

Parallele dynamische Adaption hybrider Netze für effizientes verteiltes Rechnen

Dissertation
zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultäten
der Georg-August-Universität zu Göttingen

vorgelegt von

Thomas Alrutz

aus Hann. Münden

Göttingen August 2008

D7

Referent: Prof. Dr. Gert Lube

Korreferent: Prof. Dr. Robert Schaback

Korreferent: Prof. Dr.-Ing Cord-Christian Rossow

Tag der mündlichen Prüfung: 17.9.2008

Danksagung

Ich möchte mich an dieser Stelle ganz herzlich bedanken bei Prof. Dr. Gert Lube für die motivierende Betreuung und Unterstützung bei der Anfertigung dieser Arbeit.

Mein Dank gilt auch Prof. Dr. Robert Schaback und Prof. Dr. Cord-Christian Rossow, die das Korreferat für die vorliegende Arbeit übernommen haben.

In den vergangenen Jahren hatte ich Gelegenheit, im Rahmen meiner Tätigkeit für das DLR Institut für Aerodynamik und Strömungstechnik in Göttingen, an sehr interessanten und vielschichtigen Aufgaben in der Forschung und Entwicklung teilzuhaben. Die hierbei gewonnenen Einblicke haben sowohl diese Arbeit als auch meinen weiteren beruflichen und wissenschaftlichen Weg maßgeblich geprägt. Für die vielen Diskussionen und Anregungen möchte ich mich bei meinen ehemaligen Arbeitskollegen der Abteilung Numerische Verfahren (nun Center for Computer Applications in AeroSpace Science and Engineering) bedanken. Insbesondere gilt mein Dank Dieter Schwamborn, Norbert Kroll, Tobias Knopp, Thomas Gerhold, Stefan Melber-Wilkending und Markus Rütten, die jederzeit ein offenes Ohr für richtungweisende Diskussionen hatten.

Außerdem möchte ich mich bei allen Autoren und Coautoren der hier zusammengefassten Publikationen für die gute Zusammenarbeit bedanken.

Ganz besonderer Dank gebührt meiner Lebensgefährtin Barbara Brandfass, die mich während der Anfertigung dieser Arbeit ertragen musste und dennoch die Zeit und Muße gefunden hat, diese Arbeit Korrektur zu lesen ohne zu verzweifeln.

In memoriam

Margot Alrutz
29.1.1949 - 29.1.1987

Hans-Günter Alrutz
28.1.1947 - 28.6.1989

Inhaltsverzeichnis

I Einleitung	1
1 Numerische Strömungssimulation für unstrukturierte Netze	3
2 Anforderungen an heutige Simulationstools	3
3 Zukünftige Herausforderungen	5
II Der DLR TAU-Code	9
4 Status des TAU-Codes zu Beginn der Arbeit	12
4.1 Status der Netzverfeinerung	12
4.2 Status der Netzmanipulation	13
4.3 Status des Netzpartitionierers	13
5 Zielsetzung der Arbeit als Beitrag zum wissenschaftlichen Rechnen	14
III Überblick	19
6 Publikationen	21
7 Manipulation hybrider Netze	23
8 Dynamische Verfeinerung hybrider Netze	29
9 Effiziente Datenstrukturen für paralleles verteiltes Rechnen	35
10 Komplexe Anwendungen	45
11 Ausblick	50
12 Literaturverzeichnis	55
13 Abbildungsverzeichnis	63
IV Publikationen	65
(A) Near-wall grid adaptation for turbulent flows	67
(B) A grid and flow adaptive wall-function method for RANS turbulence mo- delling	85
(C) Investigation of Vortex Breakdown over a Pitching Delta Wing applying the DLR TAU-Code with Full, Automatic Grid Adaptation	115
(D) Improvement of the Automatic Grid Adaptation for Vortex Dominated Flows using Advanced Vortex Indicators with the DLR-Tau Code	133
(E) A Vortex Axis and Vortex Core Border Grid Adaptation Algorithm	143

(F) Investigation of the parallel performance of the unstructured DLR-TAU- Code on distributed computing systems	175
(G) Parallel dynamic grid refinement for industrial applications	187
(H) Numerical simulation of maneuvering combat aircraft	209
(I) Prediction of the Unsteady Behavior of Maneuvering Aircraft by CFD Aerodynamic, Flight-Mechanic and Aeroelastic Coupling	221

Teil I

Einleitung

1 Numerische Strömungssimulation für unstrukturierte Netze

Unstrukturierte Methoden für die numerische Strömungssimulation (CFD¹) sind seit mehr als zwei Jahrzehnten in der Entwicklung. Der originäre Einsatz dieser Methoden lag in der einfacheren Behandlung von komplexen Geometrien, welche überwiegend mit einem Finite-Elemente Ansatz in der Strukturmechanik zu finden waren [99]. Als eine der ersten Anwendungen dieser Technik im Bereich der numerischen Strömungssimulation gilt die Arbeit von Bristeau et. al [8]. Sie konnten mit Hilfe einer Least-Square Galerkin Diskretisierung der *full potential equation* erfolgreich die Berechnung einer Flugzeugkonfiguration demonstrieren. Obwohl dieser Ansatz recht vielversprechend war, führte die unstrukturierte Technik im Bereich der aerodynamischen Anwendung lange Zeit ein Schattendasein. Im Vergleich zu den etablierten strukturierten Methoden wurden die unstrukturierten Methoden entweder als zu kompliziert oder als zu aufwändig angesehen. Den Durchbruch (für den Bereich der aerodynamischen Anwendungen) erlangte diese Technik erst Ende der 1980er Jahre mit der Einbindung in Finite-Volumen Verfahren [37, 36]. Hier konnte man auf existierende Lösungsalgorithmen zurückgreifen [3] und andere Beschleunigungsmethoden adaptieren [57, 58].

Seit dieser Zeit ist die Entwicklung der unstrukturierten Netzmethodik ständig vorangeschritten. Aufgrund der Flexibilität im Umgang mit komplexen Geometrien sowie der Möglichkeit, adaptive Netzverfeinerung anzuwenden [56], haben sich die unstrukturierten Methoden im Bereich der aerodynamischen Strömungssimulation etabliert. Ein sehr guter Überblick über die Entwicklungen in dieser Disziplin findet sich auch in [58].

Der vom DLR Institut für Aerodynamik und Strömungstechnik seit Anfang der 1990er Jahre entwickelte TAU-Code ist ein Simulationscode für die Untersuchung flugphysikalischer Phänomene, der auf der unstrukturierten Netzmethodik basiert [84, 31, 83]. Der sehr hohe industrielle Reifegrad der implementierten Modelle und die im Vergleich zu anderen Codes sehr gute Performance haben ihn zu einem bewährten und routinemäßig genutzten Werkzeug für die deutsche Luftfahrtindustrie werden lassen. Der TAU-Code ist, neben dem von der ONERA entwickelten blockstrukturierten elsA Code [47], einer der beiden großen europäischen Simulationscodes, die in der Flugzeugentwicklung eingesetzt werden [10]. Weiterhin wird der TAU-Code von vielen deutschen Universitäten sowie einigen anderen nationalen und internationalen Forschungsinstituten eingesetzt. Die vorliegende Arbeit basiert auf dem TAU-Code und stellt einen Beitrag zum Wissenschaftlichen Rechnen dar. Alle geschilderten Einsatzgebiete sowie die dargestellten Methoden und Algorithmen sind im TAU-Code vom Autor und den jeweiligen Co-autoren verfügbar gemacht bzw. angewandt worden. Der Status des TAU-Codes vor Beginn der hier vorgestellten Arbeiten ist in Teil II zusammengefasst.

2 Anforderungen an heutige Simulationstools

Durch die stetige Weiterentwicklung und Verbesserung im Bereich der Genauigkeit [4, 46] sind die heute verfügbaren unstrukturierten Gittermethoden bereits im täglichen Einsatz in der industriellen Forschung und Entwicklung.

¹Computational Fluid Dynamics

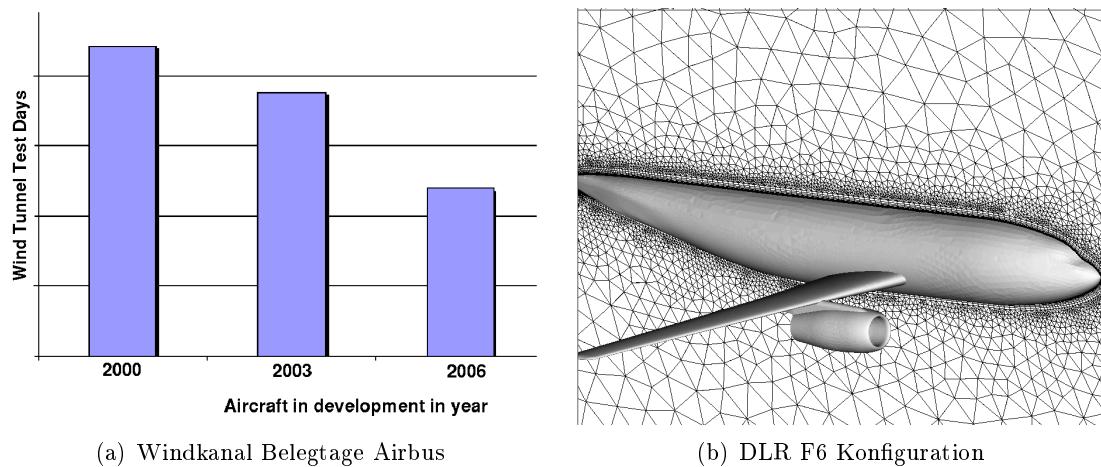


Abbildung 1: Experimente im Windkanal in der Flugzeugentwicklung (a) Quelle [44], Hybrides Netz um Flügel-Rumpf-Pylon-Gondel Konfiguration (b).

Insbesondere bei der Flugzeugentwicklung ist ein klarer Trend hin zur verstärkten Nutzung der Simulation und deutlich geringerer Verwendung von Windkanalexperimenten zu erkennen (siehe Abbildung 1 (a)). Dieser Trend spiegelt sich auch in den wachsenden Einsatzgebieten von CFD beim Entwurf und Design wieder (siehe Abbildung 2).



Abbildung 2: CFD - Einsatz in der Flugzeugentwicklung (Quelle [44]).

Durch die ständig anwachsende Komplexität der Anwendungen, sind die Anforderungen an die Simulationstools enorm gestiegen. Während sich vor ca. 8 Jahren die Komplexität noch auf Flügel-Rumpf (DLR-F4) bzw. Flügel-Rumpf-Pylon-Gondel (Abbildung 1 (b)) Konfigurationen mit ca. 2-4 Millionen Netzpunkten beschränkte, werden heutzutage schon komplett Flugzeugkonfigurationen inkl. Fahrwerk und Bodeneffekt mit bis zu 38 Millionen Netzpunkten simuliert ([83] Seite 20 ff.). Bei diesem Komplexitätsgrad

und der aufwändigen Geometriemodellierung sind die unstrukturierten Verfahren klar im Vorteil. Doch obwohl diese Methoden insbesondere bei der Netzgenerierung sehr flexibel sind [93], ist die vollautomatische Vernetzung ohne manuellen Eingriff bisher noch nicht realisierbar. Daher sind andere lokale Netzmanipulations- bzw. Netzverfeinerungsmethoden notwendig, um nicht den gesamten Netzgenerierungsprozess erneut zu durchlaufen. Ferner ist es natürlich bei derart großen Anwendung unerlässlich, eine komplett parallele Simulationskette zu verwenden, welche die Ressourcen möglichst effizient einsetzt.

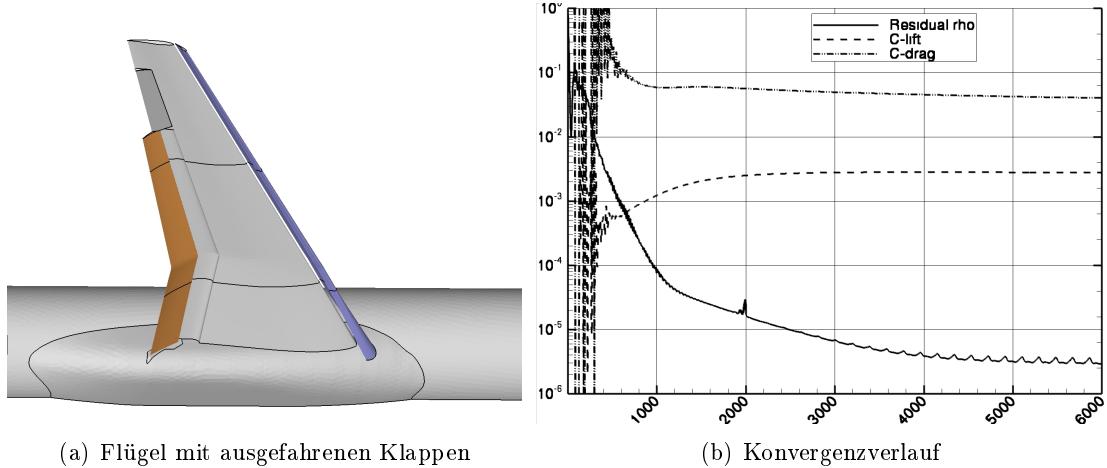


Abbildung 3: Hochauftriebskonfiguration $Re = 25 \times 10^6$, $Ma = 0.2$, $\alpha = 19^\circ$

Als Beispiel für den Aufwand der Berechnung von aerodynamischen Beiwerten (C_L^2 und C_D^3) in der Flugzeugentwicklung mag die Simulation einer Hochauftriebskonfiguration dienen (siehe Abbildung 3 (a)). Hierbei wurde ein 13.6 Millionen Punkte-Netz verwendet, um die aerodynamischen Eigenschaften der Konfiguration unter unterschiedlichen Anstellwinkeln zu berechnen (Polarenberechnung). Für die Berechnung eines Anstellwinkels (Polarenpunkt) sind ca. 5000-6000 Iterationen (siehe Abbildung 3 (b)) des TAU-Codes notwendig, was etwa 10 Stunden Rechenzeit auf 48 CPUs einer Cluster-Installation wie der in [21] erfordert. Für eine komplette Analyse sind jedoch mindestens 6 verschiedene Polarenpunkte auszuwerten, bevor eine Aussage über die aerodynamischen Eigenschaften einer spezifischen Konfiguration gemacht werden kann. Es sind also insgesamt entweder 288 CPUs für 10 Stunden (bei gleichzeitiger Berechnung) oder 60 Stunden bei 48 CPUs notwendig (vgl. auch [5]).

3 Zukünftige Herausforderungen

Im Rahmen der Agenda 2020 der europäischen Luftfahrtindustrie [5, 22], die als Antwort auf die verschärften Umweltschutzanforderungen der Europäischen Union und den

²Auftriebsbeiwert (C-lift)

³Luftwiderstand (C-drag)

wachsenden internationalen Konkurrenzdruck ins Leben gerufen wurde, sind u. a. folgende Ziele definiert:

Zum einen soll der durch den Luftverkehr verursachte Lärm halbiert werden, zum anderen ist eine Reduzierung des Kohlendioxidausstoßes und damit auch des Kerosinverbrauchs um bis zu 50% geplant. Gleichzeitig soll die Verkürzung der Entwicklungszeiten um 50% die Marktfähigkeit deutlich verbessern.

Diese ehrgeizigen Ziele lassen sich nur durch konsequente Weiterentwicklung der Simulationstechnologie erreichen. Sie erfordern insbesondere den Übergang von der Simulation einzelner physikalischer Phänomene für Einzelkomponenten zu einer multidisziplinären Simulation kompletter Systeme unter realen und instationären Einsatzbedingungen. Doch von einer Bewältigung dieser Aufgaben, zu denen in Zukunft etwa ein virtueller digitaler Erstflug [81] oder eine Manöversimulation in Echtzeit [55] gehören werden, sind die derzeit verfügbaren Simulationstools noch weit entfernt [5].

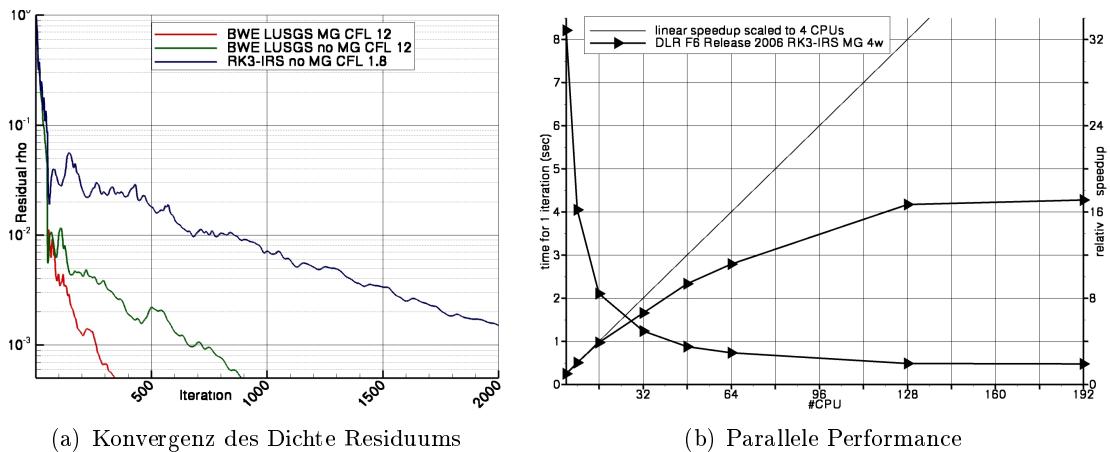


Abbildung 4: Konvergenzgeschwindigkeit (a) und parallele Performance (b) des TAU-Codes [84] mit DLR F6 2×10^6 Punkte Netz [48] gerechnet auf [42].

Durch verbesserte Berechnungsmodelle [95, 16] auf der einen Seite und durch den stetigen Ausbau von Hochleistungsrechner-Kapazitäten [42, 39, 91, 23] sowie der Nutzung parallelisierter Strömungslöser [2, 59] auf der anderen, sind die Turn-Around-Zeiten für typische Simulationsrechnungen bereits stark zurückgegangen (siehe Abbildung 4). Jedoch sind die zur Zeit auf industrieller Ebene verfügbaren Modellierungsansätze, wie RANS⁴, DES⁵ und LES⁶, zur Lösung von strömungsmechanischen Problemen nach wie vor nicht in der Lage, z. B. die Flattergrenze eines Flugzeugs mit hinreichender Genauigkeit und in absehbarer Zeit zu simulieren. So würde eine solche Simulation mit den verfügbaren Methoden und Hochleistungsrechnern mehrere Hunderte von Tagen auf Tausenden von CPUs benötigen [5].

Selbst die Verfügbarkeit von HPC⁷-Clustern mit mehreren Tausenden von CPUs ändert

⁴Reynolds-averaged Navier-Stokes

⁵Detached Eddy Simulation

⁶Large Eddy Simulation

⁷High Performance Computing

an dieser Problematik nicht grundlegend etwas, da selbst die parallelen Strömungslöser bei klassischer Parallelisierung auf verteiltem Speicher mittels MPI [68] schnell an ihre Skalierungsgrenzen stoßen (siehe auch Abbildung 4 (b)).

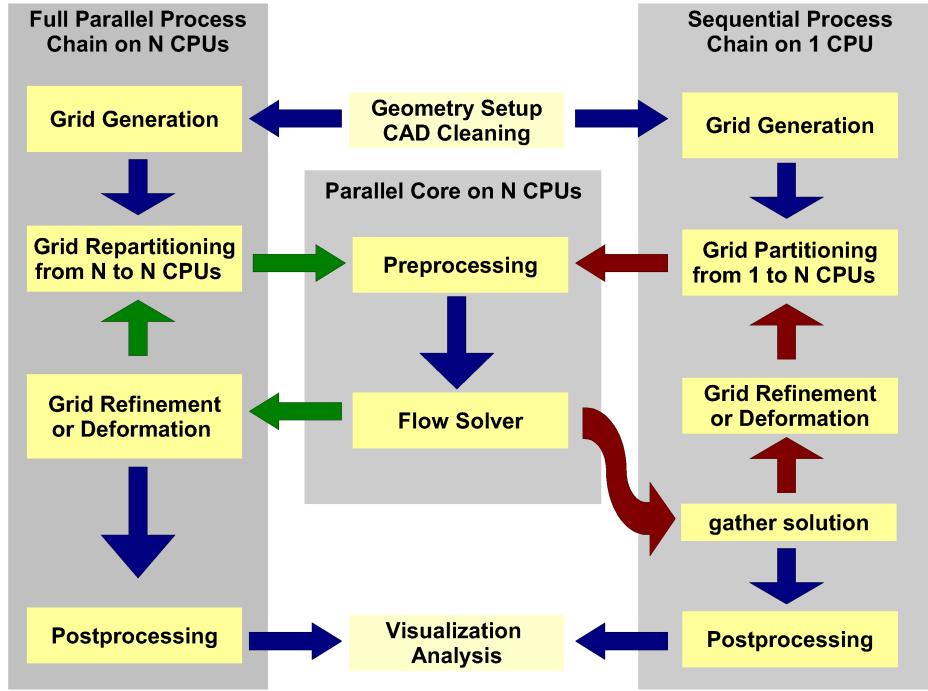


Abbildung 5: Typische Prozesskette einer Strömungssimulation.

Ein weiteres Skalierungsproblem tritt auf, wenn nicht die gesamte Prozesskette des Simulationssystems vollständig parallel und skalierbar ausgelegt ist. Dies ist fast immer der Fall, wenn z. B. Pre- oder Postprocessing-Programme sowie Netzmanipulationen nur sequentiell oder mit begrenzter Skalierung (hinsichtlich Speicherverbrauch) arbeiten. In diesem Zusammenhang sind es fast immer instationäre Anwendungen, die eine Nachführung des Rechennetzes (Chimera-Technik [54]), eine Deformation der Oberfläche (Strömungs-Struktur Kopplung [32]) bzw. die lokale Netzmanipulation (Neuvernetzung/Netzverfeinerung) erfordern [92].

Damit die anstehenden Fragestellungen im Rahmen der Agenda 2020 von Wissenschaftlern und Ingenieuren mit den heutzutage verfügbaren und validierten Methoden bearbeitet werden können, sind unterschiedliche Maßnahmen notwendig. Diese kann man grob in zwei Bereiche aufteilen, einen rein technischen und einen überwiegend algorithmischen. Betrachtet man zum Beispiel eine typische Prozesskette für die numerische Strömungssimulation und zieht gleichzeitig noch eine Netzdadtion mit in Betracht, dann wird sehr schnell klar, dass die sequentiellen Teile der Prozesskette einen Flaschenhals für die Performance der gesamten Prozesskette darstellen. Durch eine Strömungs-Struktur-Kopplung, wie in [32] beschrieben, wird dieses Problem sogar noch weiter verschärft. Um diesen Flaschenhals zu umgehen, ist die Beseitigung aller sequentiellen Anteile innerhalb einer Prozesskette notwendig (in Abbildung 5 links dargestellt). Ist

die komplette Prozesskette voll parallel einsatzfähig, bleibt am Ende noch die Frage nach der Effizienz des Strömungslösers übrig. Hierbei spielen zum einen die parallele Effizienz und zum anderen die skalare Effizienz eine sehr große Rolle. Allerdings darf auch die algorithmische Effizienz nicht vernachlässigt werden, wie Abbildung 4 (a) zeigt. Ein weiterer Ansatzpunkt findet sich in dem schonenden und intelligenten Umgang mit den jeweiligen Gittergrößen. Insbesondere für instationäre Anwendungen ist es häufig einfacher und auch schneller, ein gegebenes Netz zu modifizieren, z. B. durch Umverteilung der Netzpunkte, anstatt den gesamten Netzgenerierungsprozess erneut zu durchlaufen.

Teil II

Der DLR TAU-Code

Der DLR TAU-Code ist ein modernes Software-System für die Simulation von komplexen Geometrien in reibungsfreien bzw. reibungsbehafteten Strömungen. Der Anwendungsbereich erstreckt sich von kleinen Mach-Zahlen bis hin zum Hyperschallbereich und deckt damit den überwiegenden Teil kompressibler Außenströmungen ab. Der TAU-Code kann nicht nur mit komplett unstrukturierten Netzen umgehen, sondern auch mit sogenannten hybriden Netzen, die eine Mischung aus strukturierten und unstrukturierten Elementen darstellen (siehe auch Abbildung 6 (a)). Der Einsatz von hybriden Netzen und den damit verbundenen semi-unstrukturierten Elementen wie Prismen oder Hexaedern, ist überall dort sinnvoll, wo eine gleichmäßige höhere Auflösung bzw. Diskretisierung über reibungsbehafteten Wänden gefordert ist (z. B. im Grenzschichtbereich).

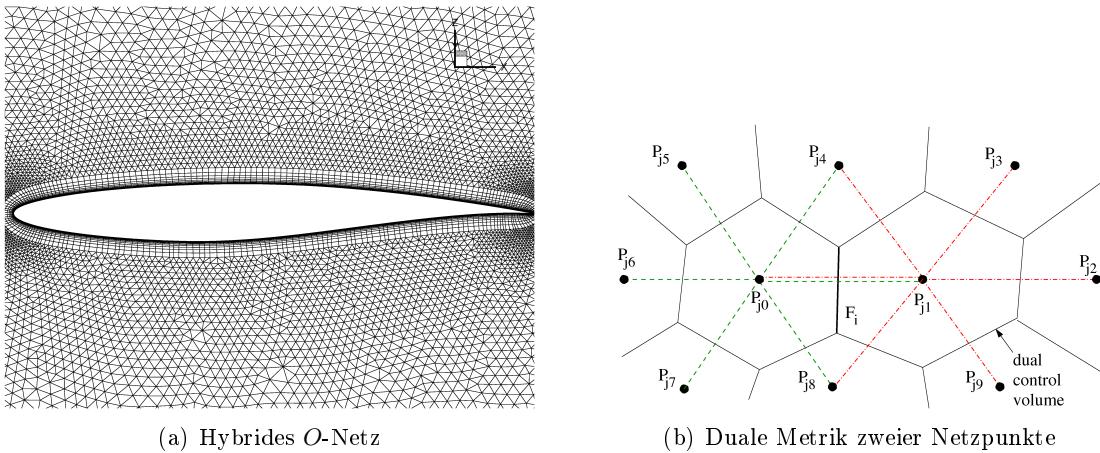


Abbildung 6: RAE 2822 Flügelprofil (a) und Skizze der Kontrollvolumina (b).

Der TAU-Code verfügt allerdings nicht über eigene Netzgenerierungsfähigkeiten. Daher liegt auch der Fokus beim Einsatz des Codes nicht unbedingt auf einer optimalen Netzgenerierung, sondern auf einer lösungsbasierten Netzadaption.

Wie viele andere unstrukturierte Simulationscodes auch (z. B. der NSU3D von Mavriplis [61]) basiert der TAU-Code auf einem Finite-Volumen-Ansatz. Die räumliche Diskretisierung erfolgt knotenbasiert (cell-vertex) und für die Berechnung der Kontrollvolumen eines jeden Netzknotens (Abbildung 6 (b)) ist die Anwendung des TAU-Preprocessings notwendig. Hierbei werden für jeden Primärgitterknoten die zugehörigen Kontrollvolumina, auch duale Zellen genannt, berechnet. Durch die Verwendung des Ansatzes eines dualen Gitters, bezüglich dessen das Verfahren zellzentriert arbeitet, ist der Code unabhängig von den unterschiedlichen Elementen eines Netzes einsatzfähig (siehe auch Beschreibung in [46] Kapitel 5, Seite 81 ff.).

Die Entkopplung und Linearisierung der nichtlinearen gekoppelten Gleichungen für die Strömungsgrößen (und gegebenenfalls für die Turbulenzgrößen) kann für statioäre Probleme durch Iteration in einer Pseudo-Zeit mittels eines expliziten K -stufigen Runge-Kutta-Verfahrens erfolgen. Alternativ ist es möglich, ein semi-implizites LU-SGS-Verfahren zu verwenden, basierend auf der Arbeit von Yoon und Jameson [98].

Für die zeitgenaue Simulation steht ein duales Zeitschritt-Verfahren nach Jameson zur Verfügung [35], bei dem die Semidiskretisierung in der Zeit mittels einer *Backward-*

Differencing Formula höherer Ordnung durchgeführt wird. In jedem Zeitschritt werden die entstandenen stationären Probleme mittels expliziten K -stufigen Runge-Kutta-Verfahren oder semi-impliziten LU-SGS-Verfahren gelöst. Weitere Details zum TAU-Code finden sich in [30, 84, 31, 15, 16, 45].

4 Status des TAU-Codes zu Beginn der Arbeit

Zu Beginn dieser Arbeit im Jahr 2002 lag der TAU-Code noch in einer überwiegend sequentiellen Prozesskette vor (siehe auch Abbildung 5 rechts dargestellt). Zwar war der *numerische Kern*, bestehend aus dem TAU-Preprocessing und dem TAU-Solver, schon voll parallel einsatzfähig (siehe auch [26, 48]), die restlichen Module jedoch nur sequentiell nutzbar. Im Folgenden beschränken wir uns auf die TAU-Adaption und den TAU-Partitionierer, die die relevanten Teile dieser Arbeit betreffen.

4.1 Status der Netzverfeinerung

Die Funktionalität der TAU-Adaption beschränkte sich auf eine hierarchische Netzverfeinerung, die bereits in (J) präsentiert wurde. Eine Entfeinerung zuvor verfeinerter Bereiche war nicht möglich. Die Verarbeitung von hybriden Netzen war auf solche begrenzt, die nur Tetraeder und Prismen bzw. Prismen und Hexaeder (in 2D) enthielten. 3D-Netze mit Pyramiden und unstrukturierten Hexaedern konnten noch nicht verarbeitet werden.

Für die lokale Erkennung von zu verfeinernden Bereichen im Netz war erst ein kantenbasierter Indikator verfügbar. Dieser Indikator basierte auf einem klassischen Gradientenansatz

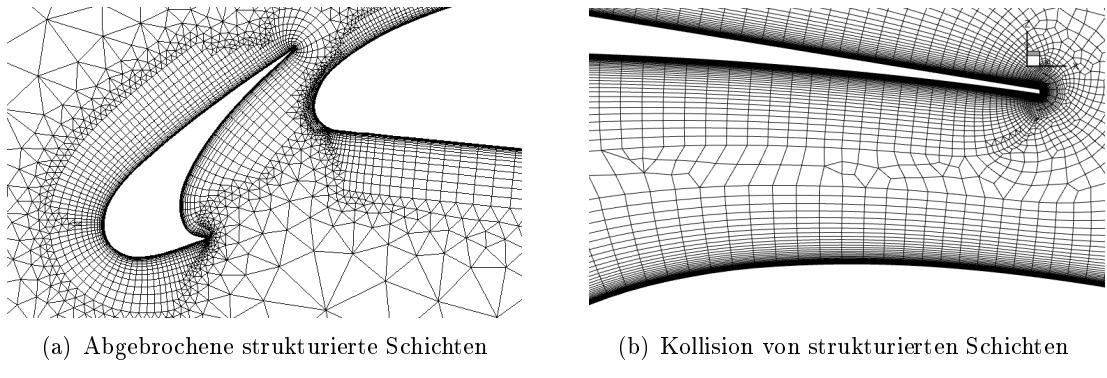
$$(4.1) \quad I_e = \Delta\Phi_e h^\alpha,$$

wobei h die Kantenlänge der Netzkante e ist, α eine Skalierungsgröße und

$$(4.2) \quad \Delta\Phi_e = \max_{i=1,\dots,N_\phi} \left(c_{\phi_i} \frac{(\Delta\phi_i)_e}{(\Delta\phi_i)_{max}} \right)$$

das jeweilige Maximum der N_ϕ verschiedenen Differenzen $\Delta\phi_i = |\phi_i(p_1) - \phi_i(p_2)|$ ausgewählter Strömungsgrößen (ϕ_i) an einer Netzkante (siehe auch [29]). Der andere implementierte Indikator, basierend auf der Arbeit von Sonar und Süli [86], war nur für nicht-viskose Strömungen geeignet.

Die möglichen Strömungsgrößen (ϕ_i), die für den Gradienten-Indikator nutzbar waren, beschränkten sich auf den Betrag des Geschwindigkeitsvektors $|\vec{v}|$, die Dichte ρ , den statischen Druck p sowie den Totaldruck p_{tot} und die totale Enthalpie h_{tot} . Es gab auch noch die Möglichkeit, den Gradienten-Indikator auf ein spezielles skalares Feld anzuwenden (*adapt_indicator*). Hierzu musste der Benutzer allerdings zuvor entweder im TAU-Strömungslöser eine entsprechende Ausgabefunktion implementieren oder durch ein anderes geeignetes Postprocessing-Modul das skalare Feld erzeugen und dem Lösungsfile des TAU-Strömungslösers hinzufügen.

Abbildung 7: Beispiele für nicht *optimale* hybride Netze

4.2 Status der Netzmanipulation

Die Neuverteilung der Netzpunkte innerhalb der strukturierten Bereiche von hybriden Netzen war nur rudimentär möglich. Ausgehend von den Arbeiten von Niederdrenk [70, 72], die ursprünglich für den DLR-eigenen strukturierten Netzgenerator *MegaCads* [19] entwickelt wurden, war im TAU-Code eine Methode zur Neuverteilung der Netzpunkte bei hybriden Netzen in wandnormaler Richtung implementiert worden. Diese Methode funktionierte allerdings nur unter *optimalen* Umständen, wenn zum Beispiel ein geschlossenes *O*-Netz zum Einsatz kam (siehe auch Abbildung 6 (a)). Sollten die strukturierten Schichten jedoch aufgebrochen sein, wie in Abbildung 7 (a), oder eventuell kollidieren, wie in Abbildung 7 (b), dann war eine Neuverteilung der Netzpunkte innerhalb der strukturierten Bereiche nicht möglich. Weiterhin gab es auch keinerlei Möglichkeit, diese Netzmanipulation auf bestimmte Bereiche zu beschränken (z. B. nur den Flügel). Ein anderes Manko dieser Version bestand darin, dass die verwendeten Datenstrukturen keine Möglichkeit vorsahen, die involvierten Netzelemente lokal darauf zu prüfen, ob durch die Neuverteilung der Punkte eine Degeneration eingetreten war. Für den Fall einer Degeneration der strukturierten Netzelemente nach einer Neuverteilung war somit das resultierende Netz unbrauchbar für eine weitere Rechnung. Daher war diese Version zwar für reine 2D Flügelprofile bestens geeignet, jedoch nicht für komplexere Geometrien.

4.3 Status des Netzpartitionierers

Der verwendete Primär-Netzpartitionierer lag ebenfalls nur in einer sequentiellen Version vor und war auch nicht in der Lage, die Verfeinerungshierarchie mit zu verteilen. Die Verwendung von frei verfügbaren Graph-Partitionierern wie z. B. Chaco [34] war aufgrund fehlender Schnittstellen nicht möglich. Weiterhin zerschnitt der Netzpartitionierer die strukturierten Schichten in hybriden Netzen beliebig (siehe auch Abbildung 8 (a)). Dies lag daran, dass der Netzpartitionierer einen rekursiven Bisektions-Algorithmus nutzte, um ein Netz in Partitionen aufzuteilen. Es hat sich auch gezeigt, dass diese Methode nicht optimal ist für das spätere Erstellen der Grobgitter im parallelen Preprocessing [33, 76]. Die Unterstützung von hybriden 3D Netzen mit unstrukturierten Hexaedern (siehe Abbildung 8 (b)) war noch nicht implementiert. Ferner gab es keinerlei

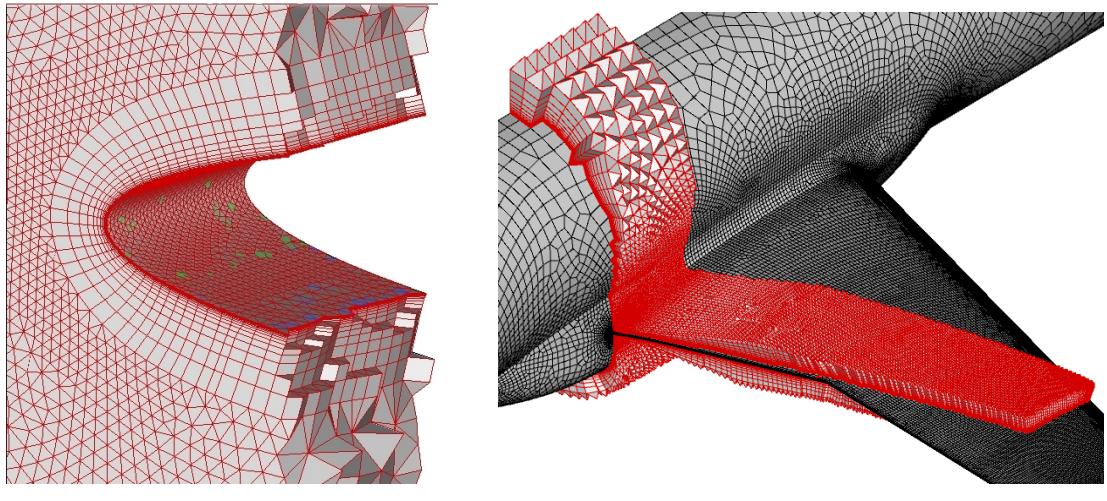


Abbildung 8: Beispiele von Netzpartitionen.

Möglichkeiten für den Benutzer, die Berechnung der Lastbalancierung zu beeinflussen.

5 Zielsetzung der Arbeit als Beitrag zum wissenschaftlichen Rechnen

Das Ziel dieser Arbeit besteht darin, den Beitrag des Verfassers zur Effizienzsteigerung der Simulationskette im TAU-Code zu dokumentieren.

Um diesen Beitrag auf internationaler Ebene entsprechend einzuordnen, erscheint es sinnvoll das Umfeld näher zu beschreiben. Es existieren auf internationaler Ebene viele industriell genutzte CFD-Codes und eine vollständige Liste aller Codes würde den Rahmen dieser Arbeit sicherlich sprengen. Daher wird im Folgenden eine Auswahl von CFD-Codes beschrieben, die an relevanten internationalen Benchmarks teilgenommen haben und auch von international anerkannten Forschungsinstitutionen eingesetzt werden (vgl. [13, 14, 78]). Die unten aufgeführten Codes basieren allesamt auf dem Finite-Volumen Ansatz, arbeiten mit unstrukturierten Netzen, unterstützen Multigitter Beschleunigung, sind parallelisiert und verfügen zum Teil über eigene Werkzeuge zur Netzmanipulation bzw. Netzverfeinerung.

Der **Edge Code**, entwickelt von FOI - *Swedish Defence Research Agency*, ist wie der TAU-Code auch ein cell-vertex (knotenbasierter) Code, der auf unstrukturierten hybriden Gittern rechnen kann. Der Edge Code verfügt über ein eigenes Werkzeug zur Netzverfeinerung und y^+ -basierter Netzmanipulation, welches allerdings nur sequentiell eingesetzt werden kann [1].

Der von der NASA entwickelte **USM3D Code**, ist ein unstrukturierter Code, der nur auf Tetraeder Netzen arbeitet. Eine Netzadaption für die Verfeinerung ist verfügbar, jedoch wird in der Beschreibung darauf hingewiesen, dass diese nur manuell erfolgen kann [69].

Der **Cobalt Code** ist eine kommerzielle Software, die unter anderem bei der US Air-

force Academy zur Untersuchung von komplexen Flugzeuggeometrien eingesetzt wird [67]. Der Code ist in der Lage auf unstrukturierten hybriden Netzen zu rechnen, aber er verfügt nicht über eine eigene integrierte Software zur Netzmanipulation bzw. Verfeinerung [85]. Bei der US Airforce Academy wird zur Netzverfeinerung ein Werkzeug von Pirzadeh verwendet, das eine Verfeinerung durch eine Neuvernetzung des betreffenden Gebietes durchführt (vgl. [73]). Dieses Postprocessing Werkzeug ist jedoch nur sequentiell verfügbar.

Der **NSU3D Code**, der von Mavriplis entwickelt wurde, ist ebenfalls ein unstrukturierter hybrider Code, der wie der TAU-Code auf Multielement-Netzen arbeiten kann. Der Code unterstützt hierarchische Netzadaption auf hybriden Netzen [62, 60], aber nur sequentiell.

Ein weniger bekanntes kommerzielles Software Paket namens **CRISP CFD** [75] ist jedoch in der Lage hybride Netze parallel dynamisch zu verfeinern. Allerdings unterscheidet sich die Methodik und die Umsetzung erheblich von der TAU-Netzadaption. So unterstützt **CRISP CFD** keine y^+ -basierte Netzmanipulation und die hybriden Netze werden nur in den strukturierten Grenzschichtbereichen hierarchisch verfeinert. Für die unstrukturierten Bereiche ist eine Art Neuvernetzung verfügbar, die auf der Einhaltung des Delaunay Kriteriums für Tetraeder Netze beruht (vgl. (J) Seite 11 ff.). Eine Kontrolle über die Anzahl der erzeugten Netzpunkte durch die Verfeinerung ist in **CRISP CFD** nicht vorhanden. Während bei der TAU-Netzadaption in der Grenzschicht anisotrop verfeinert wird und so die strukturierten Schichten erhalten bleiben, wird in **CRISP CFD** isotrop verfeinert und die strukturierten Schichten werden aufgebrochen (vgl. [9]).

Das Alleinstellungsmerkmal des TAU-Codes in diesem internationalen Vergleich ist die parallele dynamische Adaption hybrider Netze. Der Edge Code unterstützt als einziger sowohl die hierarchische Netzadaption als auch die y^+ -basierte Netzmanipulation, aber nur sequentiell. Der NSU3D Code ist zwar hinsichtlich der dynamischen Adaption hybrider Netze mindestens genauso leistungsfähig wie der TAU-Code, jedoch fehlt die y^+ -basierte Netzmanipulation und auch die Parallelisierung. Der USM3D Code rechnet nur auf reinen Tetraeder Netzen und der Cobalt Code muss sich eines sequentiellen externen Werkzeugs bedienen um überhaupt Netze verfeinern zu können. CRISP CFD ist mit Sicherheit ein sehr leistungsfähiges Werkzeug, jedoch ist der Ansatz ein komplett anderer und daher nicht direkt mit dem des TAU-Codes vergleichbar.

Der Beitrag dieser Arbeit zur Effizienzsteigerung der Prozesskette des TAU-Codes soll anhand unterschiedlicher Bausteine deutlich werden. Die Kapitel 7 und 8 befassen sich vor allem mit den Aspekten bzw. Vorteilen von hybriden Netzen.

So beschäftigt sich Kapitel 7 mit den Ergebnissen zweier Arbeiten, die sich mit der Netzmanipulation der strukturierten Grenzschichtbereiche befassen. Zum einen wird die Netzmanipulation gezielt dazu eingesetzt, die Konvergenz bzw. die Genauigkeit einer Strömungslösung signifikant zu verbessern, zum anderen wird die Methodik der adaptiven Wandfunktionen präsentiert. Mit dieser Methodik, die zunächst nur für 2D Netze validiert wurde (die Validierung für 3D Netze ist noch nicht abgeschlossen), ist eine Reduzierung der Netzauflösung in den strukturierten Grenzschichtbereichen bei nahezu gleichbleibender Genauigkeit der Lösung möglich. Die in Kapitel 7 vorgestellten

16 5. Zielsetzung der Arbeit als Beitrag zum wissenschaftlichen Rechnen

Algorithmen für diese Netzmanipulationsmethoden sind aufgrund ihrer speziellen Datenstrukturen bestens für den parallelen Einsatz geeignet. Ein aussagekräftiges Beispiel zur parallelen Performance wird später in Kapitel 9 gegeben. Die Leistungsfähigkeit der Netzmanipulationsmethoden wird anhand von industrierelevanten Konfigurationen mit zum Teil sehr komplexen Netzgeometrien demonstriert.

Die Vorteile der dynamischen Netzadaption werden in Kapitel 8 deutlich, in dem die Ergebnisse von drei Arbeiten im Bereich der Netzverfeinerung bei abgelösten Strömungen vorgestellt werden. Bei diesen Arbeiten handelt es sich im Wesentlichen um Machbarkeitsstudien, die aufzeigen sollen, in welchem Umfang sich eine Netzverfeinerung sinnvoll zur Reduzierung der Netzauflösung bei der Simulation abgelöster Strömungen einsetzen lässt. Die Arbeiten bauen inhaltlich aufeinander auf und zeigen sehr deutlich die Schwierigkeiten, die bei einer automatischen Erkennung von Regionen mit abgelösten Strömungen auftreten können. Allerdings zeigen sie auch auf, wie man im praktischen Einsatz mit diesen Schwierigkeiten umgehen und trotzdem mit akzeptablem Aufwand sehr gute Ergebnisse erzielen kann. Die Motivation für die in Kapitel 8 präsentierten Verfahren stellen die komplexen Anwendungen in Kapitel 10 dar.

Mit dem Themengebiet des effizienten parallelen Rechnens auf verteilten Daten beschäftigt sich vor allem Kapitel 9. Die beiden in diesem Kapitel zusammengefassten Publikationen untersuchen zum einen sehr genau die Anforderungen des TAU-Codes an die Hardware einer Clusterinstallation und zum anderen wird ein Ansatz zur Parallelisierung des TAU-Netzpartitionierers sowie der TAU-Netzadaption präsentiert.

Bei der Untersuchung der Hardwareanforderungen des TAU-Codes auf Cache-basierten Architekturen bestand die Motivation darin, ein genaues Anforderungsprofil für die optimale Performance des Codes zu erarbeiten. Dieser Motivation liegt die Tatsache zugrunde, dass der Lebenszyklus von Computerhardware derzeit bei ca. 3 Jahren liegt, wohingegen der Lebenszyklus industriell genutzter Softwareapplikationen bei 10 und mehr Jahren liegt. Daraus resultiert ein weiteres Problem des effizienten parallelen Rechnens und zwar bleibt die Softwareentwicklung bzw. die Anpassung der Algorithmen hinter der Hardwareentwicklung zurück (vgl. [53, 28]). Es ist daher umso wichtiger, ein klares Anforderungsprofil der Software zu haben, um entsprechende Fehlentscheidungen bei der Algorithmenentwicklung bzw. beim Hardwareeinsatz zu vermeiden.

Die Motivation für den gewählten Ansatz zur Parallelisierung der Adaption und des Partitionierers war unter anderem die Unterstützung der voll parallelen Prozesskette. Allerdings ist bei diesem Ansatz besonders die Problematik von Systemen mit verteiltem Speicher⁸ berücksichtigt worden und daher steht primär die skalierbare Umsetzung in Bezug auf den Speicherbedarf im Vordergrund. Des Weiteren wird noch die Anwendbarkeit des parallelen dynamischen Netzpartitionierers im Zusammenhang mit einem Cluster Batch-System zur Auslastungsoptimierung präsentiert. Die Leistungsfähigkeit und Flexibilität des parallelen Partitionierers sowie der parallelen Netzadaption in einer voll parallelisierten Simulationskette (vgl. Abbildung 5) wird ebenfalls in Kapitel 9 durch mehrere signifikante Benchmarks belegt.

Die beiden Veröffentlichungen in Kapitel 10 beschäftigen sich mit der konkreten Anwendung des Simulationscodes sowie mit der Validierung der Simulationskette zur Unter-

⁸Damit sind Systeme gemeint, die nur über einen verteilten Adressraum bzw. über kein gemeinsames Dateisystem verfügen.

suchungen von instationären aerodynamischen Phänomenen, wie sie üblicherweise bei abgelösten Strömungen auftreten. Grundlage dieser beiden Untersuchungen stellt das DLR-Projekt SikMa [20] dar. Kapitel 10 schildert recht eindrucksvoll welche Herausforderungen die Simulation eines Flugzeugs im Manöverflug an die Simulationssoftware stellt. Insbesondere wenn nicht nur die Aerodynamik mit der Strukturmechanik gekoppelt werden soll, sondern auch noch eine Kopplung mit der Flugmechanik vorgesehen ist. Die in den Kapiteln 7, 8 und 9 vorgestellten Methoden zur Effizienzsteigerung des TAU-Codes zielen alle darauf ab, die Simulation eines frei fliegenden Flugzeugs in Echtzeit zu ermöglichen (vgl. auch [81]). Das DLR-Projekt SikMa ist auf diesem Weg als eine Art Meilenstein zu verstehen, da hier erstmals unterschiedliche Simulations-Werkzeuge miteinander gekoppelt wurden, um ein Flugzeug im Manöverflug physikalisch korrekt zu simulieren.

In Kapitel 10 findet sich neben der Zusammenfassung der beiden Arbeiten, die die Zwischenergebnisse des SikMa Projekts beschreiben, auch noch ein eindrucksvolles Beispiel für die Anwendung der in Kapitel 8 beschriebenen Methode zur Netzadaption bei abgelösten Strömungen. Beim Vergleich der Simulation einer Deltaflügelkonfiguration mit Flugversuchsdaten, wird die Überlegenheit einer lokalen Netzadaption gegenüber vorverfeinerten Netzen deutlich. Das Beispiel ist insofern bemerkenswert, da sich der Autor der zugrundeliegenden Publikation [25] durch die Benutzung der Adoptionsmethoden in einem internationalen Benchmark zum Teil deutlich von einigen anderen Teilnehmern abgrenzen konnte [78].

Der Ausblick in Kapitel 11 zeigt wie die in dieser Arbeit präsentierten Methoden und Algorithmen weiterentwickelt werden, um für andere Effizienzsteigerungen im TAU-Code von Nutzen zu sein.

Es soll nicht verschwiegen werden, dass sich ein derart großes und komplexes Gebilde, wie ein industriell genutzter Simulationscode, nicht einfach durch die Arbeit eines Einzelnen entscheidend voranbringen lässt. Es ist demnach auch nicht verwunderlich, dass ein Großteil der hier zusammengefassten Veröffentlichungen nur durch den vereinten Einsatz mehrerer Autoren zustande gekommen ist. Die spezifischen Beiträge des Verfassers werden in Teil III jeweils geeignet charakterisiert.

Teil III

Überblick

6 Publikationen

Dieser Teil der Arbeit gibt einen Überblick über die behandelten Themengebiete, die in Kapitel 5 schon grob beschrieben worden sind. Die Ziele und Inhalte dieser Arbeit werden in neun separaten Publikationen präsentiert:

- (A) **Near-wall grid adaptation for turbulent flows,**
Thomas Alrutz, Tobias Knopp.
International Journal of Computing Science and Mathematics, 1(2–4):177–192, 2007.
- (B) **A grid and flow adaptive wall-function method for RANS turbulence modelling,**
Tobias Knopp, Thomas Alrutz, Dieter Schwamborn.
Journal of Computational Physics, 220(1):19–40, 2006.
- (C) **Investigation of Vortex Breakdown over a Pitching Delta Wing applying the DLR TAU-Code with Full, Automatic grid adaptation,**
Thomas Alrutz, Markus Rütten.
Paper 5162, *35th AIAA Fluid Dynamics Conference*, June 6-9 2005. Toronto, Canada.
- (D) **Improvement of the Automatic Grid Adaptation for Vortex Dominated Flows using Advanced Vortex Indicators with the DLR-Tau Code,**
M. Widhalm, A. Schütte, T. Alrutz, M. Orlt.
Volume 96 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 186-193, Springer, 2008.
- (E) **A Vortex Axis and Vortex Core Border Grid Adaptation Algorithm,**
Markus Rütten, Thomas Alrutz, Holger Wendland.
International Journal for Numerical Methods in Fluids, DOI: 10.1002/fld.1792, 2008.
- (F) **Investigation of the parallel performance of the unstructured DLR-TAU-Code on distributed computing systems,**
Thomas Alrutz.
Proceedings of the 17th *Parallel Computational Fluid Dynamics Conference*, pages 509-516, 2005. Elsevier.
- (G) **Parallel dynamic grid refinement for industrial applications,**
Thomas Alrutz, Matthias Orlt.
Proceedings ECCOMAS CFD 2006 Conference, September 5-8 2006. Egmond aan Zee, The Netherlands.
- (H) **Numerical simulation of maneuvering combat aircraft,**
A. Schütte, G. Einarsson, B. Schöning, T. Alrutz, W. Mönnich, J. Neumann, J. Heinecke.
High Performance Computing in Science and Engineering' 05. Part 3, pages 185-196, Springer, 2006.

- (I) **Prediction of the Unsteady Behavior of Maneuvering Aircraft by CFD Aerodynamic, Flight-Mechanic and Aeroelastic Coupling,**
 G. Einarsson, A. Schütte, A. Raichle, B. Schöning, W. Mönnich, J. Neumann, J. Arnold, T. Alrutz, J. Heinecke, T. Forkert, H. Schumann.
 NATO Research and Technology Organization, editor, *AVT-Symposium*, Volume AVT-123, 11, April 2005.

Die Ergebnisse der Publikationen werden je nach Themengebiet in den nachfolgenden Kapiteln zusammengefasst und erläutert.

Neben den in dieser Arbeit behandelten Publikationen sind nachfolgend noch weitere Publikationen bzw. Vorträge aufgeführt, die sich ebenfalls mit den Themengebieten dieser Arbeit befassen.

- (J) **Erzeugung von unstrukturierten Netzen und deren Verfeinerung anhand des Adaptationsmoduls des DLR-TAU-Codes,**
 Thomas Alrutz.
Diplomarbeit Universität Göttingen, DLR IB 224-2002 A 10, 2002.
- (K) **Hybrid Grid Adaptation in TAU,**
 Thomas Alrutz.
 Volume 89 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*.
 Chapter 7, Springer, 2005.
- (L) **Near wall grid adaption for wall functions,**
 Thomas Alrutz, Tobias Knopp.
Proceedings of International Conference on Boundary and Interior Layers 2006,
 University of Göttingen, July 24–28, Göttingen, Germany, 2006.
- (M) **Untersuchung der Performance des DLR-TAU-Code auf Linux-Clustern,**
 Thomas Alrutz.
 Vortrag, *HPCN-Workshop*, T-Systems Solutions for Research GmbH, 22.–23. April,
 Braunschweig, 2004.
- (N) **Evaluating the Parallel Performance of up-to-date Linux-Clusters with the DLR TAU-Code,**
 Thomas Alrutz.
 Vortrag, *HPC-Consortium*, Sun Microsystems, June 26–27, Dresden, 2006.
- (O) **Erste Ergebnisse des Projektes zur parallelen Effizienzsteigerung von TAU,**
 Thomas Alrutz.
 Vortrag, *HPCN-Workshop*, T-Systems Solutions for Research GmbH, 27.–28.
 September, Braunschweig, 2007.
- (P) **Recent Developments of TAU Adaptation Capability,**
 Thomas Alrutz, Daniel Vollmer.
 to appear in *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*.
 Springer, 2008.

-
- (Q) **Parallel Computing in Aircraft Design - Strategies, Methods and Applications,**

Thomas Gerhold, Norbert Kroll, Thomas Alrutz, Matthias Orlt.

Vortrag, *20th Parallel CFD Conference 2008*, Lyon, France, May 19–22, 2008.

7 Manipulation hybrider Netze

Unter Netzmanipulation versteht man im Allgemeinen die Veränderung eines bestehenden Netzes, ohne dass sich dabei die Anzahl der Elemente bzw. Punkte des Netzes ändert. Diese Manipulationen dienen fast immer dazu, die Auflösung in einem bestimmten Gebiet zu ändern oder die Qualität der Netzelemente lokal zu verbessern (z. B. durch Verschiebung einzelner Punkte). Die nachfolgend aufgeführten Publikationen beschäftigen sich damit, in Abhängigkeit einer errechneten Lösung, die Netzaufteilung über reibungsbehafteten Wänden lokal zu modifizieren.

- (A) **Near-wall grid adaptation for turbulent flows,**

Thomas Alrutz, Tobias Knopp.

International Journal of Computing Science and Mathematics, 1(2-4):177-192, 2007.

- (B) **A grid and flow adaptive wall-function method for RANS turbulence modelling,**

Tobias Knopp, Thomas Alrutz, Dieter Schwamborn.

Journal of Computational Physics, 220(1):19–40, 2006.

Bei Artikel (A) liegt der Fokus auf einer genauen Beschreibung der Algorithmen zur Neuverteilung der Netzpunkte über reibungsbehafteten Wänden sowie der Beschreibung der für (B) notwendigen Modifikationen des Algorithmus. Weiterhin wird die Parallelisierung der verwendeten Algorithmen beschrieben, wobei in der Beschreibung auch auf Artikel (G) verwiesen wird. Bei den hierbei vorgestellten Methoden ist noch zu bemerken, dass es sich stets um Algorithmen für komplexe 3D Geometrien handelt. Der 2D Anwendungsfall ist sozusagen ein Spezialfall der allgemeineren 3D Version.

Bei Artikel (B) steht die Beschleunigung des TAU-Strömungslösers bei möglichst gleichbleibender Genauigkeit der erzielten Ergebnisse im Vordergrund. Die Beschleunigung des Strömungslösers wird hierbei durch die Reduzierung der Netzaufteilung über den reibungsbehafteten Wänden realisiert (siehe auch Abbildung 9). Diese Publikation beschränkt sich auf 2D Geometrien und ist sozusagen ein Anwendungsfall von (A).

In beiden Publikationen kommt eine Technik zum Einsatz, die schon sehr früh bei der Netzgenerierung für strukturierte Netze verwendet wurde [70]. Diese sogenannte y^+ -basierte Netzdaption kann den Abstand $y(1)$ des wandnächsten Gitterpunktes oberhalb der Wand anpassen und die darüber liegenden Netzpunkte neu verteilen, ohne dass hierzu Elemente verändert werden müssen.

Grundsätzlich basiert die y^+ -Adaption auf der Annahme, dass für anliegende und auch abgelöste Strömungen der erste Netzpunkt oberhalb einer reibungsbehafteten Wand Γ_W die s. g. *low-Reynolds* Netz-Bedingung

$$(7.1) \quad y^+(1) \approx 1$$

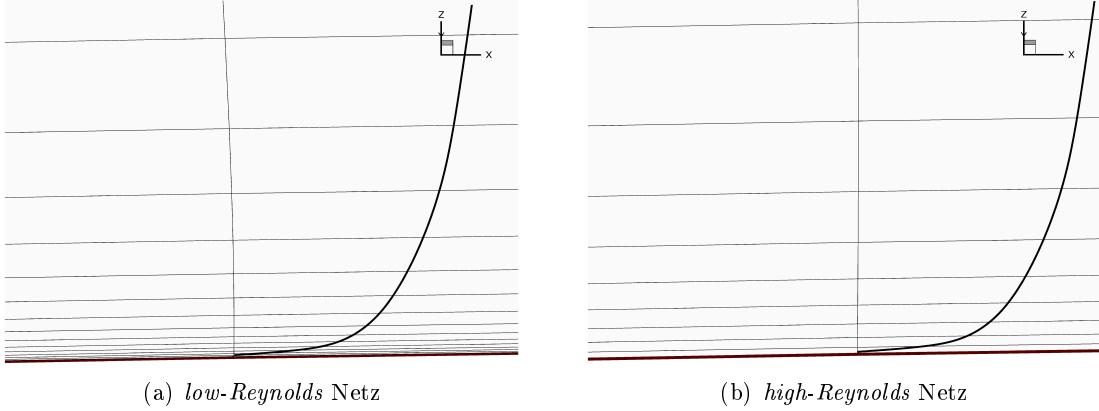


Abbildung 9: Wandnahe Auflösung der Netzlinien mit Geschwindigkeitsprofil für den klassischen Fall (a) und für die Verwendung von adaptiven Wandfunktionen (b).

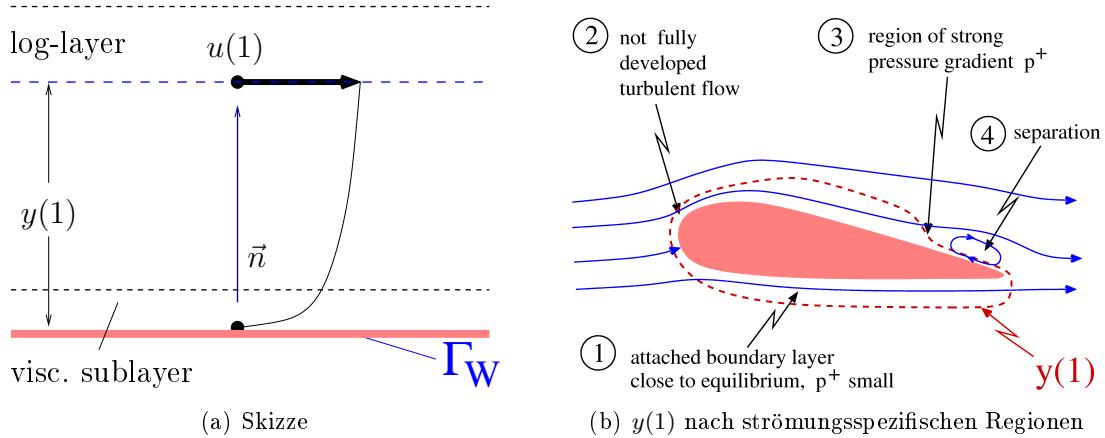


Abbildung 10: Schematische Darstellung der Bestimmung von $y^+(1)$.

erfüllen sollte (siehe auch [96]). Angenommen, $u(1)$ sei die Geschwindigkeitskomponente parallel zur Wand Γ_W am ersten Netzpunkt (siehe auch Abbildung 10 (a)). Dann lässt sich $y^+(1)$ wie folgt bestimmen:

$$(7.2) \quad y^+(1) \equiv \frac{y(1)u_\tau}{\nu}, \quad \text{wobei} \quad u_\tau = \sqrt{\nu \frac{\partial u(1)}{\partial n}}, \quad \nu = \frac{\mu}{\rho}$$

mit Dichte ρ , Viskosität μ und dem zur Wand normalen Vektor \vec{n} . Die *low-Reynolds* Netz-Bedingung sollte schon während des Netzgenerierungsprozesses möglichst eingehalten werden. Jedoch sind beim Netzgenerierungsprozess nicht notwendigerweise alle Simulationsparameter bekannt. Insbesondere u_τ wird auf Grundlage von empirischen Daten geschätzt. Hierbei wird meistens angenommen, dass sich die Strömung in Wandnähe genauso verhält wie über einer ebenen Platte [96].

Dass die Berechnung des ersten Wandabstandes, basierend auf den empirischen Daten von u_τ, ρ und μ bei der Netzgenerierung, nicht zu optimalen Netzen für unterschiedliche

Strömungssituationen führen kann, ist bei genauerer Betrachtung klar. Größere Abweichungen von der *low-Reynolds* Netz-Bedingung führen jedoch entweder zu erheblichen Abweichungen in der Genauigkeit der Lösung oder aber zu einer deutlich verschlechterten Konvergenzgeschwindigkeit. Ein instruktives Beispiel dafür ist der in Abbildung 11 dargestellte Vergleich dreier unterschiedlicher Netze für ein 2D Flügelprofil.

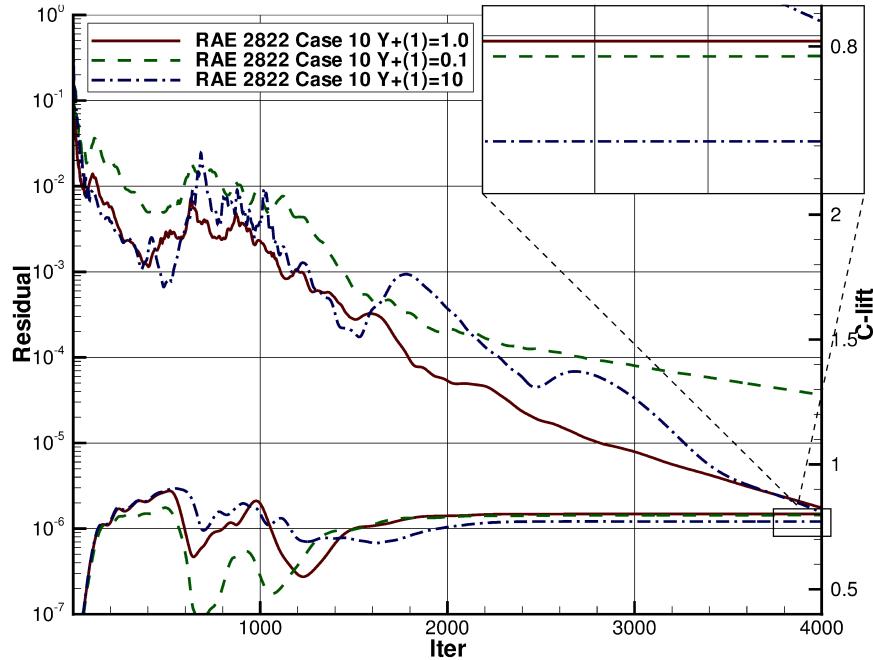


Abbildung 11: Konvergenzverlauf des Residuums der Dichte und des Auftriebsbeiwerts (C_L) auf Netzen mit unterschiedlichen $y^+(1)$ -Werten für den *RAE 2822 Fall 10*.

Hier ist exemplarisch dargestellt, welche Auswirkungen eine erhebliche Abweichung von der *low-Reynolds* Netz-Bedingung haben kann. In Rot ist der Konvergenzverlauf des *low-Reynolds* Netzes zu sehen ($C_L = 0.8016$). In Grün gestrichelt ist der deutlich schlechtere Konvergenzverlauf eines Netzes mit $y^+(1) \approx 0.1$ dargestellt, wobei hier der Auftriebsbeiwert zumindest in akzeptabler Toleranz liegt ($\Delta(C_L) < 0.6\%$). In Blau gestrichelt hingegen wird der Konvergenzverlauf eines Netzes mit $y^+(1) \approx 10$ gezeigt, der zwar etwa die gleiche Konvergenzordnung aufweist wie das *low-Reynolds* Netz, aber zu einem anderen Ergebnis im Auftriebsbeiwert konvergiert ($\Delta(C_L) \approx 3.8\%$).

Daher ist eine lokale lösungsabhängige Anpassung der Abstände der ersten Netzpunkte bzw. Schichten sehr wichtig für eine gute Konvergenz bzw. Genauigkeit der Lösung⁹.

Die erste Version dieser Methode wurde bereits in (K) (siehe Kapitel 6) vorgestellt und funktionierte nur in einer sequentiellen Prozesskette (vgl. auch Abbildung 5). Die in (A) beschriebene Methode ist auch in einer voll parallelen Prozesskette einsetzbar und verfügt weiterhin über Algorithmen zur Erkennung und automatischen Reparatur von degenerierten Elementen.

⁹Wenn man weiß, welche Strömung man simulieren möchte und das Ergebniss kennt, kann man auch ein perfektes Netz für dieses Problem generieren. Im Allgemeinen ist dies aber nicht der Fall!

In (A) 1 wird zunächst das oben beschriebene Problem erläutert und danach der Algorithmus zur Anpassung von $y(1)$ für den sequentiellen Fall beschrieben (vgl. (A) 2.2). In (A) 2.3 erfolgt die Beschreibung der notwendigen Änderungen für den primären Netzpartitionierer im TAU-Code, um bei der parallelen Anwendung die gleichen Datenstrukturen für den Netzadoptionsalgorithmus wie im sequentiellen Fall verwenden zu können. Im weiteren Verlauf werden die Änderungen am Algorithmus selbst aufgelistet, die für die korrekte Funktion auf den verteilten Daten notwendig sind. Abschließend erfolgt die Demonstration der Methode an industrierelevanten Konfigurationen.

Da die Änderungen des Netzpartitionierers sehr weitreichend sind, findet in (G) 3.2 noch eine Diskussion über die Auswirkung auf die parallele Performance des Strömungslösers statt, mehr dazu ab Seite 35.

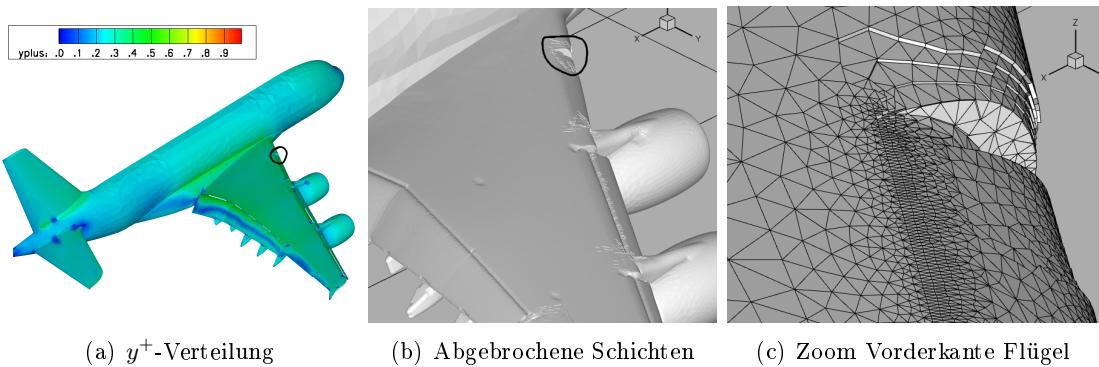


Abbildung 12: Landekonfiguration A380 ((A) Abb. 9-10): Ausgangsnetz mit $y^+(1)$ -Verteilung nicht adaptiert (a), oberste Prismenschicht im Übergang zum unstrukturierten Bereich (b), Zoom auf die Vorderkante mit abgebrochenen Prismenschichten (c).

Sämtliche Erweiterungen sowie der grundlegende Entwurf der verwendeten Datenstrukturen sind vom Verfasser selbst entwickelt und in die y^+ -Adaption implementiert worden. Der große Vorteil dieser Implementierung besteht im Gegensatz zu der vor dieser Arbeit verfügbaren generischen Methode darin, dass die Datenstrukturen für die Anpassung von $y(1)$ und die anschließende Neuverteilung der darüber liegenden Netzpunkte unabhängig von den 2D/3D Netzelementen sind. Es findet lediglich eine Extraktion der wandnormalen Netzlinien aus den jeweiligen Element-Stapeln (Prismen bzw. Hexaedern) statt. Dabei ist es absolut unerheblich, ob die Element-Stapel gemischt auftreten wie in Abbildung 13 (b) oder abgebrochen sind wie in Abbildung 12 (c) zu sehen.

Der Algorithmus zur Anpassung von $y(1)$, basierend auf Berechnung von (7.2), wird nur noch auf die extrahierten wandnormalen Netzlinien angewendet. Dies geschieht lokal für jeden Oberflächenpunkt auf einer reibungsbehafteten Wand separat. Dieses Vorgehen ermöglicht nicht nur eine Selektion bestimmter Regionen durch Ein- oder Ausblenden der betroffenen Wände, sondern es lassen sich dadurch auch unterschiedliche Werte für $y^+(1)$ lokal festsetzen.

Da die Datenstrukturen die Manipulation ganzer Netzlinien¹⁰ ermöglichen, ist ein Zugriff auf die Netz- und Lösungsdaten auch schichtweise möglich. Zum Beispiel lässt sich

¹⁰Die einzelnen Netzlinien sind in der Datenstruktur den Oberflächenpunkten zugeordnet.

so die Konnektivität der Oberflächengeometrie (Nachbarschaftsinformationen) in jeder Schicht ohne den Umweg über die Element-Konnektivität nutzen. Dies wird insbesondere für die Erkennung und das anschließende Reparieren von degenerierten Elementen nach einer Verschiebung der Netzpunkte genutzt. Ferner lassen sich so auch Glättungsalgorithmen, die auf direkten Nachbarn basieren, lokal und in jeder Schicht nutzen (sofern dies notwendig ist).

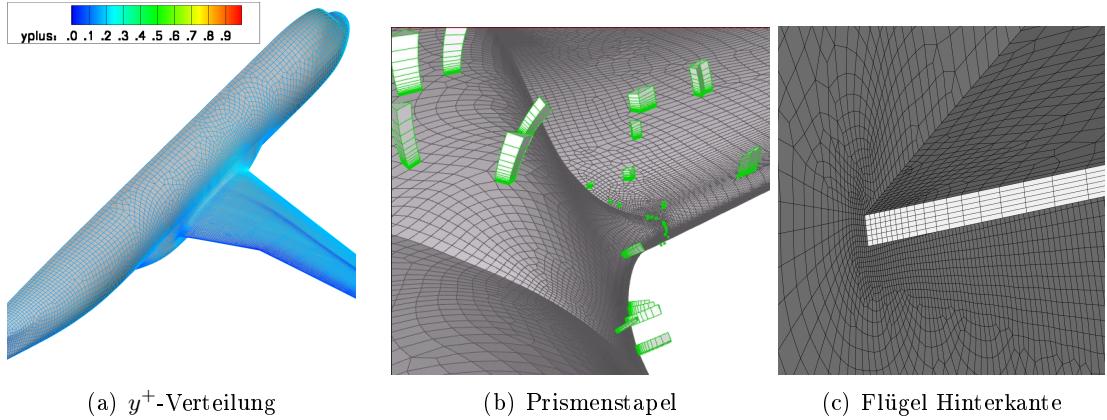


Abbildung 13: Reiseflugkonfiguration A320 ((A) Abb. 7-8): Ausgangsnetz mit unstrukturierten Vierecken und $y^+(1)$ -Verteilung nicht adaptiert (a), einzelne Prismenstapeln eingeschlossen von Hexaederstapeln (vgl. Abb. 8 (b)) (b), Hinterkante des Flügels (c).

(A) 3 befasst sich mit der in (B) vorgestellten Methode der adaptiven Wandfunktionen. Im Gegensatz zu (B) wird hier nur kurz auf die Verwendung von Wandfunktionen für turbulente Strömungen eingegangen. Statt dessen richtet sich der Fokus auf die Beschreibung der zusätzlichen Sensoren, die notwendig sind für die in (B) beschriebenen adaptiven Wandfunktionen, und auf die Modifikation des y^+ -Algorithmus zur Nutzung dieser Sensoren. Ferner wird noch an zwei 2D Beispielen die Leistungsfähigkeit der Methode demonstriert. Eine Abschätzung der durch die adaptiven Wandfunktionen erzielbaren Effizienzsteigerung im Strömungslöser wird ebenfalls gegeben.

Streng genommen ist (B) ein Anwendungsfall des in (A) vorgestellten Algorithmus zur Anpassung der ersten Abstände der wandnormalen Netzpunkte. Die Sensoren, die zur Erkennung der unterschiedlichen Regionen in (B) genutzt wurden (Skizze in Abbildung 10 (b)), konnten aufgrund der Flexibilität, die die Datenstrukturen der y^+ -Adaption ermöglichen, direkt umgesetzt werden.

Es war allerdings notwendig, den in (B) entwickelten *Sensor*

$$(7.3) \quad p^+ = \frac{\nu}{\rho u_\tau^3} \frac{dp}{dx},$$

mit Druckgradient $\frac{dp}{dx}$ in Strömungsrichtung, der ursprünglichen $y(1)$ -Berechnung hinzuzufügen und dann in Abhängigkeit des Ergebnisses den Wert von $y(1)$ lokal neu zu setzen. u_τ wird bei diesem Sensor jedoch mit der in (B) 5.1 entwickelten Methode berechnet.

Bei der Erkennung der Oberflächenkrümmung konnte ebenfalls auf die für die y^+ -Adaption entwickelten Datenstrukturen zurückgegriffen werden, um die Krümmung lokal über die Nachbarn zu ermitteln und nicht den Umweg über die Netzelemente gehen zu müssen. Daher war die Anwendung der universellen Wandfunktion zusammen mit der modifizierten y^+ -Adaption in (B) für 2D Flügelprofile auch erfolgreich.

Abschließend lässt sich für die Verwendung von adaptiven Wandfunktionen in Verbindung mit Turbulenzmodellierung für den TAU-Strömungslöser folgendes Fazit ziehen: Durch die Reduzierung der Auflösung in wandnormaler Richtung über reibungsbehafteten Wänden, ist es möglich einen Geschwindigkeitszuwachs um den Faktor 2 zu erreichen, bei akzeptabler Genauigkeit der resultierenden Lösung (siehe dazu Tabelle 1 $\Delta(C_L), \Delta(C_D) < 1\%$).

	C_L	C_D	N_y	Iterationen	CPU/sec
<i>low – Re</i>	0.7912	0.02687	36	15872	4647
$y^+(1)$	$\Delta(C_L)$	$\Delta(C_D)$	N_y	Iteration in %	CPU Zeit in %
5	0.22 %	0.55 %	32	95.18	91.24
8	1.02 %	0.18 %	29	80.89	75.10
12	0.93 %	0.26 %	27	76.10	69.89
24	0.27 %	0.21 %	24	70.02	62.14
50	0.53 %	0.04 %	20	64.83	52.61
70	0.72 %	0.40 %	18	62.88	49.64

Tabelle 1: Beschleunigung des Strömungslösers durch Verwendung von adaptiven Wandfunktionen für die *RAE 2822 Fall 10* Konfiguration durch Reduzierung der Netzpunkte (N_y) in wandnormaler Richtung mit Angabe der relativen Abweichung von C_L und C_D zu der *low-Reynolds* Netz Lösung (absolute Werte in (A) Tabelle 1).

In Abbildung 14 ist ein Vergleich der Konvergenzverläufe für zwei typische industrielle Anwendungen zu sehen. Es handelt sich dabei um die beiden Beispiele, die auch in (A) verwendet wurden um die Anwendung der y^+ -Adaption zu demonstrieren. In Blau gestrichelt ist der Konvergenzverlauf der nicht angepassten Netze dargestellt und in Rot ist der Konvergenzverlauf der y^+ -adaptierten Netze zu sehen. In beiden Abbildungen wird ersichtlich, dass sich die Konvergenz des Dichteressiduums nach der y^+ -Adaption deutlich verbessert. Bei der Reiseflugkonfiguration (Abbildung 14 (b)) war es notwenig die y^+ -Adaption mehrfach hintereinander anzuwenden, um die *low-Reynolds* Netz-Bedingung zu erfüllen.

In beiden Konfigurationen sind abgebrochene strukturierte Schichten zu finden, was insbesondere bei komplexen Geometrien sehr häufig vorkommt. Die in Abbildung 12 (b) angedeutete Position der abgebrochenen Schichten liegt genau über den ausgefahrenen Landeklappen an der Vorderkante des Flügels. Wie man in Abbildung 12 (c) erkennen kann, war der Netzgenerator nicht in der Lage, eine geschlossene Prismenschicht an dieser Stelle zu generieren. Bei der Reiseflugkonfiguration wurde aufgrund der stumpfen Hinterkante (siehe Abbildung 13 (c)) diese komplette Region von der y^+ -Adaption ausgeschlossen. Ferner bestehen die strukturierten Schichten aus unstrukturierten Hexaderstapeln mit eingebetteten Prismenstapeln (Abbildung 13 (a) und (b)). Die Version der y^+ -Adaption, die vor dieser Arbeit verfügbar war (vgl. Seite 13), wäre bei beiden

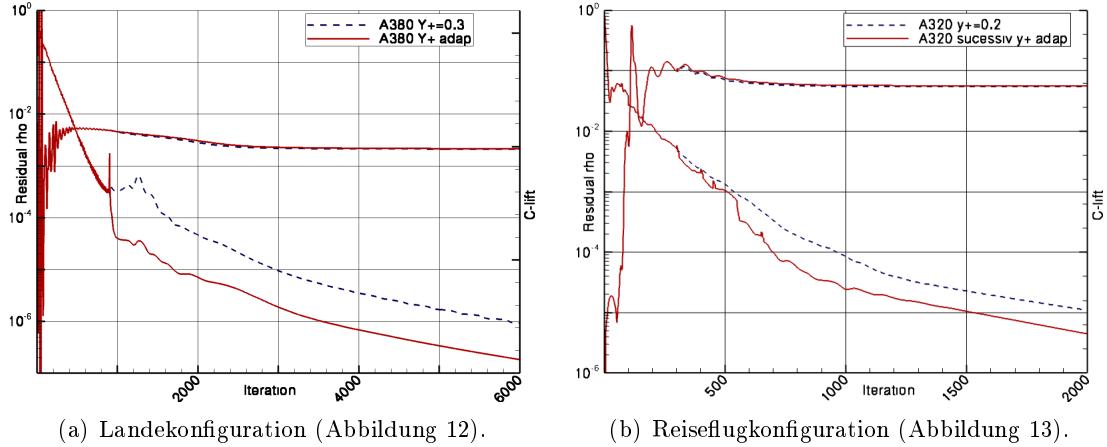


Abbildung 14: Vergleich der Konvergenzverläufe des Dichteresiduums und des Auftriebsbeiwerts (C_L) für die beiden Konfigurationen aus (A).

Konfigurationen nicht in der Lage gewesen $y(1)$ anzupassen. Dass dies jetzt bei komplexen Geometrien möglich ist, ist einzig und allein den zuvor beschriebenen grundlegenden Änderungen an den Datenstrukturen zu verdanken.

Abschließend lässt sich für die y^+ -Adaption auf *low-Reynolds* Netzen folgendes Fazit ziehen: Die y^+ -Adaption ist für den Einsatz in einer voll parallelen Prozesskette für industrielle Anwendungen geeignet und kann die Konvergenzrate und die Genauigkeit einer Rechnung verbessern.

8 Dynamische Verfeinerung hybrider Netze

Bei der Verfeinerung von Netzen handelt es sich im Allgemeinen um die Erhöhung der Netzauflösung in einem bestimmten Gebiet. Es gibt unterschiedliche Ansätze, dies zu erreichen. Zum Beispiel ist die Erhöhung der Netzauflösung auch mit den in Kapitel 7 beschriebenen Netzmanipulationsmethoden möglich, allerdings nur bei einer gleichzeitigen Reduzierung der Auflösung in einem anderen Bereich des Netzes. Diese Art der Netzverfeinerung kommt überwiegend bei strukturierten Netzen zum Einsatz, da aufgrund der speziellen Anordnung der Elemente und Netzpunkte (i, j, k -Indizierung) das Hinzufügen von neuen Elementen in ein bestehendes strukturiertes Netz zu aufwändig ist (siehe auch [71]).

Bei unstrukturierten bzw. hybriden Netzen wird eine lokale Verfeinerung des Netzes meistens durch das Einfügen neuer Punkte bzw. Elemente in das bestehende Netz realisiert. Grundsätzlich unterscheidet man dabei zwischen den Methoden die auf *Bisektions*-Algorithmen basieren und den *hierarchischen* Methoden (vgl. [6]). Der Vorteil der hierarchischen Methoden ist, dass das verfeinerte Netz wieder auf das ursprüngliche (initiale) Netz zurückgeführt werden kann, indem die durchgeführten Verfeinerungen sukzessive rückgängig gemacht werden. Um dies zu ermöglichen, muss allerdings die Verfeinerungshierarchie mit dem Netz mitgeführt werden. Ferner ist es notwendig, sich nur auf einen regulären Verfeinerungsfall pro Elementtyp zu beschränken, um eine Degeneration der

verfeinerten Elemente zu verhindern. Die anderen nicht regulären Verfeinerungsfälle eines Elementtyps existieren dann quasi als Übergangselemente, um die Konsistenz eines Netzes sicherzustellen (siehe auch (J)).

Ein wichtiger Punkt bei allen Verfeinerungsmethoden ist natürlich das Kriterium, welches festlegt wann ein Element oder eine Netzkanne verfeinert werden soll. Da die TAU-Code Adaption eigentlich nur ein Werkzeug ohne eigene Analysefähigkeiten ist, muss normalerweise die notwendige Intelligenz zur geschickten Auswahl von zu verfeinernen Gebieten bzw. Regionen von außen an das Programm herangetragen werden. Lediglich ein Satz von Kanten-Indikatoren, die auf Gradienten der Strömungsgrößen basieren, steht der TAU-Code Adaption zur Verfügung (vgl. auch (K)). Die folgende Übersicht zeigt eine Reihe von aerodynamischen Anwendungen, bei denen diese gradientenbasierten Kanten-Indikatoren ganz gute Ergebnisse erzielen :

- Verfeinerung des Stoßes bei stationären transsonischen Strömungen [29, 48, 77]
- Verfeinerung des Stoßes bei instationären transsonischen Strömungen [27]
- Verfeinerung des Stoßes bei hypersonischen Strömungen [41, 40, 50]
- Verfeinerung der Wirbelschleppen bei einer Landekonfiguration mit Hilfe des Totaldrucks p_{tot} [63]
- Verfeinerung der Triebwerksstrahlen bei einer Startkonfiguration mit Hilfe des Betrags des Geschwindigkeitsvektors $|\vec{v}|$ [64]

Es gibt allerdings eine Vielzahl von aerodynamischen Anwendungen, wo die auf Gradienten basierenden Sensoren unzureichend sind. Dazu zählen insbesondere Anwendungen in der Optimierung [17] bzw. Anwendungen mit stark abgelösten Strömungen [73].

Die nachfolgend aufgeführten Publikationen beschäftigen sich damit, in Abhängigkeit einer errechneten Lösung, das gegebene Netz sukzessive dort zu verfeinern, wo eine höhere lokale Auflösung notwendig ist. Bei den betrachteten Fällen handelt es sich ausnahmslos um aerodynamische Anwendungen mit stark abgelösten Strömungen.

- (C) **Investigation of Vortex Breakdown over a Pitching Delta Wing applying the DLR TAU-Code with Full, Automatic grid adaptation,**
 Thomas Alrutz, Markus Rütten.
Paper 5162, 35th AIAA Fluid Dynamics Conference, June 6-9 2005. Toronto, Canada.
- (D) **Improvement of the Automatic Grid Adaptation for Vortex Dominated Flows using Advanced Vortex Indicators with the DLR-Tau Code,**
 M. Widhalm, A. Schütte, T. Alrutz, M. Orlt.
Volume 96 of Notes on Numerical Fluid Mechanics and Multidisciplinary Design, pages 186-193, Springer, 2008.
- (E) **A Vortex Axis and Vortex Core Border Grid Adaptation Algorithm,**
 Markus Rütten, Thomas Alrutz, Holger Wendland.
International Journal for Numerical Methods in Fluids, DOI: 10.1002/fld.1792, 2008.

Die in (C) präsentierte Arbeit ist eine Art Machbarkeitsstudie (Proof of Concept). Die Motivation bestand hier darin zu zeigen, dass es unter Verwendung einer hierarchischen Netzverfeinerung auch möglich ist, mit einem Standardverfahren¹¹ bei der Simulation eines Nickmanövers sehr gute Ergebnisse zu erzielen. Die Idee war hier nicht wie etwa bei Morton et. al. [67], ein verbessertes Verfahren (DES) und ein sehr fein aufgelöstes Netz zu verwenden, sondern statt dessen die dynamische Netzverfeinerung mit einem auf Gradienten basierten Kanten-Indikator und speziell angepassten Parametereinstellungen einzusetzen. Besondere Aufmerksamkeit wurde dabei auf die Automatisierung der gesamten Manöversimulation gelegt.

Die in (C) verwendete dynamische Netzadaption, also die Anwendung von hierarchischer Verfeinerung und Entfeinerung, ist vom Verfasser dieser Arbeit nicht selbst implementiert worden. Jedoch basiert die gesamte Algorithmik für die hierarchische Entfeinerung auf den vom Verfasser entwickelten Datenstrukturen und Verfeinerungsfällen (vgl. (J)). Ferner sind sämtliche Verfeinerungsfälle, die in (K) bereits vorgestellt wurden, vom Verfasser für die TAU-Code Adaption entwickelt worden. Die Weiterentwicklung der vorhandenen Algorithmik und der Datenstrukturen in Richtung der dynamischen Ver- und Entfeinerung ist von Orlt durchgeführt worden (vgl. (G)).

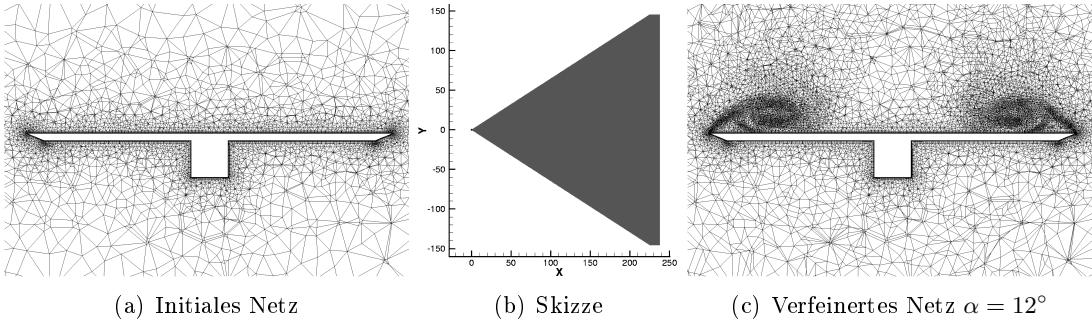


Abbildung 15: Skizze des Deltaflügels (b) und Schnitte durch die Netze bei $x = 200$.

Für den in (C) geschilderten Anwendungsfall ist die dynamische Netzverfeinerung mit den speziellen Parametereinstellungen allein noch nicht zielführend gewesen. Aufgrund der Eigenschaften des verwendeten Kanten-Indikators bestand bei der sukzessiven Anwendung der Adaption die Gefahr, dass sehr viele Elemente im Grenzschichtbereich verfeinert werden und nicht mehr genügend Netzpunkte in die Auflösung der Wirbelkerne gelangen. Um dieses Phänomen zumindest für die Unterseite des Deltaflügels auszuschließen, musste eine Methode implementiert werden, die verhindert, dass Elemente in den strukturierten Schichten oberhalb von reibungsbehafteten Wänden verfeinert werden können. Hierbei wurde der Extraktionsalgorithmus für die Element-Stapel (Prismen bzw. Hexaeder vgl. Kapitel 7) wiederverwendet, um über ausgewählten Wänden die Elemente zu identifizieren, welche von der Verfeinerung ausgeschlossen werden sollen. Mit den zuvor vorhandenen kartesischen Boxen war das Ausblenden dieser Bereiche nicht zweckmäßig. Des Weiteren wurde dieses Verfahren zu einem späteren Zeitpunkt der Simulation auch für die Oberseite des Deltaflügels eingesetzt, um die erzielte Auflösung

¹¹Instationäre RANS Simulation mit einem Spalart-Allmaras Turbulenzmodell [87] und Edwards Modifikation [18].

sozusagen einzufrieren und alle weiteren Netzpunkte in die Wirbel zu leiten (siehe auch Abbildung 16 (c)).

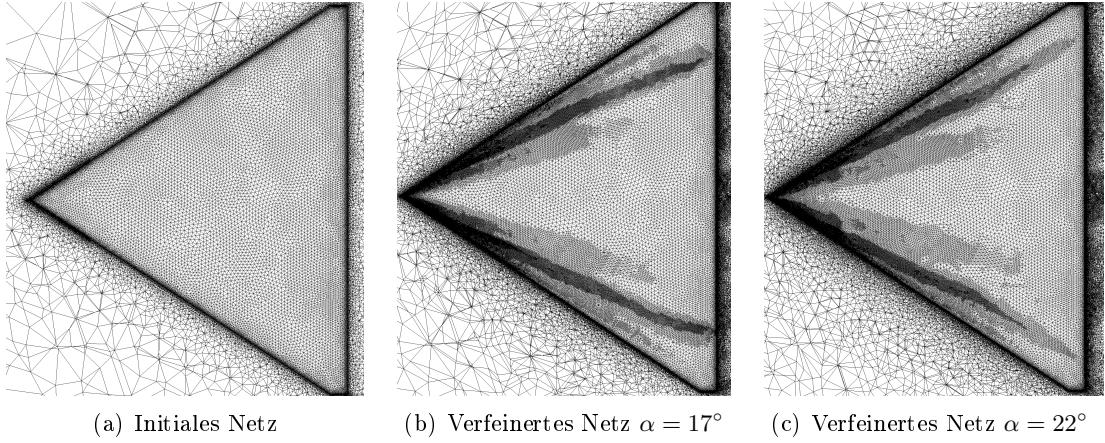


Abbildung 16: Schnitte durch die Netze bei $z = 0$ und verschiedenen Anstellwinkeln α .

Der Beitrag des Verfassers dieser Arbeit zu (C) bestand dabei nicht nur aus der Durchführung der Simulation sowie der Auswahl der Parametereinstellungen für die dynamische Netzadaption, sondern auch aus der Generierung des verwendeten initialen Netzes. Dieses Netz wurde mit dem Verfahren von Niederdrenk [97] erstellt und anschließend mehrfach verfeinert, um ein adäquates Startnetz für das Nickmanöver zu haben (Abbildung 15 und 16 (a)). Die Auswertung und Interpretation der Ergebnisse wurde von Rütten durchgeführt.

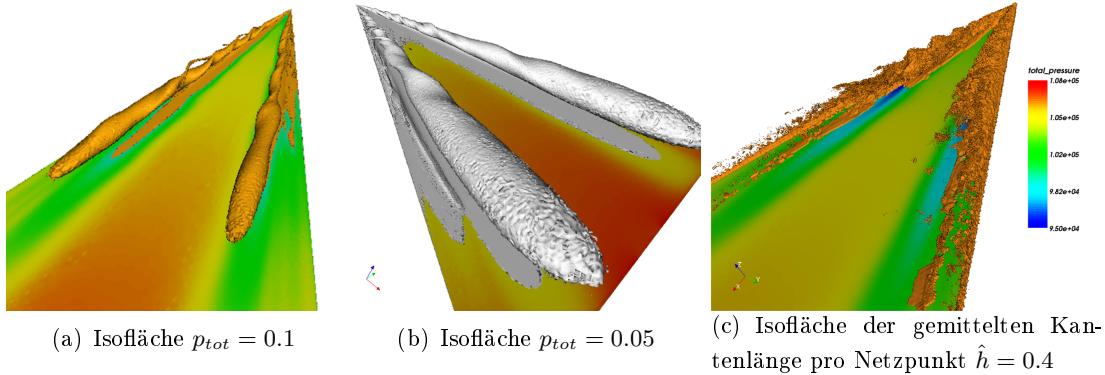


Abbildung 17: Visualisierung des Totaldrucks p_{tot} und der Kantenlänge \hat{h} bei $\alpha = 22^\circ$.

Für die Arbeiten (D) und (E) sowie die damit verbundene konsequente Weiterentwicklung der Adaption war der in (C) gewählte Ansatz richtungsweisend. Die sehr guten und detailreichen Auswertungsergebnisse (Abbildung 17 und 18), basierend auf den Arbeiten von Rütten [79, 80], führten schließlich dazu, dass die eigentlich für Postprocessing Aufgaben entwickelten Wirbelerkennungskriterien in der TAU-Code Adaption als spezielle Sensoren vom Verfasser dieser Arbeit verfügbar gemacht wurden.

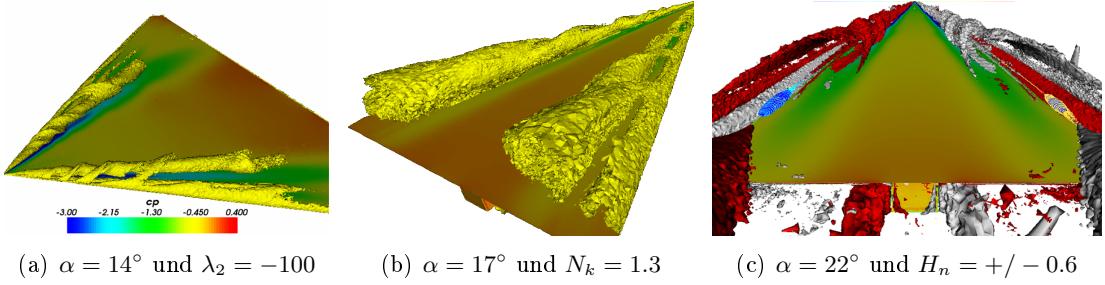


Abbildung 18: Visualisierung des Wirbelsystems mit unterschiedlichen Kriterien.

Hierzu hat Widhalm die entsprechenden Kriterien zur Wirbelerkennung als Ausgabewerte in den TAU-Strömungslösers implementiert :

1. invariant Q nach [12]
2. λ_2 Kriterium nach [38]
3. Kinematische Wirbelzahl N_k nach [94]
4. Normalisierte Helizität H_n nach [52]

Diese Werte konnten dann anschließend von der Adaption gelesen werden und mit Hilfe eines speziellen Schwellwert-Indikators zur Verfeinerung genutzt werden. Der hierfür vom Verfasser dieser Arbeit entwickelte Algorithmus entscheidet in Abhängigkeit von einer der vier zur Verfügung stehenden Größen (Q, λ_2, N_k, H_n), die der Nutzer angeben kann, welche der unten aufgeführten Vergleichsroutinen für den Kanten-Indikator I_e angewendet werden soll:

$$(8.1) \quad \Phi_e := \begin{cases} 1, & \text{falls } Q(p_1) > q \text{ oder } Q(p_2) > q \\ 0, & \text{sonst} \end{cases}$$

$$(8.2) \quad \Phi_e := \begin{cases} 1, & \text{falls } \lambda_2(p_1) < q \text{ oder } \lambda_2(p_2) < q \\ 0, & \text{sonst} \end{cases}$$

$$(8.3) \quad \Phi_e := \begin{cases} 1, & \text{falls } N_k(p_1) > q \text{ oder } N_k(p_2) > q \\ 0, & \text{sonst} \end{cases}$$

$$(8.4) \quad \Phi_e := \begin{cases} 1, & \text{falls } |H_n(p_1)| > q \text{ oder } |H_n(p_2)| > q \\ 0, & \text{sonst} \end{cases}$$

p_1 und p_2 sind hier jeweils die beiden Eckpunkte einer Kante e und q ein weiterer Parameter, der vom Benutzer gesetzt werden kann. Sinnvolle Werte für den Parameter q in Abhängigkeit der jeweiligen Größen (Q, λ_2, N_k, H_n) sind sowohl in (C) als auch in (D) diskutiert worden.

Der auf Seite 12 beschriebene gradientenbasierte Kanten-Indikator (4.1) wird dann folgendermaßen modifiziert:

$$(8.5) \quad I_e = \Phi_e h^\alpha$$

Durch diese Modifikation kann die Adaption genauso angewendet werden wie im Fall von (C). Die in (D) gezeigten Beispiele unterstreichen die Überlegenheit des implementierten Schwellwert-Indikators im Vergleich zu dem in (C) verwendeten Gradienten-Indikator. Allerdings ist auch die in (D) vorgestellte Variante noch nicht optimal für die automatische Erkennung und Verfeinerung von Wirbelsystemen. Das große Manko ist, dass der Indikator nur rein lokal auf einer Kante basiert. Damit unterliegt der Indikator gewissen Beschränkungen wie zum Beispiel der primär gewählten Netzaufteilung. Ist diese zu grob gewählt, ist auch der Schwellwert-Indikator mit den unterschiedlichen Größen (Q, λ_2, N_k, H_n) nicht in der Lage, die Wirbelsysteme zu erkennen. Ferner kann der Schwellwert-Indikator nur sukzessive eingesetzt werden, um zum Beispiel ein komplettes Wirbelsystem bis zum Fernfeldrand zu verfeinern (siehe auch Abbildung 19).

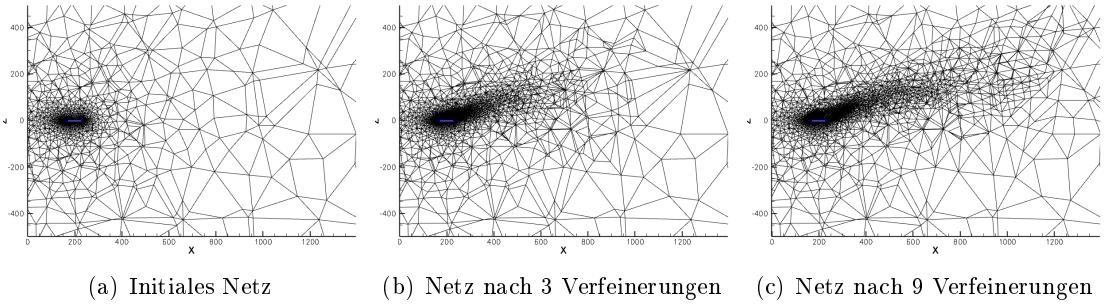


Abbildung 19: Schnitte durch die Deltaflügelnetze bei $y = -100$ und $\alpha = 12^\circ$.

In (E) wurde ein Verfahren entwickelt, dass diese Problematik berücksichtigt. Hier kommt ein von Rütten entwickeltes Postprocessing-Werkzeug zur Wirbelachsen-Bestimmung zum Einsatz. Mit Hilfe dieses Werkzeugs kann eine komplette Region bestimmt werden, in der das Netz zu verfeinern ist. Das Postprocessing-Programm führt die dazu notwendige Analyse zur Wirbelachsen- und Wirbelkern-Bestimmung durch und liefert für alle Netzpunkte Werte aus dem Intervall $[0, 1]$, wobei ein Wert von $\phi(p) = 1$ einen Punkt im Wirbelkern markiert. Diese Daten stellt das Analyseprogramm in einem Datensatz, ähnlich dem des TAU-Strömungslösers, zur Verfügung. Anschließend kann dieser Datensatz von der Adaption gelesen und unter Verwendung eines geeigneten Kanten-Indikators zur Verfeinerung genutzt werden. Da der Schwellwert in diesem Fall bereits vom Postprocessing-Werkzeug geliefert wird (vgl. (E) 4), kann man sich in der Adaption auf folgende Formulierung des Kanten-Indikators beschränken:

$$(8.6) \quad \Phi_e := \max(\phi(p_1), \phi(p_2))$$

Primär setzt die Adaption dabei auf der gleichen Algorithmik auf, die auch schon in (D) benutzt wurde, jedoch ist der externe Sensor des Postprocessing-Programms sehr viel genauer bzw. feiner (vgl. (E) 5.).

Der Beitrag des Verfassers zu den Arbeiten in (D) und (E) bestand im Wesentlichen aus der Entwicklung der Datenstrukturen und Algorithmen zur Nutzung der unterschiedlichen Kanten-Indikatoren. Dazu zählen natürlich auch die Algorithmik und die notwendigen Datenstrukturen zur Verwendung von beliebigen Strömungsgrößen des TAU-Solvers für die unterschiedlichen Kanten-Indikatoren.

Abschließend lässt sich für die Verwendung der Adaption mit speziellen Indikatoren für die Erkennung von Wirbelsystemen Folgendes festhalten:

- 1) Grundsätzlich kann man mit sehr feinen Parametereinstellungen der Adaption, dem gradientenbasierten Indikator nach (4.1) und mit dem Totaldruck (p_{tot}) als Sensor sehr gute Ergebnisse erzielen. Dies zeigt insbesondere die Arbeit von Fritz [25], der für eine komplexe aerodynamische Konfiguration mit der Adaption deutliche bessere Ergebnisse erzielt hat als mit vorverfeinerten Netzen (vgl. Kapitel 10).
- 2) Die in (D) vorgestellten Sensoren sind bei einem hinreichend gut aufgelösten Netz eine deutliche Verbesserung zu der in (C) vorgestellten Methodik. Die neuen Netzpunkte werden in den Regionen eingefügt, wo auch die Wirbel liegen. Es ist daher zu erwarten, dass die Verwendung dieser Sensoren im Vergleich zu (C) zu einer Reduzierung der gesamten Netzpunktanzahl führen wird. Ferner ist es nicht notwendig, ein Parameterfeintuning vorzunehmen. Allerdings ist die Wahl eines geeigneten Schwellwertes q (vgl. (8.1)-(8.4)) nicht ganz trivial, wie die Diskussion der Ergebnisse in (D) zeigt.
- 3) Für eine detaillierte und qualitativ hochwertige Auflösung der Wirbelstrukturen dürfte die in (E) vorgestellte Methodik am besten geeignet sein. Insbesondere die Auflösung von sekundären bzw. tertiären Wirbelstrukturen ist mit diesem Ansatz am ehesten möglich. Für eine effiziente Einbindung in die TAU-Code Prozesskette ist es allerdings notwendig, die Datensätze der Wirbelachsen und Wirbelkerne kompakt (als Verfeinerungsgebiet) an die Adaption weiterzugeben.

9 Effiziente Datenstrukturen für paralleles verteiltes Rechnen

Der fortwährend steigende Bedarf an immer genaueren und detaillierten Simulationen bei gleichzeitiger Reduzierung der Turn-Around-Zeiten haben im vergangenen Jahrzehnt dazu geführt, dass überall auf der Welt die Installationen von Cluster-Systemen mit verteiltem Speicher enorm zugenommen haben (vgl. [89] und [90]). Während vor ca. 10 Jahren noch die klassischen Vektor-Rechner und SMP¹²-Systeme die HPC Szene dominierten, sind dies heutzutage vor allem Cluster-Systeme mit Tausenden von CPU-Cores [88, 39, 42]. Bei diesen Cluster-Systemen kommen spezielle Hochgeschwindigkeitsnetze für die Kommunikation zum Einsatz, die bei der Anwendungsperformance und Skalierung eine nicht unerhebliche Rolle spielen. Für die zu erwartende parallele

¹²Shared-Memory Parallelisierung

Performance der Anwendungen, sind Kennzahlen wie die Latenz¹³ und Bandbreite¹⁴ wichtige Parameter.

Damit die Anwendungsprogramme diese massiv parallelen Systeme auch effizient ausnutzen können, sind neben den speziell für verteilten Speicher angepassten Datenstrukturen auch Algorithmen notwendig, die in Bezug auf den begrenzten Speicher pro CPU möglichst gut skalieren. Insbesondere gilt das für den Fall, dass eine Netzadaption während der Strömungssimulation durchgeführt werden soll. Sollte auch nur eines der Programme einer Simulationskette (vgl. Abbildung 5) nur sequentiell verfügbar sein, dann ist die effiziente Nutzung von vielen CPUs eines Clusters nur bedingt möglich.

Daraus folgt, dass nur eine voll parallelisierte und in Hinblick auf die Ressourcennutzung flexible Programmsuite in der Lage ist, große Clusterinstallationen effizient auszulasten. Die beiden nachfolgend aufgelisteten Publikationen beschäftigen sich zum einem damit, die Anforderungen an die Hardware für den TAU-Code zu formulieren und zum anderen mit der Präsentation von Datenstrukturen und Algorithmen für die effiziente parallele Adaption verteilter hybrider Netze mit anschließender Repartitionierung in einer parallelen Prozesskette.

(F) **Investigation of the parallel performance of the unstructured DLR-TAU-Code on distributed computing systems,**

Thomas Alrutz.

Proceedings of the 17th Parallel Computational Fluid Dynamics Conference, pages 509-516, 2005. Elsevier.

(G) **Parallel dynamic grid refinement for industrial applications,**

Thomas Alrutz and Matthias Orlt.

Proceedings ECCOMAS CFD 2006 Conference, September 5–8 2006. Egmond aan Zee, The Netherlands.

Bei der Arbeit (F) bestand die Motivation darin, eine möglichst genaue Untersuchung aktueller Hardware im Zusammenspiel mit dem TAU-Code und industrierelevanten Konfigurationen durchzuführen. Besonderes Augenmerk wurde dabei auf eine Testumgebung im realen Produktionsumfeld gelegt. Das heißt, dass die getesteten Clustersysteme für die durchgeführten Benchmarks nicht exklusiv genutzt wurden, sondern auch noch von anderen Applikationen belegt waren, so wie dies bei einem normalen Produktionssystem auch zu erwarten ist. Des Weiteren wurden die zugrunde liegenden Simulationsszenarien so ausgewählt, dass ein Rückschluss auf die benötigten Ressourcen bzw. Kennzahlen wie die Latenz und Bandbreite des Kommunikationsnetzwerks bzw. die Prozessor-Architektur und deren Anbindung an den Hauptspeicher möglich ist. Insbesondere die Untersuchung des Einflusses der Latenz und der Bandbreite des Hauptspeichers auf die Performance des TAU-Codes war der Grund, zwei unterschiedliche Verfahren zur Laufzeit-Optimierung zu verwenden (vgl. (F) 2.5).

Ferner stellt die Arbeit in (F) eine Art Leitfaden für die Anwender des TAU-Codes

¹³Unter Latenz versteht man im Allgemeinen die minimale Zeit, die für einen Verbindungsaubau zur Datenübertragung benötigt wird.

¹⁴Die Bandbreite einer Datenverbindung ist die maximale Anzahl an Daten, die in einem bestimmten Zeitintervall (pro Sekunde) übertragen werden kann.

CPUs	1	4	16	32	64	128	256
Iteration 4w	29.57s	9.13s	2.54s	1.50s	0.92s	0.64s	0.51s
Speedup 4w	1.00	3.24	12.05	19.75	32.14	46.20	57.98
Effizienz 4w	1.00	0.81	0.75	0.62	0.50	0.36	0.23
Iteration 3v	28.81s	8.86s	2.10s	1.22s	0.67s	0.39s	0.25s
Speedup 3v	1.00	3.40	13.74	23.68	42.78	73.68	116.78
Effizienz 3v	1.00	0.85	0.86	0.74	0.67	0.58	0.46

Tabelle 2: Ausgewählte Zeiten, relativer Speedup sowie parallele Effizienz für einen 3v bzw. 4w Multigitterzyklus (vgl. Abb. 20) mit dem DLR-F6 Fall (vgl. Abb. 21).

dar, in welchem Umfang die Ressourcen eines Cluster-Systems bei einer typischen Strömungssimulation sinnvoll eingesetzt werden sollten.

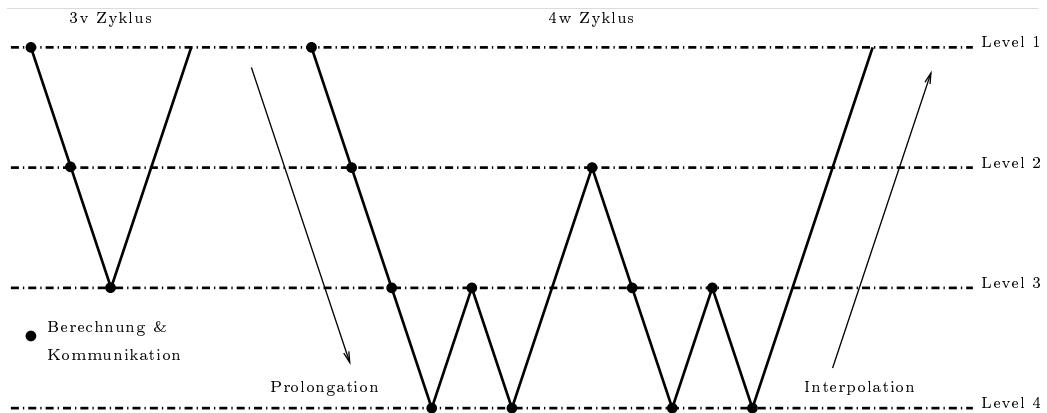


Abbildung 20: Skizze eines 3v und 4w Multigitter Zyklus.

Zum Beispiel ist es nicht sinnvoll, die DLR-F6 Konfiguration (Abbildung 1 (b)) mit mehr als 64 CPUs auf einem Cluster wie dem in [21] zu rechnen, da sonst die parallele Effizienz¹⁵ auf 50% bzw. darunter sinkt (vgl. Tabelle 2 Effizienz 4w). Die Grenze von 50% der parallelen Effizienz ist insofern sinnvoll, da bei einer Hinzunahme von mehr CPUs für eine parallele Rechnung weniger als die Hälfte der theoretischen Anwendungsperformance tatsächlich genutzt wird.

Grundsätzlich ist die parallele Effizienz eines Codes ein recht gutes Maß für die Einschätzung einer sinnvollen Verwendung von CPU-Ressourcen. Es gibt jedoch Anwendungen bzw. Clusterkonfigurationen bei denen die Grenze von 50% bewusst unterschritten wird. Dies ist fast immer der Fall, wenn der Cluster über ein sehr langsames Kommunikationsnetz (vgl. (F) Abb. 5) oder nur über sehr wenig Hauptspeicher pro Cluster-Knoten verfügt. Ist zu wenig Hauptspeicher vorhanden, muss der Anwendungsfall auf so viele Cluster-Knoten verteilt werden, dass während der Rechnung keine Daten auf die Festplatte ausgelagert werden müssen.

Eines der Ergebnisse der Arbeit in (F) war, dass der TAU-Code Strömungslöser weitaus stärker von einer hohen Bandbreite zum Hauptspeicher profitiert, als von einem höheren

¹⁵bezogen auf die Performance des TAU-Codes bei der Nutzung von einer CPU

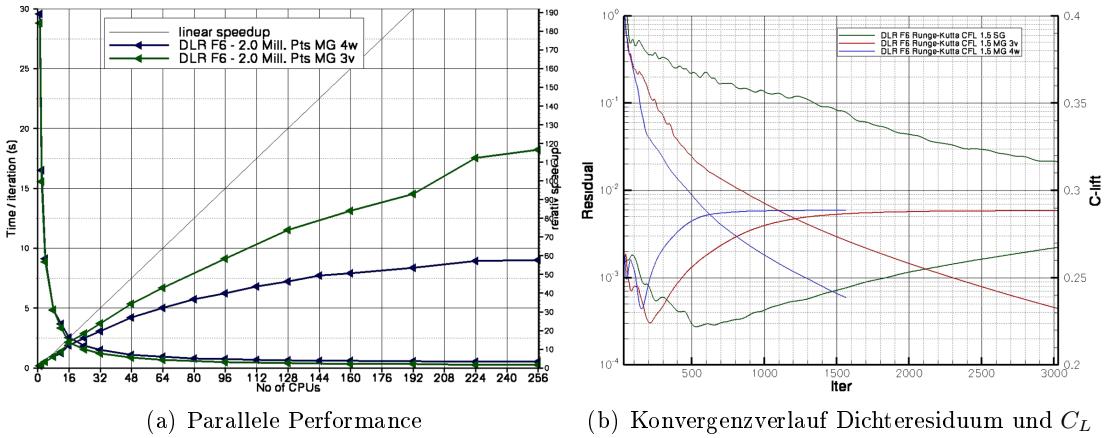


Abbildung 21: Vergleich unterschiedlicher Multigitterzyklen mit dem DLR-F6 (2.0×10^6 Netzpunkte). Parallele Performance TAU-Code (a), Konvergenzverlauf (b).

CPU Takt (siehe (F) 3.1). Dies liegt unter anderem an der indirekten Adressierung der Daten im TAU-Code Strömungslöser, die aufgrund des unstrukturierten Finite-Volumen Verfahrens notwendig ist (vgl. (F) 2.5).

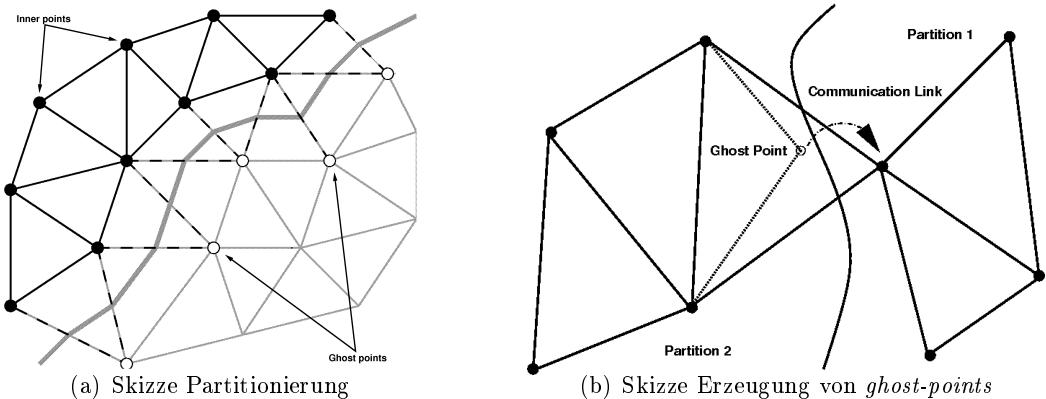


Abbildung 22: Parallelisierung des TAU-Code Strömungslösers.

Eine weitere Beobachtung war, dass bei der Verwendung der Multigitter-Beschleunigung (vgl. [57]) eine erhebliche Last-Inbalance auf den gröberen Netzen entsteht, die letztlich die parallele Performance bei 3v bzw. 4w Multigitter Zyklen beschränkt (vgl. Abbildung 21 (a)). Der Verlust an paralleler Effizienz wird dabei maßgeblich von den unausgeglichenen Netz-Partitionen der Grobgitter bestimmt. Tabelle 3 zeigt einen Vergleich der inneren Netzpunkte in Relation zu den minimalen und maximalen zusätzlichen Punkten, den so genannten *ghost-points* (siehe Abbildung 22), die für den Datenaustausch auf dem jeweiligen Gitterlevel notwendig sind. Es ist klar ersichtlich, dass auf den gröberen Netzebenen die Anzahl der zusätzlichen Punkte für die Kommunikation erheblich schwankt. Insbesondere bei der F6 Konfiguration ist zu sehen, dass auf dem grössten Netz (Level 4) die Anzahl der zusätzlichen Punkte sogar das Doppelte der inneren Punk-

	A380 9.6×10^6 Punkte		F6 2.0×10^6 Punkte			
	innere	<i>ghost-points</i>	innere	<i>ghost-points</i>		
Gitter		min	max		min	max
Level 1	312200	3%	3%	63000	10%	12%
Level 2	40500	10%	11%	9800	8%	27%
Level 3	6000	10%	20%	1500	20%	60%
Level 4	1000	24%	50%	300	50%	200%

Tabelle 3: Übersicht der Variation der dualen Gittergrößen bei der Verwendung von 32 Partitionen für die beiden Anwendungsfälle aus (F). Vergleich zwischen inneren Netzpunkten und zusätzlichen *ghost-points* für den Kommunikationsaustausch.

te betragen kann. In diesem Fall überwiegt der Aufwand für die Kommunikation den Aufwand für die Berechnung, da die Lastbalancierung der Grobgitterpartitionen stark unausgeglichen ist. Betrachtet man in diesem Zusammenhang noch den durchgeföhrten Multigitterzyklus 4w, dann ist klar, dass überwiegend auf den groben Gitterleveln 3 und 4 sowohl gerechnet als auch kommuniziert wird (4 mal auf Level 4 und 4 mal auf Level 3 siehe auch Abbildung 20). Bei dem in Tabelle 3 auch aufgeföhrten größeren Testfall (A380) ist dies nicht so dramatisch und daher ist die parallele Effizienz bei Verwendung von 32 CPUs auch deutlich besser (siehe (F) 4.).

Allerdings darf man diese Beobachtung nicht fehlinterpretieren, da die Multigitter Beschleunigung bei dem DLR-F6 Anwendungsfall (4w Zyklus anstatt 3v Zyklus) den Verlust an paralleler Effizienz mehr als ausgleicht. In Abbildung 21 (b) ist zu sehen, dass der Code ca. 1300 Iterationen (1950s bei 32 CPUs) im 4w Zyklus benötigt, um einen auskonvergierten Auftriebsbeiwert zu erzielen (Dichteresiduum bei 10^{-3}). Im 3v Zyklus benötigt der Code hingegen ca. 2300 Iterationen (2806s bei 32 CPUs), um ein ähnlich gutes Ergebnis zu erzielen. Generell darf die parallele Effizienz eines Codes nicht für sich alleine betrachtet werden, sondern es ist immer wichtig, auch die algorithmische Effizienz mit einzubeziehen.

Ein anderer Aspekt im Bereich des effizienten parallelen Rechnens ist natürlich auch die tatsächliche realisierte Auslastung einer Clusterinstallation. Üblicherweise kommen sogenannte *Batch-Systeme* beim Betrieb eines Cluster-Systems zum Einsatz um die Ressourcen möglichst gut auszulasten [66, 82]. Die Entwickler von Batch-Systemen bzw. der dazugehörigen Scheduling-Software haben dazu unterschiedliche Methoden zur Auslastungsoptimierung in ihre Software integriert. Eine dieser Methoden ist *Backfilling* [11]. Das Ziel von *Backfilling* ist, durch vorausschauende Planung möglichst viele CPUs einer Clusterinstallation auszulasten. Dazu wird versucht, die Lücken aufzufüllen, welche durch unterschiedliche Ressourcenanforderungen und Reservierungen entstehen. Wird auf der Anwendungsseite ein flexibler Ressourceneinsatz unterstützt, etwa durch den Einsatz eines Netzpartitionierers, der erst nach der eigentlichen Ressourcenvergabe die passende Partitionierung erzeugt, und unterstützt das Batch-System die variable Angabe von CPU-Ressourcen (vgl. [65] Kapitel 6, Seite 159 ff), dann kann dies in Verbindung mit *Backfilling* zu einer Erhöhung der Auslastung genutzt werden (siehe Beispiel in Abbildung 23 und 25). Diese Flexibilität auf Seite der Anwendung (hier der Netzpartitionierer) muss allerdings mit einem effizienten Umgang der Ressourcen erfolgen (in

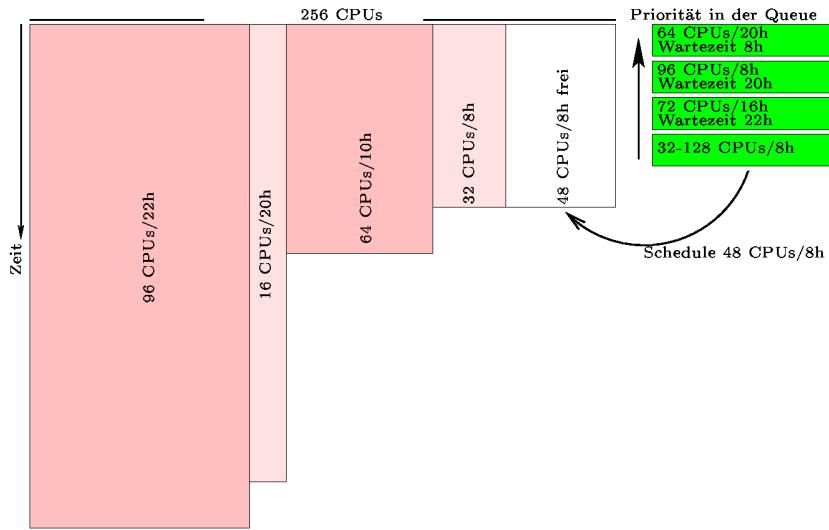


Abbildung 23: Strategie des *Backfillings*. Der Batch-Job an der 4. Stelle kann aufgrund der Ressourcenanforderungen (32-128 CPUs und 8h Laufzeit) vorgezogen werden, da alle anderen wegen ihrer Ressourcenanforderungen nicht abgearbeitet werden können.

diesem Fall der zur Verfügung stehenden Zeit).

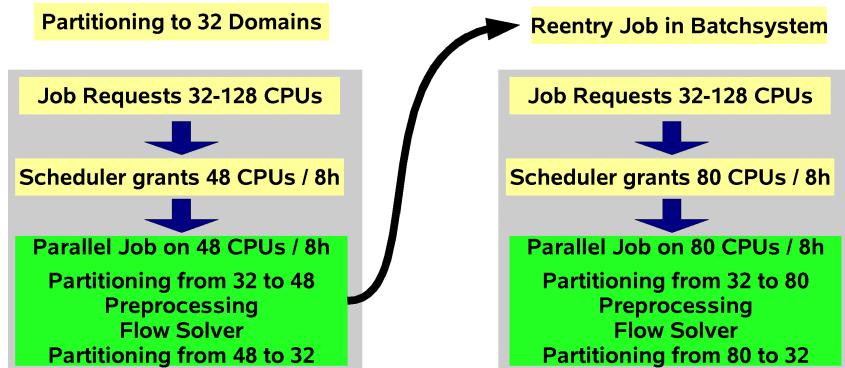


Abbildung 24: Skizze eines Batch-Jobs mit variabler CPU-Ressourcenangabe und der Rolle des dynamischen Netzpartitionierers innerhalb der Simulationskette.

Ein gutes Beispiel für das Zusammenspiel von Batchsystem mit Backfilling und parallelem Netzpartitionierer mag folgendes Szenario bieten. Ist eine Simulationsrechnung so ausgelegt, dass sie auf mindestens 32 CPUs laufen muss (aufgrund von Hauptspeicherbedarf) und kann sie auf bis zu 128 CPUs laufen (aufgrund von Effizienzüberlegungen), dann kann vor dem Starten der eigentlichen Rechnung die Partitionierung auf 32 Partitionen erfolgen und dann bei Bedarf die jeweilige Umpartitionierung von 32 auf N Partitionen zur Laufzeit der Rechnung durchgeführt werden (siehe Abbildung 24).

In diesem Szenario ist der parallel arbeitende Netzpartitionierer genau dann von Vorteil, wenn sich durch die vorher erfolgte Aufteilung der Netzdaten in 32 Partitionen nachträglich auch ein Geschwindigkeitsgewinn erzielen lässt (siehe Beispiel in Tabelle

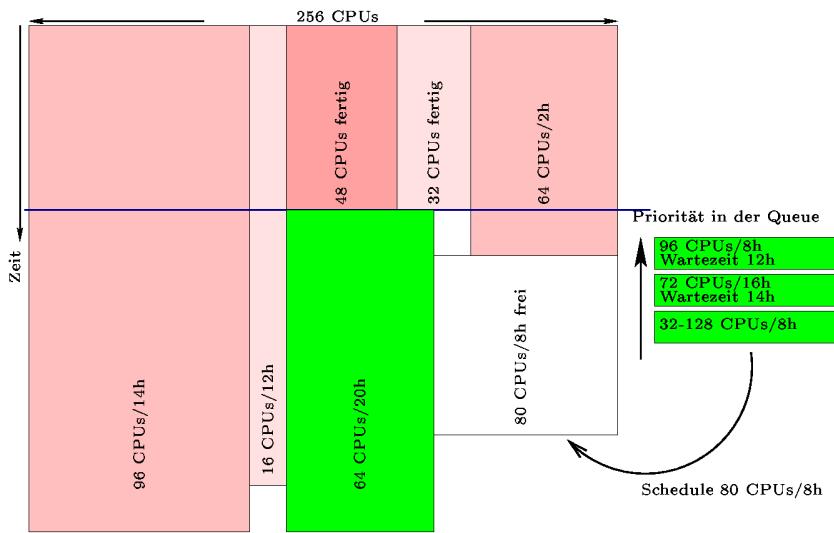


Abbildung 25: Die variable Angabe von CPU-Ressourcen ermöglicht dem Batch-Job an der 3. Stelle die Nutzung aller freien Ressourcen. Durch *Backfilling* wird dieser Job dazu genutzt, die sonst brachliegenden CPUs auszulasten.

4). Sollte während der parallelen Simulationsrechnung auch noch eine Netzverfeinerung bzw. Entfeinerung zum Einsatz kommen, dann ist der parallele Netzpartitionierer absolut unverzichtbar, da nach erfolgter Netzverfeinerung die Lastverteilung auf den einzelnen Partitionen mit ziemlicher Sicherheit nicht mehr ausgeglichen ist.

Die Arbeit (G) beschäftigt sich mit den notwendigen Maßnahmen sowie den Erweiterungen der vorhandenen Datenstrukturen im TAU-Code, um die oben geschilderten Anforderungen an den Netzpartitionierer effizient umzusetzen. Es wird unter anderem ein Verfahren vorgestellt, welches auf Hardware Plattformen mit verteiltem Datenspeicher (*distributed data*) effizient eingesetzt werden kann. Ferner beinhaltet Arbeit (G) die Beschreibung der Erweiterung der TAU-Netzadaption für die Nutzung in einer parallelen Simulationskette sowie eine Untersuchung des Einflusses der Stapelerhaltung im Partitionierer (vgl. Kapitel 7) auf die Performance des Strömungslösers.

Die Schwierigkeit bei der Umsetzung der unterschiedlichen Anforderungen bestand dabei in mehrfacher Hinsicht. Zunächst musste ein Datenlayout entworfen werden, so dass die Netzadaption parallel wie auch sequentiell in der gleichen Art und Weise funktionieren kann. Diese Grundbedingungen stellt gewisse Einschränkungen an die aufzubauende Datenstruktur sowohl für die Netzadaption als auch für den Netzpartitionierer dar. Da die Netzadaption bei der Verfeinerung auf direkte lokale Indizierung aufbaut (vgl. (G) 4.2), muss der Netzpartitionierer nicht nur die Netzdaten verteilen, sondern auch gleichzeitig den Transfer zwischen dem sogenannten *globalen Index Raum* und dem *lokalen Index Raum* durchführen (vgl. (G) 3.4). Eine weitere Problematik stellen die Chimera-Netz Informationen dar, die ebenfalls auf den einzelnen Partitionen in lokaler Indizierung vorhanden sein müssen (vgl. [54]). Hierbei ist es für die Konsistenz der Chimera-Daten absolut wichtig, dass auf allen Netzpartitionen die Reihenfolge der einzelnen Chimera-Netzblöcke nicht durch die Repartitionierung durcheinander gerät. Eine besondere Schwierigkeit liegt bei der Umsetzung der parallel dynamischen

Netzadaption mit anschließender Repartitionierung darin, dass die Verfeinerungshierarchie effizient abgespeichert und auch korrekt behandelt werden muss. Lepage et al. haben in [51] ein Verfahren für unstrukturierte Netze vorgestellt, das die gesamte globale Hierarchie-Information auf allen Netzpartitionen abspeichert. Diese Vorgehensweise ist zwar recht geradlinig, aber leider ist aufgrund der globalen Referenzstruktur eine Skalierung in Bezug auf den Speicherbedarf ausgeschlossen. Somit ist diese Lösung denkbar ungeeignet für den Einsatz auf Systemen mit verteiltem Speicher.

Der in (G) vorgestellte Ansatz setzt auf eine verteilte Hierarchie-Information mit einer lokalen Referenzstruktur, was hinsichtlich der Skalierbarkeit in Bezug auf den Speicherbedarf nahezu optimal ist, aber die Verwaltung und den Aufbau der Hierarchie erheblich erschwert.

Der Beitrag des Verfassers dieser Arbeit zu (G) besteht hauptsächlich in der Entwicklung der verteilten Netz- und Hierarchie-Datenstrukturen für die parallele dynamische Netzadaption sowie der Entwicklung der Algorithmen für den parallelen dynamischen Netzpartitionierer. Der Hauptteil der Arbeit an dem Netzpartitionierer entfiel dabei auf die Entwicklung eines skalierbaren Verfahrens zur Migration der Netzelemente und der

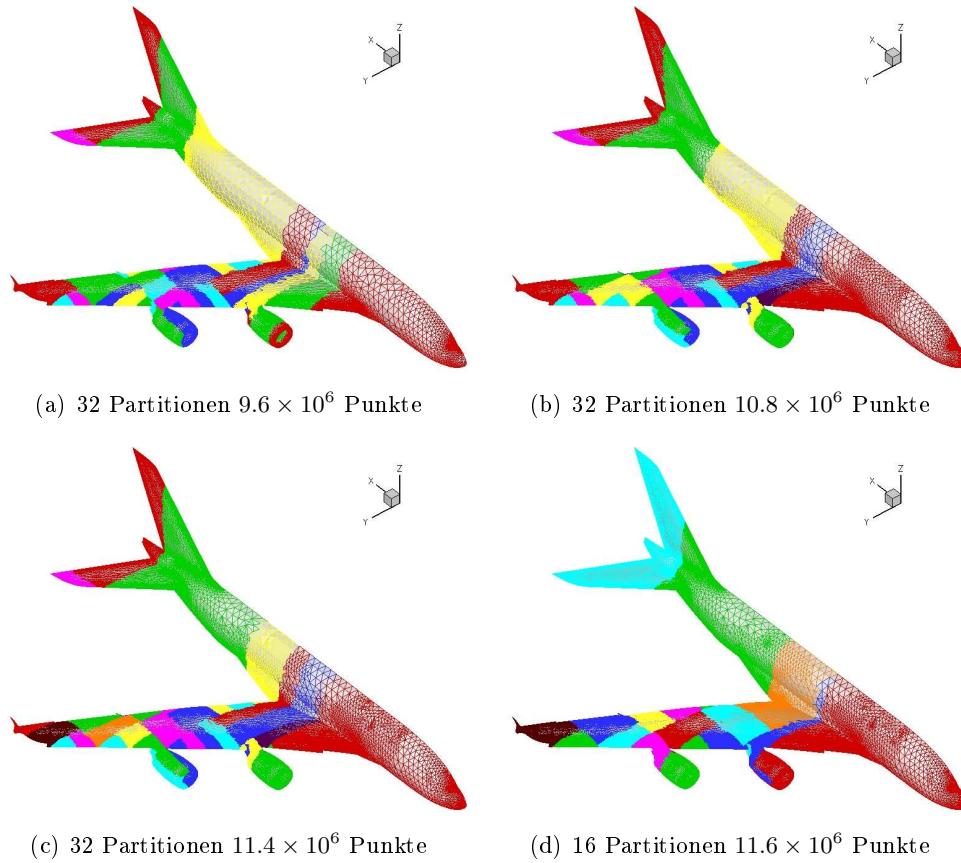


Abbildung 26: Dynamische Netzadaption mit anschließender Repartitionierung für die A380 Konfiguration. (a) Startnetz, (b) 1. Adaption, (c) 2. Adaption und (d) 3. Adaption.

Hierarchie. Des Weiteren musste die konsistente Behandlung der Zusatzinformationen

für Chimera-Netze [54] und für Netze mit periodischen Rändern bzw. Wirkscheiben (*Actuator Disc*) [74] sichergestellt werden. Die in (G) gezeigten Beispielrechnungen sind ebenfalls vom Verfasser dieser Arbeit durchgeführt worden. Die Parallelisierung der Netzverfeinerung wurde von Orlt umgesetzt.

Ein anschauliches Beispiel für die dynamische parallele Netzadaption mit anschließender Repartitionierung ist in Abbildung 26 zu sehen. Hier wurde die A380 Konfiguration aus (F) dazu verwendet, die dynamische Repartitionierung zu verdeutlichen. Die unterschiedlichen Farben stellen jeweils die einzelnen Netzpartitionen dar. Es ist sehr gut zu sehen, wie sich die Oberflächennetze der einzelnen Partitionen von Adaptionsschritt zu Adaptionsschritt verändern und gleichzeitig die Oberfläche in Teilbereichen verfeinert wird.

Anzahl CPUs (N)	32	48	64	96	96
Partitionierung M -> N	24 -> 32	32 -> 48	48 -> 64	64 -> 96	96 -> 1
Speicherbedarf pro Prozess	104 MB	89 MB	72 MB	52 Mb	30 MB ¹⁶
Rechenzeit parallel M -> N	144s	166s	177s	218s	291s
Rechenzeit sequentiell 1 -> N	505s	648s	795s	1056s	-/-

Tabelle 4: Rechenzeit auf [21] für den sequentiellen bzw. parallelen Netzpartitionierer mit der A380 Konfiguration (11.6×10^6 Punkte und 2.2×10^6 Hierarchie-Elementen).

In Tabelle 4 ist für das letzte Netz der A380 Konfiguration (Abbildung 26 (d)) der Speicherbedarf und die Laufzeit des Netzpartitionierers bei unterschiedlichen Umpartitionierungen angegeben. Zum Vergleich ist in der Tabelle auch noch die Laufzeit für den sequentiellen Netzpartitionierer angegeben¹⁷. Hier ist klar zu sehen, dass sich durch die Verwendung des parallelen Netzpartitionierers bei der Umpartitionierung ein Zeitvorteil erreichen lässt. Die sukzessive Anwendung des parallelen Netzpartitionierers auf die jeweiligen schon vorhandenen Partitionen des Netzes ist mit deutlich weniger Zeitaufwand durchführbar als die sequentielle Anwendung auf das nicht partitionierte Netz.

Tabelle 5 gibt einen guten Eindruck von der parallelen Performance der Netzadaption mit anschließender Repartitionierung für ein etwas kleineres Anwendungsbeispiel (vgl. (G) 6.1).

Anzahl CPUs	1	8	16	24	32	48	64	96	128
Netzadaption	560s	99s	47s	37s	28s	24s	20s	14s	14s
Repartitionierung	-/-	31s	18s	18s	18s	19s	15s	15s	15s
Speedup ¹⁸	1.00	4.31	8.62	10.18	12.17	13.02	16.00	19.31	19.31

Tabelle 5: Parallele Performance der TAU-Adaption auf [21] mit anschließender Repartitionierung für ein hybrides Netz bestehend aus 4.47×10^6 Punkten, 15×10^6 Volumenelementen und 4.17×10^6 Hierarchie-Elementen.

Als Abschluss ist in Tabelle 6 noch die Skalierung der y^+ -Adaption angegeben, wobei auch hier die A380 Konfiguration (vgl. Abbildung 12 auf Seite 26) verwendet wurde.

¹⁶3.3 GByte für den Prozess der alle Daten empfängt und abspeichert.

¹⁷Bei einem Speicherverbrauch von 2.7 GByte und ohne Migration der Hierarchie Daten.

Anzahl CPUs	1	8	16	32	48	64
y^+ -Adaption	698s	111s	71s	33s	21s	16s
Speedup	1.00	6.29	9.83	21.15	33.23	43.62
Speicherbedarf in MByte	3200	631	376	247	186	186

Tabelle 6: Beschleunigung der y^+ -Adaption durch die Parallelisierung für eine komplexe Landekonfiguration (Abb. 12) gerechnet auf [21].

Abschließend bleibt zu bemerken, dass Arbeit (F) neben den bereits beschriebenen Ergebnissen auf Seite 37, 38 und 39 auch als Grundlage für diverse Entscheidungen in Bezug auf Hardwareauswahl für die Nutzung des TAU-Codes gedient hat (vgl. [23]). Des Weiteren war die Arbeit richtungsweisend bzgl. der Optimierung des TAU-Codes auf skalaren Rechnerarchitekturen (vgl. (O)).

Bezüglich des in (G) präsentierten Ansatzes zur Parallelisierung der Adaption sowie des Netzpartitionieres bleibt festzuhalten, dass der gewählte Weg für Systeme mit verteiltem Speicher sehr gut in Bezug auf den Speicherbedarf skaliert. Die Skalierung in Bezug auf die Rechenzeit ist bei der y^+ -Adaption aufgrund der *super-cell* Partitionierung (vgl. (G) 3.2) auch sehr gut (siehe Tabelle 6). Die Netzadaption mit anschließender Repartitionierung zur Lastbalancierung skaliert auch recht akzeptabel (vgl. auch Beispiel in (G) 6), jedoch nicht so gut wie die y^+ -Adaption. Dies liegt unter anderem an der Problematik, dass sich die Verfeinerungsgebiete im Netz nicht a-priori vorhersagen lassen (dazu wird die Netzadaption ja grade benutzt) und daher ist es auch nicht möglich für die Netzadaption eine vernünftige Lastbalancierung zu erzeugen. Der parallele Netzpartitionierer weist aufgrund des sehr hohen Kommunikationsanteils keine so gute Skalierung wie die Adaption auf. Bei der Umpartitionierung müssen sehr viele und unterschiedliche Daten durch das Netzwerk geschickt werden, was immer einen gewissen Overhead¹⁹ zur Folge hat. Bei der Repartitionierung ist der Anteil der Kommunikation zwar etwas geringer als bei der Umpartitionierung, jedoch hängt die Skalierung immer auch von den jeweiligen verfeinerten Partitionen ab. Im schlimmsten Fall ist nur in einer Partition eine Verfeinerung erfolgt, aber alle anderen Partitionen müssen aufgrund des verwendeten rekursiven Bisektions-Algorithmus im Partitionierer nicht nur Netzdaten empfangen, sondern auch noch verschicken. Dieses Szenario ist zwar höchst unerfreulich, aber es läuft zumindest voll parallel.

Der Hauptvorteil der präsentierten Parallelisierung erschließt sich allerdings erst auf den zweiten Blick. In einer Simulationskette umgehen die parallele Adaption und der parallele dynamische Partitionierer den bisherigen Flaschenhals. Die beiden Programme (in ihrer parallelen Anwendung) ermöglichen einen sehr flexiblen Ressourceneinsatz (siehe Beispiel *Backfilling* in Abbildung 23 und 25) und ermöglichen so an unterschiedlichen Stellen der Simulationskette die Turn-Around Zeiten für typische Strömungssimulationen weiter zu reduzieren.

¹⁸Für die Netzadaption inkl. anschließender Repartitionierung.

¹⁹der von der Größe des Netzes abhängt

10 Komplexe Anwendungen

Die in diesem Kapitel zusammengefassten Publikationen befassen sich mit der konkreten Anwendung der in den Kapiteln 7 - 9 beschriebenen Verfahren. In diesem Kapitel soll auch deutlich werden, warum das interdisziplinäre Gebiet des wissenschaftlichen Rechnens nicht nur aus der Anwendung bzw. der Entwicklung eines Strömungslösers (mit entsprechender Modellierung der Physik) besteht, sondern auch aus vielen verschiedenen anderen Disziplinen unter anderem der Validierung des Strömungslösers durch hochwertige experimentelle Untersuchungen.

Die beiden unten aufgeführten Artikel beschreiben anhand des DLR-Projekts SikMa (Simulation komplexer Manöver) [20] die Notwendigkeit zur Kopplung verschiedener Disziplinen um eine Software-Kette zur genauen Simulation und Untersuchung von instationären aerodynamischen Phänomenen bei einem Flugzeug im Manöverflug aufzubauen.

- (H) **Numerical simulation of maneuvering combat aircraft,**
Andreas Schütte, Gunnar Einarsson, Britta Schöning, Thomas Alrutz, Wulf Mönich, Jens Neumann, Jörg Heinecke.
High Performance Computing in Science and Engineering' 05. Part 3, pages 185-196, Springer, 2006.
- (I) **Prediction of the Unsteady Behavior of Maneuvering Aircraft by CFD Aerodynamic, Flight-Mechanic and Aeroelastic Coupling,**
Gunnar Einarsson, Andreas Schütte, Axel Raichle, Britta Schöning, Wulf Mönich, Jens Neumann, Jürgen Arnold, Thomas Alrutz, Jörg Heinecke, Tomas Forkert, Holger Schumann.
NATO Research and Technology Organization, editor, *AVT-Symposium*, Volume AVT-123, 11, April 2005.

Das Verhalten von Flugzeugen bei transsonischer Zuströmung ist gekennzeichnet durch ein nichtlineares Strömungsfeld, welches von Stößen, Grenzschichteffekten wie Strömungsablösung und deren Interaktion mit der Hauptströmung verursacht wird. Diese instationären Vorgänge haben im Allgemeinen einen deutlichen Einfluss auf die Struktur des Flugzeugs (z.B. durch die Verbiegung der Flügeloberflächen), welches wiederum zu einer Änderung des Strömungsfeldes führt. Das ist einer der Gründe, warum es notwendig ist, für eine genaue Vorhersage der Eigenschaften einer Flugzeugkonstruktion die Strömungssimulation mit einer Struktursimulation zu koppeln.

Handelt es sich bei den zu untersuchenden Flugobjekten um Deltaflügelkonfigurationen, wird die oben angesprochene Problematik weiter verschärft. Diese Konfigurationen zeichnen sich dadurch aus, dass schon bei sehr kleinen Anstellwinkeln Wirbel durch Ablösungen an Rumpf und Flügel entstehen, die das aerodynamische Verhalten prägen. Kleinste Änderungen im Strömungsfeld können bei diesen Konfigurationen zu gravierenden Änderungen in der Lastverteilung über dem Flügel führen. Unter anderem ist das Wirbelaufplatzen einer der Gründe für diese Änderungen. Des Weiteren sind diese Konfigurationen aufgrund ihrer geringeren Masse erheblich anfälliger für plötzliche Änderungen in der Zuströmung. Moderne Kampfflugzeuge sind daher mit automatischen Flugreglern ausgestattet, um Störungen zu kompensieren und durch Korrekturen im

Triebwerksschub bzw. durch Veränderungen der Steuerklappen das Flugzeug selbst bei extremen Flugzuständen (z.B. dem Manöverflug) in der Luft zu halten.

Die Herausforderungen bei der Simulation dieser komplexen Vorgänge an die Komponenten der Software-Kette sind immens. Zum einen ist die Simulation instationärer aerodynamischer Phänomene wie zum Beispiel Böenwinde oder Wirbelaufplatzen bei hohen Anstellwinkeln sehr aufwändig²⁰ und zum anderen ist die Verifikation und Validierung der Ergebnisse anhand von geeigneten Experimenten recht schwierig.

Die beiden Artikel (H) und (I) geben einen Überblick der Zwischenresultate des DLR-Projekts SikMa [20], welches zum Ziel hatte eine Software-Kette zu entwickeln um die instationäre Aerodynamik eines frei fliegenden Kampfflugzeugs zu simulieren. Es werden in beiden Artikeln die verwendeten Werkzeuge zur Simulation von aerodynamischen, strukturmechanischen und flugmechanischen Eigenschaften beschrieben. Ferner wird auch die Integrationsumgebung dargestellt, die später die Anwendung der unterschiedlichen Werkzeuge in einer gemeinsamen Prozesskette ermöglichen soll. Eine Beschreibung der bisher durchgeführten Experimente zur Validierung der eingesetzten Werkzeuge erfolgt ebenfalls.

Die dargestellten Resultate in den beiden Artikeln sind recht ermutigend, zumal eine solche Kopplung (Strukturmechanik, Aerodynamik, Flugmechanik) zur Simulation komplexer Manöver bislang noch nicht durchgeführt worden ist.

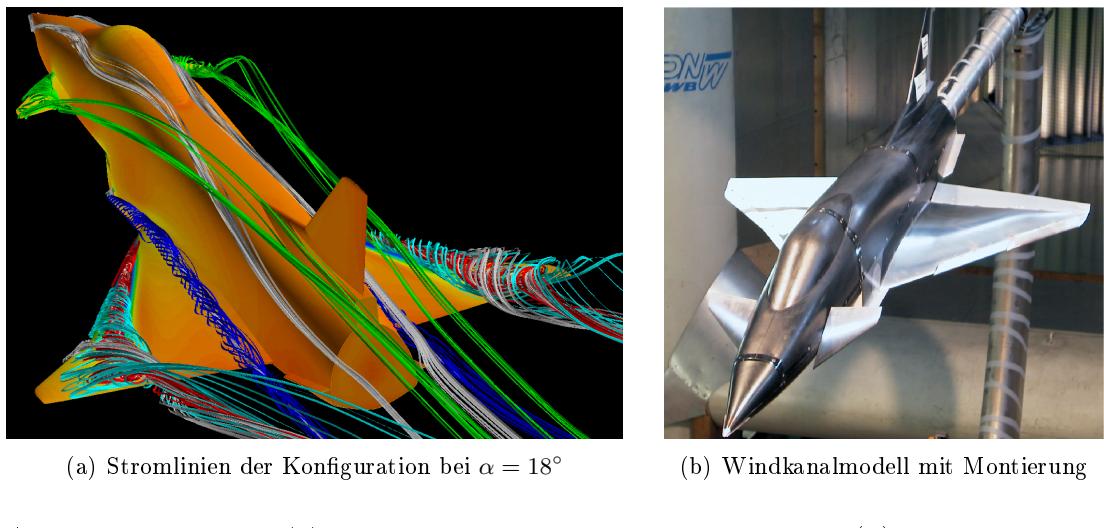


Abbildung 27: X-31: (a) zeigt die *clean wing* Konfiguration und (b) das Modell mit Steuerklappen im Windkanal.

Ein Resultat von Artikel (H) ist die gute Übereinstimmung zwischen Experiment und Rechnung bei einem geführten Nick-Manöver einer Deltaflügelkonfiguration (vgl. Abb. 2 in (H)). Ein weiteres Resultat der Arbeiten ist die grundsätzliche Übereinstimmung von Rechnung und Experiment bei der Vorhersage der Hauptströmungseigenschaften der X-31 Konfiguration (Abbildung 27 (a)) bei stationärer Anströmung. Für die *clean*

²⁰Die Simulation von instationären Strömungen muss zeitgenau durchgeführt werden. Hierfür ist unter Umständen eine sehr große Anzahl von Iterationsschritten erforderlich (vgl. [55] Kapitel II A.)

wing Konfiguration²¹ liefert die Simulation eine sehr gute Übereinstimmung mit dem Experiment (vgl. Abb. 7 und 8 in (H)). Bei der Vorhersage der Druckverteilung auf der Oberfläche weichen die Ergebnisse zu dem Experiment allerdings etwas ab (vgl. Abb. 16. in (I)). Dies liegt unter anderem daran, dass das Modell für das Windkanalexperiment mit beweglichen Steuerklappen ausgestattet war (siehe Abbildung 27 (b)) und für die Simulationsrechnung die *clean wing* Konfiguration verwendet wurde. Offensichtlich scheinen die kleinen Spalten am Frontflügel des X-31 Modells die Lage und die Saugeigenschaften der Wirbel zu beeinflussen.

Bei der Simulation eines freien Rollmanövers, welches ausgelöst wurde durch einen Klappenausschlag (Kopplung CFD und Flugmechanik), weichen die Ergebnisse zwischen Rechnung und Experiment deutlicher voneinander ab. Obwohl die Deltaflügelkonfiguration mit Klappen sowohl im Experiment als auch in der Rechnung in eine periodische Rollbewegung fällt, stimmt die Frequenz der Rollbewegung nicht überein. Als Erklärung für die Abweichung ist in (H) darauf hingewiesen worden, dass keine Berücksichtigung des mechanischen Widerstandes der Montierung des Modells bei der Rechnung stattfand. In (I) ist eine weitere Rechnung mit statischem mechanischen Widerstand durchgeführt worden. Die Dämpfung durch den mechanischen Widerstand ist hier so groß, dass die Amplitude bei der Rollbewegung geringer ist als im Experiment (vgl. Abb. 6 in (I)). Die Resultate der Kopplung von CFD und CSM²² sind zufriedenstel-

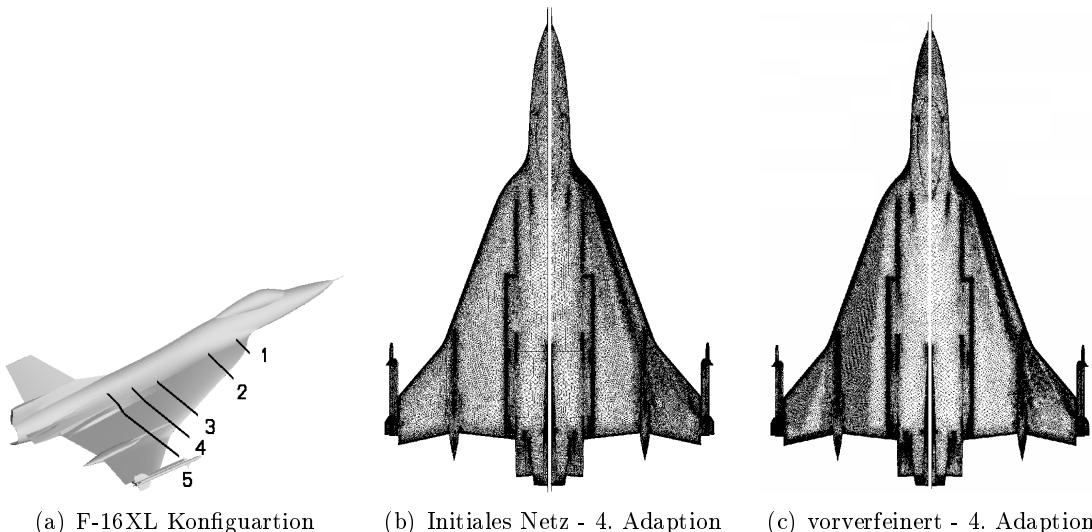


Abbildung 28: F-16XL CAWAPI Konfiguration (a) und ein Vergleich der Oberflächenauflösung der von Fritz in [25] verwendeten Netze (b) und (c).

lend. Die charakteristische Bewegung der Montierung (siehe auch Abb. 27 (b)) wird von der gekoppelten Simulation korrekt wiedergegeben (vgl. Abb. 10 in (I)).

In beiden Artikeln ist die grundsätzliche Fähigkeit zum Aufbau einer Prozesskette zur Simulation komplexer Manöver aufgezeigt worden. Die erzielten Ergebnisse sind unter

²¹Die *clean wing* Konfiguration hat keine beweglichen Steuerklappen (vgl. Abb. 27 (a)).

²²Computational Structure Mechanics - Strukturmechaniksimulation

Berücksichtigung der enormen Komplexität der instationären Vorgänge bei Deltaflügelkonfigurationen im Manöverflug als sehr gut zu bezeichnen.

Der Beitrag des Verfassers dieser Arbeit zu den beiden Publikationen besteht in der Vorbereitung der voll parallelen Simulationskette (Partitionierung und y^+ -Adaption) sowie in der Entwicklung spezieller Sensoren für die Wirbelerkennung in der Netzadaption (vgl. Kapitel 8).

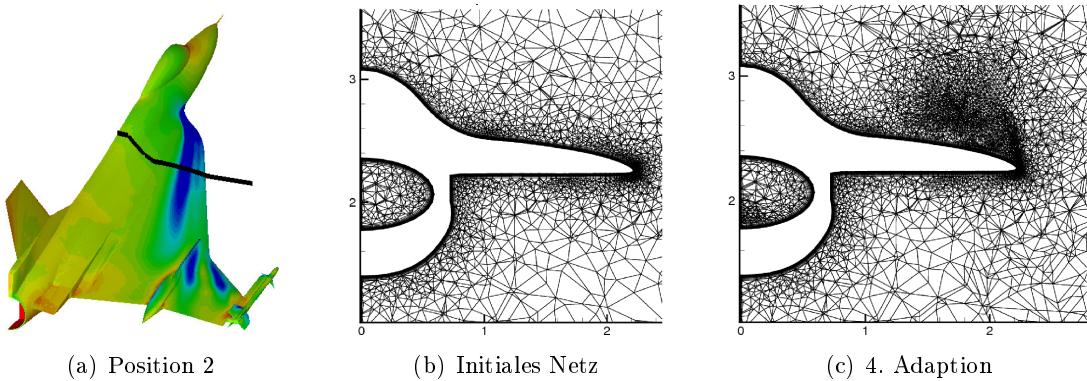


Abbildung 29: Netzschnitte an Position 2 bei FC 25 (siehe Tabelle 7).

Um zu verdeutlichen, dass die Wirbelerkennung bei der Netzverfeinerung für die oben betrachteten Flugzeugkonfigurationen einen nicht zu unterschätzenden Vorteil für die Genauigkeit der numerischen Simulation bringt, wird die Arbeit von Fritz [25] herangezogen. Die von Fritz untersuchten Flugzustände der F16-XL Konfiguration (siehe Abbildung 28²³ (a)) sind in dem *Cranked Arrow Wing Aerodynamics Project International (CAWAPI)* von der RTO²⁴ Arbeitsgruppe AVT-113²⁵ für einen Vergleichstest ausgewählt worden (vgl. [78]). An diesem Vergleichstest haben sich unterschiedliche

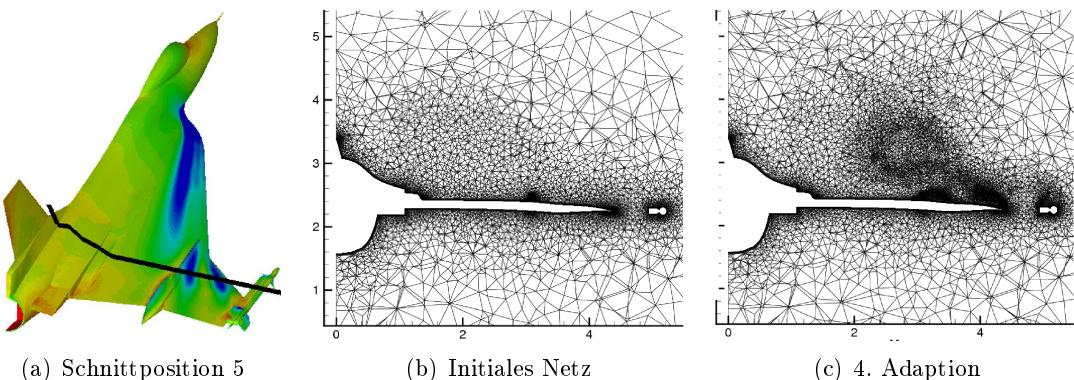


Abbildung 30: Netzschnitte an Position 5 bei FC 25 (siehe Tabelle 7).

tional (CAWAPI) von der RTO²⁴ Arbeitsgruppe AVT-113²⁵ für einen Vergleichstest ausgewählt worden (vgl. [78]). An diesem Vergleichstest haben sich unterschiedliche

²³Reproduziert von [25] mit Erlaubnis des Autors.

²⁴NATO Research and Technology Organization - Wissenschaftlicher Zweig der NATO.

²⁵Applied Vehicle Technology - Eine von sieben Gruppen innerhalb der RTO.

Forschungsinstitutionen und Flugzeugproduzenten mit ihren jeweiligen CFD-Codes beteiligt und die Ergebnisse ihrer Simulationen mit den Daten der Flugexperimente der *Cranked Arrow Wing Aerodynamics Project (CAWAP)* Datenbank verglichen [49].

Flugzustand	Machzahl	Anstellwinkel α	Reynoldszahl	Netzauflösung
FC 7	0.304	11.89	44.40×10^6	20.5×10^6
FC 9	0.360	11.85	44.80×10^6	20.5×10^6
FC 25	0.242	19.84	32.22×10^6	21.1×10^6
FC 70	0.970	4.37	88.77×10^6	18.5×10^6

Tabelle 7: Übersicht über die untersuchten Flugzustände und verwendeten Netzgrößen.

Anstatt auf die bereits vorhandenen Netze bzw. auf von Hand vorverfeinerte Netze zurückzugreifen wie alle anderen Teilnehmer des CAWAPI Benchmarks [7], hat sich Fritz dazu entschieden, die TAU-Netzadaption so einzusetzen wie in (C) beschrieben. Der einzige Unterschied zu (C) bestand darin, dass Fritz neben dem Totaldruck und dem Betrag des Geschwindigkeitsvektors noch die Helizität [52] und die Dichte als weitere Variablen zur Erkennung der Wirbel in der Netzadaption eingesetzt hat. Fritz hat insgesamt drei verschiedene Netze erzeugt und miteinander verglichen. Das initiale Netz bestand aus ca. 10.5×10^6 Netzpunkten und wurde mit der Netzadaption vier mal verfeinert bis eine entsprechende Auflösung je nach Flugzustand (vgl. Tabelle 7) erreicht wurde (siehe Abbildung 28²³ (b) sowie 29²³ und 30²³). Des Weiteren hat Fritz zur Verifikation der Ergebnisse auf den anderen Netzen noch ein manuell vorverfeinertes Netz erzeugt mit ca. 22.0×10^6 Netzpunkten (siehe Abbildung 28²³ (c)).

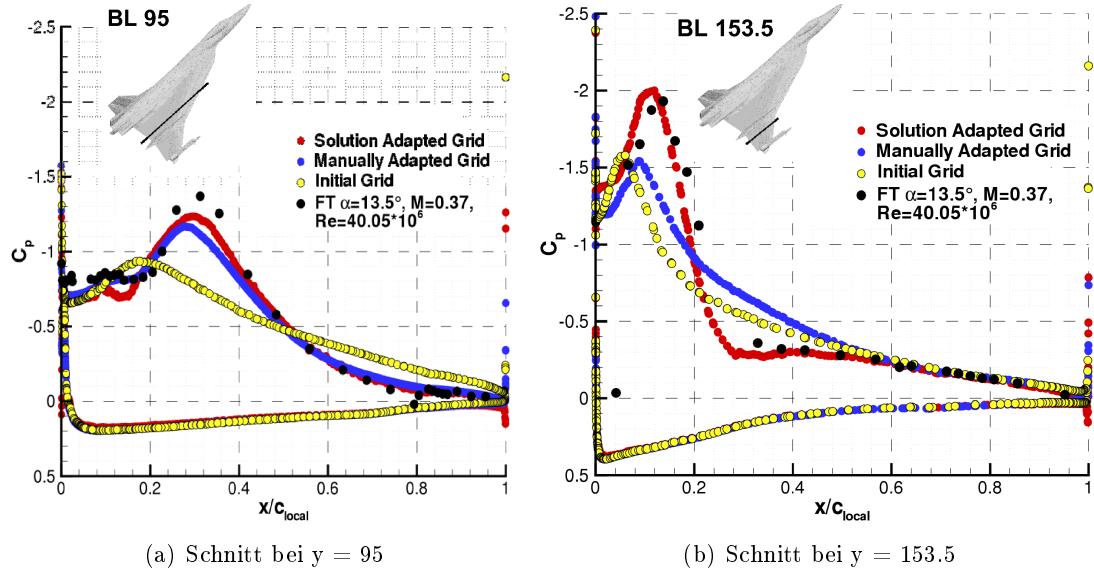


Abbildung 31: Vergleich zwischen Flugexperiment und Simulationsrechnung auf unterschiedlichen Netzen (FC 7: $\alpha = 11.89^\circ$, $Mach = 0.304$, $Re = 44.4 \times 10^6$).

In Abbildung 31²³ und 32²³ ist klar zu erkennen, dass durch die sukzessive Anwen-

dung der parallelen Netzadaption zum Teil erheblich bessere Ergebnisse erzielt werden können, als mit einem manuell vorverfeinerten Netz bzw. mit dem initialen Netz.

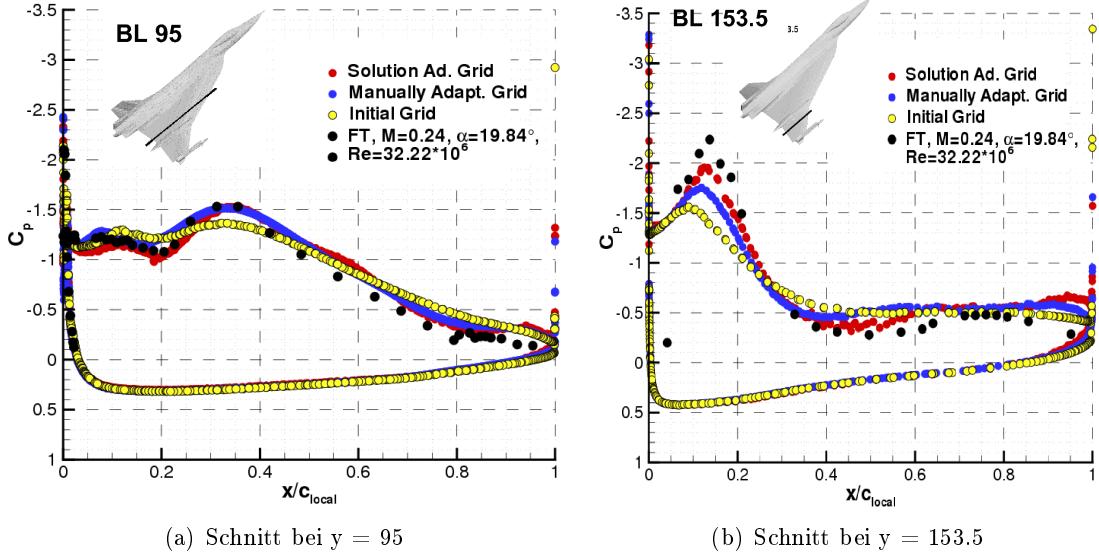


Abbildung 32: Vergleich zwischen Flugexperiment und Simulationsrechnung auf unterschiedlichen Netzen (FC 25: $\alpha = 19.84^\circ$, $Mach = 0.242$, $Re = 32.22 \times 10^6$).

Fritz konnte sich durch die Verwendung der lösungsbasierten TAU-Netzadaption bei dem CAWAPI Benchmark deutlich von einigen anderen Teilnehmern absetzen [78]. Die Arbeit von Fritz stellt damit eindrucksvoll unter Beweis, dass die automatische Netzadaption mit speziellen Wirbelerkennungsalgorithmen absolut in der Lage ist die komplexen Strömungstopologien aufzulösen, welche bei der Umströmung von Deltaflügelkonfigurationen entstehen. Aus diesem Grund stellt die parallele dynamische Netzadaption mit anschließender Repartitionierung ein Schlüsselwerkzeug innerhalb der Software-Kette zur Simulation instationärer Aerodynamik bei komplexen Konfigurationen dar.

11 Ausblick

Die Weiterführung der hier vorgestellten Arbeiten wird im Rahmen von Projekten (z.B. [24]) und Diplom- bzw. Masterarbeiten erfolgen bzw. ist bereits erfolgt.

Eine weitere konkrete Anwendung der in Kapitel 7 vorgestellten Algorithmen zur Stapelerhaltung von Hexaedern bzw. Prismen ist die Diplomarbeit von Reuss [76]. Hierbei handelt es sich um eine Untersuchung, ob durch eine anisotrope Agglomeration im Preprocessing des TAU-Codes die Stabilität des Strömungslösers im parallelen Einsatz gesteigert bzw. die Konvergenzordnung verbessert werden kann. Die Algorithmen zur Stapelerhaltung bzw. Linienextraktion sind von Reuss dazu benutzt worden, eine kontrollierte anisotrope Agglomeration für die Grobgittererzeugung im Preprocessing des TAU-Codes zu entwickeln. Durch die Anwendung dieser anisotropen Agglomeration

konnte Reuss in ihrer Arbeit bei einigen Beispielen eine Verbesserung zu der bisher verwendeten Agglomeration dokumentieren.

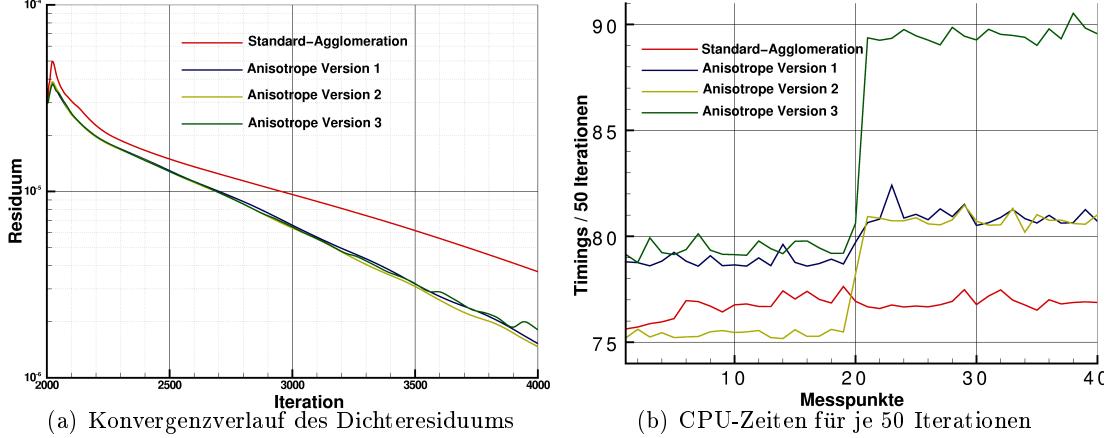


Abbildung 33: Vergleich der Standard-Aggomeration mit drei unterschiedlichen Varianten der anisotropen Agglomeration bei dem DLR F6 Fall mit 4w Multigitterzyklus auf 32 CPUs (gerechnet mit Runge-Kutta auf [21] - vgl. auch Tab. 2 auf Seite 37).

In keinem der von Reuss getesteten Fälle haben die Modifikationen an der Agglomeration zu einer Verlangsamung des Strömungslösers oder zu einer Verschlechterung der Konvergenzordnung geführt (vgl. [76] Kapitel 5). In einem Fall konnte sogar eine Steigerung der Konvergenzgeschwindigkeit bei gleichbleibender Rechenzeit erzielt werden (siehe Abbildung 33 und vgl. [76] Kapitel 5.2.2) und in einem anderen Fall konnte bei gleichbleibender Konvergenzordnung eine Reduzierung der benötigten Rechenzeit beobachtet werden (siehe Abbildung 34 und vgl. [76] Kapitel 5.4.2).

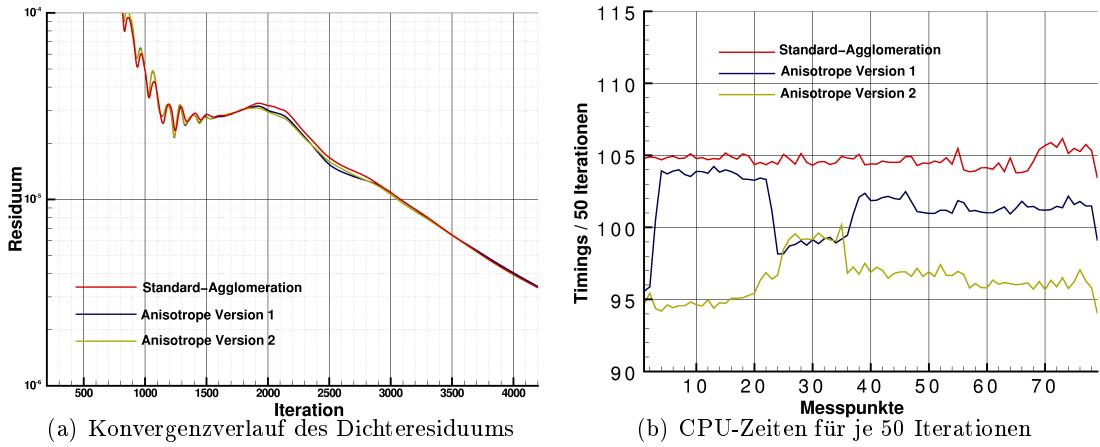


Abbildung 34: Vergleich der Standard-Aggomeration mit drei unterschiedlichen Varianten der anisotropen Agglomeration bei einer Reiseflugkonfiguration mit 3v Multigitterzyklus auf 64 CPUs (gerechnet mit LU-SGS [16] auf [21]).

Für die korrekte Anwendung der anisotropen Agglomeration im parallelen Preprocessing ist die in Kapitel 7 beschriebene Modifikation zur Stapelerhaltung im primären Netzpartitionierer ebenso notwendig wie für die parallele y^+ -Adaption.

Architektur	AMD Opteron			Intel Xeon	IBM POWER
GHz	2.2 DC	2.6 DC	2.3 QC	3.2 DC	1.9 P5+ / 4.7 P6
MPI Tasks/Knoten	4	4	8	4	8
Release 2006-p4	8.67s	8.20s	4.35s	11.00s	5.75s / 2.32s
Optimiert 2006-p4	7.28s	7.15s	3.68s	8.54s	4.85s / 2.01s
Performancegewinn	19%	15%	18%	29%	18% / 15%

Tabelle 8: Zeitbedarf auf unterschiedlichen Architekturen für eine Iteration 4w Multigitterzyklus mit dem DLR-F6 Fall und zwei unterschiedlichen TAU-Code Versionen. (DC - Dualcore, QC - Quadcore, P5+ und P6 - Dualcore mit SMT²⁷)

Ein anderes Projekt hat gezeigt, dass die parallele Effizienz des TAU-Codes noch weiter gesteigert werden kann (vgl. (O) bzw. (Q)). Die in dem Kooperationsprojekt von IBM Deutschland GmbH, T-Systems Solution for Research GmbH und dem DLR-Institut für Aerodynamik und Strömungstechnik durchgeführten Modifikationen am TAU-Strömungslöser konnten die Software um ca. 15% - 29% beschleunigen bezogen auf die Performance eines einzelnen Rechenknotens (siehe Tabelle 8).

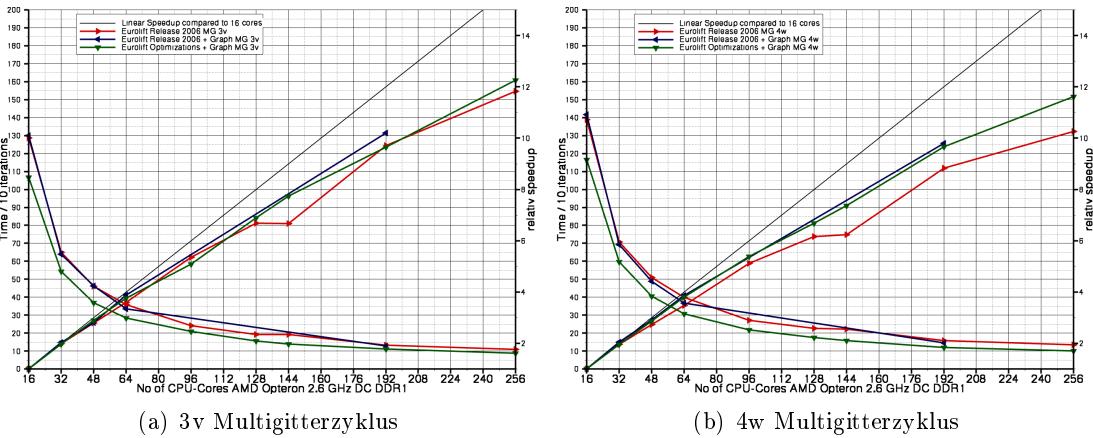


Abbildung 35: Vergleich der parallelen Performance unterschiedlicher TAU-Code Versionen mit der Hochauftreibskonfiguration (siehe Abb. 3 auf Seite 5) gerechnet auf [42].

Ein fundamentales Ergebnis dieses Kooperationsprojekts war jedoch, dass die Lastverteilung, welche der Netzpartitionierer durch einen rekursiven Bisektions-Algorithmus erzeugt, die parallele Effizienz erheblich beschränkt (vgl. Abbildung 35 rote Linie). Insbesondere tritt dieser Effekt hervor wenn deutlich mehr als 256 CPUs eingesetzt werden sollen (vgl. Abbildung 36 rote Linie).

²⁷Simultaneous Multithreading - verdoppelt die verfügbare Anzahl an logischen CPU-Cores.

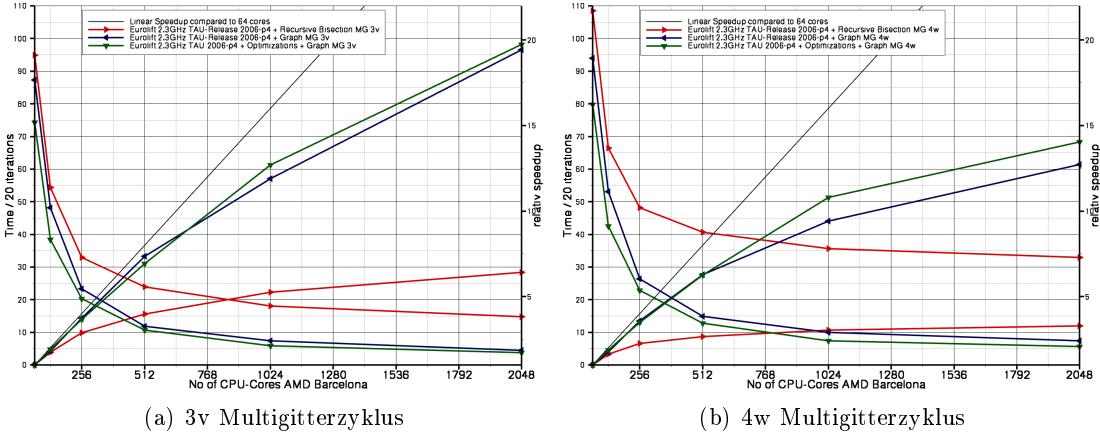


Abbildung 36: Vergleich der parallelen Performance unterschiedlicher TAU-Code Versionen mit der Hochauflösenkonfiguration (siehe Abb. 3 auf Seite 5) gerechnet auf [88].

Im Gegensatz zu dem im TAU-Code verwendeten rekursiven Bisektions-Algorithmus ist in dem Optimierungsprojekt der sequentielle Graph-Partitionierer Chaco [34] eingesetzt worden, um die Lastverteilung zu verbessern und die Anzahl der Kommunikationsnachbarn zu reduzieren. Anhand der Abbildungen 35 und 36 ist klar zu erkennen, dass der Graph-Algorithmus (blaue Linie) dem derzeitigen rekursiven Bisektions-Algorithmus (rote Linie) bei mehr als 64 CPUs deutlich überlegen ist. Daraus folgt, dass bei größeren CPU-Anzahlen ein Graph-Partitionierer eine bessere parallele Effizienz des Strömungslösers ermöglicht (siehe auch Tabelle 9).

CPUs	1	4	16	32	64	128	192
Release 2006-p4	24.52s	8.20s	2.10s	1.23s	0.73s	0.49s	0.48s
Optimiert 2006-p4 + Chaco	21.10s	7.15s	1.79s	0.97s	0.54s	0.32s	0.25s
Performancegewinn	16%	15%	17%	17%	35%	53%	92%

Tabelle 9: Rechenzeit für eine Iteration 4w Multigitterzyklus mit dem DLR-F6 Fall (vgl. auch Abb. 4 auf Seite 6) und zwei unterschiedlichen TAU-Code Versionen auf [42].

Diese Untersuchungen haben dazu geführt, dass eine Masterarbeit in der Parallelisierung des Graphpartitionierungs-Algorithmus nach Kernighan-Lin (Multi-Level KL [43]) initiiert wurde, um die verbesserten Skalierungsfähigkeiten auch im parallelen Netzpartitionierer des TAU-Codes zu nutzen.

Sowohl die Diplomarbeit von Reuss als auch die Masterarbeit in der Parallelisierung des Multi-Level KL Algorithmus tragen zur Effizienzsteigerung der Simulationskette im TAU-Code aktiv bei. Beide Arbeiten sind bzw. werden vom Verfasser dieser Arbeit betreut und basieren auf den in dieser Arbeit vorgestellten Datenstrukturen und Algorithmen bzw. auf den durchgeführten Untersuchungen.

12 Literaturverzeichnis

- [1] FOI Swedish Defence Research Agency. Edge 4.1, 2008. <http://www.foi.se/edge>.
- [2] Petra Aumann, H. Barnewitz, H. Schwarten, Klaus Becker, Ralf Heinrich, Britta Roll, M. Galle, Norbert Kroll, Thomas Gerhold, Dieter Schwamborn, and M. Franke. MEGAFLOW: Parallel complete aircraft cfd. *Parallel Computing*, 27(4):415–440, 2001.
- [3] T. J. Barth and D. C. Jespersen. The design and application of upwind schemes on unstructured meshes. paper 89-0366, AIAA, 1989.
- [4] K. Becker, N. Kroll, C.-C. Rossow, and F. Thiele. Numerical Flow Calculation for Complete Aircraft - The MEGAFLOW Project. In *DGLR-Jahrestagung, Bremen (de), Oktober 1998*, 1998.
- [5] Klaus Becker and Norbert Kroll. Numerical Simulation of Aerospace Aerodynamics - Success and Expectation, June 2007. International Supercomputing Conference ISC07, Dresden, Germany.
- [6] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.
- [7] O. Boelegs, S. Götz, S. Morton, W. Fritz, and J. Lamar. Description of the F-16XL geometry and computational grids used in CAWAPI. Paper 488, 45th AIAA Aerospace Sciences Meeting and Exhibit, Janurary 8-11 2007. Reno, Nevada.
- [8] M. O. Bristeu, O. Pirnneau, R. Glowinski, J. Periaux, P. Perrier, and G. Poirier. On the numerical solution of non-linear problems in fluid dynamics by least squares and finite-element methods (II), applications to transonic flow simulations. *Computational Methods in Applied Mechanical Engineering*, 51:363–394, 1985.
- [9] P. A. Cavallo and M. J. Grismer. Further Extension and Validation Of A Parallel Unstructured Mesh Adaptation Package. paper AIAA 2005-0924, AIAA, 2005.
- [10] Eric Chaput. Design challenges in the Airbus A380 and A350 projects. In P. Wesselink, E. Onate, and J. Periaux, editors, *Proceedings, ECCOMAS CFD 2006 CONFERENCE*, September 5-8 2006.
- [11] S.-H. Chiang and C. Fu. Re-evaluating reservation policies for backfill scheduling on parallel systems. In *Proceedings of Parallel and Distributed Computing and Systems - 2004*. MIT Cambridge, USA, November 2004.
- [12] U. Dallmann. Topological Structures of Three-Dimensional Flow Separation. IB 221-82 A 07, DFVLR, 1983.
- [13] 2nd AIAA CFD Drag Prediction Workshop, 2003. <http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/Workshop2/workshop2.html>.
- [14] 3rd AIAA CFD Drag Prediction Workshop, 2006. <http://aaac.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw>.

- [15] Richard Dwight. Time-accurate navier-stokes calculations with approximately factored implicit schemes. In C. Groth and D.W. Zingg, editors, *ICCFD3*. Springer, 2004.
- [16] Richard Dwight. *Efficiency Improvements of RANS-Based Analysis and Optimization using Implicit and Adjoint Methods on Unstructured Grids*. PhD thesis, University of Manchester, 2006.
- [17] Richard Dwight. Adjoint algorithms for the optimization of 3d turbulent configurations. In *15. DGLR-Fach-Symposium der STAB*, volume 96. Springer, 2007.
- [18] JR Edwards and S. Chandra. Comparison of Eddy Viscosity – Transport Turbulence Models for Three-Dimensional, Shock-Separated Flowfields. *AIAA Journal*, 39(4), April 1996.
- [19] DLR Institut für Aerodynamik und Strömungstechnik. MegaCads - Multiblock-Elliptic-Grid-Generation-And-Computer-Aided-Design-System, 1999. <http://www.megacads.dlr.de/>.
- [20] DLR Institut für Aerodynamik und Strömungstechnik. Projekt SikMa - Simulation komplexer Manöver, 2006. http://www.dlr.de/as/en/desktopdefault.aspx/tabcid-195/274_read-470/.
- [21] DLR Institut für Aerodynamik und Strömungstechnik. Rechnen-Cluster Einweihung im DLR Göttingen; *72 Rechenknoten mit je 2.2 GHz Dualcore AMD Opteron*, 2006. http://www.dlr.de/desktopdefault.aspx/tabcid-2666/4016_read-5996/.
- [22] DLR Institut für Aerodynamik und Strömungstechnik. $C^2 A^2 S^2 E$ - Flugzeuge aus dem Computer, Mai, 2007. http://www.dlr.de/as/desktopdefault.aspx/tabcid-127/1082_read-9237/.
- [23] DLR Institut für Aerodynamik und Strömungstechnik. $C^2 A^2 S^2 E$ - Hochleistungsrechner für die Luftfahrtforschung beim DLR in Betrieb genommen; *768 Rechenknoten mit je 1.9 GHz Quadcore AMD Opteron*, Mai, 2008. http://www.dlr.de/as/desktopdefault.aspx/tabcid-127/1082_read-12509/.
- [24] Bundesministerium für Bildung und Forschung (BMBF). Auschreibung: HPC-Software für skalierbare Parallelrechner, 2007. <http://www.bmbf.de/foerderungen/11830.php>.
- [25] W. Fritz. RANS Solutions for the CAWAPI F-16XL in Solution Adapted Hybrid Grids. Paper 492, 45th AIAA Aerospace Sciences Meeting and Exhibit, Janurary 8-11 2007. Reno, Nevada.
- [26] M. Galle, T. Gerhold, and J. Evans. Parallel Computation of Turbulent Flows around Complex Geometries on Hybrid Grids with the DLR-Tau Code. In *11th Parallel CFD Conference, Williamsburg (va), 23.-26.05.1999*, 1999.
- [27] Anthony Donald Gardner, Kai Richter, and Henning Rosemann. Simulation of Oscillating Airfoils and Moving Flaps Employing the DLR-TAU Unsteady Grid Adaptation. In *15. DGLR-Fach-Symposium der STAB*, 2006.

-
- [28] Alfred Geiger and Rolf Page. Compute-Power for $C^2A^2S^2E$, November 2007. Sun HPC Consortium.
- [29] T. Gerhold and J. Evans. Efficient Computation of 3D-Flows for Complex Configurations with the DLR-TAU Code. In W. Nitsche H.-J. Heinemann R. Hilbig, editor, *AG STAB Symposium, Berlin, 10.-12. November 1998*, volume 72, pages 178 – 185. Vieweg, 1999.
- [30] T. Gerhold, J. Evans, and M. Galle. Technical Documentation of the DLR TAU-Code. IB 223-97 A 43, DLR, 1997.
- [31] T. Gerhold, V. Hannemann, and D. Schwamborn. On the Validation of the DLR-TAU Code. In *AG STAB Symposium, Berlin, 10.-12. November 1998*, volume 72, pages 426 – 433. Vieweg, 1999.
- [32] Thomas Gerhold and Jens Neumann. The parallel mesh deformation of the DLR TAU-Code. In *15. DGLR-Fach-Symposium der STAB*, 11 2006.
- [33] S. Götz, B. Eisfeld, T. Alrutz, and A. Ronzheimer. Evaluation of TAU Flow-Solver Capabilities in the Aero-data-for-Loads Context - Part I: Megaflug. IB 124-2007/911, DLR, 2007.
- [34] Bruce Hendrickson and Robert Leland. Chaco: Software for Partitioning Graphs. <http://www.sandia.gov/bahendr/chaco.html>.
- [35] A. Jameson. Time-dependent calculations using multi-grid, with applications to unsteady flows past airfoils and wings. Paper 1596, AIAA, 1991.
- [36] A. Jameson, T. J. Baker, and N.P. Weatherhill. Calculation of inviscid transonic flow over a complete aircraft. paper 86-0103, AIAA, 1986.
- [37] A. Jameson and D. J. Mavriplis. Finite volumen solution of the two-dimensional Euler equation on a regular triangular mesh. *AIAA Journal*, 24(4):611–618, 1986.
- [38] J. Jeong and F. Hussain. On the identification of a vortex. *Journal Fluid Mech.*, 285:69–94, 1995.
- [39] Jülich Supercomputing Center (JSC). JUGENE - 16384 Rechenknoten mit je 4 PowerPC 450 Cores a 850 MHz, 2008. <http://www.fz-juelich.de/jsc/service/sco/bmBGP>.
- [40] S. Karl, K. Hannemann, J. Steelant, and A. Mack. CFD Analysis of the HyS-hot Supersonic Combustion Flight Experiment Configuration. Paper 8041, 14th AIAA/AHI Space Planes and Hypersonic Systems and Technologies Conference, Canberra, Nov. 2006.
- [41] Sebastian Karl. Numerical Investigation of Transient Flow Phenomena in Dual-Bell Nozzles. In *Proceedings of the 6th International Symposium on Launcher Technologies*, Munich, Nov. 2005.

- [42] Scientific Supercomputing Center (SSC) Karlsruhe. HP XC4000, 750 Rechenknoten mit je 2 AMD Opteron Dual Core 2.6GHz CPUs, 2006. <http://www.rz.uni-karlsruhe.de/ssck/hpxc4000.php>.
- [43] B. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *Bell Systems Technical Journal*, 49:291–308, 1970.
- [44] J. Klenner, K. Becker, M. Cross, and N. Kroll. Future Simulation Concept, September 10.-13. 2007. CEAS-2007-105, 1. CEAS European Air and Space Conference, Berlin, Germany.
- [45] Tobias Knopp. Validation of the Turbulence Models in the DLR TAU Code for Transonic Flows - A Best Practice Guide, 2006.
- [46] N. Kroll and J. K. Fassbender, editors. *MEGAFLOW — Numerical Flow Simulation for Aircraft Design Results of the second phase of the German CFD initiative MEGAFLOW presented during its closing symposium at DLR, Braunschweig, Germany, December 10th and 11th 2002*, volume 89 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*. Springer Verlag, 2005.
- [47] Norbert Kroll and Laurent Cambier. MIRACLE - A Joint DLR/ONERA Effort on Harmonization and Development of Industrial and Research Aerodynamic Computational Environment. In *7th ONERA-DLR Aerospace Symposium, ODAS 2006*, Toulouse, France, 10 2006.
- [48] Norbert Kroll, Thomas Gerhold, Stefan Melber, Ralf Heinrich, Thorsten Schwarz, and Britta Schöning. Parallel Large Scale Computations for Aerodynamic Aircraft Design with the German CFD System MEGAFLOW. In *Proceedings of Parallel CFD 2001*, 2001. Egmond aan Zee, The Netherlands, May 21-23.
- [49] J. Lamar and C. Obara. Review of Cranked-Arrow Wing Aerodynamics Project: Its international aeronautical community role. Paper 487, 45th AIAA Aerospace Sciences Meeting and Exhibit, January 8-11 2007. Reno, Nevada.
- [50] Heinrich Lüdeke. Unsteady investigation of the dlr tic nozzle by different turbulence models. In ESA, editor, *ATAC-FSCD WORKSHOP: « AFTER-BODY AND NOZZLE FLOWS »*, 06 2007.
- [51] Claude Lepage, Amik St-Cyr, and Wagdi Habashi. Parallel Ustructured Mesh Adaptation on Distributed-Memory Systems. paper 2004-2532, AIAA, 2004.
- [52] Y. Levy, D. Degani, and A. Seginer. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28(8):1347–1352, August 1990.
- [53] R. Löhner, C. Yang, J.R. Cebral, F.F.Camelli, F. Togashi, J.D. Baum, H. Luo, E.L. Mestreau, and O.A. Soto. Moore's Law, the Life Cycle of Scientific Computing Codes and the Diminishing Importance of Parallel Computing. Talk, May 2005. Parallel CFD05, College Park, MD, USA.

-
- [54] A. Madrane, A. Raichle, and A. Stürmer. Parallel Implementation of a Dynamic Overset Unstructured Grid Approach. In ECCOMAS, editor, *ECCOMAS 2004, Jyväskylä (fi), 24.-28.07.2004*, 2004.
- [55] D. Mavriplis, D. Darmofal, D. Keyes, and M. Turner. Petaflops Opportunities for the NASA Fundamental Aeronautics Program. paper AIAA 2007-4084, AIAA, Miami (FL), USA, 06 2007.
- [56] D. J. Mavriplis. Accurate multigrid solution of the Euler equations on unstructured and adaptive meshes. *AIAA Journal*, 28(2):213–221, 1990.
- [57] D. J. Mavriplis. Three-dimensional multigrid for the Euler equations. *AIAA Journal*, 30(7):1753–1761, 1992.
- [58] Dimitri J. Mavriplis. Unstructured Mesh discretizations and Solvers for Computational Aerodynamics. paper AIAA 2007-3955, AIAA, Miami (FL), USA, 06 2007.
- [59] Dimitri J. Mavriplis, Michael J. Aftosmis, and Marsha Berger. High Resolution Aerospace Applications using the NASA Columbia Supercomputer. paper, Super Computing Conference, Seattle WA, 11 2005.
- [60] D. J. Mavriplis. Unstructured Mesh Related Issues in Computational Fluid Dynamics (CFD) Based Analysis and Design. In *11th International Meshing Roundtable*, page 143, 2002.
- [61] D.J. Mavriplis. A three-dimensional mulitgrid reynolds-averaged navier-stokes solver for unstructured meshes. *AIAA*, 33(3):445–453, 1995.
- [62] D.J. Mavriplis. Adaptive Meshing Techniques for Viscous Flow Calculation on Mixed-Element Unstructured Meshes. paper 97-0857, AIAA, 1997.
- [63] S. Melber-Wilkending. 3D RANS-Simulationen für die Analyse der Wirbelschleppe eines Transportflugzeuges in Hochauftriebskonfiguration. STAB-Jahresbericht, November 2003. STAB Workshop, 4./5. November, Göttingen, Germany, 2003.
- [64] S. Melber-Wilkending. Aerodynamic Analysis of Jet-Blast using CFD considering as example a Hangar and an AIRBUS A380 configuration. In *14th AG STAB/DGLR Symposium, Bremen (de), 16.-18.11.2004*, 2004.
- [65] Sun Mircrosystems. N1 Grid Engine 6 Administration Guide, 2005.
- [66] Sun Mircrosystems. N1 Grid Engine 6, 2008. <http://gridengine.sunsource.net/documentation.html>.
- [67] S. A. Morton, J. R. Forsythe, A. M. Mitchell, and D. Hajek. Detached-Eddy Simulations and Reynolds-Averaged Navier-Stokes Simulations of Delta Wing Vortical Flowfields. *Journal of Fluids Engineering*, 124(4):924–932, December 2002.
- [68] Message Passing Interface (MPI), 2007. <http://www.mcs.anl.gov/mpi>.

- [69] NASA. TetrUSS - Tetrahedral Unstructured Software System, 2008. <http://tetruss.larc.nasa.gov/components.html>.
- [70] P. Niederdrenk. Solution - adaptive grid generation by hyperbolic/parabolized hyperbolic p.d.e.s. *3rd International Conference on Numerical Grid Generation-Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, pages 173 – 184, 1992.
- [71] P. Niederdrenk. Grid Adaptation to multiple auto-scaled solution features. In Pi-neridge Press, editor, *4th International Conference on Numerical Grid Generation in CFD and Related Fields, Swansea/Great Britain, 06.-08. April 1994*, pages 501 – 512, 1994.
- [72] Peter Niederdrenk. On the Control of Elliptic Grid Generation. *Numerical Grid Generation in Computational Fluid Simulation. 6th International Conference, July 1998*, Vol. 6:257 – 266, 1998.
- [73] S.Z. Pirzadeh. Vortical flow prediction using an adaptive unstructured grid method. paper RTO-MP-069(1), NATO Research and Technology Organization AVT-Symposium, Loen, Norway, 7-11 May 2001.
- [74] Axel Raichle, Stefan Melber-Wilkending, and Jan Himisch. A new Actuator Disk Model for the TAU Code and application to a sailplane with a folding engine. In *15. DGLR-Fach-Symposium der STAB*, 11 2006.
- [75] CRAFT Tech Combustion Research and Flow Technology Inc. CRISP CFD® Code Overview, 2008. <http://www.craft-tech.com/html/crisp.html>.
- [76] Sivia Reuss. Anisotrope Agglomeration in DLR TAU-Code. Diplomarbeit, Universität Göttingen, 2008.
- [77] Kai Richter and Henning Rosemann. Numerical Investigation on the Aerodynamic Effect of Mini-TEDs on the AWIATOR Aircraft at Cruise Conditions. In *25th Congress on International Council of the Aeronautical Sciences (ICAS 2006)*, 09 2006.
- [78] A. Rizzi, O. Boelens, A. Jirásek, and K. Badcock. What Was Learned from Numerical Simulation of F-16XL (CAWAPI) at Flight Conditions. Paper 683, 45th AIAA Aerospace Sciences Meeting and Exhibit, Janurary 8-11 2007. Reno, Nevada.
- [79] M. Rütten. Topologische Untersuchung des Wirbelplatzens zur Identifikation von Wirbelplatzparameter. Dissertation, Helmut-Schmidt-University, Hamburg, 2004.
- [80] M. Rütten. Vortex Axis Calculation by Using Vortex Features. paper AIAA 2004-2353, AIAA, 2004.
- [81] M. D. Salas. The last CFD aeronautical grand challenges. *Journal of Scientific Computing*, 28(2-3):479–505, September 2006.
- [82] Moab cluster suite, 2008. <http://www.clusterresources.com/pages/products/moab-cluster-suite.php>.

-
- [83] Dieter Schwamborn, Thomas Gerhold, and Ralf Heinrich. The DLR TAU-Code: Recent Applications int Research and Industry. In P. Wesseling, E. Onate, and J. Periaux, editors, *Proceedings, ECCOMAS CFD 2006 CONFERENCE*, September 5-8 2006.
- [84] Dieter Schwamborn, Thomas Gerhold, and Roland Kessler. The DLR-TAU Code - an Overview. In *Proceedings ODAS 99*, 1999. ONERA-DLR Aerospace Symposium 21-24 June in Paris, France.
- [85] Cobalt Solutions. Cobalt, 2008. <http://www.cobaltcfd.com/index.php/site/software/cobalt/>.
- [86] T. Sonar and Süli E. A dual graph norm refinement indicator for the DLR- TAU -Code. DLR-FB 94-24, 1994.
- [87] P.R. Spalart and S.R. Allmaras. A One equation Turbulence Model for Aerodynamic Flows. paper 1992-439, AIAA, Reno, Nevada, USA, 1992.
- [88] Texas Adavanced Computing Center (TACC). Ranger - 3936 Rechenknoten mit je 4 AMD Opteron Quad Core 2.3GHz CPUs, 2008. <http://www.tacc.utexas.edu/resources/hpcsystems/>.
- [89] Top 500 - architecture share for 11/1997, 1997. <http://www.top500.org/stats/list/10/archtype>.
- [90] Top 500 - architecture share for 11/2007, 2007. <http://www.top500.org/stats/list/30/archtype>.
- [91] Top 500, 2007. <http://www.top500.org/lists/2007/11>.
- [92] U. Tremel, S. Hitzel, K. Sorensen, and N. Weatherill. JDAM-Store Separation from an F/A-18C - An Application of the Multidisciplinary SimServer-System. paper AIAA 2005-5222, AIAA, 2005.
- [93] Udo Tremel, Frank Deister, Oubay Hassan, and Nigel P. Weatherill. Automatic unstructured surface mesh generation for complex configurations. *International Journal of Numerical Methods in Fluids*, 45(4), 2004.
- [94] C. Truesdell. The Kinematics of Vorticity. Technical report, Indiana University, 1953.
- [95] Stefan Wallin. *Engineering turbulence modelling for CFD with a focus on explicit algebraic Reynolds stress modells*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2000.
- [96] D.C. Wilcox. *Turbulence Modelling for CFD*. DWC Industries, La Cananda, 1998.
- [97] J. Wild, P. Niederdrenk, and T. Gerhold. Marching generation of smooth structured and hybrid meshes based on metric identity. *Proceedings of the 14th International Meshing Roundtable*, pages 109 – 127, 2005.

- [98] S. Yoon and A. Jameson. An LU-SSOR scheme for the Euler and Navier-Stokes equations. *AIAA Journal*, (26):1025–1026, 1988.
- [99] O. C. Zienkiewicz and R. L. Taylor. *The Finite-element method for Solid and Structural Mechanics*, volume 6th Edition. 2005. John Wiley and Sons, New York, NY.

13 Abbildungsverzeichnis

1	Experimente im Windkanal in der Flugzeugentwicklung (a) Quelle [44], Hybrides Netz um Flügel-Rumpf-Pylon-Gondel Konfiguration (b)	4
2	CFD - Einsatz in der Flugzeugentwicklung (Quelle [44]).	4
3	Hochauftriebskonfiguration $Re = 25 \times 10^6$, $Ma = 0.2$, $\alpha = 19^\circ$	5
4	Konvergenzgeschwindigkeit (a) und parallele Performance (b) des TAU-Codes [84] mit DLR F6 2×10^6 Punkte Netz [48] gerechnet auf [42].	6
5	Typische Prozesskette einer Strömungssimulation.	7
6	RAE 2822 Flügelprofil (a) und Skizze der Kontrollvolumina (b).	11
7	Beispiele für nicht <i>optimale</i> hybride Netze	13
8	Beispiele von Netzpartitionen.	14
9	Wandnahe Auflösung der Netzlinien mit Geschwindigkeitsprofil für den klassischen Fall (a) und für die Verwendung von adaptiven Wandfunktionen (b)	24
10	Schematische Darstellung der Bestimmung von $y^+(1)$	24
11	Konvergenzverlauf des Residuums der Dichte und des Auftriebsbeiwerts (C_L) auf Netzen mit unterschiedlichen $y^+(1)$ -Werten für den RAE 2822 Fall 10.	25
12	Landekonfiguration A380 ((A) Abb. 9-10): Ausgangsnetz mit $y^+(1)$ -Verteilung nicht adaptiert (a), oberste Prismenschicht im Übergang zum unstrukturierten Bereich (b), Zoom auf die Vorderkante mit abgebrochenen Prismenschichten (c).	26
13	Reiseflugkonfiguration A320 ((A) Abb. 7-8): Ausgangsnetz mit unstrukturierten Vierecken und $y^+(1)$ -Verteilung nicht adaptiert (a), einzelne Prismenstapel eingeschlossen von Hexaederstapeln (vgl. Abb. 8 (b)) (b), Hinterkante des Flügels (c).	27
14	Vergleich der Konvergenzverläufe des Dichteresiduums und des Auftriebsbeiwerts (C_L) für die beiden Konfigurationen aus (A).	29
15	Skizze des Deltaflügels (b) und Schnitte durch die Netze bei $x = 200$	31
16	Schnitte durch die Netze bei $z = 0$ und verschiedenen Anstellwinkeln α	32
17	Visualisierung des Totaldrucks p_{tot} und der Kantenlänge \hat{h} bei $\alpha = 22^\circ$	32
18	Visualisierung des Wirbelsystems mit unterschiedlichen Kriterien.	33
19	Schnitte durch die Deltaflügelnetze bei $y = -100$ und $\alpha = 12^\circ$	34
20	Skizze eines 3v und 4w Multigitter Zyklus.	37
21	Vergleich unterschiedlicher Multigitterzyklen mit dem DLR-F6 (2.0×10^6 Netzpunkte). Parallelle Performance TAU-Code (a), Konvergenzverlauf (b).	38
22	Parallelisierung des TAU-Code Strömungslösers.	38
23	Strategie des <i>Backfillings</i> . Der Batch-Job an der 4. Stelle kann aufgrund der Ressourcenanforderungen (32-128 CPUs und 8h Laufzeit) vorgezogen werden, da alle anderen wegen ihrer Ressourcenanforderungen nicht abgearbeitet werden können.	40
24	Skizze eines Batch-Jobs mit variabler CPU-Ressourcenangabe und der Rolle des dynamischen Netzpartitionierers innerhalb der Simulationskette.	40

25	Die variable Angabe von CPU-Ressourcen ermöglicht dem Batch-Job an der 3. Stelle die Nutzung aller freien Ressourcen. Durch <i>Backfilling</i> wird dieser Job dazu genutzt, die sonst brachliegenden CPUs auszulasten.	41
26	Dynamische Netzadaption mit anschließender Repartitionierung für die A380 Konfiguration. (a) Startnetz, (b) 1. Adaption, (c) 2. Adaption und (d) 3. Adaption.	42
27	X-31: (a) zeigt die <i>clean wing</i> Konfiguration und (b) das Modell mit Steuerklappen im Windkanal.	46
28	F-16XL CAWAPI Konfiguration (a) und ein Vergleich der Oberflächenauflösung der von Fritz in [25] verwendeten Netze (b) und (c).	47
29	Netzschnitte an Position 2 bei FC 25 (siehe Tabelle 7).	48
30	Netzschnitte an Position 5 bei FC 25 (siehe Tabelle 7).	48
31	Vergleich zwischen Flugexperiment und Simulationsrechnung auf unterschiedlichen Netzen (FC 7: $\alpha = 11.89^\circ$, $Mach = 0.304$, $Re = 44.4 \times 10^6$).	49
32	Vergleich zwischen Flugexperiment und Simulationsrechnung auf unterschiedlichen Netzen (FC 25: $\alpha = 19.84^\circ$, $Mach = 0.242$, $Re = 32.22 \times 10^6$).	50
33	Vergleich der Standard-Agglomeration mit drei unterschiedlichen Varianten der anisotropen Agglomeration bei dem DLR F6 Fall mit 4w Multigitterzyklus auf 32 CPUs (gerechnet mit Runge-Kutta auf [21] - vgl. auch Tab. 2 auf Seite 37).	51
34	Vergleich der Standard-Agglomeration mit drei unterschiedlichen Varianten der anisotropen Agglomeration bei einer Reiseflugkonfiguration mit 3v Multigitterzyklus auf 64 CPUs (gerechnet mit LU-SGS [16] auf [21]).	51
35	Vergleich der parallelen Performance unterschiedlicher TAU-Code Versionen mit der Hochauftriebskonfiguration (siehe Abb. 3 auf Seite 5) gerechnet auf [42].	52
36	Vergleich der parallelen Performance unterschiedlicher TAU-Code Versionen mit der Hochauftriebskonfiguration (siehe Abb. 3 auf Seite 5) gerechnet auf [88].	53

Teil IV

Publikationen

(A)
Near-wall grid adaptation for turbulent flows

Thomas Alrutz, Tobias Knopp

*DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

International Journal of Computing Science and Mathematics,
Volume 1, Issue 2–4, Pages 177–192, 2007

Abstract

This paper considers grid-adaptation techniques for the boundary-layer region of hybrid meshes, where the mesh is of regular structure consisting of prismatic resp. hexaedral elements. The method helps to improve convergence, accuracy and reliability of aerodynamic simulations for the two possible types of wall-boundary conditions for viscous flows, i.e., integrating the flow equations down to the wall (no-slip) or using wall-functions. The underlying algorithms and their parallelization are described and the method is applied to 3D aerodynamic configurations. Moreover, accuracy and performance gain using wall-functions with grid-adaptation are discussed for transonic airfoil flows.

1 Introduction

The growing demand for fast and accurate CFD tools in industry requires a continuation of efforts in numerical algorithms, regarding both the accuracy and the performance of the method. The present paper focuses on techniques for more accurate and efficient treatment of the near-wall region for aerodynamic flows. Typically more than 60% of the mesh points for an airplane simulation are resided in boundary layers. Up to 50% of the boundary-layer nodes have to be placed into the near-wall region for resolving the sharp gradients of the solution near the wall and to capture the complicated near-wall flow structures in case of separation. Moreover, this region causes numerical stiffness problems due to the small cell height in conjunction with the steep gradients.

In the present paper an industrial Finite-Volume solver is considered, but the ideas also apply to other numerical approaches and are also relevant for higher order methods. Current industrial CFD solvers are already highly optimised regarding algorithms and code design. As will be seen, the present paper provides potential for further performance acceleration, which is complementary to improvements of the nonlinear solver and mesh refinement/derefinement using flow-based sensors or target function (adjoint) based adaptation.

Consider the steady-state Favre-averaged compressible Navier-Stokes equations in a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$ ($d = 2, 3$) with the eddy-viscosity assumption for the Reynolds-stress tensor and the gradient-diffusion approximation for the turbulent heat-flux vector in conservative form

$$-\int_{\partial V} \mathbb{F} \cdot \vec{n} dS + \int_{\partial V} \mathbb{F}_v \cdot \vec{n} dS = 0$$

where V is an arbitrary control volume with closed boundary surface ∂V , and \vec{n} is the unit normal vector in outward direction. The vector of conservative variables is denoted by $\vec{W} = (\rho, \rho\vec{u}, \rho E)$, and the convective and viscous flux tensor \mathbb{F} and \mathbb{F}_v resp. are given by

$$\mathbb{F} = \begin{pmatrix} \rho\vec{u} \\ \rho\vec{u} \otimes \vec{u} + p\mathbb{I} \\ \rho E\vec{u} + p\vec{u} \end{pmatrix}, \quad \mathbb{F}_v = \begin{pmatrix} 0 \\ \mathbb{T} \\ \mathbb{T}\vec{u} - \vec{q} \end{pmatrix}$$

where $E = c_v\theta + \vec{u}^2/2$ is the total specific energy. The pressure is given by $p = (\gamma - 1)\rho(E - \frac{1}{2}\vec{u}^2)$. The heat-flux vector \vec{q} is given by $\vec{q} = -\kappa_e \vec{\nabla} \theta$ and the strain rate

tensor $\mathbb{T}(\vec{u})$ is

$$\mathbb{T}(\vec{u}) \equiv \mathbb{S}(\vec{u}) - \frac{1}{3} \vec{\nabla} \cdot \vec{u} \mathbb{I}, \quad \mathbb{S}(\vec{u}) = \frac{1}{2} \left(\vec{\nabla} \vec{u} + (\vec{\nabla} \vec{u})^T \right)$$

with effective viscosity $\mu_e = \mu + \mu_t$ and effective thermal conductivity $\kappa_e = \kappa + \kappa_t$. The turbulent viscosity μ_t is computed using a turbulence model of Spalart-Allmaras type by [6] or of $k-\omega$ type using [9] and for κ_t the concept of a turbulent Prandtl number is used.

For these models, at the wall the no-slip condition $\vec{u} = 0$ is imposed and the accurate numerical resolution of both attached and separated boundary layers requires a so-called *low-Reynolds grid* with $y^+(1) \approx 1$. Denote $y(1)$ the distance of the first node above the wall and v_t the wall-parallel component of the velocity, then $y^+(1)$ is defined by

$$y^+(1) \equiv \frac{y(1)u_\tau}{\nu}, \quad \text{where } u_\tau = \sqrt{\nu \frac{\partial v_t}{\partial n}}, \quad \nu = \frac{\mu}{\rho} \quad (1)$$

with density ρ , viscosity μ , outer normal \vec{n} and so-called friction velocity u_τ .

The condition $y^+(1) \approx 1$ has to be ensured during grid-generation by specifying the first spacing $y(1)$ and using an estimate for u_τ , e.g., from a semi-empirical relation for a flat-plate turbulent boundary layer. However, there are regions in complex geometries where the grid-generator cannot reach the specified $y(1)$ -value. Moreover, the semi-empirical guess for u_τ may deviate from the value for u_τ from the RANS solution in regions of complex flow. As a remedy, the DLR TAU-code provides a grid-adaptation module, which allows to ensure a user-defined target value for $y^+(1)$ based on the underlying RANS solution. This method will be referred to as y^+ -adaptation.

In order to remedy the arising stiffness problems and costs for the large number of near-wall grid nodes for the no-slip boundary condition, an alternative boundary condition called *wall-functions* prescribes no-penetration $\vec{u} \cdot \vec{n} = 0$ and the tangential component of the wall-shear stress at solid walls, where we use the approximative relation

$$\mathbb{T} \cdot \vec{n} = (\mathbb{I} - \vec{n} \otimes \vec{n}) \mathbb{T} \cdot \vec{n} + \vec{n} \otimes \vec{n} \mathbb{T} \cdot \vec{n} \approx -\tau_w \vec{v}_{t,\delta}$$

where $\vec{v}_{t,\delta}$ is the wall-tangential component of the velocity vector at the first node above the wall, and τ_w is the magnitude of the wall-shear stress which is computed using a wall-function method: Given the magnitude of the wall-parallel velocity at the first grid node above the wall $u(1)$ at wall distance $y(1)$, $\tau_w = \rho u_\tau^2$ is computed from a non-linear relation $u(1)/u_\tau = F(yu_\tau/\nu)$. Following [7], F is then the turbulence-model specific solution of a flat-plate turbulent boundary layer at zero pressure gradient. Such walls functions allow for solutions independent of the wall-normal spacing $y(1)$ for flows close to equilibrium, e.g., fully developed turbulent boundary layer flows at zero pressure gradient, see [7]. However, aerodynamic flows are characterised by (a) stagnation points and subsequent laminar resp. not fully developed turbulent flow, (b) regions of significant pressure gradient parameter due to a strong (adverse) pressure gradient at a typically moderate Reynolds number and (c) regions of separation and reattachment. These flow situations are very important not only for flows around airfoils and rotor blades, but even more for complex aircraft configurations with flaps, engines etc. In these critical flow situations (a)-(c) local mispredictions using wall functions

may occur, which may cause large deviations in integral coefficients of engineering interest as lift, moment and drag.

In order to use the advantages of wall-functions, the present paper considers a y^+ -grid adaptation technique to ensure a locally appropriate resolution depending on both the near-wall flow physics to be captured and the range of validity of the wall-function model. Critical regions are characterised by non-equilibrium flow situations which are detected by a flow-based sensor. The near-wall grid adaptation is then made possible due to the hybrid character of the wall-function method, e.g., $y(1)$ can be shifted in the very near wall region without introducing an error stemming from an inconsistent coupling of turbulence models. We note that in classical wall-function methods, such an error is present due to the coupling of different turbulence models, i.e., a one- resp. two-equation model for the global flow and an *ad-hoc* patched algebraic model for the near-wall region.

This paper is organised as follows: The basic y^+ -adaptation is considered in Section 2. The parallelization of the method is described in Section 2.3. The y^+ -adaptation for wall-functions is presented in Section 3, and applications to aerodynamic flows using wall-functions are given in Section 3.1.

2 Near wall grid adaptation

Based on the dual mesh approach, the TAU-code supports hybrid grids, which may be composed of tetrahedra, prisms, hexahedra and/or pyramids. In the near-wall region, this allows to use anisotropic regular meshes consisting of hexahedra and/or prisms with a high aspect ratio whose edges are aligned with the wall-normal and wall-parallel directions. These are fitted to boundary layer flows, i.e., with a relatively large spacing in streamwise and spanwise direction but with a fine spacing in wall-normal direction with a suitable stretching factor to resolve the steep wall-normal gradients of the solution.

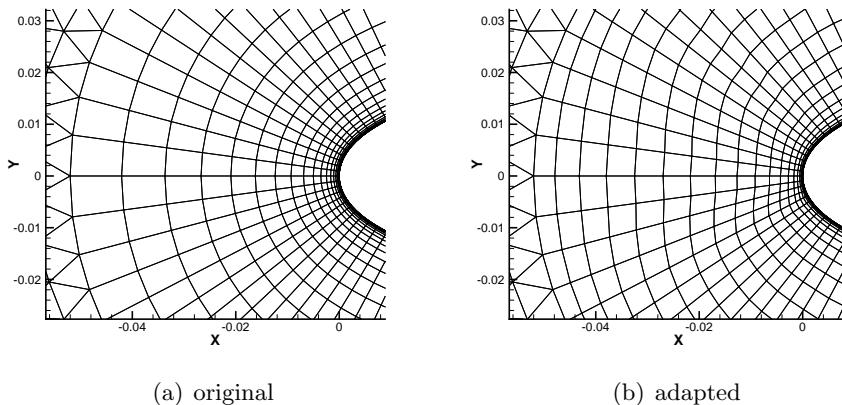


Figure 1: Example of near-wall grid adaptation.

2.1 y^+ -condition for low-Reynolds grids

An accurate integration of the RANS equations down to the wall requires a so-called *low-Reynolds grid* with $y^+(1) = y(1)u_\tau/\nu \approx 1$ for all first nodes above the wall. In 3D flows, friction velocity u_τ may be computed from

$$u_\tau = \sqrt{\frac{\mu}{\rho} |\Omega|}, \quad \text{with } |\Omega| = \sqrt{2\Omega(\vec{u}) : \Omega(\vec{u})} \quad (2)$$

where the vorticity tensor $\Omega(\vec{u})$ is defined by

$$\Omega(\vec{u}) = \frac{1}{2} (\vec{\nabla} \vec{u} - (\vec{\nabla} \vec{u})^T) \quad (3)$$

with the notation $A : B = \sum_{i,j=1}^d A_{ij}B_{ij}$ for two tensors A, B , with space dimension $d = 2, 3$.

The effect of a different $y^+(1)$ value for all first nodes above the wall on the convergence and the accuracy of the calculated solution can be seen in Figure 2. Here we investigate 3 different hybrid grids for the RAE2822 airfoil case 10 with varying values for $y^+(1)$.

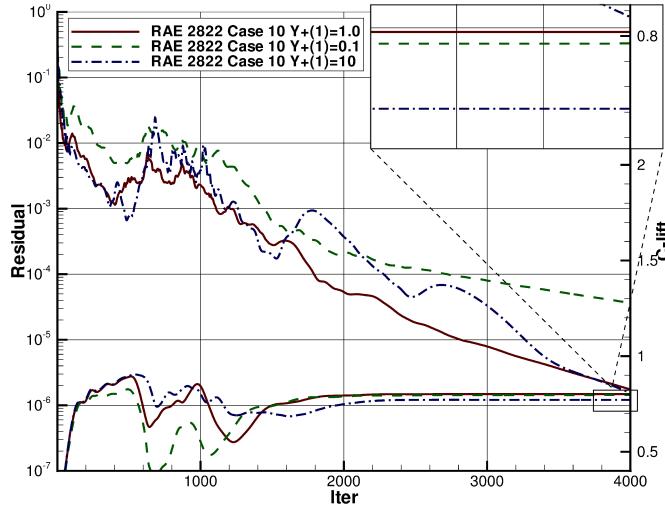


Figure 2: Convergence history of the residual and c_l

From Figure 2 we can see that on the grid with $y^+(1) \approx 0.1$ convergence is significantly slower than on the one with $y^+(1) \approx 1.0$. On the other hand, the grid with a value of $y^+(1) \approx 10$ converges with the same order but to an incorrect value of c_l . This motivates algorithms for adaptation of the near-wall grid w.r.t. $y^+(1)$ to ensure the low-Re grid condition.

2.2 Basic y^+ -adaptation algorithm

We assume a given surface discretisation composed of triangles and/or quadrilaterals. Inside the regular near-wall layer, the nodes are located on (almost) wall-normal rays starting at the corresponding wall node. We use the following notation:

- I. \vec{x}_{wp} : Wall node,
- II. \vec{x}_{np} : First node above the wall w.r.t. node wp,
- III. $\{\vec{x}_{wp} + \lambda_p \vec{r}\}$: Ray of points starting at \vec{x}_{wp} and ending at the outer edge of the regular (prismatic) layer; $\{\vec{x}_{wp} + \lambda_p \vec{r}\} \equiv \{\vec{x} \in \mathbb{R}^d \mid \vec{x} = \vec{x}_{wp} + \lambda_p \vec{r}, 0 \leq p \leq p_{\max}\}$ where the direction vector \vec{r} may be non-constant and λ_p describes the point distribution in wall-normal direction.

Moreover we assume that $\vec{x}_{np} - \vec{x}_{wp}$ is almost parallel to the surface normal vector \vec{n} . Then the algorithm for y^+ -adaptation with $y_{\text{target}}^+ = 1$ reads as follows.

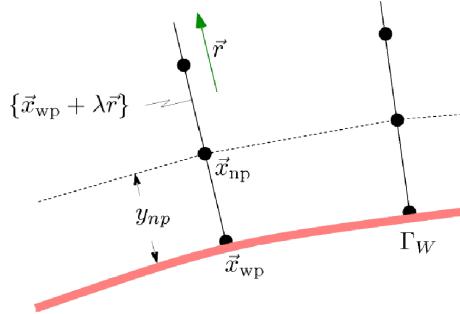


Figure 3: Illustration of grid adaptation algorithm.

- 1.** Read RANS solution and grid.
- 2.** y^+ -grid adaptation.
 - 2.1.** For each surface node \vec{x}_{wp} do:
 - 2.1.1.** Determine \vec{x}_{np} and compute u_τ from vorticity using Equation (2).
 - 2.1.2.** Determine $y_{np}^+ = y_{np} u_\tau / \nu$, $y_{np} = |\vec{x}_{np} - \vec{x}_{wp}|$
 - 2.1.3.** Set $y_{\text{new}} = y_{np} y_{\text{target}}^+ / y_{np}^+$.
 - 2.2.** Smooth the y_{new} -distribution.
 - 2.3.** For each surface node redistribute the points on its ray $\{\vec{x}_{wp} + \lambda_p \vec{r}\}$ where the last point $\{\vec{x}_{wp} + \lambda_{p,\max} \vec{r}\}$ remains unchanged.
- 3.** Interpolate RANS solution from old grid to new grid.

Smoothing of the y_{new} -distribution is performed as follows. Denote K the number of smoothing steps, $y_{np}^i = y_{\text{new}}$ of node \vec{x}_{np}^i after (2.2), $\mathcal{N}(i)$ the set of indices of neighbour surface nodes of node i and $\#\mathcal{N}(i)$ their number. Then in smoothing step k

$$y_{np}^{i,k} = (1 - \epsilon) y_{np}^{i,k-1} + \epsilon y_{\text{nei},k-1}^{\text{nei},k-1} \quad (4)$$

with smoothing parameter ϵ and

$$y_{\text{np}}^{\text{nei},k-1} = \frac{1}{\#\mathcal{N}(i)} \sum_{j \in \mathcal{N}(i)} y_{\text{np}}^{j,k-1}.$$

Along each ray the grid point distribution is of hyperbolic tangent type with two lateral controls: The first control is used for the wall node and allows to ensure the target value for $y^+(1)$. The second control can be used at the outer edge of the prismatic layer to ensure that the ratio of spacing of the last prism to the spacing of the first tetrahedron of the unstructured region is well balanced (see also [10, 11]). Figures 7-10 demonstrate the effect of the y^+ -adaptation for complex wing-body configurations.

2.3 Parallelization of the algorithm

The possibility to use the y^+ -adaptation in parallel mode is essential to encounter the growing needs of engineers for large and complex configurations (see [3]). The TAU-code uses domain decomposition for the parallelization (Figure 4). A classical

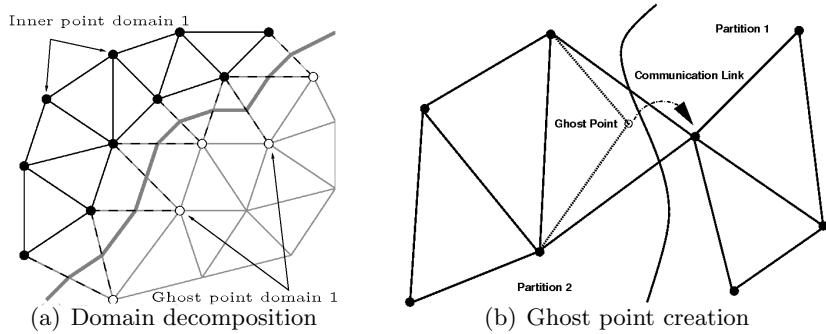


Figure 4: Domain decomposition with one cell overlapp.

way to achieve a domain decomposition of the computational grid is to use a geometric or graph based partitioner (see [1] for details). A domain decomposition without specific requirements will destroy the layer or ray information which is necessary for the

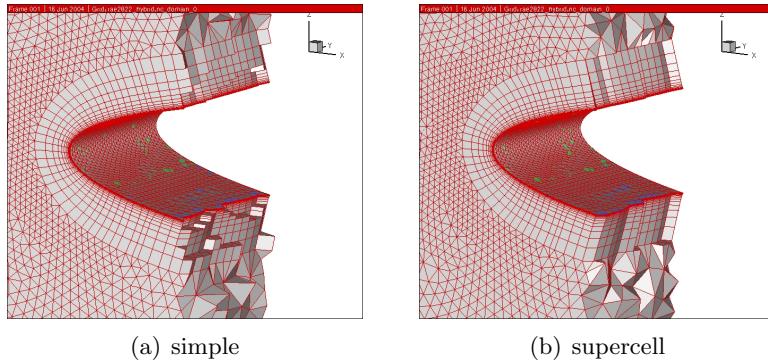


Figure 5: RAE2822 subgrids from domain decomposition.

calculation of the wall normal rays (see condition III). Figure 5 (a) shows an example of a simple domain decomposition of a 3D RAE2822 airfoil with splitted prism stacks. One way to prevent the splitting of the prism or hexahedral stacks above the wall is the collapsing of a complete element stack into a *super cell*. These *super cells* are generated by joining all prism/hexahedra above a surface triangle/quadrilateral during the calculation of the graph for the grid partitioning algorithm (see Figure 6). This has two

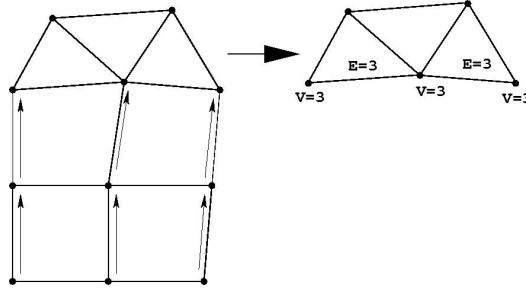


Figure 6: Illustration of super cell generation.

effects. Firstly the graph for the calculation of the load balancing is reduced. Secondly the modified graph can be used by any graph or geometric partitioner and the result will always yield to subgrids which are shaped like the example in Figure 5 (b). This is due to the fact that the grid partitioner gets only a graph with some *heavy weighted* elements and is therefore unable to split the stacks which are represented by those *super cells*. A detailed description can be found in [2]. Using the preservation of the structural layer parts in the boundary regions of the grid, the parallel y^+ -adaptation algorithm reads as follows:

1. Compute a domain decomposition from the initial grid.
2. On each CPU read the corresponding RANS solution and grid domain.
3. Parallel y^+ -grid adaptation.
 - 3.1. Let N_{own} denote the number of all *inner* surface nodes \vec{x}_{wp}^i in a domain, then do for each \vec{x}_{wp}^i with $0 \leq i < N_{own}$:
 - 3.1.1.-3.1.3 Do the same as in the basic algorithm, see section 2.2 (**2.1.1.-2.1.3.**)
 - 3.2. Exchange y_{new} to all *ghost* nodes (see Figure 4 (b)).
 - 3.3. Set $y_{np}^i = y_{new}$ for each *inner* and *ghost* surface node \vec{x}_{np}^i after **3.2..**
 - 3.4. For each smoothing step k do:
 - 3.4.1. Smooth each $y_{np}^{i,k}$, $0 \leq i < N_{own}$ with Equation (4).
 - 3.4.2. Exchange $y_{np}^{i,k}$ to all *ghost* nodes.
 - 3.5. For each *inner* surface node redistribute the points on its ray $\{\vec{x}_{wp} + \lambda_p \vec{r}\}$ where the last point $\{\vec{x}_{wp} + \lambda_{p,\max} \vec{r}\}$ remaines unchanged.

3.6. Exchange on all *ghost* surface nodes the corresponding wall-normal ray of points.

4. Interpolate RANS solution from old grid domain to new grid domain.

As an example, Figure 7-10 show the application of the y^+ -adaptation for low-Re meshes for two complex 3D configurations.

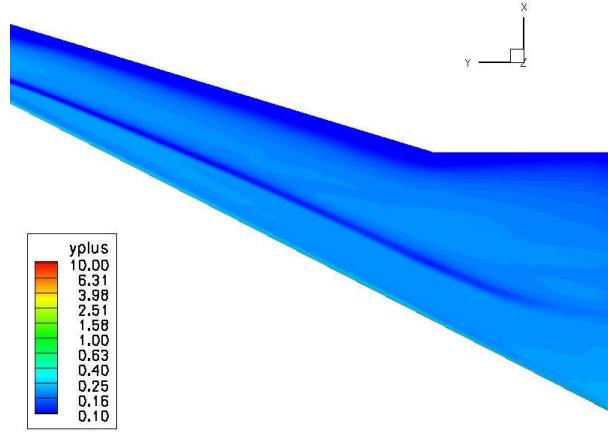


Figure 7: A320 cruise configuration ($Re = 6.5 \times 10^6$, $Ma = 0.75$, $\alpha = 2.8^\circ$): y^+ -distribution without y^+ -adaptation

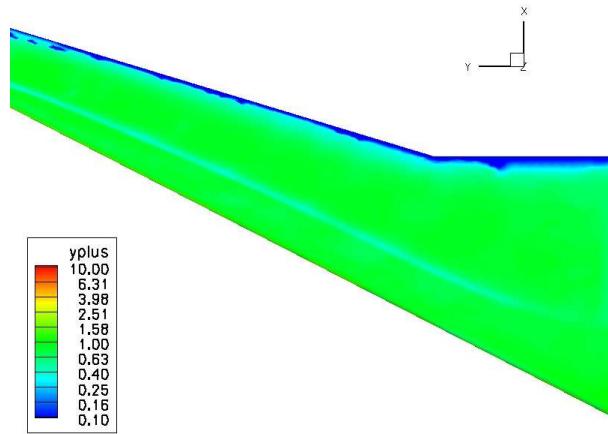


Figure 8: A320 cruise configuration ($Re = 6.5 \times 10^6$, $Ma = 0.75$, $\alpha = 2.8^\circ$): y^+ -distribution with y^+ -adaptation

3 Y^+ -adaptation for wall functions

Universal wall-functions are designed for attached, fully developed turbulent boundary layer flow at zero pressure gradient. However, in airfoil flows this assumption is violated in several situations: At the leading edge, there is a stagnation point and subsequently

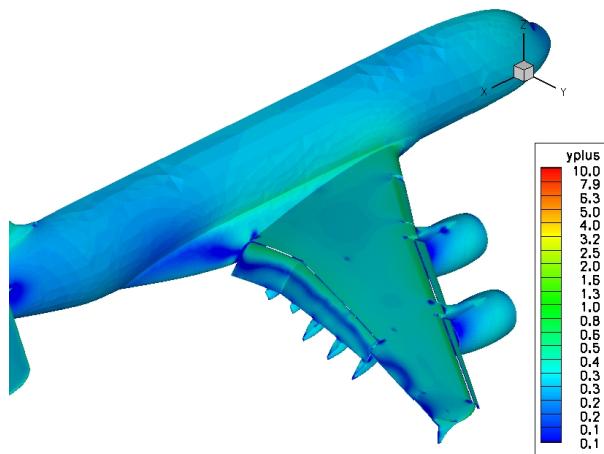


Figure 9: A380 landing configuration ($Re = 1.5 \times 10^6$, $Ma = 0.18$, $\alpha = 5.6^\circ$): y^+ -distribution without y^+ -adaptation

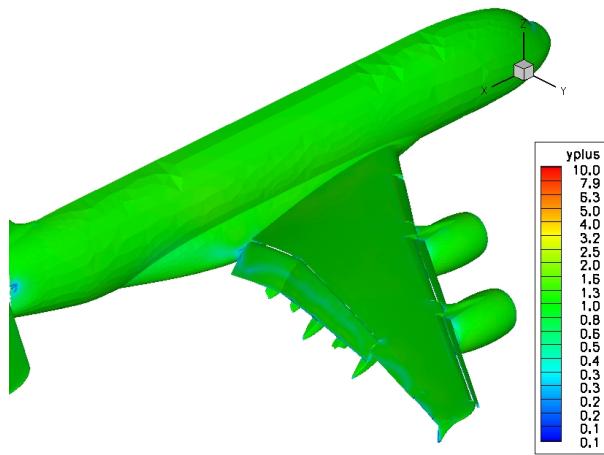


Figure 10: A380 landing configuration ($Re = 1.5 \times 10^6$, $Ma = 0.18$, $\alpha = 5.6^\circ$): y^+ -distribution with y^+ -adaptation

the flow is not fully developed turbulent. Moreover the flow around the leading edge is strongly accelerated. Secondly, regions of strong adverse pressure gradient and regions of separated flow are beyond the range of validity of wall-functions if $y^+(1)$ is too large, see [7, 8].

As a remedy, these flow situations are detected by a flow-based sensor, and the near-wall points are shifted towards the wall, i.e., the low-Re flow regime. On grids with $y^+(1) \gtrsim 6$, relation (2) ceases to be valid. Then u_τ is estimated from the following nonlinear equation to be solved using Newton's method

$$u^+(1) = F_{\text{Rei}}(y^+(1)) \quad (5)$$

where the so-called wall law by Reichardt is given by

$$\begin{aligned} F_{\text{Rei}}(y^+) &\equiv \frac{\ln(1 + 0.4y^+)}{\kappa} \\ &+ 7.8 \left(1 - e^{-\frac{y^+}{11.0}} - \frac{y^+}{11.0} e^{-\frac{y^+}{3.0}} \right) \end{aligned} \quad (6)$$

with $u^+(1) = u(1)/u_\tau$, $y^+(1) = y(1)u_\tau/\nu$, $\kappa = 0.41$ and $u(1)$ being the wall-parallel velocity at $y(1)$.

Regions of complex flow situations are detected by a flow-based sensor based on the pressure gradient parameter p^+ as indicator

$$p^+ = \frac{\nu}{\rho u_\tau^3} \frac{dp}{dx} \quad (7)$$

which is computed from the streamwise pressure gradient dp/dx . As stagnation point and separation point are approached, $u_\tau \rightarrow 0$ and thus $p^+ \rightarrow \pm\infty$. Non-small p^+ -values, say $p^+ \gtrsim 0.05$, in adverse pressure gradient flow cause a significant departure from the universal wall-law for large y^+ (see [8]).

Then in the algorithm for y^+ -adaptation, step (2.2) is modified as follows if wall-functions are used. Figure 11 provides some illustration.

2.1. For each surface node \vec{x}_{wp} do:

2.1.1. Determine \vec{x}_{np} and compute u_τ using (6). Then set $y_{\text{np}} = |\vec{x}_{\text{np}} - \vec{x}_{\text{wp}}|$ and determine $y_{\text{np}}^+ = y_{\text{np}} u_\tau / \nu$.

2.1.2. Determine the local pressure gradient parameter p^+ from (7) and check if \vec{x}_{wp} is located in a region of strong pressure gradient

2.1.3. Check if \vec{x}_{wp} is located in the leading edge region of the airfoil (or the wing resp.)

2.1.4. Check if point \vec{x}_{wp} resides in a separation region

2.1.5. Based on **2.1.2.-2.1.4.** set target value y_{target}^+ .

2.1.6. If $y_{\text{target}}^+ < y_{\text{np}}^+$ then set $y_{\text{new}} = y_{\text{np}} y_{\text{target}}^+ / y_{\text{np}}^+$, else set $y_{\text{new}} = y_{\text{np}}$.

As an additional indicator for the leading edge region the surface curvature may be used, see e.g. [4] for computational techniques. For 2D flows, regions of separated flow can be detected by the condition of recirculating flow $\vec{u}_\infty \cdot \vec{u}_{np} < 0$, where \vec{u}_∞ and \vec{u}_{np} denote the farfield velocity and the velocity at node \vec{x}_{np} resp. In 3D flows in complex geometries, detection of separated flow is more complicated. Albeit, the separation point is still indicated by large p^+ -values.

Concerning the target value for y^+ , numerical tests suggest $y_{target}^+ \in [1, 5]$ in regions of separated flow, and $y_{target}^+ \in [5, 10]$ in the leading edge region. Numerical simulations

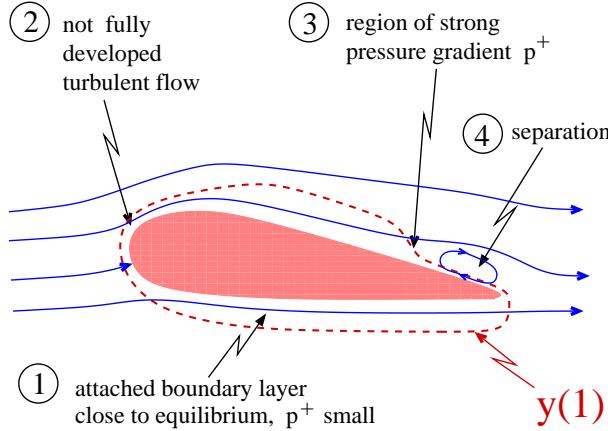


Figure 11: Illustration of flow sensor based grid-adaptation for wall-functions

show that the most important failure of universal wall functions is not related to flow separation but to too large $y^+(1)$ values in the leading edge region, in particular for small Reynolds numbers. Therefore a simplified grid-adaptation has to ensure at least $y^+(1) \lesssim 10$ around the leading edge of the airfoil.

3.1 Application to aerodynamic flows

The method is applied to the transonic flow around the RAE2822 airfoil (at $Ma = 0.75$, $Re = 6.2 \times 10^6$, and $\alpha = 2.8^\circ$ with shock induced separation). Calculations are performed on a series of O-type grids of hybrid type. The boundary layer is fully contained in the regular prismatic grid. In wall-normal direction a geometrical point distribution is used. We use grids with different spacing of the first node above the wall and a different number of nodes in the structured layer such that the thickness of the prismatic layer remains almost constant.

We are interested in an improved grid-independence of the results close to the leading edge and in the aft-shock separation region. In Figure 12 a typical distribution of $y^+(1)$ for the SA model without and with y^+ -adaptation is given. As intended, the wall-normal grid is shifted towards the low-Re regime in the vicinity of the leading edge, close to the shock and in the separation region. In Figure 13 details of the suction peak of the c_p distribution without and after y^+ -adaptation are shown. The improvement is important for grid-independent predictions of aerodynamic moments.

Table 1 shows an investigation of accuracy of the wall-function method versus gain

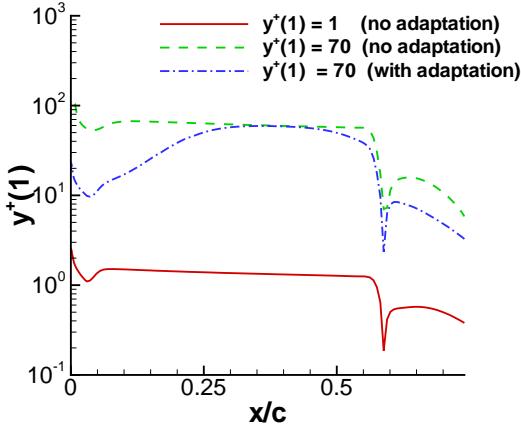


Figure 12: RAE 2822 case 10: Distribution of $y^+(1)$ on the suction side.

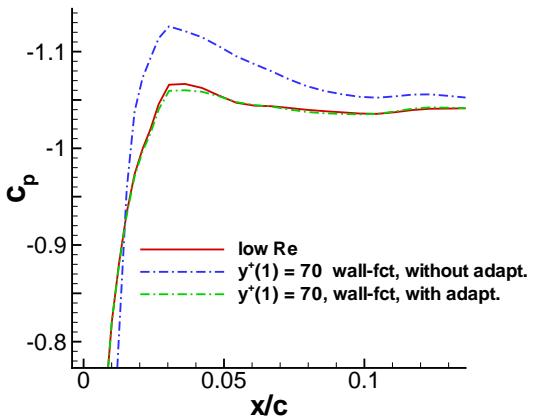


Figure 13: RAE 2822 case 10: Detail of c_p near leading edge using SA-E turbulence model.

in performance (reduction of grid points and CPU-time). Regarding the accuracy of the method, the table shows that the aerodynamic coefficients lift c_l and drag c_d are almost grid independent. The number of nodes N_y along each wall-normal ray in the prismatic layer and the number of total nodes in the mesh N_{total} are also given. The number of Runge-Kutta iterations (Iter.) for achieving a residual less than 2×10^{-5} and the CPU time required on a LINUX Opteron 2.2GHz processor are also shown. Note that the present calculations use a simple three-stage Runge-Kutta method for iterating the nonlinear fixed point problem in pseudo time without multigrid. The reason is to exclude effects by other algorithms on the convergence results, e.g., the agglomeration of dual-grid cells on coarser levels in the multigrid method, and open problems regarding an optimal choice of the pseudo time step on coarse mesh levels in explicit solvers.

$y^+(1)$	c_l	c_d	N_y	N_{total}	Iter.	CPU
low-Re	0.7912	0.02687	36	23038	15872	4647
1	0.7911	0.02685	36	23038	15020	4544
5	0.7894	0.02692	32	21467	15107	4240
8	0.7831	0.02682	29	20518	12839	3490
12	0.7838	0.02680	27	19860	12080	3248
24	0.7934	0.02693	24	18865	11114	2888
50	0.7956	0.02688	20	17461	10290	2445
70	0.7969	0.02676	18	16766	9981	2307

Table 1: Integral coefficients lift c_l and drag c_d for RAE2822 profile using SA-E model with wall-functions on meshes with different $y^+(1)$. As a reference, the results for $y^+(1) = 1$ using the no-slip condition denoted low-Re are also given.

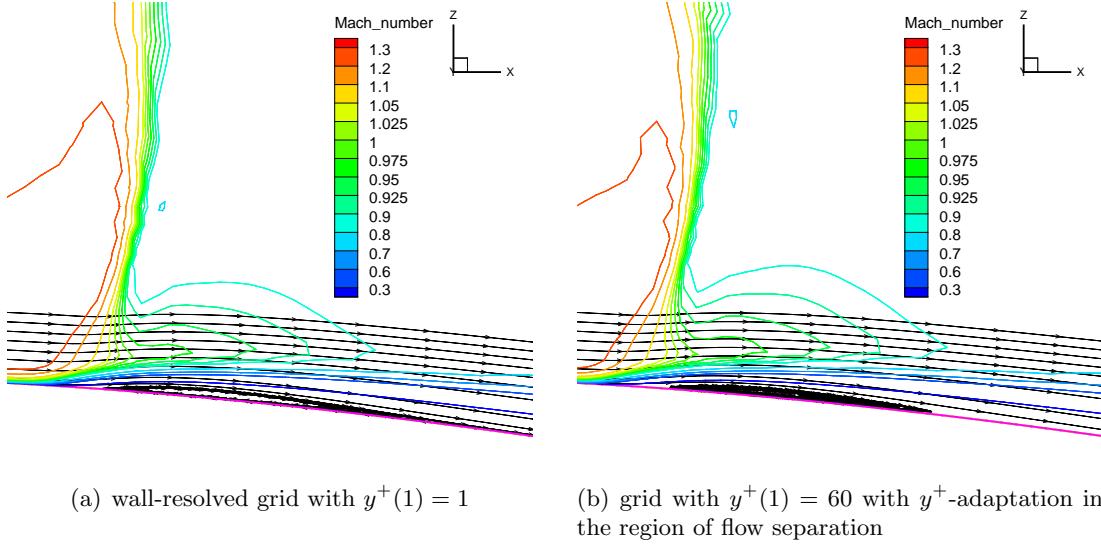


Figure 14: RAE 2822 case 10: Details of shock-pattern (Mach number isolines) and streamtraces for velocity

We point out that the RAE airfoil flow is a rather challenging 2D test-case for wall-functions. Although the number of grid nodes in the prismatic layer are reduced by a factor of two using wall-functions, the number of grid points required to resolve the shock outside the boundary layer is also large, which limits the total performance gain. For SA-E model, Figure 14 shows that even for flow-details like Mach-number contours and streamtraces of velocity, there is close agreement between the wall-resolved RANS solution and the solution with wall-functions.

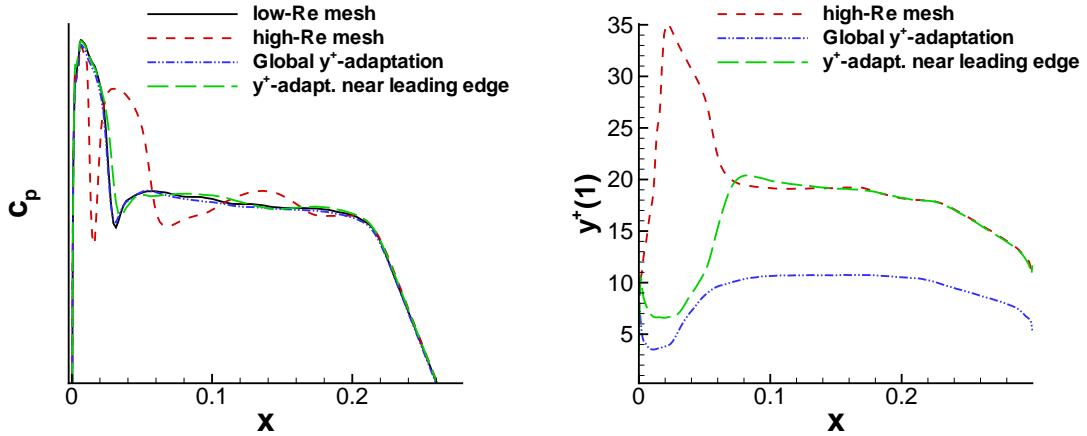


Figure 15: VC-Opt profile at transonic onflow conditions: c_p -distribution on suction side for SA-E model on low-Re and high-Re mesh without and with y^+ -adaptation.

Figure 16: VC-Opt profile at transonic onflow conditions: y^+ -distribution on suction side without and with y^+ -adaptation.

Secondly, the method is applied to the VC-Opt profile, which is similar to an AIRBUS profile, at onflow conditions $Re = 2 \times 10^6$, $Ma = 0.757$ and $T_\infty = 217\text{K}$. As industrial data are confidential, the scaling in Figure 15 for c_p is not shown. The low-Re grid uses 33 points in the prismatic layer, whereas the high-Re grid only uses 15. As wall-functions are designed for fully developed turbulent flow, the relatively low Reynolds number (from wind-tunnel testing) causes additional problems and the method fails unless the y^+ -distribution is adapted. In Figure 16 two strategies are shown, viz., (i) ensuring small $y^+(1)$ with $y_{\text{target}}^+ = 8$ only around the leading edge or (ii) choosing $y_{\text{target}}^+ = 4$ near the leading edge and $y_{\text{target}}^+ = 10$ in the region of equilibrium flow.

4 Conclusions

The presented grid-adaptation techniques for the boundary-layer region of hybrid meshes allow to significantly improve convergence, accuracy and reliability of aerodynamic simulations. In order to fully benefit from the large reduction of grid-nodes in the near-wall region using wall-functions, grid-adaptation also in the outer flow is required to reduce the number of nodes remote from boundary layers. Application of the method to 3D full aircraft configurations will be subject of future research.

Acknowledgement

Valuable discussions with Peter Niederdrenk and Kai Richter and the suggestions of the reviewers are gratefully acknowledged.

References

- [1] Thomas Alrutz. Investigation of the parallel performance of the unstructured DLR-TAU-Code on distributed computing systems. In *Proceedings of the 16th Parallel CFD Conference*, 2005. Egmond aan Zee, The Netherlands, May 21-23.
- [2] Thomas Alrutz and Matthias Orlt. Parallel dynamic grid refinement for industrial applications. In P. Wesseling, E. Oñate, and J. Periaux, editors, *Proceedings ECCOMAS 2006*, September 5-8 2006. Egmond aan Zee, The Netherlands.
- [3] Petra Aumann, H. Barnewitz, H. Schwarten, Klaus Becker, Ralf Heinrich, Britta Roll, M. Galle, Norbert Kroll, Thomas Gerhold, Dieter Schwamborn, and M. Franke. MEGAFLOW: Parallel complete aircraft cfd. *Parallel Computing*, 27(4):415–440, 2001.
- [4] T. J. Baker. Identification and preservation of surface features. In *Proceedings, 13th International Meshing Roundtable, Williamsburg, VA, Sandia National Laboratories, September 19-22 2004*, pages 299–310, 2004.
- [5] P. H. Cook, M. A. McDonald, and M. C. P. Firmin. Wind tunnel measurements of the mean flow in the turbulent boundary layer and wake in the region on the

- trailing edge of a swept wing at subsonic speeds. Technical report, RAE Tech. Rept. 79062, 1979.
- [6] J. R. Edwards and S. Chandra. Comparison of eddy viscosity-transport turbulence models for three-dimensional, shock separated flowfields. *AIAA Journal*, 34:756–763, 1996.
 - [7] G. Kalitzin, G. Medic, G. Iaccarino, and P. Durbin. Near-wall behaviour of RANS turbulence models and implications for wall functions. *Journal of Computational Physics*, 204:265–291, 2005.
 - [8] T. Knopp, Th. Alrutz, and D. Schwamborn. A grid and flow adaptive wall-function method for RANS turbulence modelling. *Journal of Computational Physics*, 220:19–40, 2006.
 - [9] F. R. Menter. Zonal two equation k/ω turbulence models for aerodynamic flows. *AIAA Paper 1993-2906*, 1993.
 - [10] Peter Niederdrenk. On the Control of Elliptic Grid Generation. In *Proceedings, 6th International Conference on Numerical Grid Generation in Computational Field Simulations, London, July 6-9, 1998*. ISBN 0-9651627-2-9.
 - [11] M. Vinokur. On 1-D Stretching Function for Finite-Difference Calculations. *Journal of Computational Physics*, 50:215, 1983.

(B)

**A grid and flow adaptive wall-function method
for RANS turbulence modelling**

Thomas Alrutz, Tobias Knopp, Dieter Schwamborn

*DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

Journal of Computational Physics,
Volume 220, Issue 1, Pages 19–40, 2006

Abstract

This paper presents a grid and flow adaptive wall-function method for RANS turbulence modelling with emphasis on aerodynamic flows. A near-wall grid adaptation technique ensures a locally appropriate resolution depending on both the near-wall flow physics to be captured and the range of validity of the wall-function model. The near-wall RANS solutions of the Spalart-Allmaras and SST $k-\omega$ turbulence model are investigated near stagnation points and subsequent not yet fully developed turbulent flow, and in regions of adverse pressure gradient before separation. These are compared with the corresponding turbulence model specific universal wall-functions and suggestions for the design of wall-function methods for non-equilibrium flows are given. Regions of non-equilibrium flow are detected by a flow based sensor and near-wall grid adaptation is then made possible due to the hybrid character of the wall-functions.

1 Introduction

In order to ensure both accuracy and efficiency of a simulation of complex flows, a computational method has to choose (*i*) an appropriate the level of complexity for physical/mathematical description and (*ii*) a numerical resolution adapted to the flow structures to be resolved and to the flow physics to be modelled. In the present paper, the two levels of flow description are hybrid universal wall functions and the full RANS & turbulence model equations. The numerical near-wall resolution depends on the near-wall flow structures to be resolved, on the range of validity of the wall-function model applied, and on the convergence acceleration for the numerical method aspired. Near-wall grid adaptation is made possible due to the hybrid character of the wall-function method.

The aim of *hybrid* (or *adaptive*) *wall-functions* is to provide a boundary condition at solid walls that enables flow solutions independent of the location of the first grid node above the wall, in particular concerning surface coefficients like pressure, skin friction and heat transfer. Denote $y^+(1)$ the distance of the first off-wall node in viscous length-scales. *Low-Re boundary conditions* impose no-slip at the wall and the RANS equations are integrated down to the wall, e.g., for the $k-\omega$, Spalart-Allmaras and $v^2 - f$ model, which requires a so-called *low-Reynolds grid* with $y^+(1) \approx 1$. *High-Re boundary conditions* are used in standard wall-function formulations and prescribe the wall-shear stress and no-penetration at the wall. The RANS equations are solved only down to the inner part of the logarithmic layer and matched with the logarithmic law of the wall at the first grid node above the wall. High-Re boundary conditions require a so-called *high-Reynolds grid* with $y^+(1)$ being located in the log-layer. Moreover we introduce the term *intermediate-Re* grid to refer to the region between viscous sublayer and log-layer, i.e., the buffer layer.

Wall functions are based on the fact that for incompressible flows the solution between the wall and the outer edge of the logarithmic layer is *universal* at least in quasi-equilibrium boundary layers. As the term "law of the wall" was originally used for the universal log-layer solution, the terms "hybrid law of the wall" or "adaptive law of the wall" have been proposed to stress validity in the entire near-wall region. The debate on whether the (hybrid) law of the wall does not vary in pressure gradients is

still open (see [9, 3] and references therein). Regarding the goal of grid-independent wall-functions, however, it is the *behaviour* of RANS turbulence models in flows with pressure gradients which is of major importance. In this paper very satisfying results are obtained without explicitly taking into account the pressure gradient.

Computational fluid dynamics (CFD) has been reaching more and more maturity as a general predictive tool in industry with "sensible" reliability. Despite the fast increase in available computing resources during the last decades, the huge computing costs are still a major limiting factor in the "appropriate" usage (in terms of the numerical discretization error) of CFD tools in industry, in particular for unsteady calculations. An additional need for improving performance arises as CFD-solvers are more and more used as part of optimization processes. This requires fast CFD-solutions for a large number of geometrical configurations without loss in accuracy.

Low-Re grids require a large number of grid points in the near-wall region. E.g., for an airfoil flow at Reynolds number 6×10^6 , a low-Re grid requires around 35 nodes in wall normal direction to resolve the boundary layer whereas a high-Re grid with $y^+(1) \approx 70$ can do with 17. Moreover numerical stiffness problems arise on low-Re grids due to the large velocity gradients at the wall in conjunction with the small cell height, which imply a very small pseudo time-step width and a large number of solver iterations. Moreover, the generation of intermediate- and high-Re grids is much simpler. As a final remark on the areas of relevance, wall-functions have been employed successfully as near-wall model for large-eddy simulation [29].

Albeit wall-functions are still strongly relevant in CFD and despite the more than thirty years enduring efforts in this topic, most of the current approaches e.g. [5, 20, 8, 27] seem to be suboptimal in the sense that their wall-functions are not consistent with the turbulence model used for the global flow problem. This causes a strong dependence of the numerical results on the location of the first off-wall grid node which leads to poor results in flows with separation. To the author's best knowledge the first paper which addressed and solved this issue of consistency at least for flows close to equilibrium is the one by [11].

In the present paper the focus is on non-equilibrium flows with emphasis on aerodynamics with (i) stagnation points and laminar/not fully developed turbulent flow, (ii) regions of significant pressure gradient parameter due to a strong (adverse) pressure gradient at a typically moderate Reynolds number and (iii) regions of separation and reattachment. These flow situations are very important not only for flows around airfoils and rotor blades, but even more for complex aircraft configurations with flaps, engines etc. For such flows, this paper proposes techniques for significant further improvement of the results in [16].

We restrict ourselves to flows in the subsonic and transonic regime. For high-speed flows at large Mach numbers with very large temperature differences near walls, the physics involved do not seem to be yet completely understood. Based on the pioneering work [26], compressible wall-functions have been proposed [31, 19] but require further improvement.

The high-Re grid condition is violated in separated flows due to strong adverse pressure gradient when pressure increases smoothly, e.g. for the flow over an airfoil in highlift configuration, the flow over a wind turbine or in a diffusor. As the separation point

is approached, $y^+(1)$ goes to zero as friction velocity is zero at the separation point. Hence a grid which is of high-Re type far upstream of separation violates more and more the high-Re constraint as the separation point is approached. Then classical high-Re wall-functions inevitably fail. For hybrid but inconsistent wall-functions [5, 8, 27] the grid-dependence causes a modelling error which accumulates as the separation point is approached and results in poor predictions.

On high-Re grids, the results for turbulence model consistent wall-functions still deviate from the corresponding low-Re solution in the flow situations (i)-(iii) mentioned above. As in these situations the wall-functions are still very close to the low-Re solution for $y^+(1) \lesssim 10$, we detect these regions using a flow-based sensor and use a $y^+(1)$ -grid-adaptation method to ensure locally a low-Re or intermediate-Re grid, which is allowed by the hybrid nature of consistent wall-functions.

This paper is organized as follows. In Section 2 the governing equations for compressible fluid flow and RANS turbulence modelling are given. In Section 3 the wall-function method is formulated as a domain decomposition with full overlap in the near-wall region. Universal wall-functions are proposed in a new closed form and the underlying modelling assumptions are investigated, see Sections 4-5. The combination of wall-functions and wall-normal grid adaptation with a flow-based sensor is presented in Section 6. In Section 7 the numerical method is described. Numerical results are given in Sections 8-9.

2 The governing equations of compressible turbulent fluid flow

2.1 RANS equations for compressible flows

We consider the steady-state Favre-averaged compressible Navier-Stokes equations in a bounded Lipschitz domain $\Omega \subset \mathbb{R}^d$ ($d = 2, 3$). We use the eddy-viscosity assumption for the Reynolds-stress tensor and the gradient-diffusion approximation for the turbulent heat-flux vector. The *low-Re formulation* reads as follows: We seek velocity $\vec{u} : \Omega \rightarrow \mathbb{R}^d$, density $\rho : \Omega \rightarrow \mathbb{R}$, pressure $p : \Omega \rightarrow \mathbb{R}$, and temperature $\theta : \Omega \rightarrow \mathbb{R}$ s.t.

$$\vec{\nabla} \cdot (\rho \vec{u}) = 0 \quad \text{in } \Omega, \quad (1)$$

$$\vec{\nabla} \cdot (\rho \vec{u} \otimes \vec{u}) - \vec{\nabla} \cdot [2\mu_e \mathbb{T}(\vec{u})] + \vec{\nabla} p = 0 \quad \text{in } \Omega, \quad (2)$$

$$\vec{\nabla} \cdot (\rho \vec{u} (h + \frac{1}{2} \vec{u} \cdot \vec{u})) - \vec{\nabla} \cdot [\vec{u} (2\mu_e \mathbb{T}(\vec{u}))] - \vec{\nabla} \cdot (\kappa_e \vec{\nabla} \theta) = 0 \quad \text{in } \Omega \quad (3)$$

with the following boundary conditions on solid walls Γ_w

$$\vec{u} = \vec{0} \quad \text{on } \Gamma_w \quad (4)$$

$$(i) \quad \kappa_e \vec{\nabla} \theta \cdot \vec{n} = 0 \quad \text{on } \Gamma_w \quad \text{or} \quad (ii) \quad \theta = \theta_w \quad \text{on } \Gamma_w \quad (5)$$

We use the Sutherland law for molecular viscosity μ and the equations of state $p = \rho R \theta$, $e = c_v \theta$ for specific internal energy, and $h = e + p/\rho = c_p \theta$ for specific enthalpy, with gas constant R , specific heat at constant volume c_v , specific heat at constant pressure

c_p , strain rate tensor $\mathbb{T}(\vec{u})$

$$\mathbb{T}(\vec{u}) \equiv \mathbb{S}(\vec{u}) - \frac{1}{3} \vec{\nabla} \cdot \vec{u} \mathbb{I}, \quad \text{with} \quad \mathbb{S}(\vec{u}) = \frac{1}{2} \left(\vec{\nabla} \vec{u} + (\vec{\nabla} \vec{u})^T \right),$$

effective viscosity $\mu_e = \mu + \mu_t$ and effective thermal conductivity $\kappa_e = \kappa + \kappa_t$ where $\kappa = c_p \mu / Pr$, $\kappa_t = c_p \mu_t / Pr_t$ with laminar and turbulent Prandtl numbers $Pr = 0.72$ and $Pr_t = 0.85$ resp.

2.2 Spalart-Allmaras turbulence model

The Spalart-Allmaras type one-equation turbulence models [23, 4] compute the eddy viscosity μ_t from the relation $\mu_t = \rho \nu_t$ with $\nu_t = f_{v1} \max(\tilde{\nu}; 0)$ where $\tilde{\nu}$ is the solution of the transport equation

$$\vec{\nabla} \cdot (\rho \vec{u} \tilde{\nu}) - \vec{\nabla} \cdot \left(\frac{\mu + \rho \tilde{\nu}}{\sigma} \vec{\nabla} \tilde{\nu} \right) - \rho \frac{c_{b2}}{\sigma} (\vec{\nabla} \tilde{\nu}) \cdot (\vec{\nabla} \tilde{\nu}) = c_{b1} \rho \tilde{S} \tilde{\nu} - c_{w1} \rho f_w \left(\frac{\tilde{\nu}}{d} \right)^2$$

with d being the distance to the closest wall and near-wall damping function $f_{v1} = \chi^3 / (\chi^3 + c_{v1}^3)$ with $\chi = \tilde{\nu} / \nu$. On walls $\tilde{\nu} = 0$ is prescribed.

2.3 Wilcox $k-\omega$ turbulence model

Several $k-\omega$ model versions (e.g. [17, 14, 21, 28]) have evolved from the original proposal [31] where $\mu_t = \rho k / \omega$ and k, ω are the solution of

$$\begin{aligned} \vec{\nabla} \cdot (\rho \vec{u} k) - \vec{\nabla} \cdot \left((\mu + \sigma_k \mu_t) \vec{\nabla} k \right) &= 2\mu_t \mathbb{T}(\vec{u}) : \vec{\nabla} \vec{u} - \beta_k \rho k \omega \\ \vec{\nabla} \cdot (\rho \vec{u} \omega) - \vec{\nabla} \cdot \left((\mu + \sigma_\omega \mu_t) \vec{\nabla} \omega \right) &= 2\gamma \rho \mathbb{T}(\vec{u}) : \vec{\nabla} \vec{u} - \beta_\omega \rho \omega^2 \end{aligned}$$

with constants $\beta_k, \beta_\omega, \gamma, \sigma_k$ and σ_ω . We impose $k = 0$ on solid walls. The solution for ω from asymptotic theory (cf. [31]) is given by $\omega = 6\nu / (\beta_\omega y^2)$ and becomes singular at the wall. In industrial RANS solvers, the boundary condition Equation (26) in [17] (abbreviated Menter b.c.) is very popular. Alternatively, Wilcox [31] suggests to prescribe ω at the first grid point above the wall Γ_δ at wall-distance y_δ located in the viscous sublayer (Wilcox b.c.):

$$\text{Menter b.c.: } \omega = C_w \omega_\delta \quad \text{on } \Gamma_w, \quad \text{with } \omega_\delta = \frac{6\nu}{\beta_\omega y_\delta^2}, \quad C_w = 10 \quad (6)$$

$$\text{Wilcox b.c.: } \omega = \omega_\delta \quad \text{on } \Gamma_\delta, \quad \text{with } \omega_\delta = \frac{6\nu}{\beta_\omega y_\delta^2}. \quad (7)$$

3 The wall-function concept

3.1 Domain decomposition method with full overlap in the near-wall region

To remedy the no-slip condition (4) instead of solving (1)-(5) in the computational domain Ω we consider a modified problem based on a domain-decomposition with full

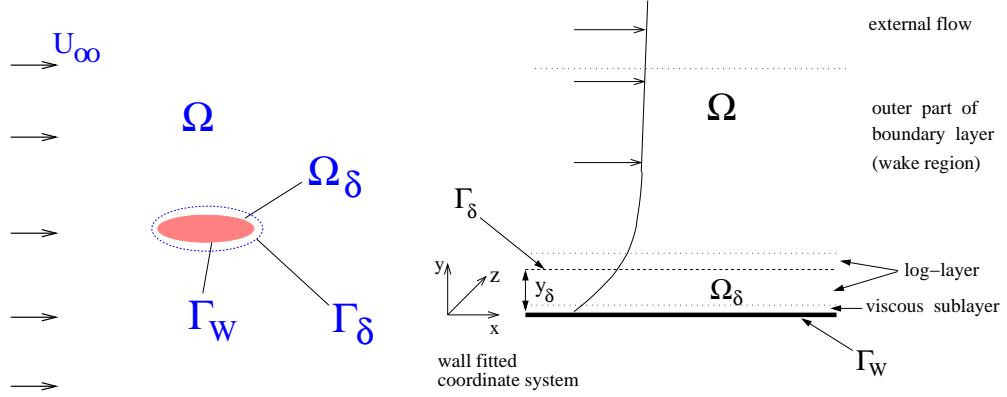


Figure 1: Domain decomposition with full overlap in the near-wall region.

overlap [25]. Denote $\Omega_\delta \subset \Omega$ the near-wall region (Fig. 1 (left)) with artificial inner boundary Γ_δ located within or below the log-layer (Fig. 1 (right)). Then we divide (1)-(5) into two problems: a *global flow* problem to be solved in the whole domain Ω with modified wall boundary condition (the wall-shear stress is imposed instead of no-slip) and a *boundary-layer problem* to be solved in the near-wall region Ω_δ , see also [25, 13]. Then the *wall-function or hybrid-Re formulation* reads as follows:
We seek a *global solution* $\rho : \Omega \rightarrow \mathbb{R}$, $\vec{u} : \Omega \rightarrow \mathbb{R}^d$, $p : \Omega \rightarrow \mathbb{R}$, $\theta : \Omega \rightarrow \mathbb{R}$ s.t.

$$\vec{\nabla} \cdot (\rho \vec{u}) = 0 \quad \text{in } \Omega, \quad (8)$$

$$\vec{\nabla} \cdot (\rho \vec{u} \otimes \vec{u}) - \vec{\nabla} \cdot (2\mu_e \mathbb{T}(\vec{u})) + \vec{\nabla} p = 0 \quad \text{in } \Omega, \quad (9)$$

$$\vec{\nabla} \cdot (\rho \vec{u} (h + \frac{1}{2} \vec{u} \cdot \vec{u})) - \vec{\nabla} \cdot [\vec{u} (2\mu_e \mathbb{T}(\vec{u}))] - \vec{\nabla} \cdot (\kappa_e \vec{\nabla} \theta) = 0 \quad \text{in } \Omega \quad (10)$$

with the following modified boundary conditions on Γ_w

$$\vec{u} \cdot \vec{n} = 0, \quad (\mathbb{I} - \vec{n} \otimes \vec{n}) 2\mu_e \mathbb{T}(\vec{u}) \vec{n} = -\tau_w(\vec{u}^{bl}, \rho^{bl}, \theta^{bl}) \vec{u}_t \quad \text{on } \Gamma_w \quad (11)$$

$$(i) \quad \kappa_e \vec{\nabla} \theta \cdot \vec{n} = 0 \quad \text{on } \Gamma_w \quad (12)$$

$$\text{or } (ii) \quad \kappa_e \vec{\nabla} \theta \cdot \vec{n} = -\dot{q}_w(\vec{u}^{bl}, \rho^{bl}, \theta^{bl}) \quad \text{on } \Gamma_w \quad (13)$$

where $\mathbb{I} - \vec{n} \otimes \vec{n}$ is a projection operator onto the tangential space of Γ_w and

$$\vec{u}_t = \frac{\vec{v}_t}{|\vec{v}_t|}, \quad \vec{v}_t = (\mathbb{I} - \vec{n} \otimes \vec{n}) \vec{u}|_{\Gamma_\delta} \quad \text{with} \quad (\mathbb{I} - \vec{n} \otimes \vec{n})_{ij} = \delta_{ij} - n_i n_j \quad (14)$$

with $\delta_{ij} = 1$ if $i = j$ and zero otherwise ($1 \leq i, j \leq d$). The wall-shear stress τ_w and the turbulent heat flux \dot{q}_w are calculated from

$$\tau_w(\vec{u}^{bl}, \rho^{bl}, \theta^{bl}) \vec{u}_t = -(\mathbb{I} - \vec{n} \otimes \vec{n}) 2\mu_e^{bl} \mathbb{T}(\vec{u}^{bl}) \vec{n} \quad \text{on } \Gamma_w \quad (15)$$

$$\dot{q}_w(\vec{u}^{bl}, \rho^{bl}, \theta^{bl}) = -\kappa_e^{bl} \vec{\nabla} \theta^{bl} \cdot \vec{n} \quad \text{on } \Gamma_w \quad (16)$$

where the *boundary layer solution* $\vec{u}^{\text{bl}} : \Omega_\delta \rightarrow \mathbb{R}^d$, $\rho^{\text{bl}} : \Omega_\delta \rightarrow \mathbb{R}$, $p^{\text{bl}} : \Omega_\delta \rightarrow \mathbb{R}$, and $\theta^{\text{bl}} : \Omega_\delta \rightarrow \mathbb{R}$ is determined by

$$\vec{\nabla} \cdot (\rho^{\text{bl}} \vec{u}^{\text{bl}}) = 0 \quad \text{in } \Omega_\delta, \quad (17)$$

$$\vec{\nabla} \cdot (\rho^{\text{bl}} \vec{u}^{\text{bl}} \otimes \vec{u}^{\text{bl}}) - \vec{\nabla} \cdot \left(2\mu_e^{\text{bl}} \mathbb{T}(\vec{u}^{\text{bl}}) \right) + \vec{\nabla} p^{\text{bl}} = 0 \quad \text{in } \Omega_\delta, \quad (18)$$

$$\begin{aligned} & \vec{\nabla} \cdot \left(\rho^{\text{bl}} \vec{u}^{\text{bl}} \left(h^{\text{bl}} + \frac{1}{2} \vec{u}^{\text{bl}} \cdot \vec{u}^{\text{bl}} \right) \right) \\ & - \vec{\nabla} \cdot \left[\vec{u}^{\text{bl}} \left(2\mu_e^{\text{bl}} \mathbb{T}(\vec{u}^{\text{bl}}) \right) \right] - \vec{\nabla} \cdot \left(\kappa_e^{\text{bl}} \vec{\nabla} \theta^{\text{bl}} \right) = 0 \end{aligned} \quad \text{in } \Omega_\delta \quad (19)$$

with the boundary conditions

$$\vec{u}^{\text{bl}} = \vec{0} \quad \text{on } \Gamma_w \quad (20)$$

$$(i) \quad \kappa_e^{\text{bl}} \vec{\nabla} \theta^{\text{bl}} \cdot \vec{n} = 0 \quad \text{on } \Gamma_w \quad \text{or} \quad (ii) \quad \theta^{\text{bl}} = \theta_w \quad \text{on } \Gamma_w \quad (21)$$

$$\theta^{\text{bl}} = \theta, \quad \vec{u}^{\text{bl}} = \vec{u} \quad \text{on } \Gamma_\delta \quad (22)$$

The boundary layer solution satisfies the original boundary conditions (4), (5) on Γ_w and is matched with the global solution on Γ_δ .

A wall-function method is called *grid independent* (or *hybrid* or *adaptive*) if (i) the method is well-defined for any location of Γ_δ and (ii) the solution of (8)-(22) is independent of the location of Γ_δ (provided Γ_δ resides in the log-layer or below). In particular, this definition implies grid-independence of surface quantities like pressure coefficient c_p and skin-friction coefficient c_f .

In practice, the aim is to avoid solving the full compressible RANS plus turbulence model equations (17)-(22) in the near-wall region and hence standard boundary layer approximations are used.

3.2 Boundary-layer approximation for universal wall functions

Numerical tests show that effects of compressibility in the near-wall region are negligible for Mach numbers smaller 1.4. Moreover, the near-wall attached flow is already surprisingly well described by the one-dimensional boundary layer equations, except very close to flow separation and reattachment, where a two-dimensional boundary layer model is superior.

Then, instead of (17)-(22) in Ω_δ , for each $\vec{x}_W \in \Gamma_W$ and given $u_\delta = \|\vec{v}_t\|$ from the global RANS solution by (14), seek the wall-parallel component of velocity $u^{\text{bl}}(y)$ such that

$$\frac{d}{dy} \left((\nu + \nu_t^{\text{bl}}) \frac{du^{\text{bl}}}{dy} \right) = f \quad \text{in} \quad \{ \vec{x}_W - y\vec{n} \mid y \in (0, y_\delta) \} \quad (23)$$

$$u^{\text{bl}}(0) = 0, \quad u^{\text{bl}}(y_\delta) = u_\delta \equiv \|\vec{v}_t\| \quad (24)$$

where $f = 0$ or $f = 1/\rho dp/dx$ assumed to be independent of y and given from the global RANS solution at Γ_δ . Therein, denote $y_\delta = \text{dist}(\vec{x}_W, \Gamma_\delta)$. The variant $f = 0$ is called equilibrium stress balance model leading to universal near-wall solutions.

It is well-known that for equilibrium boundary layers, e.g., the flow over a flat plate

at zero pressure gradient, in the region between the wall and the outer edge of the logarithmic layer, the profiles for mean flow u and turbulence quantities k , ω , $\tilde{\nu}$ and hence ν_t are universal, i.e., they collapse when scaled with friction velocity u_τ and viscosity $\nu = \mu/\rho$

$$u^+ = \frac{u}{u_\tau}, \quad y^+ = \frac{yu_\tau}{\nu}, \quad \nu_t^+ = \frac{\nu_t}{\nu}, \quad p^+ = \frac{\nu}{\rho u_\tau^3} \frac{dp}{dx}, \quad k^+ = \frac{k}{u_\tau^2}, \quad \omega^+ = \frac{\omega\nu}{u_\tau^2} \quad (25)$$

These universal near-wall profiles may be obtained by integration of (23) with $f = 0$ and the corresponding 1D boundary-layer equations for k and ω resp. $\tilde{\nu}$.

Equation (23) in plus-units reveals that it is the pressure gradient parameter p^+ which controls the validity of the equilibrium stress balance assumption

$$(1 + \nu_t^+) \frac{du^+}{dy^+} = 1 + p^+ y^+ \quad \text{in } (0, y_\delta^+) \quad (26)$$

4 Validation of the boundary layer approximation for aerodynamic flows

In this section we investigate the near-wall behaviour of the SA-E and SST $k-\omega$ model in the following two important flow situations of aerodynamic flows, viz., in an adverse pressure gradient boundary layer flow (typically on the upper side of an airfoil) and close to the stagnation point (at the leading edge of wing or fuselage).

4.1 Validation for adverse pressure gradient flow

In this section we assess the range of validity of the boundary layer approximation (23) with $f = 0$ for two adverse pressure gradient (APG) flows. Firstly, we consider the flow over a flat plate at $Re = 4.1 \times 10^7$ devised by [11]. We use a flat plate of length $L = 8\text{m}$ with farfield data $u_\infty = 78\text{ms}^{-1}$ and $\nu_\infty = 1.5 \times 10^{-5}\text{m}^2\text{s}^{-1}$. At distance $y = 0.5\text{m}$ above the wall, suction and blowing is imposed by prescribing the wall-normal velocity component $v(x) = A \exp(-b(x - x_\alpha)^2) - A \exp(-b(x - x_\beta)^2)$ with $x_\alpha = 2.5$, $x_\beta = 5.5$, $A = 0.35x$, $b = 108/6^2$ which produces a streamwise pressure gradient leading to separation. Secondly, we study the flow around the "A-airfoil" (AS239) in highlift configuration for $Ma = 0.15$ at three different $Re = 2.0 \times 10^6, 1.0 \times 10^7, 4.0 \times 10^7$, and different angle of attack $\alpha = 10.2^\circ, 13.3^\circ, 14.2^\circ, 15.3^\circ$. We investigate the near-wall behaviour of the SA-E model and the SST $k-\omega$ model.

First we recall that for fixed APG, say fixed $\alpha = 13.3^\circ$, p^+ is decreasing for increasing Re due to its definition, see Figure 2 (left). As the separation point is approached, p^+ goes to infinity. Regarding the test cases under investigation, for the SA-E model, the region where (26) has universal (i.e., Re-number independent) solutions depending only on the parameter p^+ , is relatively large, see Figure 2 (right). Therein, the A-airfoil for $\alpha = 13.3^\circ$, $Re = 2 \times 10^6$ and the zero-pressure gradient flat plate solution (ZPG) are also shown.

Interestingly, SA-E and SST $k-\omega$ model show a different behaviour in APG flow. We focus on the flat plate APG flow, which has a p^+ behaviour close before separation similar to the A-airfoil at $Re = 4 \times 10^7$. We consider the velocity profiles $u^+(y^+)$ at

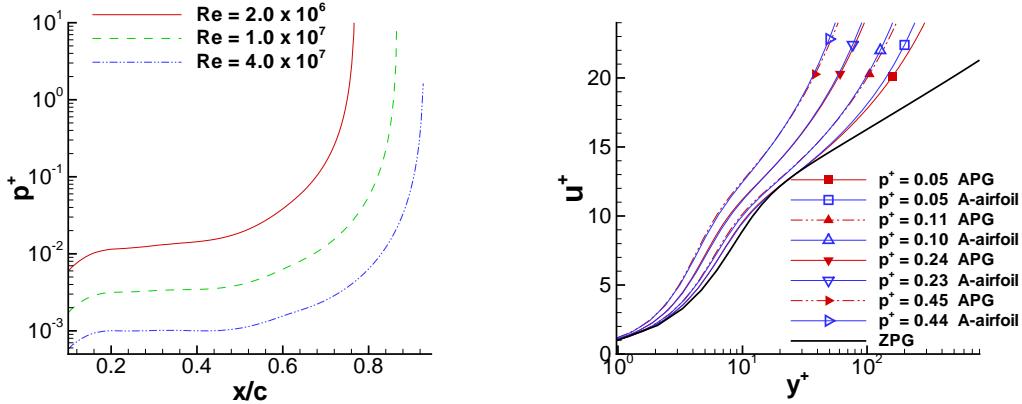


Figure 2: Reynolds number dependence of p^+ (left) and universal p^+ -dependent near-wall behaviour (right) for SA-E model.

different p^+ -stations. The SST $k-\omega$ model shows a general breakdown of the universal ZPG solution, i.e., the character of the velocity profile changes over the entire buffer layer ($5 < y^+ < 50$) and log-layer ($y^+ > 50$), see Figure 3. However, in the viscous sublayer ($y^+ < 5$) agreement with the universal ZPG solution is still very close, except at large p^+ -values very close before separation.

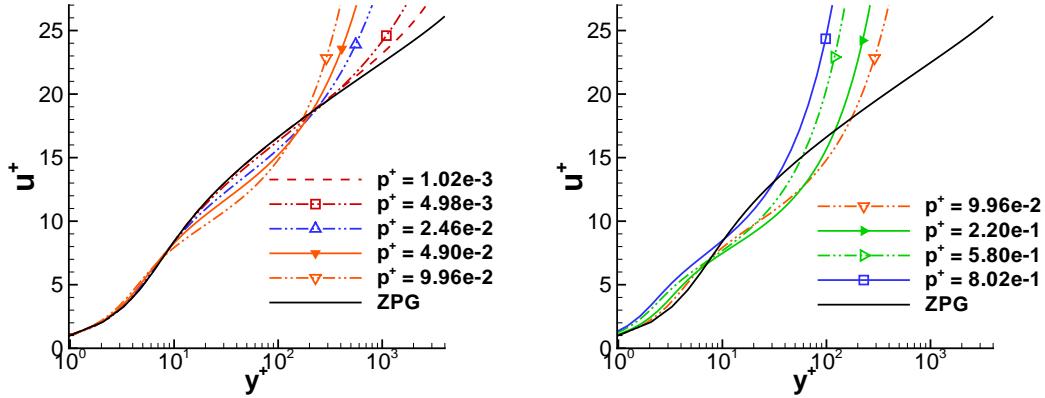


Figure 3: Near-wall velocity profile for SST $k-\omega$ model at different p^+ -stations for APG flow [11].

On the other hand, the SA-E model shows a successive breakdown of the universal ZPG solution, i.e., the region occupied by the universal ZPG solution in reduced progressively, see Figure 4 (left).

Now we consider the turbulence quantities. For the SA-E model, \tilde{v}^+ shows a much larger deviation from the universal ZPG solution than would be expected from the behaviour of the u^+ -profiles, see Figure 4 (right).

Figure 5 shows the profiles for k^+ and ω^+ . For ω a progressive reduction of the region

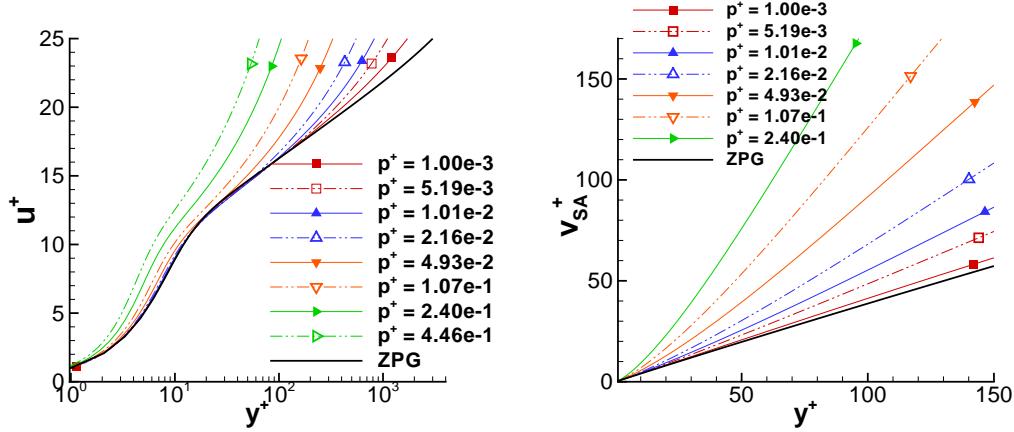


Figure 4: u^+ and ν^+ for SA-E model at different p^+ -stations for the APG flow [11].

of validity of the universal ZPG solution is observed. For k , a general breakdown of the ZPG solution in the *entire* near-wall region can be seen. This causes the general breakdown of the universal ZPG solution for u^+ .

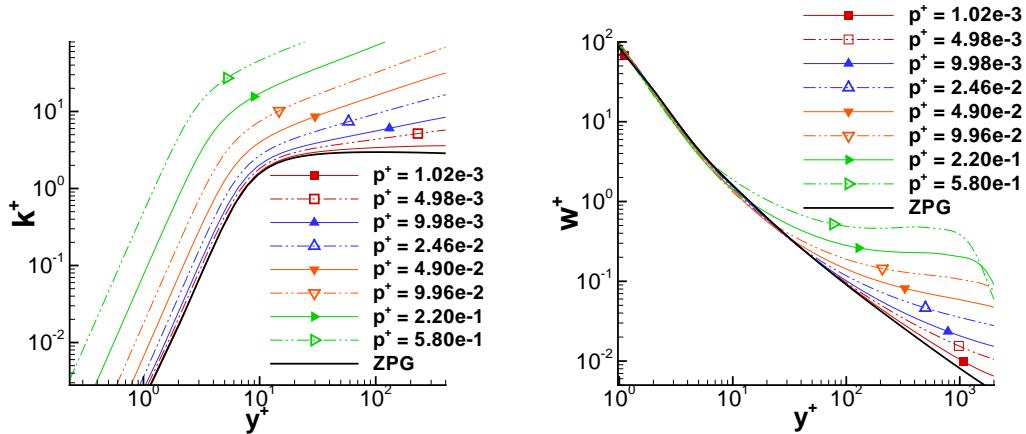


Figure 5: Profiles for k^+ and ω^+ at different p^+ -stations for APG flow [11].

Finally we study the profiles for eddy-viscosity. The deviation from the universal ZPG solution is large in the entire near wall region even for small p^+ and increasing rapidly with increasing p^+ , see Figure 6.

4.2 Near-wall behaviour of RANS solutions close to an airfoil leading edge

In this subsection, we study the near-wall behaviour of the SA-E and SST $k-\omega$ model close to the leading edge of the "A-airfoil" at $Ma = 0.15$, $Re = 2.0 \times 10^6$, and angle of attack $\alpha = 13.3^\circ$. This flow is far from being fully turbulent due to the close distance to the stagnation point at $x/c = 0.03$ (with chord length c). Moreover the flow is affected

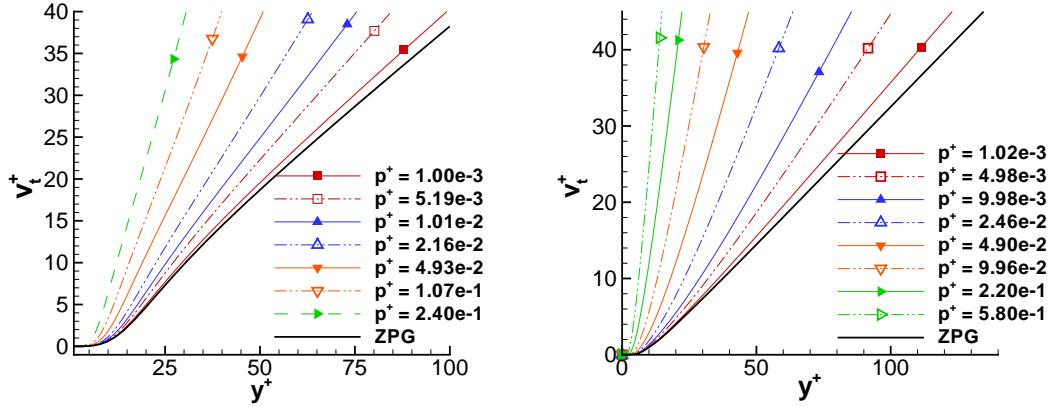


Figure 6: Eddy viscosity for SA-E model (left) and SST $k-\omega$ model (right) for APC flow [11].

by the strong variation of p^+ together with the large mean-streamline curvature. Figure 7 shows the p^+ -distribution for two airfoil flows. Note that p^+ is positive for adverse pressure gradients and negative for favourable pressure gradients. Due to its definition, p^+ goes to infinity as the stagnation and separation points are approached.

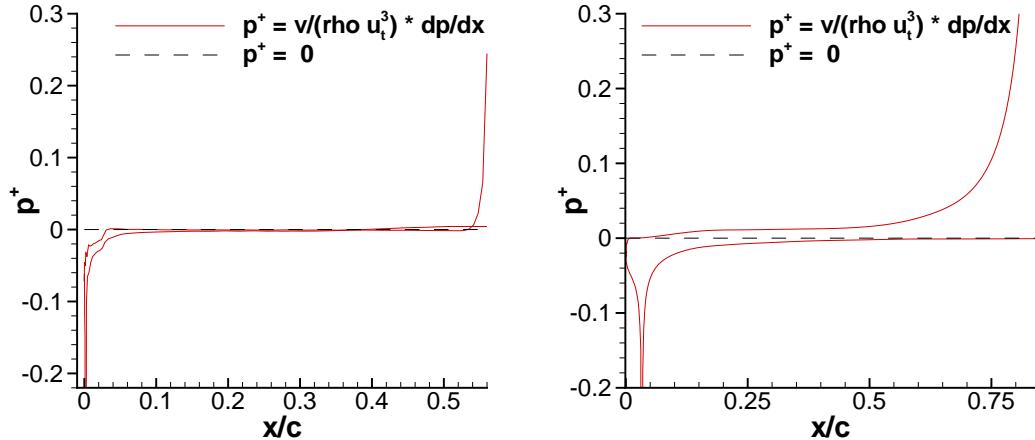


Figure 7: Pressure gradient p^+ for RAE-2822 case 10 (left) and A-airfoil (right).

We consider the solution at six stations $x_i = \xi_i c$ with $\xi_1 = 0.15$, $\xi_2 = 0.2$, $\xi_3 = 0.25$ on the lower side characterized by strong flow acceleration and $\xi_4 = 0.0003$, $\xi_5 = 0.0019$, $\xi_6 = 0.0044$ on the upper side, with corresponding p^+ -values of order 10^{-2} . Regarding the velocity profiles, Figure 8 shows that for $y^+ \lesssim 10$, the agreement with the universal ZPG solution is very good for both SA-E and SST model. However, for the SA-E model, for increasing values of y^+ , deviations from the ZPG solution become large. The turbulence quantities are shown in Figure 9. The profiles for ω are close to the universal ZPG-solution for $y^+ \lesssim 30$. However, it is noteworthy that the near-wall

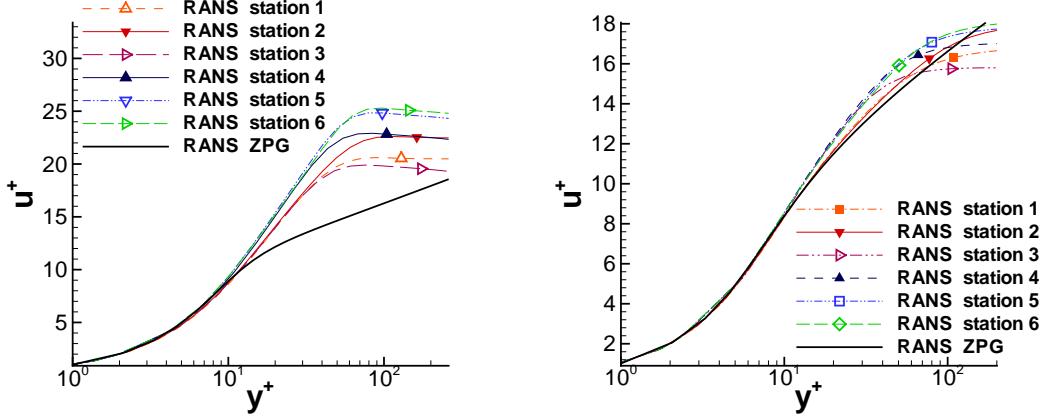


Figure 8: Profiles of u^+ around the leading edge of A-airfoil for SA-E (left) and SST $k-\omega$ (right).

solution for $\tilde{\nu}$ is far from its universal ZPG-profile.

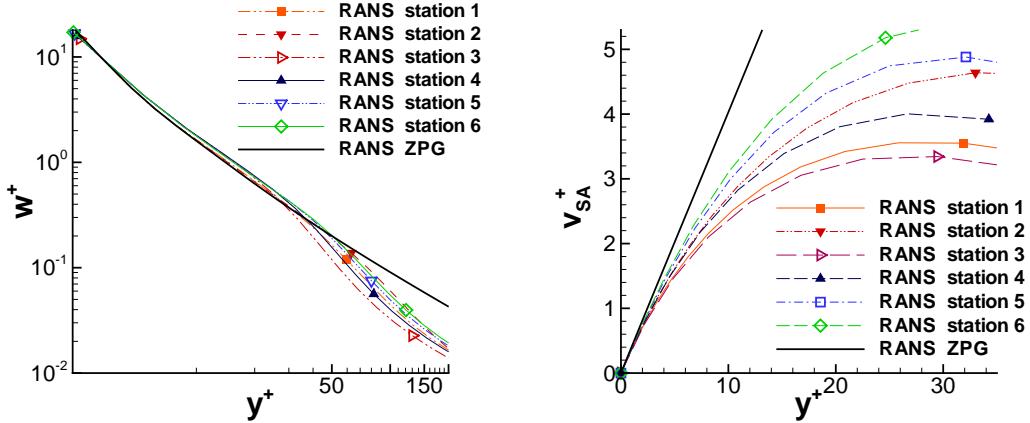


Figure 9: Profile for ω for SST model and $\tilde{\nu}$ for SA-E model at leading edge of A-airfoil.

4.3 Implications for wall-functions and motivation for near-wall grid adaptation

In this subsection we draw some implications for a suitable usage of universal wall-functions for non-equilibrium flows. In this paper, we are interested in flows with (i) stagnation points and subsequent not yet fully developed turbulent flow, (ii) regions of adverse pressure gradient with relatively large pressure gradient parameter, and (iii) regions of separation and reattachment.

The universal ZPG wall-functions for u and ω are still close to the RANS solution in the viscous sublayer even in regions (i) and (ii) provided p^+ is of at most moderate size. Moreover, it was shown in [11] pp.285, that in regions of separated flow and after

reattachment the universal profiles are still close to the wall-resolved RANS solution in the viscous sublayer.

From Figure 8 we infer that close to stagnation points, large first spacings $y^+(1)$ should be avoided, in particular for the SA model.

From Figures 4, 5, 6, and 9 we see that the near-wall RANS solutions for k , $\tilde{\nu}$ and hence ν_t show large deviations from their universal ZPG profiles in situations (i) and (ii). Thus we may conclude that off-wall boundary conditions using the universal ZPG profile for k and $\tilde{\nu}$ are sources of grid-dependent solutions and should be avoided. Instead homogeneous Dirichlet conditions are recommended, see Section 7.

As suggested by Figures 6 and 9 (right) we do not invoke $\nu_t^+ = dy^+/du^+ - 1$, which is relation (26) with neglecting the p^+ -term, as a consistency relation between u^+ and ν_t^+ , because this may be an extra source of grid-dependence in non-equilibrium situations. The profiles of ν_t^+ change much larger than would be expected from the u^+ -profiles, which is due to neglecting the p^+ -term. Note that this is also supported by an a posteriori analysis. For this purpose, we integrated (26) numerically at different p^+ stations where the corresponding profile for ν_t^+ is taken from a spline interpolation of the RANS solution. Integration is performed with and without the p^+ -term. Good agreement is found when the p^+ -term is taken into account but poor otherwise.

Finally we study the issue of large p^+ -values close before separation. For large $Re \gtrsim 10^7$, p^+ is small even close before separation and universal ZPG wall functions give almost grid independent results, as shown in [11] p.284. But for moderate Re -numbers, say $Re = 2 \times 10^6$ this term becomes important. With regard to the application of wall-functions, the fact that $u_\tau \rightarrow 0$ as separation is approached has two counteracting effects: The unfavourable effect is that close before separation p^+ becomes large such that the universal ZPG profiles for u and ω cease to be valid outside the viscous sublayer. The favourable point is that the thickness of the viscous sublayer in dimensional units is becoming larger, i.e., points at a given wall-distance located in the log-layer upstream of the separation point now may reside in the viscous sublayer.

The latter effect is illustrated in Figure 10 (left). Therein, we consider two wall-parallel grid-lines and study their y^+ distribution on the upper side of the A-airfoil at $\alpha = 13.3^\circ$. The grid-points initially reside in the log-layer. As the separation point is approached, their y^+ -values move toward the viscous sublayer.

This effect can be enforced by applying a wall-normal grid adaptation which shifts the nodes closer to the wall, see Figure 10 (right). Moreover, using such a grid adaptation, suitable values of $y^+(1)$ can be ensured also in flow regions (i) and (iii). This will be described in detail in Section 6.

5 Turbulence model consistent universal wall functions

In boundary layer flows at ZPG, the condition on turbulence model consistency $\nu_t^{\text{bl}} = \nu_t$ implies that model specific wall-functions are required. As the near-wall profiles of different versions of the Spalart-Allmaras model resp. the $k-\omega$ model almost collapse [12], it is sufficient to specify one model-consistent universal wall-function for the Spalart-

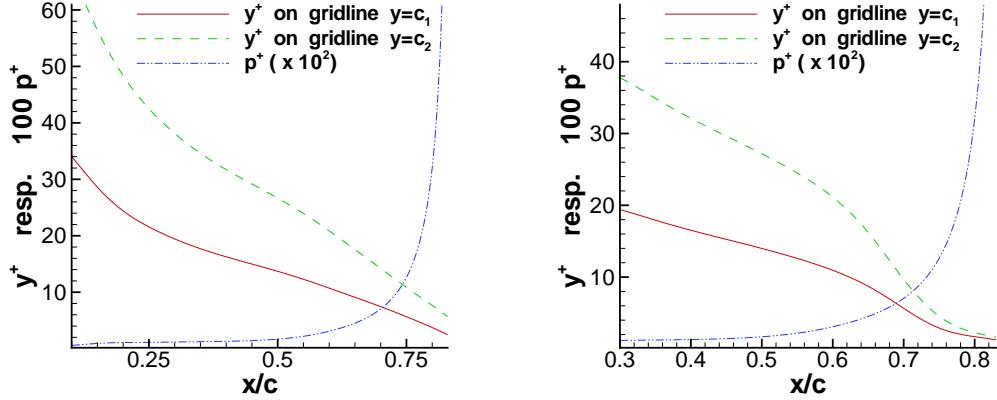


Figure 10: Wall-distance in plus-units for two wall-parallel grid-lines in APC flow around A-airfoil without wall-normal grid adaptation (left) and with adaptation (right).

Allmaras model, denoted F_{SA} , and one for the $k-\omega$ model, referred to as $F_{k\omega}$.

$$F_{SA} = (1 - \phi_{SA})F_{Sp,5} + \phi_{SA}F_{Rei,m}, \quad \phi_{SA} = \tanh(\arg^3), \quad \arg = y^+/24 \quad (27)$$

$$F_{k\omega} = (1 - \phi_{k\omega})F_{Sp,3} + \phi_{k\omega}F_{Rei,m}, \quad \phi_{k\omega} = \tanh(\arg^2), \quad \arg = y^+/50 \quad (28)$$

which are plotted in Figure 11. Therein, we use Reichardt's law of the wall

$$u^+ = F_{Rei}(y^+) , \quad F_{Rei}(y^+) \equiv \frac{\ln(1 + 0.4y^+)}{\kappa} + 7.8 \left(1 - e^{-\frac{y^+}{11.0}} - \frac{y^+}{11.0} e^{-\frac{y^+}{3.0}} \right) \quad (29)$$

with $\kappa = 0.41$. We apply the following blending with the classical log-law $F_{log} = y^+/\kappa + 5.1$, viz.,

$$F_{Rei,m} = (1 - \phi_{b1})F_{Rei} + \phi_{b1}F_{log} , \quad \phi_{b1} = \tanh(\arg^4) , \quad \arg = y^+/27 . \quad (30)$$

Spalding's law [24] with parameter $N \in \{3, 4, 5\}$ is given by the inverse formula

$$y^+ = F_{Sp,N}^{-1}(u^+) , \quad F_{Sp,N}^{-1}(u^+) \equiv u^+ + e^{-\kappa 5.2} \left(e^{\kappa u^+} - \sum_{n=0}^N \frac{(\kappa u^+)^n}{n!} \right) . \quad (31)$$

Of crucial importance is the near-wall solution for ω . Instead of the standard blending [27], a new proposal avoids the deviation of ω from its low-Re solution in the buffer layer, viz.,

$$\text{Standard blending: } \omega = \sqrt{\omega_{vis}^2 + \omega_{log}^2} , \quad (32)$$

$$\text{New proposal: } \omega = \phi \omega_{b1} + (1 - \phi) \omega_{b2} , \quad \phi = \tanh(\arg^4) , \quad \arg = \frac{y^+}{10} \quad (33)$$

with the following blending formula and the asymptotic relations

$$\omega_{b1} = \omega_{vis} + \omega_{log} , \quad \omega_{b2} = (\omega_{vis}^{1.2} + \omega_{log}^{1.2})^{1/1.2} , \quad \omega_{vis} = \frac{6\nu}{\beta_\omega y^2} , \quad \omega_{log} = \frac{u_\tau}{\sqrt{\beta_k \kappa y}} . \quad (34)$$

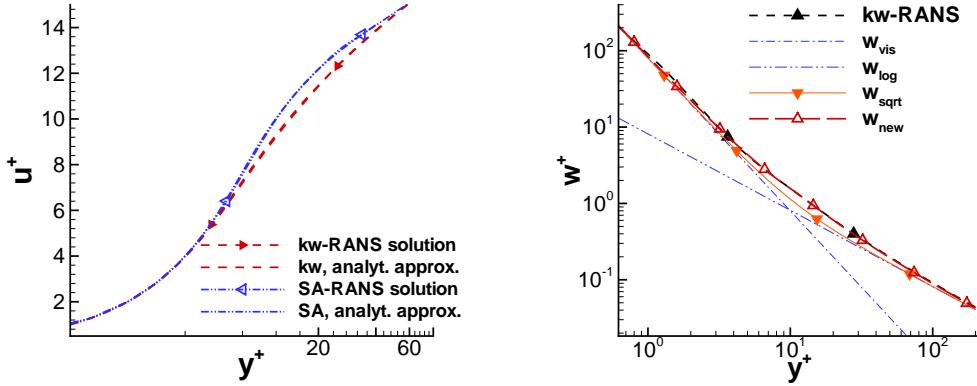


Figure 11: New wall-functions for SA- resp. $k-\omega$ type models (left), wall law for ω (right).

In [11] a cubic spline interpolation of the near-wall RANS solution instead of closed formula are used. However, the aim of the present approach is to also allow for an easy dissemination of the method to other CFD codes.

5.1 Iterative solution strategy

Suppose a solution of (23) with $f = 0$ is known in either of the two closed forms

$$u^+ = F(y^+) \Leftrightarrow \frac{u}{u_\tau} = F\left(\frac{yu_\tau}{\nu}\right) \quad \text{or} \quad y^+ = F^{-1}(u^+) \Leftrightarrow \frac{yu_\tau}{\nu} = F^{-1}\left(\frac{u}{u_\tau}\right) \quad (35)$$

then the matching condition $u^{bl} = u_\delta$ on Γ_δ and the relation $u^{bl} = u_\tau F(yu_\tau/\nu)$ imply

$$F\left(\frac{y_\delta u_\tau}{\nu}\right) = \frac{u_\delta}{u_\tau} \quad \text{resp.} \quad F^{-1}\left(\frac{u_\delta}{u_\tau}\right) = \frac{y_\delta u_\tau}{\nu} \quad (36)$$

which can be solved for u_τ using Newton's method.

Remark 1. *Newton's method for the direct form $u^+ = F(y^+)$ requires about three to four iteration steps independent of y^+ . For the inverse formula $y^+ = F^{-1}(u^+)$ convergence problems can be observed for $y^+(1) \gtrsim 300$ since Newton's method for the inverse formula is ill-conditioned. Thus the direct formulation should be used for large y^+ -values.*

Denote $TM \in \{\text{SA}, \text{k}\omega\}$ and $N \in \{3, 5\}$. For the numerical solution of

$$\frac{u_\delta}{u_\tau} = F_{TM}\left(\frac{y_\delta u_\tau}{\nu}\right), \quad F_{TM} = (1 - \phi_{TM})F_{S,N} + \phi_{TM}F_{Rei,m}$$

we proceed as follows:

1. From the initial guess $u_\tau^0 = u_\delta/y_\delta$, seek $u_{\tau,Rei}$ as solution of $u_\delta/u_\tau = F_{Rei,m}(y_\delta u_\tau/\nu)$.

2. Using the initial guess $u_\tau^0 = u_{\tau,Rei}$, seek $u_{\tau,S}$ as solution of $y_\delta u_\tau / \nu = F_{S,N}^{-1}(u_\delta/u_\tau)$.
3. Compute ϕ_{TM} and set $u_\tau = (1 - \phi_{TM})u_{\tau,S} + \phi_{TM}u_{\tau,Rei}$.

In our experience, steps (1) and (2) require each three to four iteration steps for convergence. The global hybrid-Re problem (8)-(16) with the wall function model (36) is solved iteratively until convergence is reached. Given an initial global flow solution, we can compute $\tau_w = \rho u_\tau^2$ from (36). This provides the boundary condition on Γ_w , and the new global flow solution can be computed.

6 Wall-normal grid adaptation

We may conclude from Section 4 that in the following flow situations the deviation of the universal ZPG solution from the low-Re RANS solution is becoming large as y^+ is increased: (a) stagnation points with subsequent not fully developed flow, (b) strong pressure gradients ($p^+ \gtrsim 0.05$), and (c) regions of separated flow. We point out that even if the pressure gradient parameter would be taken into the wall-function method such that grid independent results for pressure induced separation could be obtained, the problems near the leading edge and in regions of separated and reattached flow would still limit application to complex aerodynamic flows. Grid-adaptation is a solution strategy for all situations (a)-(c).

Thus, as a remedy, a *near-wall grid adaptation* [1] is employed. We assume that inside the prismatic layer, the nodes are located on rays starting at the corresponding wall node. We use the following notation, see Figure 12 (left):

- \vec{x}_{wp} : Surface (wall) node,
- \vec{x}_{np} : First node above the wall corresponding to node wp,
- $\{\vec{x}_{wp} + \lambda_p \vec{r}\}$: Ray of points starting at wall node \vec{x}_{wp} and ending at the tetrahedral layer; $\{\vec{x}_{wp} + \lambda_p \vec{r}\} \equiv \{\vec{x} \in \mathbb{R}^d \mid \vec{x} = \vec{x}_{wp} + \lambda_p \vec{r}, 0 \leq p \leq p_{\max}\}$ where the direction vector \vec{r} may be non constant.

Moreover we assume that $\vec{x}_{np} - \vec{x}_{wp}$ is almost parallel to the surface normal vector \vec{n} . Then the algorithm for y^+ -adaptation may be written as follows:

1. Read RANS solution and grid.
2. y^+ grid adaptation.
 - (a) For each surface node \vec{x}_{wp} do:
 - i. Determine \vec{x}_{np} .
 - ii. Calculate u_τ using (29). From the wall-distance $y_{np} = |\vec{x}_{np} - \vec{x}_{wp}|$ determine the wall-distance in plus units y_{np}^+ .
 - iii. Check if \vec{x}_{wp} is located in a region of flow stagnation and strong surface curvature.
 - iv. Determine p^+ from (25) and check if $|p^+| > p_0^+$ for a given threshold p_0^+ .
 - v. Check if point \vec{x}_{wp} resides in a separation region.

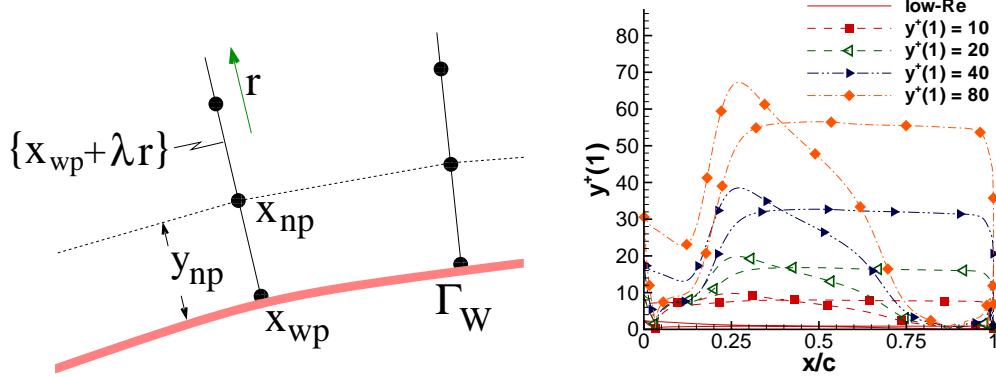


Figure 12: Left: Notation for near-wall grid adaptation. Right: $y^+(1)$ for A-airfoil on adapted grid.

- vi. Based on (2(a)iii)-(2(a)v) set target value y_{target}^+ .
- vii. If $y_{target}^+ < y_{np}^+$ then set $y_{new} = y_{np}^+ y_{target}^+ / y_{np}^+$, else $y_{new} = y_{np}$.
- (b) Smooth the y_{new} -distribution.
- (c) For each surface node redistribute the points on its ray $\{\vec{x}_{wp} + \lambda_p \vec{r}\}$ where the last point $\vec{x}_{wp} + \lambda_{p,max} \vec{r}$ remains unchanged.

3. Interpolate RANS solution from old grid to new grid.

Some technical details are described in the following. Calculating u_τ using (29) is sufficient for the adaptation. We use the threshold value $p_0^+ = 0.09$ for indicating regions of strong pressure gradient. Concerning the target value for y^+ , numerical tests suggest $y_{target}^+ \in [5, 10]$ in regions of flow stagnation and strong surface curvature and $y_{target}^+ \in [1, 5]$ in regions of separation.

Smoothing of the y_{new} -distribution is performed as follows. Denote K the number of smoothing steps, $y_{np}^i = y_{new}$ of node \vec{x}_{np}^i after (2(a)vii), $\mathcal{N}(i)$ the set of indices of neighbour (adjacent) surface nodes of node i and $\#\mathcal{N}(i)$ their number. Then in smoothing step k

$$y_{np}^{i,k} = (1 - \epsilon) y_{np}^{i,k-1} + \epsilon y_{np}^{\text{nei},k-1} \quad \text{with} \quad y_{np}^{\text{nei},k-1} = \frac{1}{\#\mathcal{N}(i)} \sum_{j \in \mathcal{N}(i)} y_{np}^{j,k-1}.$$

The wall-normal grid adaptation can also be used to increase $y^+(1)$ in regions of attached boundary layer flow close to equilibrium, where p^+ is small and almost constant. A typical area of application is the fuselage of an airfoil. This can be used to accelerate the convergence of the flow solver.

7 Discretization using an explicit finite volume method

Computations are performed using the DLR TAU-code, which is of cell-vertex type, i.e., of cell-centered type w.r.t. the dual grid cells. The convective fluxes are calculated

by a central scheme with artificial scalar dissipation [15]. The gradients of the flow variables are reconstructed using a Green-Gauss-MUSCL formula. The arising fixed-point problem is iterated in fictitious pseudo-time using a low-storage k-stage Runge-Kutta scheme by Jameson [10].

In the present method, boundary cells are half cells in the sense that the corresponding grid nodes are located on the boundary. This allows to impose homogeneous Dirichlet boundary conditions for k and $\tilde{\nu}$. Concerning the hybrid wall-function boundary condition, the standard approach for cell-centered type FVM to simply modify μ_t in the boundary cell (see [24] Equation (9a), (10a)), cannot be used. Therefore, the fluxes across the wall are prescribed using (11), (13) with the approximative relation

$$\mathbb{T} \cdot \vec{n} = (\mathbb{I} - \vec{n} \otimes \vec{n}) \mathbb{T} \cdot \vec{n} + \vec{n} \otimes \vec{n} \mathbb{T} \cdot \vec{n} \approx -\tau_w \vec{u}_t,$$

where the normal contribution of the wall-shear stress is neglected.

The global hybrid-Re problem (8)-(16) with the wall function model (36) is solved iteratively in pseudo time until convergence is reached.

8 Numerical results

8.1 Flat plate turbulent boundary layer with zero pressure gradient

The ability of the wall function proposal described in Section 5 to give solutions almost independent of the wall-normal grid spacing in zero-pressure gradient flows is demonstrated. We consider the turbulent boundary layer flow at zero pressure gradient over a flat plate of length $l = 5\text{m}$ studied experimentally by Wieghardt and recorded in [30] as Flow 1400. In agreement with the experimental setup we use $u_\infty = 33\text{m/s}$, $\nu = 1.51 \times 10^{-5}\text{m}^2/\text{s}$ and an adiabatic wall. Transition from laminar to turbulent flow is prescribed in agreement with the formula for the critical Reynolds number where transition occurs (cf. [22], p.471).

The grid-dependence using the classical wall laws (31) and (29) is shown in Figure 13 (left) by plotting the relative error in skin friction $c_f = \tau_w/(0.5\rho_\infty u_\infty^2)$ in the interval $2 \leq x \leq 4.5$ between the low-Re solution c_f^{LR} and the solution with wall-functions c_f^{WF} for the SA-Edwards model, where $y^+(1)$ denotes the wall distance of the first node above the wall in viscous units.

Figures (13) (right) and (14) show the almost grid independent results for the new wall functions. In order to obtain grid independent results for $k-\omega$ type models, it is crucial to use (7) and (33) for the ω boundary condition.

As a final remark, method (28), (7), (33) gives almost grid-independent predictions for all versions of the $k-\omega$ model considered, including non-linear variants and explicit algebraic Reynolds-stress modells, see Fig. 15. The variation in c_f due to a variation in $y^+(1)$ is significantly smaller than the different predictions among the various $k-\omega$ model versions.

8.2 Transonic airfoil flows RAE-2822 cases 9 and 10

In this section we apply the method to the transonic airfoil flows RAE-2822 case 9 (no/small separation region at $Ma = 0.73$, $Re = 6.5 \times 10^6$ and angle of attack $\alpha = 2.8^\circ$)

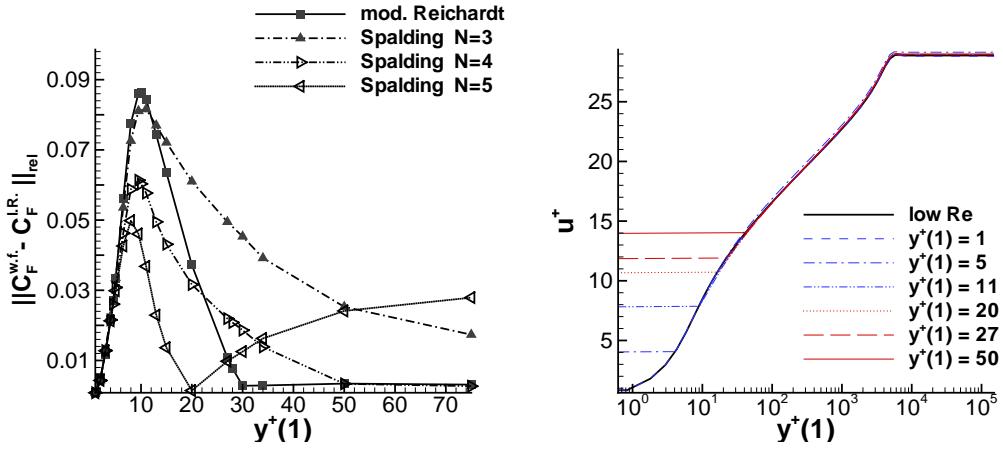


Figure 13: Grid-dependence of classical wall-functions (left), new velocity results for $k-\omega$ model (right).

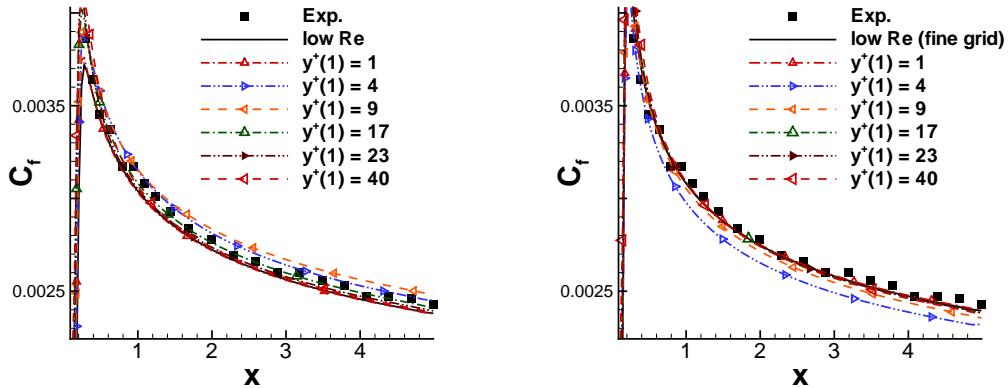


Figure 14: Prediction of c_f for SA-E model (left) and for baseline $k-\omega$ model [17] (right).

and case 10 (shock induced separation at $Ma = 0.75$, $Re = 6.2 \times 10^6$ and $\alpha = 2.8^\circ$) investigated experimentally in [2]. We consider a series of hybrid-Re grids of O-type with $y^+(1)$ varying from one to sixty, generated using the commercial grid generation tool CentaurSoft (www.centaursoft.com). Fig.16 shows the $y^+(1)$ -distribution for the SST $k-\omega$ model. The grids are built such that the thickness of the prismatic layer has an almost constant value around $0.052c$ (with chord length c) which fully contains the boundary layer. It is worthwhile pointing out that the generation of grids with $y^+(1) \gtrsim 8$ is much simpler than for low-Re grids.

We consider the predictions for the pressure coefficient c_p and the local skin friction coefficient $c_f^{(loc)}$ (based on the local dynamic pressure at the boundary layer edge q_P , [2]). Preliminary calculations show that prescribing transition at $x/c = 0.03$ according to the experiment or treating the wall as fully turbulent has only a small influence which is restricted to the region $x/c \leq 0.05$.

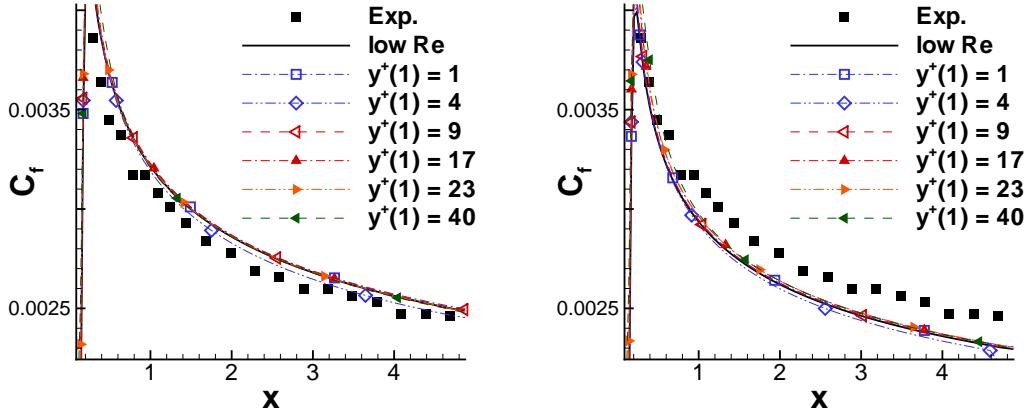


Figure 15: Application to EARSM $k-\omega$ models [21] (left) and [28] (right).

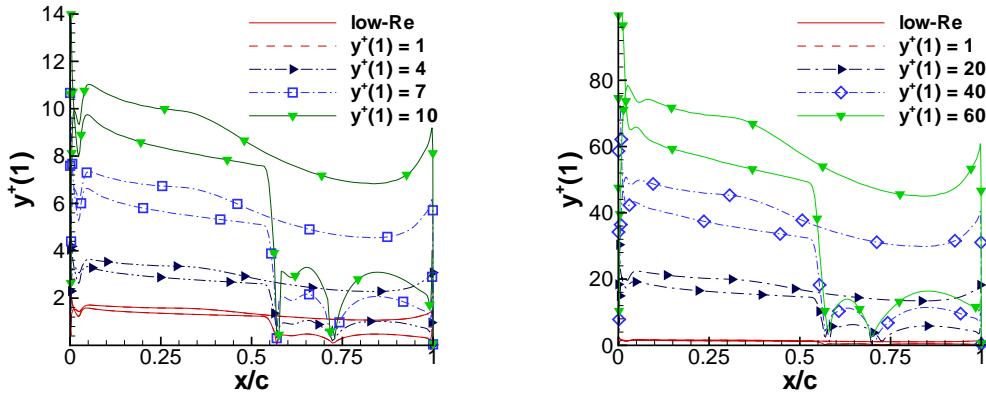
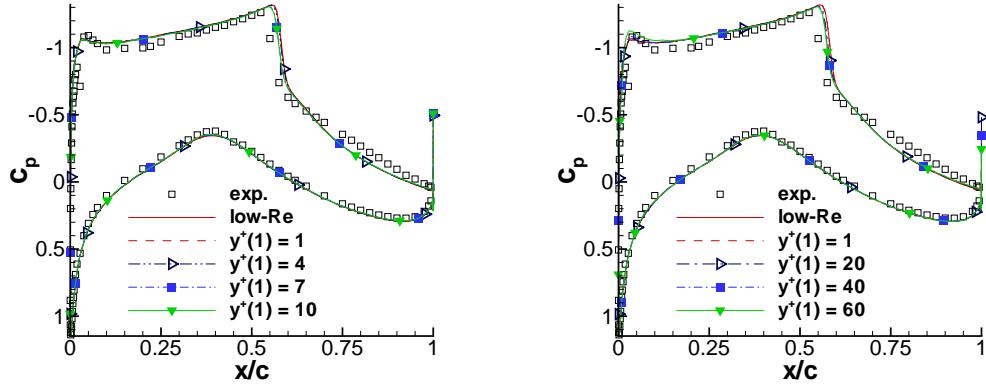
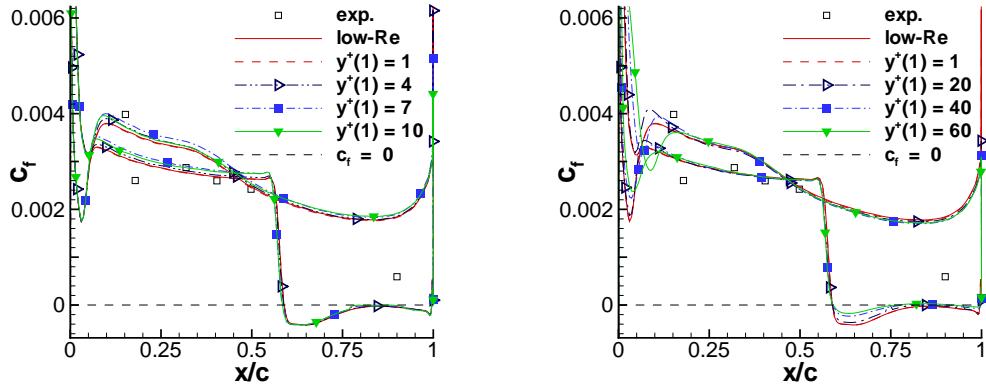


Figure 16: RAE case 10: Distribution of $y^+(1)$ for SST $k-\omega$ model [17].

8.2.1 Transonic airfoil flow RAE-2822 case 10

The results for the SA-E model are given in Figures 17-18. The agreement in c_p on the various grids is very good. On the coarser grids with $y^+(1) \gtrsim 40$, around the leading edge the deviation is small in c_p but discernible in c_f . On the upper side in the fully turbulent region before the shock ($x/c \lesssim 0.5$), the behaviour in c_f is similar to the flow over a flat plate: The predictions for $y^+(1) = 1$ and $y^+(1) \gtrsim 20$ almost collapse whereas in the intermediate region of $y^+(1)$ the deviation is slightly larger. In the separation region, the agreement on intermediate-Re grids with $y^+(1) \lesssim 10$ is surprisingly good, whereas on the coarser grids $y^+(1) \gtrsim 20$ the differences become larger.

The results for the SST $k-\omega$ model are plotted in Figures 19-20. We mention that (7) is superior to (6) in giving grid-independent results for high-Re grids $y^+(1) \gtrsim 40$, whereas for $y^+(1) \lesssim 10$ both boundary conditions are almost equal. The deviations around the leading edge are similar to the SA-E model. In the fully turbulent region on the upper side before the shock, there are moderate deviations in c_f on the intermediate-Re and

Figure 17: RAE case 10: Distribution of c_p for the SA-E model.Figure 18: RAE case 10: Distribution of c_f for the SA-E model.

high-Re grids.

8.2.2 Transonic airfoil flow RAE-2822 case 9

For case 9, the SA-E model predicts a slight shock-induced separation for the low-Re b.c. and for $y^+(1) \lesssim 20$ (with hybrid-Re b.c.), whereas for $y^+(1) \gtrsim 40$ no shock-induced separation is predicted, see Figure 21. In all cases, the flow remains attached near the trailing edge. It should be noted that on a similar low-Re grid of H-type, after reattachment a slight trailing edge separation is predicted. This stresses the influence of the grid topology on the results.

Regarding the SST $k-\omega$ model, on all grids except for $y^+(1) = 60$ a slight shock induced separation is predicted in agreement with the low-Re result, see Figure 22. In all cases the flow remains attached near the trailing edge.

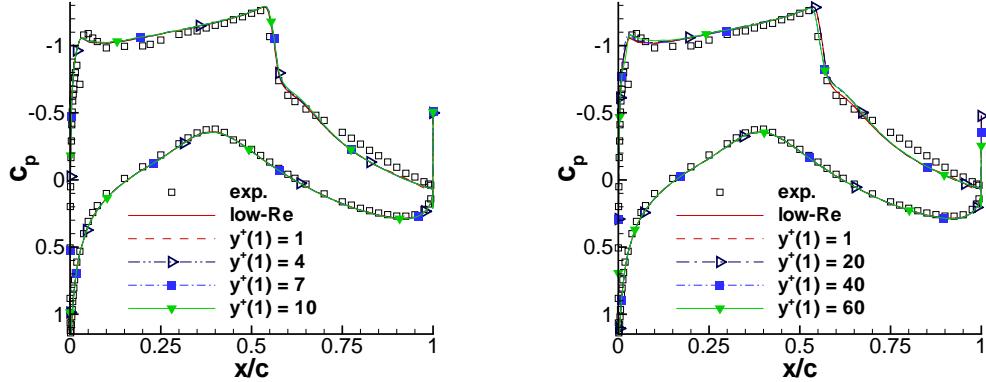


Figure 19: RAE case 10: Distribution of c_p for SST $k-\omega$ model with Wilcox b.c.

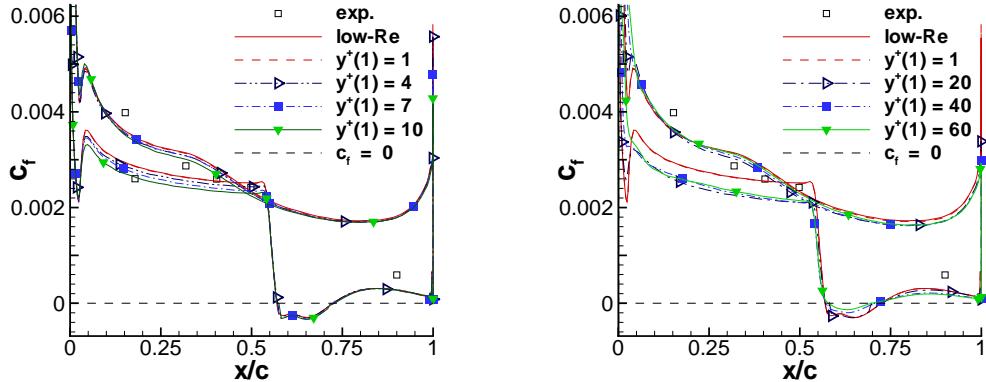
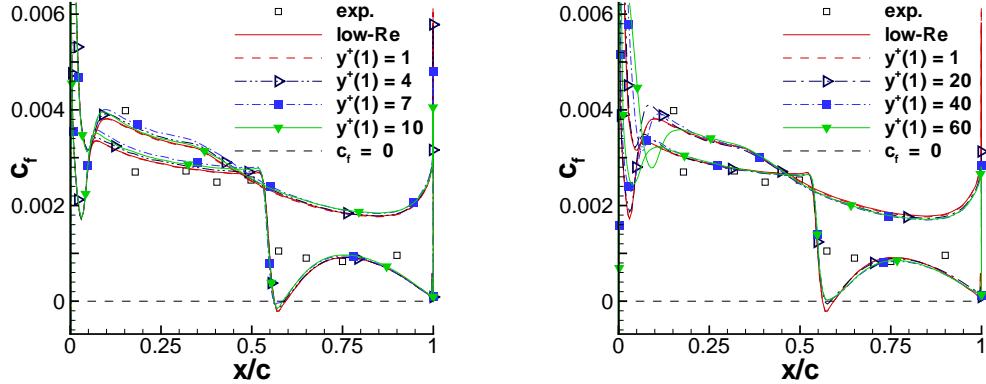
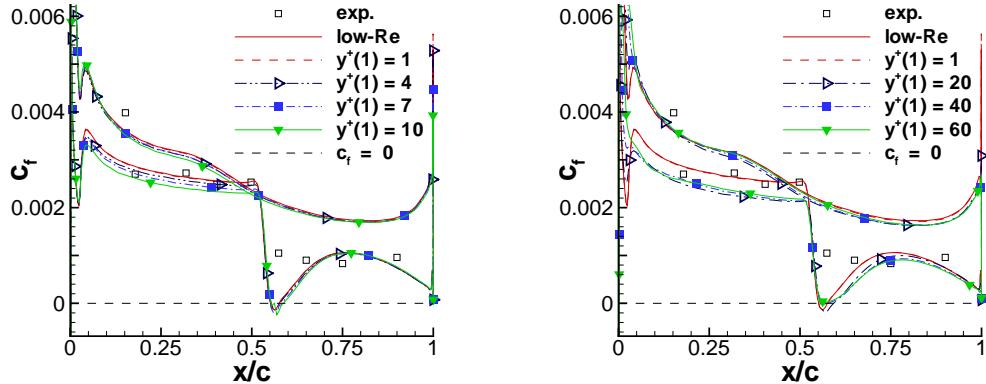


Figure 20: RAE case 10: Prediction of c_f for SST $k-\omega$ model with Wilcox b.c.

8.3 Subsonic A-airfoil in highlift configuration

In this section we apply the wall-function method to the subsonic flow around the "A-airfoil" (AS239) in highlift configuration at $Ma = 0.15$, $Re = 2.0 \times 10^6$, and angle of attack $\alpha = 13.3^\circ$, studied experimentally in [6, 7]. The strong adverse pressure gradient on the upper side causes the turbulent boundary layer to separate close to the trailing edge. In the experiment, transition was prescribed at $x/c = 0.3$ on the lower side and free transition was observed at $x/c = 0.12$ on the upper side.

However, in the present computations the airfoil surface is treated fully turbulent. The laminar boundary layer region is relatively large for the A-airfoil in the experiment and the present wall-function model relies on a fully-turbulent boundary-layer relation for the wall-shear stress. As the focus is on grid-independence of the wall-function method, the question of laminar-turbulent transition modelling is not considered here. Note that neglecting transition increases the deviation from the experimental data significantly. The O-type grids with their $y^+(1)$ -distribution are shown in Fig. 23, where the denoting value has to be seen as an average over the chord length. This test case is very

Figure 21: RAE case 9: Distribution of c_f for the SA-E model.Figure 22: RAE case 9: Distribution of c_f for the SST $k-\omega$ model with Wilcox b.c.

interesting and challenging for wall-functions, as $y^+(1)$ ceases from its maximal value near the leading edge to zero at the separation point.

The results for the SA-E model on the grids with $y^+(1) \leq 10$ are shown in Figure 24 and show close agreement with the low-Re solution. The two solutions with $y^+(1) = 40$ and $y^+(1) = 80$ suffer from local oscillations in c_p near the leading edge and are not shown here. This can be remedied by ensuring smaller values for $y^+(1)$ near the leading edge (e.g., by using a grid adaptation) or by using a modified boundary condition for $\tilde{\nu}$. Regarding the latter, we can prescribe $\tilde{\nu} = \tilde{\nu}_{cs}$ on Γ_δ , where $\tilde{\nu}_{cs}$ is the spline interpolation of the universal near-wall solution for $\tilde{\nu}$ [11], but then results become inferior on intermediate-Re grids, see [12].

For the SST $k-\omega$ model with Wilcox b.c. (7) (see Figure 25), on all grids the agreement in c_f with the low-Re solution is remarkably good, in particular for $y^+(1) \lesssim 10$. We remark that the results for the SST $k-\omega$ model with boundary condition (6) show a discernible underprediction for c_f on the high-Re grids, see [12].

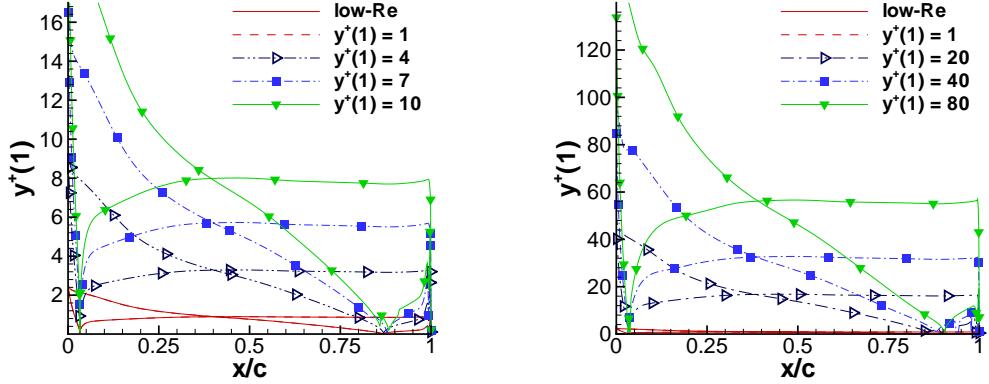


Figure 23: A-airfoil: Distribution of $y^+(1)$ for the SST $k-\omega$ model.

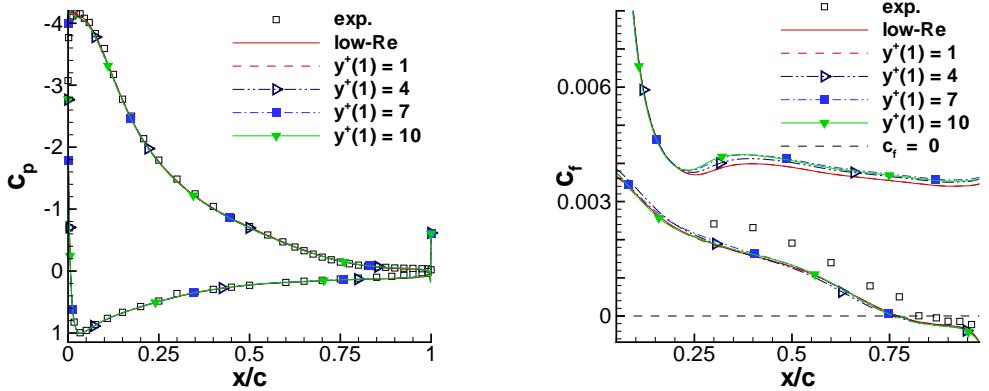


Figure 24: A-airfoil: Prediction for c_p (left) and c_f (right) for SA-E model.

9 Combination of wall-normal grid adaptation and wall-functions

In this section we apply the grid-adaptation technique described in Section 6 to the two test cases considered in the previous section. For the A-airfoil we use $\epsilon = 0.6$ and $K = 200$. Figure 12 (right) shows the y^+ -distribution on the adapted grids. Figure 26 shows that the grid-independence of the c_f -distribution is improved noticeably.

For the SA-E model, near-wall grid adaptation around the leading edge is necessary for ensuring stability of the method on the high-Re grids, as for large y^+ -values the velocity profiles are too far from the universal ZPG solution, recall Figure 8. Further tests show that limiting $y^+(1)$, e.g., $y^+(1) \lesssim 50$ around the leading edge is sufficient to ensure stability, i.e., to avoid oscillations. This can be the method of choice if large $y^+(1)$ -values are desired in order to obtain a large convergence acceleration.

Table 1 gives the separation point without and with y^+ -adaptation. Therein N_y denotes the number of wall normal nodes in the prismatic layer. Values denoted by $(\cdot)^*$ indicate

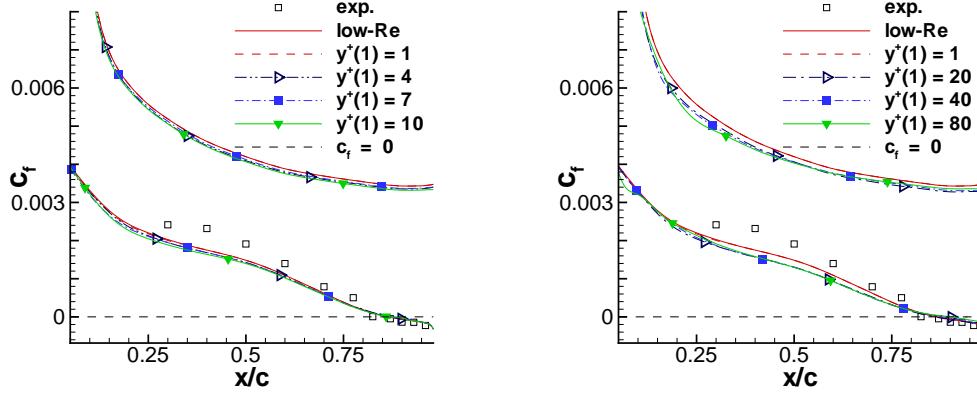


Figure 25: A-airfoil: Prediction for c_f for SST $k-\omega$ model with Wilcox b.c. (7).

that the solution suffers from strong oscillations near the leading edge.

The SST $k-\omega$ model predicts a too late separation point on the high-Re meshes without near-wall adaptation. This stems from the influence of significant p^+ -values on the near-wall velocity profile over a relatively large streamwise distance due to the moderate Reynolds-number, recall Figures 3, 10.

Finally, it is worthwhile pointing out that the grid-dependence is much smaller than the spreading due to the uncertainty associated with the turbulence model.

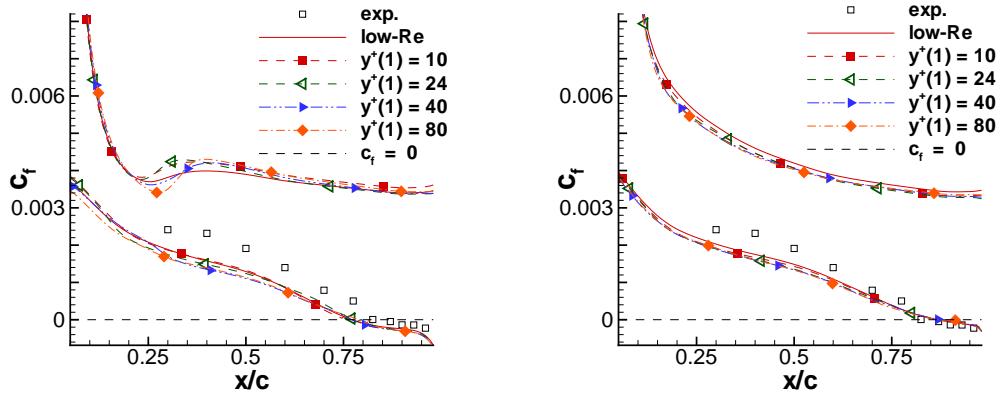


Figure 26: A-airfoil: c_f after y^+ -adaptation for SAE (left) and SST $k-\omega$ (right).

Secondly we apply the approach to the RAE 2822 case 10. Here we are interested in a better resolution of the leading edge and the aft-shock separation region. Both regions are critical to obtain correct moments e.g. for aeroelastic applications. Figure 27 (left) shows that the wall-normal grid is shifted towards the low-Re regime in the vicinity of the leading edge, close to the shock and in the separation region. The improvement of grid-independence in c_p (Figure 27 (right)) and in c_f in the separation region (Figure 28) is discernible. In particular, the prediction for the reattachment point is almost grid-independent.

$y^+(1)$ (average)	N_y	x_{sep}/c SA-E	x_{sep}/c SA-E (y^+ -adap)	x_{sep}/c SST $k-\omega$	x_{sep}/c SST $k-\omega$ (y^+ -adap)
low-Re	33	0.771	—	0.866	—
1	33	0.770	—	0.866	—
4	28	0.755	—	0.863	—
7	26	0.759	—	0.868	—
10	24	0.761	—	0.861	—
20	21	0.787	0.776	0.861	0.864
40	19	(0.838)*	0.771	0.881	0.873
80	17	(0.881)*	0.788	0.903	0.867

Table 1: Prediction of separation point for A-airfoil without and with $y^+(1)$ adaptation. Values denoted by $()^*$ suffer from local oscillations at the leading edge.

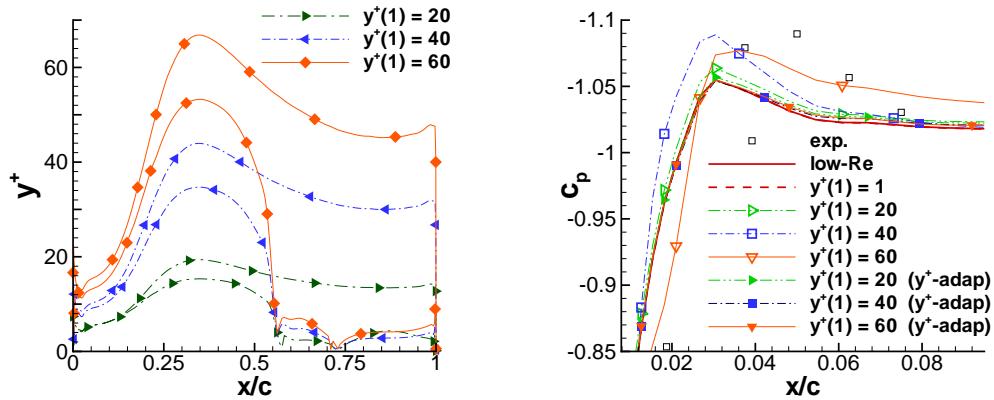


Figure 27: RAE case 10 on adapted grid: $y^+(1)$ (left) and detail of c_p for SST $k-\omega$ (right).

10 Conclusions

A new grid and flow adaptive wall-function strategy has been presented. This strategy consists of two parts, namely, a hybrid wall-function proposal suitable for non-equilibrium flows and a near-wall grid adaptation method with a flow-based sensor. Based on an investigation of the near-wall behaviour of the SA-E and the SST $k-\omega$ model in non-equilibrium flow situations, improvements of the wall-function method for applications to aerodynamic flows are suggested, most notably to avoid off-wall boundary conditions for turbulent kinetic energy and eddy viscosity using their universal equilibrium solutions.

A new combination of wall-functions and near-wall grid adaptation ensures both an appropriate resolution of near-wall flow physics and takes into account the range of validity and stability of the wall-function model.

The treatment of laminar flow regions and the question of transition from laminar to turbulent flow is subject of future research. Moreover, future work will focus on applications to unsteady flow problems. It is also intended to apply this method as a

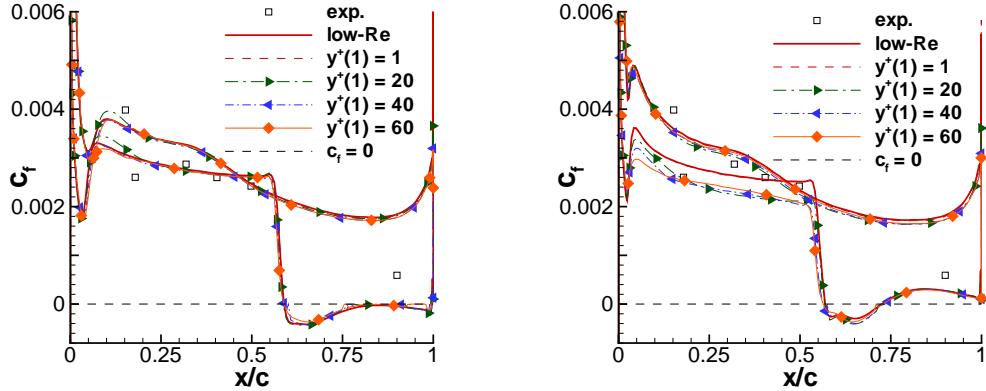


Figure 28: RAE case 10 on adapted grid for SA-E (left) and SST $k-\omega$ (right).

new near-wall model for large-eddy simulation and to a new class of so-called scale adaptive models [18], which are in between standard unsteady RANS and detached-eddy simulation. These methods require three-dimensional unsteady calculations using a very small time step and the aim is to make these methods amenable to complex configurations at affordable costs.

Acknowledgement

The author is grateful to Drs. Keith Weinman, Bernhard Eisfeld, Matthias Orlt, Volker Hannemann, Ante Soda, Stefan Melber-Wilkending, Gert Lube, Michael Leschziner and Cord C. Rossow for valuable discussions and to Dr. Christian Gleyzes for informations regarding the A-airfoil. Moreover the valuable suggestions of the reviewers are gratefully acknowledged.

References

- [1] Th. Alrutz. Hybrid grid adaptation in TAU. In N. Kroll and J. K. Fassbender, editors, *MEGAFLOW - Numerical flow simulation for aircraft design, Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, volume 89, pages 109–115, Braunschweig, 2005.
- [2] P. H. Cook, M. A. McDonald, and M. C. P. Firmin. Aerofoil RAE 2822 - Pressure distributions and boundary layer and wake measurements. Technical report, AGARD Advisory Report AR-138, pp. A6.1-A6.77, 1979.
- [3] P. A. Durbin and S. E. Belcher. Scaling of adverse-pressure-gradient boundary layers. *J. Fluid Mech.*, 238:699–722, 1992.
- [4] J. R. Edwards and S. Chandra. Comparison of eddy viscosity-transport turbulence models for three-dimensional, shock separated flowfields. *AIAA J.*, 34:756–763, 1996.

- [5] N. T. Frink. Tetrahedral unstructured Navier-Stokes method for turbulent flows. *AIAA J.*, 36:1975–1982, 1998.
- [6] Ch. Gleyzes. Opération décrochage - Résultats des essais à la soufflerie F2. Technical report, RT-DERAT 55/4004 DN, ONERA, 1988.
- [7] Ch. Gleyzes. Opération décrochage - Résultats de la 2ème campagne d'essais à F2 - Mesures de pression et vélocimétrie laser. Technical report, RT-DERAT 55/5004 DN, ONERA, 1989.
- [8] E. Goncalves and R. Houdeville. Reassessment of the wall functions approach for RANS computations. *Aerosp. Sci. Technol.*, 5:1–14, 2001.
- [9] P. S. Granville. A modified van Driest formula for the mixing length of turbulent boundary layers in pressure gradients. *J. Fluids Eng.*, 111:94–97, 1989.
- [10] A. Jameson. Transonic flow calculations. Technical report, MAE Report 1651, Princeton University, 1983.
- [11] G. Kalitzin, G. Medic, G. Iaccarino, and P. Durbin. Near-wall behaviour of RANS turbulence models and implication for wall functions. *J. Comput. Phys.*, 204:265–291, 2005.
- [12] T. Knopp. A new adaptive wall-function method for subsonic and transonic turbulent flows. Technical report, DLR IB 224-2005 A 14, 2005.
- [13] T. Knopp, G. Lube, R. Gritzki, and M. Rösler. A near-wall strategy for buoyancy-affected turbulent flows using stabilized FEM with applications to indoor air flow simulation. *Comput. Methods Appl. Mech. Eng.*, 194:3797–3816, 2005.
- [14] J. C. Kok. Resolving the dependence on freestream values for the k/ω turbulence model. *AIAA J.*, 38:1292–1295, 2000.
- [15] D. J. Mavriplis, A. Jameson, and L. Martinelli. Multigrid solution of the Navier-Stokes equations on triangular meshes. Technical report, ICASE-report No. 89-35, 1989.
- [16] G. Medic, G. Kalitzin, G. Iaccarino, and E. van der Weide. Adaptive wall functions with applications. *AIAA Paper (submitted)*, 2005.
- [17] F. R. Menter. Zonal two equation k/ω turbulence models for aerodynamic flows. *AIAA Paper 1993-2906*, 1993.
- [18] F. R. Menter, M. Kuntz, and R. Bender. A scale-adaptive simulation model for turbulent flow predictions. *AIAA Paper 2003-0767*, 2003.
- [19] B. Mohammadi and G. Puigt. Wall-laws for high speed flows over adiabatic and isothermal walls. Technical report, INRIA, Rapport de recherche no. 3948, 2000.
- [20] T. Rung. Formulation of universal wall boundary conditions for transport equation turbulence models. Technical report, Hermann-Föttinger-Institut, Technische Universität Berlin, 1999.

- [21] T. Rung, H. Lübecke, M. Franke, L. Xue, F. Thiele, and S. Fu. Assessment of explicit algebraic stress models in transonic flows. In *Engineering Turbulence Modelling and Experiments 4, Proc. 4th Int. Symposium on engineering Turbulence Modelling and Measurements, Corsica, France*, pages 659–668, 1999.
- [22] H. Schlichting. *Boundary-Layer Theory*. McGraw-Hill, New York, 1979.
- [23] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamics flows. *AIAA Paper 1992-0439*, 1992.
- [24] D. B. Spalding. A single formula for the law of the wall. *J. Appl. Mech.*, 28:455–457, 1961.
- [25] M. D. Tidriri. Domain decomposition for compressible Navier-Stokes equations with different discretizations and formulations. *J. Comput. Phys.*, 119:271–282, 1995.
- [26] E. R. van Driest. Turbulent boundary layer in compressible fluids. *J. Aeronaut. Sci.*, 18:145–160, 1951.
- [27] W. Vieser, T. Esch, and F. R. Menter. Heat transfer predictions using advanced two-equation turbulence models. Technical report, CFX Technical Memorandum CFX-VAL10/0602, 2002.
- [28] S. Wallin and A. V. Johansson. An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *J. Fluid Mech.*, 403:89–132, 2000.
- [29] M. Wang and P. Moin. Dynamic wall modeling for large-eddy simulation of complex turbulent flows. *Phys. Fluids*, 14:2043–2051, 2002.
- [30] K. Wieghardt and W. Tillman. On the turbulent friction layer for rising pressure. In D. E. Coles and E. A. Hirst, editors, *Computation of Turbulent Boundary Layers - 1968 AFOSR-IFP-Stanford Conference*, volume II, pages 98–123. Thermosciences Division, Department of Mechanical Engineering, Stanford University, 1969.
- [31] D. C. Wilcox. *Turbulence modeling for CFD*. DWC Industries, La Canada, 1998.

(C)
**Investigation of Vortex Breakdown
over a Pitching Delta Wing applying the DLR TAU-Code
with Full, Automatic Grid Adaptation**

Thomas Alrutz, Markus Rütten

*DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

Paper 5162, *35th AIAA Fluid Dynamics Conference*,
June 6–9 2005, Toronto

Abstract

This paper covers a description of the complete process chain used to investigate the vortex breakdown over a pitching delta wing. Special attention is given to the dynamic adaptation strategies used to capture the main flow structures as well as the analysis of vortex breakdown and the formation of primary and secondary vortices. The demonstrated approach with hybrid grids and a standard URANS flow solver shows extraordinary good results for this type of application.

Nomenclature

α	Edge weight factor
\bar{N}_p	Number of new grid points
$\Delta\Phi$	Edge sensor function
ω	Vorticity
ϕ	Solution variable
$\tilde{\Omega}$	asymmetric part of the velocity gradient tensor
\tilde{S}	symmetric part of the velocity gradient tensor
\vec{v}	Velocity vector
B_k	Refinement boxes
c_ϕ	Scaling factor for solution variables
e	Edge
h	Edge length
H_n	Normalized helicity density
I	Edge indicator
i, j, k	Indices
max_p	Maximal point numbers
min_h	Minimal edge length
N_ϕ	Number of solution variables
N_e	Number of edges
N_k	Kinematic vorticity
N_p	Number of grid points
p_i	Grid point index
p_{tot}	Total pressure
q	Relative factor of new grid points
t	Time
v_{rot}	velocity induced by vorticity
x, \vec{x}, \vec{x}'	Vertices

1 Introduction

One typical flow property of a sharp edged delta wing is the flow separation, which occurs at even moderate angles of attack at the whole leading edge of the wing. Here the high pressure difference between the lower and upper side of the delta wing enforces the shear layer, coming from the low wing side, to roll up and to formate a dominant

primary vortex pair on the lee side of the wing. These primary vortices, which are counterrotating, generate a significant additional lift by inducing a huge low surface pressure region on the wing without increasing the drag in the same manner. The additional lift and the demand for faster military aircraft has pushed the development of highly swept delta wings. Thus in the past most scientists since Lambourne and Bryer [18] were concentrating on highly swept delta wings. For these type of wings in general the dominant flow structures of a pitching delta wing are well known, see for example the work of Hummel [13] or Visbal [35].

Normally research work has concentrated on highly swept delta wings, but since modern fighter aircrafts have more weakly swept delta wings, the flow phenomena of these wing types become focal point of the research work. And especially the vortex breakdown phenomenon is the topic of many investigations because it has a significant influence on the flow structure and the consequent flight behaviour and manouverability of modern aircraft. For this type of wing the flow structure is more complicated. Typically besides primary, secondary and tertiary vortical structures, the weak sweep angle of the wing emphasize substructures and topological flow structure changes because the vorticity in the vortex core is not as strong as in highly swept delta wing shows. The goal of the work presented in this paper is to resolve this type of flow phenomena applying standard URANS calculations and using adaptation strategies. In the literature [8] structured grids are used which have limitations in local grid refinement, or more global refinement strategies are applied which are concentrating on flow entities which lead to inadequate resolution of substructures. One point often discussed is that URANS calculations are too dissipative due to the classical turbulence models. This results in a smoothing effect on vortical substructures. Therefore one modern tendency in calculation of the massively separated flow (i.e. the flow over delta wings) is the Detached Eddy Simulation (DES) approach. Here resolving more turbulent substructures leads to a less smoothed solution and allows to resolve more than only the main flow phenomena. However, DES is still not well understood and we wish to examine the feasibility of an adaptation strategy to resolve finer vorticity structures. Therefore, we present an approach that uses a hybrid URANS solver coupled with a sophisticated automatic grid adaptation strategy to resolve the vortical flow (in particular all main vortical structures and substructures of a stable vortical system). We apply our approach to a pitching delta wing. Furthermore we discuss the physical phenomena resolved with our approach.

1.1 The DLR TAU-Code Software package

The DLR TAU-Code is a finite-volume Euler/ Navier-Stokes solver working on unstructured hybrid grids. The code is composed of independent modules: Grid partitioner, a preprocessing module, the solver and a grid adaptation module. The grid partitioner and the preprocessing are decoupled from the solver in order to allow grid partitioning and calculation of metrics on a different platform than used by the solver. The flow solver is a three-dimensional finite volume scheme for solving the Unsteady Reynolds-Averaged Navier-Stokes equations. Further details of the TAU-Code solver may be found in [30, 9, 31].

In the next sections we give an overview of the adaptation module and concentrate on

the grid adaptation strategy used to track down the details of flow phenomena such as the vortex breakdown and the formation of substructures in the shear layer rolling up to form the leading edge vortex.

1.2 The Grid Adaptation module

The adaptation module of the TAU-Code consists of three different components for various grid manipulations to adapt a given grid to the solved flow field:

- y_+ based grid adaptation to adjusted the first wall distance over turbulent surfaces in hybrid grids
- hierarchical grid refinement and derefinement to introduce new grid points on a given edge-indicator function without producing hanging nodes
- surface approximation and reconstruction for curved surfaces after introduction of new grid points

For the investigation of the flow around a generic delta wing the y_+ based grid adaptation and the surface approximation are of minor interest. Our main focus is on the grid refinement and derefinement algorithm along with the edge-indicator function to detect regions where local refinement/derefinement is necessary.

1.2.1 The refinement and derefinement algorithm

The refinement/derefinement algorithm works on a edge-based data-structure. This means all polyhedra elements (tetrahedra, prisms, pyramids) are seen by the algorithm as sets of ordered edges. After the extraction of the global edge-list is done, the algorithm uses an edge-indicator function to determine which edges have to be refined or derefined. After the selection of the edges, the polyhedra elements in the grid are scanned, whether they are effected from the edge refinement or derefinement. Where necessary the algorithm forces regular refinement of an element by selecting further edges for refinement (see figure 1). This feature guarantees stable sequences of element

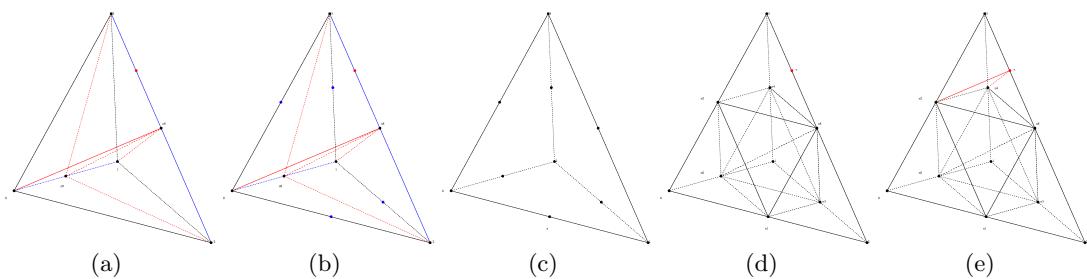


Figure 1: Marking of additional edges to force regular refinement of the parent tetra.

refinement, because in fact only the regular (isotropic) refinement case on a tetrahedron is performed[1]. To prevent the propagation of the local refinement through the grid all possible refinement cases for tetrahedra are implemented as well as two anisotropic directionel refinement cases for prisms. For pyramids there is only a small subset of

possible refinement cases implemented due to grid consistency reasons. Further details of the refinement cases maybe found in[1, 2]. For the derefinement the algorithm although keeps track of the complete refinement hierarchy of each element in the grid. This means for our edge-based element representation, that all parents which are one level away from the actual grid elements have to be considered for the calculation of the global edge-list in order to allow the edge-indicator function to select those edges for derefinement (i.e the blue edges in figure 1 (a) and (b)). Thus we are able to rebuild the initial grid at every time within the simulation process. This enable us to track down a flow phenomenon through time and space.

1.2.2 The edge-indicator sensor functions

For local mesh refinement/derefinement it is necessary to have a proper indicator. Some approaches use a residual-based indicator[10] or a adjoint approach while other make use of gradients or differences of any suitable flow variable. The latter is our choice for the adaptation module.

The approximated gradient $G_{(\Phi)}$ of a variable Φ in discrete form is $\Delta\Phi/h$ with $\Delta\Phi = \Phi_{p_1} - \Phi_{p_2}$, i.e. the difference between the point values of the two points p_1 and p_2 connected by one edge, where h is the length of the edge. We write the indicator function as:

$$I = \Delta\Phi h^\alpha \quad (1)$$

A widely used formulation is $\alpha = 1$, i.e.: $G_{(\Phi)}h^2$. The advantage of scaling the indicator with a positive value of α is that the refinement stops automatically in the corresponding area after several cycles. We will discuss the benefit of such a scaling for our investigation in section 2.

Our choice of $\Delta\Phi$ for the indicator function is :

$$\Delta\Phi_e = \max_{i=0,\dots,N_\phi} \left(c_{\phi_i} \frac{(\Delta\phi_i)_e}{(\Delta\phi_i)_{max}} \right) \quad (2)$$

with N_ϕ being the number of different flow variables considered and e the edges in the grid.

The weights c_{ϕ_i} are scaling parameters which enable the choice of different combinations of the single parts of the indicator (to be set to zero in order to turn off ϕ_i).

The reference values $(\Delta\phi_i)_{max}$ are for a balanced scaling of each part of the indicator function with

$$(\Delta\phi_i)_{max} = \max_{e=0,1,\dots,N_e} ((\Delta\phi_i)_e), \quad (3)$$

for all edges e in the grid. For the edge-indicator three differences we can use are

1. The differences (Δ_d) of the flow values

$$(\Delta_d\phi_i)_e = |\phi_i(x_{p_1}) - \phi_i(x_{p_2})| \quad (4)$$

2. The differences of the gradients (Δ_g) of the flow values

$$(\Delta_g\phi_i)_e = |grad(\phi_i(x_{p_1})) - grad(\phi_i(x_{p_2}))| \quad (5)$$

3. The differences of the reconstructed flow values (Δ_r) to the edge midfaces

$$(\Delta_r \phi_i)_e = |(\phi_i(x_{p_1}) + \frac{1}{2}x_e \cdot \text{grad}(\phi_i(x_{p_1}))) - (\phi_i(x_{p_2}) - \frac{1}{2}x_e \cdot \text{grad}(\phi_i(x_{p_2})))| \quad (6)$$

with ϕ_i the flow value, p_1 and p_2 the two edgepoints of edge e and $x_e = x_{p_1} - x_{p_2}$.

Any of the flow variables of the solver output can be used with these edge-indicators. This means all primitive and all additional user defined flow variables can be applied (e.g. x-,y-,z-velocity, total pressure, vorticity, cp, etc.).

1.2.3 Control of the refinement/derefinement

In order to have control over the number of points/vertices to be introduced or reduced by each adaptation step, we implement several parameter which give the user some control over the behaviour of the adaptation. In the following we list the set of parameters we used for our calculation of the vortex breakdown :

- Percentage of new points - This values defines the percentage of new grid points \bar{N}_p , which should be added or removed in each adaptation step, relative to the actual number of vertices N_p in the grid.

$$q = \frac{\bar{N}_p + N_p}{N_p} \quad (7)$$

- Maximum point number - This is an upper bound (limit) for the grid dimension, thus the algorithm is not allowed to increase the number of grid points over this bound.

$$\max_p \geq \bar{N}_p + N_p \quad (8)$$

- Minimum edge length - This is a lower bound (limit) for all edges in the grid. All edges which length are less than this bound may not be selected for refinement through the edge-indicator function.

$$\min_h > |x_e| \quad (9)$$

- Refinement boxes - With this parameter we are able to define one or more rectangular regions in the grid, which are not allowed to be refined through the algorithm.

$$B_k = \{x_{k,\max}, x_{k,\min}\} \quad (10)$$

- Structured layer refinement - This parameter allows to prevent the edges within the prismatic layers over selected surface groups in the grid, from being refined through the edge-indicator function.

The usage of these parameters in our adaptation strategy is discussed further in section 2.

2 Scenario for the Numerical Vortex Breakdown Investigation

In this section we describe the grid, the simulation parameters and the computational effort of our delta wing flow calculation. The main focus is on the skilled adaptation strategy to resolve flow details of vortex breakdown and substructures on primary vortices and at vortex axis.

2.1 Numerical Method

The calculated delta wing has a 57 deg sweep angle, a root chord of 237.5 mm, a wingspan of 291.0 mm at its trailing edge and is 5 mm thick (figure 2 (b)). The solutions were obtained for a freestream velocity of 68,84 m/s, and a Reynolds number of 1.2×10^6 . The angle of attack was varied by a periodic movement from 12 deg to 22 deg within 1 second. The harmonic motion has a reduced frequency based on full chord length of 0.01087 and for a time accurate calculation a physical time step of 5×10^{-5} seconds was employed. For the complete calculation we choosed the Spalart Allmaras one-equation turbulence model[33] with Edwards modification [7] and implicit time integration[6] together with a central scheme. The initial computational mesh had approximatly 1×10^6 grid points and around 2.5×10^6 elements (figure 2 (a)). It consists

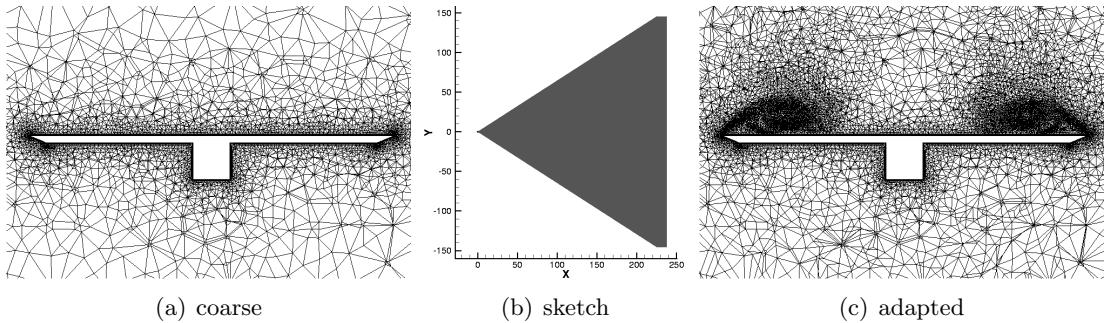


Figure 2: Sketch of delta wing (b) and cut of the grid at $x = 200$ by 12 deg angle of attack

of an inner region of 1.9×10^6 prisms in 20 layers for the boundary layer and an outer region of 650000 tetrahedra. The average edge length at the leading edge of the delta wing was about 2.5×10^{-4} of the root chord. The grid was adapted several times with the strategy described in section 2.2 to achieve a total number of 4.5×10^6 grid points (figure 2 (c)). This mesh was then used for the start of the pitching manouver. In order to reduce computing costs and storage space the grid has been adjusted through the adaptation at every 10th physical timestep (figure 3).

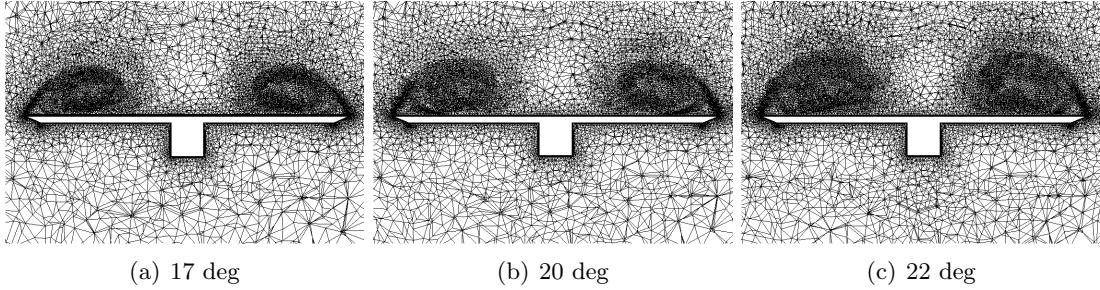


Figure 3: Evolution of grid refinement - cuts at $x = 200$ - angle of attack various from 17 deg to 22 deg

2.2 Adaptation Strategy

2.2.1 Sensor Settings

A quite good sensor for the detection of vortices is the *total pressure loss*. Melber et. al.[23] have shown that the usage of the *total pressure* as an adaptation sensor yields to a good resolution of the wake vortex for a high lift configuration. Thus we choosed for the complete simulation the difference indicator function (4) together with the computed values of p_{tot} and \vec{v} equally weighted. This mixture guarantees the resolution of the vortex core as well as the detection of the rolling-up procedure at the leading edges of the delta wing.

2.2.2 Parameter Settings

In order to prevent the indicator function from refining only edges at the leading edge of the delta wing, due to the usually strong gradients in this region, we set the limit for the *minimum edge length* to $min_h = 0.004$ mm which is about 2×10^{-6} of the root chord and therefore one magnitude smaller as in the initial grid. Thus the indicator function will not refine edges that are shorter than the given bound and after the refinement stops at the leading edge, the points are free to move into the region of the vortices. Furthermore we deselect the complete lower surface of the delta wing from refinement with the *structured layer refinement* parameter to save additional points. For each adaptation step we choosed to add 10% ($q = 0.10$) of new points to the grid until the upper limit of $max_p = 4.5 \times 10^6$ was reached. From this point of the calculation, the adaptation was only allowed to introduce new points whenever others are removed. This limit was increased to $max_p = 5.5 \times 10^6$ after the delta wing had reached 21 deg angle of attack. The refinement boxes (10) were used to prevent refinement in the unstructured region beneath the delta wing.

2.2.3 Edge Scaling

The distribution of grid points during the simulation can be controlled by the edge weight factor of α (1). This parameter allows to control the influence of the edge length h onto the indicator function I . By default $\alpha = \frac{1}{2}$ in order to strengthen the shorter edges which are typically near the surface where one would like to investigate

the flow. However, this default setting would end up into a quite good local refinement of the vortices itself but leaves us with the problem that the majority of the grid points will move inside the vortex core. Therefore we choosed to set $\alpha = \frac{3}{4}$, which leads to a rather smoother distribution of points around the main vortex core (see figure 3).

2.2.4 Summary

All those parameters were set right at the start of the simulation process and were not changed afterwards. Thus the only changes of the adaptation parameter was at the end of the simulation as the upper limit for the points was increased. So we just have to start the calculation and let it run for the complete time periode without any corrections.

3 Results

In this section we want to present results of the flow field analysis of the time dependend dataset. Therefore, we have to remind common vortex identification theories and techniques, which we want to apply on the dataset to show some results. Afterwards we visualize the main vortical structures over the delta wing and concentrate on the rolling-up process of fluid and the formation of primary and secondary vortices. Here we have a extended look at appearing subvortices and their impact on the flow field. With this example we can show the success of our adaptation techniques and strategy.

3.1 Identification of Vortices

In the first step we want to give an overview of the main flow structure of the delta wing flow. Although vortices are topic of a lot of research work there is not an exact mathematical and objective vortex definition, which can be easily applied to numerical or experimental data. This problem is well described by Lugt[20, 21], Kida[16] and Miura[24]. Similar problems occur if we have to define the vortex axis. Here often geometrical criterions are introduced and applied, see for example the vortex axis and vortex region identification scheme from Jiang, Machiraju and Thompson[15] or the work of Banks and Singer[32]. An illustrated approach is proposed by Roth [27], which was generalized and improved by Rütten[28, 29].

In the latest discussions about a stringent vortex definition there seem to be promising hypotheses, but these lead to criterions, in which higher derivatives have to be used, see for example the work of Haller [11]. But therefore, this is not practicable in use, regarding the aspect of complex geometries of modern aircraft applications. These problems in mind different approaches to identify large scale vortices have been developed and are well known. For example Dallmann [4] has developed a topological criterion as vortex definition, the so called *Invariant Q* criterion, which defines a vortex as a region, where the second invariant of the velocity gradient tensor is greater than zero. A similar criterion is the *Lambda2* criterion from Jeong and Hussain[14]. Here, we use the well known vortex definition of Truesdell[34]. He introduced the kinematic vorticity number which is defined

$$N_k = \frac{\|\tilde{\Omega}\|}{\|\tilde{S}\|} > 1. \quad (11)$$

From equation (11) it is seen that if the kinematic vorticity number is greater than one, vorticity exceeds shear, this means that the rotational part of the fluid outbalances the shear part. This is the significant condition for a point which then belongs to a vortex region. Applying this criterion on the delta wing flow leads to a hull surface within the particles are forming a vortex. In figure 4 the vortical structure is depicted.

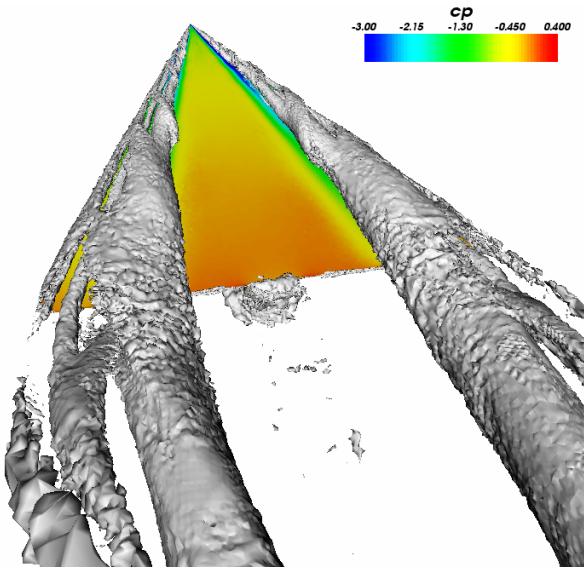


Figure 4: Vortical structure at 17 deg

One shortcoming of this definition is the loss of rotational information. To overcome this Levy et al.[19] introduce the normalized helicity density, often called stream vorticity:

$$H_n = \frac{\vec{v} \cdot \vec{\omega}}{|\vec{v}| \cdot |\vec{\omega}|}. \quad (12)$$

Equation (12) can be interpreted geometrically: It represents the cosine between the velocity vector and the vorticity vector. Following this idea a vortex region can be defined as a region with cosine values exceeding a certain limit. The rotation sense of a vortex is determined by the sign of the helicity density, so it is possible to differentiate between counterrotating vortices. This can be used to separate primary from secondary vortices and so on. For example in figure 5(a) the clockwise rotating vortices are colored in gold, the counterrotating vortices are colored in green.

This approach has another advantage too: An important point is that the vortex axis can be found at positions with maximal helicity density values, excluding walls, where the velocity vanishes. This can be used to set a start point for a segment wise streamline integration, following the extremum of helicity density.

3.2 Identification of Vortex Breakdown Bubbles

The helicity criterion can be used in another way too: Having in mind that vortex breakdown bubbles have as main feature a reversed flow, we can immediately conclude that there the sign of the helicity density has to change. Then we can calculate isosurfaces of a negative and positive isovalue of helicity and we can observe a local reversed flow region looking for embeded other colored ellipsoidal isosurfaces within a dominant flow pattern. Applying this approach we find some very small vortex breakdown bubbles on the vortex axis in the front / middle region of the delta wing, see figure 5.

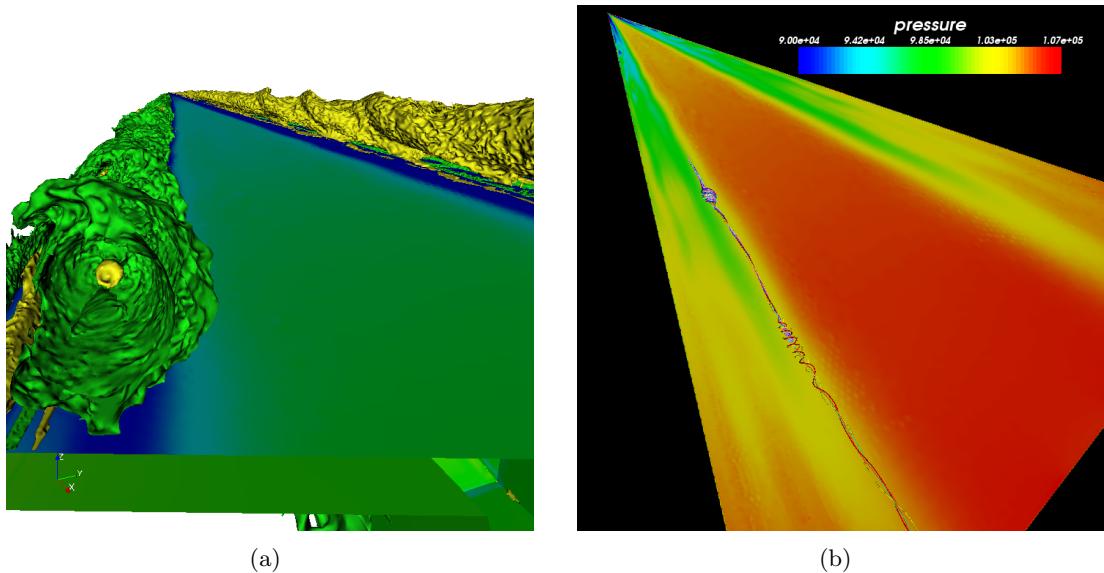


Figure 5: Helicity (a) and separatrices (b) shown on the left side of the delta wing

The reversed flow is colored in yellow and surrounded by the green colored primary vortex. The occurrence of these bubbles seems to be astonishing because under this flow conditions and angles of attack lower than 17 degrees no vortex breakdown can be observed. But we have to remark that normally it is looked for a beginning vortex breakdown in the region behind the delta wing. And furthermore we have to emphasize that these bubbles are related to another physical effect, we could observe in the flow field and we will discuss later on.

3.3 Physical impact of better resolved vortical structures

By applying our adaptation criteria, which are especially tuned for vortex detection, we could resolve the typical vortical flow structure over a delta wing consisting of primary, secondary and tertiary vortices. Also we could resolve secondary flow effects like vortex appropriate separation and reattachment flow structure (figure 6).

In addition to that, the well refined simulation grids allows us to resolve the small helical bending flow separation with the following formation of subvortices. These helical rolling-up subvortices are well known from experiments[17] and in our case

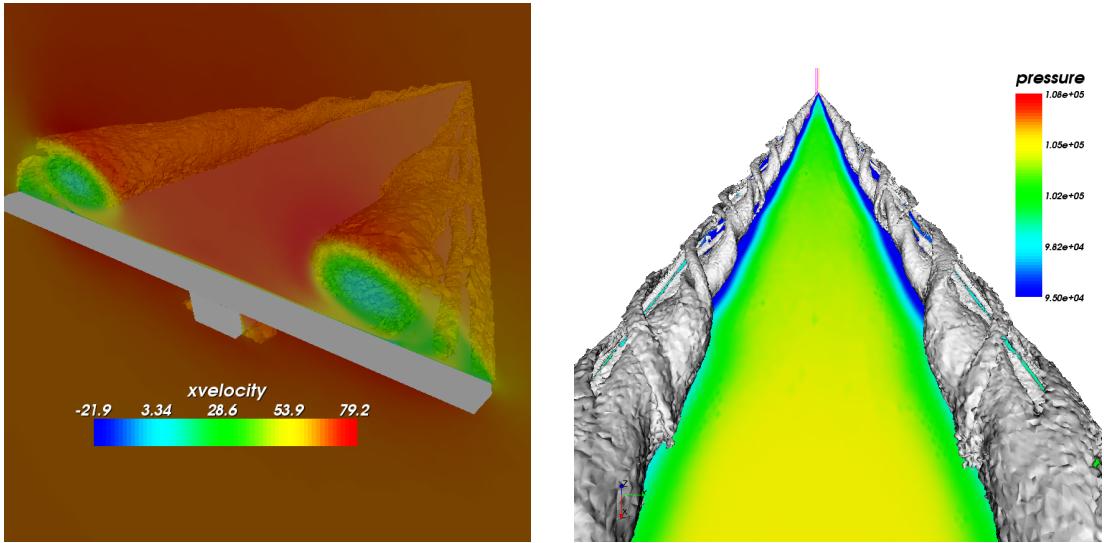


Figure 6: Flow structures over the delta wing visualized by kinematic vortex number, 17 deg. and 22 deg.

experiments[12], which were accompanying the simulation, could confirm these subvortical structures. Also DES calculations of Morton[25] show these typical flow separation subvortices, we have depicted in figure 7.

In contrary to Morton[25] in our investigation the delta wing had a weak sweep angle which leads to primary vortex with a greater cone angle than in the case of a highly swept wing. This greater cone angle has a deep impact on the particle transport in the vortical region: One consequence is that the formation of vortical substructures on the primary vortex hull is enforced in addition to the normal rolling-up of fluid particles. And if we compare the flow rolling-up to a primary vortex of a sharp swept delta wing and this of a weakly swept delta wing, then we can see that particle pathes and subvortices beginning nearby the leading edges have different twist ratios. As a second consequence the appearing subvortices are also forming small vortex tubes which are winding around the primary vortex, too.

Both together can have a big impact in the vortical flow and affect the flow at the primary vortex axis. Here the different cone angles of a strongly swept delta wing on the one hand and a weakly swept delta wing on the other hand is important: Normally, considering strongly swept delta wings, these rolling-up subvortices are on the one hand not strong enough on the other hand their twist ratio is to low to enforce stagnation points on the vortex axis. But in our simulation case the big cone angle of the primary vortex locally destabilizes the vortex so it is sensitive against the additional vorticity of the subvortices which act like the negative azimuthal vorticity described in the vortex breakdown theories of Darmofal [5] or Lopez and Brown [22]. Only if the cone angle of the primary vortex exceeds a certain value then the angle between the vortex tubes and the primary vortex axis is also big enough to allow an significant inducing force of the subvortices on the flow at the primary vortex axis. In figure 8 the subvortical structures are depicted. On the left side a volume rendering of the numerical dataset

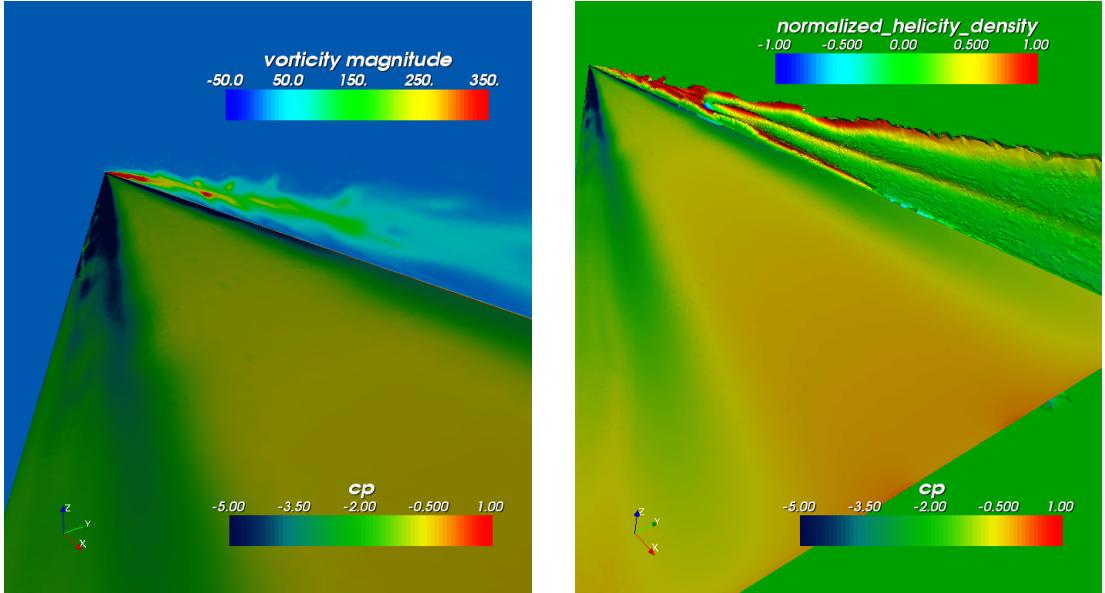


Figure 7: Vorticity magnitude and normalized helicity density mapped on a slice through the primary vortex

is performed, showing the dominating vortical structures and especially the rolling-up process of vorticity to subvortices. On the right side the figure shows an illustration of the vorticity magnitude of PIV data from [12]. The comparison between the PIV experimental results and the numerical simulation results show a good conformance in resolution of the subvortical flow pattern, taking into account the lower spatial resolution of the PIV measurements. Considering this illustrated devolution of the subvortices and following the vorticity induction law the negative azimuthal vorticity component of the subvortices will induce a reversed axial velocity on the axis and thus enforce a small vortex breakdown bubble at the vortex axis. Locally at the breakdown bubbles the primary vortex diverges but later on this vortex is feeded again by new vorticity of the rolling up process and will be stabilized. Therefore, the overall structure hides these small breakdown bubbles and the main flow structures change appears with the trailing edge vortex breakdown.

To cause the explained flow effect the subvortices must have a small rolling-up radius, therefore, this small breakdown bubble can only appear at the beginning of the primary vortex respectively over the first part of the delta wing. Otherwise the distance between the subvortices and the primary vortex axis is to big to have a significant influence. Considering the formula of vorticity induction:

$$\vec{v}_{rot}(\vec{x}, t) = -\frac{1}{4\pi} \int_V \frac{(\vec{x} - \vec{x}') \times \vec{\omega}(\vec{x}', t)}{|\vec{x} - \vec{x}'|^3} dV, \quad (13)$$

one can see that the distance from the axis to the vortex hull goes square and inverse. In the equation (13) \vec{x} is the point, at which the velocity is induced and \vec{x}' stands for the location from which the inducing effect starts. Thus, in the further streamdown development of the primary vortex the radius from the vortex axis to the vortex border

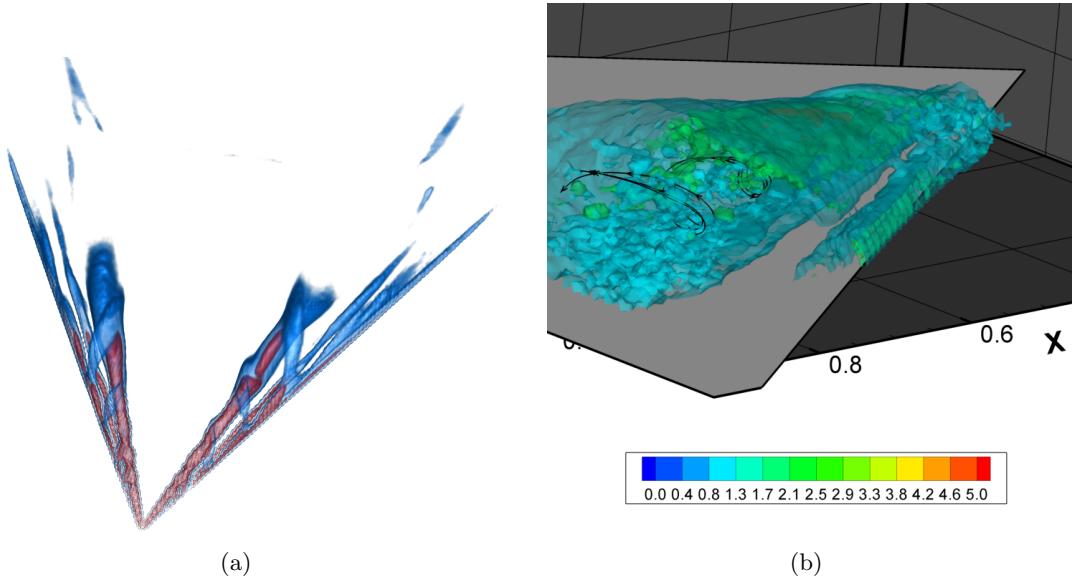


Figure 8: Comparison of volume rendered numerical (a) and PIV experimental results (b) at 17 deg

is too large to allow such vorticity induction effects on the vortex axis, and so the vortex stabilizes again. Therefore, the small breakdown bubbles in the front part of the delta wing are an effect of the rolling-up of subvortices on the primary vortex. Later on, with increasing angle of attack, the vortex will build up a new vortex breakdown structure, but then this breakdown is caused by the primary vortex itself.

From the numerical simulation point of view one have to consider the different dimension of primary vortices and rolling-up subvortices. To emerge the described vortex breakdown process the subvortices have to be spatially resolved. This means at least 10 grid cells have to be placed there to allow the formation of such a subvortex. Without an appropriate adaptation scheme, which can use highly customized formulations, the only possible way to resolve such flow effects would be to refine the grid globally to get the crucial spatial resolution. But in this case the number of grid cells or points in this region would be nearly ten times higher than actually needed and demonstrated by our approach. Regarding our results from this point of view we think that the used adaptation strategy (see section 2.2) was successful. Especially if we look at figure 3, we can demonstrate the local grid adaptation that follows the flow automatically.

4 Conclusion

Our simulation combined with the described adaptation strategy shows many significant primary and secondary flow structures, not normal seen in URANS code. So we could get a good impression of the rolling-up process of vorticity to subvortices on the primary vortex hull. Furthermore the impact of these structure on the flow field, the unexpected development of a vortex breakdown bubble at the front section of the

delta wing, could be resolved. Only DES calculations could provide comparable results, because in this case the grid resolution would be high enough. Here, the use of our dynamic adaptation strategies in combination with advanced adaptation criteria allows us to use common URANS simulation techniques on unstructured hybrid grids to get the described dominant and secondary flow features. Considering the underlaying integration of the automatical adaptation step inside the simulation process, the presented work shows extraordinary good results. From this point of view we could avoid the effort normally needed for such type of simulation. Therefore, in the near future we want to concentrate on the development of further adaptation techniques. Especially the point of flow feature based adaptation will be addressed, because the here presented adaptation approach allows us easily to integrate new concepts and formulations of refinement sensors.

Acknowledgments

The authors are grateful to Keith Weinman for helpful discussion. This research work was supported by the Institute of Aerodynamics and Flow Technology of the German Aerospace Center (DLR) Göttingen.

References

- [1] Thomas Alrutz. Erzeugung von unstrukturierten Netzen und deren Verfeinerung anhand des Adaptationsmoduls des DLR-TAU-Codes. Diplomarbeit, Universität Göttingen, 2002.
- [2] Thomas Alrutz. *MEGAFLOW — Numerical Flow Simulation for Aircraft Design Results of the second phase of the German CFD initiative MEGAFLOW presented during its closing symposium at DLR, Braunschweig, Germany, December 10th and 11th 2002*, volume 89 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, chapter 7 Hybrid Grid Adaptation in TAU, pages 109–116. Springer Verlag, Berlin, 2005.
- [3] D. Banks and B. Singer. Vortex tubes in turbulent flows: Identification, representation, reconstruction. Proceedings of IEEE Visualization, pages 132–139, October 1994.
- [4] U. Dallmann. Topological Structures of Three-Dimensional Flow Separation. IB 221-82 A 07, DFVLR, 1983.
- [5] D. L. Darmofal. The role of vorticity dynamics in vortex breakdown. paper 93-3036, AIAA, 1993.
- [6] Richard Dwight. Application of approximately factored implicit schemes to unsteady Navier-Stokes calculations. *Proceedings of the ICCFD3 Conference, Toronto*, 2004.

- [7] JR Edwards and S. Chandra. Comparison of Eddy Viscosity – Transport Turbulence Models for Three-Dimensional, Shock-Separated Flowfields. *AIAA Journal*, 39(4), April 1996.
- [8] J. A. Ekaterinas and L. B. Schiff. Vortical flows over delta wings and numerical prediction of vortex breakdown. paper 1990-0102, AIAA, Reno, NV, USA, 1990.
- [9] Martin Galle, Thomas Gerhold, and John Evans. Parallel Computation of Turbulent Flows around Complex Geometries on Hybrid Grids with the DLR-Tau Code. In A. Ecer and D.R. Emerson, editors, *Proceedings of the 11th Parallel CFD Conference*. Verlag "North Holland", 1999. 23-26 May in Williamsburg, VA.
- [10] W.G. Habashi, J. Dompierre, Y. Bourgault, M. Fortin, and M.-G. Vallet. Certifiable computational fluid dynamics through mesh optimization. *AIAA*, 36(5):703–711, 1998.
- [11] G. Haller. On objective definition of a vortex. *Journal Fluid Mech.*, 525:1–26, 2005.
- [12] A. Henning, M. Rütten, C. Wagner, and M. Raffel. Stereo PIV Investigation of a Vortex Breakdown above a Delta Wing with an Analysis of the Vorticity Field. paper 2005-4908, AIAA, 2005.
- [13] D. Hummel. Documentation of separated flows for computational fluid dynamics validation. *AGARD-CP-247*, 2:15–28, 1988.
- [14] J. Jeong and F. Hussain. On the identification of a vortex. *Journal Fluid Mech.*, 285:69–94, 1995.
- [15] M. Jiang, R. Machiraju, and D. Thompson. Detection and visualization of vortices. Technical report, IEEE Visualization, October 2002.
- [16] S. Kida and H. Miura. Identification and analysis of vortical structures. *Eur. Journal Mech. B/Fluids*, 17(4):471–488, 1998.
- [17] S. M. Klute, P. P. Vlachos, and D. P. Telionis. High-Speed Digital-Particle-Image-Velocimetry Study of Vortex Breakdown. *AIAA*, 45(3):624–650, march 2005.
- [18] N.C. Lambourne and D.W. Bryer. The bursting of leading edge vortices. some observations and discussion of the phenomenon. *ARC R. & M*, 3282, 1962.
- [19] Y. Levy, D. Degani, and A. Seginer. Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, 28(8):1347–1352, August 1990.
- [20] H. J. Lugt. The dilemma of defining a vortex. In U. Müller, K.G. Roesner, B.Schmidt(eds.), *Theoretical and Experimental Fluid Mechanics*, pages 309–321, 1979.
- [21] H. J. Lugt. *Introduction to Vortex Theory*. Vortex Flow Press, Inc., Maryland, 1996.

- [22] Lopez J. M. and Brown G. L. Physical mechanism of axisymmetric vortex breakdown. *Tenth Australasian Fluid Mechanics Conference, University of Melbourne*, 1989.
- [23] S. Melber-Wilkending, R. Wilhelm, and H. Frhr. von Geyr. RANS Solutions for a Complex High-Lift Configuration of a Transport Aircraft with Engine Including Improved Resolution of the Nearfield. paper 2004-5081, AIAA, 2004.
- [24] H. Miura and S. Kida. Identification of central lines of swirling motion in turbulence. Proceedings of Intl. Conference on Plasma Physics, pages 866–896, Nagoya, Japan, 1996.
- [25] S. A. Morton, J. R. Forsythe, A. M. Mitchell, and D. Hajek. Detached-Eddy Simulations and Reynolds-Averaged Navier-Stokes Simulations of Delta Wing Vortical Flowfields. *Journal of Fluids Engineering*, 124(4):924–932, December 2002.
- [26] Jens-Dominik Müller. Anisotropic adaptation and multigrid for hybrid grids. *Numerical Methods in Fluids*, 40(3-4):445 – 455, 2002.
- [27] M. Roth. *Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization*. PhD thesis, Swiss Federal Institute of Technology Zürich, 2000.
- [28] M. Rütten. *Topologische Untersuchung des Wirbelplatzes zur Identifikation von Wirbelplatzparametern*. PhD thesis, Helmut-Schmidt-University, Hamburg, Germany, 2004.
- [29] M. Rütten. Vortex Axis Calculation by Using Vortex Features. paper AIAA 2004-2353, AIAA, 2004.
- [30] Dieter Schwamborn, Thomas Gerhold, and Roland Kessler. The DLR-TAU Code - an Overview. In *Proceedings ODAS 99*, 1999. ONERA-DLR Aerospace Symposium 21-24 June in Paris, France.
- [31] Dieter Schwamborn and Keith Weinman. On the Influence of Turbulence Modeling on Steady and Unsteady Flows. In H. Sobieczky, editor, *IUTAM Symposium Transsonicum IV*, volume 73 of *Fluid Mechanics and its Applications*. Kluwer, 2003.
- [32] B. Singer and D. Banks. A predictor-Corrector Scheme for Vortex Identification. Nasa contractor report 194882, icase report no. 94-11, NASA Langley Research Center, Hampton, VA, Maryland, 1994.
- [33] P.R. Spalart and S.R. Allmaras. A one equation Turbulence Model for Aerodynamic Flows. paper 1992-439, AIAA, Reno, Nevada, USA, 1992.
- [34] C. Truesdell. The Kinematics of Vorticity. Technical report, Indiana University, 1953.
- [35] M. R. Visbal. Computational and physical aspects of vortex breakdown on delta wing. paper 1995-0585, AIAA, San Diego, CA, USA, 1995.

(D)

Improvement of the Automatic Grid Adaptation for Vortex Dominated Flows using Advanced Vortex Indicators with the DLR-Tau Code

Markus Widhalm¹, Andreas Schütte¹, Thomas Alrutz², Matthias Orlt²

¹ *DLR Institute of Aerodynamics and Flow Technology,
38108 Braunschweig, Germany*

² *DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

Notes on Numerical Fluid Mechanics and Multidisciplinary Design,
Volume 96, New Results in Numerical and Experimental Fluid Mechanics VI,
Pages 186–193, Springer, 2008

Abstract

Vortex dominated flows appear in many flow simulations such as wake turbulence of an aircraft or a delta wing at a high angle of attack. For detailed investigations of vortex breakdown, vortex interactions or tracing vortex cores, an automated grid adaptation with suitable vortex indicators is essential. Physical indicators, e.g. the vorticity magnitude or the total pressure loss, are in most cases not sufficient for correctly identifying a vortex core.

This paper presents advanced vortex core indicators which properly identify a vortical structure independent of the flow case. These vortex indicators are tested in typical flow applications to determine the right cut-off value which is important for an automated adaptation procedure.

A grid refinement for a delta wing testcase in combination with the newly introduced vortex indicators will demonstrate the improvements compared to the standard pressure loss indicator.

1 Introduction

Over the past few years physical vortex indicators, e.g. the magnitude of vorticity or total pressure loss were used with the DLR Tau-Code [4] grid adaptation for vortical structure refinements. Recently, vortex identifications based on the kinematics implied by the velocity gradient tensor ∇V have been proposed in the literature and implemented in the Tau-Code. In addition, the normalized helicity as a vortex indicator has also been introduced. These indicators are local or point-methods where a function can be evaluated grid point by grid point. According to a criterion based on the point values it can classify each point being inside or outside a vortex. For resolving vortex dominated flows, an automated grid adaptation used with flow independent vortex indicators will be an efficient approach. Adapting only the relevant vortical structures is the main issue for many applications and will be presented here for a delta wing testcase.

2 Galilean Invariant Indicators

Chong et al [3] used the critical point theory to describe the topological features of flow patterns by forming a local Taylor series expansion of the flow field. The solution trajectories can be related and classified with three matrix invariants.

In a fluid flow the velocity gradient tensor $\nabla V_{i,j}$, computed from the nondimensionalised velocity vector, can be decomposed into a symmetric (S) and antisymmetric (Ω) part:

$$\nabla V_{i,j} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix} = S_{i,j} + \Omega_{i,j} \quad (1)$$

The symmetric part $S_{i,j}$ is defined by using the index notation with the Einstein sum-

mation convention as:

$$S_{i,j} = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & \frac{1}{2} \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \\ \frac{1}{2} \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) & \frac{\partial v}{\partial y} & \frac{1}{2} \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ \frac{1}{2} \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) & \frac{1}{2} \left(\frac{\partial w}{\partial y} + \frac{\partial v}{\partial z} \right) & \frac{\partial w}{\partial z} \end{pmatrix} \quad (2)$$

and $\Omega_{i,j}$ is:

$$\Omega_{i,j} = \begin{pmatrix} 0 & \frac{1}{2} \left(\frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \right) & \frac{1}{2} \left(\frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \right) \\ \frac{1}{2} \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) & 0 & \frac{1}{2} \left(\frac{\partial v}{\partial z} - \frac{\partial w}{\partial y} \right) \\ \frac{1}{2} \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} \right) & \frac{1}{2} \left(\frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} \right) & 0 \end{pmatrix} \quad (3)$$

The symmetric Part $S_{i,j}$ is the rate-of-strain tensor and $\Omega_{i,j}$ is the rotation tensor. Computing the eigenvalues of $\nabla V_{i,j}$ in Eq. 1 the following characteristic equation is satisfied:

$$\lambda^3 + P\lambda^2 + Q\lambda + R = 0 \quad (4)$$

P, Q and R are the three Galilean invariants and read as

$$P \equiv u_{i,i}, \quad Q \equiv \frac{1}{2} (u_{i,i}^2 - u_{i,j}u_{j,i}), \quad R \equiv \det(u_{i,j}). \quad (5)$$

2.0.1 Second Invariant Q

Hunt et al. [5] defined a vortex as the region with a positive second invariant, $Q > 0$. The second invariant is derived from the characteristic Eq. 4 and is defined as written in Eq. 5.

2.0.2 Kinematic Vorticity Number N_k

Truesdell [12] defined the kinematic vorticity number to measure "the quality of rotation". He defined N_k as:

$$N_k = \frac{\|\Omega\|}{\|S\|} \quad (6)$$

Melander and Hussein's [8] investigations of vortex core dynamics identified the core as a region with $N_k > 1$. N_k is non-dimensionalized by the magnitude of strain rate and identifies vortices with large and small vorticity as long as the quality of rotation is the same for both.

2.0.3 λ_2 Criterion

Jeong and Hussain [6] introduced a definition of a vortex in terms of the eigenvalues of the symmetric tensor $S^2 + \Omega^2$. The vortex core is defined as a region where two negative eigenvalues of $S^2 + \Omega^2$ appear. Note, that $S^2 + \Omega^2$ is symmetric and the eigenvalues are real values. The vortex core is identified with the requirement:

$$\lambda_1 \leq \lambda_2 \leq \lambda_3 \quad \text{and } \lambda_2 < 0 \quad (7)$$

3 Normalized Helicity H_n - Stream Vorticity

In comparison to the indicators above Levy, Degani and Seginer [7] introduced the normalized helicity. Two major shortcomings of the above indicators are the inability of indicating the swirl direction of the vortex and they are unable to differentiate between primary and secondary vortices. To locate and identify coherent structures the normalized helicity is used:

$$H_n = \frac{\vec{v} \cdot \text{rot} \vec{v}}{|\vec{v}| \cdot |\text{rot} \vec{v}|} = \cos \alpha. \quad (8)$$

High values of helicity reflect regions with high velocity and vorticity whenever both vectors get close to parallel. The cosine between the two vectors provides a sign and shows the direction of swirl in relation to the streamwise velocity field. H_n differentiates between primary and secondary vortices because it is directly related to the velocity vector. H_n was successfully used for a delta wing at a high angle of attack by Alrutz and Rütten [1].

4 Cut-off Analysis for the Advanced Indicators

Cut-off values control the representation of the vortex core by iso-surfaces. Each grid cell inside the iso-surface will be refined by the adaptation procedure. Due to their invariant behavior these thresholds almost remain constant for a wide range of applications. But, as mentioned by Miliou [10], some indicators, e.g. λ_2 criterion, are very sensitive.

Figure 1, 2 and 3 show the λ_2 criterion for a low aspect ratio wing with RAE 2822 airfoil sections. Figure 1 indicates the λ_2 iso-surfaces very close to the theoretical value of zero. Far behind the wing many grid cells are visible with a very small vortex activity and they are not important for the main vortex structure at the wing's side edges. Refining all these grid cells would add points not required for vortex identification. Figure 2 shows the λ_2 iso-surface at a cut-off value of 0.001. Most of the grid cells far behind the wing are not considered anymore. Some grid cells are still marked where the vortex core is weak. Figure 3 shows a stable vortex core on each side of the wing.

5 The Automated Flow and Grid Adaptation Chain

5.1 Gathering experience with different vortex indicators

The EC 145 is a midsize two engine helicopter. Experimental measurements have shown a separation at the back door below the tail rotor boom under certain flow conditions. Subsequently, a flow simulation was carried out to investigate the separation behind the helicopters fuselage in more detail. The computation was performed with a Mach number of 0.2081 and a Reynolds number of 4.33 million. The angle of attack is zero degrees. The main and rear rotor were modeled by actuator discs. In Figure 4-9 the iso-surfaces of the indicators at different cut-off values are shown. The vorticity magnitude $|\omega|$ and the total pressure P_{tot} values are found through inspection of visualized

iso-surfaces. The invariant indicators and H_n had to be changed slightly from their theoretical value and in these cases the iso-surfaces indicate the region of the grid which will be refined. The figures are shaded with the first component of the vorticity vector to detect the rotation direction, except the normalized helicity H_n which provides the sign of swirl directly.

All the indicators were able to detect the separation at the tail rotor boom. The normalized helicity H_n was able to split the swirl action from the rotor discs. Q , N_k and the $|\omega|$ detected vortices coming from the rotor discs. All approaches except H_n erroneously detect vortex structures on the fuselage. Finally, only H_n identifies the important sections at the rear fuselage for an efficient adaptation of the separation.

5.2 Use of vortex indicators in the Tau adaptation

Tau adaptation uses an edge based local refinement strategy. The main steps are the indication of edges to be divided considering a current solution and the subdivision of all elements which are needed to get a valid grid with respect to the indicated edge subdivisions [2]. Usually the edge indication uses differences of solution values, e. g. P_{tot} or $|\omega|$, or differences of gradients as sensors in order to minimize these differences which are supposed to be large in regions with large local errors.

Under some circumstances the differences of physical variables produced by the flow solver in the region of vortices are not large enough to resolve these flow phenomena. A possible explanation could be, that the initial grid resolution is too coarse for the flow solver to simulate the vortices sufficiently. So a flattened solution prevents the edge indication step from refining some vortex regions. Some simulations with pre-refined grids [11] seem to back this observation.

On the one side it may be hard to produce pre-refined initial grids for an unknown flow and on the other side it will be very expensive in terms of computational costs to make pre-refinements in all regions which are possible to contain unresolved vortices. Additionally, this method contradicts the idea of an automated grid adaptation. So the use of vortex indicators could be an alternative approach.

The vortex indication of Tau adaptation uses one of the sensors described above for the edge indication. All edges with a point which is found to be in a vortex region are considered. The needed target point number is found by scaling these edges with a power of its length and marking all edges with an indicator larger than a limit.

5.3 Adaptation results for a 65° delta wing with rounded leading edges

The flow field around a delta wing is well suited as a typical aerodynamic application for vortex dominated flows. The initial grid of the delta wing has a size of 1.8 million grid points. The flow conditions are a Mach number of 0.4, a Reynolds number of 3 million and an angle of attack of 13 degrees. The indicators are computed with the velocity vector V which is nondimensionalised by $(M * \sqrt{\gamma * p_r / \rho_r})$ where M is the Mach number, p_r and ρ_r are the reference pressure and density.

In this case the flow topology is different to a sharp leading edge case where two primary vortices are formed right from the apex at the leading edge. In a rounded leading edge case two primary vortices on each side of the wing rotate in the same direction, an inner

weaker and a stronger outer vortex. The formation of the inner vortex first occurs close to the apex. The stronger outer vortex is formed further downstream. Without going into more detail of the flow physics, this case is well suited to estimate the ability to accurately resolve different kinds of vortex structures within the flow field.

The impact of the grid adaptation is seen after several adaptation steps in Figure 10 left by using the total pressure P_{tot} and in comparison in Figure 10 right by using the normalized helicity H_n . The final grid using P_{tot} contains about 10 million and for H_n about 5 million grid points. The resulting grids are compared in a cut plane at 80 per cent of cord length. It is first of all seen that the main flow features (i.e. the vortices) are correctly predicted by the solver and detected by both adaptation indicators, as seen in Figure 10 for the left and right cut plane. It is evident that more flow structures are detected by the H_n indicator and these flow structures are more concentrated in the vortex region than in the case of using P_{tot} . One can see that in the case of H_n (right Figure 10) in addition to the inner and outer vortex as well the horse-shoe vortex generated by the strut is well detected in contrast to the P_{tot} case while introducing only the half number of grid points during the adaptation procedure. In the case of P_{tot} adaptation the addition of new grid points is concentrated on the strong outer vortex where the total pressure gradients are high. In case of H_n adaptation the addition of new grid points is more balanced and weak vortex structures will also be detected and refined.

In Figure 11 the pressure distribution of both CFD solutions in comparison to experimental data obtained by PSP (Pressure Sensitive Paint) measurements is shown. The center and right figure show the solutions of P_{tot} and the normalized helicity H_n adaptation. As discussed before, both numerical solutions are predicting the main flow features correctly compared to the experiment. In both solutions the inner vortex is predicted too weakly. However in the case of H_n adaptation the inner vortex is better refined as discussed before. In both cases the outer vortex is predicted to be too strong and appearing too far upstream.

This over-prediction of the outer vertex is mainly dependant on the two equation turbulence models used. Each of the available models have many different modifications or corrections implemented to make allowance for different physical phenomena. An improvement might be a higher order turbulence model solving this vortex dominated flow.

6 Conclusion

The advanced indicators with different cut-off values were presented to accurately identify vortex cores. From the proposed indicators the normalized helicity H_n is a very powerful and reliable indicator for complex flows. The Galilean invariant indicators Q , N_k and λ_2 are able to identify vortex cores but they tend to identify strong shear flows as well and finding the appropriate cut-off value becomes more sensitive. Thus it appears that the applicable cut-off values need further investigations for each indicator. A very effective and efficient way is demonstrated for vortex core refinement on the delta wing. The advantage of the specific vortex core refinement is essential and was the proposed aim for implementing the mentioned indicators. Another feature with

Galilean invariant and H_n in comparison to physical indicators is the refinement of the vortical structure only. Nevertheless, it seems practicable that both physical and advanced indicators are brought together in the grid adaption as sensors for refinement.

References

- [1] Alrutz T., Rütten M.: *Investigation of Vortex-Breakdown over a Pitching Delta Wing applying the DLR TAU-Code with Full Automatic Grid Adaptation.* 35th AIAA Fluid Dynamics, 6-9 June, Toronto, paper 5162, 2005.
- [2] Alrutz T., Orlt M.: *Parallel dynamic grid refinement for industrial applications.* In Proceedings of ECCOMAS 2006, Egmond aan Zee, The Netherlands, September 5-8
- [3] Chong M.S., Perry A.E., Cantwell B.J.: *A general classification of three-dimensional flow fields.* Phys. Fluids, A 2, 765, 1990.
- [4] Gerhold T., Friedrichs O., Evans J., Galle M.: *Calculation of Complex three-dimensional configurations employing the DLR TAU.* AIAA-97-0167, 1997
- [5] Hunt J.C.R, Wray A.A., Moin P.: *Eddies, stream and convergent zones in turbulent flows.* Center for Turbulent Research Report CTR-S88, p. 318, 1988.
- [6] Jeong J., Hussain F.: *On the identification of a vortex.* J. Fluid Mech., pp. 69-94, 1994.
- [7] Levy Y., Degani D., Seginer A.: *Graphical Visualization of Vortical Flows by Means of Helicity.* AIAA Journal, Vol. 28, No. 8, 1990.
- [8] Melander M.V., Hussain F.: *Polarized vorticity dynamics on a vortex column.* Phys. Fluids, A 5, 1992, 1993.
- [9] Metcalfe R., Hussain F., Menon S., Hayakawa M.: *Coherent structures in a turbulent mixing layer.* Springer, A 5, 1985.
- [10] Miliou A., Mortazavi I., Sherwin S.: *Cut-off analysis of coherent vortical structure identification in a three-dimesnional external flow.* Comptes Rendus Mecanique, pp. 211-217, 2005.
- [11] Schütte, A.; Einarsson, G.; Schöning, B.; Raichle, A.; Mönnich, W., Neumann, J.; Arnold, J.; Alrutz, T.: *Prediction of the Unsteady Behavior of Maneuvering Aircraft by CFD Aerodynamic, Flight-Mechanic and Aeroelastic Coupling.* RTO AVT-Symposium Budapest, April 2005.
- [12] Truesdell C.: *The kinematics of vorticity.* Indiana University 1953.

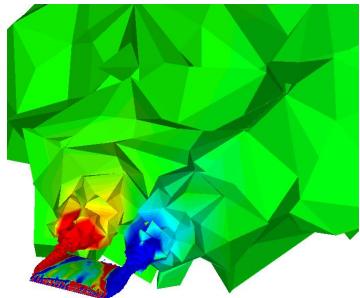


Figure 1: Iso-surface with λ_2 values ≤ 0

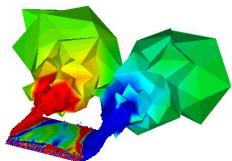


Figure 2: Iso-surface with λ_2 values ≤ -0.001

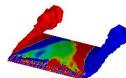


Figure 3: Iso-surface with λ_2 values ≤ -1.0

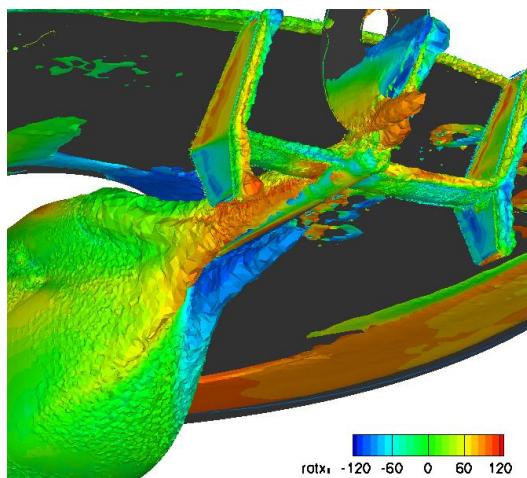


Figure 4: Iso-surface of $|\omega| = 100$

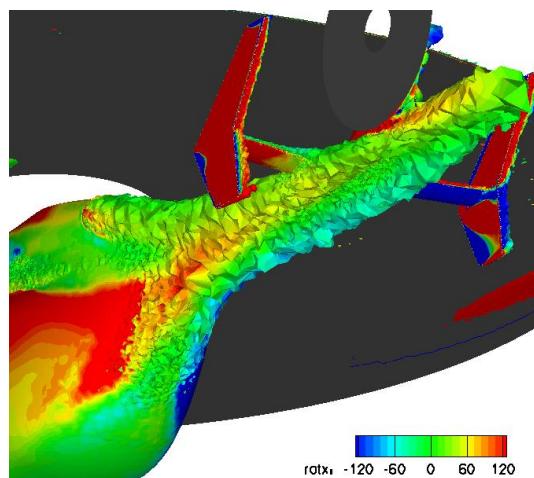


Figure 5: Iso-surface of $P_{tot} = 89.000$

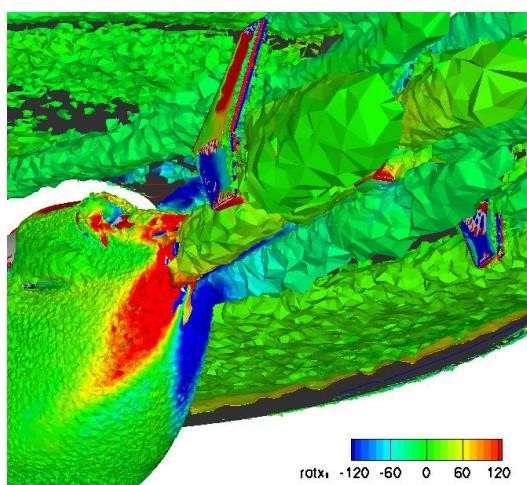


Figure 6: Iso-surface of $Q = 0.1$

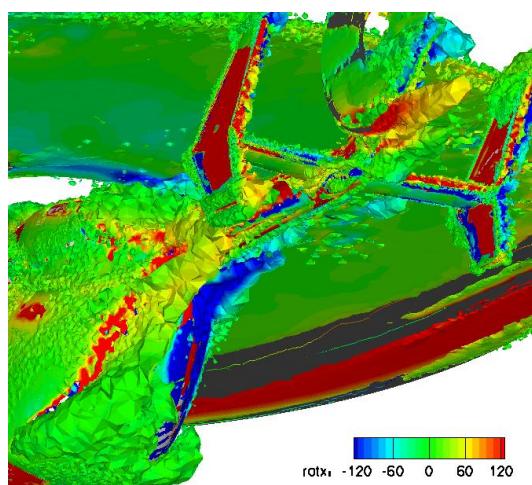
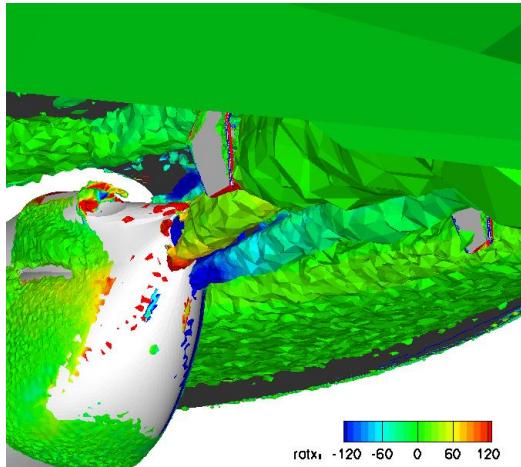
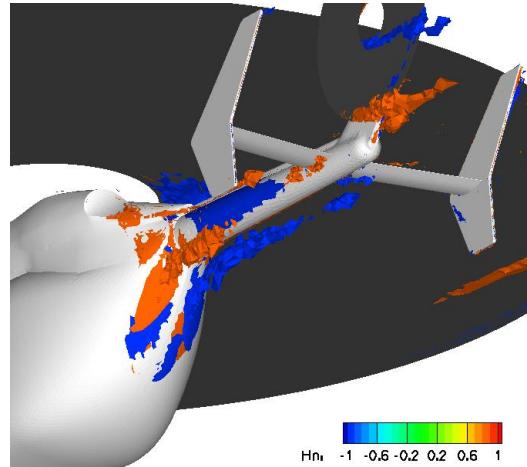
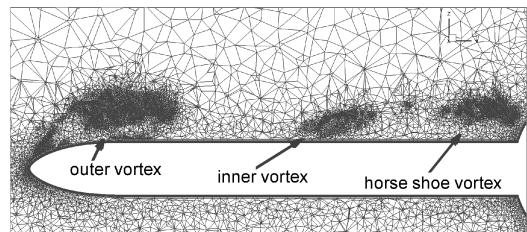
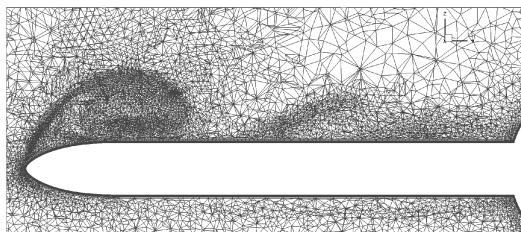
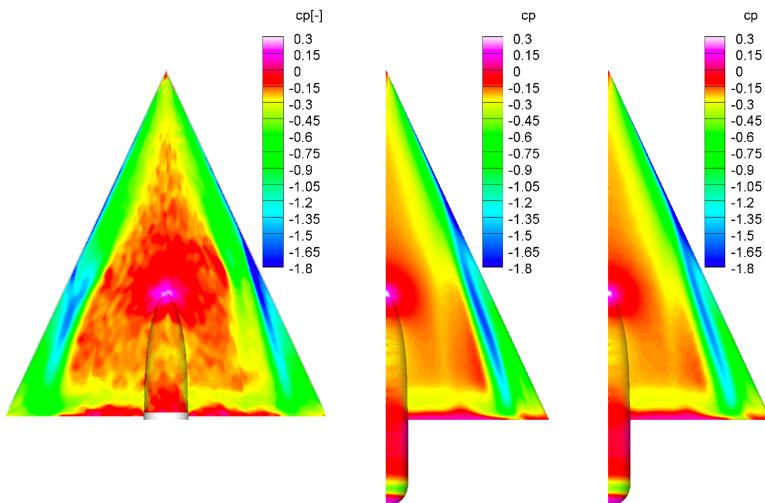


Figure 7: Iso-surface of $\lambda_2 = -1.0$

Figure 8: Iso-surface of $N_k = 1.1$ Figure 9: Iso-surface of $H_n = \pm 0.9$ Figure 10: Adapted grid with P_{tot} (left) and H_n (right) at 80 per cent cord length.Figure 11: Pressure distribution on the delta wing surface measured by PSP (left) and flow computation after several adaptation steps with the P_{tot} (middle) and H_n indicator (right).

(E)

A Vortex Axis and Vortex Core Border Grid Adaptation Algorithm

Markus Rütten¹, Thomas Alrutz¹, Holger Wendland²

¹ *DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

² *Department of Mathematics, University of Sussex,
Brighton BN1 9RF, England*

International Journal for Numerical Methods in Fluids,
DOI: 10.1002/fld.1792, 2008

Abstract

In the design process of hydro- and aerodynamical technical applications the numerical simulation of massively separated vortical flow is crucial for predicting, for example, lift or drag. To obtain reliable numerical results, it is mandatory to accurately predict the physical behavior of vortices. Thus, the dominant vortical flow structures have to be resolved in detail, which requires a local grid refinement and certain adaptation techniques. In this paper, a vortex flow structure adaptation algorithm is presented, which is particularly designed for local grid refinement at vortex axes positions and associated vortex core border locations. To this end, a fast and efficient vortex axis detection scheme is introduced and the algorithm for the vortex core border determination is explained. Since the interaction between vortices makes the assignment of grid points to a certain vortex axis difficult, a helicity based vortex distinction approach in combination with a geometrical rotational sensor is developed. After describing the combined different techniques in detail, the vortex feature adaptation algorithm is applied to analytical and more realistic examples, which show that the described grid adaptation algorithm is able to enhance the grid cell resolution locally such that all significant vortical flow phenomena are resolved.

1 Introduction

In modern aircraft and vehicle design, as well as in the development process of hydrodynamic components of technical applications, the numerical flow simulation plays an increasingly important role. In particular, one crucial point in the simulation process is the prediction of required design criteria, which have to be satisfied by an optimized solution. Moreover, the flow structure, which has to be resolved accurately, determines the turn around times in design and optimization processes. Often, massively separated flow occurs and, as a result, vortices start to develop, which then dominate the global flow structure. These vortices have significant influence on lift, drag, or pressure loss. Once they have developed, the time dependent evolution of the vortices determines the overall aerodynamical or hydrodynamical characteristics of the application and thus the efficiency. Hence, for the numerical simulation of flows it is essential to resolve all occurring vortices with all their main features such as the vortex axis, the core width, the vortex hull, their decay and also the interaction between vortices.

From the physical point of view, vortices are characterized by strong gradients in pressure, velocity, and the associated flow field. This requires the numerical simulation to resolve such gradients, which is only possible with an appropriate grid cell resolution. Only if the spatial resolution is sufficiently fine to resolve the involved gradients adequately a numerical simulation will be able to accurately predict the design objective criteria. However, in the light of computing power and economic costs it is, even nowadays, not possible to discretize the entire simulation domain uniformly at such a fine level such that all flow features are automatically detected and well represented. Hence, to achieve the overall goal, namely the improvement of the results of the numerical simulation, local grid adaptation techniques, based upon physical features, as well as grid refinement, and de-refinement strategies are crucial and have to be utilized.

Unfortunately, the precise definition of the flow features themselves is still an open field of research. This makes the design of a feature-based adaption algorithm particularly

difficult. A first step in this direction has been made in [1], where the vortex breakdown over a pitching delta wing has been investigated. However, here we are more interested in the actual feature extraction.

Thus, we will concentrate on resolving two of the major features of a vortex, the vortex axis and the border of the vortex core. To this end, we present an advanced vortex core detection and analysis method combined with an adaptation technique for local grid refinement.

The paper is organized as follows. In the next section, we will give a short and comprehensive overview of the most significant physical aspects of vortices, including methods to identify them and to localize their axis and core border. After that, we describe possible ways of how to classify and distinguish vortices. This will be useful to understand the physical properties of interacting vortices, which is necessary for the adaptation sensor used in our refinement algorithm, which is described in section three. The main part of this paper consists of describing the new vortex feature adaptation technique in detail (section four) and its application to academic test cases and to a more realistic technical configuration (section five). We conclude the paper by resuming the main results and indicating possible future work.

2 Vortices

Although most scientists working in fluid dynamics have a common idea of the term *vortex*, a precise mathematical description is difficult and not complete in the sense of covering all features of this kind of rotational fluid. This lack of a general vortex definition has led to various vortex identification schemes, which are relatively successful in practice but which are mainly restricted to specific applications. Hence, we briefly discuss the vortex related terms, pointing out that different vortex definitions and detection methods often lead to similar or identical terms. In the first part, without the intention of completeness, we give only those vortex definitions and identification schemes, which are relevant for our adaptation algorithm. Hence, we concentrate on purely kinematical vortex definitions, since we intend to use only the velocity field and its derivatives. In the second step, we are going to refer to those vortex axis definitions and axis detection schemes which are needed for the adaptation algorithm itself. Finally, the vortex core will be explained which is our main adaptation objective and which plays a crucial role in the physical development of the flow structure.

2.1 Definition and Detection of Vortices

Referring to the intrinsic difficulties of a general vortex definition, Lugt [24] speaks of a classical dilemma. Nonetheless, as an orientation he proposes the following definition of a vortex:

A vortex is the rotating motion of a magnitude of material particles around a common center.

This intuitive definition describes very well the visual observations in nature. Unfortunately, it shifts the problem of defining a vortex to the problem of defining the common

center of rotation. Furthermore, this definition is not valid in general, because it is not Galilean invariant under moving reference frames. Robinson [29] extended Lugt's definition by introducing a concrete observation time and a general moving reference frame:

A vortex exists when instantaneous streamlines mapped onto a plane normal to the vortex core exhibit a roughly circular or spiral pattern, when viewed from a reference frame moving with the center of the vortex core.

Mathematically, this implies that the vortex core and its motion have to be known in advance. But the vortex core is one of the major quantities of a vortex itself and in most technical applications its location is unknown *a priori*. Robinson's definition has another major drawback: There is no formal definition of the vortex core. It can only be defined with respect to the *vortex axis*, which is the common center of rotation.

Both definitions above are Lagrangian vortex definitions. In the Lagrangian approach the motion of particles is observed. In practice, particles have to be seeded and traced. To reduce the computational complexity, the initial positions of the particles have to be chosen as a function of the just mentioned common center. This requires again *a priori* knowledge about the vortical flow, which is often not given. Furthermore, Robinson's approach is also not invariant under Galilean transformations. As a remedy, Cucitore [11] proposed a vortex definition using the relative displacement of two neighboring particles over a short time interval.

For our purposes, the use of a Lagrangian vortex definition based on particle motion is not feasible. Since our algorithm is intended to be incorporated into an entire CFD solving process, we can only use the grid and the flow solution at a certain, fixed time and cannot use a sequence of temporal grids and solutions. Hence, a particle tracing can not be applied. This means that we have to concentrate on Eulerian and local vortex definitions, where local flow properties are investigated at fixed locations.

Almost all Lagrangian and Eulerian vortex definitions imply that a vortex is a regional flow pattern. This leads to the assumption that it is possible to define a bounding vortex hull surface, separating the vortical rotational flow region from the outer irrotational flow. This is an ideal, theoretical point of view and only valid for ideal fluids. Based on this assumption, Saffmann [32] found an elegant way of defining vortex filaments. Unfortunately, for real fluids, with the influence of viscosity, it is impossible to have an exact separating border, in particular if the unsteady vortex evolution process includes diffusion of vorticity coupled with strain. Nonetheless, from a practical point of view, vortex definitions which are providing a sharp separating border are useful for confining the region, which has to be regarded.

In the search for a pointwise, local Eulerian vortex definition Levy et al. [23] introduced the *normalized helicity density*, often also called *stream vorticity*:

$$h_n = \frac{\vec{v} \cdot \vec{\omega}}{|\vec{v}| |\vec{\omega}|}. \quad (1)$$

The normalized helicity density represents the cosine of the angle between the velocity vector \vec{v} and the vorticity vector $\vec{\omega}$. The physical idea behind employing the normalized helicity density for the vortex detection is the following one. In a vortical flow region,

the vortex lines and streamlines are winding around the vortex axis. Furthermore, they are entangled. This highly complicated relation between vortex lines and streamlines results locally in a simple geometrical relation: Near vortex core regions, the angle between the velocity \vec{v} and vorticity $\vec{\omega}$ is assumed to be small. Following this observation, a vortical region can be defined as a region where h_n differs significantly from zero. This vortex definition has the following, additional advantage: The orientation of the rotation of a vortex can be determined by the sign of the helicity density. Hence, it is possible to distinguish between counter-rotating vortices. The helicity can be used to separate primary from secondary vortices. This important property will be crucial for our adaptation algorithm. Unfortunately, this criterion is sensitive to fluctuations in the data, which may come from discretization errors or coarsely resolved flow structures. Since we are aiming at data sets coming from a numerical CFD computation, this is an issue we have to take care of. This can be done by using this criterion either in combination with one of the following vortex criteria or by setting appropriate thresholds. Another disadvantage is that this criterion is again not Galilean invariant, and hence only of practical relevance if a fixed reference frame is given.

Other pointwise local vortex definitions and identification schemes are based on properties of the local velocity gradient tensor $\nabla \vec{v}$. For example, one important feature of a rotational fluid is the existence of *complex-valued* eigenvalues of the velocity gradient tensor. This can be utilized in another vortex definition. Dallmann et al. [12] and Chong [10] defined a vortex using the discriminant

$$D = 27R^2 + (4P^3 - 18PQ)R + (4Q^3 - P^2Q^2) \quad (2)$$

of the velocity gradient tensor $\nabla \vec{v}$. A positive value of the discriminant D is equivalent to the existence of complex-valued eigenvalues and thus indicates a vortex. In (2), the discriminant is defined using the three invariants P , Q , and R of the velocity gradient tensor, which we state below using the strain tensor S and rotation tensor Ω , i.e.

$$S = \frac{1}{2}(\nabla \vec{v} + \nabla \vec{v}^T), \quad \Omega = \frac{1}{2}(\nabla \vec{v} - \nabla \vec{v}^T)$$

and the eigenvalues $\lambda_1, \lambda_2, \lambda_3$ of $\nabla \vec{v}$. The first invariant P is given by

$$P = -\text{trace}(\nabla \vec{v}) = -\sum_{i=1}^3 S_{ii} = -(\lambda_1 + \lambda_2 + \lambda_3). \quad (3)$$

For the velocity field \vec{v} , this corresponds to the negative sum of the principle rates of strain or the negative divergence of the velocity field, respectively. For an incompressible flow P vanishes.

The second invariant Q is the sum of the product of the eigenvalues of the sub-determinants of $\nabla \vec{v}$ and compares the influence of pure shear to the influence of fluid element rotation:

$$\begin{aligned} Q &= \frac{1}{2} \left[P^2 - \text{trace}(\nabla \vec{v})^2 \right] \\ &= \frac{1}{2} \left[P^2 - \sum_{i,j=1}^3 (S_{ij}S_{ji} + \Omega_{ij}\Omega_{ji}) \right] \\ &= \lambda_1\lambda_2 + \lambda_2\lambda_3 + \lambda_3\lambda_1. \end{aligned} \quad (4)$$

For completeness, we also state the third invariant R , which describes the stability of the local flow structure. It is given by the negative determinant of the velocity gradient tensor, i.e.:

$$\begin{aligned}
R &= -\det(\nabla \vec{v}) \\
&= \frac{1}{3} \left[-P^3 + 3PQ - \text{trace}(\nabla \vec{v})^3 \right] \\
&= \frac{1}{3} \left[-P^3 + 3PQ - \sum_{i,j,k=1}^3 (S_{ij}S_{jk}S_{ki} + 3\Omega_{ij}\Omega_{jk}\Omega_{ki}) \right] \\
&= -\lambda_1\lambda_2\lambda_3.
\end{aligned}$$

Dallmann [13] and Hunt et al. [18] suggested a vortex definition only based upon the second invariant Q of the velocity gradient tensor. An invariant $Q > 0$ means that rotation dominates shearing and indicates a vortex. This physical interpretation becomes more obvious when considering the kinematic vorticity number N_k introduced by Truesdell [36]. The kinematic vorticity number is defined as the dimensionless scalar

$$N_k = \frac{\|\Omega\|}{\|S\|} = \frac{1}{\sqrt{2}} \frac{|\vec{\omega}|}{\|S\|}, \quad (5)$$

using the Frobenius norm $\|\Omega\|$ of Ω , i.e.

$$\|\Omega\|^2 = \text{trace}(\Omega\Omega^T) = \sum_{i,j=1}^3 \Omega_{ij}\Omega_{ij}$$

and the relation $\|\Omega\|^2 = \frac{1}{2}|\vec{\omega}|^2$.

A point belongs to a vortex region if the rotational part of $\nabla \vec{v}$ out-balances the shear part and this is given by $N_k > 1$. For incompressible flow both definitions are equivalent expressions. Using the symmetry of S and the anti-symmetry of $\Omega = -\Omega^T$, we see that (4) can be rephrased as

$$2Q = -\|S\|^2 + \|\Omega\|^2, \quad (6)$$

which leads to

$$1 + \frac{2Q}{\|S\|^2} = \frac{\|\Omega\|^2}{\|S\|^2} = N_k^2. \quad (7)$$

Obviously, for an invariant Q larger than zero the kinematic vorticity number will be larger than one and vice versa.

A more sophisticated vortex definition comes from Jeong and Hussain [19], who introduced the λ_2 -criterion. After a compelling derivation they formulated the following condition for a vortex related pressure field:

$$\Omega^2 + S^2 = -\frac{1}{\rho} (\partial_{ij} p). \quad (8)$$

Following Jeong and Hussain [19], a vortex exists at locations, where the symmetric rotational vortical flow and the related symmetric shear produces a pressure minimum.

However, they also point out that this vortex induced pressure minimum may not be identical with the original pressure minimum in the flow domain. The latter one can also be caused by other superposed effects such as shocks, strong shear, or acceleration. The symmetric tensor $\Omega^2 + S^2$ has three real eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. To satisfy the condition of a pressure minimum, two eigenvalues have to be less than zero. This can be used as a formal definition for the vortex.

A vortex is a region where $S^2 + \Omega^2$ has two negative eigenvalues, i.e. where $\lambda_2 < 0$.

In a recent discussion about local vortex definitions between Wu [37] and Chakraborty [9], the latter one derived the so called *inverse spiraling compactness* criterion, which is based upon the observation that a vortical flow region may be characterized by the ratio of the real part to the imaginary part of complex eigenvalues. This criterion is a measure for the spatial extent of the local spiraling motion. In particular Chakraborty compared his criterion to the vortex criteria and definitions above with respect to their regional extend and he showed that, although all criteria predict slightly different bounding vortex hull surfaces, they all cover the main features: the vortex core border and the vortex axis. He also noticed that his *inverse spiraling compactness* criterion gives the largest region in contrast to the Q criterion, which predicts the smallest vortical related flow region. In our case, the exclusion of borderline cases is an important advantage, which can improve the distinction between different vortices. In fact, for our purposes, the investigation of a smaller region has the additional advantage of reducing the computational complexity.

For our goal of detecting the vortex axis and vortex core border, the discussion above leads to the following two conclusions. First, we can use the normalized helicity density to distinguish between different vortices. Second, we can, in principle, use one of the local vortex definitions to restrict the flow domain, to which we have to apply our adaptation algorithm. The last point has a significant impact on the number of points we have to consider for our algorithm. Finally, though the λ_2 -criterion predicts nearly the same vortex region, it involves the computation of eigenvalues. The Q criterion can be computed faster and is thus our preferred choice.

Finally, it is worth mentioning that besides the vivid discussion on a *Galilean* vortex definition, there is also an ongoing debate on a possible *objective* vortex definition, see for example the work of Haller [16]. However, all existing objective vortex criteria are based on higher order derivatives. In our situation, where we want to improve the grid resolution especially at locations, where the grid is rough, a calculation of higher order derivatives is problematic because of the lack of accuracy. Unfortunately, this automatically excludes more advanced vortex criteria.

2.2 The Vortex Axis and its Detection

The starting point for our vortex feature based adaptation algorithm will be the detection of the *vortex axis*. Unfortunately, as in the case of the vortex, there is no exact definition for the vortex axis. Banks and Singer [27] numerically integrated vortex lines, where, in addition, each newly calculated position is corrected according to a

sectional pressure minimum. Miura and Kida [27] presented a similar approach based on a minimum of a reduced pressure field. Kenwright et al. [21] concentrated on lines following the sectional maximum of vorticity magnitude. Sujudi and Haimes [35] developed a technique, where the vortex axis line is identified with locations, where the velocity vector points in the same direction as the real eigenvector of the velocity gradient tensor. They implicitly assumed that the other two eigenvalues are complex conjugated and that in the region of complex eigenvalues a fluid element rotates around the real eigenvector. Roth [30] and Peikert and Roth [28] generalized this concept and developed a so called *parallel vectors operator*, which analyzes the geometrical relations between two appropriate vector fields on grid cell faces to identify extremal lines, ridge and valley lines as well as the vortex axis. In particular, this parallel operator yields positions on the faces where the associated vectors are parallel.

Queued cell face points, where the parallel condition is satisfied, are connected to the vortex axis. However, their algorithm is sensitive to noisy data and often advanced filter techniques have to be applied to build the vortex line. Another advanced approach comes from Sahner et al. [33], who introduced a so called *feature flow field* based on the λ_2 vortex definition. A detailed description can be found in [20]. Although they have a very convincing concept, their method has the drawback that second order derivatives have to be computed.

However, in our application, the detection of the vortex axis is the most important step. All following steps will be based upon it. Since we might also have a coarse grid resolution, i.e. a rather bad resolution or even noisy data, we cannot use any method employing higher order derivatives. We also want to avoid filtering. Thus, we will apply a combinatorial vortex axis detection algorithm, which was presented in [31]. The basic idea of this algorithm can be described as follows. We introduce a cut plane and search for points on that plane having a velocity vector parallel to an associated derivative vector of the velocity. This condition defines the location where the vortex axis crosses the cut plane. The so determined points on the plane are used as starting points for a streamline integration into the whole 3D simulation domain, assuming implicitly that a vortex axis is also a streamline.

Compared to the algorithm in [31], one important improvement of our vortex axis calculation algorithm is the following one. Whenever the integration process crosses a volume cell face, we verify whether both vectors are parallel. This guarantees on the one hand that we are further integrating the vortex axis. On the other hand, we minimize the absolute error of vortex axis line integration by avoiding accumulation of integration errors. Moreover, we can use this check for parallelism as a termination criterion for the integration process. If the parallel check sequentially fails at more than a certain number of crossed cell faces then we will stop integrating. Another verification is done by repeating the complete vortex axis calculation procedure for a cut plane, which is slightly shifted in space. This allows us to detect vortex axes, which could not be found before. If a newly integrated vortex line hits the same grid cells as a formerly detected axis then this axis will not be considered. Consequently, this approach leads to the calculation of vortex axes for the main vortices, i.e. the primary, secondary, and tertiary vortices [31]. In this paper, we have tested the improved vortex axis algorithm in two cases, using two different derivative vectors of the velocity vector for our test for

parallelism. In the first case we compared the velocity vector to the acceleration vector. In the second case we compared the velocity vector to the vorticity vector. Both cases lead to nearly the same axes. However, since our vortex distinction criterion is based on the helicity density we will use the vorticity vector as input for the parallel operator.

2.3 The Vortex Core

As mentioned above, we understand a vortex to be a region with a sharp bounding hull surface, establishing a separating border between the region with a vortex and the region without a vortex. A *vortex core* is also a region. This region comprises the inner part of the vortex from the vortex axis to locations, where locally the magnitude of circumferential velocity reaches its maximum. In the outer part of the vortex, outside of the vortex core region but within the vortex hull surface, the circumferential velocity decreases again. A formal definition is given by Lugt [25], who considered solutions of the axial symmetric Navier-Stokes equations. From the analytical point of view, the maximum of the circumferential velocity magnitude denotes the vortex core border. Hence, this local maximum can be used as a suitable physical parameter to determine a hull surface comprising the whole vortex core.

Within the vortex core, the impact of dynamical forces results into a low pressure region with a minimum almost at the vortex axis. This is a result of the cyclostrophic equilibrium between the centripetal force coming from the acceleration of the fluid elements and the radial pressure gradient. Within the core region, directly at the vortex axis, the fluid rotates like a rigid body. Closer to the vortex core border, the influence of the viscosity and thus the shear part changes. Outside the vortex core but within the vortex hull, the circumferential velocity reaches an inflection point, where the vortex starts to change its basic flow behavior again. The influence of vortical motion related shear vanishes and the vortex becomes more and more a circumferential velocity distribution like a potential vortex. Another physical aspect can be investigated by observing the temporal or spatial development of the vortex core radius. When the core radius grows, the vortex decays and vorticity is transported to the outer direction resulting also into a decreasing radial pressure gradient. As a result, this leads to an increasing axial pressure gradient and a decreasing axial velocity which leads to a wake-like axial velocity component. Later on, the vortex breakdown phenomenon may occur. In the other case if the vortex core radius decreases, the vortex tube will become constricted. Then, vorticity is transported into the direction of the vortex axis, and, eventually, the vortex will become stronger with a higher circumferential velocity component. Because of this, the radial pressure gradient increases as well as the axial velocity. This finally leads to a jet like axial velocity profile.

The physical significance of the vortex core border is reflected in the fast change of the derivatives of the circumferential velocity component. If we exemplarily look at an analytically defined vortex such as the Burgers vortex, which is a solution to the axis-symmetric Navier-Stokes equations, we may get an impression of the specific requirements for a grid refinement. In Figure 1, the curvature progression of the circumferential velocity component illustrates the strong flow structure change at the vortex core border.

This shows that an a priori globally refined grid topology for the numerical simulation

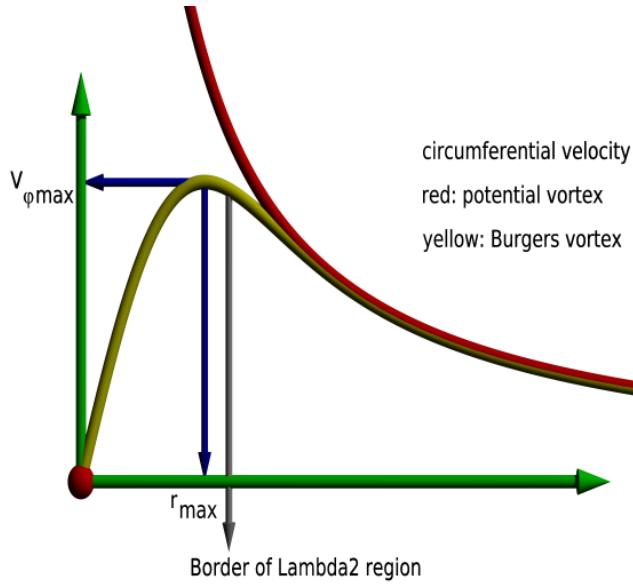


Figure 1: Sketch of the core radius and the circumferential velocity component of a Burgers vortex. The core radius is determined by the maximum of the circumferential velocity (r_{max}), while the inflection point of the velocity is the location where the Burgers vortex becomes a potential vortex [8].

is in general not possible. Thus, a dynamic grid adaptation is required.

From the discussion above, we can identify further significant difficulties we have to face. Along a developing vortex, different local circumferential velocity magnitudes will occur, as well as different vortex core radii from the current axis point position to the vortex core border. This has to be taken into account when formulating an appropriate adaptation sensor.

3 Grid Adaptation Algorithm

We are now explaining in detail which adaptation technique we will use for refining and de-refining the grid cells. After that we will formulate the vortex analysis algorithm and the sensor for our grid adaptation.

We will employ the adaptation module of the DLR TAU-Code. This module comprises three different components for various grid manipulations, which can be used to adapt a given grid to the solved flow field (see [4, 3]):

- y_+ -based grid adaptation for adjusting the first wall distance over turbulent surfaces in hybrid grids,
- hierarchical grid refinement and de-refinement for introducing new grid points using a given edge-indicator function without producing hanging nodes,

- surface approximation and reconstruction for curved surfaces after the introduction of new grid points.

For the refinement of vortices the y_+ -based grid adaptation and the surface approximation are of minor interest.

Hence, we will concentrate on using the grid refinement and de-refinement algorithm together with the edge-indicator function for detecting regions, which require local refinement or de-refinement.

3.1 The Refinement and De-refinement Algorithm

A local refinement strategy has to specify how to detect grid areas, which need refinement, and a method of element subdivisions, which are a consequence of the insertion of new points into these areas.

A very rough description of the basic refinement algorithm is given by:

1. Build the edge list and the element to edge references.
2. Evaluate the edge indicator I_e for every edge.
3. Refine the edge list, taking the edge indicators, the target point number and the grid conformity into account.
4. Calculate the coordinates of new points.
5. Construct new elements.
6. Interpolate the solution to the new points.

The refinement/de-refinement algorithm is based on an edge-based data structure. This means that the refinement indicators are evaluated for all edges. New points are chosen as the edges' mid points; the element subdivisions are determined by the configurations of the refined edges. Step 3 of the algorithm consists of two nested loops. The outer refinement loop finds the limit L for the indicator which results in the target number of new points if all edges with $I_e > L$ are initially marked for refinement. The inner loop ensures consistency by checking all elements for a valid subdivision and by inserting additional points by marking the corresponding edges in the case of a violation. This loop has to be repeated until no more edges are marked. To prevent the propagation of the local refinement through the grid all possible refinement cases for tetrahedra are implemented as well as two anisotropic directional refinement cases for prisms. For pyramids there is only a small subset of possible refinement cases implemented because of grid consistency reasons. For a detailed description of all available refinement cases see [2, 3, 4]. After the determination of the final edge subdivision all marked edges are identified with new point numbers and the coordinates of these new points are calculated (step 4). The new elements are constructed using the appropriate subdivisions in step 5. Finally, the solution is interpolated to the new points (step 6). For the de-refinement, the algorithm keeps track of the complete refinement hierarchy for each element in the grid. For our edge-based element representation this means that all parents, which are one level above the actual grid elements, have to be considered for

the calculation of the global edge-list in order to allow the edge-indicator function to select these edges for de-refinement. The preliminary step 1 of the basic algorithm is extended to the following algorithm:

- 1.1 Determine the direct parents of the actual elements.
- 1.2 Build the edge list of elements and their direct parents. Build the element to edge reference of elements/parents.
- 1.3 Restore mid points of the parent edges.
- 1.4 Flag out all edges which are not part of an isotropic refinement.
- 1.5 Flag out all edges which have a parent point for de-refinement.

Step 1.1 is easily done by flagging all parents which are referenced to by other parents and then considering only the remaining parents. Step 1.2 is basically the former step 1. The steps 1.3 – 1.5 ensure consistency of the basic algorithm. Finally, step 3 has to be modified to support de-refinement:

- 3.1 Set an initial guess for the indicator limit L .
- 3.2 Mark all parent edges with $I_e < L$ for de-refinement.
- 3.3 Mark all edges with $I_e > L$ for refinement.
- 3.4 Mark additional edges for refinement to ensure valid refinement cases (consistency loop).
- 3.5 Count the marked edges. If the number of new points is too large or too small modify L , reset all marked edges and go to 3.2.

To avoid the degeneration of elements, non-isotropically refined elements are not further refined. If such an element has to be refined, because some of its edges are marked for refinement, the direct parent element is isotropically refined and the remaining edge marks are considered for the resulting children. This guarantees stable sequences of element refinements.

3.2 The Edge-Indicator Sensor Functions

For local grid refinement/de-refinement a proper sensor or indicator is necessary. There are several approaches to define such a sensor. One of the most common approaches is to use a residual-based indicator [17]. There are also approaches to use an adjoint sensor or to use gradients or differences of an appropriate flow variable. In the implementation of the adaption module of the TAU-code, there are edge-indicator functions based upon the gradient or various differences types. These functions are defined as follows.

The approximate gradient $G\Phi$ of a function Φ along an edge e with vertices p_1 and p_2 is given by $G\Phi(e) = \Delta\Phi(e)/h$. Here, $\Delta\Phi(e)$ is the difference between the point values of Φ at the two vertices p_1 and p_2 of the edge e , i.e. $\Delta\Phi(e) = \Phi(p_1) - \Phi(p_2)$,

and $h = |p_1 - p_2|$ is the length of the edge e . Then, one way of defining an indicator function is given by setting

$$I_e = \Delta\Phi h^\alpha = G\Phi h^{\alpha+1} \quad (9)$$

with a parameter $\alpha > 0$ at our disposal. This parameter is often simply set to $\alpha = 1$ such that $I = G\Phi h^2$. An advantage of such a scaling parameter α is that, for a proper choice, the refinement will automatically stop after several iterations in the designated area. For $N_\phi + 1$ different flow variables, we choose the following modification of $\Delta\Phi$ for the indicator function along an edge e of the grid:

$$\Delta\Phi(e) = \max_{0 \leq i \leq N_\phi} \left(c_{\phi_i} \frac{\Delta\phi_i(e)}{(\Delta\phi_i)_{\max}} \right). \quad (10)$$

The weights c_{ϕ_i} are scaling parameters which give a sufficient flexibility to choose different combinations of the single flow variables of the indicator (for example, setting one c_{ϕ_i} to zero simply corresponds to “switching” ϕ_i off).

The reference values $(\Delta\phi_i)_{\max}$ are given by

$$(\Delta\phi_i)_{\max} = \max_{0 \leq j \leq N_e} \Delta\phi_i(e_j),$$

where $N_e + 1$ denotes the total number of edges in the grid and e_0, \dots, e_{N_e} is an enumeration of all these edges. The maximum values guarantee a balanced scaling of each contribution to the indicator function. As the edge indicator function we can use, for example, one of the following differences:

1. The differences (Δ_d) of the flow values

$$\Delta_d\phi_i(e) = |\phi_i(p_1) - \phi_i(p_2)|$$

2. The differences of the gradients (Δ_g) of the flow values

$$\Delta_g\phi_i(e) = |\nabla\phi_i(p_1) - \nabla\phi_i(p_2)|$$

3. The differences of the reconstructed flow values (Δ_r) to the edge mid-points

$$\Delta_r\phi_i(e) = \left| (\phi_i(p_1) + \frac{1}{2}x_e \cdot \nabla\phi_i(p_1)) - (\phi_i(p_2) - \frac{1}{2}x_e \cdot \nabla\phi_i(p_2)) \right|,$$

where $x_e = p_1 - p_2$.

Any of the flow variables of the solver output can be used with these edge-indicators. This means all primitive and all additionally user-defined flow variables can be employed (e.g. x-, y-, z-velocity, total pressure, vorticity, pressure coefficient cp, etc.).

Unfortunately, for the refinement of vortex cores and vortex borders this technique is insufficient. In [1], we have shown how to use the above described sensor function with the total pressure p_{tot} as the flow variable to automatically refine a given grid in regions of total pressure losses. With this method we were able to resolve all significant flow features, but it was necessary to adjust additional parameters to steer the refinement process. Furthermore, that method is less precise than the one presented in this paper. In order to employ the sensor described in section 4 with the adaptation tool, a new indicator function has to be developed. The edge-indicator defined in (10) will be redefined as

$$\Delta\Phi(e) = \max(\phi(p_1), \phi(p_2)). \quad (11)$$

4 The Vortex Analysis and the Adaptation Sensor

In the sequel, we will explain the main part of our vortex axis and core border adaptation algorithm. Having in particular hydro- and aerodynamical applications in mind, we will, for simplicity, restrict ourselves to columnar vortices. This means particularly that we can assign a rotational sense to each vortex.

As lined out in the last section, we have to construct an adaptation sensor which delivers signals for refining those grid cells, where the numerical flow solution indicates a vortex axis or the vortex core border.

The main idea of our adaption algorithm is to assign a value between zero and one to each vertex of the given grid. Cells having a vertex with a value larger than a given threshold will then be considered for refinement.

However, if we consider the typical graph of the circumferential velocity component of the Burgers vortex as a function of the radius then we see that the radial velocity derivative becomes zero at the vortex core border, because the maximum of the circumferential velocity is attained here. Before and behind the core border the gradients are very steep. Thus, we have to adapt those cells with small radial velocity gradients in contrast to the usual procedure to adapt cells with steep gradients. Therefore we have to construct an algorithm which is not based on derivatives, but also delivers signal for indicating the grid cells or their edges.

The main steps of our algorithm can be summarized as follows.

1. To enhance efficiency, a filtering step based on the Q -criterion ($Q > 0$) is employed to identify points that are located within a vortex. Only these points are considered in the following steps. The other points and thus their corresponding grid cells will be marked as not adaptable.
2. The vortex axes locations are computed. This is done by the following sub-steps.
 - (a) First a cut plane is calculated. Its base point can, for example, be determined by finding a global minimum of the λ_2 values or by finding the global maximum of the Q values. The plane's normal may be defined by the free stream velocity direction. This gives reasonable results especially for delta wing applications. In some cases it might be necessary to set more than one cut plane depending on the configuration and the flow separation. By cutting the elements of the tetrahedralized grid we finally get a plane consisting only of triangles.
 - (b) Then, we map the velocity and the derived vector field onto the plane. The derived vector field could be the acceleration, the real eigenvector of the velocity gradient tensor or the vorticity field. Its actual choice depends on the expected curvature of the vortex axes (see the discussion in Roth [30]). Moreover, the eigenvalues of the velocity gradient tensor are also mapped onto the plane, providing further vertex information.
 - (c) Next, we apply the parallel vector operator only to those triangles of the cut plane, where the vertices have a velocity gradient tensor with complex eigenvalues. In the case of existence, the parallel operator yields the positions

on the triangles where both vectors are parallel. These positions are used as the starting points of a streamline integration for the vortex axes. The integration is done in both directions, upstream and downstream, such that the streamlines extend to both directions.

- (d) During the integration procedure grid cells are crossed and marked as processed. Whenever a grid cell is left and a new cell is reached the integration process stops at the cell side face and the parallel operator is applied to this face; the integration process is restarted with the new initial position given by the parallel operator. This avoids the accumulation of integration errors and hence improves the accuracy of the algorithm.
- (e) The integration process stops when the parallel operator does not find any further positions. However, since we handle noisy data we allow a certain number of failure signals when crossing a cell face.
- (f) Depending on the noise level of the data the whole vortex axis calculation process (steps (a) to (e)) is repeated for a new cut plane slightly shifted to the original one. Only when the new vortex axis integration does not hit marked grid cells the newly integrated vortex axis is accepted.

All cells which are crossed by a vortex axis are marked for adaption.

3. Once the vortex axes are defined, each grid point with $Q > 0$ is assigned to an appropriate vortex using the normalized helicity density, proximity, and local flow direction.
4. To determine the vortex core radius, for each assigned grid point, the sensor value is computed using the local tangential velocity normalized by the local circumferential velocity for the vortex. Any values above a given threshold are set to unity.

The last two steps are explained in detail in the rest of this section.

4.1 Rotational Sense of Vortices

As we have seen, the vortex core can only be detected when the associated vortex axis is already known. However, for general complex flow configurations more than one axis may and will occur. In fact, primary dominant vortices are accompanied by secondary or tertiary vortices or by other up-rolling vortical structures generated by instabilities in the flow field. In addition to that, vortex interaction is inevitably enforced by the vortex induction law. Therefore, we have to distinguish between single vortices to be able to calculate the right vortex core. As outlined in Section 2.1, the normalized helicity density criterion can be used to determine the rotational sense of the vortex. Positive values in a flow region indicate a clockwise rotation of the fluid elements around the vortex axis, when looked at the flow in down stream direction. Accordingly, negative values in down stream direction indicate counter-clockwise rotation. This gives the two principal types of vortical structures: clockwise and counter-clockwise rotating vortices. Thus, we calculate the normalized helicity density for all points in a vortical region. Additionally, we interpolate the normalized helicity density onto the vortex axis points.

4.2 Point to Vortex Assignment

After having determined the axes, we have to assign the single grid points to their *associated* vortex axis. This can be considered as a specific *nearest neighbor* problem. Since each vortex axis is given by a set of points, we can approximately determine the distance of a point to a given vortex axes by finding its nearest neighbor from such a set.

To find the nearest neighbors, we first built a search tree structure for the points of each vortex axis separately using *kd-trees*. Such trees can be built efficiently and each nearest neighbor search can be executed efficiently, see for example [6]. Then, for a given grid point, the results of the searches can be compared and the axes with the least distance is a good candidate for the point to be assigned to.

As a further improvement, the vortex axis point search trees are sorted into two lists: one list for clockwise and one list for counter-clockwise rotating vortices. Since the normalized helicity density at the current grid point indicates its rotation orientation, we actually only have to compare with those axes, which are in flow regions with equally signed normalized helicity. This allows us to reduce the computational complexity even further.

However, though we can determine the right type of vortex axis in this way, we are now confronted with another problem. Often vortical flow structures are dominated by a strong primary vortex accompanied by sub-vortices with the same rotational sense which, as a consequence, have the same helicity density sign. An impression of such vortical structures can be gained when looking at the flow over a delta wing in the cut plane topology, which is depicted in Figure 14. Unfortunately, the sub-vortices are significantly smaller in their core radii than the primary vortex. Consequently, a grid point may have a shorter distance to a sub-vortex axis, but belongs actually to the core of the primary vortex axis. Thus, we cannot simply use a point to vortex axis point distance algorithm, as described above. We also have to take the different scales of the core radii into account.

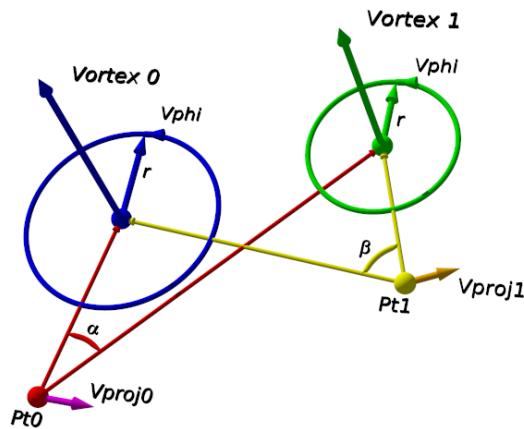


Figure 2: Sketch of the principal flow geometry between co-rotating vortices

To this end, we introduce a geometrical method to identify the correct vortex axis. For a given grid point, we first determine the *two* nearest vortex axes of equal rotation as described above. Next, we connect the grid point to each of the nearest neighbors from both vortex axes. This gives us two vectors and our criterion will be based upon the angle between them. If this angle is less than 90 degrees then we assign the grid point to the nearest vortex axis. Figure 2 shows this geometrical relation of the principal flow behavior.

Otherwise, if the grid point is located between both vortex axes and the angle is larger than 90 degrees then we use the local rotational orientation as additional information to determine the correct vortex axis. This case is illustrated in Figure 3.

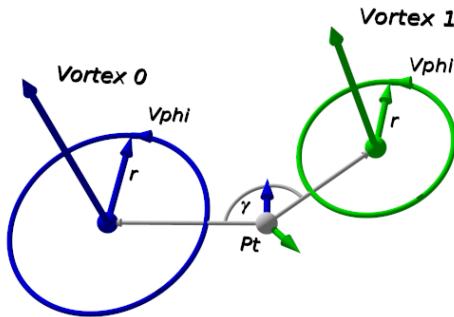


Figure 3: Critical point - vortex axis assignment case

Let us consider a two dimensional projection of the flow between the two co-rotating vortices equipped with an appropriate coordinate system. Then, we can observe that the flow of one vortex is going upwards and, separated by a strong shear layer, the flow of the other vortex is going downwards (see Figure 4). We use this observation in our assignment algorithm.

To this end, we introduce a local coordinate system for each vortex axis. The local radial unit vector \vec{e}_r of such a coordinate system is given by the normalized vector being orthogonal to the vortex axis and pointing from the vortex axis to the grid point. The axial unit vector \vec{e}_z points from the vortex axis into downstream direction along the vortex axis. By calculating the cross product of these two vectors we can define the circumferential direction vector

$$\vec{e}_\varphi := \frac{\vec{e}_z \times \vec{e}_r}{|\vec{e}_z \times \vec{e}_r|}.$$

Next, for both axes, we project the velocity vector of the given grid point onto the plane spanned by \vec{e}_r and \vec{e}_φ and compute the inner products between the projected velocity vector and the circumferential unit vectors \vec{e}_φ . Now there are two possibilities depending on the angle between \vec{e}_φ and the projected vector and the rotation orientation of the vortices. First, let us consider two vortices rotating counter-clockwise in down stream direction, as shown in Figures 4 and 5. Here, the grid point belongs to that region, where the angle is less than 90 degrees or, equivalently, the inner product is positive. If the inner product is negative, the grid point does not belong to this vortex.

In the same way, for two vortices rotating clockwise in down stream direction, the grid point belongs to that vortex region, where the angle is larger than 90 degrees, or where the inner product is positive.

The integration is done in both directions, upstream and downstream.

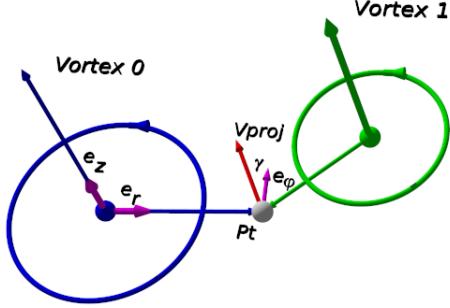


Figure 4: Sketch of the geometry of the point assignment algorithm. Point Pt is assigned to vortex 0

This criterion requires that both vortex axes are more or less parallel, which is the case for the dominant vortices above a delta wing. In flow fields where neighboring vortex axes are tilted we have strong shear flows in between. This flow region is excluded by the vortex criteria explained above. Therefore the algorithm can be extended by the constraint that a point can only be assigned to an axis if all grid points between the point and axis are assigned to a vortical region by using one of the explained criteria. In this way, we can uniquely assign each grid point to a certain vortex axis point.

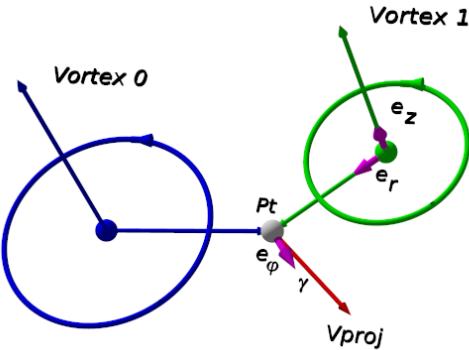


Figure 5: Sketch of the geometry of the point assignment algorithm. Point Pt is assigned to vortex 1

During the assignment process, we count how often each vortex axis point is employed. This additional information will later on be used for the normalization of the circumferential velocity components.

4.3 The Vortex Core Border Detection

With assigning each grid point to its vortex axis point, we have also for each point in a vortical region the appropriate local vortical coordinate system. This is used to determine the circumferential component of the velocity vector at each grid point. For constructing our unique sensor value needed by our adaptation algorithm, we have to normalize these velocity components. In order to obtain a normalization factor we construct for each grid point a ray pointing from the associated vortex axis point over the grid point itself outwards in radial direction. On this ray we calculate the maximum circumferential velocity component going stepwise from the axis outwards to the border of the vortical region. The step-size is determined by a 5th of the local grid cell size. Now we can normalize the circumferential velocity components of all grid points by the calculated maximum. After all these steps we get the adaptation sensor field consisting of values from zero as lower limit to values close to one. Values in a range close to one indicate the vortex core border and as mentioned above also the vortex axis. Hence these values are used to mark the grid cell edges for refinement. For further simplification of the edge-cell marking we set all values larger than 0.98 to one. Then we are able to apply our grid adaptation.

From the practical point of view, it is clear that the vortex core border can not be well resolved in a single grid refinement step. Thus, in the first adaptation step the threshold for the sensor has to be explicitly lower than 1, but it can be gradually increased step by step.

5 Application and results

In this section, we present the results of our grid adaptation algorithm by applying it to several examples.

In the first application, we consider an analytical flow in a cylinder to explain the main aspects of our approach. We start with a very coarse unstructured tetrahedral grid, on which we define the analytical vortex. The cylinder has a diameter of 16m and a height of 2m. The initial grid has an average grid cell edge of 0.05 m.

An impression of the spatial grid resolution can be gained from Figure 6, where the cell distribution is illustrated for one slice and the outer boundary of the cylinder.

As the basic flow field we use two simple Burgers vortices

$$\begin{aligned} v_r &= -ar \\ v_\varphi &= \frac{\kappa}{r} \left[1 - \exp \left(-\frac{ar^2}{2\nu} \right) \right] \\ v_z &= az \end{aligned} \quad (12)$$

with the kinematic viscosity $\nu = 1.5 \cdot 10^{-5} m^2 s^{-1}$. The parameter a is set to $a = 0.2 s^{-1}$. In our test cases the vortex strength κ is varied to simulate co-rotating and counter-rotating vortices with different strengths and hence different interactions. We choose a distance of 10m between the analytical middle points of the vortices.

For the first vortex, the vortex strength is given by $\kappa = 6 m^2 s^{-1}$, while it is $\kappa = 4 m^2 s^{-1}$ for the second vortex. These values are indicating two co-rotating vortices

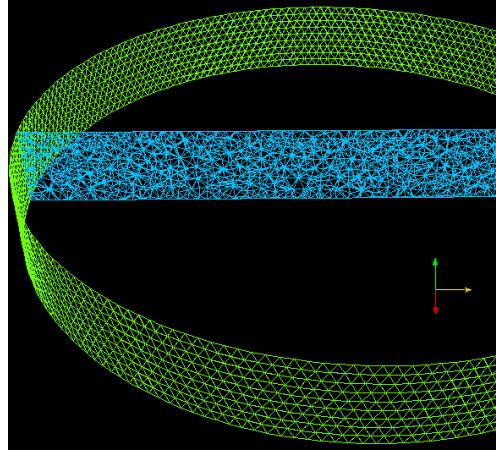


Figure 6: Slices through the unstructured tetrahedral grid

with comparable strength. The numerically calculated vortex axes have hardly shifted in comparison to the predefined centers of the analytical vortices. Figure 7 shows one slice planar through the grid, color-coded by the invariant Q of the velocity gradient tensor and elevated by the values of our vortex core adaptation sensor. Further details are shown in Figure 8.

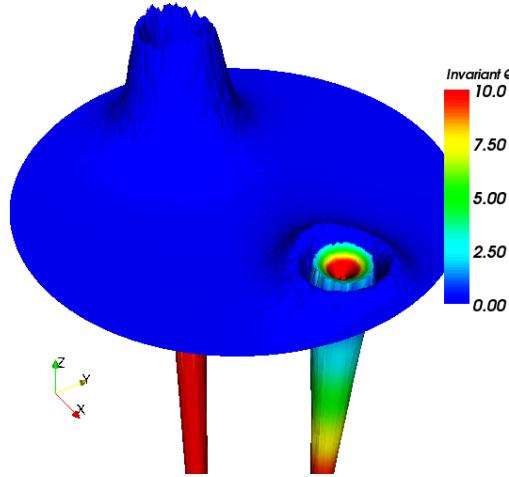


Figure 7: Velocity magnitude on a slice, grid structure elevated by the calculated vortex core adaptation sensor

The given ring like structure of both vortices is well represented and looks like the structure of a volcano crater. Moreover, as pointed out above, the Q values are significantly different from zero at the ridge line of the crater. However, the spatial resolution is obviously low which is revealed by the spiky contour of the raised cells. Nonetheless, the sensor delivers the same strong signal for both vortices for cell refine-

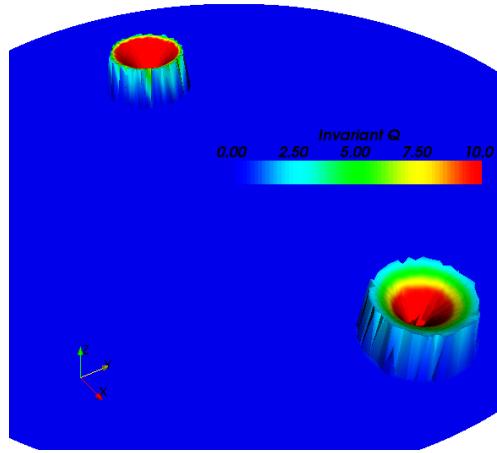


Figure 8: Co-rotating Vortices, cutting plane elevated by vortex core adaptation sensor values, ridge line points have values of one, additional color coding by the invariant Q of the velocity tensor

ment by values close to one. The result of the following cell adaptation step is depicted in Figure 9. The locally refined zone is clearly visible as a ring shaped crater lake. In this figure, to give a better impression of the vortex core, those cells are excluded, which are associated to the vortex axes.

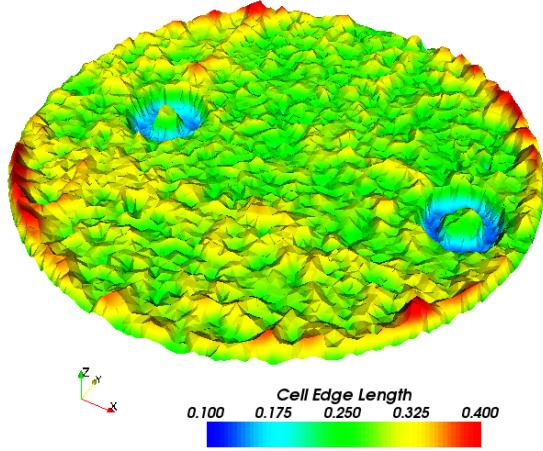


Figure 9: Co-rotating vortices, cell edge lengths after local grid refinement in cut plane, elevation scaled by 5 times the edge lengths

In the second case, we consider two counter-rotating vortices. Thus, we set the vortex strength to $\kappa = 40 \text{ m}^2\text{s}^{-1}$ for the first vortex, and to $\kappa = -8 \text{ m}^2\text{s}^{-1}$ for the second vortex. The vortex strengths have a significantly different magnitude which leads to a strong interaction of these vortices. As a result, the stronger vortex shifts the vortex axis of the other vortex by 0.55m outwards while its axis moves only by 3cm .

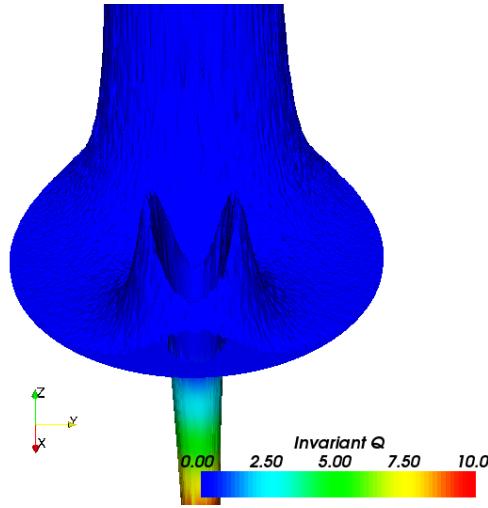


Figure 10: Vortical structures of two counter rotating vortices with strongly different vortex strength

Figure 10 gives an impression of the local flow structure. Here we choose the vortex definition based upon the invariant Q again to highlight the asymmetry of the secondary vortex.

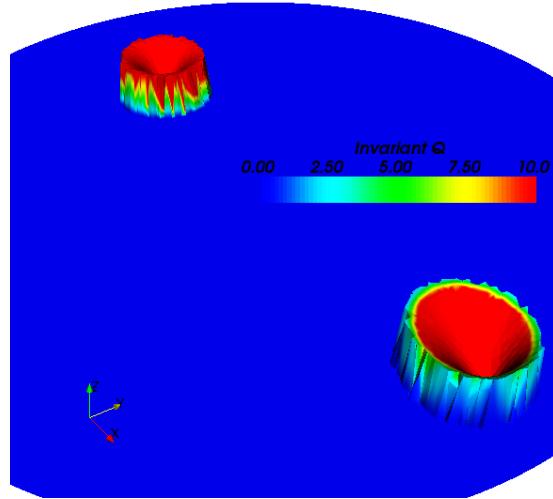


Figure 11: Counter rotating vortices, cutting plane elevated by vortex core adaptation sensor values, ridge line points have values of one, additional color coding by the invariant Q of the velocity tensor

Figure 11 shows the results analogous to Figure 8; here, the calculated vortex core adaptation sensor is used to elevate the vortex core. In contrast to the definition by the invariant Q , our algorithm finds the core border very precisely even in this highly asymmetric case. However, in our algorithm it is important to take the shifts of the

vortex axes also into account. Using the original axes which are employed to initiate the velocity field leads naturally to wrong results. This can be explained by analyzing the vorticity flux rate in the vortical regions.

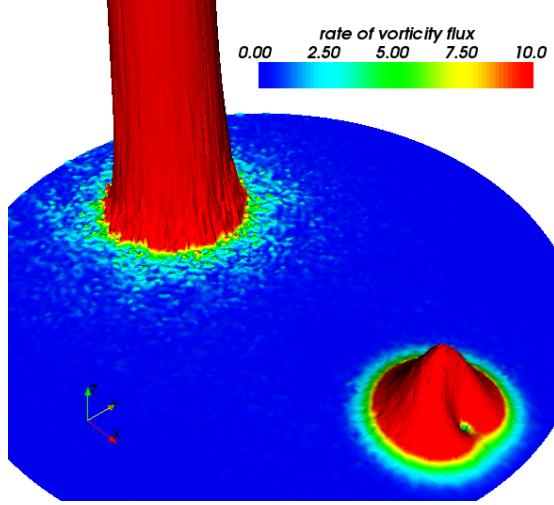


Figure 12: Elevation of the vorticity flux rate on a cutting plane

Figure 12 shows the effect of the vortex interaction on the vorticity flux rate. For the weaker vortex the asymmetric location of the vortex axis correlates with the spot-like decline of the vorticity flux rate. Despite this asymmetry, we receive a similar result as in Figure 9 when we apply our adaptation module, see Figure 13.

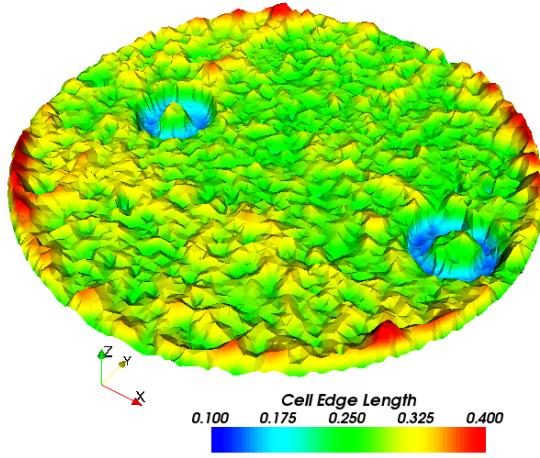


Figure 13: Counter rotating vortices, cell edge lengths after local grid refinement in cut plane, elevation scaled by 5 times the edge lengths

Again, we used the cell edge lengths to visualize the refined vortex core border grid region.

Next, we want to concentrate on the more realistic case of a pitching delta wing, which has been numerically computed. The flat, 65 degrees swept, generic prismatic delta wing has a chord length of 1 m and a height of 5 mm. The delta wing has sharp edges with an edge angle of 15 degrees. On account of this, the leading edge separation of the flow is exactly predefined and a wavy vortex is suppressed. We performed an *Unsteady Reynolds Averaged Navier-Stokes* (RANS) simulation on a Linux cluster using the DLR finite volume solver TAU for compressible flow [5], which has a second order spatial and temporal discretization. The computations were performed using a highly pre-adapted [1], hybrid grid, which consists of 3 million points and 11 million cells, 8.8 million tetrahedra and 3 million prisms in 24 layers. The on-flow Mach number was 0.2 and the Reynolds number was 1.2 millions. For turbulence modeling, the Spalart-Almaras turbulence model with the Edwards modification was used, which shows adequate results. Time accurate simulations were performed with a time step of 0.0001 seconds. The angle of attack was varied from 16 to 24 degrees. The simulation has been sped up by flux splitting to perform inner sub-iterations. In addition, a 3-v multi-grid cycle for solving the linear systems has been performed to further accelerate the computations. Since another goal of our investigation was to analyze the symmetric or asymmetric behavior of the vortical delta wing flow, the assumption of symmetry in a plane could not be used and therefore, the flow field of the full delta wing configuration had to be computed. The flow field of such a symmetric delta wing, which is typical of modern fighter aircrafts, consists of large scale vortical structures on its lee side. Typically there are two primary vortices and smaller secondary, tertiary and other subtype vortices, which are strongly interacting. Normally, the primary vortices are dominating the overall flow structure yielding the main part of the additional lift apart from the potential lift of the attached flow, see the sketch in Figure 14.

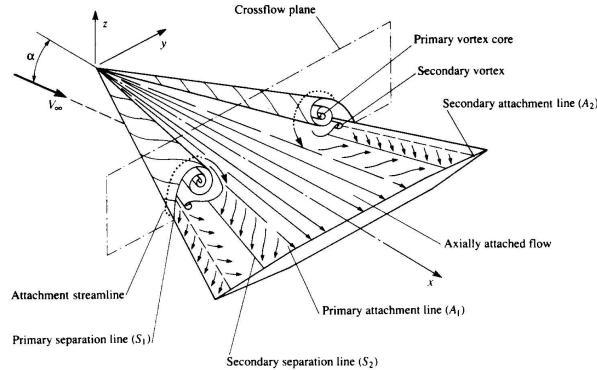


Figure 14: Sketch of the vortical flow field above a pitching delta wing

For the rest of this example, we will concentrate on an angle of attack of 24 degrees. Although, at this angle of attack, the vortex breakdown phenomenon occurs, which results eventually in a vortex breakdown bubble, see Figure 15, we are particularly interested in resolving the stable flow field in front of the breakdown flow structures. Therefore, we decided to reduce the grid region, where we want to apply our algorithm.

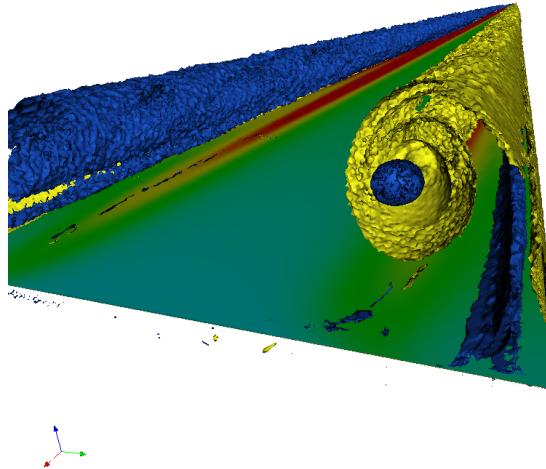


Figure 15: Vortex breakdown bubble visualized by iso-surfaces of the normalized helicity density criterion

Thus, we choose a representative region, where the dominant primary and secondary vortices are stable and no vortex breakdown occurs. The cutout-box is depicted in Figure 16.

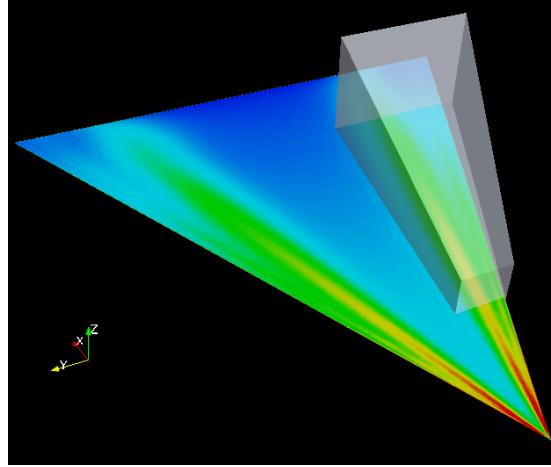


Figure 16: Cutout-box of the grid adaptation region

For this region, we have generated a tetrahedral mesh with nearly isotropic grid cell sizes, on which we interpolated the flow data and to which we applied our adaptation algorithm. The principle flow structure within this box is illustrated in Figure 17, which shows the color coded normalized helicity distribution on a slice highlighted by the vortex Q criterion. Additionally, the vortex axis for the primary and secondary vortex are depicted.

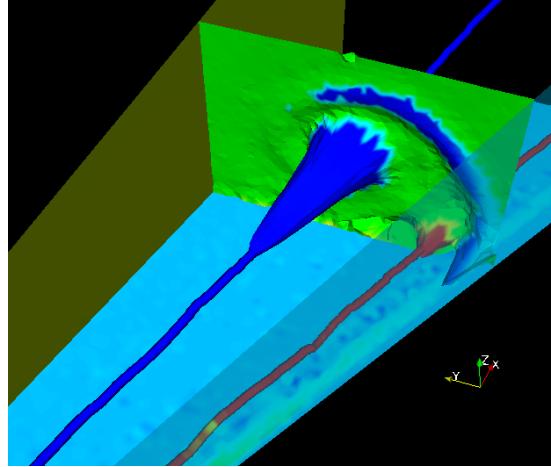


Figure 17: Principal flow structure in the cut out subregion, normalized helicity distribution on a slice highlighted by the vortex Q criterion, primary (blue) and secondary (red) vortex axis

To compare our proposed algorithm to a vortex-feature algorithm based upon scalar values, we first adapt the grid using the vortex Q criterion as the adaptation indicator. Figure 18 shows the results for this case. The normalized helicity density distribution on a slice has been color coded and amplified by the average cell edge length. The grid adaptation leads to a region with smaller cell sizes, which are less amplified than unrefined cells.

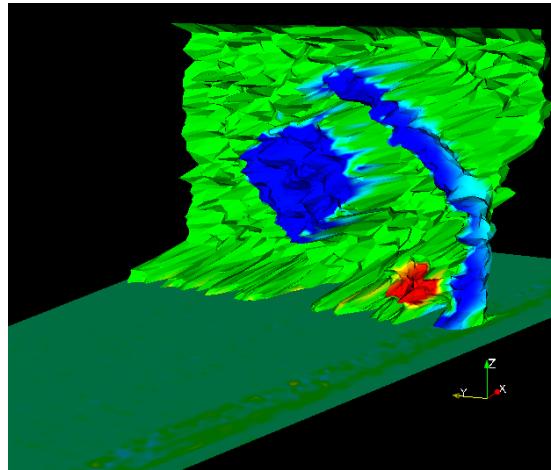


Figure 18: Cut through the adapted grid, vortex Q used as adaptation indicator, color coded normalized helicity density distribution warped by the middle cell edge length

Besides the expected finer resolution of the primary and the secondary vortex, using the vortex Q indicator leads also to the additional refinement of the vortex shedding flow structure. In contrast to the vortex Q indicator based adaptation our proposed algorithm is able to distinguish between the shedding flow structure and the vortices

and only refines the vortical region. Figure 19 contains the results of our algorithm, using the same visualization techniques as in Figure 18.

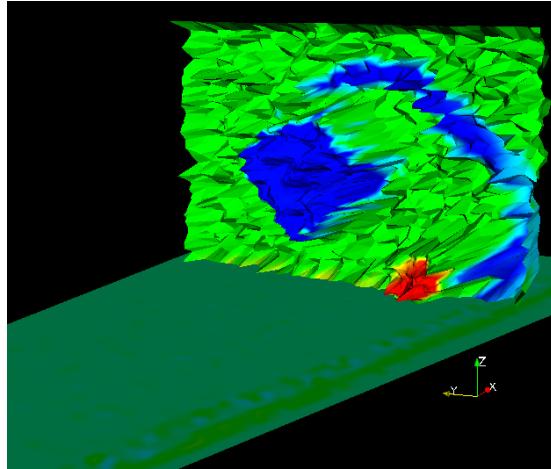


Figure 19: Cut through the adapted grid, vortex feature based adaptation, color coded normalized helicity density distribution warped by the middle cell edge length

Obviously, our proposed algorithm can be used to refine vortex related grid regions selectively, showing one remarkable advantage of our method. Moreover, the refined region is slightly smaller than the refined region in the first case. Finally, note that cells crossed by the vortex axis are also refined, which leads to a nearly identical grid structure in the centers of the vortices.

6 Conclusion

In this paper, we have proposed a feature-based vortex core border grid adaptation technique, which is based upon the idea of finding the maximum of the circumferential velocity. After introducing well known vortex definitions and criteria, we have outlined our algorithm in detail focusing on those vortex features necessary for our adaptation indicator. In particular, our vortex axis detection and our vortex core border calculation are crucial for our adaptation approach.

Another important point concerned the principles of our cell handling, in particular the edge refinement algorithm and the derivation of new cells. To this end, we described the cell development step by step, which allows the reader to understand our usage of the adaptation indicator. Since we think that also a de-refinement should be part of an adaptation strategy to allow a development of the flow solution and to avoid a freezing of an underdeveloped solution, we sketched our implementation of cell de-refinement. After that, we concentrated on simple test cases of co- and counter-rotating vortices. They demonstrated that our algorithm is able to detect and refine the grid region of the vortex core border even when the interaction of vortices leads to distorted vortical structures. Furthermore, we have shown that our algorithm is able to distinguish between real vortices and vortex sheets. Finally, we have shown that our algorithm

does not only work for simple test cases but also for realistic problems. However, so far we purely concentrated on the geometric topology of the grid and used a given flow solution to calculate our adaptation indicator. Thus, we have to remark that this is only one side of the problem of resolving flow field features. The other side is the flow solution itself. In particular the roughness of the solution has to be considered in future work on grid adaptation. We intend to develop algorithms for identification and reconstruction of vortical flow features under the circumstances of a bad spatial resolution.

References

- [1] T. Alrutz and M. Rütten. Investigation of Vortex Breakdown over a Pitching Delta Wing applying the DLR TAU-Code with Full, Automatic grid adaptation. Paper 5162, 35th AIAA Fluid Dynamics Conference, 6-9 June 2005. Toronto.
- [2] Thomas Alrutz. Erzeugung von unstrukturierten Netzen und deren Verfeinerung anhand des Adaptationsmoduls des DLR-TAU-Codes. Diplomarbeit, Universität Göttingen, 2002.
- [3] Thomas Alrutz. *MEGAFLOW — Numerical Flow Simulation for Aircraft Design Results of the second phase of the German CFD initiative MEGAFLOW presented during its closing symposium at DLR, Braunschweig, Germany, December 10th and 11th 2002*, volume 89 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, chapter 7 Hybrid Grid Adaptation in TAU, pages 109–116. Springer Verlag, Berlin, 2005.
- [4] Thomas Alrutz and Orlt Matthias. Parallel dynamic grid refinement for industrial applications. In P. Wesseling, E. Onate, and J. Periaux, editors, *Proceedings ECCOMAS 2006*, September 5-8 2006. Egmond aan Zee, The Netherlands.
- [5] Dieter Schwamborn, Thomas Gerhold, and Roland Kessler. The DLR-TAU Code - an Overview. In *Proceedings ODAS 99*, 1999. ONERA-DLR Aerospace Symposium 21-24 June in Paris, France.
- [6] Arya, S. and D. M. Mount: Approximate nearest neighbor searching, In Proc. 4th Ann. ACM-SIAM Symposium on Discrete Algorithms, ACM Press, New York, 271-280, 1993.
- [7] Banks, D., Singer, B.: Vortex tubes in turbulent flows: Identification, representation, reconstruction. In *Proceedings of IEEE Visualization '94*, pp. 132-139, 1994
- [8] Burgers, J. M.: A mathematical model illustrating the theory of turbulence. *Adv. Appl. Mech.* 1, pp. 171-199, 1948
- [9] Chakraborty, P., Balachandar, S., Adrian, R. J.: On the relationships between local vortex identification schemes. *J. Fluid Mech.*, vol. 535, pp. 189-214, 2005
- [10] Chong, M. S., Perry, A. E., Cantwell, B. J.: A general classification of three-dimensional flow fields. *Phys. Fluids A*, vol. 2, no. 5, pp. 261-265, 1990

- [11] Cucitore, R., Quadrio, M., Baron, A.: On the effectiveness and limitations of local criteria for the identification of a vortex. *Eur. J. Mech. B / Fluids* 18, vol. 4, pp. 261-282, 1999
- [12] Dallmann, U.: Three-dimensional vortex structures and vorticity topology. In Proceedings of the IUTAM Symposium on Fundamental Aspects of Vortex Motion, Tokyo, Japan, 1987
- [13] Dallmann, U.: Analysis of Simulations of Topologically Changing Three-Dimensional Separated Flows. IUTAM Symposium on "Separated Flows and Jets", Novosibirsk, 1990.
- [14] Dubrue, Y., Delcayre, F.: On coherent-vortex identification in turbulence. *J. Turbul.*, vol. 1, pp. 1-22, 2000
- [15] Gerhold, T., Evans, J.: Efficient Computation of 3D-Flows for Complex Configurations with the DLR Tau-Code Using Automatic Adaptation. *Notes on Numerical Fluid Mechanics*, vol. 72, Vieweg, 1999
- [16] Haller, G.: On objective definition of a vortex. *J. Fluid Mech.*, vol. 525, pp. 1-26, 2005
- [17] Habashi, W.G., Dompierre, J., Bourgault, Y., Fortin, M., Vallet, M.-G.: Certifiable Computational Fluid Dynamics Through Mesh Optimization. *AIAA Journal*, vol 36 (5)pp. 703-711, 1998
- [18] Hunt, J.R.C., Wray, A.A., Moin, P.: Eddies, streams and convergence zones in turbulent flows. In Center of Turbulence Research Report CTR-S88, pp.193-208, 1988
- [19] Jeong, J., Hussain, F.: On the identification of a vortex. *J. Fluid Mech.*, vol. 285, pp. 69-94, 1995
- [20] Jiang, M., Machiraju, R., Thompson, D.: Detection and Visualization of Vortices. In *Proceedings of IEEE Visualization 2002*, pp. 413-416, 2002
- [21] Kenwright, D. N., Haimes, R.: Vortex Identification - applications in aerodynamics: A Case study. In *Proceedings of IEEE Visualization '97*, pp. 413-416, 1997
- [22] Kida, S., Miura, H.: Identification and analysis of vortical structures. *Eur. J. Mech. B / Fluids* 17, vol. 4, pp. 471-488, 1998
- [23] Levy, Y., Degani, D., Seginer, A.: Graphical visualization of vortical flows by means of helicity. *AIAA Journal*, vol. 28, no. 8, pp 1347-1352, 1990
- [24] Lupt, H. J.: The dilemma of defining a vortex. In U. Müller, K.G. Roesner, B. Schmidt(eds.), *Theoretical and Experimental Fluid Mechanics*, pp. 309-321, Springer-Verlag, 1979
- [25] Lupt, H. J.: *Introduction to Vortex Theory*. Vortex Flow Press, Inc., Maryland, 1996

- [26] Perry, A. E., Chong, M. S.: A description of eddying motions and flow patterns using critical-point concepts. *Ann. Rev. Fluid Mech.*, vol. 19, pp. 125-155, 1987
- [27] Miura, H., Kida, S.: Identification of central lines of swirling motion in turbulence. In *Proceedings of Intl. Conference on Plasma Physics*, Nagoya, Japan, pp. 866-896, 1996
- [28] Peikert, R., Roth, M.: The “parallel vectors” operator - a vector field primitive. In *Proceedings of IEEE Visualization '99*, San Francisco, CA, 1999
- [29] Robinson, S. K.: Coherent motions in the turbulent boundary layer. *Ann. Rev. Fluid Mech.*, vol. 23, pp. 601-639, 1991
- [30] Roth, M.: *Automatic Extraction of Vortex Core Lines and Other Line-Type Features for Scientific Visualization*. PhD thesis, Swiss Federal Institute of Technology Zrich, 2000
- [31] Rttén, M.: Vortex Axis Calculation by Using Vortex Features. AIAA Paper 2004-2353, 34th AIAA Fluid Dynamics Conference and Exhibit, Portland, Oregon, 2004
- [32] Saffman, P. G.: *Vortex Dynamics*, Cambridge University Press, 1992
- [33] Sahner, J., Weinkauf, T., Hege, H.-C.: Galilean Invariant Extraction and Iconic Representation of Vortex Core Lines. In *Eurographics - IEEE VGTC Symposium on Visualization*, 2005
- [34] Singer, B., Banks, D.: A predictor-Corrector Scheme for Vortex Identification. *NASA Contractor Report 194882, ICASE Report No. 94-11*, NASA Langley Research Center, Hampton, VA, Mar. 1994
- [35] Sujudi, D., Haimes, R.: Identification of swirling flow in 3D vector fields. AIAA Paper 95-1715, 12th AIAA CFD Conference, San Diego, CA, 1995
- [36] Truesdell, C.: *The Kinematics of Vorticity*. Indiana University, 1953
- [37] Wu, J. Z., Xiong, A. K., Yang, Y. T.: Axial stretching and vortex definition. *Phys. Fluids* 17, 038108, 2005

(F)

**Investigation of the parallel performance of the
unstructured DLR-TAU-Code
on distributed computing systems**

Thomas Alrutz

*DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

Proceedings of the 17th Parallel CFD Conference 2005,
Pages 509–516, Elsevier

Abstract

High Performance Computing is widely used in the Computational Fluid Dynamics (CFD) community to satisfy the increasing needs for a faster prediction of simulated flows. Despite the advances in computer manufacturing and chip design over the last years, there is still a lack of computational resources required to solve CFD problems in real time. Moreover, for most research centers it is difficult to buy a supercomputer like the NEC-SX 6 based Earth Simulator out of their budgets. To encounter this gap, there is an increasing installation of supercomputers built out of commodity components like PC's or workstations connected via high speed networks. These cluster systems often lack tight integration between the processor, network and main memory. This often results in inefficiencies in the communication subsystem, such as high software overheads and/or message latencies. In this paper we investigate the parallel performance of the unstructured DLR TAU-Code on a set of up-to-date Linux-Clusters. The TAU-Code flow solver is parallelized by domain decomposition and uses MPI-1 for the inter domain communication. The influence of the different cluster configurations on the parallel performance is investigated with two real life applications, which are typical for the day-to-day work of a CFD-engineer or researcher. The parallel efficiency of the TAU-Code is investigated with two different versions of cache optimization techniques, which will have a significant impact on the scalability of the code. The potential of the installed high speed network to handle the enormous amount of data transfer caused by a parallel CFD calculation will be evaluated to determine the most promising platform for this type of CFD-Code.

1 Introduction

The improvements in high speed networks, and the exponential increase in processor performance over the last years, have made commodity clusters a growing alternative to classical supercomputers (vector computers or shared memory systems). On the other hand most of the systems built out of commodity components lack tight integration between processor, memory and network. Along with these drawbacks another problem of the commodity systems is that the used hardware is not designed to support parallel computation. Therefore the latency of the messages sent with a MPI [8] implementation is poor compared to classical supercomputers [2]. Thus the benefit of a commodity cluster installation depends highly on the applications that will run on such a system. For common CFD codes the requirements on the processor main memory and the network interconnection are high [10], especially if the code is an unstructured code which uses multigrid as a convergence accelerator [7]. In this paper we evaluate the parallel efficiency of the unstructured DLR TAU-code on a set of Linux-clusters with two typical aerodynamic configurations. For the investigation of the limits of the used high speed network, we apply two different versions of cache optimization integrated in the code. Furthermore we provide a guideline how to test the currently available hardware to select the most promising platform.

2 Performance evaluation method

2.1 The TAU-Code

The DLR TAU-Code is a finite-volume Euler/Navier-Stokes solver working on unstructured hybrid grids. The code is composed of independent modules: Grid partitioner, preprocessing module, flow solver and grid adaptation module. The grid partitioner and the preprocessing are decoupled from the solver in order to allow grid partitioning and calculation of metrics on a different platform than used by the solver. The flow solver is a three-dimensional finite volume scheme for solving the Unsteady Reynolds-Averaged Navier-Stokes equations. Further details of the TAU-Code may be found in [5].

2.2 Flow solver settings

For the performance evaluation a RANS calculation for both of the aerodynamic configurations is selected. The following settings for the flow solver reflects a standard parameter setup for most of the calculations computed with TAU in production:

- Turbulence model: Spalart Allmaras
- Solver type: central scheme
- Number of Runge-Kutta steps: 3
- Multigrid: 4w cycle
- Number of iterations: 30 (only for benchmark)

Due to the way the solver is parallelized, there is an exchange of messages between the domains on every grid level. Thus rather small messages are exchanged on the coarsest grid level and larger messages on the finer grid levels. Keeping this in mind, we can investigate the influence of latency and bandwidth with respect to message passing on the parallel efficiency of the code with the selected multigrid cycle.

2.3 Aerodynamic applications

In order to simulate the requirements on a CFD code by the typical day-to-day work of a CFD-engineer/researcher, we selected the DLR F6 configuration presented in [6] and a A380 configuration obtained from Airbus industries. The DLR F6 (figure 1) is a wing/body/pylon/nacelle configuration from which a hybrid unstructured grid with 2 million grid points and 5 million volume elements (tetra, prisms and pyramids) was generated. The A380 (figure 2) is a very complex configuration from which a grid with 9 million grid points and 25 million volume elements was generated. The memory requirements for the A380 on 1 CPU exceeds the resources of most of the tested systems, such that we had to start the benchmarks for this testcase on 8 CPUs.

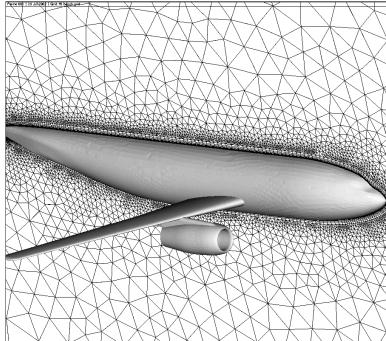


Figure 1: DLR F6 configuration

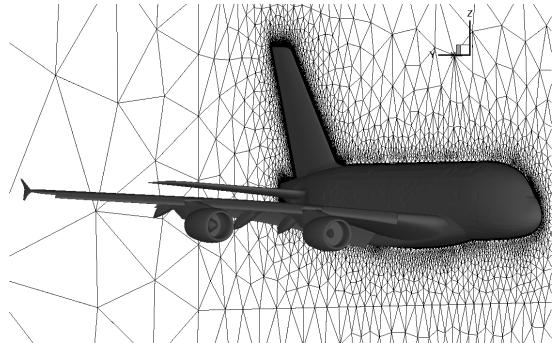


Figure 2: A380 configuration

2.4 Partitioning

The parallelization of the flow solver is based on a domain decomposition of the computational grid. Due to the explicit time stepping scheme, each domain can be treated as a complete grid. First of all we had to split the grids into the number of desired domains for every parallel run of the flow solver. The partitioning is done by a sequential run of the primary grid partitioner, which uses a recursive bisection algorithm to compute the desired number of domains. Figure 3 gives an overview of the maximal dimensions (elements and points) of the partitioned grids for each configuration.

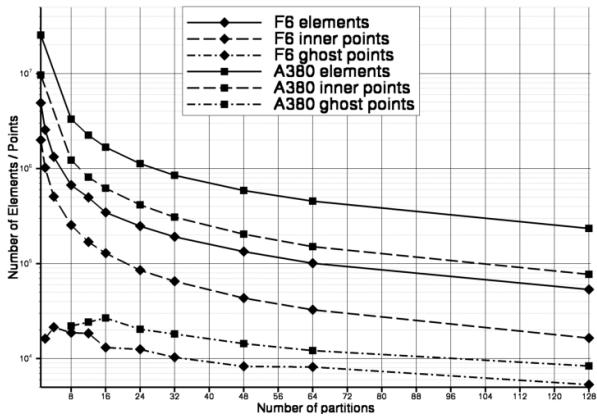


Figure 3: Maximal grid dimensions

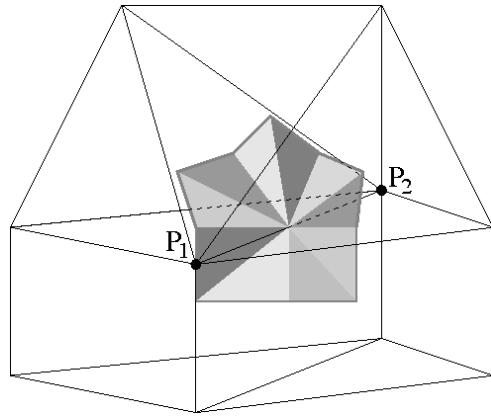


Figure 4: Dual mesh face

2.5 Cache optimization

After the partitioning the parallel preprocessor is started to compute the dual grids out of the partitioned grids. Furthermore the preprocessor is responsible for the cache or vector optimization. A major part of the calculations in the flow solver is performed by considering the grid faces one by one, while the computed values (e.g. fluxes over the face/edge) are adjoined to the respective points on both sides of the face (figure 4). Therefore, the cache optimization for scalar machines is achieved by a sorting of faces/edges in the dual grid. The main difference between the two implemented

cache optimization techniques, with respect to performance issues, is the strategy to sort the faces/edges. In the basic version (released 2002) the faces/edges are sorted into different colors depending on a threshold which the user can define as an input parameter. The aim is to prevent cache misses while looping over the edges/faces within the same color. In the enhanced version (released 2004) the strategy is to minimize the disadvantage of the indirect addressing. In this version the edges and the adjoined points are sorted into colors to achieve direct strided data access in each color.

2.6 Computing platforms

For our benchmarks we have selected a couple of up-to-date cluster systems, but we were not able to run the benchmarks on all architectures currently present in the TOP 500 [1]. However, we were able to test a broad variety of cluster interconnects along with a major set of currently used processors in the High Performance Computing field. The relevant technical details of the tested systems are listed in table 1:

System	Chip	GHz	Ram	Network	Latency	Bandwidth	CPU/Nodes
p690	Power4+	1.70	333 MHz	HPS[4]	8 μs	0.72 GB/s	160/5
Cluster	Xeon	3.06	133 MHz	Myrinet[9]	22 μs	0.03 GB/s	64/32
Cluster	Opteron	1.40	155 MHz	Gigabit	70 μs	0.02 GB/s	32/16
Cluster	Opteron	2.40	166 MHz	Infiniband[12]	10 μs	0.15 GB/s	256/128
XD1[11]	Opteron	2.20	200 MHz	RapidArray	2 μs	0.23 GB/s	72/36

Table 1: Technical details of the tested systems (MPI latency and bandwidth are taken from [3])

2.7 Performed benchmarks

Normally a CFD-engineer/researcher is interested in the turn around time for a complete computation of an (un)steady state problem. Unfortunately the number of iterations the flow solver needs until the solution of the flow field is converged depends highly on the type of flow problem and on the aerodynamic configuration. Thus we will focus on the wall clock or real time (t_w) that the solver needs to perform 1 iteration of the selected multigrid cycle. For the analysis of the parallel performance on the tested systems we calculate the relative speedup

$$s_n = \frac{t_{w_k}}{t_{w_n}}, \quad (1)$$

with $k \in \{1, 8\}$ and $n \in \{1, 2, 4, 8, 12, 16, 24, 32, 48, 64, 96, 128\}$ for each configuration and cache optimization strategy. Thus we have to perform 4 parallel benchmark runs with a different number of domains (n) on each of the systems in table 1 in order to get a significant data basis for our discussion. If we have the relative speedup s_n we are able to conclude about the parallel efficiency of the code

$$e_n = \frac{q_n}{s_n}, \quad \text{with } q_n = \frac{n}{k}. \quad (2)$$

3 Results of the benchmarks

3.1 Analysis of the sequential benchmarks with the DLR F6

In [6] we have presented a quite good parallel efficiency (95 % on 12 CPUs) for the code on a NEC SX-5 vector machine with the DLR F6 configuration. In order to get an idea of how fast the tested cluster systems are compared to the well known vector computers, we have done a reference calculation on the old NEC SX-5 on 1 CPU. The obtained wall clock time for 1 iteration with the settings from 2.2 on the NEC SX-5 was

$$\bar{t}_{w_1} = 33.03s \approx 926 \text{ MFlops.} \quad (3)$$

We take the time from (3) as a reference value and calculate the relative performance

$$p^i = \frac{\bar{t}_{w_1}}{t_{w_1}^i} \quad (4)$$

for a comparison of all of the tested cluster nodes in the following table 2:

Chip	Computing platform			Basic optimization			Enhanced optimization		
	GHz	GB/s	L2 MB	t_{w_1}	p	MFlops	t_{w_1}	p	MFlops
Power4+	1.70	6.8	0.7	70.62 s	0.47	433	38.36 s	0.86	797
Xeon	3.06	4.2	0.5	65.64 s	0.50	466	39.16 s	0.84	781
Opteron	1.40	4.8	1.0	85.83 s	0.43	356	47.05 s	0.70	650
Opteron	2.40	5.2	1.0	48.26 s	0.67	634	32.03 s	1.03	955
Opteron	2.20	6.4	1.0	49.40 s	0.67	619	32.96 s	1.00	928

Table 2: The relative performance of cluster nodes scaled to NEC SX-5

The enhanced cache optimization speeds up the computation by a minor factor of 1.51 and a maximal factor of 1.83 for the tested nodes. The performance gain of the enhanced cache optimization is bigger on the systems which have a smaller memory bandwidth (Xeon) or a third level cache (Power4+). Furthermore it can be expected that the benchmark with the enhanced cache optimization will stress the installed high speed network of the cluster far more than the one with the basic cache optimization. Another observation is that the 2.4 GHz Opteron which has a smaller memory bandwidth than the one with 2.2 GHz is slightly faster with the enhanced cache optimization (2.3 % versus 2.9 %). The results in table 2 leads to the acceptance, that an unstructured CFD code will benefit more from a higher memory bandwidth than from a higher CPU clock.

3.2 Analysis of the Parallel benchmarks

In order to analyse the results of the parallel benchmarks the mainloop time for 1 iteration and the relativ speedup to 1/8 CPUs over the number of used processors for each of the tested system is plotted in the following diagrams. The acceptance that the enhanced cache optimization has a significant impact on the parallel efficiency of the code is correct for all of the systems with one exception. Indeed the gigabit

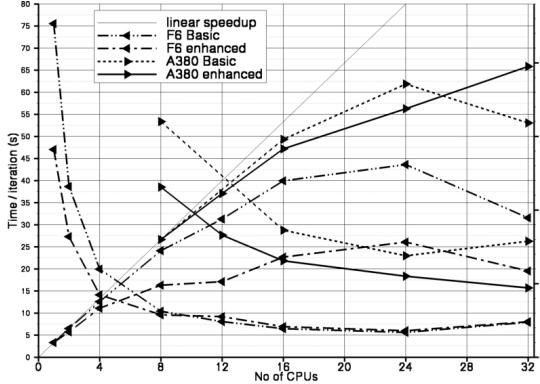


Figure 5: Opteron 1.4 GHz with Gigabit

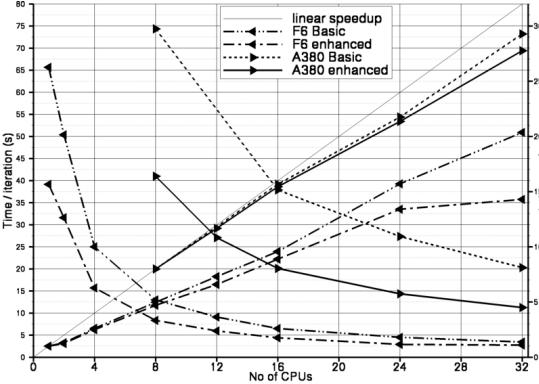


Figure 6: Xeon 3.06 GHz with Myrinet

opteron cluster shows a better parallel performance for the A380 configuration with the enhanced cache optimization turned on. This is not expected, but will have no influence of the overall bad scalability of the code on this system (figure 5). The

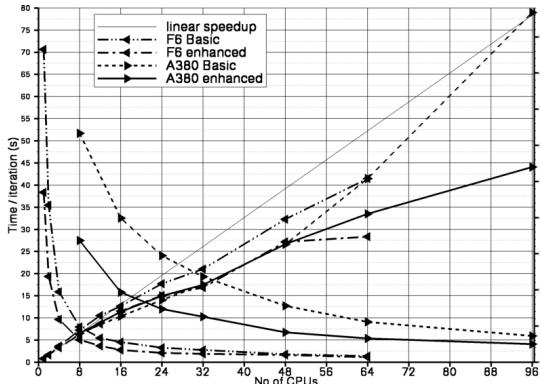


Figure 7: Power4+ 1.7 GHz with HPS

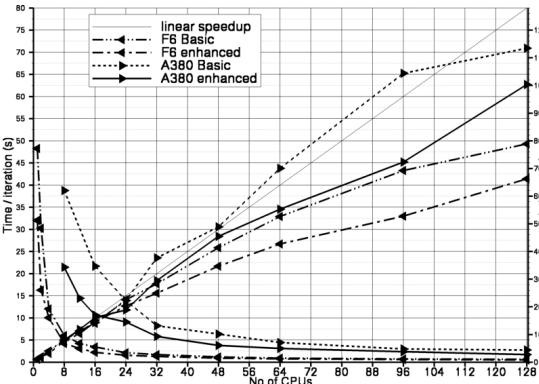


Figure 8: Opteron 2.4 GHz with Infiniband

parallel performance degrades for both configurations at 24 CPUs. Here it is quite obvious that the latency of the gigabit network is the bottleneck, which can be seen by a closer view on the speedup curve for the F6 with the enhanced cache optimization turned on.

The Myrinet system on the other hand performs quite well up to 32 CPUs, but with the F6 configuration the system shows its limitations (figure 6). The Myrinet system is not capable to scale-up as good as the XD1 or the Infiniband system. This could be explained by the larger latency of the Myrinet network and the smaller memory bandwidth of the Xeon bus topology when two CPUs of a node share the memory controller.

The p690 system shows a different behavior (figure 7). This system is the only one with a third level cache and the effect on the parallel performance can be seen with A380 configuration. With the basic cache optimization the speed-up curve increases when 48 CPUs or more are used. For the F6 the cache effect starts at 32 CPUs.

When the enhanced cache optimization is turned on, the cache effect is not so strong (A380 at 32 CPUs) and the speed-up curve is reasonable for both configurations. The limitations of the network can be seen by a closer look at the degradation of the parallel performance at 64 CPUs for the F6. The Infiniband system seems to have enough

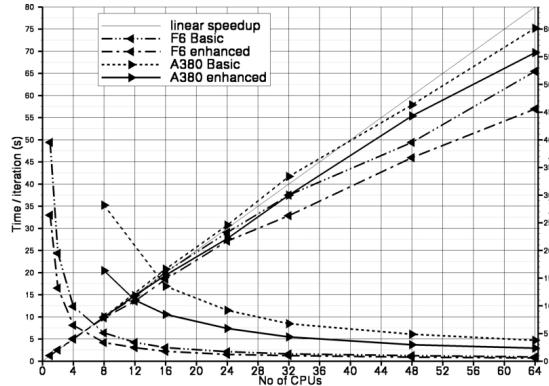


Figure 9: Opteron 2.2 GHz with RapidArray

resources to provide a sustainable performance for both configurations (figure 8). There is no significant degradation in the parallel performance and the speedup curves are reasonable with the exception of the A380 with basic cache optimization. A possible explanation for the super linear speed-up could be the performance on 8 CPUs (38.77 s) compared to the XD1 (35.25 s). Furthermore we can see a cache effect at 128 CPUs for the A380 with enhanced cache optimization, which was also observed for the p690 at 48 CPUs.

The XD1 (figure 9) is the only system which shows a nice speed-up curve for all configurations. No degradation in the parallel performance is observed and it seems that the system is not affected by a bad performance in some combinations of testcase and number of CPUs. We can see a super linear speedup for the A380 with basic cache optimization for up to 32 CPUs, but no other side effect is observed.

4 Conclusion

The parallel efficiency of the TAU-code is evaluated on different distributed memory systems. For the parallel performance and scalability of the code, the high speed network has the strongest influence. The memory bandwidth of the cluster nodes has a major influence as well, but is found to be not so important whenever the enhanced cache optimization is used. The Gigabit cluster gives acceptable performance for up to 16 CPUs with an efficiency of 43%¹. The Myrinet cluster performs quite well up to 32 CPUs (54% efficiency¹) and it can be expected that the network will provide enough resources for configurations like the A380 to calculate with up to 128 CPUs (87% efficiency² on 32 CPUs). The p690 system shows a good performance up to 64 CPUs (54% efficiency¹) and it can be expected to give the same performance like the

¹F6 with enhanced cache optimization

²A380 with enhanced cache optimization

Myrinet system for larger configurations. The Infiniband cluster shows the potential of the attached high speed network and performs very good up to 128 CPUs (52%¹ and 78%² efficiency). On the XD1 the code shows the best scalability. The parallel efficiency is about 71%¹ and 87%² at 64 CPUs and therefore better as on all other systems. Due to the good balance between memory bandwidth, high speed network and processor, the XD1 is the most promising platform for an unstructured code like TAU.

Acknowledgments

We wish to thank Cray Inc and Dr. Monika Wierse for the use of the XD1 system and the provided benchmark results.

References

- [1] TOP 500. <http://www.top500.org/lists/2005/06>, 2005.
- [2] Christian Bell, Dan Bonachea, Yannick Cote, Jason Duell, Paul Hargrove, Parry Husbands, Costin Iancu, Michael Welcome, and Katherine Yellick. An evaluation of current high-performance networks. Paper lbnl-52103, Lawrence Berkeley National Laboratory, January 2003.
- [3] HPC Challenge. http://icl.cs.utk.edu/hpcc/hpcc_results.cgi, 2005.
- [4] IBM. <http://www-03.ibm.com/servers/eserver/pseries>, 2005.
- [5] N. Kroll and J. K. Fassbender, editors. *MEGAFLOW — Numerical Flow Simulation for Aircraft Design Results of the second phase of the German CFD initiative MEGAFLOW presented during its closing symposium at DLR, Braunschweig, Germany, December 10th and 11th 2002*, volume 89 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, Berlin, 2005. Springer Verlag.
- [6] Norbert Kroll, Thomas Gerhold, Stefan Melber, Ralf Heinrich, Thorsten Schwarz, and Britta Schöning. Parallel Large Scale Computations for Aerodynamic Aircraft Design with the German CFD System MEGAFLOW. In *Proceedings of Parallel CFD 2001*, 2001. Egmond aan Zee, The Netherlands, May 21-23.
- [7] D.J. Mavriplis. Parallel Performance Investigation of an Unstructured Mesh Navier-Stokes Solver. *The International Journal of High Performance Computing*, 2(16):395–407, 2002.
- [8] MPICH. <http://www-unix.mcs.anl.gov/mpi/mpich>, 2005.
- [9] Myrinet. <http://www.myri.com/myrinet/overview/index.html>, 2005.
- [10] M. Sillen. Evaluation of Parallel Performance of an unstructured CFD-Code on PC-Clusters. paper 2005-4629, AIAA, Toronto, Ontario, Canada, 2005.

- [11] Cray Inc. Seattle USA. Cray XD1 Technical Specifications, [http : //www.cray.com/products/xd1/specifications.html](http://www.cray.com/products/xd1/specifications.html), 2004.
- [12] Voltaire. [http : //www.voltaire.com/hpc_solutions.htm](http://www.voltaire.com/hpc_solutions.htm), 2005.

(G)
Parallel dynamic grid refinement for industrial applications

Thomas Alrutz, Matthias Orlt

*DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

Proceedings ECCOMAS CFD 2006 Conference,
September 5–8 2006, Egmond aan Zee

Abstract

This paper covers a description of the algorithms and data-structures used in the parallel DLR TAU-Code Adaptation tool. Attention is given to the basic algorithms of the parallel anisotropic refinement and de-refinement, as well as the data-structures for distributed grids and grid hierarchy. The parallelization issues for distributed data are discussed along with the dynamic repartitioning and the load-balancing algorithm.

To demonstrate the dynamic character of the TAU-Code adaptation, we choose a simulation for a complex aircraft configuration and perform a parallel CFD calculation using varying numbers of CPUs. The obtained results show good parallel scalability with respect to memory and CPU-time consumption.

1 Introduction

Local grid refinement for unstructured hybrid meshes is a common approach to improve the accuracy of a solution for a given CFD problem or to reduce the amount of the points needed for a numerical calculation [4, 12]. Typically, this method starts with an initial grid for a particular geometry. Subsequently, the grid is repeatedly locally adapted by inserting or removing points based on a local error estimation or an equidistribution of differences of appropriate values derived from the current solution. The solution of larger CFD problems in an industrial environment requires the use of parallel software on distributed memory machines [9]. A good load balance between the various processors is needed to achieve a high efficiency in a parallel CFD simulation. Each local refinement of the grid requires the grid partitions to be checked. Therefore, a dynamic grid partitioner with an efficient load-balancing algorithm [8] has to be used. Moreover distributed grid data-structures and a distributed grid hierarchy are needed. In this paper, we present the data-structures and algorithms used in the TAU-Code adaptation and grid partitioner and demonstrate their capability in a parallel CFD simulation for a fairly complex example.

2 The DLR TAU-Code

The DLR TAU-Code is a finite-volume Euler/ Navier-Stokes solver working on unstructured hybrid grids. The code is composed of independent modules: Grid partitioner, a preprocessing module, the flow-solver and a grid adaptation module. The grid partitioner and the preprocessing are decoupled from the solver in order to allow grid partitioning and calculation of metrics on a platform other than that used by the solver.

The parallelization of the flow solver is based on a domain decomposition of the computational grid. Due to the explicit time stepping scheme, each domain can be treated as a complete grid. During a flux integration, data has to be exchanged between different domains several times. In order to enable the correct communication, there is one layer of ghost points located at each interface between two neighbouring domains (see figure 1). The solid drawn vertices are *inner* points of the left domain as they are located on the left side of the cut. Each vertex of the right domain, which is connected to one of the *inner* points of the left domain, is a *ghost* point of the left domain. All cut edges

are part of both domains and are used for the communication between those domains. An exchange of updated point data is necessary whenever an operation on the edges is performed. The exchange of the point data is done by employing the Message Passing Interface (MPI).

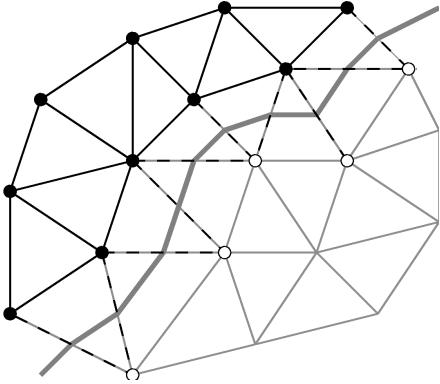


Figure 1: Domain decomposition

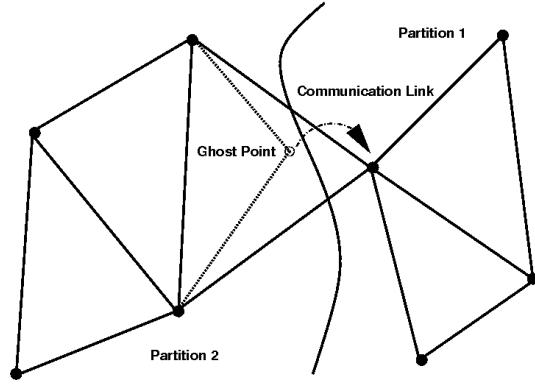


Figure 2: Ghost point creation

Further details of the TAU-Code solver and preprocessing may be found in [13]. In the next sections we give a detailed overview of the algorithms and data-structures used in the grid partitioner and adaptation module.

3 Grid partitioning

First the computational grid is split into the number of desired domains for a parallel run of the TAU-Code. The grid partitioner uses a recursive bisection algorithm to compute the desired number of domains. This step includes the computation of the number of *inner* points for each domain as well as the migration of all attached elements of these points to form a complete grid. The formation of the domain-to-domain overlap region is achieved by adding the *ghost* points to each of the domains (see figure 2). The result is a one cell overlap, which is sufficient for the flow solver communication. The first partitioning of the grid can be done in sequential or parallel mode.

3.1 Distributed grid data-structure

The initial grid data-structure consists of a list of vertices and surface/volume elements along with some required information for the flow solver like boundary markers. The grid data-structure supports a various set of 3D and 2D polyhedra (table 1).

	Elements \mathcal{E}_ℓ						
	Volume				Surface		
Polyhedra	Tetrahedra	Prism	Pyramid	Hexahedra	Triangle	Quadrilateral	
# vertices	4	5	6	8	3	4	

Table 1: Element overview

Let $\Omega \subset \mathbb{R}^3$ be the complete domain and $\Gamma \subset \Omega$ the boundary. The TAU-grid \mathcal{T} that covers Ω is therefore uniquely defined as a collection of the following sets :

1. A set of vertices $\mathcal{V} = \{x_i \in \Omega, 0 \leq i < N_{\mathcal{V}}\}$, a member of which is defined by its coordinates.
2. One or more sets of volume and surface elements $\mathcal{E}_\ell = \{T_i, 0 \leq i < N_{\mathcal{E}_\ell}\}_{\ell=0,1,\dots,5}$, where each type of element T_i is defined by the indices of the corner vertices.
3. A set of boundary markers for the surface elements $\mathcal{M} = \{M_i, 0 \leq i < N_{\mathcal{E}_4} + N_{\mathcal{E}_5}\}$, a member of which is defined by an associated boundary treatment.

However, the distributed grid data-structure requires some additional information. In order to calculate the communication partners for the solver during the preprocessing and allow the grid partitioner to work on the distributed grid domains, it is necessary to have the following sets as well for each partition of the complete grid:

4. A set of ghost points $\bar{\mathcal{V}}^{(k)} = \{\bar{x}_i \in \Omega, N_{\mathcal{V}^{(k)}} \leq i < N_{\bar{\mathcal{V}}^{(k)}} + N_{\mathcal{V}^{(k)}}\}$, a member of which is connected with at least one of the *inner* points of the partition k , with an edge.
5. A set of global point indices $\mathcal{I}^{(k)} \subset \mathcal{I} = \{0, 1, \dots, N_{\mathcal{V}} - 1\}$, where each entry is a reference to the unique global index of a vertex in the initial grid.
6. A set of partition owners $\mathcal{O}^{(k)} = \{d_i, 0 \leq i < N_{\bar{\mathcal{V}}^{(k)}} | 0 \leq d_i < N_{Proc}, d_i \neq k\}$ of the ghost points to relocate the partition where the original vertex is stored.
7. A set of the remote local indices $\mathcal{R}^{(k)} = \{r_i | 0 \leq r_i < \mathcal{V}_{d_i}, 0 \leq i < N_{\bar{\mathcal{V}}^{(k)}}\}$ of the ghost points. This set holds the indices of the original vertex on the remote partition.

A partition of a grid is then defined as a collection of the sets

$$\mathcal{T}^{(k)} := \left\{ \tilde{\mathcal{V}}^{(k)}, \mathcal{I}^{(k)}, \mathcal{E}_\ell^{(k)}, \mathcal{M}^{(k)}, \mathcal{O}^{(k)}, \mathcal{R}^{(k)} \right\}, \quad (1)$$

with $\tilde{\mathcal{V}}^{(k)} = \mathcal{V}^{(k)} \cup \bar{\mathcal{V}}^{(k)}$.

Considering we have N_{proc} partitions then it holds for every two partition p and q

$$\mathcal{V}^{(p)} \bigcap_{p=0, q=0}^{N_{proc}-1} \mathcal{V}^{(q)} = \emptyset, ; p \neq q \implies \mathcal{V} = \bigcup_{k=0}^{N_{proc}-1} \mathcal{V}^{(k)}. \quad (2)$$

The sets $\mathcal{O}^{(k)}$ and $\mathcal{R}^{(k)}$ are used to calculate the communication interfaces for the MPI-transfer.

3.2 Super cell generation

For hybrid grids with quasi structured sub-layers containing prism or hexahedral stacks, we have implemented an algorithm to join each prism or hexahedra stack into a single surface element for the calculation of the domain decomposition. We call those elements

"super cells". The collapsing of a complete element stack into a *super cell* (figure 3), has two side effects. First is the reduced graph for the load balancing calculation. Secondly this feature prevents split stacks due to the fact that the load balancing algorithm has no information about collapsed elements and is therefore only able to split a *heavy weighted* element. This specific feature is used in the *y+* grid adaptation[5].

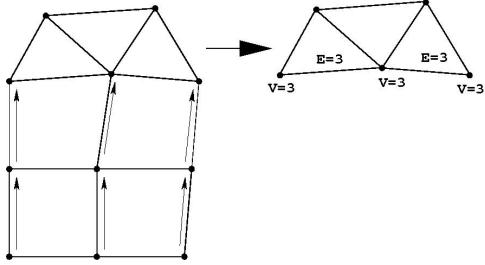


Figure 3: Super cell generation

#CPU	Timings for 30/1 iterations (4w) in seconds			
	Standard		Super cell	
	30	1	30	1
1	811.6s	27.05s	811.6s	27.05s
2	419.3s	13.98s	425.1s	14.17s
4	207.1s	6.90s	209.2s	6.97s
8	107.8s	3.59s	113.0s	3.77s
16	59.0s	1.97s	58.3s	1.94s
24	43.0s	1.43s	42.7s	1.42s
32	33.4s	1.11s	33.6s	1.12s

Figure 4: Parallel performance DLR-F6[3]

3.3 Partitioning and load-balancing

The implemented recursive bisection algorithm is a effective way to achieve domain decomposition on a given grid[7]. The main idea is to split the grid along the medial x,y,z-axis and try to find a decomposition of the domain with equally balanced partitions. Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be the graph calculated from a given grid \mathcal{T} and \mathcal{D} be the set which indicates the current partition of each vertex. First, all vertices are sorted according to their coordinates in each dimension. The algorithm considers a cutplane for each dimension which is perpendicular to the coordinate axis and located at the coordinate value calculated by the median of

$$\sum_{i=0}^{N_V-1} \omega(\sigma_d(x_i)), \quad (3)$$

with ω being the point-weight function, σ_d the permutation of the vertices x_i after the sort in dimension d . By construction, each of the cut-planes will bisect the grid. The algorithm then selects the cut-plane that cuts the smallest number of edges. The subgrids are then considered recursively using the same technique. At the end of the algorithm for each vertex $x_i \in \mathcal{V}$ exactly one partition $d_i, 0 \leq d_i < N_{Proc}$ is stored in \mathcal{D} .

For the use together with the *super cells* some changes are necessary. In order to have a similar load balancing (compared to the case without the *super cells*) it is necessary to correct the edge- and point-weights in the *super cells*. This is done by a summation of all edge- and point-weights in the elements corresponding to the *super cell* (see figure 3). Then a reduction of the set of vertices and edges is performed, such that the recursive bisection algorithm receives only the reduced Graph $\tilde{\mathcal{G}}(\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$.

We have performed several numerical experiments to investigate the influence of this approach to the overall runtime of the flow solver and it was found, that there is no performance penalty greater than the measurement error (see figure 4). Further examination of the calculation of the load-balancing leads to the assumption that, in the case of a hybrid grid, there are always enough unstructured elements which can be used to compensate imbalance introduced by *super cell* generation.

The parallelization of the recursive bisection algorithm is straight forward for distributed grids.

3.4 Parallel migration of unadapted grids

After the decomposition of the vertices on each process is done, we determine for each volume element $T_i \in \mathcal{E}_\ell$ the processor / partition it will be migrated to. Each element will be migrated to those partitions where its vertices are migrated to.

Let $\mathcal{P} = \{0, \dots, N_{Proc} - 1\}$ denote the available processors and $\tilde{\mathcal{D}}^{(k)} = \{d_i, 0 \leq i < N_{\tilde{\mathcal{V}}^{(k)}} | 0 \leq d_i < N_{Proc}\}$ be the set of the new partition owners of the vertices in $\mathcal{T}^{(k)}$.

We define the map $\nu_k : \tilde{\mathcal{V}}^{(k)} \mapsto \mathcal{P}$ given by $\tilde{\mathcal{D}}^{(k)}$. An element $T_i^{(k)} \in \mathcal{E}_\ell^{(k)}$ will then be moved to processor p if and only if one of its vertices will be moved to p :

$$T_i^{(k)} \in \mathcal{E}_\ell^{(p)}, \ell = 0, 1, 2, 3 \iff \exists id \in T_i^{(k)} | \nu_k(id) = p. \quad (4)$$

For consistency reasons it is necessary to migrate the surface elements $\mathcal{E}_4^{(k)}, \mathcal{E}_5^{(k)}$, and the boundary markers $\mathcal{M}^{(k)}$ along with the volume elements. Thus if a volume element and a surface element share a common face

$$T_j^{(k)} \subset T_i^{(k)}, T_j^{(k)} \in \mathcal{E}_i^{(k)}, i = 4, 5, T_i^{(k)} \in \mathcal{E}_\ell^{(k)}, \ell = 0, 1, 2, 3 \quad (5)$$

the surface element has also to be moved to the partition where the volume element goes.

Before the volume and surface elements can be sent to their new partitions, all their *local* indices have to be transferred into the *global index space*, which are the original indices of the associated vertices. This is done by rewriting an index $id \in T_i^{(k)} \in \mathcal{E}_\ell^{(k)}$ with the corresponding value from $\mathcal{I}^{(k)}$. The mapping of the local indices into the *global index space* guarantees the correct reordering of the point indices at the receiving partition.

At the end of the migration process, the vertices are migrated corresponding to their new owners given by $\tilde{\mathcal{D}}^{(k)}$. This is done in several steps, because of the differentiation between *inner* and *ghost* points. The *inner* points can be sent directly to their new owners together with their global indices from $\mathcal{I}^{(k)}$. The *ghost* points on the other hand must be calculated for each partition separately. A *ghost* point is always created in the case when not all vertices of an element are moved to the same partition. Consider an element $T_i^{(k)}$ which is sent to the partitions p and q . For each vertex $x_{id}, id \in T_i^{(k)}$, which is not sent to q a ghost point has to be created on the partition q . Thus

$$\forall id \in T_i^{(k)} \text{ with } \nu_k(id) \neq q \quad (6)$$

we create a *ghost* point entry with vertex x_{id} for the partition q . This is done for all partitions to where the element $T_i^{(k)}$ is migrated. Then the collection of *ghost* points can

be sent to these partitions together with the corresponding global indices and partition owners. On the receiver side it is quite easy to collect all elements and store them in the corresponding sets. For the vertices some additional work is required. First receive all the *inner* points and sort them in increasing order corresponding to their global indices. Then receive all *ghost* points and kill double entries. At the end remap all global point indices in the elements back to local partition indexing. This can be done for each element with the effort of $O(\log(N_{\tilde{V}(p)}))$ by simply sorting the set of global indices $\mathcal{I}^{(k)}$ and employing a binary search on the sorted set for each global index. The migration finish with the reestablishment of the set $\mathcal{R}^{(p)}$ by sending all the global indices of the *ghost* points to their owner partitions for a calculation of the local remote indices. This data is then sent back to the initiating partition / process and the migration of the grid is complete.

4 Grid adaptation

The adaptation module of the TAU-Code consists of three different components for various grid manipulations to adapt a given grid to the solved flow field:

- y_+ based grid adaptation to adjust the first wall distance over turbulent surfaces in hybrid grids,
- hierarchical grid refinement and derefinement to introduce new grid points on a given edge-indicator function without producing hanging nodes,
- surface approximation and reconstruction for curved surfaces after introduction of new grid points.

In this paper our main focus is on the grid refinement and derefinement algorithms along with the parallelization issues. An overview of the y_+ based grid adaptation can be found in [5].

4.1 Grid refinement

The basic concept of the local grid refinement and derefinement is similar to the red/green refinement [6, 10]. After briefly summarising the TAU specific features of the basic algorithm [4], the attention is turned to the concepts and algorithms of derefinement and parallel adaptation.

4.2 Basic algorithm of the local refinement

The main requirements of a local refinement strategy are the detection of grid areas which are to be refined and the method of element subdivisions, which result from insertion of new points to these areas. A very rough draft of the basic algorithm reads as follows:

- 1 Build edge list and element to edge reference.
- 2 Evaluate edge indicators I_e .

- 3** Refine edge list considering the edge indicators, the target point number and the grid conformity.
- 4** Calculate coordinates of new points.
- 5** Construct new elements.
- 6** Interpolate solution to new points.

TAU-adaptation uses an edge based approach, i.e. The refinement indicators are evaluated for all edges, new points are inserted to the edge mid points and the element subdivisions are determined from the configuration of refined edges. Therefore, it builds the edge list and the element to edge reference as the first step.

Using the current solution an indicator value is calculated for each edge. TAU-adaptation offers three types of indicator. The simplest one is the difference based one

$$I_e = \max_i \left(C_i \frac{\Delta\phi_{i,e}}{\max_e \Delta\phi_{i,e}} \right) h_e^\alpha, \quad (7)$$

where $\Delta\phi_{e,i}$ is the difference between the values of the sensor variable ϕ_i on the edge points of the edge e . The density ρ , the velocities v , the total pressure P_t and enthalpy H_t and each user defined variable written out by TAU-solver are available as sensors. The Maximum $\max_e \Delta\phi_{i,e}$ over all edges is used as a normalisation constant if the sensors are of different order of magnitude. This allows the user to weight the different sensors by C_i . Additionally, the differences are scaled with the α power of the edge length h_e . The choice of the parameter α controls the stronger refinement of areas with smaller or larger cell sizes or inhibits the refinement of discontinuities..

The other available indicator types are a gradient based indicator using the differences of the gradients of the sensor variables instead of the values itself. Finally, the reconstruction based indicator considers the differences of the second order representation of the flow variable within both the edge points dual cells on the edge mid point, i.e. at the border face of the dual cells of used vertex centered Finite Volume Method. However, each combination of indicator type and sensor indicates areas with a lack of a accurate representation of the current solution in some sense.

Step **3** of the simplified program flow consists of two nested loops. The outer refinement loop finds the Limit L for the indicator which results in the target number of new points if all edges with $I_e > L$ are initially marked for refinement. In the inner consistency loop goes over all elements inserts additional points to get a valid subdivision case for each element and has to be repeated until no points have to be added because changes can run through the grid.

The set of implemented refinement cases of the element types mainly determines how many edges have to be marked unintended. A special feature of TAU-adaptation is the implemented complete set of tetrahedra subdivisions (see Figure 5). A continued marking of edges not initially marked occurs to preserve the semistructured areas of Navier-Stokes-grids but stops if a tetrahedral area is found. The admissible prism refinement cases (Figure 6) have an identical refinement of base and top triangle to obtain only new prisms. Pyramids are considered to be an unintended exception where

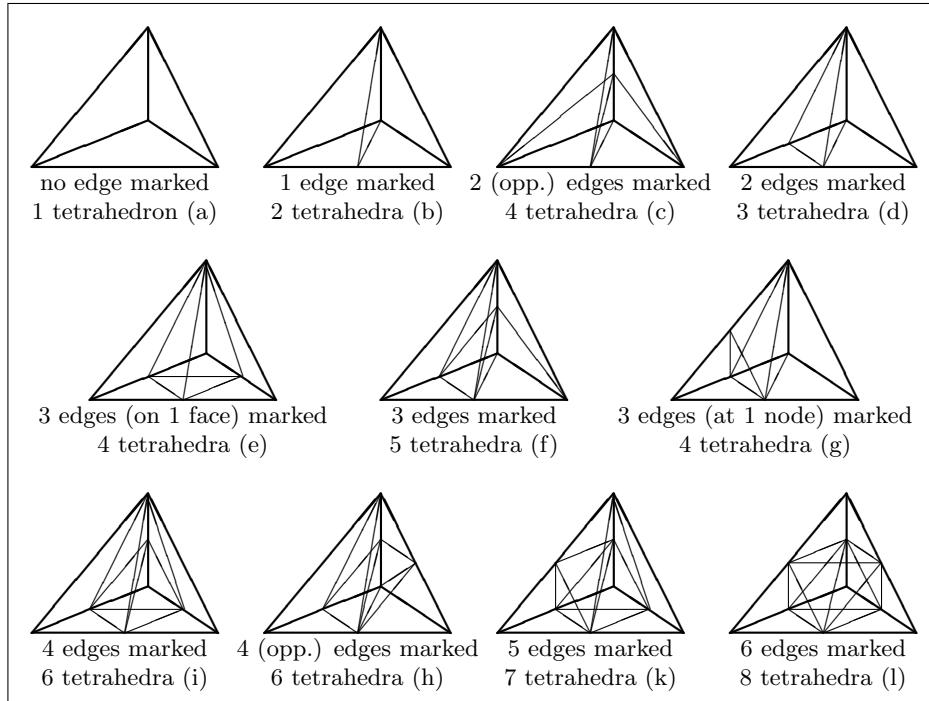


Figure 5: Tetrahedra refinement cases

the height of prism layers varies. So, there is only one refinement case for pyramids (Figure 7) and the continued edge marking in pyramids has to be stopped in the tetrahedral neighbourhood.

After determination of the final edge subdivision all marked edges are identified with new point numbers and the coordinates of these new points are calculated using a spline interpolation for new surface points to consider curved surfaces (step 4 of basic algorithm). The new elements are constructed using the appropriate subdivisions (Figure 5, 6, 7) in step 5. Finally, the solution is interpolated to the new points (step 6).

In the strict sense, this representation of TAU-adaptation is correct only for the first adaptation step. All element subdivision cases except the trivial refinement cases (a) for tetrahedra and prisms and the isotropic refinement for tetrahedra with an additional condition for the choice of the inner diagonal (l) and semi-isotropic refinement for prisms (c), potentially lead to more stretched elements and to a lower grid quality. Therefore TAU-adaptation uses a red/green like technique even in the refine-only-mode to avoid degeneracy of elements. Because its algorithm uses the grid refinement hierarchy, the description is postponed until the next subsection about derefinement of TAU-adaptation.

4.3 Refinement hierarchy for derefinement and no-degeneracy

The application of local grid refinement strategy to an unsteady simulation requires the possibility to remove previously inserted grid refinements if the flow phenomenon

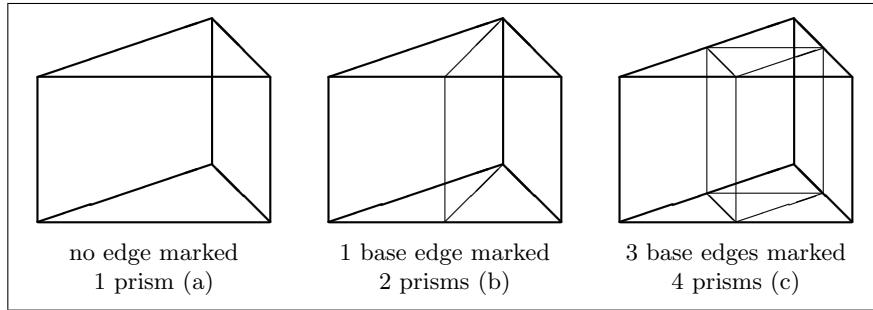


Figure 6: Prism refinement cases

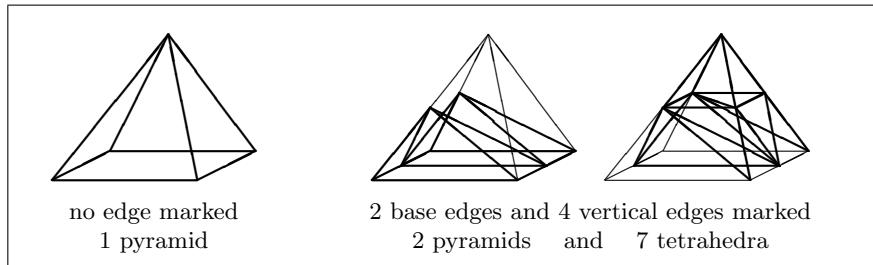


Figure 7: Pyramid refinement cases

causing this refinement has departed from the grid region. Also in a steady calculation it can be helpful to use derefinement if flow phenomena are resolved only after several adaptation steps and change the character of the flow to some extent. In this case the combined re- and derefinement is able to remove points inserted by early adaptation steps based on not fully converged solutions and not needed for the converged solution. Because TAU-adaptation does not perform a proper grid coarsening it has to store the refinement hierarchy in order to restore the original elements later. The refinement hierarchy contains lists of parent elements of each element type and stores the actual hierarchical reference for these parent element lists. This is done in a bottom to top manner, i.e. each element and each parent have references to their direct parent (parent reference), i.e. the element (not existing in the current grid) whose subdivision generated it. Figure 8 illustrates the situation. Some elements may and some parents must have the information that they are initial elements instead of the parent reference. Additionally all elements and parents having a parent have the information which element of the element subdivision they are (child type). Also the information about the applied subdivision case (refine type) is stored for all parents.

Now TAU-adaptation works on the current elements and their direct parents switching from one to another in case of refinement or derefinement, respectively. The preliminary step 1 of the basic algorithm is extended to the algorithm sample shown in the following pseudo code.

1.1 Determine direct parents of the actual elements.

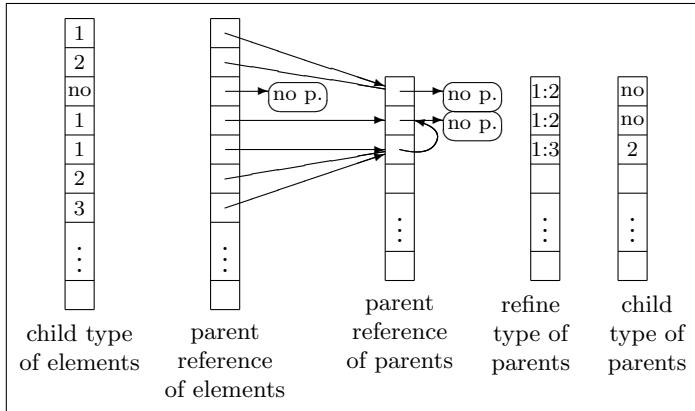


Figure 8: Refinement hierarchy of TAU-adaptation.

1.2 Build edge list of elements and direct parents.

Build element to edge reference of elements and of direct parents.

1.3 Restore mid points of the parent edges.**1.4** Forbid all edges not in isotropic refinements for indication.**1.5** Forbid all edges having a parent point for derefinement.

Step **1.1** is easily done by flagging all parents which are referenced to by other parents and considering the remaining parents. While step **1.2** basically works as the former step **1** did steps **1.3 – 1.5** use the refine type of the direct parents and the child typ of the actual elements and of course the element/parent to edge references.

The refinement step also has to be modified for derefinement:

3.1 Set an initial guess for the indicator limit L .**3.2** Mark all parent edges with $I_e < L s$ for derefinement.**3.3** Mark all edges with $I_e > L/s$ for refinement.**3.4** Mark additional edges for refinement

if needed to get a valid refinement case (consistency loop).

3.5 Count marked edges (i.e. number of new points).

If percentage of new points is to large or too small
correct L , reset re- and derefinements and goto **3.2**.

The parameter s with $\frac{1}{2} < s \leq 1$ of steps **3.2** and **3.3** is intended to produce some inertia for the point movement in the grid and so to stabilise a multiple adaptation and improve the convergence especially of a steady calculation. In an unsteady simulation one might want to set $s = 1$ especially if the local grid refinement has difficulties in following the flow phenomena which are to be resolved maybe if the time steps are fairly large or one do not adapt in each time step.

To avoid element degeneracy elements resulting from non-isotropic subdivisions are not further refined. If such an element has to be refined because some of its edges are marked for refinement, the direct parent element is isotropically refined and the remaining edge marks considered in the resulting children. This leads to the further refinement of some of the new elements which become parents themselves and are written to the parent lists and considered in the hierarchy data structure. This situation is illustrated in the top line of Figure 9. In the same way, if the edge refinement of a

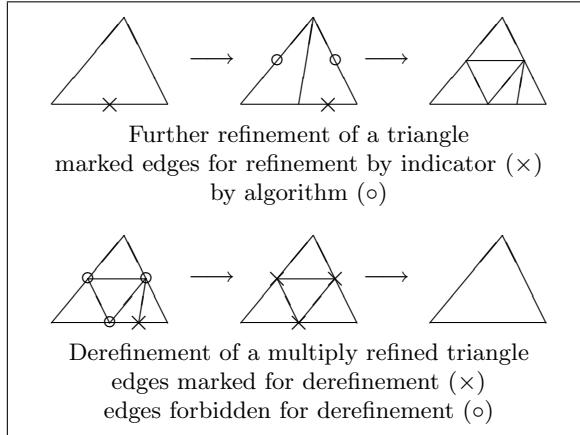


Figure 9: Further refinement and derefinement.

direct parent has changed algorithm works on this parent. This parent can change its refinement type or be completely derefined. In the latter case it goes back to the elements list and is removed from the parent element list. On the other hand if the algorithm works on an element and the element has to be subdivided, the children are written to the new grid, the former element is written to the parent lists and the corresponding hierarchy information is added.

Because TAU-adaptation considers only direct parents only the latest refinement level can be removed (see bottom line of Figure 9). A multiple refinement and the following complete derefinement would generate different intermediate grids as a consequence.

4.4 Parallel grid refinement

The TAU-Code uses the vertex orientated type of domain decomposition holding the additional points of the overlapping elements with lists storing their owner domain and local point number in their own domain (communication map). The main steps of TAU-adaptation work in an element orientated way and each of the parent elements of the grid hierarchy is stored only in one of the domains. So the adaptation needs some simple rules to clearly determine the ownership of elements and edges, and some communication to reinforce the TAU-Code format of grid domains in the end.

Because the parallel adaptation must provide the same result as a serial adaptation, the indicator limit has to be the same on all domains. Therefore one cannot expect to have load balanced domains after a parallel local grid adaptation without grid repartitioning.

In examples with a strong local refinement, e.g. at discontinuities, it is difficult or impossible to get a load balance without distributing the children or grand children of an initial element over various domains in general. So the grid hierarchy needs an extension allowing to refer to other domains.

In order to use the same algorithms as in the serial version and to minimise the number of communications between domains, all elements having a parent are sent to the domain of its direct parent. In general, this step enlarges the number of additional points, i.e. points of other domains needed in this domain to define the overlap elements and the children of own (overlapping) direct parents. A simple rule of element ownership is the element belongs to and is considered by the domain of its direct parent. All initial elements having only own points of the domain are known only in this domain and are refined or derefined there. Initial elements of the overlap area are identified by its additional points, i.e. they have domain own points and points belonging to other domains. These elements determine their owner domain with a simple rule, e.g. the owner of the smallest global point number.

These extensions lead to the following modification of the algorithm:

- 1.1** Send all elements to the domain of their direct parents.
- 1.2** Determine direct parents of the actual elements.
- 1.3** Build edge list of elements and direct parents.
Build element to edge reference of elements and of direct parents.
- 1.4** Build communication tables for edges.
- 1.5** Restore mid points of the parent edges.
- 1.6** Forbid all edges not in isotropic refinements for indication.
- 1.7** Forbid all edges having a parent point for derefinement.

A lot of edge information has to be communicated between domains because the information about restored points on the parent edges, about the admissibility of an initial mark for refinement and about the refinement state is needed in all domains having the edge. So in favour of efficiency it is worth to build edge tables holding the edge numbers corresponding to a position in lists to send to and receive from other domains. Therefore step **1.4** prepares steps **1.5–1.7** and especially the communication of the refinement state within the refinement loop (compare to steps **3.3** and **3.4** of pseudo code). Different from the communication of point related information in the TAU-solver, the adaptation has to communicate the edge information from each domain to each other domain containing the edge. In order to minimise the total amount of transferred information, an owner domain of the edge is defined by the following: the domain owning the edge point with the smaller global point number is the owner of the edge. All edges send their state information to the owner domain of the edge. After determining the resulting state, the edge information is sent back to all other domains having this edge.

After the re- or derefinement is done and the new elements are constructed some additional steps are needed to re-establish the point oriented domain decomposition.

- 5.1** Construct new elements.
- 5.2** Determine the owner domain of the new points.
- 5.3** Update point communication map.
- 5.4** Send elements to the owner partition.
- 5.5** Remove pure addpoint elements.
- 5.6** Permute points.

Data structures of TAU-Code and algorithms of TAU-adaptation allow to arbitrarily assign the new points to the domains, e.g. to the owner domain of the edge the new point originates from. The explicit determination of the new point owner in step **5.2** takes into consideration additional requirements of Y+-adaptation and partitioning. In the next step **5.3** the communication map is updated, i.e. the lists holding the owner domain and the local point numbers of the additional points, for the new points using the edge communication tables again.

At this point of algorithm all the new elements and the parent elemenets reside only in one of the domains. Thus after sending each element to every domain possessing at least one point of the element (step **5.4**) each domain has got all its own and overlapping elements. As in step **1.1** this may enlarge the number of additional points. Because of the unique occurrence of all elements before **5.4** there is no need to check or sort the received elements after **5.4**. In the parent element lists still no parent occurs twice and there is no need to change this.

Because of step **1.1** and because of a possible subdivision of a former overlapping element a domain can have elements consisting of addpoints only. These elements are not needed and removed in step **5.5**. The derefinement and the last step can lead to disconnected points. Therefore step **5.6** re-establishes the order of own and additional points and removes the disconnected ones.

5 Repartitioning

As described in section 4 several information are required for an adapted grid. If we take a closer look at this structure, it can be seen that the parents are stored and handled the same way as the children. However, the element hierarchy information is required to be stored.

5.1 Distributed grid hierarchy

Along with the sets from section 3.1 a hierarchy for a given grid \mathcal{T} is given by the following sets (e.g figure 8):

8. One or more sets of volume and surface parents $\hat{\mathcal{E}}_\ell = \{\hat{T}_i, 0 \leq i < N_{\hat{\mathcal{E}}_\ell}\}_{\ell=0,1,\dots,5}$, where each type of a parent \hat{T}_i is defined by the indices of the corner vertices.

9. One or more sets of hierarchy references for the children $\mathcal{H}_\ell = \{pid_i, 0 \leq i < N_{\mathcal{E}_\ell} \mid -1 \leq pid_i < N_{\hat{\mathcal{E}}_\ell}\}_{\ell=0,1,\dots,5}$. Each entry refers to a parent in $\hat{\mathcal{E}}_\ell$ or is set to -1 in case of no parent.
10. One or more sets of hierarchy references for the parents $\hat{\mathcal{H}}_\ell = \{pid_i, 0 \leq i < N_{\mathcal{E}_\ell} \mid -1 \leq pid_i < N_{\hat{\mathcal{E}}_\ell}\}_{\ell=0,1,\dots,5}$. Each entry refers to a parent in $\hat{\mathcal{E}}_\ell$ or is set to -1 in case of no parent.
11. One or more sets of child type classification for the children $\mathcal{C}_\ell = \{ct_i, 0 \leq i < N_{\mathcal{E}_\ell}\}_{\ell=0,1,\dots,5}$.
12. One or more sets of child type classification for the parents $\hat{\mathcal{C}}_\ell = \{ct_i, 0 \leq i < N_{\hat{\mathcal{E}}_\ell}\}_{\ell=0,1,\dots,5}$.
13. One or more sets of parent type classification for the children $\mathcal{S}_\ell = \{pt_i, 0 \leq i < N_{\mathcal{E}_\ell}\}_{\ell=0,1,\dots,5}$.
14. One or more sets of parent type classification for the parents $\hat{\mathcal{S}}_\ell = \{pt_i, 0 \leq i < N_{\hat{\mathcal{E}}_\ell}\}_{\ell=0,1,\dots,5}$.
15. One or more sets of refine type classification for the parents $\hat{\mathcal{Z}}_\ell = \{rt_i, 0 \leq i < N_{\hat{\mathcal{E}}_\ell}\}_{\ell=0,1,\dots,5}$. Each entry indicates the type of refinement of the associated parent.

In order to handle a distributed grid hierarchy additional information is needed:

16. A set of parent points $\hat{\mathcal{V}}^{(k)} = \{\hat{x}_i \in \Omega, N_{\mathcal{V}^{(k)}} + N_{\bar{\mathcal{V}}^{(k)}} \leq i < N_{\mathcal{V}^{(k)}} + N_{\bar{\mathcal{V}}^{(k)}} + N_{\hat{\mathcal{V}}^{(k)}}\}$.
17. A set of partition owners $\hat{\mathcal{O}}^{(k)} = \{d_i, 0 \leq i < N_{\hat{\mathcal{V}}^{(k)}} \mid 0 \leq d_i < N_{Proc}, d_i \neq k\}$ of the parent points to relocate the partition where the original vertex is stored.
18. A set of remote local indices $\hat{\mathcal{R}}^{(k)} = \{r_i \mid 0 \leq r_i \mathcal{V}_{d_i}, 0 \leq i < N_{\hat{\mathcal{V}}^{(k)}}\}$ of the parent points. This set holds the indices of the original vertex on the remote partition.
19. One or more sets of partition owners of the parents $\mathcal{L}_\ell^{(k)} = \{d_i, 0 \leq i < N_{\mathcal{E}_\ell^{(k)}} \mid 0 \leq d_i < N_{Proc}, d_i \neq k\}_{\ell=0,1,\dots,5}$ for the children.
20. One or more sets of partition owners of the parents $\hat{\mathcal{L}}_\ell^{(k)} = \{d_i, 0 \leq i < N_{\hat{\mathcal{E}}_\ell^{(k)}} \mid 0 \leq d_i < N_{Proc}, d_i \neq k\}_{\ell=0,1,\dots,5}$ for the parents.

In the case of an adapted partitioned grid we write the set of vertices as a union of *inner*, *ghost* and *parent* points as in section 3.1 ($\tilde{\mathcal{V}}^{(k)} = \mathcal{V}^{(k)} \cup \bar{\mathcal{V}}^{(k)} \cup \hat{\mathcal{V}}^{(k)}$).

As the parents are only considered in the grid adaptation module, there is no need to store them on more than one partition. This means a parent is stored uniquely at one location and the sets of different partitions are disjoint to each other. Furthermore there is no need to consider the parents for the creation of a partition or for the load-balancing algorithm. This means for repartitioning after grid adaptation the algorithm described in section 3 is used to calculate the partitions.

5.2 Parallel migration of grid hierarchy

First for each of the parents the partition where the parent should be migrated to is determined. Let $\mathcal{P} = \{0, \dots, N_{Proc} - 1\}$ be as in section 3.4 and $\tilde{\mathcal{D}}^{(k)} = \{d_i, 0 \leq i < N_{\tilde{\mathcal{V}}^{(k)}} | 0 \leq d_i < N_{Proc}\}$ the set like in section 3.4 extended with the information about the parent points. Again we will define the map $\nu_k : \tilde{\mathcal{V}}^{(k)} \mapsto \mathcal{P}$ defined by $\tilde{\mathcal{D}}^{(k)}$. Let $n(\ell)$ denote the number of corner vertices of a parent element (table 1).

We determine the new partition p of a parent $\hat{T}_i^{(k)} \in \hat{\mathcal{E}}_\ell^{(k)}, \ell = 0, 1, 2, 3$ by

$$\hat{T}_i^{(k)} \in \hat{\mathcal{E}}_\ell^{(p)} \iff \#\{t_j \mid \nu_k(t_j) = p\}_{0 \leq j < n(\ell)} \geq \#\{t_j \mid \nu_k(t_j) = q\}_{0 \leq j < n(\ell)}, t_j \in \hat{T}_i^{(k)}, \quad (8)$$

with $0 \leq p < q < N_{Proc}$. The surface parents $(\hat{\mathcal{E}}_4^{(k)}, \hat{\mathcal{E}}_5^{(k)})$ are migrated in the same way as their children described in section 3.4. All the other information like the hierarchy reference is migrated along with the elements and parents. The main problem is the correct reordering of the parent references $\mathcal{H}_\ell^{(k)}$ and $\hat{\mathcal{H}}_\ell^{(k)}$. To do this properly in parallel the same approach as with the indices of the vertices is employed. Thus before sending the parents and the hierarchy reference information to the new partition owners, all the reference information in $\mathcal{H}_\ell^{(k)}$ and $\hat{\mathcal{H}}_\ell^{(k)}$ has to be mapped into the *global hierarchy index space* $\hat{\mathcal{I}}_\ell = \mathcal{H}_\ell \cup \hat{\mathcal{H}}_\ell$.

The global hierarchy index of a parent is obtained by summing up the number of parents on all partitions for each type. After the total number of each parent type is calculated, the unique global hierarchy index \widehat{pid} of a parent on partition k can be calculated with

$$\widehat{pid}_i := i + \sum_{p=0}^{k-1} N_{\hat{\mathcal{E}}_\ell^{(p)}}, \forall \hat{T}_i^{(k)} \in \hat{\mathcal{E}}_\ell^{(k)}, \ell = 0, 1, \dots, 5. \quad (9)$$

The calculated values for all parents are stored in the set $\hat{\mathcal{I}}_\ell^{(k)}$ in order to convert the local hierarchy references into global hierarchy references. We define the map $\gamma_\ell^{(k)} : \mathcal{H}_\ell^{(k)} \mapsto \mathcal{H}_\ell$ and $\gamma_\ell^{(k)} : \hat{\mathcal{H}}_\ell^{(k)} \mapsto \hat{\mathcal{H}}_\ell$ given by $\hat{\mathcal{I}}_\ell^{(k)}$. All elements and parents which satisfy

$$pid_i \in \mathcal{H}_\ell^{(k)} \mid d_i = k, d_i \in \mathcal{L}_{pt_i}^{(k)}, pt_i \in \mathcal{S}_\ell^{(k)}, \quad (10)$$

$$pid_i \in \hat{\mathcal{H}}_\ell^{(k)} \mid d_i = k, d_i \in \hat{\mathcal{L}}_{pt_i}^{(k)}, pt_i \in \hat{\mathcal{S}}_\ell^{(k)} \quad (11)$$

can be handled direct with $\gamma_\ell^{(k)}$. In the case of a remote parent $d_i \neq k$, the information has to be obtained by sending a query to the partition where the remote parent is stored using the owner information from $\mathcal{L}_\ell^{(k)}$ or $\hat{\mathcal{L}}_\ell^{(k)}$. The global hierarchy index is then obtained as follows

$$pid_i \in \mathcal{H}_\ell^{(k)} \mid d_i = p, d_i \in \mathcal{L}_{pt_i}^{(k)}, pt_i \in \mathcal{S}_\ell^{(k)} \implies \widehat{pid}_i = \gamma_\ell^{(p)}(pid_i), \quad (12)$$

$$pid_i \in \hat{\mathcal{H}}_\ell^{(k)} \mid d_i = p, d_i \in \hat{\mathcal{L}}_{pt_i}^{(k)}, pt_i \in \hat{\mathcal{S}}_\ell^{(k)} \implies \widehat{pid}_i = \gamma_\ell^{(p)}(pid_i) \quad (13)$$

and sent back together with the information about the new owner partition of this parent.

After all the information is gathered together the hierarchy references $\mathcal{H}_\ell^{(k)}, \hat{\mathcal{H}}_\ell^{(k)}$ can be mapped into the global hierarchy index space on each partition without further communication effort. All the information about the hierarchy is then packed together with the elements and parents and is sent to the new owner partitions. On the receivers side p the data is collected and the parents are sorted due to their unique global hierarchy index in $\hat{\mathcal{I}}_\ell^{(p)}$ in increasing order. The remapping to the local hierarchy index is done as described in section 3.4 for the indices of the elements. In order to get the local hierarchy index of all parents which are not on the local partition, the same algorithm as in section 3.4 for the *ghost* points is used, to determine the remote indices. The conversion of the indices of the vertices in the sets of parents from local to global and back again is done by using the algorithm described in section 3.4 for the elements. The migration finishes with the reestablishment of the set $\hat{\mathcal{R}}^{(p)}$, which is done by employing the algorithm used for the calculation of $\mathcal{R}^{(p)}$ in section 3.4.

6 Application examples

In this section we apply the parallel adaptation and the parallel grid partitioner for two aerodynamic configurations. The first is a generic delta wing presented in [1] and the second is a A380 configuration presented in [11].

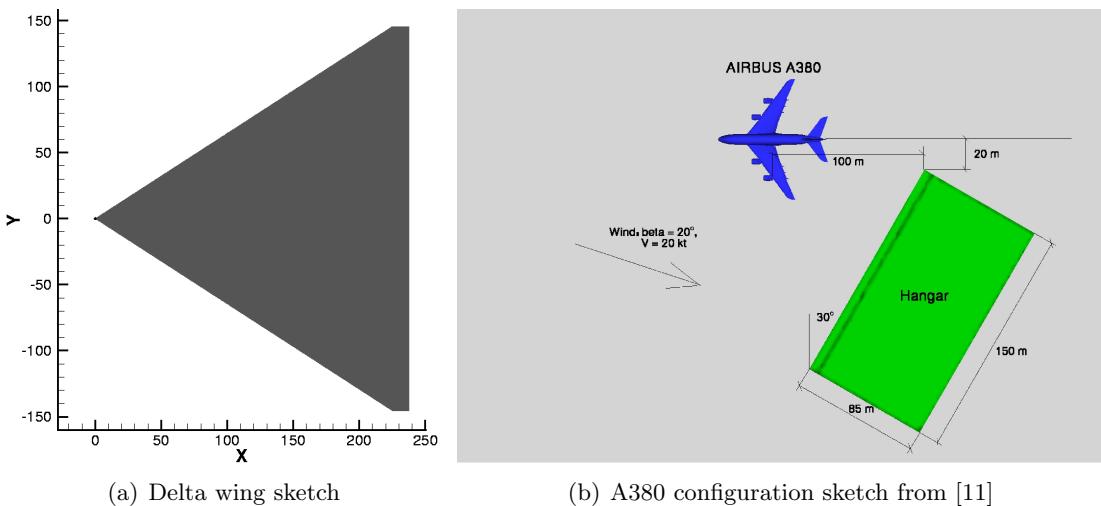


Figure 10: Sketch of delta wing (a) and A380 configuration (b)

6.1 Delta wing

The first example consists of adapting the mesh for a generic delta wing used to investigate the vortex breakdown in a pitching manouver. The initial mesh has approximately 1×10^6 grid points and around 2.5×10^6 volume elements (figure 10 (a)).

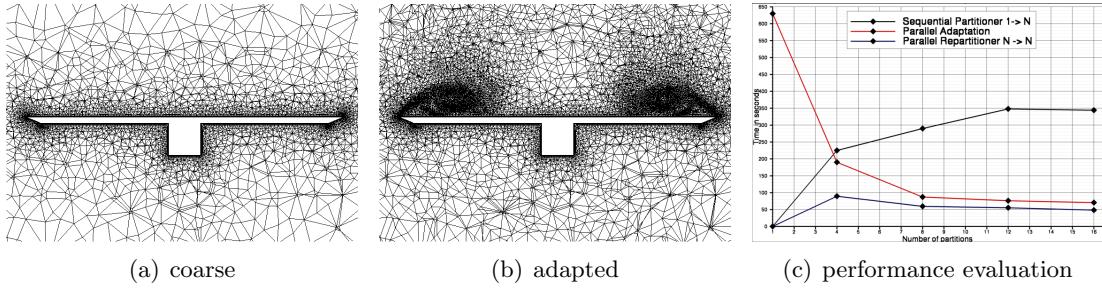


Figure 11: Cut of the grids at $x = 200$ (a,b), performance on a Linux Opteron cluster with Gigabit (c)

It consists of an inner region of 1.9×10^6 prisms in 20 layers for the boundary layer and an outer region of 650000 tetrahedra. After several steps of adaptation we achieved a total number of 4.47×10^6 grid points, around 15×10^6 volume elements and 4.17×10^6 parents. This grid was then used for a performance evaluation of the parallel adaptation method. It can be seen in figure 11 (c) that the parallel adaptation and the partitioner scale quite well. Even the overall time from the initial sequential grid partitioning, the parallel adaptation and the parallel repartitioning is smaller than the time needed for the sequential adaptation. In this case a speedup of 4.4 for 8 CPUs and 5.4 for 16 CPUs could be achieved.

6.2 A380 configuration

The second example consists of adapting the mesh for a AIRBUS A380 in take off configuration used to simulate the jet-flow behind the aircraft on ground ($\text{mach} = 0$) including the interaction of the jet-flow with a hangar (see sketch in figure 10 (b)). The initial mesh has approximately 6.65×10^6 grid points and around 25×10^6 volume elements (figure 12 (a)). The mesh is 6 times adapted using the indicator in equation

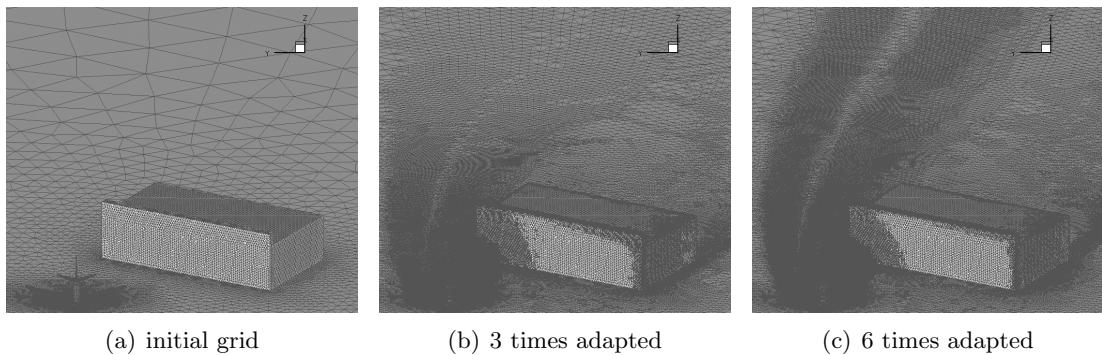


Figure 12: Surface mesh initial (a), 3 times adapted (b) and 6 times adapted (c)

(7) with $\alpha = 1.00$ and $\phi_i = |v|$ to achieve a total number of 12×10^6 grid points and 55×10^6 volume elements (figure 12 (c)).

In figure 13 it can be seen, that the repartitioning algorithms performs quite well, as the

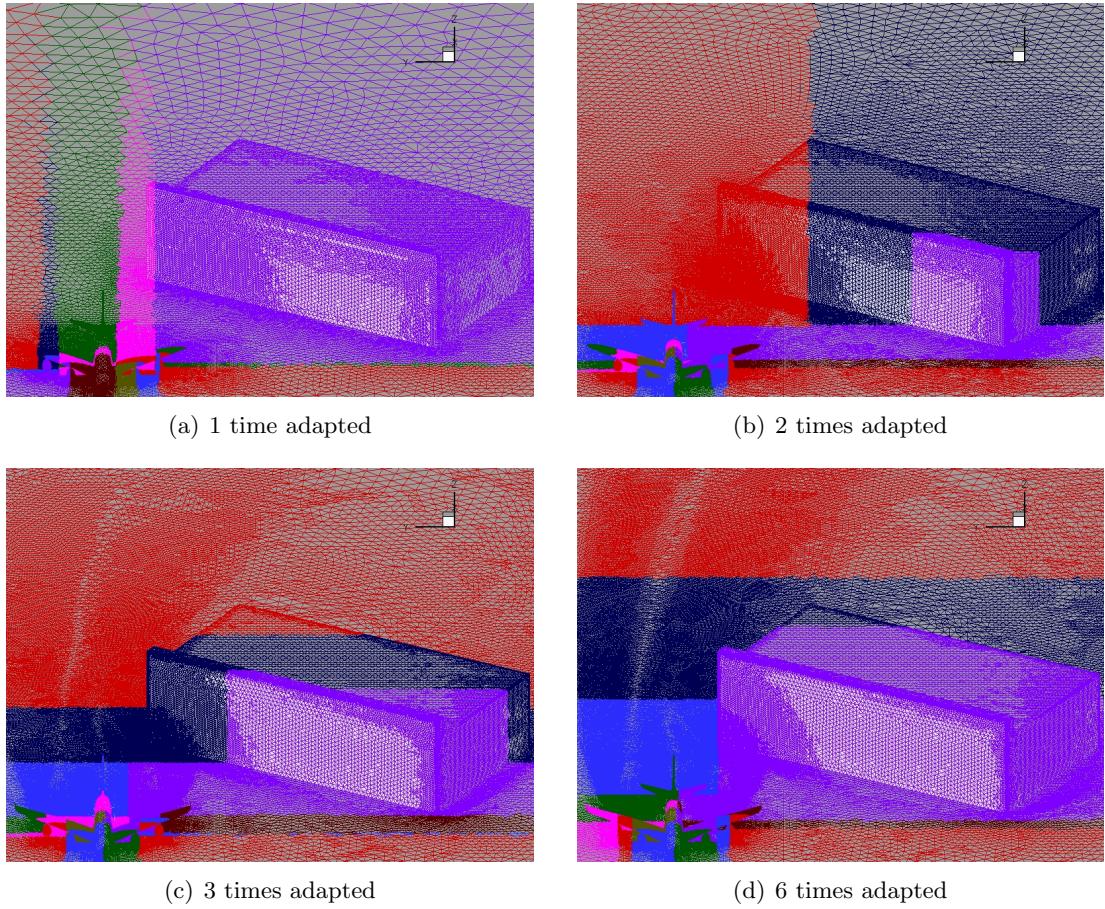


Figure 13: Evolution of the adapted grids splitted in 16 partitions

points move into the area of the jet-blast with each subsequent adaptation step. Here the repartitioning ensures equally loaded partitions after refinement and de-refinement. Furthermore it can be seen that the adaptation removes points from the area of the hangar and distributes them into the area around the jet-blast (figure 13 (b,c)).

	Adaptation steps - timinings include repartitioning					
#CPU	1	2	3	4	5	6
1	-/-	484.37s	-/-	635.85s	-/-	769.37s
16	54.53s	54.97s	55.16s	57.07s	55.08s	83.05s
#Points	$7.5 * 10^6$	$9.3 * 10^6$	$10.2 * 10^6$	$11.9 * 10^6$	$11.5 * 10^6$	$12 * 10^6$
Speedup	-/-	8.81	-/-	11.52	-/-	9.26

Table 2: Comparison of the performance of the parallel and sequential adaptation with the A380 configuration (performed on a Linux Opteron cluster with Infiniband interconnect)

Finally the timing statistics for the A380 configuration are listed in table 2 together

with the dimension of the constructed grids. The overall speedup gain by using the parallel adaptation and partitioner, compared to the sequential adaptation is between 8.8 and 11.5.

7 Conclusions

The presented approach to handle distributed grids with grid hierarchy information in the TAU-adaptation shows a good overall scalability. Together with the dynamic repartitioning after adaptation, the code can be used in a fully parallel simulation system enabling an engineer or researcher to adapt several times without suffering from the bottleneck of a sequential grid adaptation or to change at any time within a calculation the number of partitions on the fly. Compared to a approach presented in [8], where the parallel mesh adaptation has to keep a background mesh on each processor in order to calculate a appropriate refinement sensor, the TAU-adaptation is capable to scale also with respect to memory consumption. This is due to the fact, that the complete grid with its hierachy can be distributed over all the available CPUs for a numerical simulation without storing a global array on any processor.

Despite the established features, there is still need for a further speedup of the refinement and de-refinement algorithm and for parallel initial grid partitioning.

References

- [1] T. Alrutz and M. Rütten. Investigation of Vortex Breakdown over a Pitching Delta Wing applying the DLR TAU-Code with Full, Automatic grid adaptation. Paper 5162, AIAA, 2005.
- [2] Thomas Alrutz. Erzeugung von unstrukturierten Netzen und deren Verfeinerung anhand des Adaptationsmoduls des DLR-TAU-Codes. Diplomarbeit, Universität Göttingen, 2002.
- [3] Thomas Alrutz. Investigation of the parallel performance of the unstructured DLR-TAU-Code on distributed computing systems. In *Proceedings of Parallel CFD 2005 (accepted)*, 2005. Egmond aan Zee, The Netherlands, May 21-23.
- [4] Thomas Alrutz. *MEGAFLOW — Numerical Flow Simulation for Aircraft Design Results of the second phase of the German CFD initiative MEGAFLOW presented during its closing symposium at DLR, Braunschweig, Germany, December 10th and 11th 2002*, volume 89 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, chapter 7 Hybrid Grid Adaptation in TAU, pages 109–116. Springer Verlag, Berlin, 2005.
- [5] Thomas Alrutz and Tobias Knopp. Near wall grid adaption for wall functions. In *Proceedings of International Conference on Boundary and Interior Layers 2006 (submitted)*, 2006. University of Göttingen, July 24-28.
- [6] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.

- [7] Feng Cao, John R. Gilbert, and Shang-Hua Teng. Partitioning meshes with lines and planes. Technical report, Xerox Palo Alto Research Center, January 1996.
- [8] Amik St-Cyr Claude Lepage and Wagdi Habashi. Parallel Ustructured Mesh Adaptation on Distributed-Memory Systems. paper 2004-2532, AIAA, 2004.
- [9] Norbert Kroll, Thomas Gerhold, Stefan Melber, Ralf Heinrich, Thorsten Schwarz, and Britta Schöning. Parallel Large Scale Computations for Aerodynamic Aircraft Design with the German CFD System MEGAFLOW. In *Proceedings of Parallel CFD 2001*, 2001. Egmond aan Zee, The Netherlands, May 21-23.
- [10] D.J. Mavriplis. Adaptive Meshing Techniques for Viscous Flow Calculation on Mixed-Element Unstructured Meshes. paper 97-0857, AIAA, 1997.
- [11] S. Melber-Wilkending. Aerodynamic analysis of jet-blast using cfd considering as example a hangar and an airbus a380 configuration. In *14th AG STAB/DGLR Symposium, Bremen (de), 16.-18.11.2004*, 2004.
- [12] Jens-Dominik Müller. Anisotropic adaptation and multigrid for hybrid grids. *Numerical Methods in Fluids*, 40(3-4):445 – 455, 2002.
- [13] Dieter Schwamborn, Thomas Gerhold, and Roland Kessler. The DLR-TAU Code - an Overview. In *Proceedings ODAS 99*, 1999. ONERA-DLR Aerospace Symposium 21-24 June in Paris, France.

(H)
Numerical simulation of maneuvering combat aircraft

Andreas Schütte¹, Gunnar Einarson¹, Britta Schöning¹, Axel Raichle¹,
Thomas Alrutz², Wulf Mönnich³, Jens Neumann⁴, Jörg Heinecke⁵

¹ *DLR Institute of Aerodynamics and Flow Technology,
38108 Braunschweig, Germany*

² *DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

³ *DLR Institute of Flight Systems,
38108 Braunschweig, Germany*

⁴ *DLR Institute of Aeroelasticity,
37073 Göttingen, Germany*

⁵ *DLR Simulation and Software Technology,
51147 Köln, Germany*

High Performance Computing in Science and Engineering' 05,
Part 3, Pages 185–196, Springer, 2006

Abstract

An overview about recent results of the DLR-Project SikMa-“Simulation of Complex Maneuvers” is presented. The objective of the SikMa-Project is to develop a numerical tool to simulate the unsteady aerodynamics of a free flying aeroelastic combat aircraft, by use of coupled aerodynamic, flight-mechanic and aeroelastic computations. To achieve this objective, the unstructured, time accurate flow-solver TAU is coupled with a computational module solving the flight-mechanic equations of motion and a structural mechanics code determining the structural deformations. By use of an overlapping grid technique (chimera), simulations of a complex configuration with movable control-surfaces are possible.

Nomenclature

t	Time
F	Reference area
l_i	Chord length of the model
Ma	Mach number
$\text{Re} = \frac{V_\infty \cdot l_i}{\nu}$	Reynolds number
$c_p = \frac{p - p_\infty}{q_\infty}$	Pressure coefficient
$q_\infty = \frac{\rho_\infty}{2} V_\infty^2$	Dynamic pressure
$\omega^* = 2\pi f \cdot l_i / V_\infty$	Reduced frequency
$c_L = \frac{L}{q_\infty F}$	Lift coefficient
$c_M = \frac{M}{q_\infty Fl_i}$	Pitching moment coefficient
$c_l = \frac{l}{q_\infty Fl_i}$	Rolling moment coefficient
V_∞	Free stream velocity
Θ	Incident angle, pitch at $\Phi = 0^\circ$
Φ	Roll angle
Φ_0	Initial roll angle
η	Flap deflection angle
α	Angle of attack
$\Delta\alpha$	Angle-of-attack amplitude

1 Introduction

The improvement of maneuverability and agility is a substantial requirement of modern fighter aircraft. Currently, roll-rates of $200^\circ/s$ and more can be achieved, especially if the design of the aircraft is inherently unstable. Most of today’s and probably future manned or unmanned fighter aircraft will be delta wing configurations. Already at medium angles of attack the flow field of such configurations is dominated by vortices developed by flow separation at the wings and the fuselage. The delay in time of vortex position and condition to the on-flow conditions of the maneuvering aircraft can lead to significant phase shifts in the distribution of loads. In such a case, reliable results for the analysis of the flight properties can only be achieved by a combined non-linear integration of the unsteady aerodynamics, the actual flight motion, and the elastic deformation of the aircraft structure.

Today, these types of data can only be obtained by flight tests, and not during the design period. Flight tests, as well as modifications after the design phase, lead normally to an increase in costs. In order to decrease the costs incurred by extensive flight-tests and the post-design phase modifications, it would be helpful to have a tool which enables aircraft designers to analyze and evaluate the dynamic behavior during the design phase.

The main objective of this paper is to focus on the necessity for developing an interactive, multidisciplinary engineering tool for predicting the unsteady critical states of complex maneuvering aircraft. Such a simulation environment has to bring together aerodynamics, aeroelasticity and flight mechanics in a time accurate simulation tool. In order to deliver such a tool in the near future, the DLR Project SikMa-“Simulation of Complex Maneuvers” has been initiated to combine these three disciplines into one simulation environment.

For validating the numerical simulations several wind tunnel experiments in both the low speed and transonic regime will be done within the SikMa project.

2 Numerical Approach

2.1 CFD Solver TAU

The behavior of the fluid-flow affecting the object of interest is simulated with the TAU-Code, a CFD tool developed by the DLR Institute of Aerodynamics and Flow Technology [3][4]. The TAU-Code solves the compressible, three-dimensional, timeaccurate Reynolds-Averaged Navier-Stokes equations using a finite volume formulation. The TAU-Code is based on an hybrid unstructured-grid approach, which makes use of the advantages that prismatic grids offer in the resolution of viscous shear layers near walls, and the flexibility in grid generation offered by unstructured grids. The grids used for simulations in this paper were created with the hybrid grid generator Centaur, developed by CentaurSoft [1]. A dual-grid approach is used in order to make the flow solver independent from the cell types used in the initial grid. The unstructured grid approach is chosen due to its flexibility in creating grids for complex configurations, e.g. a full-configured fighter aircraft with control surfaces and armament, the capability of grid adaptation and straightforward parallelization of all the main TAU modules.

The TAU-Code consists of several different modules, among which are:

- The Preprocessor module, which uses the information from the initial grid to create a dual-grid and the coarser grids for multigrid.
- The Post-processing module, which is used to convert TAU-Code result-files to formats usable by popular visualization tools.
- The Adaptation module, which refines and derefines the grid in order to capture flow phenomena like vortex structures and shear layers near viscous boundaries, among others.
- The Deformation module, which propagates the deformation of surface coordinates to the surrounding grid.

- The Post-processing module, which is used to convert TAU-Code result files to formats usable by popular visualization tools.

In the Solver module, several upwind schemes, as well as a central scheme with artificial dissipation, are available for the spatial discretization. Both Spalart-Allmaras and $k\omega$ turbulence models are implemented. For steady calculations an implicit LU-SSOR multistage Runge-Kutta time stepping scheme is used [2]. For time accurate computations, an implicit dual-time stepping approach is used. The TAU-Code is parallelized using grid partitioning, and a multigrid approach is used in order to increase the performance. The TAU-Code can handle simulations containing multiple bodies in relative motion with one another, e.g motion of control surfaces with respect to the aircraft, by use of a hierarchical motion-node structure. The motion of each body can either be calculated internally by the TAU-Code, or supplied by an external program through a Python implemented external interface.

2.2 TAU-Code Extension: Chimera Technique

The TAU-Code can handle simulations containing multiple bodies in relative motion with one another, e.g motion of control surfaces with respect to the aircraft, by use of a hierarchical motion-node structure. The motion of each body can either be calculated internally by the TAU-Code, or supplied by an external program through a Python implemented external interface.

2.3 Flight Mechanics

For the numerical simulation of the flight mechanics, the simulation environment SIMULA developed at the DLR Institute of Flight Systems is used [6]. SIMULA provides the three basic functionalities necessary for flight simulation and flight control purposes: trimming, i.e. the determination of the initial state and control values, linearization and stability analysis, and simulation, i.e. the numerical integration of the equations of motion.

linearization and stability analysis, and simulation, i.e. the numerical integration of the equations of motion.

2.4 CSM-Code

For the coupling of the aerodynamic and structural dynamic simulations in the time domain, a loose coupling scheme has been implemented. The coupling scheme is conservative with regards to the forces, moments and the work performed on both the aerodynamic and structure dynamic side. Furthermore, it is verified that no dissipation or accumulation of net energy occurs.

The main characteristics of the aeroelastic fluid structure interaction in the time domain are as follows:

- loose coupling of computational fluid dynamics (CFD) and computational structure dynamics (CSD) through file input/output,

- use of an implicit or explicit Newmark algorithm for the time integration of the CSD equations of motion,
- use of an implicit or explicit Newmark algorithm for the time integration of the CSD equations of motion,
- data exchange based on adjusted conventional serial staggered (CSS) algorithm modified with a predictor-corrector scheme,
- structural behavior is described by the complete FE-Model of the delta-wing and the support.

2.5 Integration Framework

The integration framework TENT [8] provides a graphical user interface for controlling and monitoring coupled simulation workflows. The various codes used in the SikMa simulations will be made available in the TENT system, where a simulation workflow can be built by connecting icons representing each code using a graphical workflow editor. Java wrappers containing the basic control functionality for the TAU and SIMULA applications are already integrated in the TENT environment. The wrapper for the CSM-Code as well as the extension of the functionality to handle the coupling between all three disciplines is under development.

The integration framework TENT [8] provides a graphical user interface for controlling and monitoring coupled simulation workflows. The various codes used in the SikMa simulations will be made available in the TENT system, where a simulation workflow can be built by connecting icons representing each code using a graphical workflow editor. Java wrappers containing the basic control functionality for the TAU and SIMULA applications are already integrated in the TENT environment. The wrapper for the CSM-Code as well as the extension of the functionality to handle the coupling between all three disciplines is under development.

3 Experimental Data

For the validation of the numerical simulation software, various wind-tunnel experiments, designed specifically for the SikMa project, are performed. Experimental data, both steady and unsteady, are available for a 65°-swept delta-wing-fuselagemodel-configuration which has been tested in the DNW Transsonic-Wind-Tunnel Göttingen (DNW-TWG). The model has movable trailing-edge flaps an can be used for both guided and free-to-roll maneuver simulations around its longitudinal axis. The model has a chord length of 482mm and a span of 382mm. For the verification of the aerodynamic-structure coupling a steady and dynamic system identification of the delta-wing and the support within the wind tunnel is done. The system parameters are used to setup the FE-model for the coupled simulation.

The main experiments are done in the DNW Low-Speed-Wind-Tunnel Braunschweig (DNW-NWB). In order to perform these experiments, a wind-tunnel model has been designed and built for the SikMa project. The model, shown in Figure 1, is based on the X-31 experimental high angle-of-attack aircraft configuration. The X-31 model is

a 1:7 scaled model with a span of 1000mm and an overall length of 1800mm. The model is equipped with remote controlled moveable control devices driven by internal servo-engines. Measurement equipment is installed to determine the aerodynamic forces and moments on the model, as well as span-wise pressure distributions at locations of 60% and 70% chord length. The experiments include steady-state measurements using PSP-“Pressure Sensitive Paint”, which provide detailed information on the surface pressure distribution for the whole wing. The experiments will culminate with maneuver simulations, where the movement of the aircraft and the control devices will be synchronized. For the maneuver experiments the model will be mounted on the MPM-“Model Positioning Mechanism” of the DNW-NWB.

4 Results

For the verification and validation of the simulation environment the results of the numerical simulations are compared against data collected from various experimental simulations. To show the capability of the TAU-Code to predict the unsteady aerodynamic behavior of configurations with vortex dominated flow fields the deltawing-configuration described in section 3 is used. Figure 2 shows a result of a deltawing in rigid body pitching motion. In this calculation the $k!$ turbulence model is used. For all simulations presented a central spatial discretization scheme is used. Depicted is the hysteresis loop of the lift coefficient over the angle-of-attack α . The wing is pitching around $\alpha = 9^\circ$ with an amplitude of $\Delta\alpha = \pm 6^\circ$. The wing is oscillating with a reduced frequency of $\omega^* = 0.56$. It is seen that the calculation compares well with the experimental data. For higher angles-of-attack the calculation tends to predict higher lift, because with the common turbulence models a different load distribution over the wing is predicted compared to the experiment. At higher angles of attack the vortex structure and the location of vortex breakdown will not be correctly predicted. The same behavior has been found to occur in simulations using the structured DLR FLOWer-Code.

In Figure 3 the result of a coupled simulation between CFD and flight-mechanics is shown using the delta-wing with trailing-edge flaps. To simulate the motion of the control surfaces (trailing-edge flaps) the chimera approach has been used. The maneuver shown in Figure 3 is a 1 DoF rotation around the longitudinal axis of the delta-wing induced by an asymmetric deflection of the flaps by $\eta = \pm 3^\circ$. The initial aircraft attitude is at $\alpha = 9^\circ$ and $\Phi_0 = 0^\circ$. It is seen that the wing enters a periodic roll motion in both the numerical and experimental simulations. One of the sources for the difference seen between the numerical and experimental results is the mechanical friction in the experimental setup, which is not taken into account in the numerical simulation, thus leading to a higher frequency of rotation. Currently only a one equation turbulence model is used in the coupled simulations. It is known that for sharp leading-edge delta-wings the $k\omega$ turbulence model delivers better results and will be used in further simulations. The capability to predict the elastic deformations of the delta-wing configuration during a guided roll maneuver is shown in Figure 4. In the coupled simulation between the TAU-Code and the structural mechanics tool developed within SikMa the deltawing and the rear-sting support are considered to be elastically deformable. For the coupled

simulation the finite element model [7] takes into account both the delta wing configuration as well as the flexible support. The FE-Modell is validated based on results of both ground vibration and static deformation tests.

In Figure 5 the corresponding history of the model deflection due to the elasticity is seen. Depicted is the displacement of the delta-wing nose-tip and sting relative to the rigid-body motion case. It is seen that the wing tip is describing an elliptic motion during the rotation. The green loop in Figure 5 shows the tip movement from the system identification in the experiment due to the integration of the acceleration. Because of the integration the shift in z-direction can not be captured. However, it is shown that the numerics predict the characteristic movement of the wing tip accurately. Due to this deformation the effective angle-of-attack at the same roll angle is higher in the elastic case. This leads to a higher amplitude of the rolling moment, as is seen in Figure 4.

For the X-31 configuration, results from steady-state numerical simulations have been obtained. These simulations show the capability of the TAU-Code to simulate complex delta-wing configurations with rounded leading-edges. Figure 6 shows the numerically simulated 3D flow field over the X-31 configuration, which is a good indication of the complexity of the vortex flow topology over the wing and fuselage. Comparisons with experimental data show good agreement regarding the vortex topology. In Figure 7 an oil flow picture of the X-31 clean-wing from low speed experiments is shown. The angle-of-attack is $\alpha = 18^\circ$ at a Reynolds number of 1.0Mio. The separation line of the strake vortex and the main wing vortex as well as the attachment line of the main wing vortex near the leading-edge is emphasized. In Figure 8 the corresponding CFD calculation is depicted. It is seen that the flow topology from the calculation fits quite well with the experiment. Further experimental results delivering steady pressure distributions upon the wing were done within another X-31 test campaign. The PSP result at $\alpha = 18^\circ$ at a Reynolds number of 2.07Mio is shown in Figure 9. Comparing the pressure distribution in Figure 9 with the CFD calculation in Figure 10 shows that the main footprints of the vortices are captured by the numerics, but the suction-strength and the vortex location are not. However, in this case the experiment is done with mounted leading-edge flaps. The gaps between the leading-edge flaps probably influence the vortex formation considerably. Additional calculations will follow taking all geometric features into account.

5 Conclusions

In this paper the activities and recent results of the DLR-Project SikMa were presented. In SikMa a simulation tool will be developed that is capable of simulating a maneuvering elastic aircraft with all its moveable control devices. The simulation tool combines time-accurate aerodynamic, aeroelastic and flight-mechanic calculations to achieve this objective. Preliminary verification of the functionality of the simulation tool has been shown by simulating a sharp leading-edge delta-wing during a guided motion pitching maneuver and free-roll maneuvers due to flap deflections. Furthermore, first perspectives were presented regarding the time accurate coupling between the TAU-Code and the numerical Structure-Mechanical Tool. Initial results of the steady flow field around

the X-31 configuration were presented.

Acknowledgement

The authors would like to thank Dr. Burkhard Gölling for providing the wind-tunnel data and for his engagement managing the experimental simulation tasks at DLR Göttingen within the project SikMa.

References

- [1] *Centaur Soft*: <http://www.Centaursoft.com>
- [2] Dwight, R.: *Time-Accurate Navier-Stokes Calculations with Approximately Factored Implicit Schemes*. Proceedings of the ICCFD3 Conference Toronto, Springer, (2004).
- [3] Galle, M.; Gerhold, T.; Evans, J.: *Technical Documentation of the DLR TAU-Code*DLR-IB 233-97/A43 1997
- [4] Gerhold, T.; Galle, M.; Friedrich, O.; Evans, J.: *Calculation of Complex Three-Dimensional Configurations employing the DLR TAU-Code*AIAA-97-0167 1997
- [5] Madrane, A.; Raichle, A.; Stürmer, A.: *Parallel implementation of a dynamic overset unstructured grid approach*.ECCOMAS Conference Jyväskylä Finland, 24.-28. July 2004.
- [6] Mönnich, W.; Buchholz, J.: *SIMULA - Ein Programmepaket fuer die Simulation dynamischer Systeme - Dokumentation und Benutzeranleitung - Version 2DFVLR* Institutsbericht IB 111-91/28, 1991
- [7] Neumann, J.: *Strukturmechanische und strukturdynamische Finite Element Modelle des Windkanalmodells "AeroSUM" mit Halterung*.DLR-IB, IB 232-2003-J01, (2003).
- [8] Schreiber, A.: *The Integrated Simulation Environment TENT*.Concurrency and Computation: Practice and Experience, Volume 14, Issue 13-15, S.1553-1568, 2002.
- [9] Schütte, A.; Einarsson, G.; Schöning, B.; Madrane, A.; Mönnich, W., Krüger, W.: *Numerical simulation of manoeuvring aircraft by aerodynamic and flight mechanic coupling*.RTO AVT-Symposium Paris, 22.-25. April 2002.



Figure 1: X-31 Remote control model in the DNW-NWB Braunschweig.

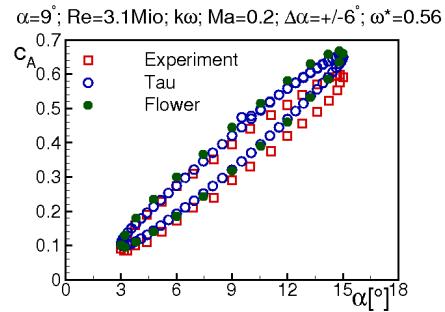


Figure 2: Lifting coefficient vs. angle of attack in pitching motion of a 65° -swept deltawing.

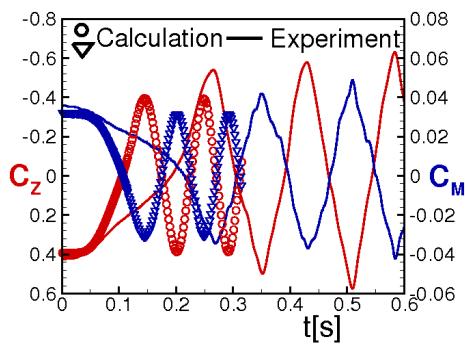


Figure 3: 1 DoF Free-to-Roll maneuver of delta-wing-flap conf. through trailing-edge flap deflection of $\eta = \pm 3^\circ$.

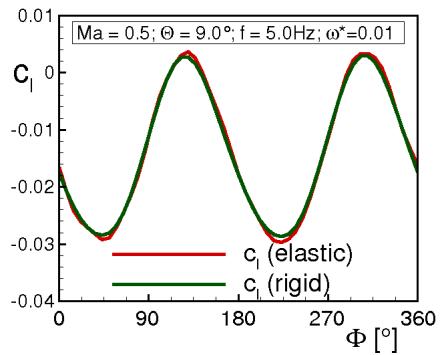


Figure 4: Comparison of rolling moment between rigid- and elastic-body motion of delta-wing during constant rotation. Time accurate coupled CFD(Euler)-CSM simulation.

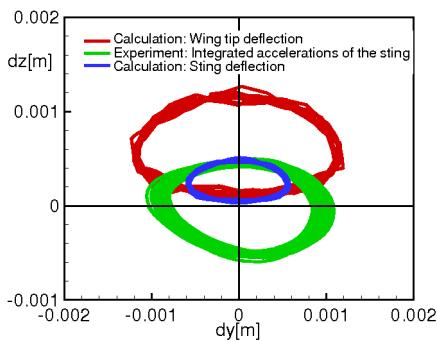


Figure 5: History of the delta-wing nose and sting deflection during elastic-body motion comparison with experiment. Time accurate coupled CFD(Euler)-CSM simulation.

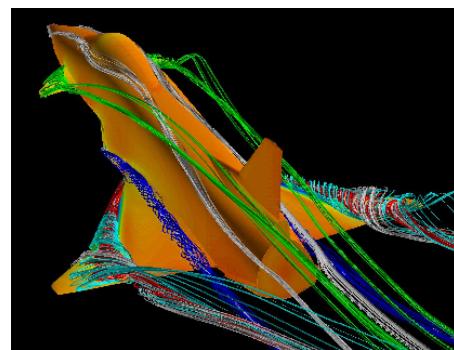


Figure 6: 3D fbw field over the X-31 Configuration at 18° angle-of-attack.

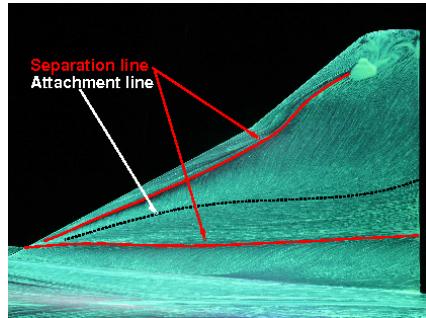


Figure 7: Oil fbw visualization of the X-31 clean wing at $\alpha = 18^\circ$, $Re=1.0\text{Mio}$.

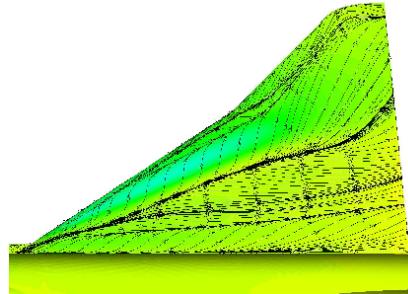


Figure 8: TAU calculation: Visualization of surface streamlines at $\alpha = 18^\circ$, $Re=1.0\text{Mio}$.

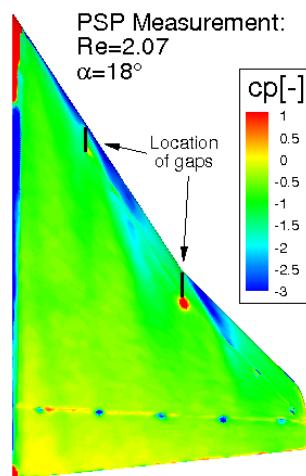


Figure 9: Steady PSP measurement of the pressure distribution over the X-31 wing.

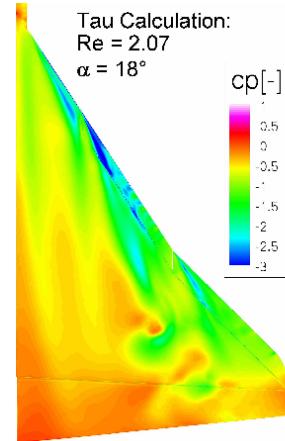


Figure 10: Steady TAU RANS calculation of the pressure distribution over the X-31 wing (clean wing).

(I)

Prediction of the Unsteady Behavior of Maneuvering Aircraft by CFD Aerodynamic, Flight-Mechanic and Aeroelastic Coupling

Andreas Schütte¹, Gunnar Einarson¹, Axel Raichle¹, Britta Schöning¹,
Wulf Mönnich², Jens Neumann³, Jürgen Arnold³, Thomas Alrutz⁴,
Jörg Heinecke⁵, Tomas Forkert⁵, Holger Schumann⁵

¹ *DLR Institute of Aerodynamics and Flow Technology,
38108 Braunschweig, Germany*

² *DLR Institute of Flight Systems,
38108 Braunschweig, Germany*

³ *DLR Institute of Aeroelasticity,
37073 Göttingen, Germany*

⁴ *DLR Institute of Aerodynamics and Flow Technology,
37073 Göttingen, Germany*

⁵ *DLR Simulation and Software Technology,
51147 Köln, Germany*

Proceedings of the RTO AVT-123 Symposium,
April 24–28 2005, Budapest

Abstract

An overview about recent results of the DLR-Project SikMa-“Simulation of Complex Maneuvers” is presented. The objective of the SikMa-Project is to develop a numerical tool to simulate the unsteady aerodynamics of a free flying aeroelastic combat aircraft, by use of coupled aerodynamic, flight-mechanic and aeroelastic computations. To achieve this objective, the unstructured, time-accurate flow-solver TAU is coupled with a computational module solving the flight-mechanic equations of motion and a structural mechanics code determining the structural deformations. By use of an overlapping grid technique (chimera), simulations of a complex configuration with movable control-surfaces are possible.

Nomenclature

t	Time
F	Reference area
l_i	Chord length of the model
Ma	Mach number
$\text{Re} = \frac{V_\infty \cdot l_i}{\nu}$	Reynolds number
$c_p = \frac{p - p_\infty}{q_\infty}$	Pressure coefficient
$q_\infty = \frac{\rho_\infty}{2} V_\infty^2$	Dynamic pressure
$\omega^* = 2\pi f \cdot l_i / V_\infty$	Reduced frequency
$c_L = \frac{L}{q_\infty F}$	Lift coefficient
$c_M = \frac{M}{q_\infty Fl_i}$	Pitching moment coefficient
$c_l = \frac{l}{q_\infty Fl_i}$	Rolling moment coefficient
l_μ	Aerodynamic mean cord
V_∞	Free stream velocity
f	Frequency
dy	Deformation increment in y-direction
dz	Deformation increment in z-direction
Θ	Incident angle, pitch at $\Phi = 0^\circ$
Φ	Roll angle
Φ_0	Initial roll angle
η	Flap deflection angle
α	Angle-of-attack
$\Delta\alpha$	Angle-of-attack amplitude

1 Introduction

The improvement of maneuverability and agility is a substantial requirement of modern fighter aircraft. Currently, roll-rates of $200^\circ/s$ and more can be achieved, especially if the design of the aircraft is inherently unstable. Most of today’s and probably future manned or unmanned fighter aircraft will be delta wing configurations. Already at medium angles of attack the flow field of such configurations is dominated by vortices developed by flow separation at the wings and the fuselage. The delay in time of vortex

position and condition to the on-flow conditions of the maneuvering aircraft can lead to significant phase shifts in the distribution of loads. In such a case, reliable results for the analysis of the flight properties can only be achieved by a combined non-linear integration of the unsteady aerodynamics, the actual flight motion, and the elastic deformation of the aircraft structure.

Today, these types of data can only be obtained by flight tests, and not during the design period. Flighttests, as well as modifications after the design phase, lead normally to an increase in costs. In order to decrease the costs incurred by extensive flight-tests and the post-design phase modifications, it would be helpful to have a tool which enables aircraft designers to analyze and evaluate the dynamic behavior during the design phase.

The main objective of this paper is to focus on the necessity for developing an interactive, multidisciplinary engineering tool for predicting the unsteady critical states of complex maneuvering aircraft. Such a simulation environment has to bring together aerodynamics, aeroelasticity and flight mechanics in a time-accurate simulation tool. In order to deliver such a tool in the near future, the DLR Project SikMa-“Simulation of Complex Maneuvers” has been initiated to combine these three disciplines into one simulation environment [18][19].

For validating the numerical simulations several wind tunnel experiments in both the low speed and transonic flow regime are be done within the SikMa project.

2 Numerical Approach

2.1 CFD Solver TAU

The behavior of the fluid-flow affecting the object of interest is simulated with the TAU-Code, a CFD tool developed by the DLR Institute of Aerodynamics and Flow Technology [9][10]. The TAU-Code solves the compressible, three-dimensional, time-accurate Reynolds-Averaged Navier-Stokes equations using a finite volume formulation. The TAU-Code is based on a hybrid unstructured-grid approach, which makes use of the advantages that prismatic grids offer in the resolution of viscous shear layers near walls, and the flexibility in grid generation offered by unstructured grids. The grids used for simulations in this paper were created with the hybrid grid generator Centaur, developed by Centaur Soft [3]. A dual-grid approach is used in order to make the flow solver independent from the cell types used in the initial grid. The unstructured grid approach is chosen due to its flexibility in creating grids for complex configurations, e.g. a full-configured fighter aircraft with control surfaces and armament, the capability of grid adaptation and straight forward parallelization of all the main TAU modules.

The TAU-Code consists of several different modules, among which are:

- The Pre-processor module, which uses the information from the initial grid to create a dual-grid and the coarser grids for multi-grid.
- The Pre-processor module, which uses the information from the initial grid to create a dual-grid and the coarser grids for multi-grid.

- The Adaptation module, which refines and de-refines the grid in order to capture flow phenomena like vortex structures and shear layers near viscous boundaries, among others.
- The Deformation module, which propagates the deformation of surface-coordinates to the surrounding grid.
- The Post-processing module, which is used to convert TAU-Code result-files to formats usable by popular visualization tools.

The Solver module contains several upwind schemes, as well as a central scheme with artificial dissipation, which are available for the spatial discretization. For simulations of turbulent flows, the oneequation Spalart-Allmaras and several two-equation turbulence models are implemented. For steady computations either an explicit Runge-Kutta type time-stepping or an implicit LU-SSOR-scheme [5] are used in combination with the multi-grid technique. For time-accurate simulations an implicit dual-time stepping approach is used.

The TAU-Code can handle simulations containing multiple bodies in relative motion with one another, e.g motion of control surfaces with respect to the aircraft, by use of a hierarchical motion-node structure. The motion of each body can either be calculated internally by the TAU-Code, or supplied by an external program through a Python implemented external interface.

2.1.1 TAU-Code Module: Deformation

The Deformation module accepts deformed surface-coordinates either as absolute positions, or as relative displacements from the previous surface grid. User defined, rigid body motions of the surface coordinates can also be used to specify the grid deformation. The input used for the cases presented in this paper is a deformed surface-grid created by an external program. A deformed surface-grid, containing points with new absolute positions, is written to a file which the Deformation module reads. The deformation of the surface grid is propagated through the primary grid, and a new primary grid is created. The Preprocessor module uses this new primary grid to create the dual fine and coarse grids required by the Solver module.

2.1.2 TAU-Code Module: Motion

The Motion module is not a stand-alone executable but a library of functions that handle the rigid-body translational and rotational transformation matrix calculations for the TAU-Code. The module is built to take advantage of naturally occurring hierarchical motion structures, where for example flaps and slats inherit the motion of the wing to which they are attached. Several modes of motion description are allowed, of which the most common are the following:

- periodic - which allows the user to enter a reduced frequency (usually obtained from experimental data), and describe the motion using a combination of Fourier and polynomial series. For periodic motions the user has to specify the number of

time-steps per period, such that the Motion module can calculate the maximum time-step allowed, based on the specified reduced frequency.

- rigid - which allows the user to specify a physical time-step size while using the same type of motion description as for the periodic motion. For periodic motions the user has to calculate the appropriate time-step based on the desired number of time-steps per period. For non-periodic motions the user can select a time-step which can sufficiently resolve the prescribed motion.
- rotate - which allows the user to specify a constant rotation around a given axis using a reduced frequency as input parameter. The user has to specify the number of time-steps per period, such that the Motion module can calculate the maximum time-step allowed, based on the specified reduced frequency.
- external - which allows the user to create motion parameters (angles, rates, translation, displacement) in an external program and send those to the TAU-Code through a Python interface to the Motion module.

The rotation of a body can be described around either the body-fixed coordinate axis (as defined by the DIN 9300 standard), or around a vector defined in space (a so-called hinge-line vector). The translation of a body is specified in the body-fixed reference frame of the parent-node in the motion hierarchy (the inertial reference frame being the parent-node for the entire simulation); an exception to this is when the translation is obtained from the flight-dynamics interface, in that case the translation is specified in the body-fixed frame of the current node itself.

The Motion module uses the given input to create the transformation matrices required to determine the current position of the surface-grids relative to the inertial system, and the relative position of one grid with respect to another for multi-body simulations.

2.1.3 TAU-Code Extension: Chimera Technique

The chimera technique provides the capability to perform calculations with systems of overlapping grids. By allowing large relative body movement without the need for local remeshing or grid deformation, the technique is invaluable for the simulation of maneuvering combat aircraft, where large-amplitude control surface deflections and/or store release are a standard part of the simulation. The current implementation can handle multi-body simulations where the overlapping grid boundaries have been pre-defined; a version that allows ‘automatic-hole-cutting’ is currently under development. The chimera search algorithm, which is based on a state-of-the-art alternating digital tree (ADT), is available for both sequential and massively parallel architectures. A more detailed description of the chimera approach is given in [13].

2.2 Flight Mechanics

For the numerical simulation of the flight mechanics, the simulation environment SIMULA developed at the DLR Institute of Flight Systems is used [14]. SIMULA provides the three basic functionalities necessary for flight simulation and flight control purposes: trimming, i.e. the determination of the initial state and control values, linearization

and stability analysis, and simulation, i.e. the numerical integration of the equations of motion.

Single and multi-body flight-mechanic models, ranging from 1 to 6 degrees of freedom, are made available to the simulation by SIMULA. The amount of data that is exchanged between SIMULA and TAU is of a scale that can be easily communicated directly through a TCP/IP socket connection, which is offered by the simulation environment TENT.

2.3 Structural Dynamics

For the coupling of the aerodynamic and structural dynamic simulations in the time domain, two different and independent approaches have been implemented to gain simulation redundancy and to minimize the project risk. Furthermore, it is necessary to have different approaches with adjusted structural dynamic models depending on the complexity of the simulation problem. Both approaches are based on a loose coupling scheme and use different software for the spatial coupling, structural dynamics and flight mechanics simulation. For the presented applications the loose coupling scheme is sufficient. The time increments on the CFD side are small enough so that the data exchange in each pseudo time step is not necessary. The quality of the coupling is considered by equilibrium verification of loads, energies and work at each physical time step.

For the numerical verification of the coupled procedures, validated FE-Models of the generic delta wing configuration which comprises the flexible delta wing mounted on the flexible wind tunnel support are available [11][15]. These models have been developed based on results of both ground vibration and static deformation tests.

2.3.1 Modal Approach

The approach described here is characterized by the use of the multibody system SIMPACK [12] to account for the elastic structure as well as the flight mechanics and its loose coupling to the computational fluid dynamics software TAU. It is called modal approach since the structural elasticity is introduced from a modal solution of the discrete FE-Model, thus receiving a linearly approximated and reduced elastic model which is based on a small number of modal degrees of freedom only. A reasonable number of structural modes have to be chosen to represent the appropriate dynamic behavior. The generic cosimulation interface of the multibody system (MBS) is used for the data exchange to the non-standard CFD partner code which provides the time-accurate aerodynamic solution. The exchanged data is spatially interpolated with the mesh coupling software MpCCI [8] and transferred through a TCP/IP socket.

The features of the modal approach comprising the topics of time and spatial coupling, structural and flight mechanical models are the following:

- Loose coupling for constant communication interval with master (CFD-TAU) and slave (MBS Simpack) process; the underlying time coupling scheme is the “Conventional Serial Staggered” (CSS) algorithm [7].

- MBS time integration method is an implicit BDF2 algorithm (SODASRT, DASSL based).
- Use of conservative and non-conservative, element based interpolation algorithms to map the aerodynamic forces and the deformed mesh coordinates, respectively (MpCCI library).
- Description of structural elasticity with a reduced, modal approximation computed from the FE-Model.
- Consideration of all translation and rotation degrees of freedom in terms of flight mechanics from the MBS functionality.

The computation of the aerodynamic loads in TAU initiates the coupled computation and acts as master of the co-simulation (loose coupling). SIMPACK delivers the deformed coordinates of the coupling surface as slave. The deformed coordinates are interpolated by MpCCI, and then propagated by the TAU deformation tool into a deformed CFD mesh which is pre-processed for the CFD solver. A more comprehensive description of the simulation platform developed in the modal approach is given in [1].

2.3.2 Discrete Approach

The second aeroelastic method is the so called discrete approach. The underlying spatial coupling scheme is conservative with regards to the forces, moments and the work performed on both the aerodynamic and structure dynamic side. Furthermore, it is verified that no dissipation or accumulation of net energy occurs. This means that at each time-step the sum of kinetic and potential energy of the structure mechanical model and the performed work of the forces on the aerodynamic surface are in equilibrium. The main characteristics of this fluid structure interaction in the time domain are as follows:

- loose coupling of computational fluid dynamics (CFD) and computational structure dynamics (CSD) through netcdf file input/output,
- time coupling scheme based on adjusted “Conventional Serial Staggered” (CSS) [6] algorithm modified with a predictor-corrector scheme,
- use of an implicit or explicit Newmark algorithm for the time integration of the CSD equations of motion [16],
- use of different scattered data interpolation methods with and without compact support radius for coupling in space domain [2],
- description of the structure mechanic behavior by the reduced, discrete FE-Model of the deltawing and the support.

Within this approach for the numerical integration of the structure mechanics equations of motion with the Newmark algorithm, reduced system matrices M_{AA} and K_{AA} from a NASTRAN eigenvalue solution are used and updated in each time-step to represent

the position change of the delta wing during the flight maneuver. Inside of these matrices only the translatory degrees of freedom that are involved in the spatial coupling algorithm are included. One advantage of this approach is, that all modes of the reduced finite elements structure model and their dependency of each other are considered in the coupling between the different discretized aerodynamic and structure mechanics models.

The software package for this approach is called COUPLING and developed at the DLR Institute of Aeroelasticity. It consists of different subroutines and is written in the MATLAB language. The tool can also be used independently from MATLAB if it has been compiled with the appropriate MATLAB compiler in the C or C++ language [6].

2.4 Integration Framework

For providing an applicable engineering tool it is necessary to have an integration framework organizing the communication between the applications, the management of the simulation scenarios, the data transfer and the capability to distribute the simulation on different computational platforms adapted for the different numerical codes.

The integration framework TENT [17] provides a graphical user interface for controlling and monitoring coupled simulation workflows. The various codes used in the SikMa simulations will be made available in the TENT system, where a simulation workflow can be built by connecting icons representing each code using a graphical workflow editor. Java wrappers containing the basic control functionality for the TAU and SIMULA applications are already integrated in the TENT environment. The wrapper for the CSMCode as well as the extension of the functionality to handle the coupling between all three disciplines is under development.

While TENT is providing the data transfer, the communication between the applications and the distribution of the applications on different computational platforms, the communication logic for the simulation workflow is contained within a coupling manager script. The coupling manager is a userextensible script based on a Python and Java interface, where functionality to control the flow of the simulation has been implemented. In Figure 1 the graphical user interface of TENT is shown as well the preprocessing tool SimBrowser. The SimBrowser provides the capability to setup the model hierarchy, to define the motion of each element (e.g. flaps, ruder...) and is delivering the necessary motion- and hierarchy-input files for the flow solver.

3 Experimental Data

For the validation of the numerical simulation software, various wind-tunnel experiments, designed specifically for the SikMa project, are performed. Experimental data, both steady and unsteady, are available for a 65°-swept delta-wing-fuselage-configuration which has been tested in the DNW Transonic- Wind-Tunnel Göttingen (DNW-TWG). The model has movable trailing-edge flaps and can be used for both guided and free-to-roll maneuver simulations around its longitudinal axis. The model shown in Figure 2 has a chord length of 482mm and a span of 382mm. For the verifica-

tion of the aerodynamic-structure coupling a static and dynamic system identification of the delta-wing and the support within the wind tunnel is done. The system parameters are used to set up the FE-Model for the coupled simulation.

The main experiments for the SikMa project are done in the DNW Low-Speed-Wind-Tunnel Braunschweig (DNW-NWB). In order to perform these experiments, a wind-tunnel model has been designed and built for the SikMa project. The model, shown in Figure 3, is based on the X-31 experimental high angle-of-attack aircraft configuration. The model is equipped with remote controlled moveable control devices driven by internal servo-engines, as seen in Figure 4. Measurement equipment is installed to determine the aerodynamic forces and moments on the model, as well as span-wise pressure distributions at locations of 60% and 70% chord length. The experiments include steady-state measurements using PSP-“Pressure Sensitive Paint”, which provide detailed information on the surface pressure distribution for the whole wing. The experiments will culminate with maneuver simulations, where the movement of the aircraft and the control devices will be synchronized. For the maneuver experiments the model will be mounted on the MPM-“Model Positioning Mechanism” of the DNW-NWB. Figure 5 shows the X-31- Remote-Control model mounted on the MPM-System.

4 Results

For the verification and validation of the simulation environment, the results of the numerical simulations are compared against data collected from various experimental simulations. To show the capability of the TAU-Code to predict the unsteady aerodynamic behavior of configurations with vortex dominated flow fields the delta-wing-configuration described in section 3 is used.

In Figure 6 the result of a coupled simulation between CFD and flight-mechanics is shown using the delta-wing with trailing-edge flaps. To simulate the motion of the control surfaces (trailing-edge flaps) the chimera approach has been used. The maneuver shown in Figure 6 is a 1 DoF rotation around the longitudinal axis of the delta-wing induced by an asymmetric deflection of the flaps by $\eta = \pm 3^\circ$. The initial aircraft attitude is at $\alpha = 9^\circ$ and $\Phi = 0^\circ$. The dashed line represents the calculation without any mechanical friction. The solid line represents the calculation supposing a constant dynamic friction. It is seen that the wing tends to go into a periodic roll motion in both the numerical and experimental simulations. It is also seen that the determination of the initial conditions are not correct and the static friction leads in the calculation to a different aerodynamic behavior at the beginning of the simulation. Beside the incorrect determination of the static friction, only a one equation turbulence model is used in this coupled simulation. It is known that for sharp leading-edge delta-wings the $k\omega$ turbulence model delivers better results and will be used in further simulations.

In Figure 7 a CFD - flight-mechanics coupled simulation of the delta-wing with trailing edge flaps is depicted. The initial attitudes are now $\alpha = 17^\circ$ and $\Phi_0 = 0^\circ$. The trailing-edge flaps are deflected by $\eta = \pm 5^\circ$ once the model has been released. The turbulence model used for this simulation is the $k\omega$ -model with the Kok-TNT-rotational correction approach [4]. Two calculations are done using this configuration. The first calculation is done without taking into account the effects of mechanical friction, while for the second

calculation a mechanical friction of 3.5Nm, which is approximately determined from the experimental data, is implemented. The characteristic movement of the model, as well as the roll-moment coefficient, are well predicted by the second calculation. By analyzing the roll-moment coefficient we observe the following:

- An asymmetric surface force-distribution develops due to the asymmetric trailing-edge flap deflection, which in turn leads to a rotational acceleration around the longitudinal axis of the model.
- The maximum roll-moment coefficient is reached after a simulation time of 0.05s, where the flaps are at $\eta = 2.5^\circ$ deflection. After this the roll-moment coefficient decreases and reaches a temporary plateau at $t = 0.1s$, at which time the flaps are fully deflected at $\eta = 5^\circ$.
- The model reaches a trim-point at $\Phi = 31^\circ$, where the combined roll-moment is not large enough to overcome the mechanical friction.

The reason for the movement of the model is graphically explained in Figure 8. At the start of the simulation the wing is accelerated due to the asymmetric flap deflection, see Figure 8 (1). The vortex on the luff side of the wing is strengthened with the increasing roll angle. The effective sweep angle on the luff side of the wing is decreasing, which in turn increases the normal component of the on-flow vector. This causes a stronger primary vortex on the luff side, which is located closer to the surface, thus leading to a higher local lift on the luff side. On the lee side the opposite effect happens. The wing vortex gets weaker and the distance from the wing surface higher as the roll angle increases, which leads to a lower local lift on the lee side, see Figure 8 (2). This effect causes the wing to decelerate, which in turn leads to the trim-point at $\Phi = 31^\circ$, Figure 8 (3) and Figure 8 (4).

Both examples show the capability of CFD-flight-mechanics coupling by means of a delta-wing with trailing edge flaps. The main aerodynamic effects are qualitatively well predicted, but in order to predict the quantitative aerodynamic values obtained experimentally it is necessary to have the same starting conditions and environment parameters that were in effect during the experimental maneuver scenario. If the starting conditions are not the same, the prediction of the experimentally obtained final condition can not be guaranteed.

The capability to predict the elastic deformations of the delta-wing configuration during a guided roll maneuver should be demonstrated as follows. In the coupled simulation between the TAU-Code and the structural mechanics tool developed within the discrete approach, the delta-wing and the rear-sting support are considered to be elastically deformable. For the coupled simulation, the FE-model [15] takes into account both the delta wing configuration as well as the flexible support, as seen in Figure 9. The FE-Model is validated based on results of both ground vibration and static deformation tests. The aerodynamic simulations base on Euler calculations using a coarse mesh topology to reduce the unsteady calculation time on the CFD side. For the verification of the capabilities this procedure is sufficient.

In Figure 10 the corresponding history of the model deflection due to the elasticity is seen. Depicted is the displacement of the sting support at the trailing-edge of the

delta wing. It is seen that the sting is describing an elliptic motion during the rotation. The green loop in Figure 10 shows the sting movement from the experiment due to the integration of the measured accelerations. Because of the integration the shift in z-direction can not be captured. However, it is shown that the numerical simulations predict the characteristic movement of the sting accurately. Due to this deformation the effective angle-of-attack of the delta wing at the same roll angle is higher in the elastic case than for the rigid body motion. This leads to higher amplitudes of the rolling moment, as is seen in Figure 11.

The time-history results of the rigid and the coupled elastic simulation of the free-to-roll wind tunnel maneuver are compared in Figure 12 for the roll angle ϕ and in Figure 13 for the rolling moment L, respectively. The on flow mach number is at $Ma = 0.5$, the angle-of-attack is at 9° and the initial roll angle is at $\Phi_0 = 45^\circ$. As described before, the maneuver is initiated by the unsymmetrical load distribution over wing surface due to the initial roll angle. Furthermore, the same effects occur as in the guided maneuver simulation. The deformations of the model in y- and z-direction lead to locally higher angles of attack and thus to higher rolling moments, as seen in Figure 12. In comparison to the rigid body motion in case of the discrete approach the maneuver shows a higher damping, as seen in Figure 13. The same effect is expected for the modal approach but in this case the CFD solution is not sufficiently converged to get the same effects. However, the capability and necessity is shown to consider the structural behavior in maneuver simulations.

For the X-31 configuration, results from steady-state numerical simulations have been obtained. These simulations show the capability of the TAU-Code to simulate complex delta-wing configurations with rounded leading-edges. Figure 14 shows the numerically simulated 3D flow field over the X-31 configuration, which is a good indication of the complexity of the vortex flow topology over the wing and fuselage. Comparisons with experimental data show good agreement regarding the vortex topology. In Figure 15 (a) an oil flow picture of the X-31 clean-wing from low speed experiments is shown. The angle-of-attack is $\alpha = 18^\circ$ at a Reynolds number of 1.0Mio. The attachment line of the strake vortex and the main wing vortex as well as the separation line of the main wing vortex near the leading-edge is emphasized. In Figure 15 (b) the corresponding CFD calculation is depicted. It is seen that the flow topology from the calculation fits quite well with the experiment. Further experimental results delivering steady pressure distributions upon the wing were done within another X-31 test campaign. The PSP result at $\alpha = 18^\circ$ at a Reynolds number of 2.07Mio is shown in Figure 16 (a). Comparing the pressure distribution from the PSP measurement with the CFD calculation in Figure 16 (b) it is seen that the main footprints of the vortices are captured by the numeric, but not accurately the suction-strength and the vortex location. Further investigations will be done for this complex fighter aircraft configuration to capture the correct aerodynamics by CFD calculations.

5 Conclusions

In this paper the activities and recent results of the DLR-Project SikMa were presented. In SikMa a simulation tool will be developed that is capable of simulating a

maneuvering elastic aircraft with all its moveable control devices. The simulation tool combines time-accurate aerodynamic, aeroelastic and flight-mechanic calculations to achieve this objective. Preliminary verification of the functionality of the simulation tool has been shown by simulating a sharp leading-edge delta-wing during free-to-roll maneuvers due to flap deflections. Furthermore, first perspectives were presented regarding the timeaccurate coupling between the TAU-Code and the numerical Structure Mechanical Tool. Initial results of the steady flow field around the X-31 configuration were presented. The next steps in the SikMa project will be amongst others a coupled simulation of CFD and CSM using a RANS model on the CFD side instead of the Euler simulation and the unsteady maneuver simulation of the X-31 by simulating all control devices. Finally an engineering tool for the simulation of flight maneuvers using high performance numerical tools will be available.

References

- [1] Arnold, J.; Einarsson, G.; Gerhold, T.: *Time-Accurate Simulation of a Maneuvering Aeroelastic Delta Wing at High Angle-of-Attack*. Proc ICAS 2004, Yokohama, Japan; 2004.
- [2] Beckert, A.; Wendland, H.: *Multivariate Interpolation for Fluid-Structure-Interaction Problems using Radial Basis Functions*. Aerospace Science and Technology, Art. No. 5125, 2001.
- [3] *Centaur Soft*: <http://www.Centaursoft.com>
- [4] Dol, H.S.; Kok, J.C.; Oskam, B.: *Turbulence modeling for leading-edge vortex flows*. AIAA 2002-0843, Reno, Jan. 2002.
- [5] Dwight, R.P.: *Time-Accurate Navier-Stokes Calculations with Approximately Factored Implicit Schemes*. Proceedings of the ICCFD3 Conference Toronto, Springer, 2004.
- [6] Einarsson, G.; Neumann, J.: *Multidisciplinary Simulation of a Generic Delta Wing: Aerodynamic, Flight-Dynamic, and Structure-Mechanic Coupling*. ECCOMAS, International Conference on Computational Methods for Coupled Problems in Science and Engineering, Santorini Island, 2005.
- [7] Farhat C.; Lesoinne M.: *Higher-Order Staggered and Subiteration Free Algorithms for Coupled Dynamic Aeroelasticity Problems*. AIAA 98-0516, 1998.
- [8] Fraunhofer Institute for Algorithms and Scientific Computation SCAI: *MpCCI Mesh based parallel code coupling interface*. Specification of MpCCI version 2.0, 2003.
- [9] Galle, M.; Gerhold, T.; Evans, J.: *Technical Documentation of the DLR TAU-Code* DLR-IB 233-97/A43 1997
- [10] Gerhold, T.; Galle, M.; Friedrich, O.; Evans, J.: *Calculation of Complex Three-Dimensional Configurations employing the DLR TAU-Code*. AIAA-97-0167 1997.

- [11] Hoffmann, D.; Neumann, J.; Sinapius, M.: *Strukturmechanische Identifikation von Halterung und Windkanalmodell "AeroSUM"*.DLR-IB, IB 232-2002-C06, 2002.
- [12] Lugner, P.; Arnold, M.; Vaculin, O. (Eds): *Vehicle System Dynamics (Special issue in memory of Professor Willi Kortüm)*.Vol. 41, No. 5, 2004.
- [13] Madrane, A.; Raichle, A.; Stürmer, A.: *Parallel implementation of a dynamic overset unstructured grid approach*.ECCOMAS Conference Jyväskylä Finland, 24.-28. July 2004.
- [14] Mönnich, W.; Buchholz, J. J.: *SIMULA - Ein Programm Paket für die Simulation dynamischer Systeme - Dokumentation und Benutzeranleitung - Version 2DFVLR* Institutsbericht IB 111-91/28, 1991.
- [15] Neumann, J.: *Strukturmechanische und strukturdynamische Finite Element Modelle des Windkanalmodells "AeroSUM" mit Halterung*.DLR-IB, IB 232-2003-J01, 2003.
- [16] Schulze, S.: *Numerische Integration der aeroelastischen Bewegungsgleichungen eines Flgelprofils*.DLR-IB, IB 232-94 J 06, 1994.
- [17] Schreiber, A.: *The Integrated Simulation Environment TENT*.Concurrency and Computation: Practice and Experience, Volume 14, Issue 13-15, S.1553-1568, 2002.
- [18] Schütte, A.; Einarsson, G.; Schöning, B.; Madrane, A.; Mönnich, W., Krüger, W.: *Numerical simulation of manoeuvring aircraft by aerodynamic and flight mechanic coupling*.RTO AVTSymposium Paris, 22.-25. April 2002.
- [19] Schütte, A.; Einarsson, G.; Schöning, B.; Raichle, A.; Mönnich, W.; Neumann, J.; Arnold, J; Heinecke, J.: *Numerical simulation of maneuvering combat aircraft*.AG-STAB, STAB-Symposium, Nov.2004, Bremen.

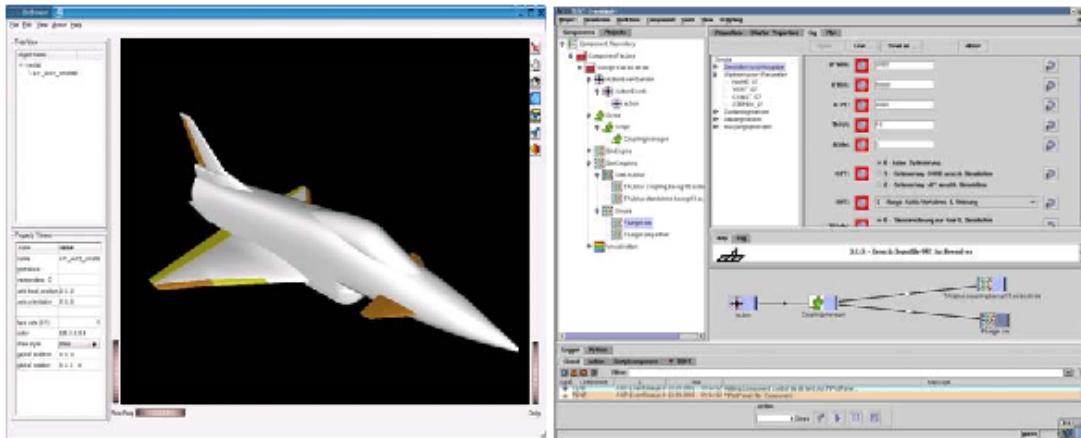


Figure 1: Graphical user interface of TENT framework environment and pre-processing tool SimBrowser.



Figure 2: 65°-swept delta-wing-fuselage-model-configuration with remote controlled trailing-edge flaps on the roll-rig device in the DNW-TWG Göttingen.



Figure 3: X-31 Remote control model in the DNW-NWB Braunschweig.

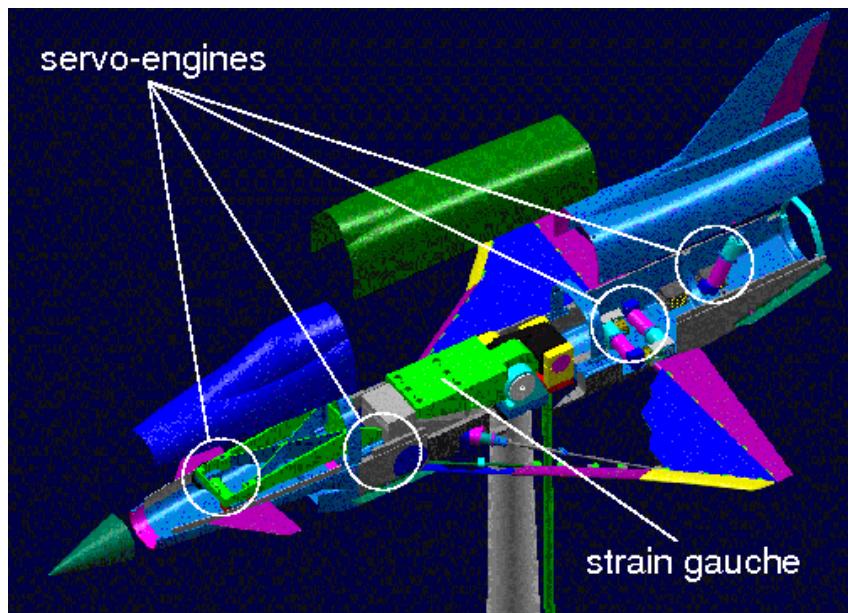


Figure 4: CATIA image of the X-31 wind tunnel model.

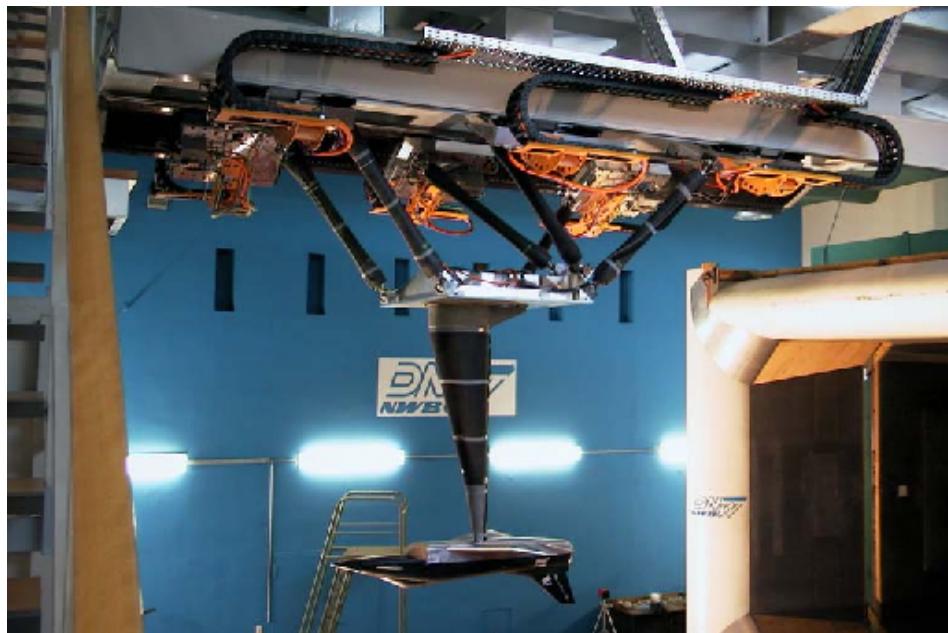


Figure 5: X-31 model on the Model-Positioning-Mechanism in the DNW-NWB wind tunnel.

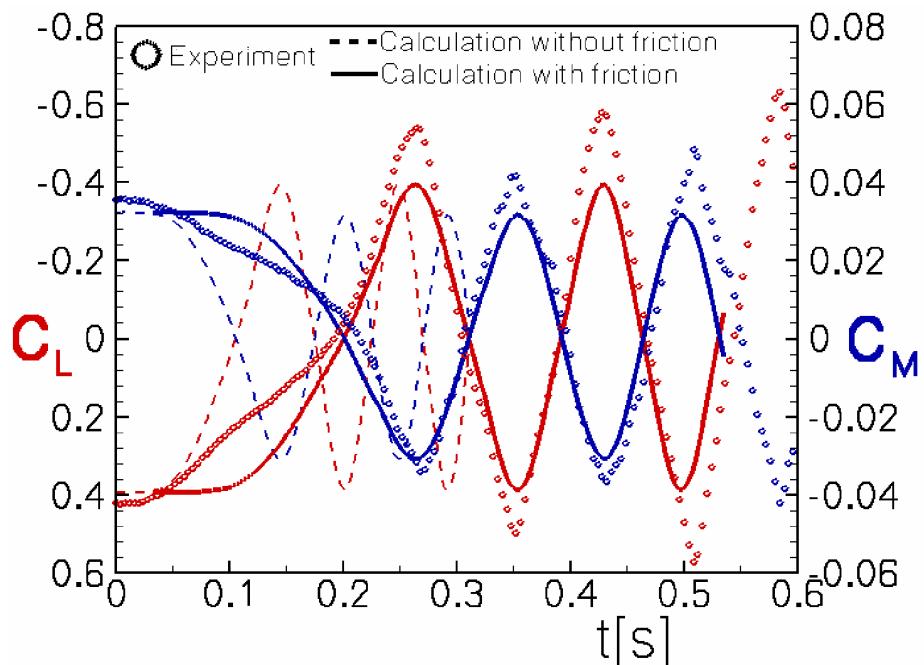


Figure 6: 1 DoF Free-to-Roll maneuver of delta-wing-flap conf. through trailing-edge flap deflection. $Ma = 0.85$, $Re = 5\text{Mio.}$, $\Theta = 9^\circ$, $\Phi_0 = 0^\circ$, $\eta = \pm 3^\circ$.

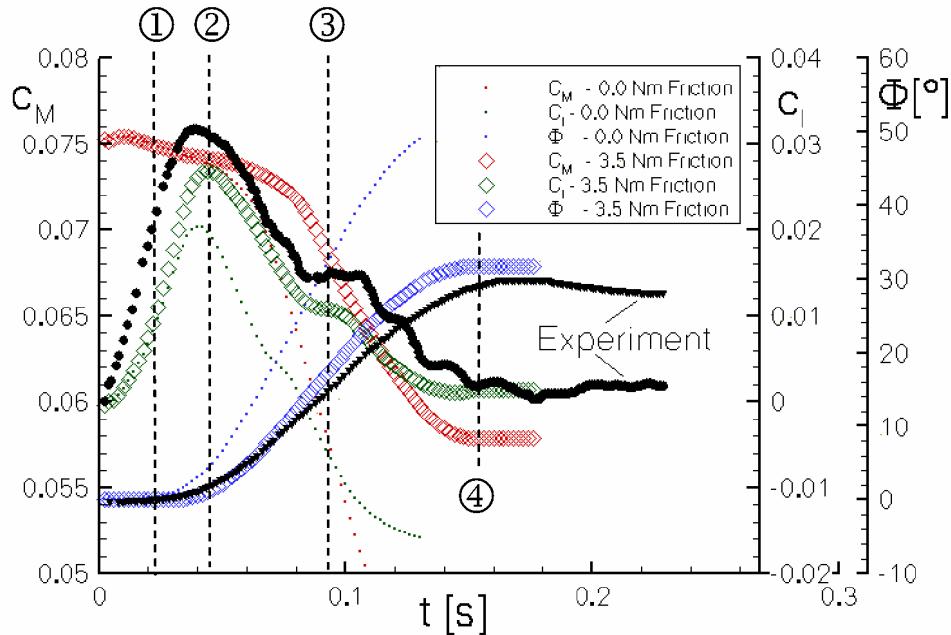


Figure 7: 1 DoF Free-to-Roll maneuver of delta-wing-flap conf. through trailing-edge flap deflection. $\text{Ma} = 0.5$, $\text{Re} = 3.8\text{Mio.}$, $\Theta = 17^\circ$, $\Phi_0 = 0^\circ$, $\eta = \pm 5^\circ$.

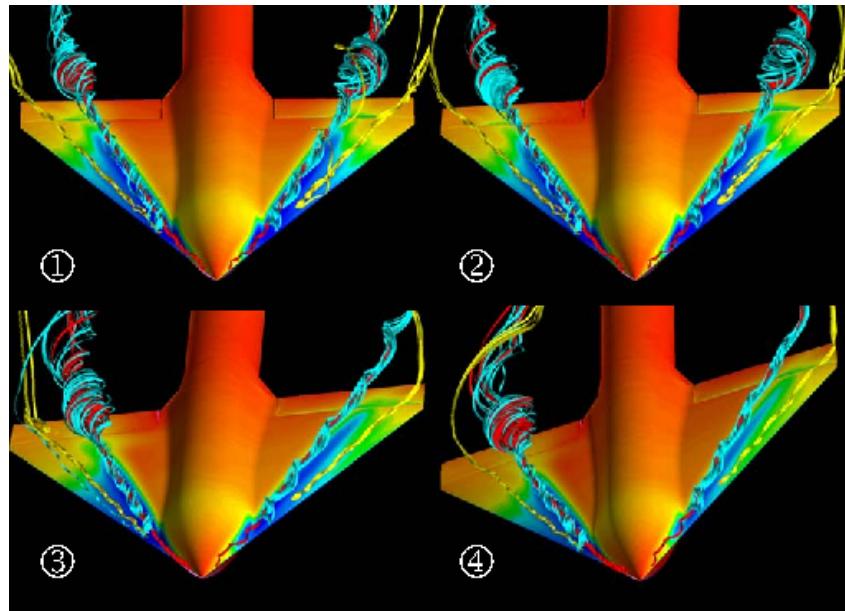


Figure 8: 1 DoF Free-to-Roll maneuver of delta-wing-flap conf. through trailing-edge flap deflection. Flow topology at four different stages. $\text{Ma} = 0.5$, $\text{Re} = 3.8\text{Mio.}$, $\Theta = 17^\circ$, $\Phi_0 = 0^\circ$, $\eta = \pm 5^\circ$.

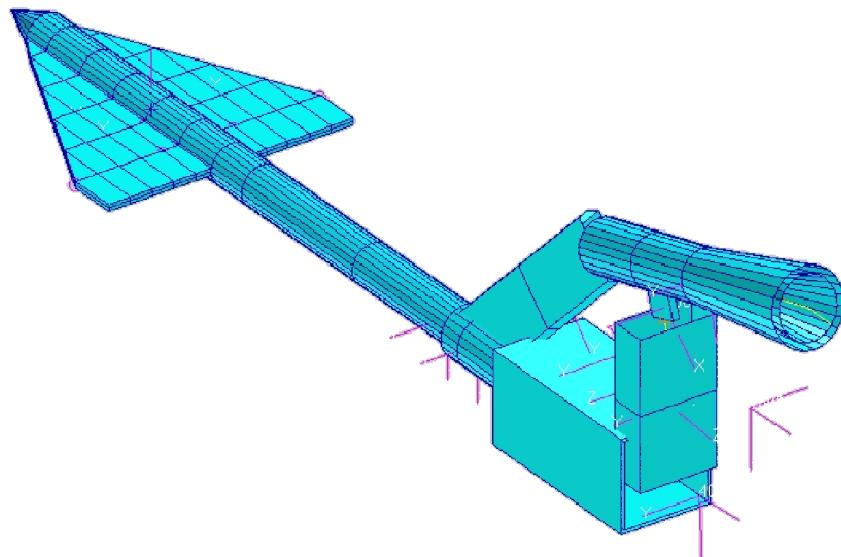


Figure 9: FE-Model of the wind tunnel setup comprising the generic delta wing and the model sting.

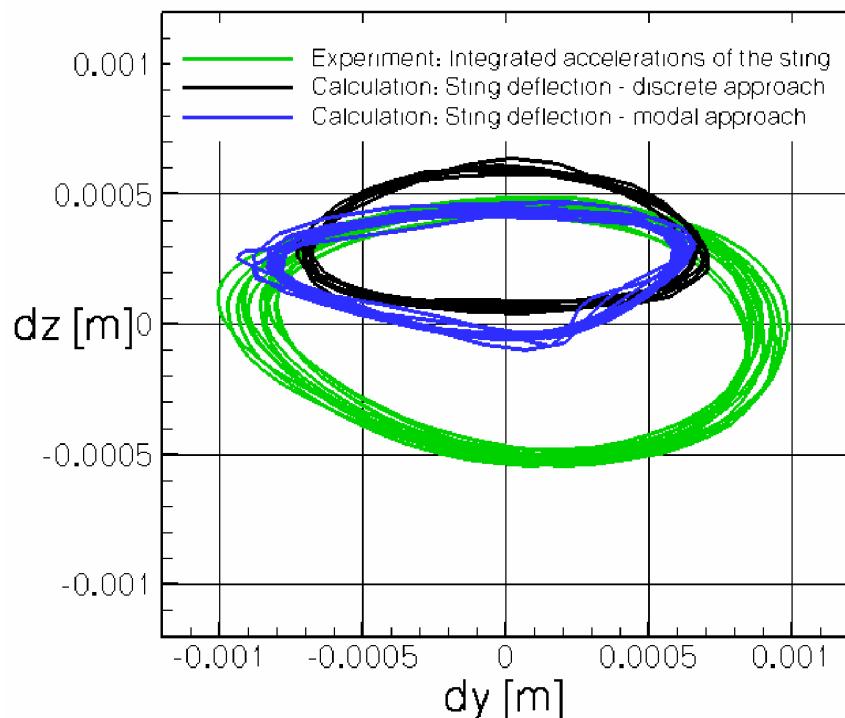


Figure 10: History of the delta-wing sting deflection during elastic-body motion comparison with experiment. Time-accurate coupled CFD(Euler)-CSM simulation.

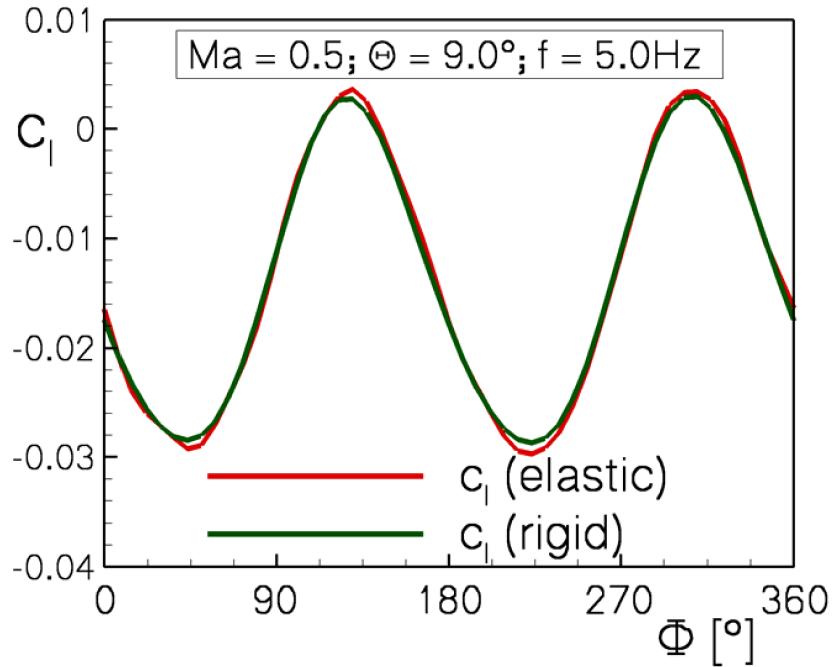


Figure 11: Comparison of rolling moment between rigid- and elastic-body motion of delta-wing during constant rotational movement. Time-accurate coupled CFD(Euler)-CSM simulation.

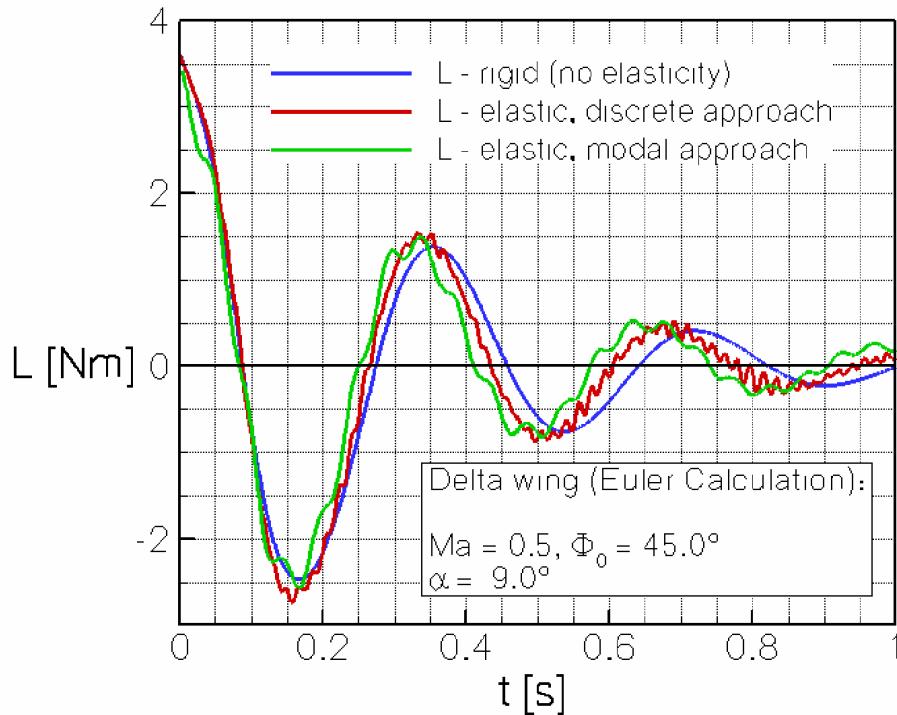


Figure 12: Comparison of the aerodynamic moment L for a free to roll maneuver for rigid motion and elastic motion with different approaches.

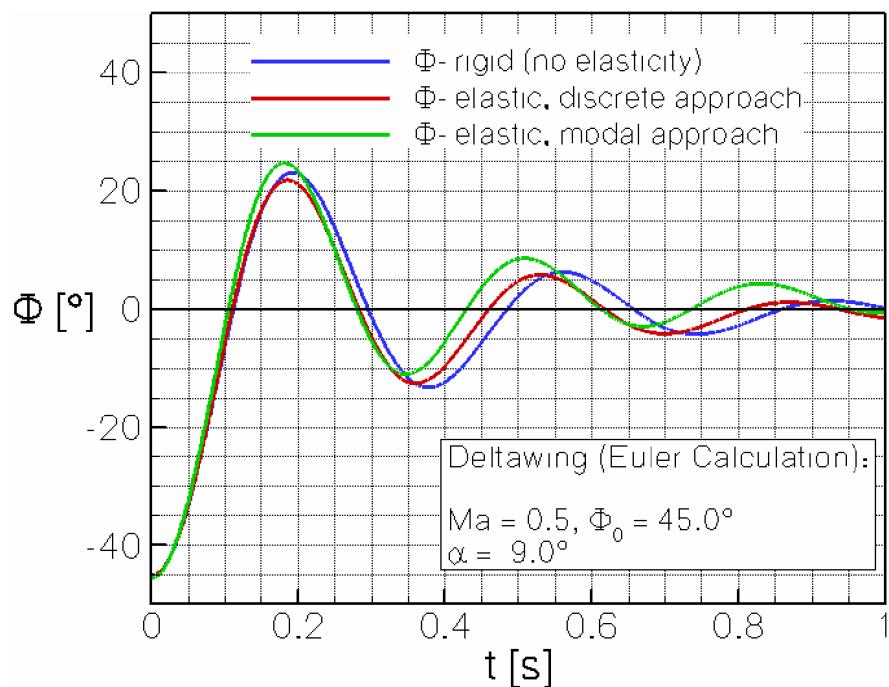


Figure 13: Comparison of the roll angle Θ for a free to roll maneuver for rigid motion and elastic motion with different approaches.

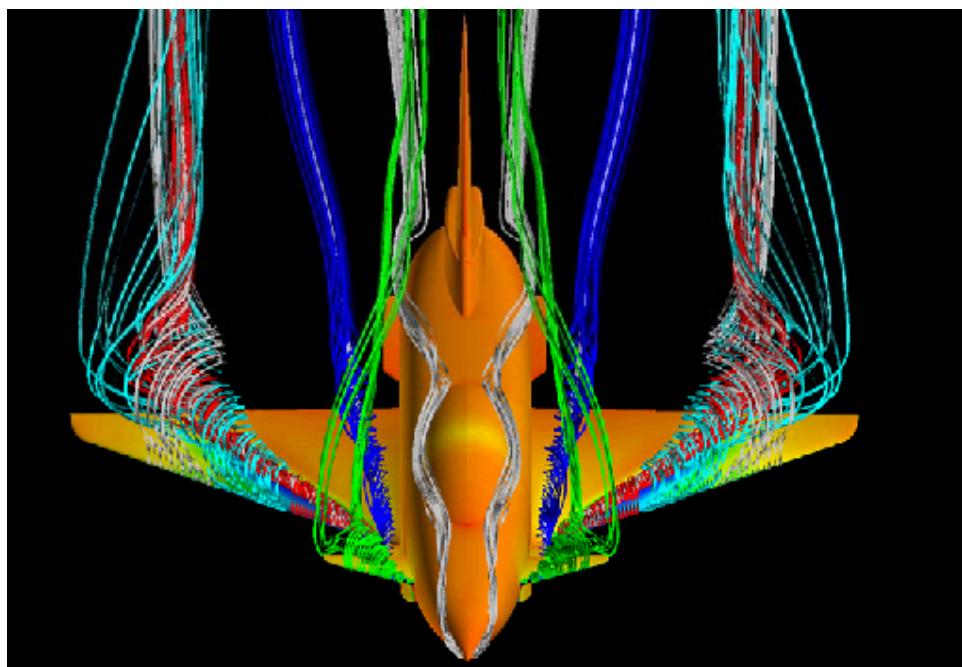


Figure 14: 3D flow field over the X-31 configuration at 18° angle-of-attack.

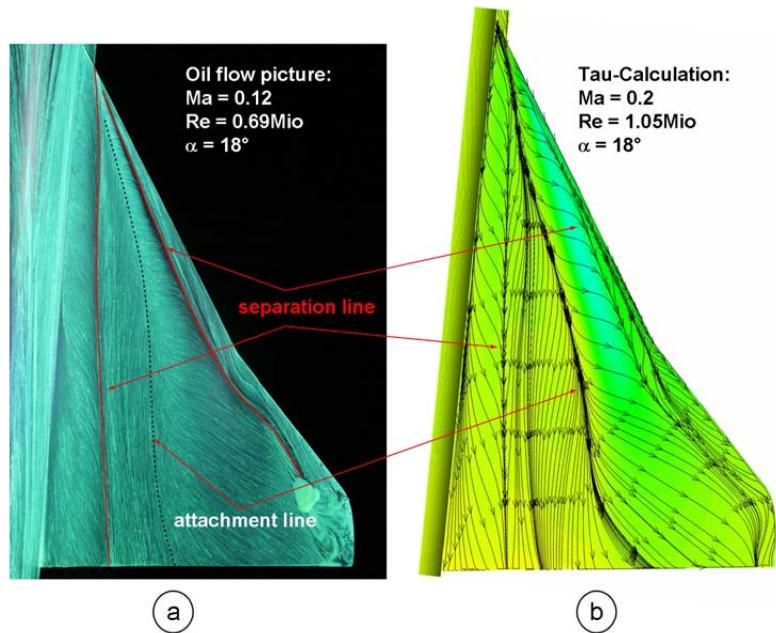


Figure 15: (a) Oil flow visualization of the X-31 clean wing at $\alpha = 18^\circ$.
(b) TAU calculation: Visualization of surface streamlines at $\alpha = 18^\circ$.

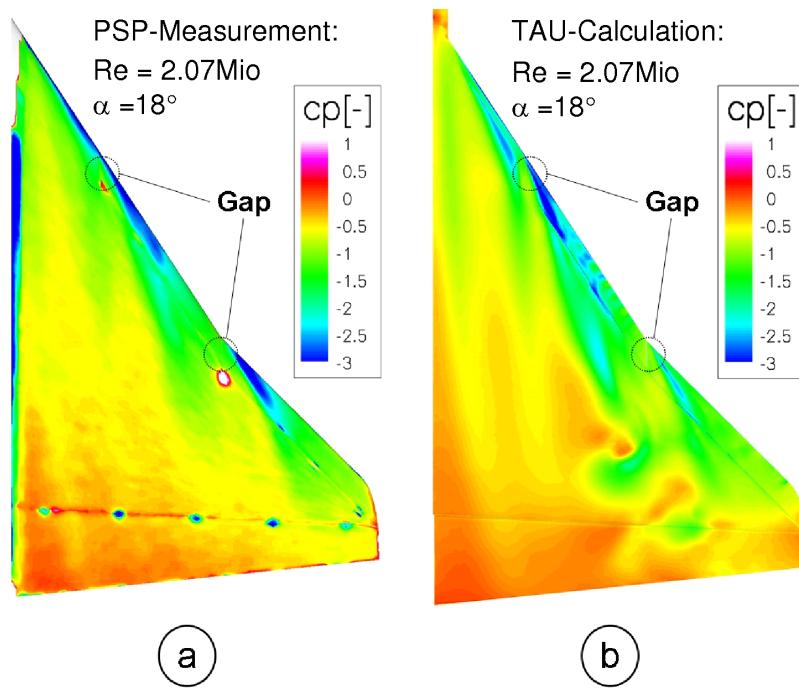


Figure 16: (a) Steady PSP measurement of the pressure distribution over the X-31 wing.
(b) Steady TAU RANS calculation of the pressure distribution over the X-31 wing.

Curriculum vitae

Persönliche Daten

Name	Thomas Alrutz
Geburtsdatum	19.11.1969 in Münden
Familienstand	ledig
Staatsangehörigkeit	deutsch

Schulbildung

08/1976 - 07/1980	Grundschule Hermanshagen, Hann. Münden
08/1980 - 08/1982	Orientierungsstufe I, Hann. Münden
08/1982 - 05/1990	Grotefend Gymnasium, Hann. Münden

Studium

10/1990 - 06/2002	Studium an der Georg-Augusta-Universität Göttingen. Schwerpunkte: Angewandte Mathematik, Informatik Abschluß: Diplom-Mathematiker Thema: Erzeugung von unstrukturierten Netzen und deren Verfeinerung anhand des Adaptationsmoduls des DLR TAU-Codes
06/2002 - 07/2008	Promotionsvorhaben am Institut für Numerische und Angewandte Mathematik Schwerpunkt: Wissenschaftliches Rechnen Thema: Parallele dynamische Adaption hybrider Netze für effizientes verteiltes Rechnen

Wissenschaftlicher & beruflicher Werdegang

11/1992 - 08/1993	Wissenschaftliche Hilfskraft am Max-Planck Institut für Strömungsphysik Göttingen.
06/1997 - 06/2002	Wissenschaftliche Hilfskraft am DLR Göttingen Institut für Aerodynamik und Strömungstechnik Abteilung Numerische Verfahren.
07/2002 - 04/2008	Wissenschaftlicher Mitarbeiter am DLR Göttingen Institut für Aerodynamik und Strömungstechnik Abteilung Numerische Verfahren.
seit 05/2008	System Architekt & Application Manager High Performance Computing T-Systems Solution for Research GmbH, Göttingen