeightRelative"
ZONE T = "cut-definition"
i=3, j=1, k=1, ZONETYPE=Ordered
DATAPACKING = POINT
DT=(SINGLE)
0.5
0.6
0.7
```

Tabelle 5.17: Beispielhafter Aufbau einer Schnittdefinitionsdatei

Im Anschluss daran werden die eingelesenen Schnittparameter auf die lokalen Programmstrukturen übertragen. Für die Schnittdefinitionsdatei gelten dabei folgende Voraussetzungen:

- Definition als eine Zone
- Definition einer Schnittvariable (relativer Massenstrom oder relative Kanalhöhe) über deren Namen („ChannelHeightRelative“ oder „MassFlowRelative“)
- Schnittvektor-Dimension 0D (ein Schnitt) oder 1D (mehrere Schnitte)
- Schnittpositionen zwischen 0 und 1

Zur Verschneidung wird neben der Schnitt-Spezifikation die Vernetzung der S2M-Fläche als Basisnetz benötigt. Da der S2M-Block im Rahmen des vorangegangenen *buildFinalS2M*-Tasks von allen Prozessen erstellt worden ist, kann jeder von ihnen auf diesen zugreifen. Im Allgemeinen können für das Datenset verschiedene S2M-Flächen nebeneinander existieren, welche auf unterschiedlichen Mittelungsarten basieren, da der Benutzer den *buildFinalS2M*-Task mehrfach ausführen kann. Zur Erstellung der Schaufelschnitte ist aber der Mittelungstyp unerheblich, weil die hierfür benötigten Variablen, der rela-

tive Massenstrom und die relative Kanalhöhe, von diesem unabhängig sind.
Ebenso wie beim *intersect3DS2MPlane*-Task (*Kap. 5.4*) basiert auch beim *generateBladeCuts*-Task die Verschneidung wiederum auf den beiden dimensionslosen Koordinaten ξ und η, welche auf einen Bereich von 0 bis 100 normiert sind. Zur Vorbereitung der eigentlichen Verschneidung ist folglich für jeden Netzknoten des S2M-Schnittnetzes dessen ξ- und η-Wert zu definieren. Dabei stellt ξ die Koordinate in Meridionalrichtung und η diejenige in Normalrichtung dar. Die Schnittkoordinate entspricht somit der η-Richtung. Bei η handelt es sich daher, je nach Benutzervorgabe, um den relativen Massenstrom oder die relative Kanalhöhe.

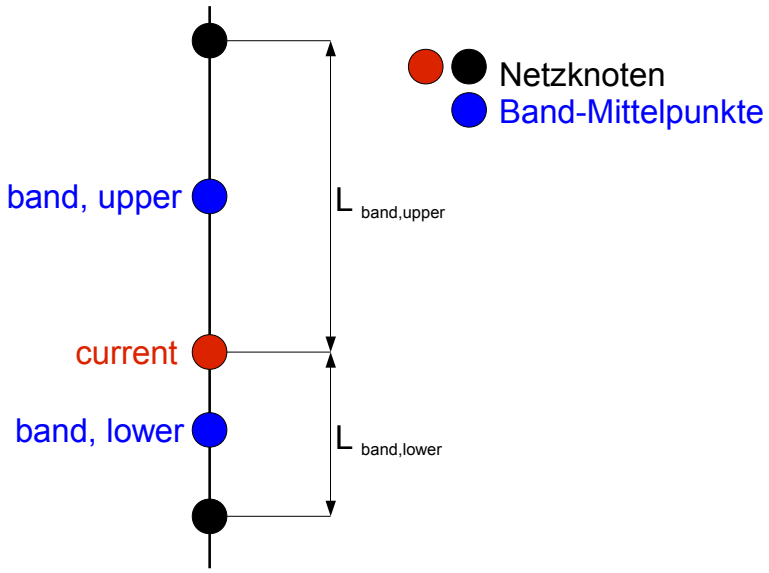


Abb. 5.24: Schema zur Interpolation der η-Werte auf die Netzknoten

Zunächst werden die η-Koordinaten der Vertices des S2M-Netzes ermittelt. Hierbei ist zu beachten, dass im vorangegangenen Task aufgrund fehlender Unterstützung seitens *Tecplot 360* der Speicherort der Strömungslösung temporär von *IFaceCenter* auf *Vertex* geändert worden ist. Die Mittelwerte der Variablen eines Bandes sind somit auf dessen nabenseitigen Vertex gespeichert. Zur Belegung des η-Felds werden den Vertices der Naben-Netzlinie jeweils der Wert 0 und denjenigen der Gehäuse-Netzlinie der Wert 100 zugewiesen. Alle inneren Vertices der Netzlinien von der Nabe zum Gehäuse werden nach folgendem Schema interpoliert (siehe *Abb. 5.24*):

$$\eta_{current} = \frac{\eta_{band,lower} \cdot L_{band,upper} + \eta_{band,upper} \cdot L_{band,lower}}{L_{band,lower} + L_{band,upper}} \tag{5.16}$$

Die ξ-Werte, welche für die Erstellung der Schaufelschnitte und S1-Stromflächen eigentlich irrelevant sind, da keine Bänder erstellt werden, werden mittels der S2M-Netzdefinition indexbasiert berechnet (siehe *Kap. 5.3.3*) und ebenfalls auf den Bereich von 0 bis 100 normiert.
Nachdem alle benötigten Informationen zur Schnittgenerierung bereitgestellt worden sind, erfolgt die Verschneidung derjenigen lokal auf einem Prozess vorhandenen Blöcke, welche zu den vom Benutzer angegebenen Familien gehören. Die Verschneidung basiert wiederum auf *INTERSEC* und der *VTK*-Bibliothek. Wie bereits erwähnt, handelt es

sich bei den zu verschneidenden Blöcken entweder um die Schaufeloberflächen, also um zweidimensionale Boundary-Blöcke, oder um dreidimensionale Blöcke. Die aus der Verschneidung jeweils resultierenden 1D- bzw. 2D-Blöcke werden wiederum dem Datenset angefügt und nach dem Ende des Tasks kommuniziert und synchronisiert. Anzumerken ist hierbei, dass ein Schaufelschnitt bzw. eine S1-Stromfläche im Allgemeinen aus mehreren Blöcken besteht. Setzt sich die Schaufeloberfläche beispielsweise aus zwei 2D-Blöcken zusammen, so besteht der resultierende Schaufelschnitt auch aus zwei einzelnen Linien, da die Verschneidung blockweise erfolgt.
Ebenso wie beim *intersect3DS2MPlane*-Task (*Kap. 5.4*) wird bei der 3D-Verschneidung für jeden η-Schnitt eine eigene Familie erstellt, deren Name dem Schnittnamen, also zum Beispiel „ISO_ETA_CUT_50.000000“, entspricht. Die neuen Blöcke weisen den Typ *S1_ANALYSIS_BLOCK* auf. Sie besitzen zudem einen Namen der Form „S1_Analysis_H_50.000“. Der Buchstabe „H“ gibt dabei an, dass die relative Kanalhöhe als Schnittvariable verwendet wurde, „M“ hingegen steht für den relativen Massenstrom. Die abschließende Zahl definiert die Position des Schnittes. Das Ergebnis des *generateBladeCuts*-Tasks für die Option „flow“ zeigt *Abb. 5.25* und das für die Option „blade“ *Abb. 5.26* sowie *Abb. 5.27*. Für beide Optionen sind zunächst die jeweiligen Interpolationstasks aufzurufen, um die Strömungslösung auf den Schnitten zu erhalten.

Die 1D-Schnittlinien werden als kartesisch definierte unstrukturierte Netze gespeichert. Die Blöcke gehören dabei zur Familie des verschnittenen 2D-Blocks. Der Blocktyp ist *BLADE_CUT_LINE* und der Zonenname wird wie folgt gebildet:

[Familienname]_V[M/H][NN.NNN]_Z[NNN]_L[N]

Dabei gilt:

- [Familienname]: entspricht dem Blockgruppen/Schaufelreihen-Familiennamen
- [M]: Schnittvariable ist der relative Massenstrom
- [H]: Schnittvariable ist die relative Kanalhöhe

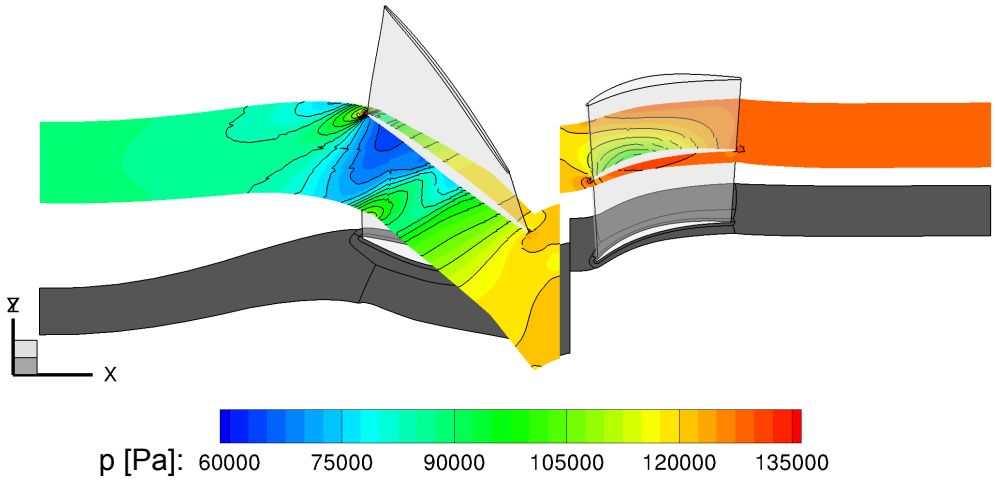


Abb. 5.25: S1-Analysefläche bei 50% relativer Kanalhöhe mit Verteilung des statischen Drucks und einer Isolinienstufung von 5000 Pa (Testfall THD R1)

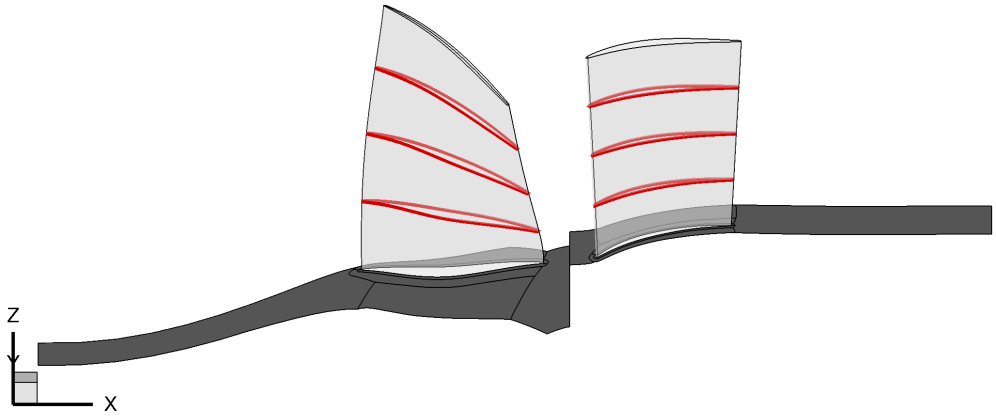


Abb. 5.26: Schaufelschnitte für beide Schaufelreihen bei 25, 50 und 75% relativer Kanalhöhe (Testfall THD R1)

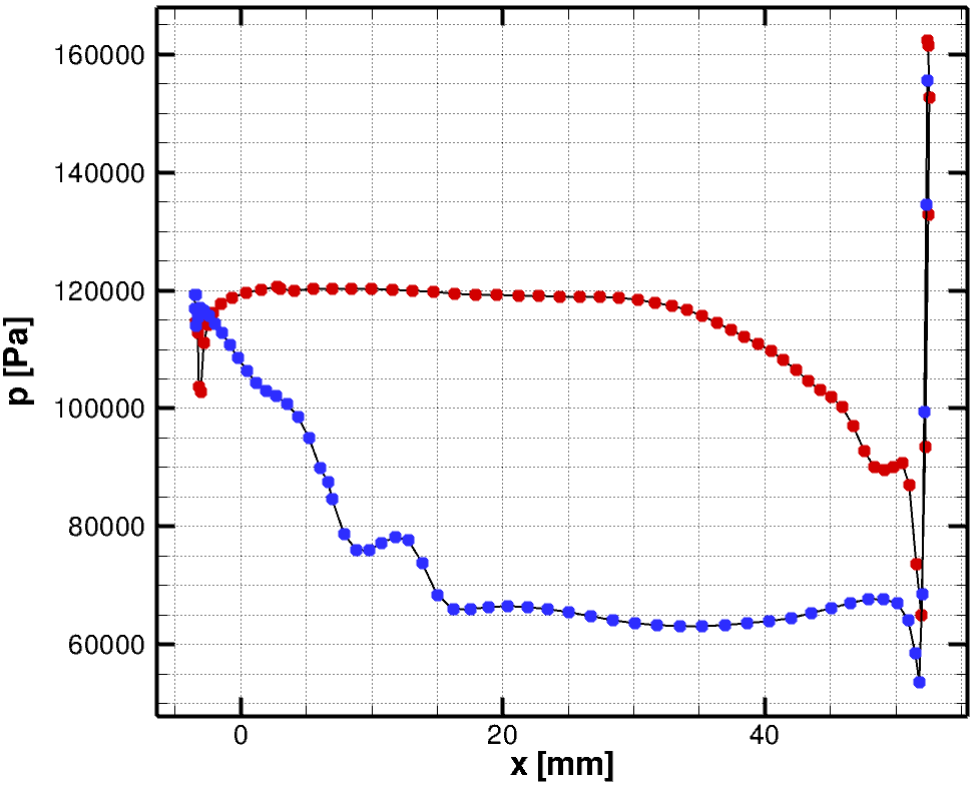


Abb. 5.27: Verlauf des statischen Drucks entlang der Rotor-Schaufeloberfläche bei 50% relativer Kanalhöhe (THD Rotor 1)

- [NN.NNN]: entspricht der η-Schnittposition
- Z[NNN]: entspricht der ID des verschnittenen Blocks
- L[N]: entspricht der Nten Schnittlinie des Blocks

Diese Definition gewährleistet die Eindeutigkeit des Zonennamens. Falls dieser Name die maximale Zeichenzahl von 32 überschreitet, wird der Blockgruppen-Familienname entsprechend gekürzt.

Name	finishBladeCuts		
Aufruf	-fBlC		
Optionen	-sol	Name der Lösung	obligatorisch
	-avg	area/mass/flux	obligatorisch

Tabelle 5.18: Übersicht über Aufruf und Optionen des *finishBladeCuts*-Tasks

Im Anschluss an den Task *generateBladeCuts* erfolgt jeweils eine Interpolation, um die benötigten Strömungsvariablen auf die neu generierten Schnittflächen und -linien zu bekommen. Für die S1-Stromflächen wird der *interpolate2D*-Task aufgerufen und für die Schaufel-Linienschnittsegmente der *interpolate1D*-Task. Wie bereits erwähnt, handelt es sich bei der implementierten Variante des *interpolate2D*-Tasks um eine lineare Interpolation. Um die Werte der Variablen des 3D-Felds auf die S1-Stromflächen zu interpolieren, werden dabei die Gradienten über die Methode der kleinsten Quadrate bestimmt. Damit ergibt sich ein resultierender Interpolationsfehler 2.Ordnung.
Da der *interpolate1D*-Task bei Abschluss der Arbeit noch nicht zur Verfügung stand, werden für jede Edge der Schnittlinien die Variablen derjenigen 2D-Zelle des Elternblocks kopiert, aus welcher die Edge im Verschneidungsprozess entstanden ist. Für die Schaufelschnitte folgt im Anschluss der Task *finishBladeCuts* (siehe *Tab. 5.18*). Im Rahmen dieses Tasks werden für die 1D-Netzlinien zusätzlich die Verteilungen des Druckbeiwerts c_p und der isentropen Mach-Zahl Ma_{is} bestimmt. Hierfür werden folgende Formeln verwendet:

$$c_p = \frac{p - p_{inlet}}{p_{t,inlet} - p_{inlet}} \tag{5.17}$$

$$Ma_{is} = \sqrt{\frac{2}{\gamma - 1} \cdot \left(\left(\frac{p_{t,inlet}}{p} \right)^{\frac{\gamma}{\gamma - 1}} - 1 \right)} \tag{5.18}$$

Beide Kennzahlen benötigen den Referenzdruck am Inlet der jeweiligen Schaufelreihe. Deshalb erfolgt zunächst eine Sortierung aller für das Datenset vorhandenen 1D-Blöcke mit dem Typ *BLADE_CUT_LINE* nach der zugehörigen Schaufelreihe. Die Schaufeloberflächen dieser Schaufelreihe können beliebig oft verschnitten worden sein, sodass eine weitere Ordnung der Blöcke nach deren η-Schnittposition erfolgt. Am Ende der Sortierroutine befinden sich dann alle lokalen Blöcke eines η-Schnitts einer Schaufelreihe in einem eigenen Array.
Daraufhin wird für die einzelnen Schaufelreihen der jeweilige Inlet ermittelt, bei welchem es sich meist um ein Interface handelt. Hierzu wird im ersten Schritt die Inletfamilie bzw. die Inlet-Schaufelreihe der Gesamtkonfiguration gesucht. Über das Outlet-Interface

dieser Blockgruppe kann dann das Inlet-Interface der folgenden Blockgruppe ermittelt werden. Der Algorithmus springt solange von Interface zu Interface bis dasjenige gefunden ist, welches zur Blockgruppe des aktuellen Schnittlinien-Arrays gehört. Bei diesem handelt es sich dann um die benötigte Inletfamilie der aktuellen Schaufelreihe. Diese Vorgehensweise versagt jedoch dann, wenn mehr als eine Blockgruppe auf eine vorhergehende folgt. Splitter-Konfigurationen sind somit momentan nicht behandelbar. Wie bereits im Rahmen des *globalTurbomachineAnalysis*-Tasks erwähnt, wird der Benutzer zukünftig die Inlet-Panelfamilien der Schaufelreihen über *GMC* vorgeben können. Die Suche der Inlet-Interfaces der Schaufelreihen würde somit entfallen.
Zur Ermittlung der Referenzdrücke am Einlass der Schaufelreihe ist dann zu entscheiden, welche der IntegralValues-Strukturen zu verwenden ist. Hierzu ist vom Benutzer über „-sol“ der Name der Lösung, meist „FlowSolution“, anzugeben und über „-avg area/mass/flux“ der gewünschte Mittelungstyp der Referenzgrößen. Falls für den vorgegebenen Mittelungstyp die Mittelung fehlgeschlagen ist, erfolgt automatisch das Umschalten von Fluss- auf Massen- und von dieser auf Flächenmittelung. Folglich ist es sinnvoll in *Kap. 5.5.2* mehrere Mittelungsarten durchzuführen, zumindest aber die Flächenmittelung, um ein Umschalten zu ermöglichen.
Da zum aktuellen Zeitpunkt in *POST 7* noch keine Bandstruktur für Interfaces aufgebaut wird, wird zur Berechnung von c_p und Ma_{is} auf panelintegrale Referenzdrücke zurückgegriffen. Sobald die Bandstruktur jedoch implementiert ist, können die Referenzwerte für jeden η-Schnitt basierend auf den bandintegralen Werten des Interfaces interpoliert werden. Für den Schaufelschnitt bei $\eta = 50$ würden also die Referenzdrücke des Inlet-Interfaces ebenfalls bei $\eta = 50$ verwendet werden. Die Schnittposition sowie die verwendete Schnittvariable, der relative Massenstrom oder die relative Kanalhöhe, sind dabei dem jeweiligen Blocknamen der 1D-Schnittlinien zu entnehmen.

Kapitel 6

Validierung der Implementierung

Im Rahmen dieses Kapitels erfolgt die Validierung der Implementierung der automatisierten globalen Turbomaschinen-Analyse. Dies beinhaltet vor allem den Nachweis der Lauffähigkeit sowie die Überprüfung der Sinnhaftigkeit der Resultate für die fünf Testfälle neben dem THD R1, dessen Ergebnisse bereits im Rahmen des vorigen Kapitels beispielhaft vorgestellt wurden. Im Einzelnen werden eine Axialturbinen-, zwei Axialverdichter-, eine Radialverdichter- und eine Fan-Konfiguration betrachtet. Eine der beiden Axialverdichter-Konfigurationen weist dabei eine nicht umfangssymetrische Nabenkonturierung auf. Die Testfälle wurden dabei mit unterschiedlichen Versionen von *TRACE* gerechnet. Ausgewertet werden jeweils stationäre Strömungsfelder.
Die Validierungsrechnungen für jeden der Testfälle umfassen den Gesamtprozess der Turbomaschinen-Analyse. Zunächst wird also ausgehend von der eingelesenen dreidimensionalen CGNS-Datei die automatische Erstellung des S2M-Schnittnetzes durchgeführt, auf dessen Basis im Anschluss die Verschneidung der 3D-Konfiguration erfolgt. Daraufhin schließen sich die Mittelungsroutinen an, welche jeweils für alle drei angebotenen Mittelungsarten mit den Default-Einstellungen ausgeführt werden. Dann folgt der Aufbau der finalen S2M-Flächen, ebenfalls für alle drei Mittelungsarten. Auf deren Basis wiederum werden für alle Schaufeln der jeweiligen Konfiguration bei 25, 50 und 75% relativer Kanalhöhe Schaufelschnitte erstellt. Für die gleichen Schnittpositionen werden für alle Blockgruppen beziehungsweise Schaufelreihen zudem S1-Stromflächen generiert. Die relevanten Resultate aller Tasks werden abschließend in eine CGNS-Datei geschrieben und mit *Tecplot 360* ausgewertet. Weniger wichtige Zwischenergebnisse wie zum Beispiel die θ-Schnittlinien des *generateThetaCuts*-Tasks werden hingegen vor dem *output*-Task mit Hilfe des *deleteBlocks*-Task aus dem Datenset entfernt.
Anzumerken ist, dass sowohl der *globalTurbomachineAnalysis*-Task als auch der *finishBladeCuts*-Task nicht ausgeführt und damit auch nicht getestet werden können, da zum Zeitpunkt der Fertigstellung der Arbeit die hierfür benötigten Interface-Informationen in *POST 7.5*, für welches die globale Turbomaschinen-Auswertung implementiert wurde, noch nicht zur Verfügung standen. Für den *GTA*-Task erfolgte bereits für *POST 7.3* eine Validierung anhand mehrerer Testfälle. Die 7.3er Variante basiert jedoch auf einem *TRACE*-Initialisierungsschritt, wohingegen die neue 7.5er Variante komplett auf *POST*-Strukturen arbeitet.

In der folgenden Auswertung werden aufgrund der generierten Datenmenge nicht alle Ergebnisse jedes Testfalls vorgestellt, sondern es erfolgt eine Beschränkung auf die jeweils relevanten Ergebnisse, insbesondere die S2M-Flächen. Ebensowenig wird auf radiale Verteilungen und Schaufeldruckverläufe eingegangen. Diese wurden für den Testfall THD R1 bereits in *Abb. 5.19* und *Abb. 5.20* sowie *Abb. 5.27* betrachtet. Im selben Kapitel (*Kap. 5.5.3*) erfolgte auch ein Vergleich der Mittelungsergebnisse von *POST* und *TRACE*.
Die Rechnungen der globalen Turbomaschinen-Analyse wurden alle von jeweils einem Kern des DLR-Clusters ausgeführt, da die momentane Implementierung noch nicht für eine parallele Rechnung auf mehreren Kernen lauffähig ist. Die wichtigsten Charakteristika des DLR-Clusters sind in *Tab. 6.1* aufgeführt. Sowohl die Simulation mit *TRACE* als auch das Postprocessing mit *POST* erfolgte dabei jeweils für eine Schaufelpassage. Zur 3D-Darstellung der Geometrie der Testfälle werden jeweils zwei Schaufelpassagen gezeigt. Diese Duplizierung wurde vom *duplicateData*-Task von *POST 7.5* durchgeführt.

RAM		74,16 GB
CPU	Name	Intel Xeon E5540
	Taktung	2,53 GHz
	Cache	8192 kB

Tabelle 6.1: Charakteristika des DLR-Clusters

6.1 Validierung anhand des Testfalls „LISA“

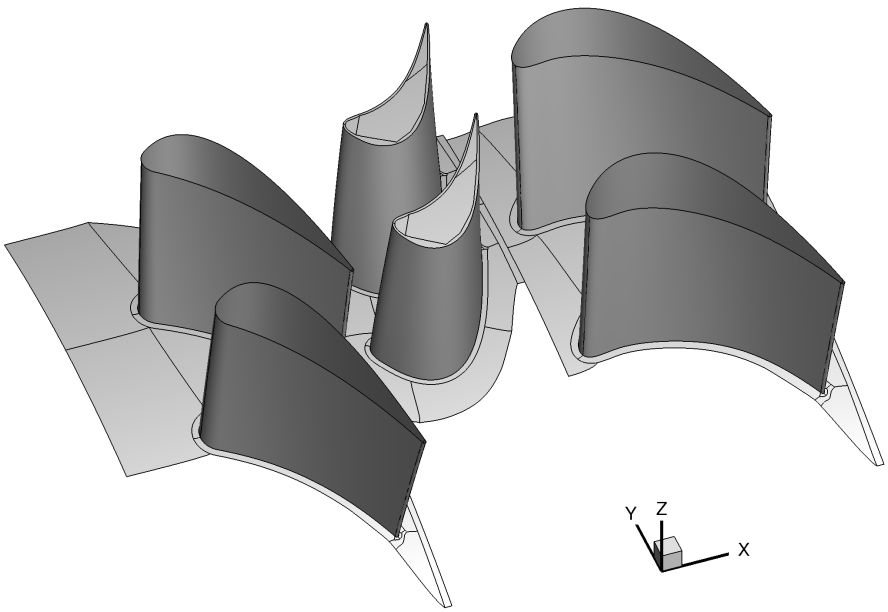


Abb. 6.1: 3D-Darstellung der LISA-Axialturbine

Dieser Abschnitt behandelt die Validierung der Implementierung der Turbomaschinen-Analyse anhand des Testfalls LISA. Dessen 3D-Ansicht ist in *Abb. 6.1* dargestellt, weitere Informationen, wie zum Beispiel die *TRACE*-Version, mit welcher die Simulation durchgeführt wurde, sind *Tab. 6.2* zu entnehmen. Der Kommandozeilenaufruf lautet wie folgt:

> POST -i TRACE.cgns -gtc -hub row01Hub__Wall row02Hub__Wall row03Hub__Wall -tip row01Tip__Wall row02Tip__Wall row03Tip__Wall -nCuts 18 -cGeomI -task geo -geo -type max -dB -type cutHub -dB -type cutTip -genS2 -xieta length -cGeomI -task i3DS2 -i3DS2 -cconv -ip2D -avgf -avgm -avga -avgi -dB -type analysis2d -bS2m -sol FlowSolution -avg area -switch band -bS2m -sol FlowSolution -avg mass -switch band -bS2m -sol FlowSolution -avg flux -switch band -genBlC -load 1dcutBlade.dat -option blade -family row01Blade_Wall row02Blade_Wall row03Blade_Wall -genBlC -load 1dcutFlow.dat -option flow -family S1 R1 S2 -ip2D -o LISA_LENGTH_COMPLETE.cgns

Typ	1,5-stufige Axialturbine
verwendete *TRACE*-Version	7.3
Zellenzahl des Rechennetzes	600.496
Blockzahl des Rechennetzes	35
***POST*-Rechenzeit**	ca. 8 min

Tabelle 6.2: Übersicht des Testfalls LISA

Im ersten Schritt werden die Routinen zur automatischen Generierung des S2M-Schnittnetzes aufgerufen (*Kap. 5.3*), auf welchem die anschließende Verschneidung der 3D-Konfiguration beruht. Das Ergebnis für das Netz zeigt *Abb. 6.2*. Dieses verfügt über 782 Punkte in Meridionalrichtung und über 72 Punkte in Normalrichtung. Die letzte Zahl ist vorgegeben, wohingegen die Punktzahl in meridionaler Richtung von der Auflösung der verschnittenen Blöcke der Nabe- und Gehäusefamilien abhängt.

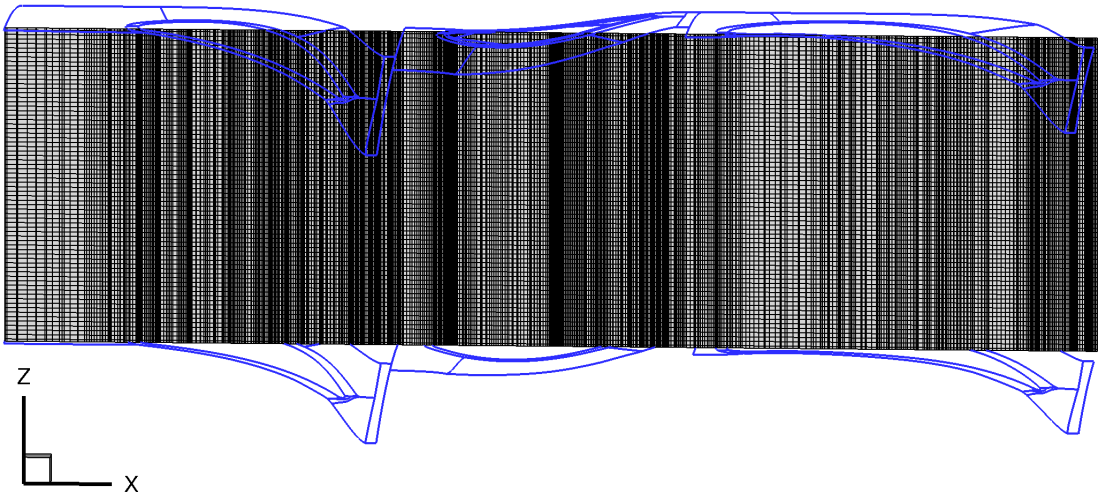


Abb. 6.2: Automatisch generiertes S2M-Schnittnetz (Testfall LISA)

Abb. 6.3 und *Abb. 6.4* zeigen die ξ-Verteilung des temporären S2M-Netzes für die beiden möglichen Optionen (längen- bzw. indexbasiert). Wie in *Kap. 5.3.3* bereits dargestellt, wird bei der längenbasierten Variante der ξ-Wert jedes Netzpunkts als dessen Abstand in Laufrichtung vom ersten Punkt derselben Netzlinie geteilt durch die Gesamtlänge dieser Netzlinie definiert. Bei der indexbasierten Variante entspricht der ξ-Wert jeweils dem Index des Punkts in Meridionalrichtung geteilt durch die Gesamtanzahl an Punkten der Netzlinie, für den LISA-Testfall also 782. Wie ein Vergleich beider Varianten zeigt, bewirkt die indexbasierte Variante eine stärkere Gewichtung von Bereichen höherer Punktdichte. Dies hat zur Folge, dass in diesen Bereichen mehr Schnittflächen liegen werden, wohingegen sich für die längenbasierte Variante eine äquidistante Verteilung ergibt. Die geometrische Lage der Analyseflächen wird dabei durch die ξ-Isolinien (Stufung $\Delta\xi = 5$) dargestellt.

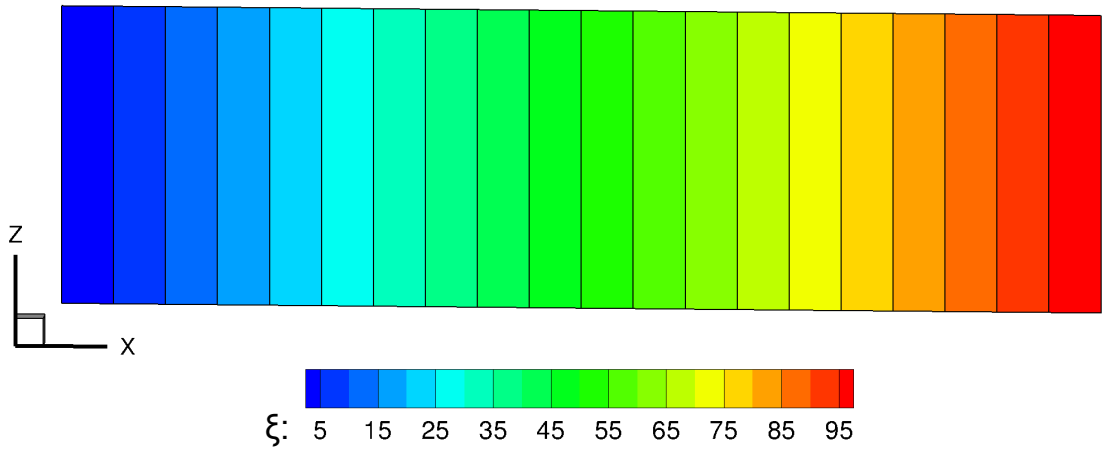


Abb. 6.3: Längenbasierte ξ-Verteilung des S2M-Schnittnetzes (Testfall LISA)

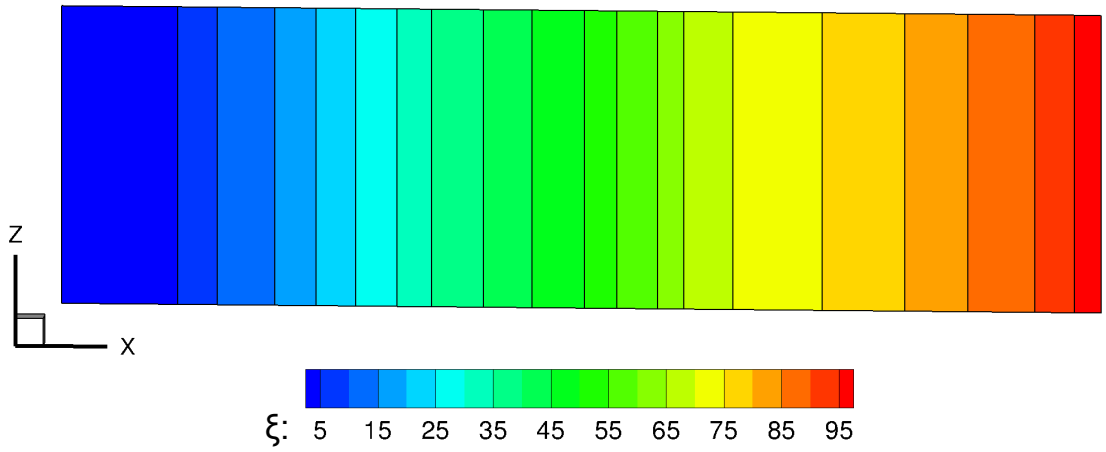


Abb. 6.4: Indexbasierte ξ-Verteilung des S2M-Schnittnetzes (Testfall LISA)

Abb. 6.5 und *Abb. 6.6* zeigen schließlich das resultierende S2M-Netz, wiederum für beide ξ-Varianten, mit Verteilung der flächengemittelten Mach-Zahl im Absolutsystem. Beide verfügen über 101 Schnittebenen beziehungsweise Netzlinien in meridionaler Richtung und über 71 Bänder bzw. 72 Netzlinien in normaler Richtung.

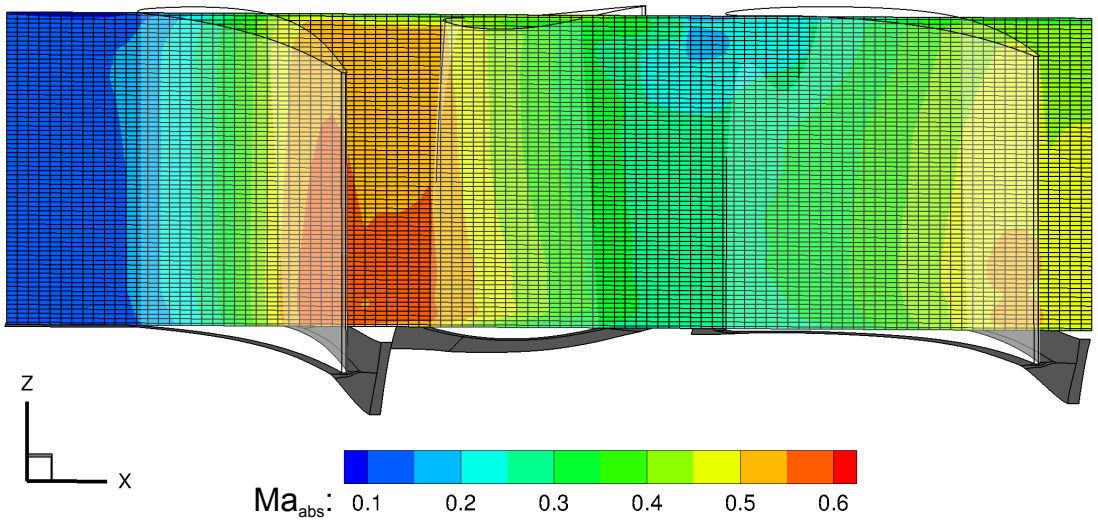


Abb. 6.5: Resultierende S2M-Fläche für die längenbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall LISA)

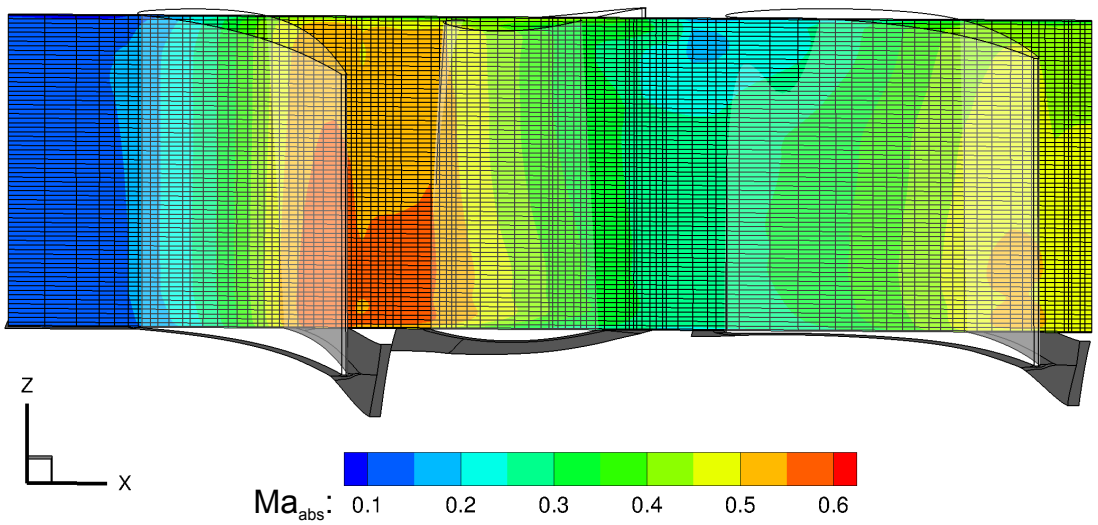


Abb. 6.6: Resultierende S2M-Fläche für die indexbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall LISA)

Abb. 6.7 stellt einen Vergleich der umfangsgemittelten Strömungslösung für die drei angebotenen Mittelungsarten am Beispiel der Verteilung der absoluten Mach-Zahl dar. Hieraus wird ersichtlich, dass die drei Optionen vergleichbare, jedoch nicht identische Resultate liefern.
Auffällig sind die beiden Punkte beziehungsweise Sprünge im Niveau der Mach-Zahl bei der Flussmittelung. Diese Sprünge entstehen, da an diesen Stellen jeweils ein Band von einem Interface geschnitten wird. Somit befindet sich ein Teil dieses Bandes im drehenden und ein Teil im ruhenden Bezugssystem. Der Fehler ergibt sich dann dadurch, dass für das gesamte Band nur eine Drehzahl kommuniziert werden kann, mit welcher der Strömungszustand im Absolutsystem aus demjenigen im Relativsystem ermittelt wird beziehungsweise mit welcher die Berechnung von $v_{\theta,abs}$ aus $v_{\theta,rel}$ erfolgt.

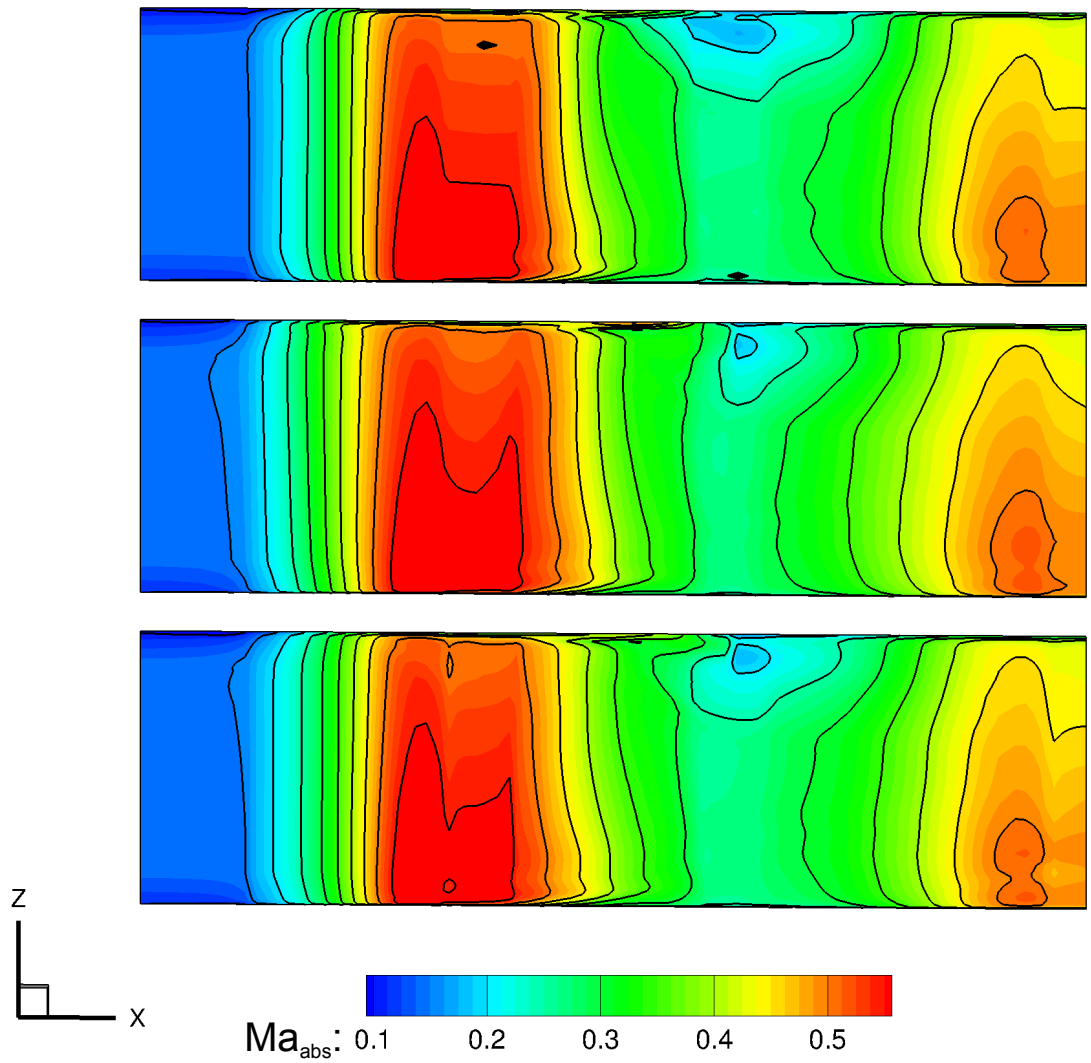


Abb. 6.7: Vergleich der Mach-Zahl-Verteilung der resultierenden S2M-Flächen für, von oben nach unten, Fluss-, Massen- und Flächenmittelung mit einer Isolinienstufung von $\Delta Ma_{abs} = 0,05$ (Testfall LISA)

Bei der Flächen- und Massenmittelung tritt dieses Problem nicht auf, da bereits für alle 2D-Zellen jedes Bands beziehungsweise jedes Blocks alle Variablen, welche den Strö-

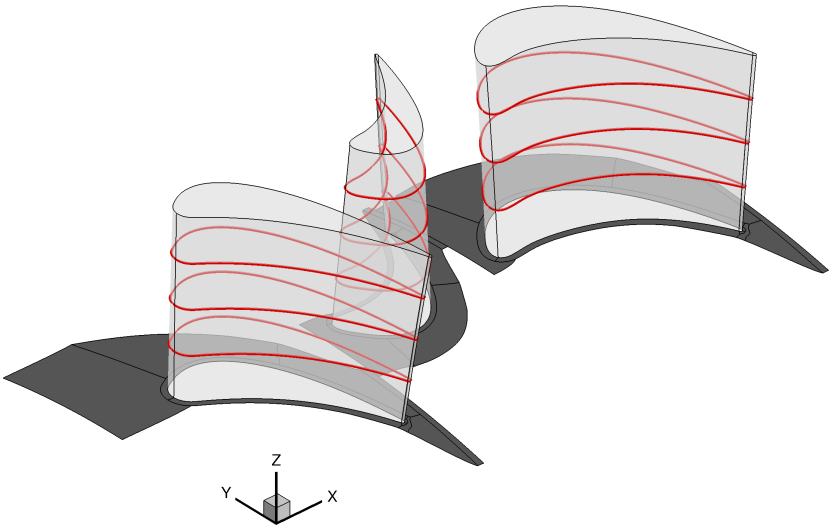


Abb. 6.8: Schaufelschnitte bei 25, 50 und 75% relativer Kanalhöhe (Testfall LISA)

p [Pa]:

Abb. 6.9: S1-Stromfläche bei 25% relativer Kanalhöhe mit Verteilung des statischen Drucks (Testfall LISA)

mungszustand beschreiben, sowohl im Absolutsystem als auch im Relativsystem, ermittelt werden. Denn jedem Block ist eine eindeutige Drehzahl zugeordnet. Die Schnittfamilien setzen sich aber aus mehreren dieser Blöcke zusammen und besitzen somit keine eindeutige Drehzahl mehr. Dasselbe gilt für ein Band, welches sich über ein Interface hinweg erstreckt. Weitere Informationen zur Implementierung der Mittelung sind *Kap. 5.5* zu entnehmen.
Auf Basis der für die S2M-Flächen zusätzlich abgeleiteten Größen, der relativen Kanalhöhe sowie dem relativen Massenstrom, erfolgt eine weitere Verschneidung der Konfiguration, um S1-Stromflächen und Schaufelschnitte zu generieren (*Kap. 5.8*). Für die Schaufelschnitte sind dabei die zu verschneidenden 2D-Schaufeloberflächen und für die S1-Stromflächen die 3D-Blockgruppen über ihren jeweiligen Familiennamen vorzugeben. Gleichzeitig muss eine ASCII-Schnittdefinitionsdatei angegeben werden, welche die Schnittvariable, die Anzahl der Schnitte sowie deren Positionen als dimensionslose η-Werte enthält. *Abb. 6.8* zeigt das Ergebnis für die Schaufelschnitte und *Abb. 6.9* das für die S1-Stromfläche. Für Erstere kann schließlich noch der c_p- und Ma_{is}-Verlauf bestimmt werden, nachdem die Interpolation der Strömungsvariablen auf die 1D-Schnittlinien erfolgt ist. Zudem sind beispielsweise Schaufeldruckverläufe darstellbar.

6.2 Validierung anhand des Testfalls „Rig250K“

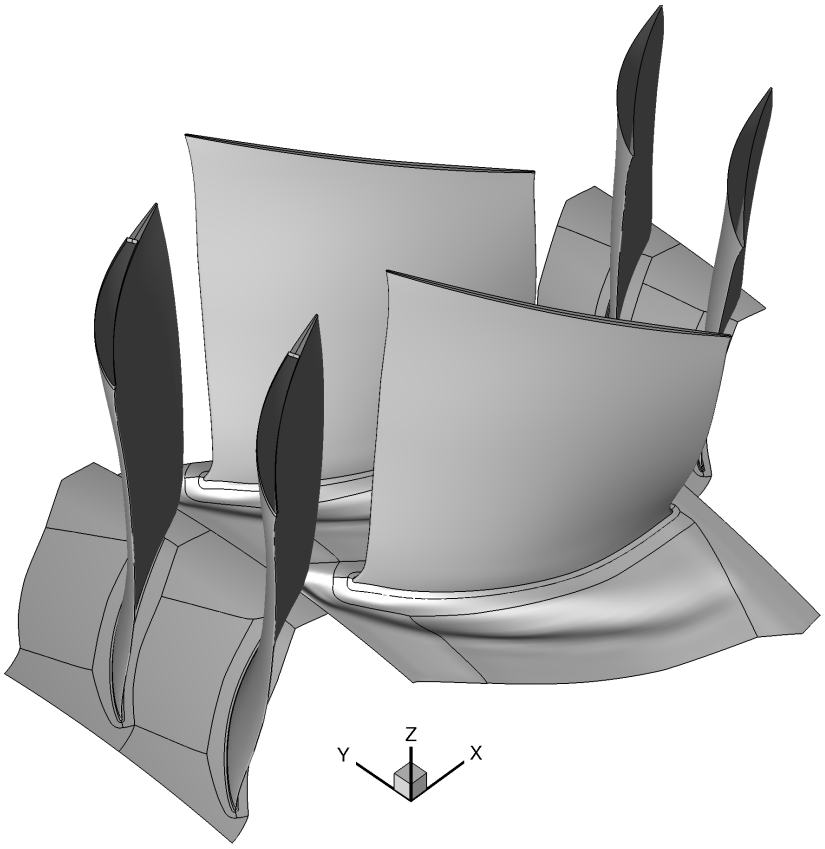


Abb. 6.10: 3D-Darstellung des Rig250-Axialverdichters mit Nabenkonturierung

Im Rahmen des folgenden Unterabschnitts wird die Validierung der Implementierung der globalen Turbomaschinen-Auswertung anhand des Testfalls Rig250 mit Nabenkonturierung (Rig250K), dargestellt in *Abb. 6.10*, betrachtet. Bei diesem handelt es sich um die 5. - 7. Schaufelreihe des Testfalls Rig250, welcher im Anschluss vorgestellt wird. Die Nabenkonturierung ist dabei nicht umfangssymetrisch. Weitere Informationen zu diesem Testfall sind *Tab. 6.3* zu entnehmen. Der Kommandozeilenaufruf lautet:

> POST -i TRACE.cgns -gtc -hub row05Hub_Wall row06Hub_Wall row07Hub_Wall -tip row05Tip_Wall row06Tip_Wall row07Tip_Wall -nCuts 30 -cGeomI -task geo -geo -type max -genS2 -reference hub -xieta length -dB -type cuttip -dB -type cuthub -cGeomI -task i3DS2 -i3DS2 -cconv -ip2D -avga -avgm -avgf -avgi -dB -type analysis2d -bS2m -sol FlowSolution -avg area -switch band -bS2m -sol FlowSolution -avg mass -switch band -bS2m -sol FlowSolution -avg flux -switch band -genBlC -load 1dcutBlade.dat -option blade -family row05Blade_Wall row06Blade_Wall row07Blade_Wall -genBlC -load 1dcutFlow.dat -option flow -family row05_VANE01 row06_BLADE01 row07_VANE01 -ip2D -o RIG250K_COMPLETE_LENGTH.cgns

Typ	1,5-stufiger Axialverdichter
verwendete *TRACE*-Version	6.7
Zellenzahl des Rechennetzes	1.946.298
Blockzahl des Rechennetzes	29
***POST*-Rechenzeit**	ca. 13 min

Tabelle 6.3: Übersicht des Testfalls Rig250K

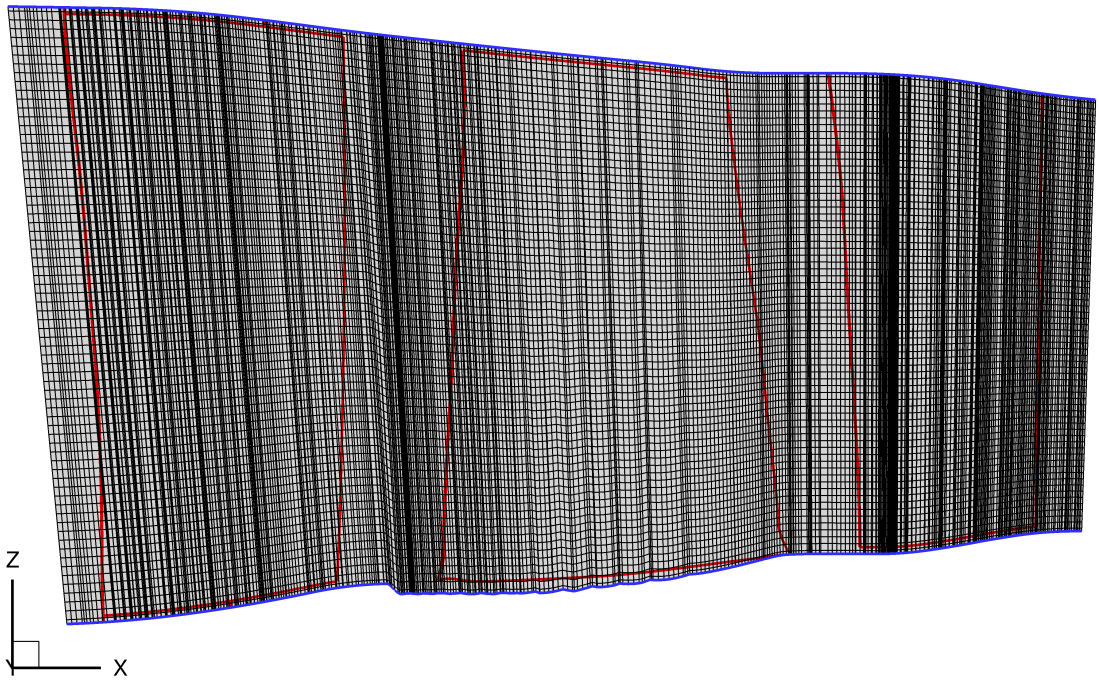


Abb. 6.11: Automatisch generiertes S2M-Schnittnetz (Testfall Rig250K)

Im ersten Schritt wird das S2M-Schnittnetz unter Benutzung der Maximalkontur-Option generiert. Hierbei werden abweichend vom Defaultwert 30 θ-Schnitte gesetzt, um die Nabenkonturierung möglichst genau zu erfassen (*Kap. 5.3.1*). Zudem wird als Referenzlinie die Nabenlinie spezifiziert, um sicherzustellen, dass deren Auflösung beziehungsweise deren Verlauf durch ein Remapping nicht verfälscht wird (*Kap. 5.3.3*). Insgesamt weist das Schnittnetz 574 Netzpunkte in meridionaler Richtung auf. Das Ergebnis der automatischen Netzgenerierung zeigt *Abb. 6.11*.
Abb. 6.12 und *Abb. 6.13* veranschaulichen die resultierende ξ-Verteilung für die längen- und indexbasierte Variante. Auffällig ist hierbei die starke Punktverdichtung im Bereich um $\xi \approx 70$ der indexbasierten Variante beziehungsweise an der entsprechenden Stelle im S2M-Schnittnetz. Diese resultiert wahrscheinlich weniger aus der Punktverteilung der verschnittenen Naben-Panelfamilie, sondern vielmehr dadurch, dass sich in diesem Bereich häufig Linien schneiden und somit die Punktdichte durch die Speicherung der ermittelten Linienschnittpunkte erhöht wird (*Kap. 5.3.2*).

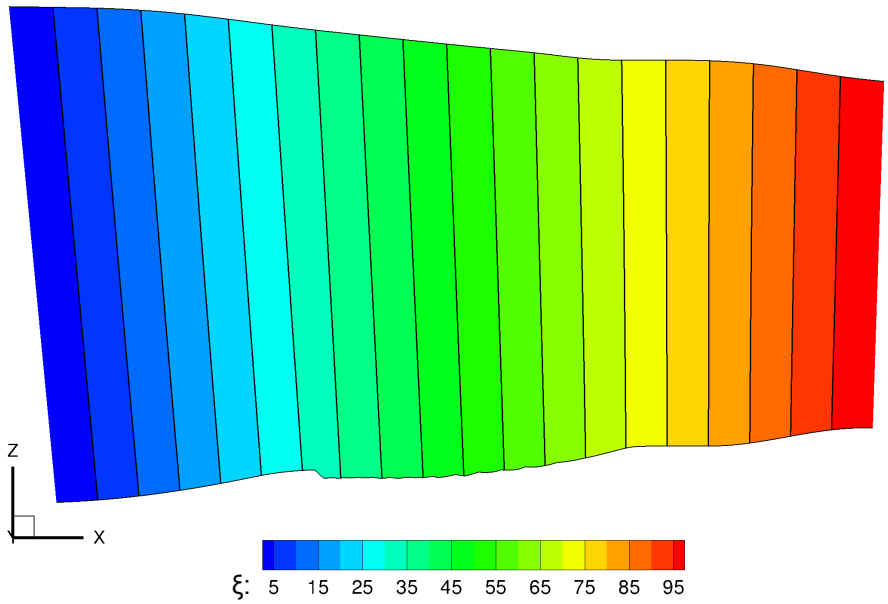


Abb. 6.12: Längenbasierte ξ-Verteilung des S2M-Schnittnetzes - Isolinienstufung $\Delta\xi = 5$ (Testfall Rig250K)

Die anschließende Verschneidung der 3D-Konfiguration erfolgt unter Verwendung der Standardwerte von 101 Schnitten in meridionaler und 71 Bändern in normaler Richtung. Eine dieser Schnittebenen beziehungsweise Analysefamilien im Bereich der Nabenkonturierung zeigt *Abb. 6.14*.
In *Abb. 6.15* und *Abb. 6.16* sind schließlich die resultierenden S2M-Flächen, welche jeweils auf flächengemittelten Werten basieren, dargestellt. Wie anhand der in die *xr*-Ebene projizierten Schaufeloberflächen ersichtlich ist, wird die Nabenkonturierung der mittleren Schaufelreihe sehr gut erfasst. *Abb. 6.17* zeigt wiederum den Vergleich der drei implementierten Mittelungsarten, nämlich der Flächen-, der Massen- und der Flussmittelung. Wie schon beim Testfall LISA zeigen sich auch hier die Sprünge des Niveaus der Mach-Zahl im Bereich der Interfaces. Wie bereits erwähnt, resultieren diese Fehler daraus, dass sich an diesen Stellen jeweils ein Band über ein Interface erstreckt und es

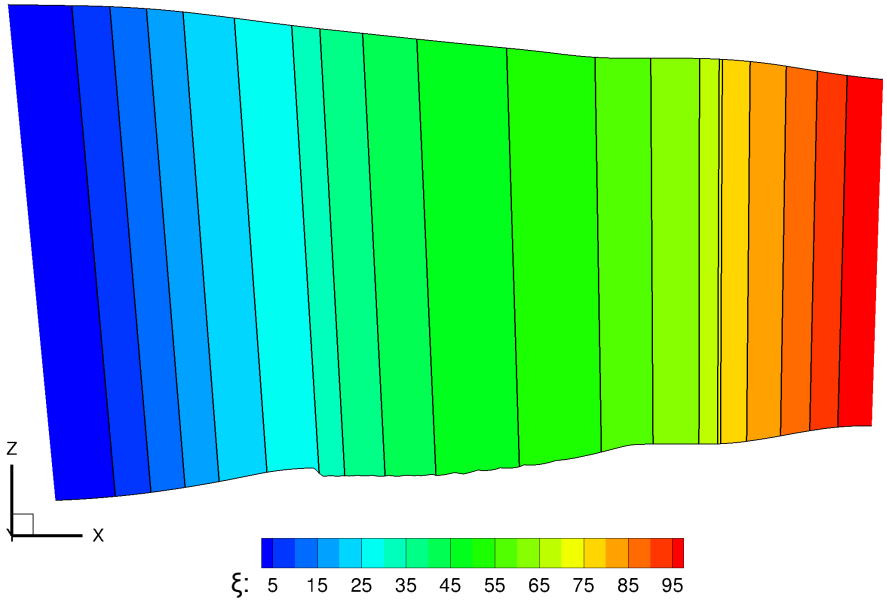


Abb. 6.13: Indexbasierte ξ-Verteilung des S2M-Schnittnetzes - Isolinienstufung $\Delta\xi = 5$ (Testfall Rig250K)

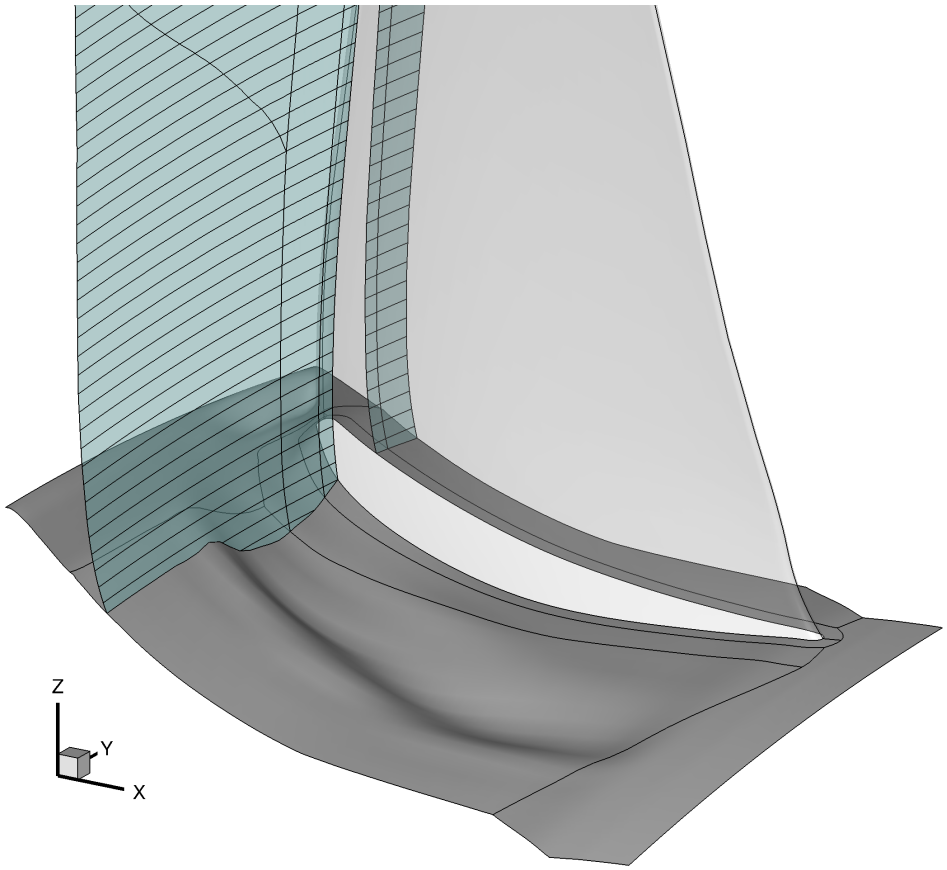


Abb. 6.14: Analysefamilie im Bereich der Nabenkonturierung (Testfall Rig250K)

somit nicht mehr eindeutig im ruhenden oder rotierenden Bezugssystem liegt.

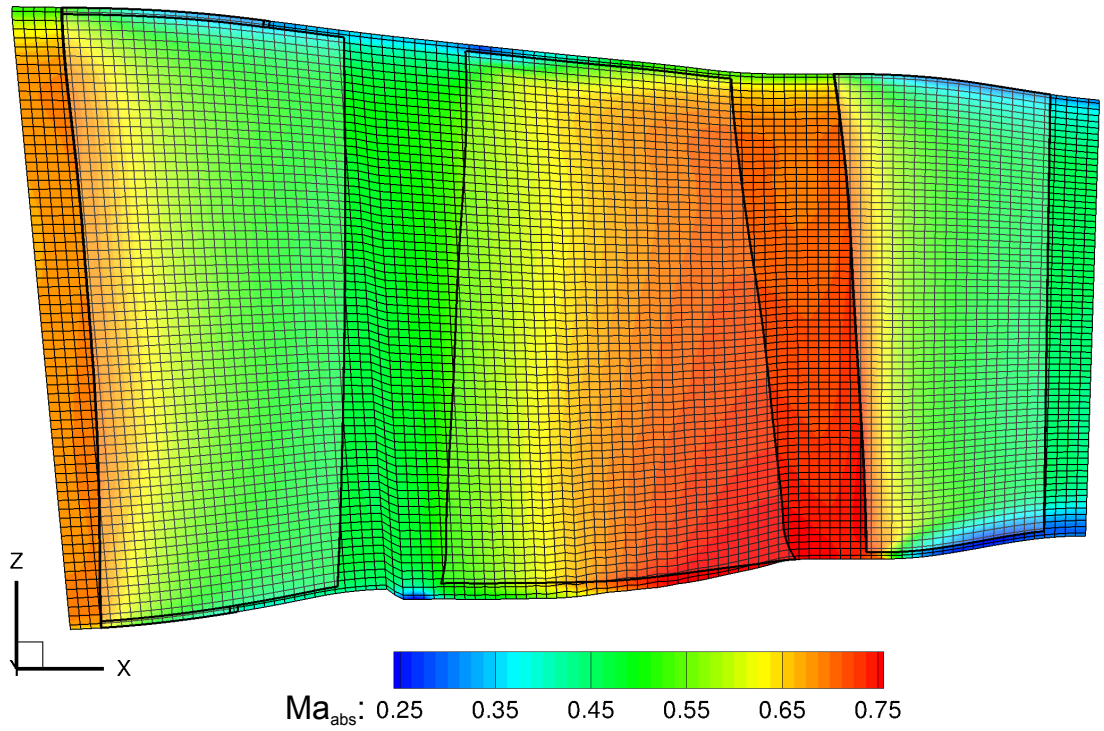


Abb. 6.15: Resultierende S2M-Fläche für die längenbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall Rig250K)

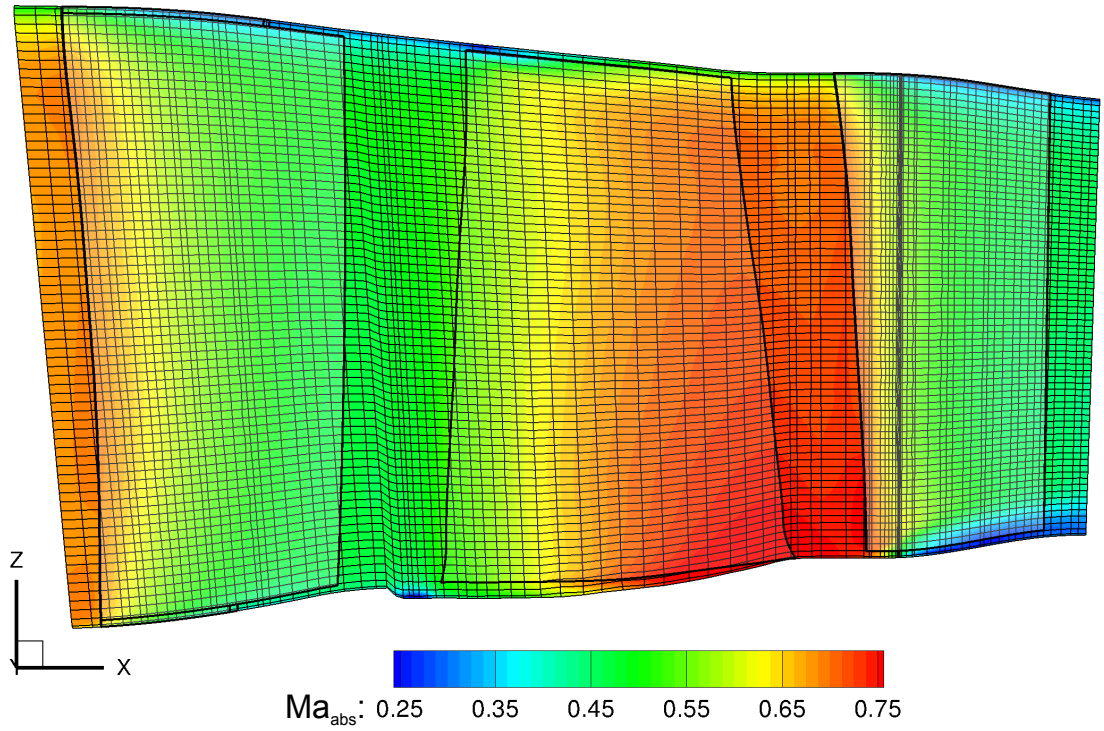


Abb. 6.16: Resultierende S2M-Fläche für die indexbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall Rig250K)

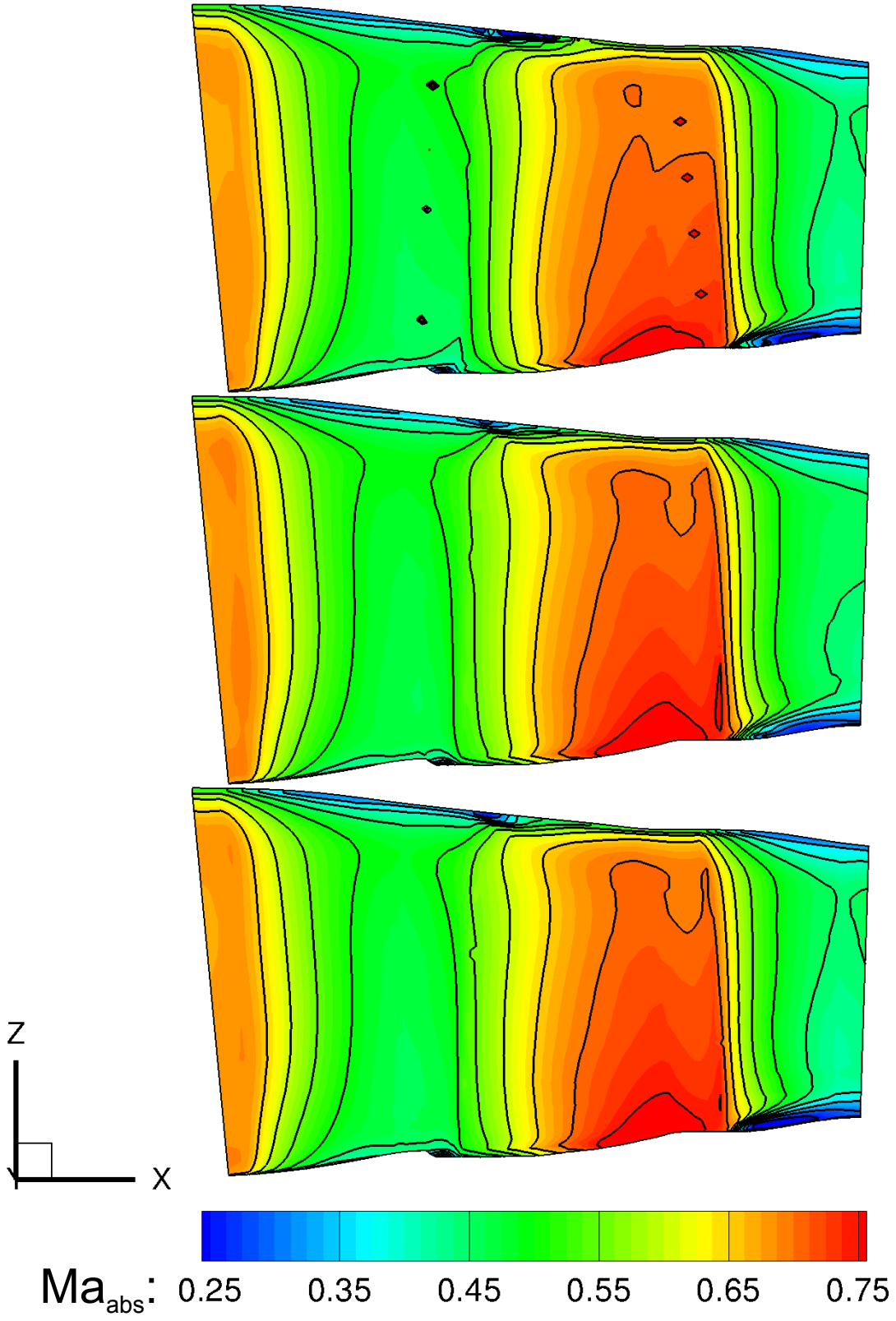


Abb. 6.17: Vergleich der Mach-Zahl-Verteilung der resultierenden S2M-Flächen für, von oben nach unten, Fluss-, Massen- und Flächenmittelung mit einer Isolinienstufung von $\Delta Ma_{abs} = 0,05$ (Testfall RIG250K)

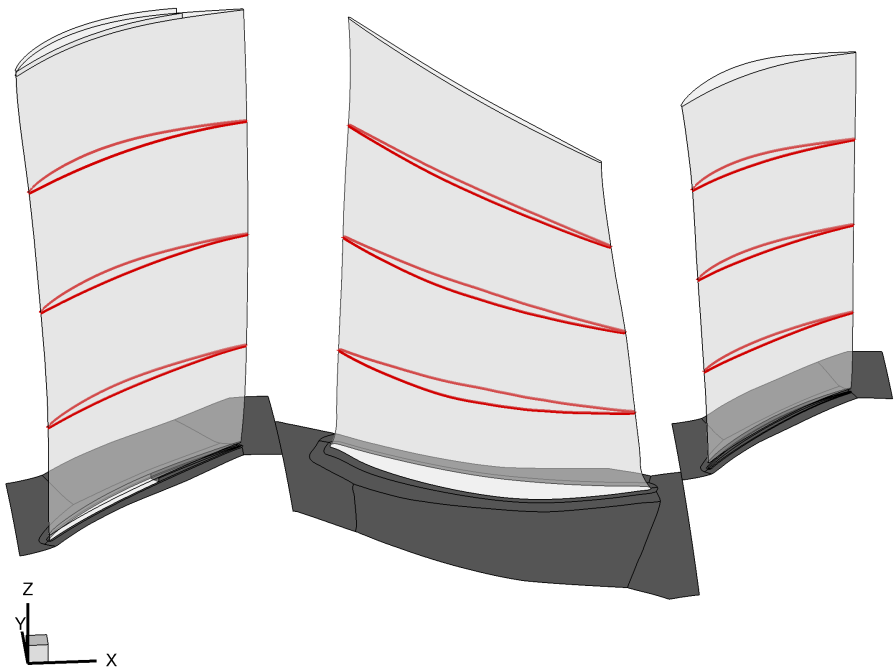


Abb. 6.18: Schaufelschnitte bei 25, 50 und 75% relativer Kanalhöhe (Testfall RIG250K)

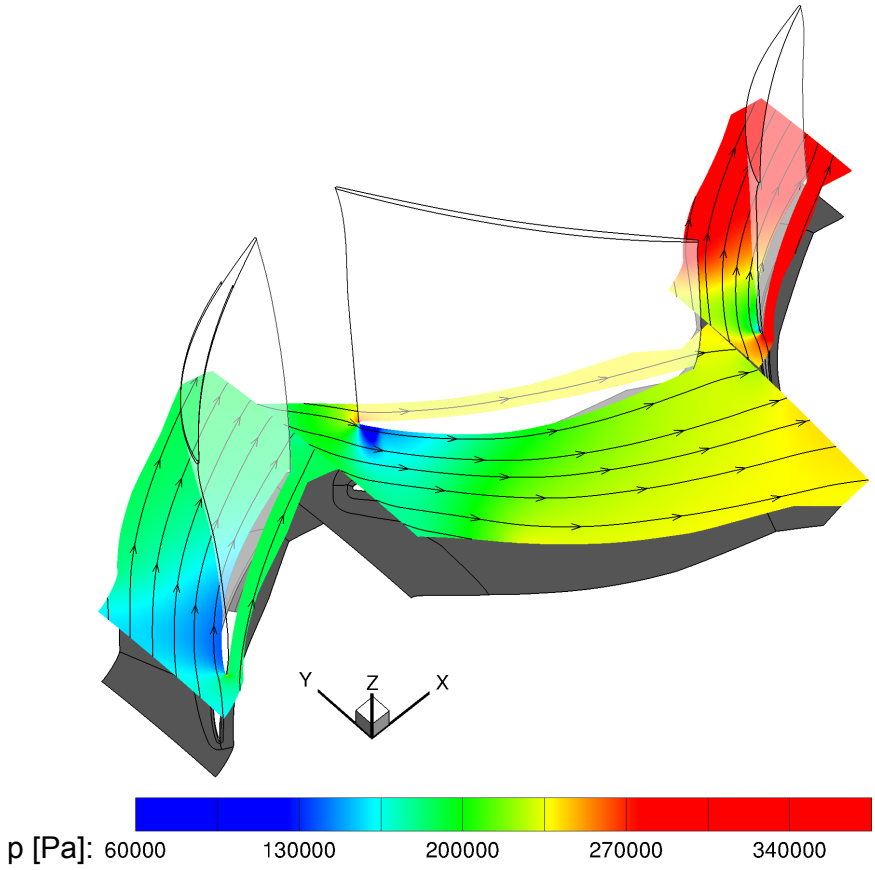


Abb. 6.19: S1-Stromfläche bei 25% relativer Kanalhöhe mit Verteilung des statischen Drucks (Testfall Rig250K)

Abb. 6.18 und *Abb. 6.19* zeigen abschließend die Ergebnisse des *generateBladeCuts*-Tasks (*Kap. 5.8*). Die erste Abbildung beinhaltet die Darstellung der resultierenden Schnittlinien der Schaufeloberfläche und *Abb. 6.19* eine S1-Stromfläche mit Verteilung des statischen Drucks. Da diese über die relative Kanalhöhe definiert wird, stellt sie eine geometrische Stromfläche dar (siehe *Kap. 2.4.4*).

6.3 Validierung anhand des Testfalls „Rig250“

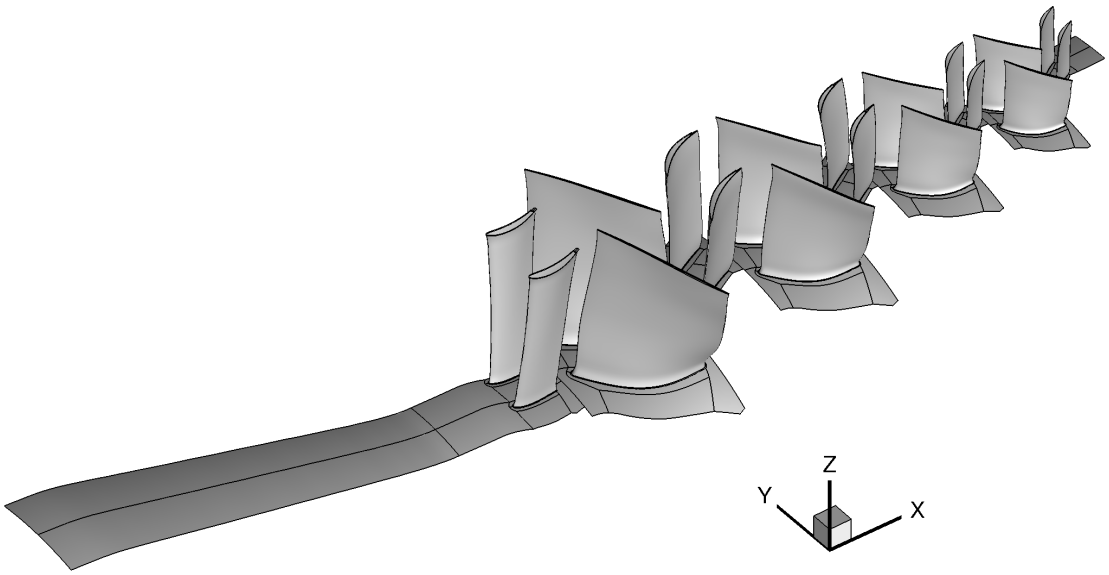


Abb. 6.20: 3D-Darstellung des Rig250-Axialverdichters

Typ	4,5-stufiger Axialverdichter
verwendete *TRACE*-Version	6.7
Zellenzahl des Rechennetzes	8.066.708
Blockzahl des Rechennetzes	89
***POST*-Rechenzeit**	ca. 48 min

Tabelle 6.4: Übersicht des Testfalls Rig250

Dieses Unterkapitel beschäftigt sich mit der Auswertung der Resultate für den Testfall Rig250 (*Abb. 6.20*), bei welchem es sich um den größten im Rahmen der Validierung behandelten Testfall handelt. Die wichtigsten Informationen sind in *Tab. 6.4* aufgeführt. Der Kommandozeilenaufruf lautet:

> POST -i pt5.cgns -gtc -hub row01Hub_Wall row02Hub_Wall row03Hub_Wall row04Hub_Wall row05Hub_Wall row06Hub_Wall row07Hub_Wall row08Hub-_Wall row09Hub_Wall -tip row01Tip_Wall row02Tip_Wall row03Tip_Wall row-04Tip_Wall row05Tip_Wall row06Tip_Wall row07Tip_Wall row08Tip_Wall row-09Tip_Wall -nCuts 18 -cGeomI -task geo -geo -type max -dB -type cutHub -dB -type cutTip -genS2 -reference hub -xieta length -cGeomI -task i3DS2 -

i3DS2 -nCuts 300 -cconv -ip2D -avga -avgm -avgf -avgi -dB -type analysis2d -bS2m -sol FlowSolution -avg area -switch band -bS2m -sol FlowSolution -avg mass -switch band -bS2m -sol FlowSolution -avg flux -switch band -genBlC -load 1dcutBlade.dat -option blade -family row01Blade_Wall row02Blade_Wall row03Blade_Wall row04Blade_Wall row05Blade_Wall row06Blade_Wall row07-Blade_Wall row08Blade_Wall row09Blade_Wall -genBlC -load 1dcutFlow.dat -option flow -family row01VANE01 row02BLADE01 row03VANE01 row04BLA-DE01 row05VANE01 row06BLADE01 row07VANE01 row08BLADE01 row09VA-NE01 -ip2D -o RIG250_COMPLETE_LENGTH.cgns

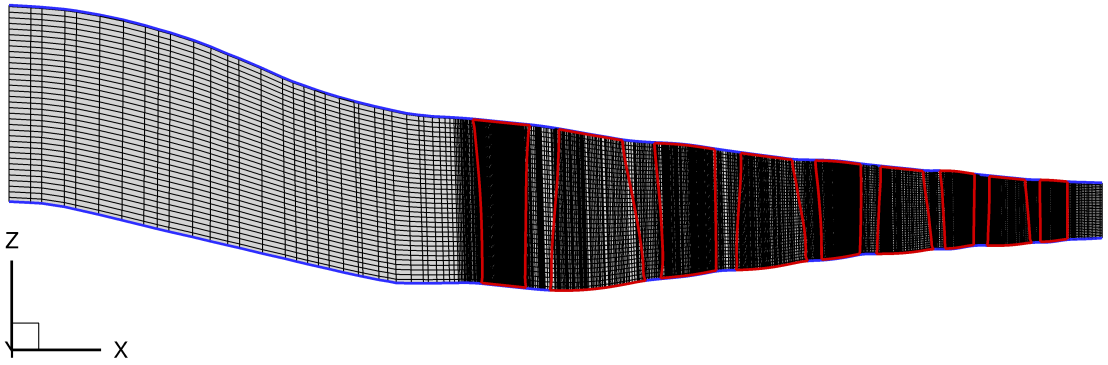


Abb. 6.21: Automatisch generiertes S2M-Schnittnetz - Gesamtansicht (Testfall Rig250)

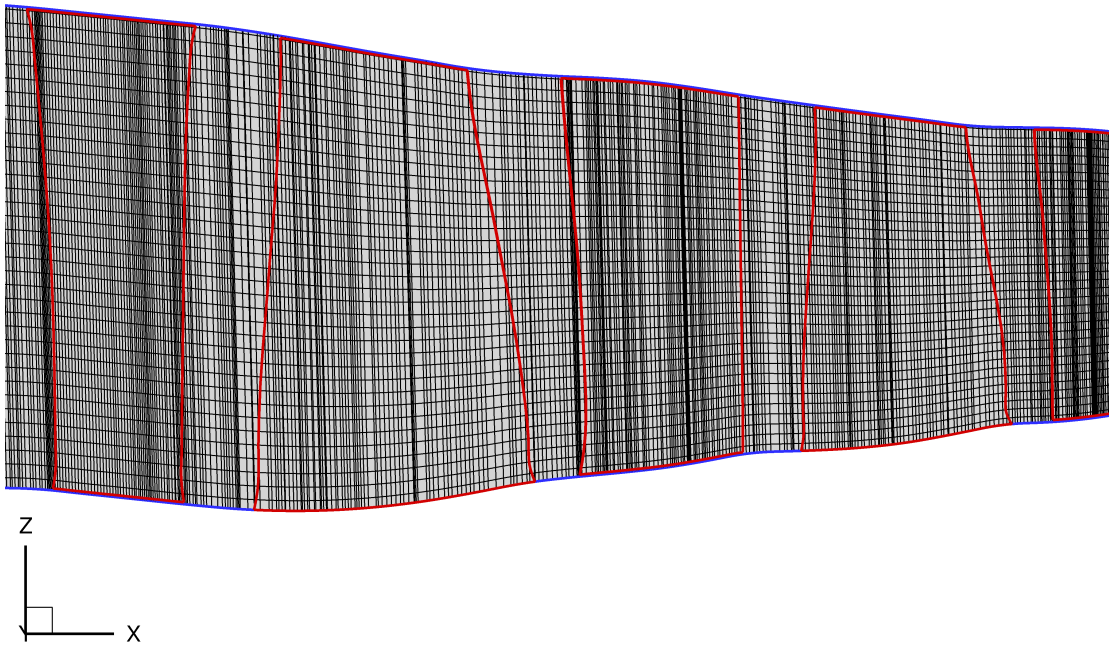


Abb. 6.22: Automatisch generiertes S2M-Schnittnetz - Nahansicht (Testfall Rig250)

Bei der automatischen Netzgenerierung treten, ebenso wie beim Testfall LISA, keine besonderen Schwierigkeiten auf. Zur Verschneidung der Naben- und Gehäuseflächen wurde der Defaultwert von 18 θ-Schnitten verwendet, welcher für die umfangssymetrische Konfiguration ausreichend ist. Das Ergebnis für das S2M-Schnittnetz ist in *Abb. 6.21* sowie *Abb. 6.22* dargestellt. Bei Letzterer handelt es sich um eine Detailansicht der ersten

fünf Schaufelreihen. Zur besseren Übersicht ist dabei jeweils nur jede zweite Netzlinie in Normal- und Meridionalrichtung abgebildet. Gut zu erkennen ist, dass sich die Punktverteilung des automatisch generierten S2M-Netzes an derjenigen der verschnittenen Naben- und Gehäuseflächen orientiert, was in einer eher groben Auflösung des schaufelfreien Bereichs und in einer feineren Auflösung des beschaufelten Bereichs resultiert. Insgesamt weist das S2M-Netz in meridionaler Richtung 1616 Netzpunkte auf.

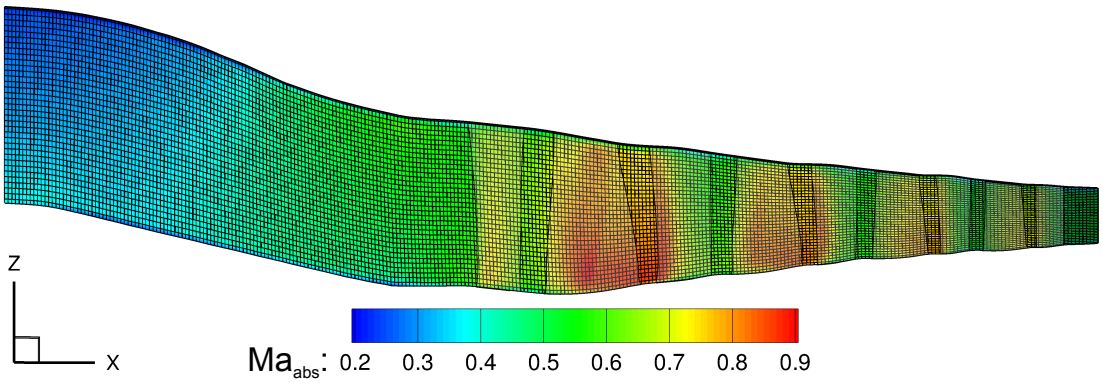


Abb. 6.23: Gesamtansicht der resultierenden S2M-Fläche für die längenbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall Rig250)

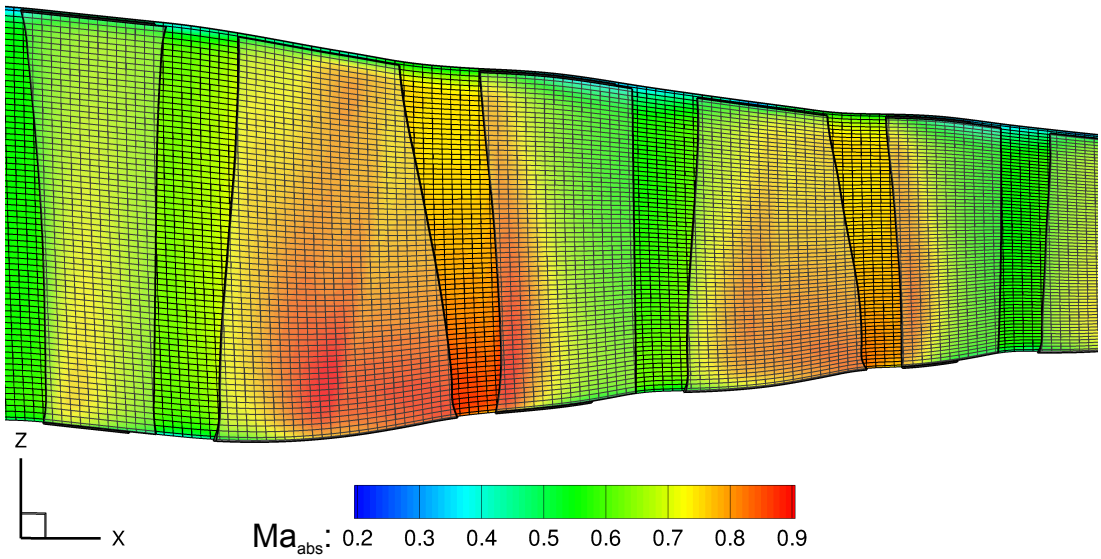


Abb. 6.24: Nahansicht der resultierenden S2M-Fläche für die längenbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall Rig250)

Abb. 6.23 zeigt das resultierende S2M-Netz mit der Verteilung der flächengemittelten Mach-Zahl im Absolutsystem und dem zugrunde liegenden Netz für die längenbasierte ξ-Variante. Die indexbasierte Variante läuft ebenfalls ohne weitere Probleme, wird hier jedoch nicht näher behandelt. Wie aus *Abb. 6.23* ersichtlich, führt die dargestellte längenbasierte ξ-Variante auf eine äquidistante Vernetzung. Das Netz verfügt über 300 Netzpunkte in meridionaler und über 72 Netzpunkte in normaler Richtung. Eine Detailansicht der ersten sechs Schaufelreihen liefert *Abb. 6.24* . In *Abb. 6.25* ist schließlich

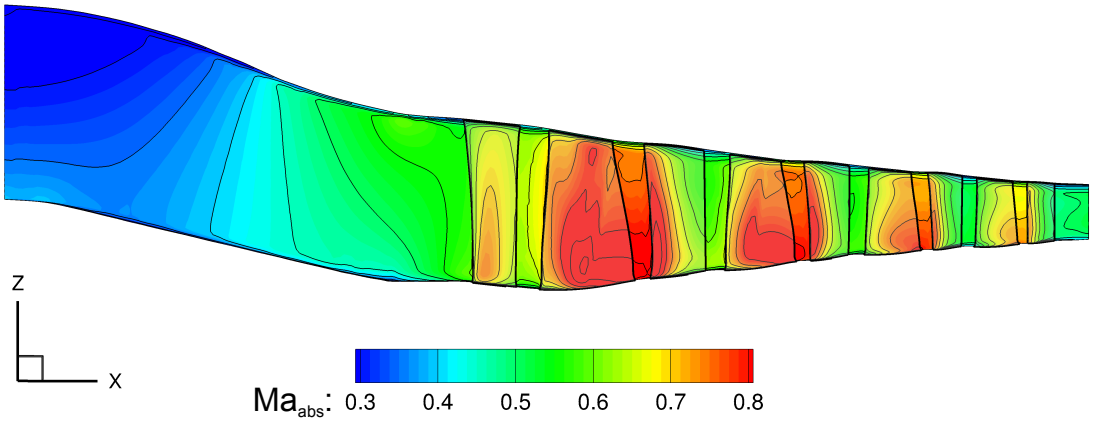


Abb. 6.25: Unvernetzte Gesamtansicht der resultierenden S2M-Fläche mit Verteilung der Mach-Zahl und einer Isolinienstufung von $\Delta Ma_{abs} = 0,05$ (Testfall Rig250)

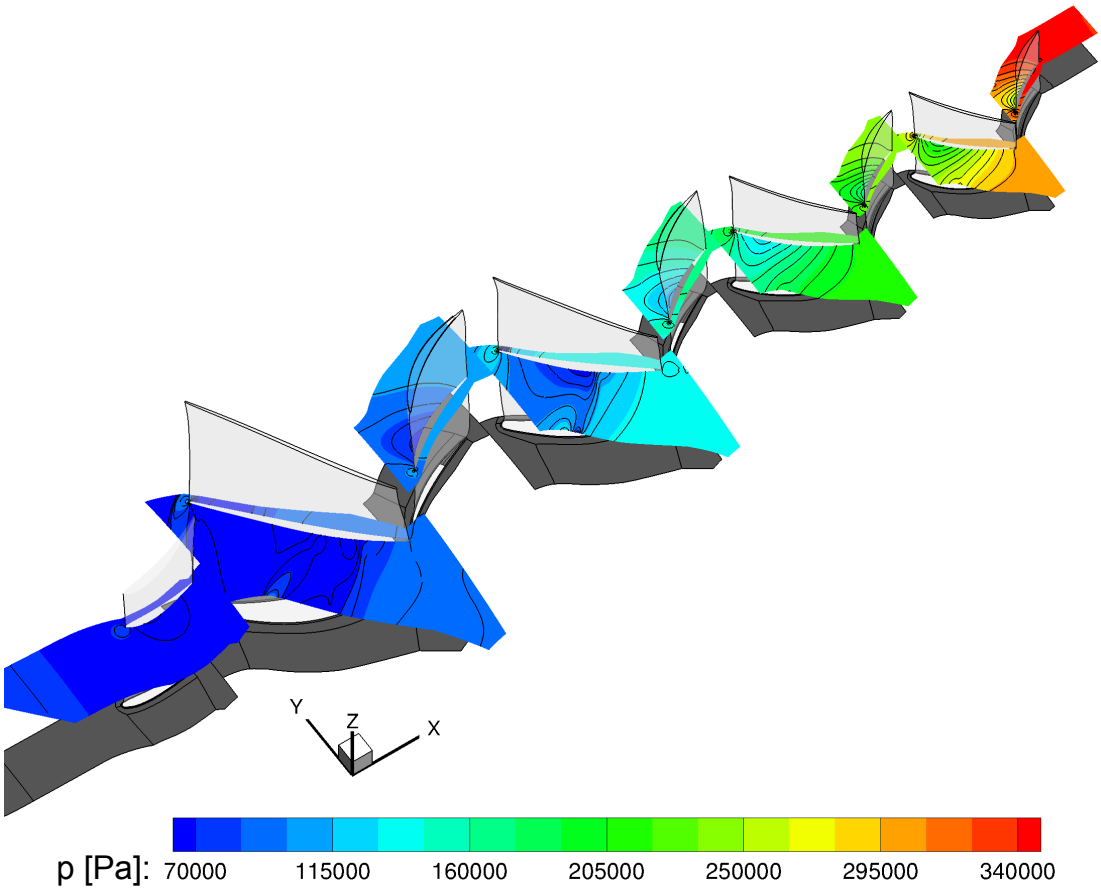


Abb. 6.26: S1-Stromfläche bei 50% relativer Kanalhöhe mit Verteilung des statischen Drucks und einer Isolinienstufung von $\Delta p = 10250\,Pa$ (Testfall Rig250)

die Verteilung der umfangsgemittelten Mach-Zahl über der kompletten Konfiguration dargestellt, wobei zur Verbesserung der Übersicht das Netz ausgeblendet ist.
Auf Basis des finalen S2M-Netzes erfolgt schließlich die Erstellung von Schaufelschnitten und S1-Stromflächen. *Abb. 6.26* zeigt die geometrische S1-Stromfläche bei 50% relativer Kanalhöhe mit der Verteilung des statischen Drucks für die gesamte untersuchte Konfiguration.

6.4 Validierung anhand des Testfalls „VITAL“

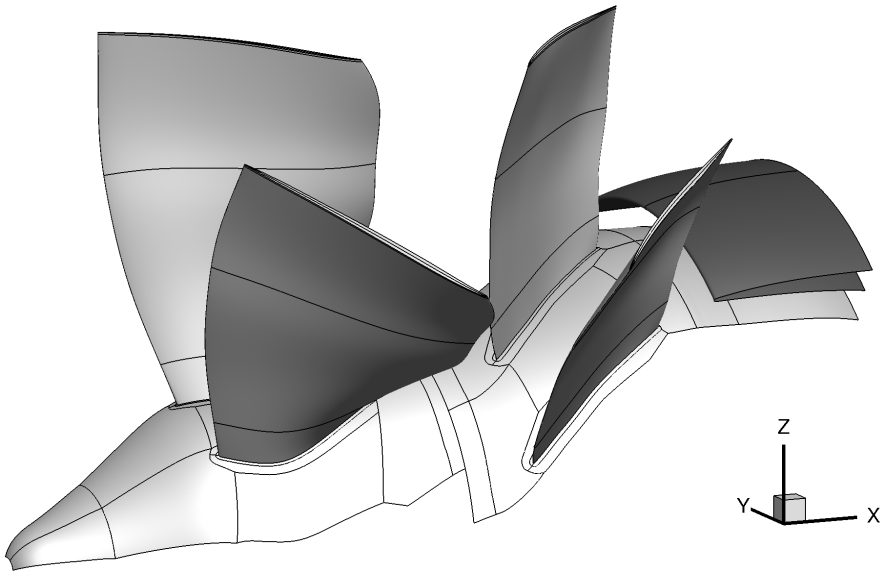


Abb. 6.27: 3D-Darstellung des VITAL-Testfalls

Typ	Fan mit gegenläufigen Rotoren und Splitter
verwendete *TRACE*-Version	7.3
Zellenzahl des Rechennetzes	6.102.800
Blockzahl des Rechennetzes	78
***POST*-Rechenzeit**	ca. 42 min

Tabelle 6.5: Übersicht des Testfalls VITAL

Der folgende Abschnitt befasst sich mit den Ergebnissen des VITAL-Testfalls nach *Abb. 6.27*. Von besonderem Interesse ist für diesen Testfall der Nachweis, dass die automatisierte Turbomaschinen-Auswertung auch auf Splitter-Konfigurationen mit einer Auftrennung der Strömung in einen Kern- und einen Nebenstrom anwendbar ist. Die relevanten Informationen zu diesem Testfall liefert *Tab. 6.5*. Folgender Kommandozeilenaufruf wird verwendet:

> POST -i TRACE.cgns -gtc -hub row01Hub_Wall symmetrie row02Hub_Wall Row02Hub_Wall_Splitter -tip row01Tip_Wall row02Tip_Wall -nCuts 18 -cGeomI -task geo -geo -type max -genS2 -reference hub -xieta length -cGeomI -

task i3DS2 -i3DS2 -cconv -ip2D -avga -avgm -avgf -avgi -dB -type analysis2d -bS2m -sol FlowSolution -avg area -switch band -bS2m -sol FlowSolution -avg mass -switch band -bS2m -sol FlowSolution -avg flux -switch band -genBlC -load 1dcutBlade.dat -option blade -family row0Blade_Wall_ps row01Blade_Wall_ss row02Blade_Wall_ps row02Blade_Wall_ss -genBlC -load 1dcutFlow.dat -option flow -family row01_BLADE01 row02_BLADE01 -ip2D -o VITAL_COMPLETE-_LENGTH.cgns

Abb. 6.28 zeigt das Resultat für das automatisch generierte S2M-Netz. Die Abbildung verdeutlicht besonders gut, dass der Algorithmus des *generateExtremalOutline*-Tasks (*Kap. 5.3.2*) die über die spezifizierten Naben- und Gehäuseflächen vorgegebene Punktverteilung auf das S2M-Schnittnetz überträgt. Denn der beschaufelte Bereich weist eine signifikant höhere Punktdichte auf als die Bereiche der Zu- und Abströmung. Da das S2M-Schnittnetz eine strukturierte Topologie besitzt, erstreckt sich dieses auch über den unvernetzten Splitter. Wie für umfangssymetrische Konfigurationen üblich, werden zur Generierung des S2M-Schnittnetzes 18 θ-Schnitte verwendet. Das Netz weist schließlich 650 Punkte in meridionaler Richtung auf.

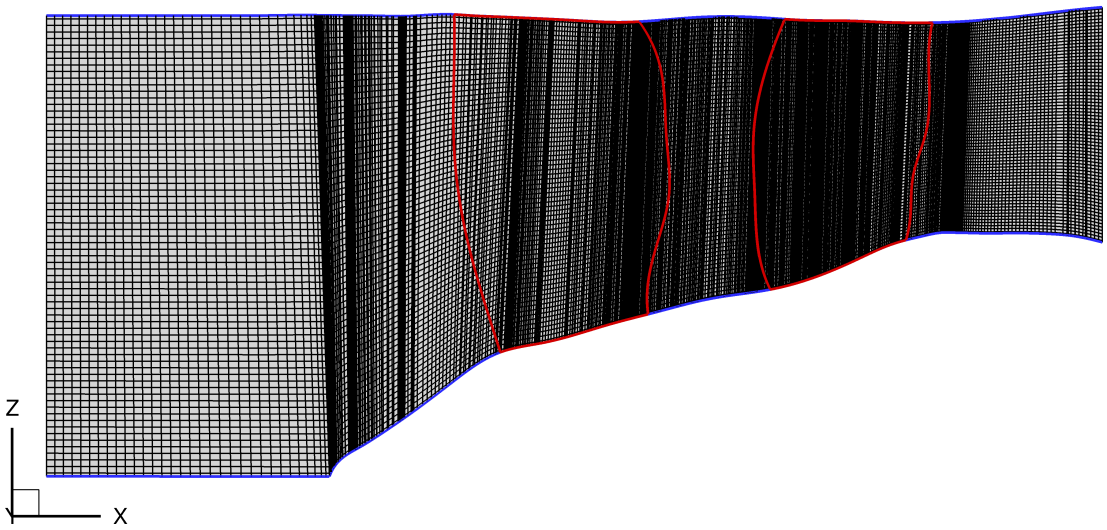


Abb. 6.28: Automatisch generiertes S2M-Schnittnetz (Testfall VITAL)

Abb. 6.29 sowie *Abb. 6.30* veranschaulichen die finale S2M-Fläche für die längenbasierte bzw. für die indexbasierte ξ-Variante, jeweils mit dem zugrunde liegenden Netz und der Verteilung der flächengemittelten Mach-Zahl im Absolutsystem. Das Netz setzt sich gemäß den Standardwerten aus 101 Netzlinien in meridionaler und 72 Netzlinien in normaler Richtung zusammen. Es zeigt sich, dass die indexbasierte im Gegensatz zur längenbasierten ξ-Variante bewirkt, dass im Bereich einer hohen Punktdichte des S2M-Schnittnetzes mehr Schnittebenen liegen als in den weniger dicht vernetzten Bereichen. Wie bereits erwähnt, führt die längenbasierte Variante dagegen auf eine äquidistante Vernetzung.
Im Bereich des Splitters werden bei der Verschneidung der 3D-Konfiguration für die dort liegenden Analysefamilien keine Bänder im unvernetzten Bereich aufgebaut. Dadurch, dass jedoch die finale S2M-Fläche auf einem strukturierten Gitter basiert, benötigt jede

Analysefamilie eine identische Anzahl an Bändern. Folglich werden in diesem Bereich die fehlenden Bänder als sogenannte Dummy-Bänder erstellt und alle Variablen der Strömungslösung mit Null belegt (siehe auch *Kap. 5.7*). Dieser Bereich der Null-Lösung ist beispielsweise in *Abb. 6.31* sichtbar.

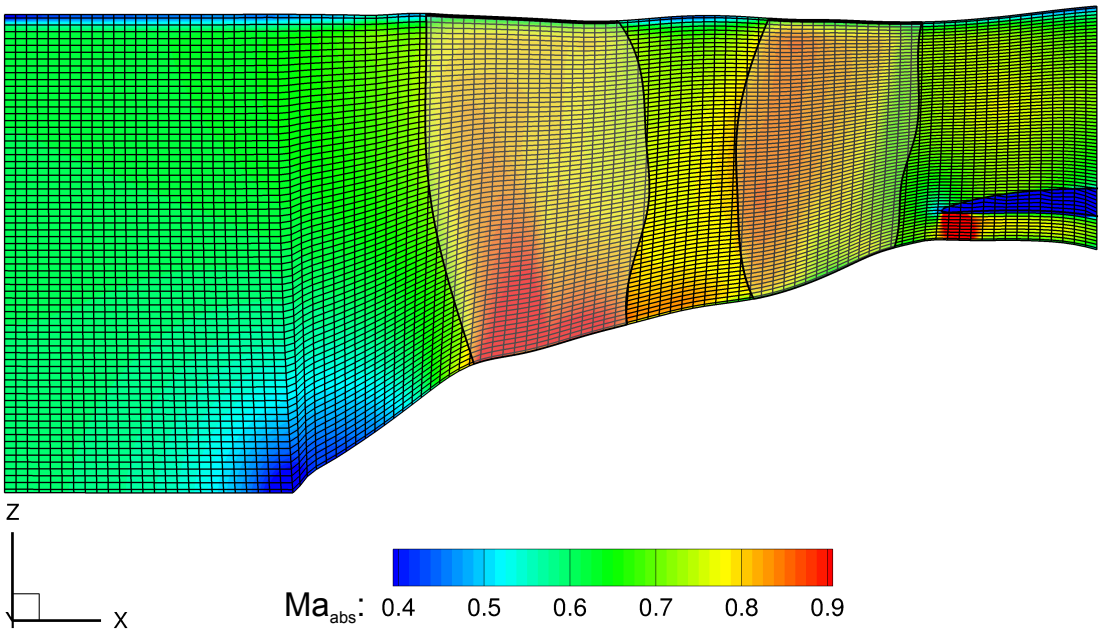


Abb. 6.29: Resultierende S2M-Fläche für die längenbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall VITAL)

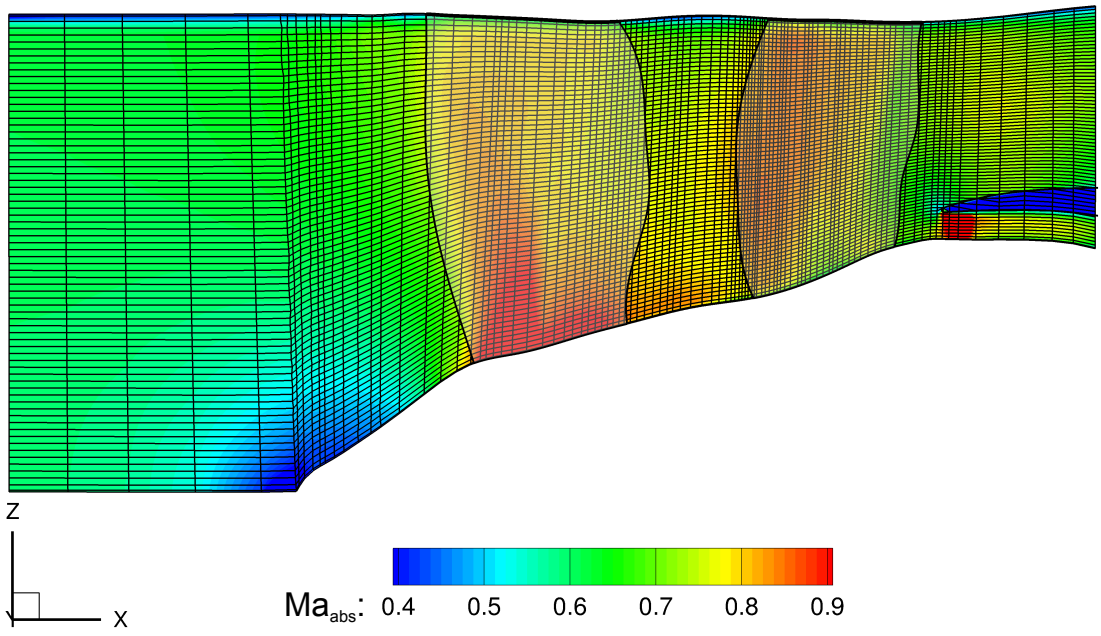


Abb. 6.30: Resultierende S2M-Fläche für die indexbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall VITAL)

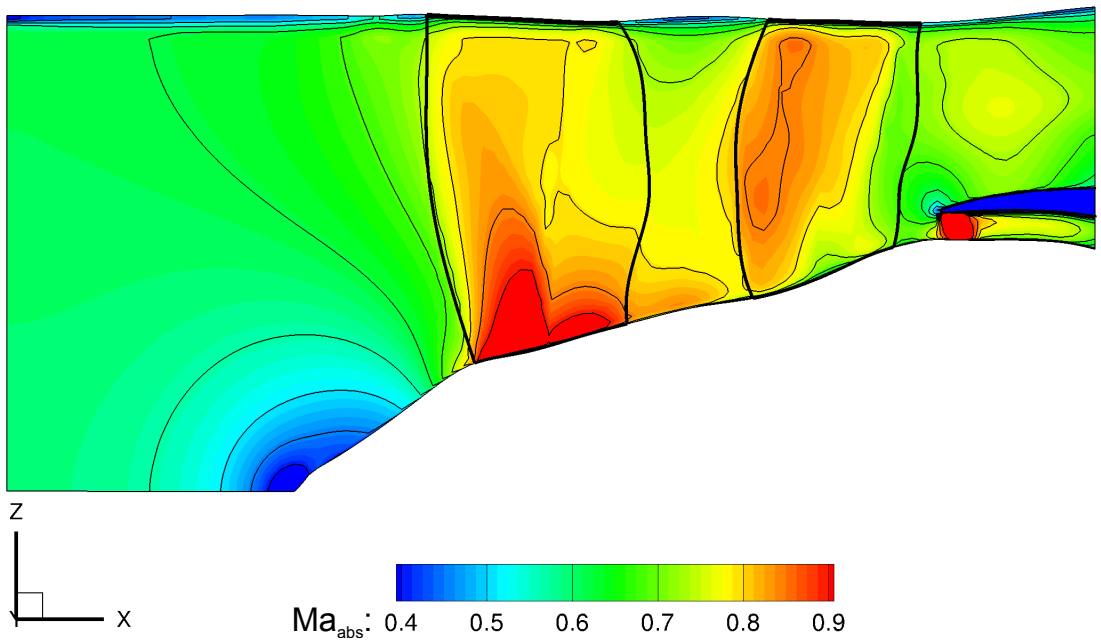


Abb. 6.31: Resultierende Verteilung der Mach-Zahl auf der S2M-Fläche (unvernetzt) mit einer Isolinienstufung von $\Delta Ma_{abs} = 0,05$ (Testfall VITAL)

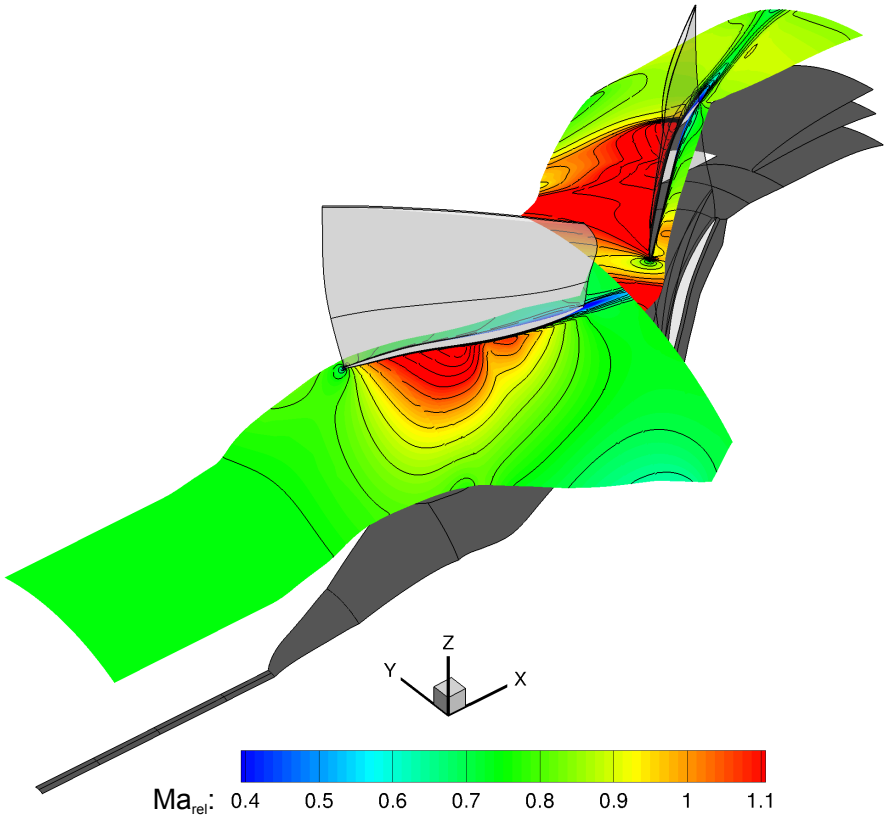


Abb. 6.32: Verteilung der Mach-Zahl auf der S1-Stromfläche bei 50% relativer Kanalhöhe mit einer Isolinienstufung von $\Delta Ma_{rel} = 0,05$ (Testfall VITAL)

Abb. 6.32 stellt schließlich die Verteilung der relativen Mach-Zahl auf der geometrischen S1-Fläche bei 50% relativer Kanalhöhe dar.

6.5 Validierung anhand des Testfalls „NRW-Hightech“

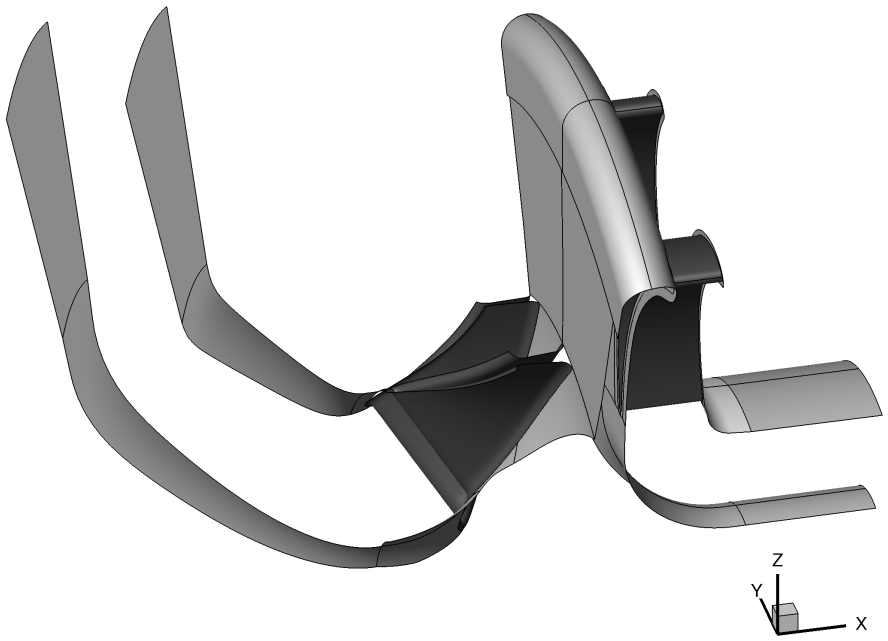


Abb. 6.33: 3D-Darstellung des Radialverdichter-Testfalls NRW-Hightech

Typ	1-stufiger Radialverdichter
verwendete *TRACE*-Version	7.5
Zellenzahl des Rechennetzes	3.237.760
Blockzahl des Rechennetzes	17
***POST*-Rechenzeit**	ca. 35 min

Tabelle 6.6: Übersicht des Testfalls NRW-Hightech

Im Folgenden werden die Ergebnisse des Testfalls NRW-Hightech (*Abb. 6.33*) vorgestellt. Die Besonderheit dieses Testfalls besteht darin, dass es sich im Gegensatz zu den bisher betrachteten Axialmaschinen um einen Radialverdichter handelt. Es ist also nachzuweisen, dass die Tasks der Turbomaschinen-Auswertung unabhängig vom Typ der betrachteten Maschine funktionieren. Die relevanten Informationen zu diesem Testfall sind *Tab. 6.6* zu entnehmen. Der Kommandozeilenaufruf der Auswertung für den Radialverdichter-Testfall lautet wie folgt:

> POST -i TRACE.cgns -rS2m -lS2 S2M_i800_k97_WFWF.dat -xieta index -cGeomI -task i3DS2 -i3DS2 -nCuts 250 -cconv -ip2D -avga -avgm -avgf -avgi -dB -type analysis2d -bS2m -sol FlowSolution -avg area -switch band -bS2m -sol FlowSolution -avg mass -switch band -bS2m -sol FlowSolution -avg flux -switch

band -genBlC -load 1dcutBlade.dat -option blade -family row01Blade_Wall_ss row01Blade_Wall_ps row02Blade_Wall_ps row02Blade_Wall_ss -genBlC -load 1dcutFlow.dat -option flow -family InletDuct row01_BLADE01 STATOR -o TONI_COMPLETE_INDEX.cgns

Abb. 6.34 zeigt das automatisch generierte S2M-Schnittnetz, wobei nur jede zweite Netzlinie in normaler und jede dritte Netzlinie in meridionaler Richtung angezeigt werden. Wie zu erkennen ist, kann zwar die Kontur des Radialverdichters durch den *generateExtremalOutline*-Task korrekt ermittelt werden (*Kap. 5.3.2*), jedoch scheitert die Vernetzung des von der Naben- und Gehäuselinie umschlossenen Gebiets. Hierfür ist der im *generateS2intersecMesh*-Task implementierte Algorithmus nicht geeignet (*Kap. 5.3.3*). Dieser generiert über eine Remapping-Routine Punkte gleicher relativer Lauflänge auf der Naben- und Gehäuselinie und verbindet diese miteinander. Zur Umverteilung der Netzknoten wird dabei die Punktverteilung einer der beiden Linien als Referenz herangezogen und folglich nicht verändert. Da sich die Lauflänge in meridionaler Richtung im Bereich des Rotors beziehungsweise im Bereich der ersten Biegung zwischen Nabe und Gehäuse stark voneinander unterscheidet, liegen die Netzlinien nicht, wie eigentlich erwünscht, orthogonal zur Strömungsrichtung. Im weiteren Verlauf kommt es sogar zur Überschneidung von Netzlinien mit der Naben-Konturlinie. Eine Lösung für dieses Problem stellt die Verwendung einer transfiniten Interpolation unter Verwendung mehrerer Stützlinien zur automatisierten Netzgenerierung dar.

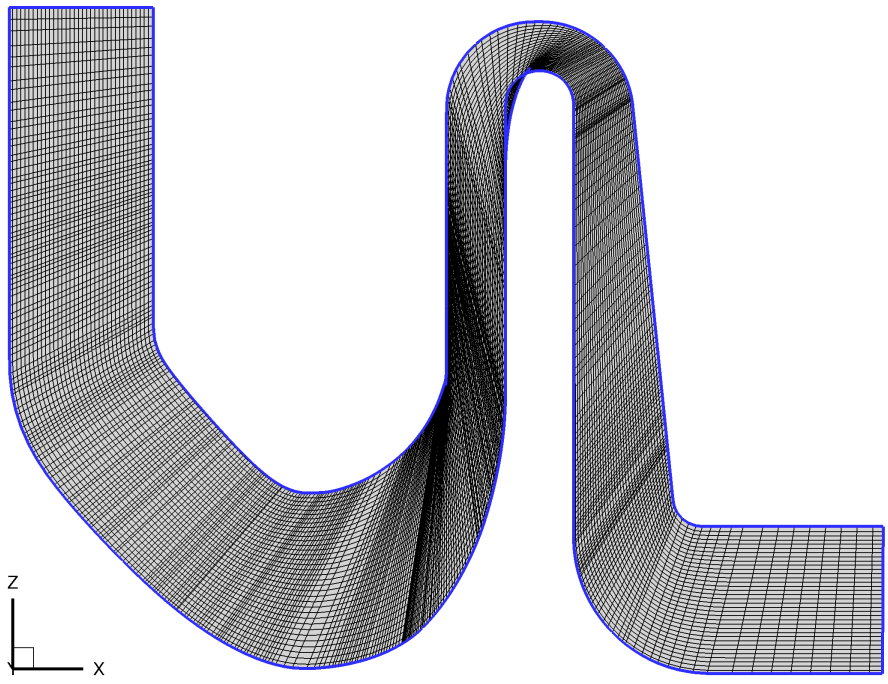


Abb. 6.34: Automatisch generiertes S2M-Schnittnetz - reduzierte Punktzahl (Testfall NRW-Hightech)

Obwohl es sich beim Testfall NRW-Hightech um eine umfangssymetrische Konfiguration handelt, werden zur Erstellung des automatisch generierten S2M-Netzes 35 θ-Schnitte benötigt, um den Konturverlauf vollständig zu erfassen (*Kap. 5.3.1*). Andernfalls bleiben Lücken, in welchen sich keinerlei Schnittlinien befinden. Der Grund hierfür ist

wahrscheinlich, dass der von der Gesamtkonfiguration eingeschlossene Winkel $\Delta\theta$ in Umfangsrichtung für diesen Testfall vergleichsweise groß ist, wodurch sich die Dichte der Schnittebenen verringert. Das resultierende, jedoch fehlerhafte S2M-Schnittnetz verfügt schließlich über 1215 Punkte in meridionaler Richtung.
Da es also mit dem momentan implementierten Algorithmus des *generateS2intersecMesh*-Tasks nicht möglich ist, das S2M-Schnittnetz automatisch zu generieren, wird ein bereits vorhandenes, mit *G3DMESH* erstelltes Netz über den *readS2m*-Task eingelesen (*Kap. 5.3.4*). Dieses besitzt 611 Punkte in meridionaler sowie 97 Punkte in normaler Richtung und ist in *Abb. 6.35* abgebildet.

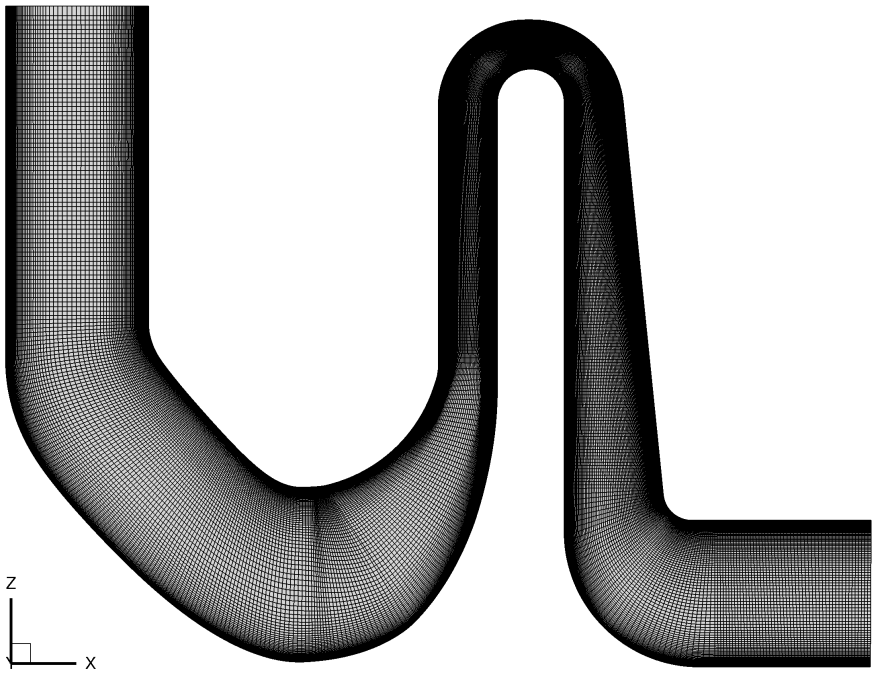


Abb. 6.35: Eingelesenes und verwendetes S2M-Schnittnetz (Testfall NRW-Hightech)

Für das eingelesene Netz erfolgt dann die Bestimmung der dimensionslosen ξ- und η-Koordinaten jedes Netzknotens. *Abb. 6.36* und *Abb. 6.37* zeigen hierbei, dass zur Verschneidung der 3D-Konfiguration beziehungsweise zur Erstellung der finalen S2M-Fläche nur die indexbasierte ξ-Variante verwendet werden kann, da für die längenbasierte Variante aufgrund der lokal stark unterschiedlichen relativen Lauflänge von Naben- und Gehäuselinie dasselbe Problem auftritt wie bei der oben beschriebenen automatischen S2M-Generierung. Die Schnittfamilien würden somit nicht orthogonal zur Strömung liegen, sondern den in *Abb. 6.36* dargestellten Isolinien folgen.
Auf Basis des eingelesenen S2M-Netzes mit der definierten ξ- und η-Verteilung wird im Anschluss die Verschneidung der 3D-Konfiguration durchgeführt. Für diesen Testfall werden dabei 250 Schnittebenen in meridionaler Richtung benutzt. Wie üblich, besitzt jede der Analysefamilien 71 Bänder.
Abb. 6.38 zeigt die resultierende S2M-Fläche mit der zugrunde liegenden Vernetzung und der Verteilung der flächengemittelten Mach-Zahl des Absolutsystems. Gut zu erkennen ist, dass sich durch die Verwendung der indexbasierten ξ-Variante die Schnittflächen und damit die Vernetzung des resultierenden S2M-Netzes am Verlauf der Netzlinien des

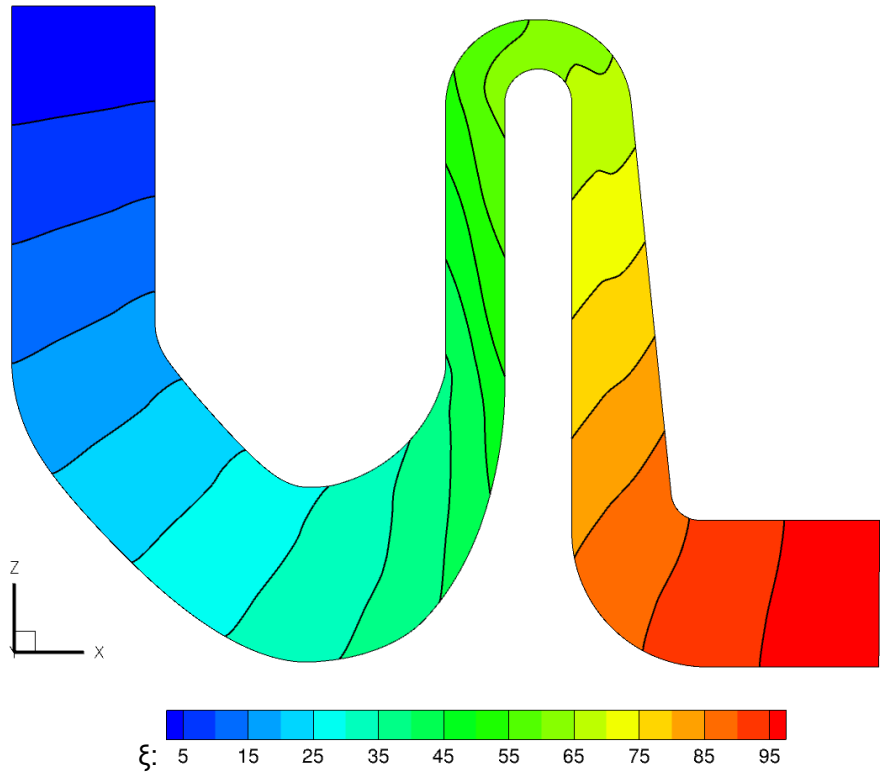


Abb. 6.36: Längenbasierte ξ-Verteilung des eingelesenen S2M-Schnittnetzes - Isolinienstufung $\Delta\xi = 5$ (Testfall NRW-Hightech)

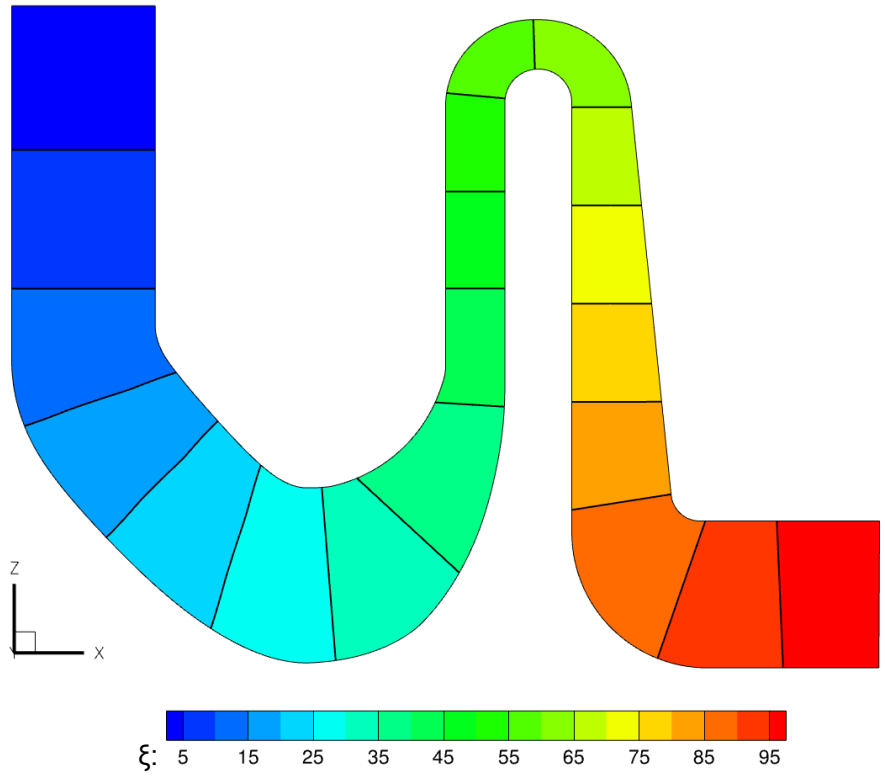


Abb. 6.37: Indexbasierte ξ-Verteilung des eingelesenen S2M-Schnittnetzes - Isolinienstufung $\Delta\xi = 5$ (Testfall NRW-Hightech)

ursprünglichen S2M-Netzes orientieren. Es kann somit gewährleistet werden, dass die Schnittflächen ungefähr orthogonal zur Strömung verlaufen. Voraussetzung hierfür ist, dass dies bei der Generierung des S2M-Netzes berücksichtigt worden ist. Prinzipiell wäre es also möglich eine nahezu identische Vernetzung von S2M-Ausgangsnetz und resultierendem S2M-Netz zu erreichen. Hierfür ist die Anzahl an Schnitten und Bändern, welche im Rahmen des *intersect3DS2MPlane*-Tasks (*Kap. 5.4*) vorgegeben werden, mit der Anzahl an Netzlinien in meridionaler beziehungsweise normaler Richtung des S2M-Schnittnetzes gleichzusetzen. Das eingelesene und das letztendlich resultierende S2M-Netz sind aber nicht exakt identisch, da In- und Outlet der Konfiguration durch die Analyseflächen nicht exakt erfasst werden. Denn die erste Schnittfamilie besitzt gegenüber der Einlassfläche einen gewissen Sicherheitsabstand. Dasselbe gilt am Austritt.

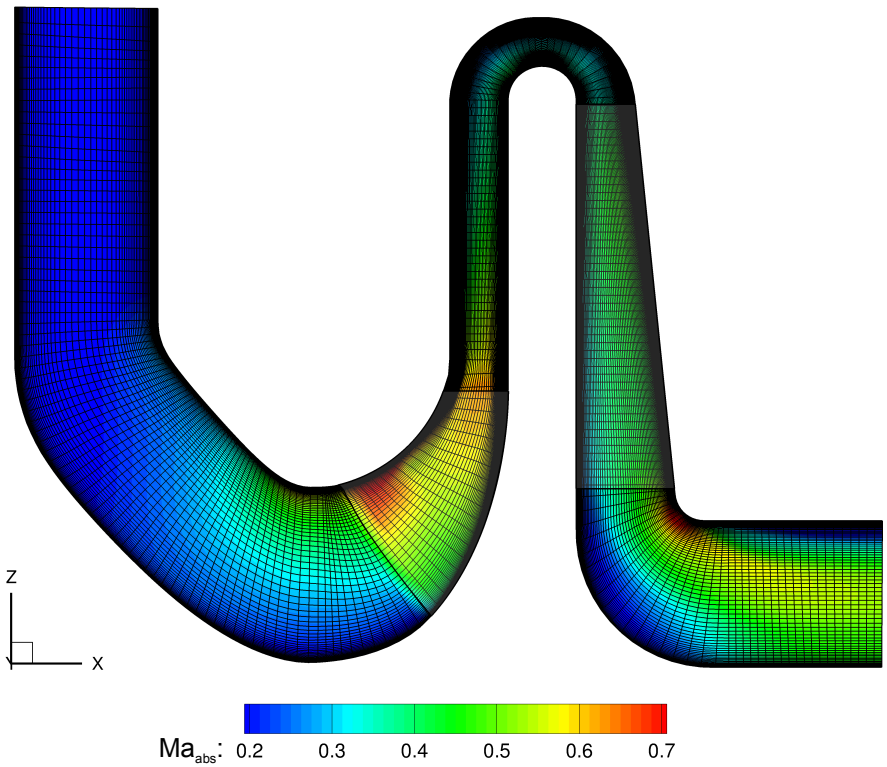


Abb. 6.38: Resultierende S2M-Fläche für die indexbasierte ξ-Variante mit Verteilung der Mach-Zahl (Testfall NRW-Hightech)

Abb. 6.39 zeigt ebenfalls die Verteilung der flächengemittelten Mach-Zahl im Absolutsystem auf der S2M-Fläche, wobei der Übersichtlichkeit halber, das Netz ausgeblendet ist. Zusätzlich sind Stromlinien eingezeichnet, welche jeweils über v_x, $\mathrm{v}_{\theta,abs}$ und v_r definiert sind. Diese drei Geschwindigkeitskomponenten stellen ein Ergebnis der Mittelung unter Verwendung der GTA-Option dar.

Auf Basis des für die S2M-Flächen abgeleiteten relativen Massenstroms und der relativen Kanalhöhe erfolgt schließlich einerseits die Verschneidung der vom Benutzer angegebenen Schaufeloberflächen zur Erzeugung von Schaufelschnitten und andererseits die Verschneidung der spezifizierten Blockgruppen zur Erzeugung von S1-Stromflächen.

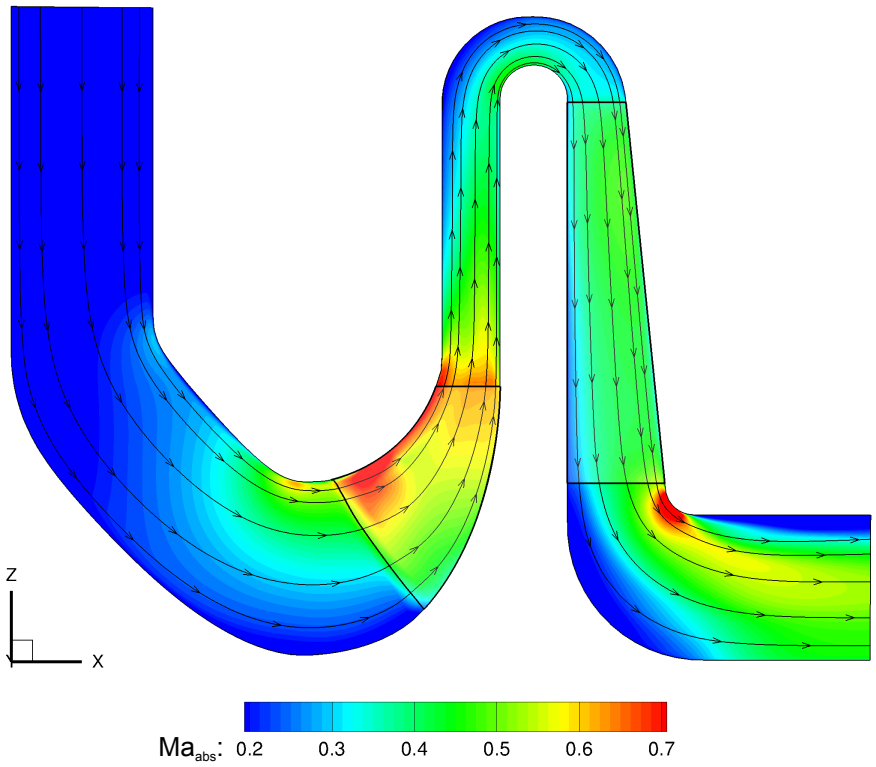


Abb. 6.39: Resultierende S2M-Fläche (unvernetzt) mit Verteilung der Mach-Zahl (Testfall NRW-Hightech)

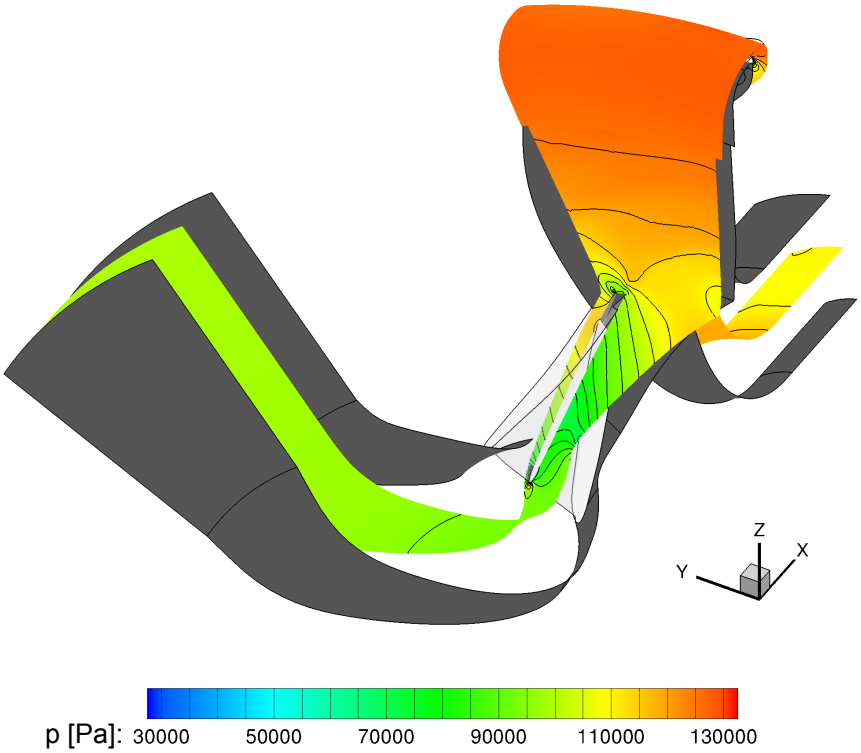


Abb. 6.40: S1-Stromfläche bei 50% relativer Kanalhöhe mit Verteilung des statischen Drucks und einer Isolinienstufung von $\Delta p = 5000\,Pa$
(Testfall NRW-Hightech)

Abb. 6.40 zeigt hierfür die geometrische Stromfläche bei 50% relativer Kanalhöhe mit der Verteilung des statischen Drucks.

Kapitel 7

Zusammenfassung und Ausblick

Das Ziel dieser Semesterarbeit ist die Implementierung einer automatisierten globalen Turbomaschinen-Auswertung für *POST 7*. Die Analyse basiert dabei ausschließlich auf der vom Solver *TRACE* gelieferten dreidimensionalen CGNS-Datei und ist unabhängig von der Topologie und vom Typ der untersuchten Turbomaschine.
Zunächst wird ein temporäres S2M-Schnittnetz erstellt, welches die Abmessungen der 3D-Konfiguration aufweist. Dies geschieht in mehreren Schritten. Zuerst wird der *generateThetaCuts*-Task ausgeführt, welcher Schnittlinien zur Erfassung des Verlaufs der Naben- und Gehäusekontur generiert. Hierbei tritt das Problem auf, dass von *INTERSEC*, welches die eigentliche Verschneidung unter Zuhilfenahme von *VTK* bewerkstelligt, kein kreisförmiges Schnittnetz unterstützt wird. Das schließlich verwendete Vierecks-Schnittnetz kann zwar für alle Testfälle problemlos verwendet werden, beinhaltet jedoch die Annahme, dass die Spannweitenrichtung der z-Richtung entspricht.
Der sich anschließende *generateExtremalOutline*-Task, welcher basierend auf den Schnittlinien eine vom Inlet zum Outlet der Konfiguration durchgehende Konturlinie für Nabe und Gehäuse erzeugt, funktioniert für alle untersuchten Fälle problemlos. Es zeigt sich zudem, dass sich die Punktverteilung der generierten Linie, wie erwünscht, an derjenigen der verschnittenen Naben- und Gehäusefamilien orientiert. Denn Bereiche hoher Punktdichte der 3D-Konfiguration führen auf eine Punktverdichtung an dieser Stelle im S2M-Schnittnetz. Anzumerken ist, dass dieser Task einige fixe Konstanten enthält, welche eventuell für bestimmte Konfigurationen zu modifizieren sind.
Auf Basis der generierten Naben- und Gehäuselinie wird der von diesen eingeschlossene Bereich durch den *generateS2intersecMesh*-Task mit einem strukturierten Gitter vernetzt. Zunächst erfolgt dabei ein Remapping einer der beiden Randkurven. Der Radialverdichter-Testfall (NRW-Hightech) hat gezeigt, dass der implementierte Algorithmus nicht für alle Fälle verwendet werden kann. Denn für komplexere Konfigurationen kann es auftreten, dass sich einzelne Netzlinien mit der Konturlinie schneiden. Das Problem kann durch die Implementierung einer transfiniten Interpolation zur automatischen Netzgenerierung unter Vorgabe mehrerer Stützlinien behoben werden. Für alle untersuchten Axialmaschinen liefert der *generateS2intersecMesh*-Task aber brauchbare Resultate.
Zudem kann als weitere Option ein bereits vorhandenes, vom Benutzer generiertes S2M-

Netz eingelesen und verwendet werden.
Im Anschluss an die Bereitstellung des S2M-Schnittnetzes folgt die Verschneidung der 3D-Konfiguration. Diese basiert auf der Geometrie sowie der ξ- und η-Verteilung des S2M-Netzes. Zur Ermittlung der dimensionslosen Koordinaten sind zwei verschiedene Möglichkeiten vorgesehen, die index- und die längenbasierte Variante. Die indexbasierte Variante führt dazu, dass sich die Lage der Schnittflächen an den Netzlinien des S2M-Ausgangsnetzes orientiert. Diese Variante kann jedoch an bestimmten Stellen zu Schwingungen im Verlauf der Analyseflächen führen. Für die längenbasierte Variante ist dies nicht der Fall, jedoch führt diese auf eine äquidistante Verteilung der Schnittebenen. Die Information über die ursprüngliche Punktverteilung geht somit verloren. Meist ist es also sinnvoller die indexbasierte Variante zu verwenden.
Nach der Erstellung einer beliebigen Zahl von Schnittebenen bzw. Analyseflächen durch den *intersect3DS2MPlane*-Task folgt die S2-Mittelung für alle 2D-Stromflächen, wobei zunächst die Strömungslösung auf diese interpoliert wird. Ein Vergleich der Ergebnisse zeigt, dass die Abweichungen von *POST* gegenüber *TRACE* vernachlässigbar gering sind und die Routinen der Fluss-, der Massen- und der Flächenmittelung somit korrekt implementiert sind. Kleinere Probleme treten nur bei der Flussmittelung auf, wenn sich einzelne Bänder über ein Interface erstrecken. Denn für diese Bänder ist nach der Kommunikation der Blockdaten an die Familien nicht mehr eindeutig definiert, in welchem Bezugssystem sie liegen. Dies führt schließlich zu Sprüngen im Geschwindigkeitsniveau. Dieses Problem wäre dadurch zu beheben, dass sich Bänder nicht mehr über Interfaces erstrecken dürfen. Am Ende der S2-Mittelung liegen dann integrale Werte für alle Bänder und die Gesamtfläche durchströmter Panel- und Analysefamilien vor.
Auf den integralen Bandwerten basierend wird daraufhin die finale S2M-Fläche für einen vom Benutzer spezifizierten Mittelungstyp aufgebaut. Diese wiederum bildet die Grundlage der Generierung von Schaufelschnitten sowie S1-Stromflächen. Die Verschneidung erfolgt dabei entweder für einen konstanten relativen Massenstrom oder eine konstante relative Kanalhöhe. Nach der Generierung der Schnittlinien und Schnittflächen wird eine Interpolation der Strömungslösung auf diese durchgeführt. Da sich die Schaufelschnitte aus mehreren unstrukturierten Linien zusammensetzen, wäre es sinnvoll diese zu einem Profil zusammenzufügen sowie eventuell eine Punktumverteilung durchzuführen. Interessant wäre zudem eine automatische Auftrennung der Schaufelprofil-Schnitte in Druck- und Saugseite.
Anzumerken ist, dass der *finishBladeCuts*-Task und der *globalTurbomachineAnalysis*-Task zwar implementiert sind, aber noch nicht getestet werden konnten, da in *POST* noch keine Interface-Informationen zur Verfügung standen. Für den *GTA*-Task ist zudem noch die Auswertung beliebiger Kontrollvolumina, definiert über Analyseflächen als Berandung, zu implementieren.
Des Weiteren ist im nächsten Schritt die Turbomaschinen-Analyse bzw. die *POST*-Strukturen dahingehend anzupassen, dass eine parallele Auswertung auf mehreren Prozessen stattfinden kann. Hierzu ist unter anderem eine Familien-Kommunikation und -Synchronisation zu implementieren.
Insgesamt zeigen die im Rahmen der Validierung der Implementierung untersuchten Testfälle, dass das Konzept der automatisierten globalen Turbomaschinen-Analyse grundsätzlich richtig ist sowie korrekt umgesetzt wurde. Denn es ist nun möglich, nur auf der

von *TRACE* gelieferten 3D-CGNS-Datei basierend eine Analyse der untersuchten Maschine durchzuführen. Diese ist unabhängig vom Typ der betrachteten Konfiguration. Es können also sowohl Axial- als auch Radialmaschinen, Turbinen sowie Verdichter, Konfigurationen mit oder ohne Nabenkonturierung und Konfigurationen mit oder ohne Aufteilung der Strömung in Haupt- und Nebenstrom behandelt werden. Die Auswertung läuft dabei nach Vorgabe des Kommandozeilenaufrufs automatisch, also ohne Eingriffe seitens des Benutzers.

Literaturverzeichnis

[AHK10] Ashcroft, Graham ; Heitkamp, K. ; Kügeler, E.: High-order accurate implicit runge-kutta schemes for the simulation of unsteady flow phenomena in turbomachinery. In: *Proceedings of V European Conference on Computational Fluid Dynamics ECCOMAS CFD*. Lisbon, Portugal, 2010

[BHK10] Becker, Kai ; Heitkamp, K. ; Kügeler, E.: Recent progress in a hybrid-grid cfd solver for turbomachinery flows. In: *Proceedings of V European Conference on Computational Fluid Dynamics ECCOMAS CFD*. Lisbon, Portugal, 2010

[Bla01] Blazek, J.: *Computational fluid dynamics: principles and applications*. 1. Aufl. Elsevier, 2001

[Brä09] Bräunling, Willy J. G.: *Flugzeugtriebwerke*. 3. Aufl. Springer, 2009

[CH06] Cumpsty, N.A. ; Horlock, J.H.: Averaging Nonuniform Flow for a Purpose. In: *Journal of Turbomachinery* ,Vol. 128 (2006), S. 120–129

[don10] *http://donald.ia.kp.dlr.de/wiki/index.php5/MODELLING_wilcox_kw_model*. aufgerufen am 12.6.2010

[don11a] *http://donald/wiki/index.php5/TRACE_general_process_overview*. aufgerufen am 2.6.2011

[don11b] *http://donald/wiki/index.php5/GMC*. aufgerufen am 6.6.2011

[don11c] *http://donald/wiki/index.php5/PREP*. aufgerufen am 6.6.2011

[Hic10] Hickel, Stefan: *Angewandte CFD*. Technische Universität München, SS 2010. – Skriptum

[Hir09] Hirsch, Christoph: *Technische Verbrennung*. Technische Universität München, SS 2009. – Skriptum

[Hof10] Hoffmann, B.: *SMESH - Turbomachine Structure Mesh Generator*. DLR - Deutsches Zentrum für Luft- und Raumfahrt e. V., 2010

[Ian08] Iannetti, Anthony C.: *CFD General Notation System - Mid-Level Library*. Document Version 2.5.3, 2008

[K+09a] Kau, Hans-Peter u. a.: *Flugantriebe 1 und Gasturbinen*. Technische Universität München, WS 2008/09. – Skriptum

[K+09b] Kau, Hans-Peter u. a.: *Grundlagen der Strömungsmaschinen.* Technische Universität München, WS 2008/09. – Skriptum

[K+09] Kau, Hans-Peter u. a.: *Turboverdichter.* Technische Universität München, SS 2009. – Skriptum

[Kai07] Kainz, Marc: *Untersuchung der Vorhersagegüte eines Zweigleichungs-Turbulenzmodells für Spaltströmungen in Turbomaschinen.* Technische Universität München, 2007. – Semesterarbeit

[Kal10] Kaller, Thomas: *Parameterstudie zur Optimierung der numerischen Abbildungstreue einer Freistrahl-Strömung mit TRACE.* Technische Universität München, 2010. – Semesterarbeit

[Kla07] Klauke, Michael: *Untersuchung der Strömungsverhältnisse im Brennraum eines Gasmotors.* Technische Universität München, 2007. – Diplomarbeit

[May08] Mayer, Florian: *Entwicklung eines Stromlinienkrümmungsverfahrens für die Turboladervorentwicklung.* Technische Universität München, 2008. – Diplomarbeit

[NESZ01] Nürnberger, D. ; Eulitz, F. ; Schmitt, S. ; Zachcial, A.: Recent Progress in the Numerical Simulation of Unsteady Viscous Multistage Turbomachinery Flow. In: *Proceedings of the 15th International Symposium on Air Breathing Engines*, 2001. – ISABE 2001-1081

[O+09] Oertel, Herbert u. a.: *Strömungsmechanik.* 5. Aufl. Vieweg + Teubner, 2009

[S+06] Schlichting, Hermann u. a.: *Grenzschicht-Theorie.* 10. Aufl. Springer, 2006

[S+10a] Schilling, Rudolf u. a.: *Numerische Simulation realer Strömungen.* Technische Universität München, SS 2010. – Skriptum

[S+10b] Schroeder, W. u. a.: *The VTK User's Guide.* 11. Aufl. Kitware Inc., 2010

[Web10a] Weber, A.: *G3DMESH - 3D Structured Grids for Multistage/Multipassage Turbomachines and Linear Cascades, Version 4.9.* DLR - Deutsches Zentrum für Luft- und Raumfahrt e. V., 2010

[Web10b] Weber, A.: *G3DMESH - User's Manual Application Subroutines, Version 4.7.* DLR - Deutsches Zentrum für Luft- und Raumfahrt e. V., 2010

[WF06] Weber, A. ; Fox, R.: *TRACE - User's Manual, Version 5.3.* DLR - Deutsches Zentrum für Luft- und Raumfahrt e. V., 2006

[Wil06] Wilcox, David: *Turbulence Modeling for CFD.* 3. Aufl. DCW Industries, 2006

[Wu52] Wu, Chuang-Hua: A general theory of three-dimensional flow in subsonic and supersonic turbomachines of axial-, radial-, and mixed-flow types / NACA. 1952. – Forschungsbericht. – TN-2604

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe bzw. unerlaubte Hilfsmittel angefertigt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

München, 16.6.2011
Kaller, Thomas