Deutsches Zentrum
für Luft- und Raumfahrt e.V.
in der Helmholtz-Gemeinschaft

RWTH AACHEN UNIVERSITY

Parallele, iterative Lösung dünnbesetzter linearer Gleichungssysteme aus der Strömungstechnik

Parallel iterative solution of sparse linear systems in computational fluid dynamics

Bachelorarbeit

in der Studienrichtung

Computational Engineering Science

an der

Rheinisch-Westfälischen Technischen Hochschule Aachen

von

Melven Zöllner

Externer Betreuer:
Dr. Achim Basermann (DLR)

Betreuer:
Prof. Dr. Arnold Reusken
Christoph Lehrenfeld

23. November 2011

Zusammenfassung

In dieser Arbeit werden iterative Krylov-Unterraum-Verfahren zur Lösung großer, dünnbesetzter linearer Gleichungssysteme auf Parallelrechnern mit verteiltem Speicher untersucht. Den Schwerpunkt dieser Untersuchung bilden Methoden zur Vorkonditionierung, die auf einem Schur-Komplement-Ansatz beruhen.
Diese können als eine iterative Verbesserung einer Block-Jacobi-Vorkonditionierung auf Basis unvollständiger LU-Zerlegungen von Diagonalblöcken der Matrix formuliert werden.
Als Beispiele dienen Gleichungssysteme aus Anwendungen zur Strömungssimulation des Deutschen Zentrum für Luft- und Raumfahrt.
Das Verhalten der Methoden zur Vorkonditionierung wird anhand empirischer Tests mit diesen Gleichungssystemen bezüglich der Laufzeit und der numerischen Eigenschaften erörtert.
Die Ergebnisse der durchgeführten Testläufe zeigen, dass sich die parallele Skalierbarkeit des Lösungsverfahrens bei einer Schur-Komplement-Vorkonditionierung deutliche gegenüber einer Block-Jacobi-Vorkonditionierung verbessert.

Abstract

This thesis deals with the solution of large sparse systems of linear equations on many-core architectures with distributed memory using iterative krylov methods. It focusses on preconditioning techniques, especially those that are based on a Schur-complement-Ansatz.
These preconditioners can be considered as an iterative correction of block-Jacobi preconditioning that uses an incomplete LU-decomposition to approximately invert the diagonal blocks of the matrix.
Sample linear systems stem from CFD (computational fluid dynamics) simulation systems, which were developed by DLR (German Aerospace Center).
The runtime and numerical characteristics of the parallel preconditioned solvers developed are discussed by means of empirical tests with these systems.
The results of these tests demonstrate that Schur-complement preconditioners can considerably improve the parallel scalability of an iterative solver in comparison to block-Jacobi preconditioning.

Danksagung

An dieser Stelle möchte ich mich bei den Personen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben.
Besonderer Dank gilt dem externen Betreuer meiner Arbeit Dr. Achim Basermann von der DLR-Einrichtung Simulations- und Softwaretechnik.
Für die Beantwortung meiner zahlreicher Fragen zu den Anwendungen TRACE und TAU danke ich Dr. Hans-Peter Kersken vom Institut für Antriebstechnik des DLR und Dr. Anna Naumovich vom Institut für Aerodynmik und Strömungstechnik.
Für die Betreuung meiner Bachelorarbeit von Seiten der RWTH-Aachen danke ich Professor Dr. Arnold Reusken vom Institut für Geometrie und Praktische Mathematik, dem insbesondere am Herzen lag, dass ich etwas über die hier dargestellten numerischen Verfahren lerne, und Christoph Lehrenfeld, der mir zu vielen Abschnitten der Arbeit hilfreiche Ratschläge gegeben hat.
Ich möchte mich ebenfalls bei Gregor Matura von der DLR-Einrichtung Simulations- und Softwaretechnik bedanken, der Teile der Arbeit Korrektur gelesen hat.

Eidesstattliche Erklärung

Ich versichere hiermit, die vorgelegte Arbeit in dem gemeldeten Zeitraum ohne unzulässige fremde Hilfe verfasst und mich keiner anderen als der angegebenen Hilfsmittel und Quellen bedient zu haben.
Diese Arbeit wurde bisher weder im In- noch im Ausland als Prüfungsarbeit vorgelegt.

Aachen, den 23. November 2011

(Vorname Nachname)

Inhaltsverzeichnis

1. Einleitung

Kürzeste Ausführungszeiten werden auf einem Parallelrechner von Verfahren erreicht, die sich in viele ähnlich aufwendige Teilprobleme zerlegen lassen, die gleichzeitig und möglichst unabhängig voneinander gelöst werden können. Dies sind nicht unbedingt die Verfahren mit dem geringsten Aufwand (im Sinne der Zahl der benötigten Rechenoperationen) zur Lösung eines bestimmten Problems.

In dieser Arbeit werden iterative Verfahren zur Lösung großer, dünnbesetzter linearer Gleichungssysteme untersucht. Den Schwerpunkt dieser Untersuchung bilden Methoden zur Vorkonditionierung auf Parallelrechnern, insbesondere solche, die auf einem Schur-Komplement-Ansatz beruhen.
Als Beispiele dienen Gleichungssysteme aus Anwendungen der Strömungstechnik. Hier werden zwei spezielle Anwendungen des DLR (Deutsches Zentrum für Luft- und Raumfahrt[1]) vorgestellt, deren Hauptaufwand in der Lösung großer linearer Gleichungssysteme besteht.
Die in dieser Arbeit beschriebenen Verfahren werden jeweils im Hinblick auf die spezifischen Eigenschaften dieser Gleichungssysteme betrachtet, sind aber auch zur Lösung dünnbesetzter linearer Gleichungssysteme aus anderen Anwendungen geeignet.

Die Arbeit ist folgendermaßen aufgebaut:
Zunächst werden kurz die beiden Anwendungen aus der Strömungstechnik beschrieben und einzelne lineare Gleichungssysteme aus diesen Anwendungen als Testfälle vorgestellt.
Es folgt eine Darstellung der Grundlagen von Krylov-Unterraum-Verfahren.
Anschließend wird die Frage geklärt, welche besonderen Anforderungen an ein Verfahren sich durch die Verwendung eines Parallelrechners ergeben.
Im Hinblick darauf werden verschiedene Möglichkeiten der Vorkonditionierung beschrieben und dann eine spezielle Schur-Komplement-Methode dargestellt.
Nach einer groben Beschreibung der Implementierung eines parallelen iterativen Lösers mit Schur-Komplement-Vorkonditionierung wird das Verhalten dieses Lösers bezüglich der Laufzeit und numerischer Eigenschaften anhand von Gleichungssystemen aus der Strömungstechnik erörtert. Als Letztes werden die Schlussfolgerungen aus den vorherigen Abschnitten der Arbeit zusammengefasst sowie Probleme und mögliche Verbesserungen der hier verwendeten Methoden angesprochen.

[1] Siehe www.dlr.de

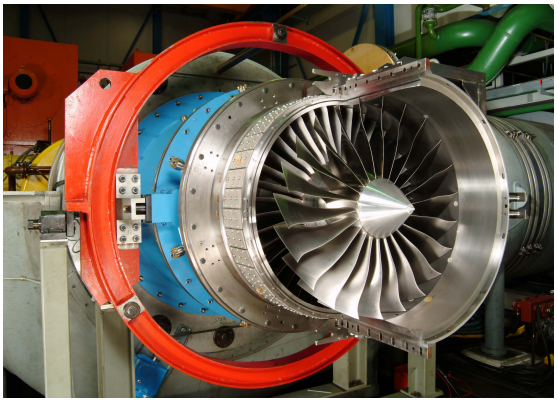


Abbildung 1: Testaufbau des UHBR-Fans (Quelle: Institut für Antriebstechnik, DLR)

2. Anwendungsbeispiele

In den folgenden Abschnitten werden zwei spezielle Anwendungen aus der Strömungstechnik beschrieben, die das Lösen großer, dünnbesetzter linearer Gleichungssysteme erfordern.
Die Diskretisierungen, die zu diesen Gleichungssystemen führen, werden jeweils kurz skizziert. Da in beiden Fällen turbulente Strömungen betrachtet werden, sei hier zum Verständnis auf [9] verwiesen. Eine ausführliche Darstellung von Finite-Volumen-Methoden zur Diskretisierung findet man in [8].

2.1. Aeroelastische Analyse in TRACE [7]

Das Simulationssystem *TRACE* („Turbomachinery Research Aerodynamics Computational Environment“) dient der Berechnung von dreidimensionalen Turbinenströmungen. Es wird am Institut für Antriebstechnik des DLR entwickelt und von verschiedenen Universitäten und dem Triebwerkshersteller „MTU Aero Engines“[2] verwendet.
TRACE basiert auf einer Diskretisierung der kompressiblen, instationären Reynolds-Gleichungen[3] (*RANS*, „Reynolds-Averaged-Navier-Stokes“) mit Hilfe von Finiten Volumen auf strukturierten oder unstrukturierten Gittern. Diese Gleichungen lassen sich in Erhaltungsform schreiben als:

$$\frac{\partial q}{\partial t} + \operatorname{div} F(q) + S(q) = 0 \tag{1}$$

Dabei bezeichnet $q : \mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{R}^5$, $q := \begin{pmatrix} \rho & \rho U_1 & \rho U_2 & \rho U_3 & \rho E \end{pmatrix}^T$ den Vektor der Erhaltungsvariablen, hier die Dichte ρ, den Geschwindigkeitsvektor in drei Richtungen $\begin{pmatrix} U_1 & U_2 & U_3 \end{pmatrix}^T$ und die Energie E. Da hier ein rotierendes Bezugssystem in einer Turbine betrachtet werden soll, werden mit dem Quellterm $S(q)$ Koriolis- und Zentrifugalkraft

[2] www.mtu.de
[3] eine vereinfachte Form der Navier-Stokes-Gleichungen, bei der die Turbulenz der Strömung modelliert und nicht auf einem feinen Gitter aufgelöst wird

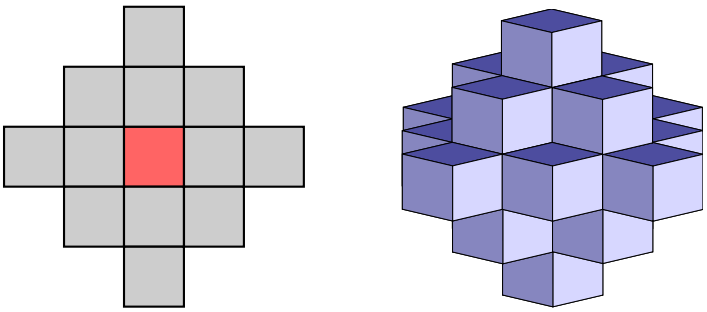


Abbildung 2: Abhängigkeitsbereich einer Zelle im zwei- und dreidimensionalen Fall

berücksichtigt. Der Term $F(q)$ ist die Flussfunktion. Modelliert wird in den hier betrachteten Fällen eine turbulente, kompressible Strömung eines viskosen idealen Gases.
Zerlegt man das Rechengebiet in Teilvolumina (Zellen), erhält man mit Hilfe des Satzes von Gauß als Formulierung für die Finite-Volumen-Diskretisierung mit den Volumen der Zellen V_i, den Flüssen über die Ränder der Zellen $F'_{ij}(\mathbf{q})$, den Quellen in jeder Zelle $S_i(\mathbf{q})$ und den Zellmittelwerten der Erhaltungsvariablen q_i:

$$\frac{\partial}{\partial t}(V_i q_i) + \sum_{\text{Seiten } j \text{ der Zelle } i} F'_{ij}(\mathbf{q}) + S_i(\mathbf{q})V_i = 0 \tag{2}$$

Fett gedruckte Zeichen (wie $\mathbf{x}$ und $\mathbf{q}$) bezeichnen hier Vektoren auf dem Finite-Volumen-Gitter diskretisierter Variablen.
In den hier verwendeten Testfällen werden nur strukturierte Hexaeder-Gitter verwendet. Für die Berechnung der Flüsse wird das MUSCL-Schema verwendet, eine Art Upwind-Diskretisierung zur Darstellung der räumlichen Ableitungen von q. Dadurch sind die Werte q_i aus Zellen im Inneren des Rechengebietes wie in Abbildung 2 skizziert nur von den Werten aus einigen umliegenden Zellen abhängig.
Da eine Turbine aus vielen einzelnen, meist identischen Schaufeln aufgebaut ist, kann die Turbinenströmung anhand der Geometrie einer oder einiger weniger Schaufeln und der Verwendung periodischer Randbedingungen berechnet werden.

Anwendungsgebiete für einen linearen Gleichungssystemlöser sind unter anderem die Module *adjointTRACE* zur Formoptimierung und *linearTRACE* zur aeroelastischen Analyse.
Hier wird nur das Modul *linearTRACE* betrachtet, mit dem die Auswirkungen von Schwingungen der Turbinenschaufeln auf die Strömung untersucht werden können. Diese Schwingungen werden als kleine harmonische Störungen der Koordinaten $\mathbf{x}$ des Finite-Volumen-Gitters (und insbesondere der Koordinaten der Ränder) modelliert:

$$\mathbf{x}(t) = \mathbf{x}^0 + \mathrm{Re}\left(\tilde{\mathbf{x}}(\mathbf{x}^0)e^{i\omega t}\right) \tag{3}$$

In dieser Formel bezeichnet $\mathbf{x}(t) \in \mathbb{R}^{3n_G}$ die aufgrund der Schwingungen zeitlich variablen Gitterkoordinaten (zu einem der n_G Gitterpunkte gehören jeweils drei Koordinaten). Der Vektor $\mathbf{x}^0 \in \mathbb{R}^{3n_G}$ enthält die Koordinaten der *Ruheposition*, um die die

Gitterpunkte mit einer Kreisfrequenz von $\omega \in \mathbb{R}_+$ schwingen. Die Amplitude und die Phasenverschiebung der Schwingung jedes Gitterpunktes wird durch $\tilde{\mathbf{x}}(\mathbf{x}^0) \in \mathbb{C}^{3n_G}$ festgelegt.
Die Strömung – das heißt die Vektoren der Erhaltungsvariablen q_i aus jeder Zelle – hängt dann von den zeitlich variierenden Gitterkoordinaten $\mathbf{x}(t)$ ab und lässt sich als Funktion $\mathbf{q}(\mathbf{x}(t), t) \in \mathbb{R}^{5n_Z}$) darstellen ($n_Z$ ist die Anzahl der Zellen), die durch die diskretisierten Erhaltungsgleichungen (2) und die Randbedingungen definiert wird.
Unter der Voraussetzung, dass man ohne Störung (also für $\tilde{\mathbf{x}} = 0$) eine stationäre Lösung $\mathbf{q}^0(\mathbf{x}^0)$ erhält (die wegen der Turbulenz keine stationäre Strömung darstellen muss), kann die Funktion um den Punkt $\mathbf{x}^0$ linearisiert werden und die Lösung in folgender Form dargestellt werden:

$$\mathbf{q}(\mathbf{x}, t) = \mathbf{q}^0(\mathbf{x}^0) + \mathrm{Re}\left(\tilde{\mathbf{q}} e^{i\omega t}\right),\ \tilde{\mathbf{q}} \in \mathbb{C}^{5n_Z} \tag{4}$$

Die stationäre Lösung $\mathbf{q}^0(\mathbf{x}^0)$ wird in *TRACE* mit Hilfe einer aufwendigen Rechnung auf Basis einer Zeitdiskretisierung der Erhaltungsgleichungen berechnet.
Für den Vektor $\tilde{\mathbf{q}}$ ergibt sich ein lineares Gleichungssystem:

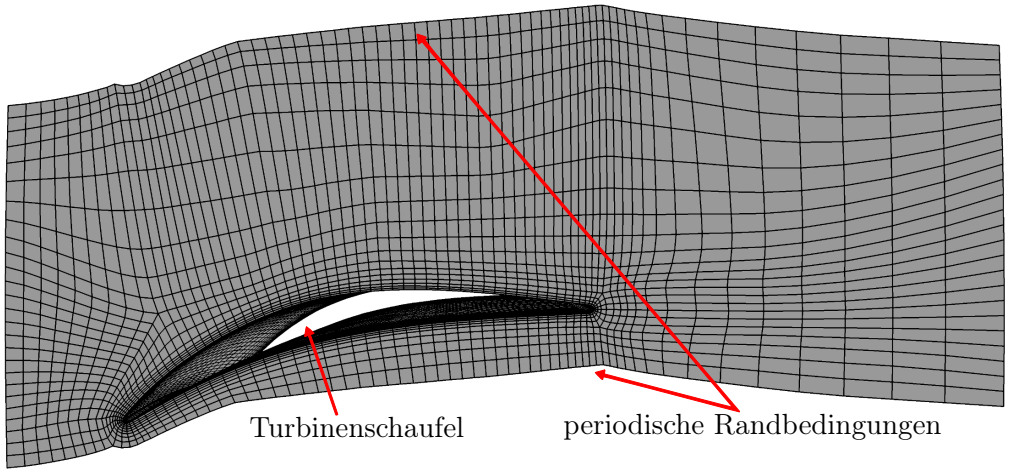
$$\mathbf{A}\tilde{\mathbf{q}} = \mathbf{b} \qquad \text{mit } \mathbf{A} \in \mathbb{C}^{n \times n},\ \tilde{\mathbf{q}}, \mathbf{b} \in \mathbb{C}^n \tag{5}$$

Das Gleichungssystem ist dabei abhängig von der stationären Lösung $\mathbf{q}^0$, den Koordinaten $\mathbf{x}^0$, der komplexen Amplitude der Störung $\tilde{\mathbf{x}}$ und der Kreisfrequenz ω.
Diese Linearisierung ermöglicht die Untersuchung von Auswirkungen von Störungen verschiedener Frequenzen und Phasenwinkel auf die Strömung.[7]

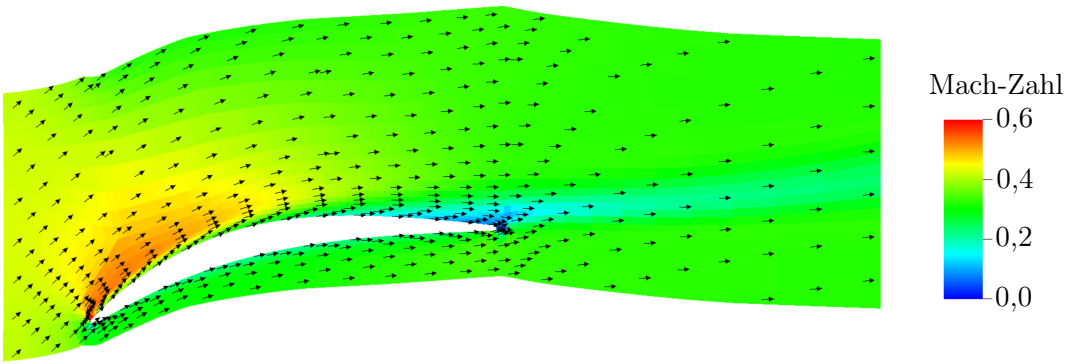
In dieser Arbeit werden zwei Gleichungssysteme aus *linearTRACE* untersucht. Abbildung 3 zeigt das Rechengebiet und die simulierte Strömung des THD-Testfalles, im UHBR-Testfall wird ein Ausschnitt der Strömung im UHBR-Fan (siehe Abbildung 1) simuliert. Die hier betrachteten linearen Gleichungssysteme können folgendermaßen charakterisiert werden:
In jeder Zelle werden fünf komplexwertige Variablen gespeichert. Jede Zelle ist nur von einigen umliegenden Zellen abhängig (meistens von 24 umliegenden Zellen nach Abbildung 2; für Zellen an den Rändern des Rechengebietes ist dies anders). Die Koeffizientenmatrix ist folglich dünnbesetzt, da in jeder Zeile nur ungefähr $24 \cdot 5$ Einträge ungleich Null sind. Ihre Einträge sind ebenfalls komplex. Weiterhin besteht die Matrix aus dichten 5×5-Blöcken und ist aufgrund der Upwind-Diskretisierung (MUSCL-Schema) der räumlichen Ableitungen unsymmetrisch. Die Belegungsstruktur der Matrix ist nahezu symmetrisch, lediglich die Struktur einiger Zeilen und Spalten weicht zur Vereinfachung der Implementierung von der Symmetrie ab.

2.2. Gradienten-basierte Optimierung in TAU[4, 3]

Das Softwaresystem TAU wird vom Institut für Aerodynamik und Strömungstechnik des DLR entwickelt. Es dient der Simulation von dreidimensionalen Strömungen um komplexe Geometrien, ein Beispiel ist die Außenströmung eines Flugzeuges.
Dazu werden die kompressiblen, instationären Reynolds-Gleichungen (RANS, siehe (1)



(a) Rechengebiet mit Finite-Volumen-Gitter



(b) Mach-Zahl und Strömungsrichtung der gemittelten Strömung in einer Schnittebene

Abbildung 3: Der THD-Testfall (Quelle der Daten: Institut für Antriebstechnik, DLR)

und (2), hier entfällt jedoch der Quellterm $S(q)$ bzw. S_iV_i) mittels Finiter-Volumen auf unstrukturierten Gittern diskretisiert. Die Darstellung der räumlichen Ableitungen zur Berechnung der Flüsse erfolgt über zentrale und Upwind-Schemata (zweiter Ordnung). Eine Besonderheit von TAU ist die Möglichkeit der automatischen Anpassung des Gitters an die Strömungsverhältnisse und die Verwendung von Mehrgitterverfahren.

In dieser Arbeit wird ein spezielles lineares Gleichungssystem betrachtet, dessen Lösung in einem Optimierungsverfahren benötigt wird; betrachte dazu folgendes Optimierungsproblem:
Sei $\alpha \in \Lambda \subset \mathbb{R}^{n_\alpha}$ ein Satz von Parametern, die die stationäre Lösung einer Strömungssimulation $\mathbf{q}(\alpha) \in \mathbb{R}^{5n_Z}$ beeinflussen. Gesucht werden nun Parameter $\alpha^* \in \Lambda$, die eine Zielfunktion $I(\mathbf{q}(\alpha), \alpha) \in \mathbb{R}$ minimieren.
Hier wird nun angenommen, dass die Funktionen $q(\cdot)$ auf Λ und $I(\cdot,\cdot)$ auf $\{(\mathbf{q}(\alpha), \alpha) : \alpha \in \Lambda\}$ differenzierbar sind. Da in einer kompressiblen Strömung Unstetigkeiten in Form von Überschall-Stößen auftreten können ((1) bildet hier ein System hyperbolischer Differentialgleichungen), ist diese Annahme nicht immer erfüllt. Gilt diese, können zur Bestimmung eines Minimums Verfahren eingesetzt werden, die den Gradienten (beziehungsweise die Jacobi-Matrix) von I bezüglich α an Punkten $\alpha \in \Lambda$ benötigen. Dieser Gradient kann mit Hilfe des folgenden Ansatzes berechnet werden:

$$\nabla_\alpha I = \frac{\partial I}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \alpha} + \frac{\partial I}{\partial \alpha} \tag{6}$$

Eine Alternative dazu ist die Darstellung über eine adjungierte Formulierung (siehe [3]), die hier nicht betrachtet wird.
Problematisch dabei ist die Bestimmung von $\frac{\partial \mathbf{q}}{\partial \alpha}$, da $\mathbf{q}(\alpha)$ das Ergebnis einer aufwendigen Strömungssimulation mit den Parametern α ist. Der Zusammenhang zwischen $\mathbf{q}$ und α kann jedoch über eine implizite Funktion dargestellt werden:

$$\mathbf{R} : \mathbb{R}^{5n_Z} \times \mathbb{R}^{n_\alpha} \to \mathbb{R}^{5n_Z} \quad \text{mit} \quad \forall \alpha \in \Lambda : \mathbf{R}(\mathbf{q}(\alpha), \alpha) = 0 \tag{7}$$

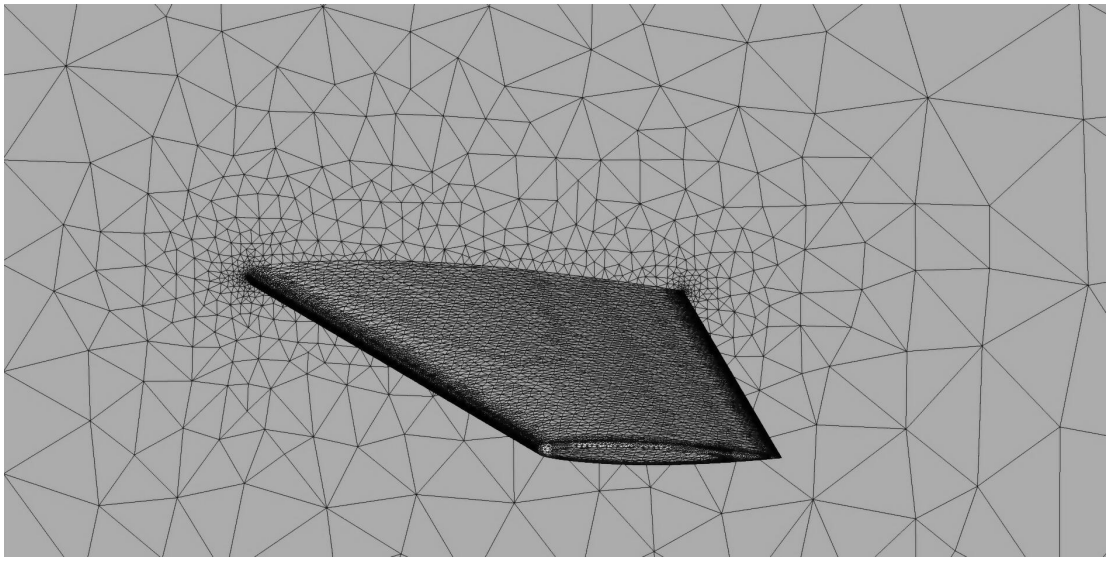
Die Funktion $\mathbf{R}$ erhält man aus der Summe der Flüsse über die Ränder jeder Zelle und den Randbedingungen der Finite-Volumen-Diskretisierung. Insbesondere gilt damit bei einem geeigneten Raum Λ und unter der Annahme, dass $\mathbf{R}(\cdot,\cdot)$ ebenfalls auf $\{(\mathbf{q}(\alpha), \alpha) : \alpha \in \Lambda\}$ differenzierbar ist:

$$\nabla_\alpha \mathbf{R} = \frac{\partial \mathbf{R}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \alpha} + \frac{\partial \mathbf{R}}{\partial \alpha} = 0 \tag{8}$$

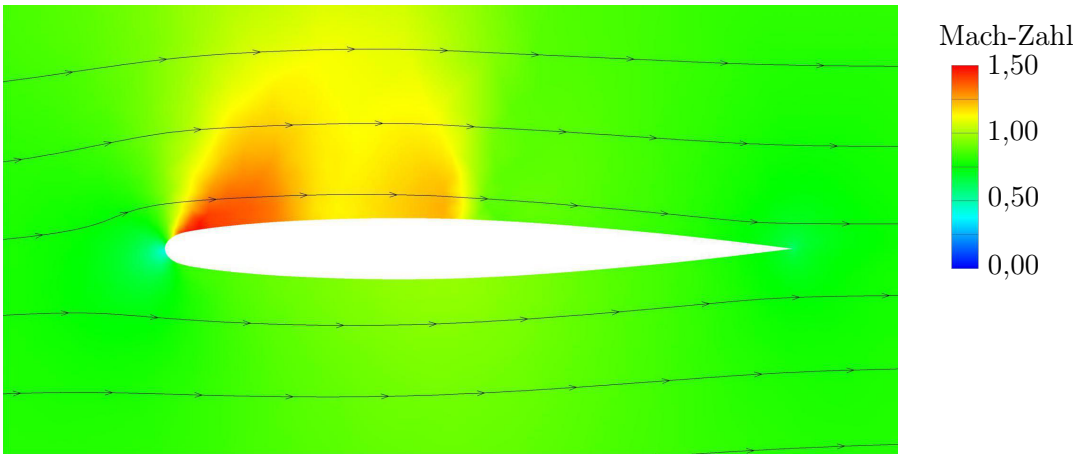
Die partiellen Ableitungen $\frac{\partial \mathbf{R}}{\partial \mathbf{q}}$ und $\frac{\partial \mathbf{R}}{\partial \alpha}$ können mit Hilfe der Darstellung von $\mathbf{R}$ bestimmt werden und lassen sich somit für gegebene α und $\mathbf{q}$ berechnen. Dadurch erhält man zur Bestimmung von $\frac{\partial \mathbf{q}}{\partial \alpha_i}$ ein lineares Gleichungssystem:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial \alpha_i} = -\frac{\partial \mathbf{R}}{\partial \alpha_i} \qquad \text{mit } \frac{\partial \mathbf{R}}{\partial \mathbf{q}} \in \mathbb{R}^{(5n_Z)\times(5n_Z)} \text{ und } \frac{\partial \mathbf{q}}{\partial \alpha_i}, -\frac{\partial \mathbf{R}}{\partial \alpha_i} \in \mathbb{R}^{5n_Z} \tag{9}$$

Zur Optimierung in TAU muss nun der Gradient $\nabla_\alpha I$ nacheinander für unterschiedliche



(a) Ausschnitt des Rechengebietes mit Finite-Volumen-Gitter



(b) Mach-Zahl und Strömungsrichtung der gemittelten Strömung in einer Schnittebene

Abbildung 4: Der TAU-Testfall Onera M6 Flügel (Quelle der Daten: Institut für Aerodynamik und Strömungstechnik, DLR)

Parameter α bestimmt werden. Dazu werden jeweils die Lösungen der linearen Gleichungssysteme für einen festen Satz von Parametern α benötigt.
Interessant ist, dass in der obigen Darstellung das Optimierungsproblem und die Gradienten für eine diskretisierte Strömung ($\mathbf{q}(\alpha) \in \mathbb{R}^{5nz}$) formuliert werden, man könnte analog aber auch eine Zielfunktion für eine kontinuierliche Strömung $q : \mathbb{R}^3 \times \mathbb{R}_+ \to \mathbb{R}^5$ definieren und den Gradienten $\nabla_\alpha I$ mit Hilfe der Erhaltungsgleichung aus (1) darstellen.

In dieser Arbeit wird beispielhaft das Gleichungssystem aus (9) für die Strömung um einen Flügel (siehe Abbildung 4) mit dem Anstellwinkel des Flügels als Parameter α_i betrachtet; zur Vereinfachung wird in diesem Fall hier die Viskosität vernachlässigt (Euler-Gleichungen). Das zu lösende lineare Gleichungssystem kann wie folgt charakterisiert werden:
In jeder Zelle werden fünf reelle Variablen gespeichert und jede Zelle ist nur von einigen umliegenden abhängig. Durch die Verwendung eines unstrukturierten Gitters ist die Zahl der umliegenden Zellen jedoch variabel. Das Gleichungssystem ist folglich reell, die Koeffizientenmatrix dünnbesetzt und die Zahl ihrer Einträge pro Zeile variiert von Zeile zu Zeile. Weiterhin besteht die Matrix aus dichten 5×5-Blöcken und ist aufgrund der Upwind-Diskretisierung der räumlichen Ableitungen nicht symmetrisch.

2.3. Typische Eigenschaften

In den vorangegangenen Abschnitten wurden Beispiele aktueller Anwendungen aus der Strömungstechnik beschrieben, die das Lösen großer, dünnbesetzer linearer Gleichungssysteme erfordern. Im Folgenden werden drei Testfälle aus diesen Anwendungen verwendet. Eine Übersicht dazu gibt Tabelle 1:

Testmatrix	TRACE-THD	TRACE-UHBR	TAU
Körper	$\mathbb{C}$	$\mathbb{C}$	$\mathbb{R}$
Dimension	378.400	4.497.520	541.980
Nichtnulleinträge (NNE)	45.456.500	619.672.700	170.610.950
Blockgröße	5×5	5×5	5×5
min. NNE pro Zeile	70	70	90
max. NNE pro Zeile	125	1270	605
NNE / Dimension	120	138	315

Tabelle 1: Übersicht über Testmatrizen

Die Nichtnullelemente der Matrix sind hier immer in Blöcken fester Größe (Blockgröße) angeordnet. Die hohen Werte der maximalen Anzahl an Nichtnullelementen pro Zeile treten bei TRACE bei Zellen an den Rändern des Rechengebietes aufgrund von speziellen Einström- oder Ausström-Randbedingungen auf.

Anhand dieser Testfälle können einige typische Eigenschaften von linearen Gleichungssystemen aus der Strömungstechnik dargestellt werden:
Die Dimension der Systeme, das heißt die Anzahl der Unbekannten, liegt aktuell in den hier betrachteten Anwendungen im Bereich einiger Millionen (Testfall TRACE-UHBR, die anderen beiden Systeme dienen eher Testzwecken). In den nächsten Jahren sind

auch Rechnungen mit deutlich mehr Unbekannten geplant. Ein Beispiel dafür ist der TRACE-UHBR-Testfall: Hier wird die Strömung um einen Ring von Turbinenschaufeln simuliert. Geplant ist aber auch die Simulation mehrerer hintereinander liegender Ebenen von Schaufeln.[4]
Die Nichtnulleinträge der Matrix sind aufgrund der Diskretisierung in kleinen quadratischen Blöcken fester Größe angeordnet, hier von der Größe 5×5. Insbesondere sind hier die Blöcke auf der Diagonalen der Matrix regulär.
Eine Besonderheit der in dieser Arbeit betrachteten Anwendungen ist der große Abhängigkeitsbereich der einzelnen Zellen der Diskretisierung, in TRACE sind das zwei Zellen je Richtung. Dadurch ist die Zahl der Nichtnulleinträge der Matrix pro Zeile höher als bei der Verwendung einer Diskretisierung, bei der die Werte aus einer Zelle nur mit den Werten aus direkten Nachbarzellen verknüpft werden.

[4]Quelle: Institut für Antriebstechnik, DLR

3. Iterative Verfahren zur Lösung linearer Gleichungssysteme

Das Problem, ein lineares Gleichungssystem zu lösen, lässt sich folgendermaßen formulieren:
Bestimme für eine gegebene, nicht-singuläre Matrix $A \in \mathbb{C}^{n\times n}$ der Dimension $n \in \mathbb{N}$ und eine rechte Seite $b \in \mathbb{C}^n$ einen Vektor $x \in \mathbb{C}^n$, für den gilt:

$$Ax = b \tag{10}$$

Nicht-singulär bedeutet, dass die Inverse $A^{-1} \in \mathbb{C}^{n\times n}$ der Matrix existiert, für die $A^{-1}A = I$ ist; dabei bezeichnet I hier immer die Einheitsmatrix $I = \mathrm{diag}(1, \ldots, 1)$ passender Größe.
Folglich ist die eindeutig Lösung bestimmt durch

$$x = A^{-1}b. \tag{11}$$

Es ist jedoch nur in seltenen Fällen numerisch sinnvoll, die Matrix A^{-1} explizit zu berechnen. Verfahren zur Lösung eines linearen Gleichungssystems können grob in zwei Kategorien eingeteilt werden:
Direkte Verfahren bestimmen eine Zerlegung der Matrix A, beispielsweise in untere und obere Dreiecksmatrizen $L, U \in \mathbb{C}^{n\times n}, LU = A$, die mit geringerem Aufwand invertiert werden kann (hier durch Vorwärts- und Rückwärtseinsetzen $x = U^{-1}y, y := L^{-1}b$). Die Zerlegung einer großen, dünnbesetzten Matrix ist meistens jedoch deutlich dichter als die Ursprungsmatrix, sodass der benötigte Speicher und der Rechenaufwand stark anwachsen.
Iterative Verfahren berechnen schrittweise Näherungen für die Lösung x (in Arithmetik endlicher Genauigkeit ist auch die Lösung eines direkten Verfahrens nur eine Näherung). Sie verwenden Vektor-Operationen und Matrix-Vektor-Multiplikationen (mit der Matrix A), dadurch müssen nur die Matrix A und einige Vektoren gespeichert werden.
Durch die *Vorkonditionierung* des Gleichungssystems ist es möglich, die Anzahl der benötigten Iterationen zur Bestimmung einer ausreichend genauen Schätzung von x und damit den Rechenaufwand zu reduzieren. Viele Techniken zur Vorkonditionierung basieren auf Ansätzen aus direkten Verfahren. Das Ergebnis aus einem direkten Verfahren kann wiederum durch *Nachiteration* verbessert werden, falls es aufgrund von Rundungsfehlern zu ungenau ist.

3.1. Lineare iterative Verfahren[13]

Die Idee bei iterativen Verfahren besteht zunächst darin, eine invertierbare Matrix $K \in \mathbb{C}^{n\times n}$ zu finden, die die Matrix A approximiert und dessen Inverse einfach darzustellen ist. In dem Sinne, dass $\tilde{x} \in \mathbb{C}^n$ mit

$$\tilde{x} := K^{-1}b \quad \text{für } K \approx A \tag{12}$$

mit geringem Aufwand berechnet werden kann. Der Vektor $\tilde{x}$ ist eine Schätzung für x. Der Fehler kann mit Hilfe des Residuums $r \in \mathbb{C}^n$ gemessen werden:

$$r := b - A\tilde{x} \tag{13}$$

Zur Verbesserung der Schätzung $\tilde{x}$ kann man die benötigte Korrektur $\delta x \in \mathbb{C}^n$ mit $x = \tilde{x} + \delta x$ betrachten:

$$\begin{aligned} & A(\tilde{x} + \delta x) = b \\ \Leftrightarrow \quad & A\delta x = b - A\tilde{x} \\ \Leftrightarrow \quad & A\delta x = r \end{aligned}$$

Die Korrektor δx kann erneut durch $K^{-1}r$ approximiert werden. Damit erhält man eine verbesserte Schätzung $\tilde{x}' \in \mathbb{C}^n$ mit:

$$\tilde{x}' := \tilde{x} + K^{-1}r \tag{14}$$

Wenn man nun $\tilde{x} := \tilde{x}'$ setzt und diesen Korrekturschritt wiederholt, erhält man ein lineares iteratives Verfahren. Dies kann so oft durchgeführt werden, bis die Schätzung von x ausreichend genau ist, gemessen beispielsweise an der Norm des Residuums: Das heißt, die Iteration wird abgebrochen, sobald für ein vorgegebenes $\epsilon \in \mathbb{R}_+$ folgende Bedingung erfüllt ist:

$$\|r\|_2 \leq \epsilon \tag{15}$$

Damit erhält man:

Algorithmus 1. Lineares iteratives Verfahren

1: Wähle x_0
2: $r_0 \leftarrow b - Ax_0$
3: $k \leftarrow 0$
4: **while** $\|r_k\| > \epsilon$ **do**
5: $\quad x_{k+1} \leftarrow x_k + K^{-1}r_k$
6: $\quad r_{k+1} \leftarrow b - Ax_{k+1}$
7: $\quad k \leftarrow k + 1$
8: **end while**

Hier bezeichnet $x_k \in \mathbb{C}^n$ die Schätzung für x aus dem Iterationsschritt $k \in \mathbb{N}_0$ und $r_k \in \mathbb{C}^n$ das zugehörige Residuum.

3.1.1. Matrix-Splitting: Jacobi-, Gauß-Seidel- und SSOR-Verfahren[10]

Ein Ansatz zur Bestimmung der Matrix K ist das Matrix-Splitting: Dazu zerteilt man die Matrix A beispielsweise in eine Diagonalmatrix D, eine obere Dreiecksmatrix E und eine untere Dreiecksmatrix F mit $D, E, F \in \mathbb{C}^{n \times n}$

$$A = D - E - F \tag{16}$$

Beim *Jacobi*-Verfahren wählt man $K = D$. Dadurch ergibt sich unter der Annahme $\det(D) \neq 0$ die Rechenvorschrift:

$$\begin{aligned} & x_{k+1} = x_k + D^{-1}(b - Ax_k) \\ \Leftrightarrow \quad & Dx_{k+1} = Dx_k + b - (D - E - F)x_k \\ \Leftrightarrow \quad & Dx_{k+1} = (E + F)x_k + b \end{aligned} \tag{17}$$

Die Einträge von x_{k+1} können direkt durch Division mit dem jeweiligen Diagonalelement aus der letzten Gleichung ermittelt werden. Hier ist auch eine Zerlegung von A in eine Block-Diagonalmatrix D_b,

$$D_b = \begin{pmatrix} D_1 & & & \\ & D_2 & & \\ & & \ddots & \\ & & & D_k \end{pmatrix}, \quad \text{mit } k \in \mathbb{N},\ D_i \in \mathbb{C}^{n_i \times n_i},\ n_i \in \mathbb{N}, \sum_i n_i = n, \tag{18}$$

und eine untere und eine obere Dreiecksmatrix möglich (hier wird $\det(D_b) \neq 0$ angenommen). Dann müssen in jedem Iterationsschritt Gleichungssysteme mit den (kleinen) Diagonalblöcken gelöst werden (*Block-Jacobi*-Verfahren).

Eine andere Wahl von K führt zu dem *Gauß-Seidel*-Verfahren: Bei der Variante *Forward Gauß-Seidel* wählt man $K = D - F$ (unter der Annahme $\det(D - F) \neq 0$) und erhält hier:

$$\begin{aligned} & x_{k+1} = x_k + (D - F)^{-1}(b - Ax_k) \\ \Leftrightarrow \quad & (D - F)x_{k+1} = (D - F)x_k + b - (D - E - F)x_k \\ \Leftrightarrow \quad & (D - F)x_{k+1} = Ex_k + b \end{aligned} \tag{19}$$

Mit der letzten Gleichung lassen sich die einzelnen Einträge von x_{k+1} durch Vorwärtseinsetzen direkt berechnen.
Bei der Variante *Backward Gauß-Seidel* wird hingegen $K = D - E$ gesetzt (unter der Annahme $\det(D - E) \neq 0$). Es ergibt sich analog:

$$(D - E)x_{k+1} = Fx_k + b \tag{20}$$

Hier kann x_{k+1} durch Rückwärtseinsetzen direkt berechnet werden.

Die Konvergenz des Gauß-Seidel-Verfahrens kann in vielen Fällen durch die Einführung eines Relaxationsparameters $\omega \in [1, 2)$ mit $K = D - \omega F$ beziehungsweise $K = D - \omega E$ verbessert werden. Hier wird angenommen, dass $\det(D - \omega F) \neq 0$ und $\det(D - \omega E) \neq 0$ gilt. Verwendet man nun abwechselnd diese beiden Varianten, erhält man das *SSOR*-Verfahren (*symmetric successive over-relaxation*) mit dem Zwischenschritt $x_{k+1/2} \in \mathbb{C}^n$:

$$\begin{aligned} (D - \omega F)x_{k+1/2} &= (\omega E + (1 - \omega)D)x_k + \omega b \\ (D - \omega E)x_{k+1} &= (\omega F + (1 - \omega)D)x_{k+1/2} + \omega b \end{aligned} \tag{21}$$

Auch dieses Verfahren lässt sich nach obigem Schema als lineares iteratives Verfahren beschreiben, nur erhält man hier einen längeren Ausdruck für K.

3.1.2. Konvergenzanalyse[2]

Ein lineares iteratives Verfahren lässt sich mit Hilfe der Iterationsvorschrift

$$x_{k+1} = x_k + K^{-1} r_k \tag{22}$$
$$= x_k + K^{-1}(b - Ax_k) \tag{23}$$

beschreiben. Die Matrix K kann auch als Vorkonditionerung des Gleichungssystems aufgefasst werden:

$$K^{-1}Ax = K^{-1}b \tag{24}$$
$$\Leftrightarrow \quad \tilde{A}x = \tilde{b} \tag{25}$$

Dabei bezeichnet $\tilde{A} = K^{-1}A \in \mathbb{C}^{n\times n}$ die vorkonditionierte Matrix und $\tilde{b} = K^{-1}b \in \mathbb{C}^n$ die transformierte rechte Seite.
Als Iterationsvorschrift für das vorkonditionierte System erhält man nun mit dem vorkonditionierten Residuum $\tilde{r}_k = K^{-1}r_k \in \mathbb{C}^n$:

$$x_{k+1} = x_k + \tilde{r}_k \tag{26}$$
$$\Leftrightarrow \quad \tilde{A}x_{k+1} = \tilde{A}x_k + \tilde{A}\tilde{r}_k$$
$$\Leftrightarrow \quad \tilde{b} - \tilde{A}x_{k+1} = \tilde{b} - \tilde{A}x_k - \tilde{A}\tilde{r}_k$$
$$\Leftrightarrow \quad \tilde{r}_{k+1} = \tilde{r}_k - \tilde{A}\tilde{r}_k$$
$$\Leftrightarrow \quad \tilde{r}_{k+1} = (I - \tilde{A})\tilde{r}_k \tag{27}$$

Daraus folgt für den Fehler $e_k := x_k - x$ ebenfalls:

$$\Leftrightarrow \quad (\tilde{b} - \tilde{A}x_{k+1}) = (I - \tilde{A})(\tilde{b} - \tilde{A}x_k)$$
$$\Leftrightarrow \quad \tilde{A}(x - x_{k+1}) = (I - \tilde{A})\tilde{A}(x - x_k)$$
$$\Rightarrow \quad e_{k+1} = (I - \tilde{A})e_k \tag{28}$$

Damit erhält man:

$$e_k = (I - \tilde{A})^k e_0 \tag{29}$$

Das Verfahren konvergiert folglich, falls $\lim_{k\to\infty}(I - \tilde{A})^k \to 0$ gilt. Das ist genau dann der Fall, wenn der Spektralradius $\rho(I-\tilde{A})$ kleiner 1 ist (Betrag des betragsmäßig größten Eigenwertes von $(I - \tilde{A})$).[2]
Die Reduktion des Fehlers in jedem Iterationsschritt gemessen in einer geeigneten Vektornorm $\|\cdot\|$ für $\mathbb{C}^n$ und der durch diese induzierten Matrixnorm auf $\mathbb{C}^{n\times n}$ (für diese gilt insbesondere immer $\|I - \tilde{A}\| \geq \rho(I - \tilde{A})$) kann abgeschätzt werden durch:

$$\|e_{k+1}\| \leq \|I - \tilde{A}\|\|e_k\| \tag{30}$$

Das bedeutet jedoch, dass für eine gegebene Matrix A eine Vorkonditionierung K^{-1} bestimmt werden muss, die die Bedingung $\rho(I - \tilde{K}^{-1}A) < 1$ erfüllt, um die Konvergenz garantieren zu können. Dadurch sind spezielle lineare Verfahren nur für bestimmte Klassen von Matrizen geeignet, beispielsweise diagonaldominante Matrizen. Weiterhin kann auch für diese Matrizen der Spektralradius nahe bei 1 liegen, wodurch das Verfahren möglicherweise nur langsam konvergiert.

3.2. Krylov-Unterraum-Verfahren[13]

Im Folgenden bezeichnet $Ax = b$ das vorkonditionierte System, sodass auf die Schreibweise mit $\tilde{\cdot}$ verzichtet werden kann.
Nach (27) hat das Residuum bei einem linearen iterativen Verfahren die Form

$$r_k = (I - A)^k r_0. \tag{31}$$

Mit der Iterationsvorschrift aus (26) erhält man:

$$\begin{aligned} & x_{k+1} = x_k + r_k \\ \Leftrightarrow \quad & x_{k+1} = x_k + (I - A)^k r_0 \\ \Leftrightarrow \quad & x_{k+1} = x_0 + \sum_{l=0}^{k} (I - A)^l r_0 \end{aligned} \tag{32}$$

Der Startvektor x_0 kann ohne Einschränkung der Allgemeinheit auf Null gesetzt werden. Denn einen von Null verschiedenen Startvektor kann man durch eine Verschiebung der rechten Seite darstellen (hier bezeichnen die Vektoren mit $\tilde{\cdot}$ die Größen des Gleichungssystems mit einem Startvektor $\tilde{x}_0 \neq 0$):

$$\begin{aligned} & A\tilde{x} = \tilde{b} \\ \Leftrightarrow \quad & A\underbrace{(\tilde{x} - \tilde{x}_0)} = \underbrace{\tilde{b} - A\tilde{x}_0} \\ \Leftrightarrow \quad & A \quad x \quad = \quad b \end{aligned} \tag{33}$$

Für das verschobene Gleichungssystem erhält man $x_0 = 0$ und $r_0 = b$.
Somit bedeutet (32), dass der Vektor x_{k+1} in einem speziellen Unterraum des $\mathbb{C}^n$ liegt:

$$x_{k+1} \in \operatorname{span}\left\{r_0, Ar_0, \ldots, A^k r_0\right\} \tag{34}$$

Hat dieser Raum die Dimension $(k + 1)$, wird er als der $(k + 1)$-*dimensionale Krylov-Unterraum* von A und r_0 bezeichnet:

$$\mathcal{K}^{k+1}(A, r_0) := \operatorname{span}\left\{r_0, Ar_0, \ldots, A^k r_0\right\} \tag{35}$$

Die Dimension des Raumes $\operatorname{span}\left\{r_0, Ar_0, \ldots, A^k r_0\right\}$ kann auch kleiner als $k + 1$ sein. Das heißt jedoch, dass die Lösung x (in exakter Arithmetik) bereits in diesem Raum liegt:
Sind nämlich die Vektoren r_0, Ar_0, …, $A^k r_0$ linear abhängig, kann r_0 als gewichtete Summe der Vektoren Ar_0, …, $A^k r_0$ dargestellt werden:

$$r_0 = \sum_{i=1}^{k} \alpha_i A^i r_0 \qquad \text{mit } \alpha_i \in \mathbb{C}$$

Daraus folgt für die Lösung x wegen $Ax = r_0$:

$$x = A^{-1}r_0 = A^{-1}\sum_{i=1}^{k}\alpha_i A^i r_0 = \sum_{i=1}^{k}\alpha_i A^{i-1} r_0$$

$$\Rightarrow \quad x \in \mathcal{K}^k(A, r_0)$$

Weiterhin erhält man für ein beliebiges $x_k \in \mathcal{K}^k(A, r_0)$ für das Residuum:

$$r_k = r_0 - Ax_k$$

$$\Rightarrow \quad r_k \in \mathcal{K}^{k+1}(A, r_0) \tag{36}$$

Daraus folgt, dass alle iterativen Verfahren, bei denen sich x_{k+1} durch eine Linearkombination der vorherigen Schätzungen $x_0, \ldots, x_k$ und der Residuen $r_0, \ldots, r_k$ darstellen lässt, Elemente aus Krylov-Unterräumen steigender Dimension produzieren. Dabei ist es oft hilfreich, die Vektoren aus dem (k+1)-dimensionalen Krylov-Unterraum mit Polynomen k-ten Grades der Matrix A darzustellen:
Für $Q_k(A) := \sum_{i=0}^{k}\alpha_i A^i$ mit $\alpha_i \in \mathbb{C}$ gilt:

$$x_{k+1} \in \mathcal{K}^{k+1}(A, r_0) \quad \Leftrightarrow \quad x_{k+1} = Q_k(A)r_0 \tag{37}$$

Analog lässt sich das Residuum mit einem Polynom (k+1)-ten Grades $P_{k+1}(A)$ darstellen:

$$r_{k+1} = P_{k+1}(A)r_0 := b - AQ_k(A)r_0 = (I - AQ_k(A))r_0 \tag{38}$$

Hier erhält man zusätzlich die Eigenschaft $P_{k+1}(0) = 1$. Die Matrixpolynome werden in keinem der hier vorgestellten Verfahren direkt ausgewertet, sondern immer Produkte der Polynome mit einem Vektor bestimmt.

Die Idee bei *Krylov-Unterraum-Verfahren* liegt nun darin, aus dem Raum $\mathcal{K}^k(A, r_0)$ einen Vektor x_k zu bestimmen, der die Lösung x möglichst gut approximiert. Dazu gibt es verschiedene Ansätze:[13]

- *Ritz-Galerkin*: Wähle x_k so, dass das Residuum orthogonal zu dem Krylov-Unterraum $\mathcal{K}^k(A, r_0)$ ist, also:

$$x_k \in \mathcal{K}^k(A, r_0) : (b - Ax_k) \perp \mathcal{K}^k(A, r_0) \tag{39}$$

- *Petrov-Galerkin*: Wähle x_k so, dass das Residuum orthogonal zu einem geeigneten anderen k-dimensionalen Unterraum $\widetilde{\mathcal{K}^k}$ ist, also

$$x_k \in \mathcal{K}^k(A, r_0) : (b - Ax_k) \perp \widetilde{\mathcal{K}^k} \tag{40}$$

  mit $\widetilde{\mathcal{K}^k} \subset \mathbb{C}^{n\times n}$ und $\dim\left(\widetilde{\mathcal{K}^k}\right) = k$.

- *Minimale Norm des Residuums*: Bestimme x_k mit dem minimalen Residuum r_k, also

$$x_k \in \mathcal{K}^k(A, r_0) : \|b - Ax_k\|_2 = \min_{y \in \mathcal{K}^k(A,r_0)} \|b - Ay\|_2 \tag{41}$$

Gemeinsam ist diesen Ansätzen, dass sie Bedingungen für das Residuum $r_k := b - Ax_k$ angeben, daher wird die Schätzung x_k in den meisten Fallen nicht für die Iteration benötigt, sondern nur aus dem Update des Residuums der Form $r_{k+1} = r_k + Ap_k, p_k \in \mathbb{C}^n$ mit $x_{k+1} = x_k + p_k$ bestimmt.
Wesentlich bei der Rechnung in Arithmetik endlicher Genauigkeit ist es, eine numerisch stabile Basis für die Krylov-Unterräume zu finden:
Die Vektoren $\{r_0, Ar_0, \ldots, A^k r_0\}$ sind ungeeignet, da der Anteil in Richtung des Eigenvektors des größten Eigenwertes der Matrix A immer größer wird.
Ideal ist eine orthonormale Basis; diese kann beispielsweise dadurch erzeugt werden, dass nach jeder Matrixmultiplikation der resultierende Vektor zu den vorherigen Basisvektoren orthogonalisiert wird (*Arnoldi*-Prozess).
Definiert man zu einem Satz von Basisvektoren $v_1, \ldots, v_k$ des Raumes $\mathcal{K}^k(A, r_0)$ die Matrix $V_k \in \mathbb{C}^{n \times k}, V_k = (v_1, \ldots, v_k)$, kann das Gleichungssystem mit dieser auf den Krylov-Unterraum $\mathcal{K}^k(A, r_0)$ projiziert werden:

$$AV_k = V_{k+1} H_{k+1,k} \tag{42}$$

Dabei erhält man eine obere Hessenbergmatrix $H_{k+1,k} \in \mathbb{C}^{(k+1)\times k}$.
Die obigen Ansätze basieren auf verschiedenen Vorgehensweisen, eine geeignete Basis $v_1, \ldots, v_k$ zu bestimmen und mit dieser das Gleichungssystem auf einen Krylov-Unterraum zu projizieren.
Im folgenden werden das CG-Verfahren, das BICGSTAB-Verfahren und das GMRES-Verfahren vorgestellt, die auf den Ansätzen von *Ritz-Galerkin* und *Petrov-Galerkin* und dem Ansatz der *minimalen Norm des Residuums* aufbauen. Eine systematische Darstellung von Krylov-Unterraum-Verfahren findet man in [13]. Algorithmen zur iterativen Lösung linearer Gleichungssysteme und Techniken zur Vorkonditionierung werden ausführlich in [10] beschrieben.

3.2.1. Das CG-Verfahren

Für Gleichungssysteme mit einer hermitesch positiv definiten Matrix A (symmetrisch positiv definit für reelle Matrizen) ist das CG-Verfahren am besten geeignet.[13] Die in dieser Arbeit betrachteten Anwendungsbeispiele führen zu unsymmetrischen Gleichungssystemen, der hermitesch positiv definite Fall ist jedoch hilfreich für das Verständnis anderer Krylov-Unterraum-Verfahren.
Ist die Matrix A hermitesch positiv definit, lässt sich folgendes Skalarprodukt definieren:

$$< x, y >_A := x^H Ay \tag{43}$$

Dies ermöglicht eine wichtige geometrische Interpretation des *Ritz-Galerkin*-Ansatzes:

$$\begin{aligned} & (b - Ax_k) \perp \mathcal{K}^k(A, r_0) \\ \Leftrightarrow \quad & A(x - x_k) \perp \mathcal{K}^k(A, r_0) \\ \Leftrightarrow \quad & (x - x_k) \perp_A \mathcal{K}^k(A, r_0) \end{aligned}$$

Das bedeutet, dass der Fehler $x - x_k$ orthogonal zu dem Raum $K^k(A, r_0)$ in Bezug auf das obige Skalarprodukt $< \cdot, \cdot >_A$ ist. Folglich ist der Fehler in der Norm, die durch dieses Skalarprodukt induziert wird, minimal:

$$\Leftrightarrow \quad \|x - x_k\|_A = \min_{y \in \mathcal{K}^k(A, r_0)} \|x - y\|_A \tag{44}$$

Das zeigt insbesondere, dass der Fehler $\|x - x_k\|_A$ mit jeder Iteration kleiner wird, da sich der Suchraum vergrößert ($\mathcal{K}^{k-1}(A, r_0) \subset \mathcal{K}^k(A, r_0)$) und die Matrix A vollen Rang hat.

Definiert man eine orthonormale Basis $v_1, \ldots, v_k$, erhält man wegen der Orthogonalität $b - Ax \perp \mathcal{K}^k(A, r_0)$:

$$\begin{aligned} & V_k^H(b - Ax_k) = 0 \\ \Leftrightarrow \quad & V_k^H A x_k = V_k^H b \end{aligned}$$

Stellt man nun x in der Basis $v_1, \ldots, v_k$ als $x = V_k y, y \in \mathbb{C}^k$ dar und setzt $b = r_0 = \|r_0\|_2 v_1$ ein, ergibt sich die orthogonale Projektion des Systems auf den Unterraum $\mathcal{K}^k(A, r_0)$:

$$\begin{aligned} \Rightarrow \quad & V_k^H A V_k y = V_k^H \|r_0\|_2 v_1 \\ \Leftrightarrow \quad & (V_k^H A V_k) y = \|r_0\|_2 e_1 \end{aligned} \tag{45}$$

Der Vektor $e_1 \in \mathbb{C}^k$ bezeichnet hier den kanonischen Einheitsvektor. Die Matrix des projizierten Systems $H_{k,k} \in \mathbb{C}^{k \times k}$ erhält man nach (42) und

$$H_{k,k} := V_k^H A V_k = V_k^H V_{k+1} H_{k+1,k}$$

durch Weglassen der letzten Zeile aus $H_{k+1,k}$. Wegen der Symmetrie von A ist $H_{k,k}$ ebenfalls symmetrisch und damit tridiagonal. Dadurch lassen sich die Einträge von $H_{k,k}$, die Vektoren r_k und Hilfsvektoren $p_k \in \mathcal{K}^k(A, r_0)$ zur Ermittlung von x_k rekursiv aus $H_{k-1,k-1}$, r_{k-1} und p_{k-1} ermitteln. Das resultierende Verfahren ist das CG-Verfahren (*conjugate gradients*).
Mit der Konditionszahl der Matrix $\kappa(A) = |\lambda_{max}|/|\lambda_{min}|$ (Quotient des betragsmäßig größten und kleinsten Eigenwertes der Matrix) kann man folgende Fehlerschranke herleiten (siehe beispielsweise [13, 2]):

$$\|x_k - x\|_A \leq 2 \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^k \|x_0 - x\|_A \tag{46}$$

Das CG-Verfahren weist damit folgende Merkmale auf:

- geeignet für hermitesch positiv definite Matrizen
- konstanter Aufwand pro Iterationsschritt (eine Matrix-Vektor-Multiplikation, mehrere Vektor-Operationen) durch einen rekursiven Ansatz – erlaubt die Suche der Lösung in Unterräumen hoher Dimension
- garantierte Reduktion des Fehlers in jedem Iterationsschritt (in exakter Arithmetik), die Konvergenzrate hängt von $\sqrt{\kappa(A)}$ ab

3.2.2. Das BICGSTAB-Verfahren

Wenn man das CG-Verfahren auf unsymmetrische (bzw. nicht hermitesche) Matrizen übertragen möchte, ergeben sich zwei Probleme:
Zum einen lässt sich mit der Matrix A im Allgemeinen kein Skalarprodukt wie in (43) definieren. Dadurch ist nicht sichergestellt, dass der Fehler $x-x_k$ in einem Iterationsschritt nicht steigen kann. Zum anderen erhält man keine Projektion auf ein Tridiagonalsystem durch die Bedingung, dass das Residuum senkrecht zu dem Krylov-Unterrraum $\mathcal{K}^k(A, r_0)$ sei, sodass sich keine rekursive Vorschrift zur Berechnung von r_k und x_k ergibt.
Alternativ kann man jedoch x_k so bestimmen, dass das Residuum senkrecht zu einem anderen, geeigneten Raum ist (*Petrov-Galerkin*-Ansatz): Ist nun $w_1, \ldots, w_k$ eine Basis dieses Raumes und die Matrix $W_k \in \mathbb{C}^{n\times k}$ definiert durch $W_k := (w_1, \ldots, w_k)$ erhält man dadurch die Gleichung:

$$W_k^H(b - Ax_k) = 0 \tag{47}$$

Sind nun die Basisvektoren $v_1, \ldots, v_k$ und $w_1, \ldots, w_k$ bi-orthogonal zueinander, das heißt

$$< w_i, v_j > = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{48}$$

erhält man aus (42) mit einer oberen Hessenbergmatrix $H_{k,k} \in \mathbb{C}^{k\times k}$:

$$W_k^H A V_k = W_k^H V_{k+1} H_{k+1,k} = H_{k,k} \tag{49}$$

Wenn $H_{k,k}$ tridiagonal sein soll, gilt dies ebenfalls für das adjungierte System $V_k^H A^H W_k$. Folglich können die Basisvektoren $w_1, \ldots, w_k$ mit A^H und einem Vektor $\hat{r}_0 \in \mathbb{C}^n$ gebildet werden, das Residuum r_k wird dann so bestimmt, dass es zu dem Raum $\mathcal{K}^k(A^H, \hat{r}_0)$ senkrecht ist. Das entsprechend adjungierte Gleichungssystem kann mit $A^H\hat{x} = \hat{r}_0$ und $\hat{r}_k := \hat{r}_0 - A^H\hat{x}_k$ für $\hat{x}, \hat{r}_k, \hat{x}_k \in \mathbb{C}^n$ bezeichnet werden (für $\hat{r}_0$ kann man beispielsweise r_0 einsetzen).
Ein Paar bi-orthogonaler Basisvektoren muss jedoch nicht immer existieren; ein Verfahren mit diesem Ansatz schlägt fehl, falls in einem Schritt $w_k^H v_k = 0$ gilt (*break down*). In Gleitkommarithmetik ist auch der Fall $w_k^H v_k \approx 0$ kritisch.

Mit diesen Überlegungen ergibt sich das Bi-CG-Verfahren analog zu dem CG-Verfahren aus (47) und $b = r_0 = \|r_0\|v_1$ und $x_k = V_k y_k, y \in \mathbb{C}^k$:

$$\begin{aligned} & W_k^H A x_k = W_k^H b \\ \Rightarrow \quad & W_k^H A V_k y_k = W_k^H \|r_0\| v_1 \\ \Leftrightarrow \quad & W_k^H A V_k y_k = \|r_0\| w_1^H v_1 e_1 \\ \Leftrightarrow \quad & H_{k,k} y_k = (\|r_0\| d_1) e_1 \end{aligned} \tag{50}$$

Man erhält folglich wie im CG-Verfahren ein Gleichungssystem mit einer tridiagonalen Matrix. Dadurch können wie zuvor die Einträge von $H_{k,k}$ die Vektoren x_k, r_k, $\hat{r}_k$ und Hilfsvektoren $p_k \in \mathcal{K}^k(A, r_0)$, $\hat{p}_k \in \mathcal{K}^k(A^H, \hat{r}_0)$ rekursiv aus $H_{k-1,k-1}$ und den Vektoren r_{k-1}, $\hat{r}_{k-1}$, p_{k-1} und $\hat{p}_{k-1}$ berechnet werden. Man benötigt jedoch in jedem Iterationsschritt eine Multiplikation mit der Matrix A und eine Multiplikation mit der Matrix A^H.

Man kann die obigen Vektoren auch analog zu (37) und (38) mit Matrixpolynomen $P_k(A)$, $T_k(A)$ darstellen als

$$\begin{aligned} r_k &= P_k(A) r_0, \\ \hat{r}_k &= P_k(A^H) \hat{r}_0, \\ p_k &= T_{k-1}(A) r_0, \\ \hat{p}_k &= T_{k-1}(A^H) \hat{r}_0. \end{aligned} \tag{51}$$

Aus der Rekursion zur Bestimmung der Vektoren erhält man für die Matrixpolynome P_k und T_k folgende Rekursion:

$$\begin{aligned} P_k(A) &= P_{k-1}(A) - \alpha_k A T_{k-1}(A) && \text{mit } \alpha_k \in \mathbb{C}, \\ T_k(A) &= P_k(A) + \beta_k T_{k-1}(A) && \text{mit } \beta_k \in \mathbb{C} \end{aligned} \tag{52}$$

und $P_0(A) = T_0(A) = 1$.[13]
Im Bi-CG-Verfahren werden zur Bestimmung von r_k, p_k und x_k die Vektoren $\hat{r}_k$ und $\hat{p}_k$ nicht direkt benötigt, sondern nur für Polynome $Q_i(A)$ und $Q_j(A^H)$ Skalarprodukte der Form

$$< Q_i(A) r_0, Q_j(A^H) \hat{r}_0 > . \tag{53}$$

Diese sind äquivalent zu:

$$< Q_j(A) Q_i(A) r_0, \hat{r}_0 > \tag{54}$$

Aus diesem Grund können alle benötigten Werte rekonstruiert werden, ohne dass Multiplikaitonen mit der Matrix A^H benötigt werden und ohne dass die Vektoren $\hat{r}_k$ und $\hat{p}_k$ explizit aufgestellt werden.

Insbesondere lassen sich auch folgende andere Residuuen $r_k^{CGS} = P_k(A) P_k(A) r_0$ (mit dem Matrixpolynom des Bi-CG-Verfahren von oben) und entsprechende Schätzungen

der Lösung $x_k^{CGS} : r_k^{CGS} = b - Ax_k^{CGS}$ mit dem gleichen Aufwand wie beim Bi-CG-Verfahren berechnen. Dies wird als das CGS-Verfahren bezeichnet. Durch den quadratischen Faktor $P_k^2(A)$ benötigt dieses Verfahren in manchen Fällen nur halb so viele Iterationen wie das Bi-CG-Verfahren. Das Residuum des Bi-CG-Verfahrens $r_k = P_k(A)r_0$ kann jedoch auch größer als das Anfangsresiduum r_0 sein, in diesen Fällen vergrößert sich der Fehler beim CGS-Verfahren gemessen am Residuum ebenfalls quadratisch. Die unregelmäßige Konvergenz (d.h. in einem Schritt gilt beispielsweise $\|r_k\|_2 \gg \|r_0\|_2$) führt bei einer Rechnung in endlicher Genauigkeit zu einem Fehler der Lösung x_k.

Das Bi-CG- und das CGS-Verfahren basieren darauf, dass die Vektoren (die den Raum $\mathcal{K}^{k+1}(A, r_0)$ aufspannen) $r_i = P_i(A)r_0$, $i = 1, \ldots, k$ bi-orthogonal zu den Basisvektoren $\hat{r}_j = P_j(A^H)r_0$, $j = 1, \ldots, k$ des Raumes $\mathcal{K}^{k+1}(A^H, \hat{r}_0)$ sind. Man kann jedoch auch andere Basisvektoren $\hat{r}'_j = Q_j(A^H)\hat{r}_0, j = 1, \ldots, k$ des Raumes $\mathcal{K}^{k+1}(A^H, \hat{r}_0)$ auswählen, die die Bi-Orthogonalitätsbedingung

$$< P_i(A)r_0, Q_j(A^H)\hat{r}_0 > = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{55}$$

erfüllen. Damit kann man analog zu oben beim CGS-Verfahren auch die Residuen $r'_k = Q_k(A)P_k(A)r_0$ und entsprechende Schätzungen der Lösung $x'_k : r'_k = b - Ax'_k$ bestimmen. Eine mögliche Wahl für das Polynom $Q_k(A)$, mit der sich eine einfache Rekursion zur Berechnung von r'_k ergibt, ist:

$$Q_k(A) := \prod_{i=1}^{k} (I - \omega_i A) \qquad \text{mit } \omega_i \in \mathbb{C} \tag{56}$$

Wird der Wert von ω_k in jedem Schritt so gewählt, dass das Residuum r'_k minimal wird (ähnlich einer GMRES(1)-Iteration, s.u.), erhält man das BICGSTAB-Verfahren.
Dieses weist aufgrund der Minimierung von r'_k bezüglich ω_k eine regelmäßigere Konvergenz als das CGS-Verfahren auf und ermöglicht dadurch eine numerisch stabilere Implementierung als letzteres.

Die Merkmale des BICGSTAB-Verfahren sind also:

- geeignet für beliebige reelle und komplexe (invertierbare) Matrizen
- konstanter Aufwand pro Iterationsschritt (zwei Matrix-Vektor-Multiplikationen, mehrere Vektor-Operationen) durch einen rekursiven Ansatz – erlaubt die Suche der Lösung in Unterräumen hoher Dimension
- möglicher Abbruch des Algorithmus (*break-down*), bevor eine ausreichend genaue Lösung ermittelt wird
- unregelmäßige Konvergenz, die Norm des Residuums kann im Laufe der Rechnung steigen und auch größer als die Norm des Residuums des Startwertes werden – gemessen an der Norm des Residuums keine Garantie in einem Iterationsschritt einen genaueren Schätzwert für die Lösung zu erhalten

Einen Abbruch des BICGSTAB-Verfahren kann man mittels *Look-Ahead*-Strategien vermeiden. Mehrere Basisvektoren der Krylov-Unterräume $K^k(A, r_0)$ und $K^k(A^H, r_0)$ werden dazu blockweise bi-orthogonalisiert.[13]
Durch eine andere Wahl des Matrixpolynoms $Q_k(A)$ lassen sich weitere Verfahren konstruieren. Man kann beispielsweise auch das Residuum über zwei aufeinander folgende Bi-CG-Iterationen minimieren (eine Kombination von Bi-CG und GMRES(2)) und erhält damit das BICGSTAB(2)-Verfahren. Dies ist auch allgemein mit l aufeinander folgenden Iterationen möglich (BICGSTAB(l)-Verfahren). Der Aufwand jedes Iterationsschrittes dieses Verfahren ist größer als bei dem einfachen BICGSTAB-Verfahren, dafür verbessert sich eventuell (deutlich) die Konvergenzgeschwindigkeit. Desweiteren kann dadurch in einigen Fällen ein Abbruch des Verfahrens vermieden werden (für den Fall $\omega_k \approx 0$ im BICGSTAB-Verfahren).[13]

3.2.3. Das (F)GMRES-Verfahren

Ein anderer Ansatz für unsymmetrische (bzw. nicht hermitesche) Matrizen besteht darin, auf eine rekursive Darstellung des Residuums r_k zu verzichten und dafür in jedem Iterationsschritt dessen Norm zu minimieren (*Minimale Norm des Residuums*).
Dazu betrachtet man die Norm des Residuums $\|b - Ax_k\|$ und die Projektion des Gleichungssystems auf den Krylov-Unterraum: Der Vektor $x_k \in \mathcal{K}^k(A, r_0)$ lässt sich mit Hilfe der Matrix der orthogonalen Basisvektoren V_k darstellen durch $x_k = V_k y_k, y_k \in \mathbb{C}^k$. Analog ist die rechte Seite $b = r_0 = \|r_0\|_2 v_1 = \|r_0\|_2 V_{k+1} e_1$, mit dem ersten kanonischen Einheitsvektor passender Dimension $e_1 \in \mathbb{C}^{k+1}$. Durch (42) ($AV_k = V_{k+1} H_{k+1,k}$) erhält man nun:

$$\|b - Ax_k\|_2 = \left\|b - AV_k y_k\right\|_2 = \left\|\|r_0\|_2 V_{k+1} e_1 - V_{k+1} H_{k+1,k} y_k\right\|_2$$

Da die Norm $\|\cdot\|_2$ eines Vektors bezüglich der Multiplikation mit einer unitären Matrix invariant ist, erhält man mit $V_n \in \mathbb{C}^{n \times n}$, V_n unitär und $V_n = (v_1, \ldots, v_k, \ldots)$:

$$\left\|V_n^H V_{k+1} \left(\|r_0\|_2 e_1 - H_{k+1,k} y_k\right)\right\|_2 = \left\|\|r_0\|_2 e_1 - H_{k+1,k} y_k\right\|_2$$

Es gilt also:

$$\|b - Ax_k\|_2 = \left\|\|r_0\|_2 e_1 - H_{k+1,k} y_k\right\|_2, \quad \text{mit } x_k = V_k y_k \tag{57}$$

Das bedeutet, die Minimierung der Norm des Residuums über den Krylov-Unterraum $\mathcal{K}^k(A, r_0)$ kann auch aufgefasst werden als das folgende lineare Ausgleichsproblem mit der oberen Hessenbergmatrix $H_{k+1,k} \in \mathbb{C}^{(k+1) \times k}$:
Bestimme $x_k = V_k y_k$ mit

$$\left\|\|r_0\|_2 e_1 - H_{k+1,k} y_k\right\|_2 = \min_{z \in \mathbb{C}^k} \left\|\|r_0\|_2 e_1 - H_{k+1,k} z\right\|_2. \tag{58}$$

Dieses lineare Ausgleichsproblem kann mit Hilfe einer QR-Zerlegung gelöst werden:

$$H_{k+1,k} = Q_{k+1,k} R_k \tag{59}$$

Hier bezeichnet $Q_{k+1,k} \in \mathbb{C}^{(k+1)\times k}$ eine Matrix mit orthonormalen Spaltenvektoren und $R_k \in \mathbb{C}^{k\times k}$ eine obere Dreiecksmatrix. Damit erhält man

$$y_k = R_k^{-1} Q_{k+1,k}^H (\|r_0\|_2 e_1) \tag{60}$$

und kann auch x_k bestimmen. Zur Iteration werden x_k und y_k jedoch nicht benötigt.
Die Matrix $Q_{k+1,k}$ kann implizit durch einzelne Givens-Rotationen dargestellt werden und die obere Dreiecksmatrix R_k wird aus R_{k-1}, v_k und den Givens-Rotationen berechnet. Da die obere Hessenbergmatrix $H_{k+1,k}$ und damit auch die obere Dreicksmatrix R_k hier dicht besetzt sind, können die benötigten Vektoren v_k nicht rekursiv bestimmt werden. Daher muss zur Berechnung von v_k in jedem Iterationsschritt der Vektor Av_{k-1} bezüglich der Vektoren $v_1, \ldots, v_{k-1}$ orthogonalisiert werden (*Arnoldi-Prozess*). Insgesamt ergibt sich das GMRES-Verfahren.
Da der Aufwand mit jeder Iteration steigt (ausschlaggebend ist die Orthogonalisierung mit $v_1, \ldots, v_{k-1}$), wird das Verfahren meistens nach einer vorgegebenen Anzahl m von Iterationen neu gestartet (GMRES(m)). Das Problem dabei ist, dass durch einen kleinen Wert von m eventuell die Möglichkeit einer beschleunigten Konvergenz verschenkt wird; es kann aber ebenso sein, dass der Rechenaufwand unverhältnismäßig steigt, falls das Verfahren erst später neu gestartet wird.

Eine Zusammenfassung der Eigenschaften des GMRES-Verfahren ergibt:

- geeignet für beliebige reelle und komplexe (invertierbare) Matrizen
- steigender Aufwand mit jedem Iterationsschritt (eine Matrix-Vektor-Multiplikation, eine steigende Zahl an Vektor-Operationen, insbesondere Skalarprodukten), daher wird das Verfahren meistens nach einer vorgegebenen Anzahl an Schritten neu gestartet – die Lösung kann in der Regel nur in Unterräumen niedriger Dimension effizient gesucht werden
- robust, die Norm des Residuums kann (in exakter Arithmetik) nicht steigen (da ja $\|b - Ax_k\|_2$ minimiert wird); es ist aber auch eine sehr langsame Konvergenz möglich – in vielen praktischen Fällen ist die Konvergenz jedoch superlinear

Theoretisch kann für GMRES(m) mit $m < n$ die Norm des Residuums auch stagnieren, sodass gar keine Konvergenz eintritt, in der Praxis ist das aber unwahrscheinlich.[13]

Bezüglich der Vorkonditionierung erhält man hier einen wichtigen Aspekt: Betrachtet man ein explizit vorkonditioniertes Gleichungssystem der Form

$$AK^{-1}z = b \qquad \text{mit } x = K^{-1}z,\ z \in \mathbb{C}^n, \tag{61}$$

kann man dieses mit (42) auf den Krylov-Unterraum projizieren. Dadurch gilt mit der Matrix $W_k \in \mathbb{C}^{n\times k}$, $W_k = (w_1, \ldots, w_k)$:

$$AW_k = \tilde{V}_{k+1}\tilde{H}_{k+1,k} \qquad \text{mit } w_i = K^{-1}\tilde{v}_i,\ i = 1, \ldots, k \tag{62}$$

Hier bezeichnen $\tilde{V}_{k+1} \in \mathbb{C}^{n\times(k+1)}$, $\tilde{V}_{k+1} = (\tilde{v}_1, \ldots, \tilde{v}_{k+1})$ eine Matrix mit orthonormalen Spaltenvektoren und $\tilde{H}_{k+1,k} \in \mathbb{C}^{(k+1)\times k}$ eine obere Hessenbergmatrix.

Stellt man nun den vorkonditionierten Lösungsvektor mittels $z_k = V_k y_k$ dar und betrachtet die Norm des vorkonditionierten Residuums

$$\left\|b - AK^{-1}z_k\right\|_2 = \left\|b - AK^{-1}V_k y_k\right\|_2 = \|b - AW_k y_k\|_2 \tag{63}$$

erhält man mit der obigen Projektion wieder:

$$\Rightarrow \qquad \left\|b - AK^{-1}z_k\right\|_2 = \left\|\tilde{V}_{k+1}\|r_0\|_2 e_1 - \tilde{V}_{k+1}\tilde{H}_{k+1,k}y_k\right\|_2 \tag{64}$$

Der Vektor y_k, der diesen Term minimiert, kann wie zuvor mit Hilfe einer QR-Zerlegung der oberen Hessenbergmatrix $\tilde{H}_{k+1,k}$ ermittelt werden.
Zu dem zuvor vorgestellten (nicht vorkonditionierten) GMRES-Verfahren ergeben sich dann zwei Unterschiede:
Zum Einen wird hier die Norm des vorkonditionierten Residuums über den Krylov-Unterraum minimiert, das Ergebnis minimiert daher im Allgemeinen nicht den Term $\|b - Ax_k\|_2$.
Zum Anderen kann man den gesuchten Lösungsvektor x_k direkt aus $x_k = W_k y_k$ ($= K^{-1}V_k y_k$) ermitteln, falls die Spaltenvektoren w_i, $i = 1, \ldots, k$ der Matrix W_k gespeichert werden. Der Vorkonditionierer K^{-1} fließt in diesem Fall nur in die Berechnung der Vektoren w_i aus v_i ein. Insbesondere ist es nicht notwendig, dass in jedem Schritt die selbe Matrix K^{-1} verwendet wird. Das obige Minimierungsproblem kann (in vielen Fällen) auch für eine Matrix W_k mit Spalten $w_i = K_i^{-1}v_i$ aufgestellt werden.
Auf Kosten der zusätzlichen Speicherung der Vektoren $w_1, \ldots, w_k$ ist es also möglich, in jedem Schritt einen anderen Vorkonditionierer $K_k^{-1} \in \mathbb{C}^{n\times n}$ zu wählen – dieser kann beispielsweise auch aus einem iterativen Verfahren bestehen. Dies wird als das FGMRES-Verfahren bezeichnet.
Problematisch dabei ist jedoch, dass die Projektion auf den Unterraum (mit der Matrix W_k) für unterschiedliche Vorkonditionierer K_k^{-1} nicht orthogonal ist, dadurch kann die Matrix $\tilde{H}_{k+1,k}$ singulär werden (oder beinahe singulär, im Falle nicht exakter Arithmetik ebenso schwerwiegend). Das muss folglich bei der Verwendung verschiedener Vorkonditionierer in jedem Iterationsschritt beachtet werden. Ein Ansatz das zu vermeiden, besteht darin, diese möglichst ähnlich zu wählen.
Ein anderes Krylov-Unterraum-Verfahren, das die variable Vorkonditionierung ohne diese Schwierigkeiten ermöglicht, ist das GMRESR- beziehungsweise GMRES$\star$-Verfahren, das in [13] beschrieben wird.

4. Paralleler Aspekt

Die Lösung der Gleichungssysteme, die hier betrachtet werden, kann mit den heutigen Computersystemen nur auf parallelen Rechnern bestimmt werden.
Der Grund dafür liegt unter anderem in der Menge der Daten, die für die Rechnung benötigt werden. Zur Veranschaulichung kann man die benötigte Speichermenge für die Einträge der Matrix abschätzen: Jede komplexe Zahl benötigt in doppelter Genauigkeit $2 \cdot 64/8 = 16$ Byte Speicher. Die Matrix aus dem Testfall TRACE-UHBR hat ungefähr 500 Millionen Nichtnulleinträge, das entspricht 8Gb ($8 \cdot 1024^3$ Byte). Weiterhin müssen hier mehrere Vektoren aus jeweils ungefähr 5 Millionen komplexen Zahlen gespeichert werden und zur Vorkonditionierung ist bei einigen Verfahren mindestens ebenso viel Speicher erforderlich wie für die Koeffizientenmatrix.
Heutige Computersysteme, die effizient mit diesen Datenmengen rechnen, führen viele Prozesse parallel aus. Diese sind bei einem Cluster-System beispielsweise hierarchisch strukturiert: ein Node besteht aus mehreren identischen Prozessorkernen, die jeweils einen Prozess ausführen, und verschiedene Nodes sind über ein Netzwerk verbunden. Dabei hat jeder Node seinen eigenen Arbeitsspeicher, die Daten müssen hier folglich verteilt gespeichert werden.
Zur Vereinfachung kann man annehmen, dass jeder Prozess von einem Prozessorkern ausgeführt wird, der Zugriff auf seine lokalen Daten hat. Um auf Daten anderer Prozesse zuzugreifen, wird Kommunikation zwischen den beteiligten Prozessen benötigt. Diese Kommunikation führt zu zusätzlichem Rechenaufwand und zu Wartezeiten, falls ein Prozess Daten benötigt, die auf einem anderen erst berechnet werden müssen.
Damit lassen sich Anforderungen an einen parallelen Algorithmus formulieren:

- möglichst wenig Kommunikation
- Wartezeiten vermeiden (Load-Balancing)
- Skalierung auf unterschiedlich viele Prozesse möglichst unabhängig von den Eingangsdaten

Die ersten beiden Punkte sind notwendig, um die Effizienz der Rechnung im Parallelen zu ermöglichen. Load-Balancing bedeutet, die Daten und den Rechenaufwand gleichmäßig auf die Prozesse zu verteilen. Der dritte Punkt stellt sicher, dass der Algorithmus auf unterschiedlichen Systemen und mit unterschiedlich vielen Prozessen verwendet werden kann.

Iterative Verfahren zur Lösung linearer Gleichungssysteme sind aufgebaut aus Multiplikationen der Koeffizientenmatrix mit einem Vektor und aus verschiedenen Vektor-Operationen, wie dem Skalarprodukt, der Addition zweier Vektoren und der Multiplikation eines Vektors mit einem Skalar. Diese Operationen müssen daher nach den obigen Anforderungen implementiert werden. Für die Vektor-Operationen ist das unproblematisch, Möglichkeiten der parallelen Berechnung des Matrix-Vektor-Produkts werden im folgenden Abschnitt dargestellt.

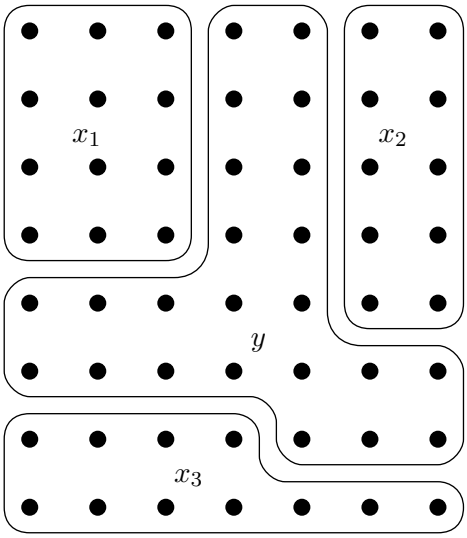
4.1. Methoden zur verteilten Matrix-Vektor-Multiplikation

Der Grundgedanke besteht darin, die Zeilen und Spalten der Matrix A zu permutieren und die transformierte Matrix in Blöcke zu zerlegen, die auf die einzelnen Prozesse verteilt werden. Analog werden die zur Rechnung benötigten Vektoren (der Vektor, mit dem multipliziert wird, und das Ergebnis) auf die Prozesse verteilt.
Die Kommunikation ist dann notwendig, wenn auf einem Prozess Elemente eines Vektors benötigt werden, die auf einem anderen gespeichert sind oder wenn Zwischenergebnisse (falls auf einem Prozess nur ein Teil einer Matrixzeile liegt) ausgetauscht werden müssen.
Da die Matrix dünnbesetzt ist, können Zerlegungen der permutierten Matrix bestimmt werden, für die der Kommunikationsaufwand bei der Matrix-Vektor-Multiplikation möglichst klein ist. Ideal wäre eine Zerlegung in Diagonalblöcke, bei der außerhalb dieser Diagonalblöcke nur Nulleinträge sind. Insbesondere ließe sich in diesem Fall das lineare Gleichungssystem in voneinander unabhängige Teile zerlegen.
In den hier betrachteten Fällen ist das jedoch im Allgemeinen nicht möglich: Da das Gleichungssystem aus einer Diskretisierung auf einem dreidimensionalen Gitter stammt, stellt jede Zeile der Matrix A eine Gleichung aus einer Zelle des Gitters dar, die von den Variablen aus einigen umliegenden Zellen abhängt. Eine Zerlegung der Matrix in Blöcke von Zeilen entspricht daher einer Gebietszerlegung des Gitters (zur Veranschaulichung siehe Abbildung 5). Die Abhängigkeiten zwischen den Gebieten ergibt sich aus den Gleichungen aus den Zellen ihrer Ränder.

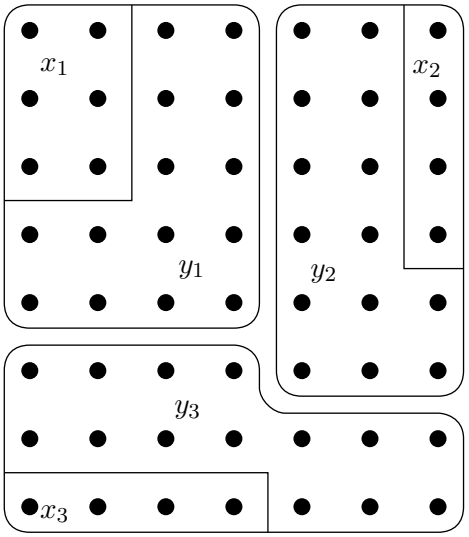
Es gibt hier also grundsätzlich zwei mögliche Ansätze zur Bestimmung einer geeigneten Verteilung der Zeilen des Gleichungssystems: Diese kann auf Basis einer Zerlegung des Diskretisierungsgebietes ermittelt werden oder mit Hilfe eines Algorithmus zur Graph-Partitionierung (die Zeilen der Matrix entsprechen dabei Knoten des Graphen, und Nichtnulleinträge Kanten zwischen diesen Knoten).
Der erste Ansatz ist nur dann durchführbar ist, wenn das Gleichungssystem tatsächlich aus einer Diskretisierung auf einem Gitter stammt und die notwendigen Informationen aus der Diskretisierung bekannt sind. Der zweite Ansatz ist hingegen für beliebige dünnbesetzten linearen Gleichungssysteme geeignet.
Aus diesem Grund wird in dieser Arbeit nur der Ansatz der Graph-Partitionierung verfolgt, die Darstellung mittels Gebietszerlegungen dient hier nur der Veranschaulichung. Im folgenden werden zwei Mögliche Darstellungen zur Verteilung des Gleichungssystems beschrieben.

4.1.1. Variante A

Durch Zeilen- und Spaltenpermutationen lässt sich also die Matrix A beispielsweise in folgender Form mit den quadratischen Blöcken $D_1, \ldots, D_m$ und G und den Blöcken



(a) Variante A



(b) Variante B

Abbildung 5: 2d-Gitter-Zerlegungen: x_i bezeichnen jeweils Vektoren von Unbekannten aus Gitterpunkten, die nicht direkt voneinander abhängig sind (bei einem Abhängigkeitsbereich von zwei Gitterpunkten in jede Richtung); über die Vektoren y bzw. y_i werden die Gebiete gekoppelt.

$E_1, \ldots, E_m$ und $F_1, \ldots, E_m$ darstellen:

$$P_z A P_s^{-1} = \begin{pmatrix} D_1 & & & & E_1 \\ & D_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & D_m & E_m \\ F_1 & F_2 & \cdots & F_m & G \end{pmatrix} \tag{65}$$

Dabei sind die Matrizen P_z und P_s^{-1} die dazu notwendigen Zeilen- beziehungsweise Spaltenpermutationen.
Die Permutationen kann man auch auf das gesamte Gleichungssystem anwenden:

$$Ax = b \qquad \Leftrightarrow \qquad \left(P_z A P_s^{-1}\right)(P_s x) = (P_z b) \tag{66}$$

Dadurch beschränkt sich der zusätzliche Aufwand für ein iteratives Verfahren durch die Zeilen- und Spaltenpermutationen darauf, vor Beginn der Iteration den Vektor $P_z b$ und die verteilte Matrix $P_z A P_s^{-1}$ zu bestimmen und am Ende die Lösung aus dem permutierten Vektor $P_z x$ zu berechnen. Die Anwendung einer Permutationsmatrix auf einen Vektor der Dimension n ist mit höchstens n Vertauschungen möglich und damit vom Rechenaufwand gegenüber anderen Operationen oft vernachlässigbar.
Das gesamte Gleichungssystem hat damit folgende Form:

$$\begin{pmatrix} D_1 & & & & E_1 \\ & D_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & D_m & E_m \\ F_1 & F_2 & \cdots & F_m & G \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \\ y \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \\ g \end{pmatrix} \tag{67}$$

Dabei stellen die Vektoren $x_1, \ldots, x_m$ und y die Unbekannten dar und die Vektoren $f_1, \ldots, f_m$ und g die rechte Seite (vergleiche Abbildung 5a).
Von diesem Gleichungssystem können nun beispielsweise jeweils Zeilen aus (67) auf einem Prozess gespeichert werden, das heißt die Blöcke D_1 und E_1 der Matrix und die Vektoren x_1 und f_1 liegen auf dem ersten Prozess, die Blöcke D_2 und E_2 und die Vektoren x_2 und f_2 auf dem zweiten und so weiter, der letzte Prozess erhält die Matrixblöcke $F_1, \ldots, F_k$ und G und die Vektoren y und g. Dadurch wird durch entsprechende Zeilen- und Spaltenpermutationen zwar das gesamte Kommunikationsvolumen minimiert, problematisch ist jedoch das Load-Balancing auf unterschiedlich vielen Prozessen für eine gegebene Matrix A (alle Kommunikation läuft über den letzten Prozess und der unterste Teil des Systems wird für größere Werte von k ebenfalls größer). Weiterhin unterscheidet sich die Struktur der Daten auf dem letzten Prozess von den anderen Prozessen, das erhöht die Komplexität einer effizienten Implementierung.

4.1.2. Variante B

Um die Probleme der obigen Blockstruktur zu beheben, kann man die Matrixelemente auch so umsortieren, dass sich die Blöcke aus der letzten Zeile und Spalte aus (65)

wiederum aufteilen lassen:

$$P_z A P_s^{-1} = \left(\begin{array}{cccc|cccc} D_1 & & & & E_1 & & & \\ & D_2 & & & & E_2 & & \\ & & \ddots & & & & \ddots & \\ & & & D_m & & & & E_m \\ \hline F_1 & & & & G_{1,1} & G_{1,2} & \cdots & G_{1,m} \\ & F_2 & & & G_{2,1} & G_{2,2} & \cdots & G_{2,m} \\ & & \ddots & & \vdots & \vdots & \ddots & \vdots \\ & & & F_m & G_{m,1} & G_{m,2} & \cdots & G_{m,m} \end{array}\right) \tag{68}$$

Dabei bezeichnen P_z und P_s^{-1} die notwendigen Permutationsmatrizen und D_i, E_i, F_i, $G_{i,j}$ für $i, j = 1, \ldots, m$ Blöcke der transformierten Matrix. Das gesamte Gleichungssystem wird dazu in die folgende Form gebracht:

$$\left(\begin{array}{cccc|cccc} D_1 & & & & E_1 & & & \\ & D_2 & & & & E_2 & & \\ & & \ddots & & & & \ddots & \\ & & & D_m & & & & E_m \\ \hline F_1 & & & & G_{1,1} & G_{1,2} & \cdots & G_{1,m} \\ & F_2 & & & G_{2,1} & G_{2,2} & \cdots & G_{2,m} \\ & & \ddots & & \vdots & \vdots & \ddots & \vdots \\ & & & F_m & G_{m,1} & G_{m,2} & \cdots & G_{m,m} \end{array}\right) \left(\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_m \\ y_1 \\ y_2 \\ \vdots \\ y_m \end{array}\right) = \left(\begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_m \\ g_1 \\ g_2 \\ \vdots \\ g_m \end{array}\right) \tag{69}$$

Jedem Prozess i können nun aus diesem Gleichungssystem jeweils eine Zeile aus dem oberen Teil von (69) und die entsprechende aus dem unteren Teil zugewiesen werden, das sind die Matrixblöcke D_i und E_i und F_i und $G_{i,j}, j = 1, \ldots, m$ und die Vektoren x_i und f_i und y_i und g_i (vergleiche Abbildung 5b).
Bei dieser Zerlegung ist die Struktur der Daten auf allen Prozessen ähnlich, dadurch ermöglicht sie eine gleichmäßige Lastverteilung für eine variable Anzahl Prozesse m. Für die Berechnung des Matrix-Vektor-Produkts wird hier nur für die Multiplikation der Blöcke $G_{i,j}, i \neq j$ mit den entsprechenden Elementen des Vektors Kommunikation benötigt. Die meisten dieser Blöcke können in den hier betrachteten Anwendungsfällen für größere Werte von m Null sein; werden nämlich den Prozessen zusammenhängende Gebiete des Diskretisierungsgitters zugewiesen, sind Gleichungen auf einem Prozess nur von Variablen aus einigen angrenzenden Gebieten abhängig.

Das generelle Problem dabei, dass der Kommunikationsaufwand für eine größere Anzahl Prozessoren m ansteigt, bleibt auch auf diese Weise bestehen.
Das kann durch die Zerlegung des Diskretisierungsgitters an einem Beispiel veranschaulicht werden: Ein Würfel mit einem strukturierten Gitter mit 100 Zellen in Richtung jeder Koordinatenachse besteht aus insgesamt 1 Millionen Zellen, das entspricht der Größenordnung des UHBR-Testfalls. Weitherhin ist in linearTRACE jede Zelle in jeder Koordinatenrichtung von zwei weiteren Zellen abhänging. Zerlegt man nun das Gitter

des Würfels in zwei gleich große Teile, entstehen an der Schnittfläche $4 \cdot 100^2$ Zellen, die von Werten aus dem jeweils anderen Teil abhängen. Das entspricht einem Anteil von 4% aller Zellen. Für jeden weiteren Schnitt kommen ungefähr wieder 4% hinzu. Damit steigt der Anteil der Zellen an Schnittflächen auf ungefähr 12% bei einer Zerlegung in 8 Teilgebiete und grob 20% bei 16 Teilgebieten.

4.1.3. Allgemeiner Algorithmus

Das Problem der verteilten Berechnung des Matrix-Vektor-Produktes $b \leftarrow Ax$ mit einer der obigen Zerlegungen lässt sich allgemein wie folgt darstellen:
Die Vektoren x und b werden aufgeteilt in m Teilvektoren $x = (x_1, x_2, \ldots, x_m)^T$ und $b = (b_1, b_2, \ldots, b_m)^T$ mit $x_i, b_i \in \mathbb{C}^{n_i}$ und $n_i \in \mathbb{N}$. Analog wird die Matrix A in die Submatrizen $A_i \in \mathbb{C}^{n_i \times n_i}$ und $A_{i,j} \in \mathbb{C}^{n_i \times n_j}$ (die abhängig von der Blockstruktur der Matrix Null oder dünnbesetzt sind) zerlegt:

$$\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} \leftarrow \begin{pmatrix} A_1 & A_{1,2} & \cdots & A_{1,m} \\ A_{2,1} & A_2 & \cdots & A_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m,1} & A_{m,2} & \cdots & A_m \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \tag{70}$$

Jeder Prozess i speichert nun eine Zeile des obigen Systems. Dadurch ergibt sich folgender paralleler Algorithmus:

Algorithmus 2. Verteiltes Matrix-Vektor-Produkt

1: Berechne $b_i \leftarrow A_i x_i$
2: Tausche benötigte Teile des Vektors $(x_1, x_2, \ldots, x_m)^T$ aus
3: **for** $j = 1 \rightarrow m, j \neq i$ **do**
4: Berechne $b_i \leftarrow b_i + A_{i,j} x_j$
5: **end for**

Zur effizienten Implentierung können hier Rechnung und Kommunikation auch überlagert werden.

5. Vorkonditionierung

Die Konvergenz eines iterativen Verfahrens hängt wesentlich von der Vorkonditionierung des linearen Gleichungssystems ab.
Die Idee der Vorkonditionierung besteht darin, einen zu A ähnlichen und mit wenig Aufwand invertierbaren Operator K zu bestimmen, so dass das System $K^{-1}Ax = K^{-1}b$ leichter zu lösen ist – in dem Sinne, dass ein Krylov-Unterraum-Verfahren weniger Iterationen benötigt, um eine ausreichend genaue Approximation der Lösung zu bestimmen. Für $K = A$, und damit das vorkonditionierte System $Ix = K^{-1}b$, erhält man die Lösung nach einem Schritt eines Krylov-Unterraum-Verfahrens. Das Problem $K^{-1}b$ zu bestimmen führt jedoch wieder auf das ursprüngliche Gleichungssystem. Das Problem, einen geeigneten Vorkonditionierer zu bestimmen, lässt sich folgendermaßen formulieren:[13]
Bestimme für ein gegebenes lineares Gleichungssystem $Ax = b$ einen Operator $K^{-1} : \mathbb{C}^n \to \mathbb{C}^n$ mit den Eigenschaften:

- Der Operator K ähnelt der Matrix A (in gewissem Sinne), beziehungsweise $K^{-1}A$ ist näher an I als A (beispielsweise in dem Sinne, dass die Eigenwerte der vorkonditionierten Matrix $K^{-1}A$ näher an 1 liegen).
- Der Aufwand für die Berechnung von $K^{-1}p$ für einen Vektor p ist möglichst klein, zumindest geringer als der für die Berechnung von $A^{-1}p$.
- Der Aufwand zur Konstruktion des Operators K^{-1} ist nicht zu groß.

Das Ziel der Vorkonditionierung besteht darin, den gesamten Rechenaufwand zu reduzieren, also die Summe des Aufwands zur Konstruktion des Vorkonditionierers und des Aufwands des vorkonditionierten iterativen Verfahrens.
In den meisten Fällen ist es schwierig, theoretische Aussagen zu den Auswirkungen eines bestimmten Vorkonditionierers für ein gegebenes lineares Gleichungssystem zu machen. Es kann auch vorkommen, dass sich das vorkonditionierte Gleichungssystem deutlich schlechter lösen lässt als das ursprüngliche System, obwohl die Vorkonditionierung mit einem ähnlichen Operator K^{-1} sehr gute Ergebnisse erzielt hat (siehe dazu das Kapitel zur Vorkonditionierung in [13]).

Ein lineares Gleichungssystem kann auf verschiedene Arten mit einem gegebenen Operator K^{-1} vorkonditioniert werden:[13]

Links-Vorkonditionierung: Das iterative Verfahren wird auf das Gleichungssystem

$$(K^{-1}A)x = K^{-1}b \tag{71}$$

angewendet. Das entspricht einer Vorkonditionierung des Residuums $K^{-1}(b - Ax_k)$.
Beim GMRES-Verfahren wird dann die Norm des vorkonditionierten Residuums minimiert, die im Allgemeinen nicht äquivalent zur Norm des Residuums ist.

Rechts-Vorkonditionierung: Das iterative Verfahren wird auf das Gleichungssystem

$$(AK^{-1})z = b \tag{72}$$

angewendet und aus dem Ergebnis für $z \in \mathbb{C}^n$ die Lösung $x = K^{-1}z$ ermittelt.

Beidseitige Vorkonditionierung: Das iterative Verfahren wird auf das Gleichungssystem

$$(K_L^{-1}AK_R^{-1})z = K_L^{-1}b \qquad \text{mit } K^{-1} = (K_L K_R)^{-1} \tag{73}$$

angewendet und aus dem Ergebnis für $z \in \mathbb{C}^n$ die Lösung $x = K_R^{-1}z$ ermittelt. Dies ist eine Kombination der beiden vorherigen Ansätze. Der Vorkonditionierer K^{-1} muss dazu jedoch in faktorisierter Form vorliegen.

Die Verteilung der Eigenwerte der vorkonditionierten Matrix ist (zumindest in exakter Arithmetik) für alle Varianten gleich, das gilt jedoch nicht für die Anteile des vorkonditionierten Residuums in Richtung der Eigenvektoren spezifischer Eigenwerte. Deshalb ändert sich das Konvergenzverhalten eines Krylov-Unterraum-Verfahrens für unterschiedliche Arten der Vorkonditionierung mit demselben Operator K^{-1}, beim GMRES-Verfahren ändert sich auch das zugrunde liegende Minimierungsproblem.

5.1. Klassen von Vorkonditionierungen

In den folgenden Abschnitten werden einige verschiedene Ansätze zur Vorkonditionierung vorgestellt, die in [13] oder [10] ausführlicher dargestellt sind.
Hier werden insbesondere Methoden betrachtet, die für große unsymmetrische Gleichungssysteme geeignet sind (und nicht spezielle Informationen aus der zugrunde liegenden Diskretisierung verwenden), wie die Vorkonditionierung mit Hilfe eines *linearen iterativen Verfahrens* oder einer *unvollständigen Zerlegung* der Koeffizientenmatrix.
Ein wichtiger Punkt dabei ist der parallele Aspekt, also die Frage, wie die Vorkonditionierung mit verteilten Daten effizient parallel berechnet werden kann.
Dies berücksichtigen Ansätze zur parallelen Vorkonditionierung; hier vorgestellt werden Möglichkeiten zur Vorkonditionierung mittels *Schwarz*-Zerlegung der Koeffizientenmatrix und mittels *Schur-Komplement*-Transformation des linearen Gleichungssystems.
Zur Darstellung der Zusammenhänge zwischen verschiedenen Verfahren wird zusätzlich die Idee *algebraischer Mehrgitter*-Verfahren (AMG) skizziert.
Weitere interessante, für die Anwendungsfälle geeignete Methoden, wie beispielsweise die Bestimmung einer *dünnbesetzten approximierten Inversen* (*sparse approximate inverse*), können im Rahmen dieser Arbeit nicht behandelt werden.

5.1.1. Lineare iterative Verfahren

Zur Vorkonditionierung eines Krylov-Unterraum-Verfahrens wird nur das Ergebnis $(K^{-1}v)$ für Vektoren $v \in \mathbb{C}^{n\times n}$ benötigt. Daher kommen alle Verfahren als Vorkonditionierer in Betracht, die für Vektoren v ein Ergebnis $v' \in \mathbb{C}^{n\times n}$ bestimmen, das sich theoretisch als Anwendung eines linearen Operators K^{-1} auf den Vektor v darstellen lässt.
Das trifft insbesondere auf alle linearen iterativen Verfahren zu, wenn man die Anzahl der Iterationen fest vorgibt. Beispiele, die auf einem Matrix-Splitting Ansatz beruhen

(siehe Abschnitt 3.1.1), sind die *Block-Jacobi-Vorkonditionierung* und die *SSOR-Vorkonditionierung*.

Bei der Block-Jacobi-Vorkonditionierung wählt man die Matrix $K^{-1} = D_b^{-1}$, wobei D_b die Matrix mit Diagonalblöcken von A aus dem Block-Jacobi-Verfahren bezeichnet (siehe (18)). In jeder Iteration des Krylov-Unterraum-Verfahrens müssen dann unabhängige Gleichungssysteme mit Diagonalblöcken aus A gelöst werden. Zur Vorkonditionierung genügt es, die Lösung mit einer geeigneten Methode zu approximieren.
Dadurch lassen sich mit diesem Ansatz effiziente parallele Verfahren mit verteilt gespeicherter Matrix wie in (70) entwickeln. Der Nachteil dieser Methode ist, dass die Einträge der Matrix A außerhalb der Diagonalblöcke vernachlässigt werden; dadurch scheint diese Vorkonditionierung nur dann sinnvoll, wenn diese Einträge klein sind gegenüber den Diagonalblöcken (im Sinne ihrer Norm); das schränkt unter anderem die Skalierbarkeit ein (auf je mehr Prozesse die Matrix A verteilt wird, desto größer wird der Anteil der Einträge außerhalb der Diagonalblöcke).

Zur Vorkonditionierung kann man auch eine feste Anzahl Iterationen des SSOR-Verfahrens verwenden. Der Vorteil dieses Verfahrens liegt darin, dass kein zusätzlicher Speicherplatz und Rechenaufwand zur Konstruktion des Vorkonditionierers benötigt wird. Man erwartet aber auch nur für bestimmte Gleichungssysteme einen Nutzen, beispielsweise solche mit diagonaldominanter Koeffizientenmatrix.
Da im SSOR-Verfahren Gleichungssysteme mit unterer und oberer Dreiecksmatrix durch Vorwärts- beziehungsweise Rückwärtseinsetzen gelöst werden, ist es schwierig, diese Methode effizient auf Parallelrechnern mit einer verteilt gespeicherten Matrix zu implementieren.

Ähnliche Überlegungen ergeben sich auch für andere lineare iterative Verfahren. Für flexible Verfahren (wie FGMRES), kommen zur Vorkonditionierung auch nichtlineare Krylov-Unterraum-Verfahren in Betracht. Dies wird unter anderem in [13] im Kapitel zu GMRES-artigen Verfahren diskutiert.

5.1.2. Unvollständige LU-Zerlegungen

Aus den direkten Verfahren stammt die Idee, die Matrix A in zwei Faktoren, eine untere Dreiecksmatrix L und eine obere Dreiecksmatrix U, zu zerlegen, für die $A = LU$ gilt. Diese können mittels *Gauß-Elimination* berechnet werden:

Algorithmus 3. Gauß-Elimination (ikj-Variante)[10]

1: **for** $i = 1, \ldots, n$ **do**
2: {Initialisiere Zeile aus A'}
3: $a'_{i,1:n} \leftarrow a_{i,1:n}$
4:
5: {Eliminiere Einträge aus der aktuellen Zeile}
6: **for** $k = 1, \ldots, i-1$ **do**
7: $a'_{i,k} \leftarrow a'_{i,k} / a'_{k,k}$
8: $a'_{i,k+1:n} \leftarrow a'_{i,k+1:n} - a'_{i,k}\, a'_{k,k+1:n}$
9: **end for**

10: **end for**

Hier wird angenommen, dass dieser Algorithmus durchführbar ist, also für alle Diagonalelemente $a'_{k,k} \neq 0$ gilt.
Die Matrix L erhält man aus den linken unteren Einträgen der resultierenden Matrix $A' := (a'_{i,j}) \in \mathbb{C}^{n\times n}$ und Diagonaleinträgen von eins, die Matrix R aus der Diagonalen und den rechten oberen Einträgen von A'.
Die Lösung x des linearen Gleichungssystems $Ax = b$ kann aus L, U und b mittels *Vorwärtseinsetzen* $y := L^{-1}b, y \in \mathbb{C}^n$ und *Rückwärtseinsetzen* $x = U^{-1}y$ ermittelt werden.

Die Matrizen L und U können deutlich dichter besetzt sein als die Matrix A. Dadurch ist der Speicher- und Rechenaufwand zur direkten Lösung mittels LU-Zerlegung für die hier betrachteten Fälle unverhältnismäßig groß (gegenüber iterativen Verfahren). Eine dünnbesetzte, approximierte Zerlegung der Form $\tilde{L}\tilde{U} = A - R, R \in \mathbb{C}^{n\times n}$ mit einer unteren Dreiecksmatrix $\tilde{L} \in \mathbb{C}^{n\times n}$ und einer oberen Dreiecksmatrix $\tilde{U} \in \mathbb{C}^{n\times n}$ kann jedoch zur Vorkonditionierung mit $K^{-1} = (\tilde{L}\tilde{U})^{-1}$ verwendet werden. Dies wird als *unvollständige LU-Zerlegung* (ILU) bezeichnet. Algorithmen zur Berechnung einer solchen Zerlegung können aus der Gauß-Eliminiation abgeleitet werden, indem man einen Teil der Einträge aus A' auf Null setzt. Dazu wird die Menge der Matrixeinträge (i, j) in $\tilde{L}$ und $\tilde{U}$ definiert, die ungleich Null sein dürfen; es gilt dann:

$$M := \left\{(i,j) \in [1,\ldots,n]^2 : \tilde{l}_{i,j} \neq 0 \vee \tilde{u}_{i,j} \neq 0\right\} \tag{74}$$

Algorithmus 4. Unvollständige Gauß-Elimination (ikj-Variante)[10]

1: **for** $i = 1,\ldots,n$ **do**
2: {Initialisiere Zeile aus A'}
3: **for** $j = 1,\ldots,n$ **do**
4: **if** $(i,j) \in M$ **then**
5: $\tilde{a}'_{i,j} \leftarrow a_{i,j}$
6: **else**
7: $\tilde{a}'_{i,j} \leftarrow 0$
8: **end if**
9: **end for**
10:
11: {Eliminiere Einträge aus der aktuellen Zeile}
12: **for** $k = 1,\ldots,i-1 \wedge (i,k) \in M$ **do**
13: $\tilde{a}'_{i,k} \leftarrow \tilde{a}'_{i,k}/\tilde{a}'_{k,k}$
14: **for** $j = k+1,\ldots,n \wedge (i,j) \in M$ **do**
15: $\tilde{a}'_{i,j} \leftarrow \tilde{a}'_{i,j} - \tilde{a}'_{i,k}\,\tilde{a}'_{k,j}$
16: **end for**
17: **end for**
18: **end for**

Hier wird nun ebenfalls angenommen, dass dieser Algorithmus durchführbar ist, also für die Diagonalelemente $\tilde{a}'_{k,k} \neq 0$ gilt.
Die Dreiecksmatrizen $\tilde{L}$ und $\tilde{U}$ erhält man wie oben aus $\tilde{A}' = (\tilde{a}'_{i,j}) \in \mathbb{C}^{n\times n}$.

Es gibt viele Möglichkeiten mit Hilfe dieses Ansatzes konkrete Methoden zur Berechnung von $\tilde{L}$ und $\tilde{U}$ zu entwickeln. Eine Darstellung einiger Methoden findet man in [10]. Die Reihenfolge der Berechnung der Einträge aus $\tilde{L}$ und $\tilde{U}$ kann verändert werden, dies führt beispielsweise zur *Crout*-Formulierung, bei der der linke untere Teil von $\tilde{A}$ (also $\tilde{L}$) spaltenweise ermittelt wird. Statt mit einzelnen Einträgen der Matrix zu rechnen, lässt sich das Verfahren auch auf Blöcke der Matrix erweitern (siehe Algorithmus 11), das ist für die hier betrachteten Anwendungsfälle mit dichten 5×5-Blöcken hilfreich. Für die Existenz der Zerlegung ist es notwendig, dass die (sukzessive berechneten) Diagonalblöcke der Zerlegung invertierbar sind.
Auch zur Wahl von M, also der Struktur der Nichtnulleinträge von $\tilde{L}$ und $\tilde{U}$ gibt es verschiedene Ansätze:
Diese kann beispielsweise durch die Struktur der Nichtnulleinträge von A festegelegt werden (erlaube nur Nichtulleinträge an Stellen, die in A ungleich Null sind) oder während der Rechnung über die Norm der berechneten Matrixeinträge (erlaube nur Nichtnulleinträge, deren Norm größer als ein bestimmter Wert ist). Hier lassen sich beliebige weitere Kriterien entwickeln, ein weiteres Beispiel basiert auf einer Abschätzung der Normen von $\tilde{L}$ und $\tilde{U}$ bei der Crout-Formulierung (siehe [10]).
Ein weiterer wichtiger Aspekt ergibt sich daraus, dass die LU-Zerlegung von der Reihenfolge der Gleichungen und Variablen des linearen Gleichungssystem abhängt; das heißt, sie ist nicht invariant gegenüber Permutationen der Zeilen und Spalten der Matrix A. Bei einer Permutation ändert sich insbesondere die Anzahl und die Struktur der Nichtnulleinträge einer vollständigen Faktorisierung LU. Das beeinflusst den Rechenaufwand einer unvollständigen Zerlegung sowie deren Qualität für die Vorkonditionierung. Aus den direkten Methoden stammen Verfahren zur Bestimmung von Permutationen, bei denen die Anzahl der Nichtnullelemente in L und U möglichst gering ist. Desweiteren beeinflusst die Reihenfolge der Zeilen und Spalten die Stabilität des (unvollständigen) Gauß-Eliminationsalgorithmus.
Die Existenz einer unvollständigen LU-Zerlegung (ermittelt mit einem gegebenen Verfahren) ist nur in Sonderfällen gesichert (im Gegensatz zur vollständigen LU-Zerlegung bei geeigneter Permutation der Matrix). Dadurch besteht die Möglichkeit, dass ein Algorithmus zur Bestimmung einer unvollständigen LU-Zerlegung in der Form von Algorithmus 4 fehlschlägt (*break-down*); das ist genau dann der Fall, wenn beim Eliminationsschritt eine Division mit einem Diagonalelement ($\tilde{a}'_{k,k}$ in Zeile 13) von Null auftritt. Auch wenn die Zerlegung existiert, kann sie schlecht konditioniert sein (beispielsweise bei einem Diagonalelement $\tilde{a}'_{k,k}$ nahe Null), sodass die berechneten Faktoren wenig über die ursprüngliche Matrix aussagen.
Trotz dieser theoretischen Probleme erhält man in vielen Fällen mit Hilfe der unvollständigen LU-Zerlegung effektive Vorkonditionierer.

Es ist jedoch schwierig, diesen Ansatz zur Vorkonditionierung effizient auf parallelen Rechnerarchitekturen mit verteiltem Speicher zu implementieren. Das liegt daran, dass zur Berechnung der (unvollständigen) LU-Zerlegung auf jeweils zuvor berechnete Werte zugegriffen werden muss. Dasselbe gilt auch für die Anwendung des Vorkonditionierers in der Form $K^{-1} = (LU)^{-1}$ auf einen Vektor.
In Kombination mit anderen Ansätzen kann man auch parallele Verfahren entwickeln,

ein Beispiel ist eine *Block-Jacobi-ILU-Vorkonditionierung*, bei der die Lösung der Gleichungssysteme mit den Diagonalblöcken des Block-Jacobi-Verfahrens mit Hilfe einer unvollständigen LU-Zerlegung approximiert wird.

5.1.3. Schwarz-Verfahren

Schwarz-Verfahren bieten einen Ansatz zur Vorkonditionierung auf Parallelrechnern. Die grundlegende Idee stammt aus der parallelen Lösung partieller Differentialgleichungen mit Hilfe von Zerlegungen des Rechengebietes: für vorgegebene (geschätzte) Randbedingungen der Teilgebiete können die Lösungen auf diesen Gebieten dann unabhängig voneinander ermittelt werden; mit diesen Lösungen kann man wiederum eine bessere Schätzung für die Randbedingungen der Teilgebiete ermitteln. Dadurch ergibt sich ein iteratives Verfahren zur Lösung des ursprünglichen Problems. Eine ausführliche Darstellung und Untersuchung verschiedener Methoden der Gebietszerlegung zur parallelen Lösung (elliptischer) partieller Differentialgleichungen findet man in [12].

Dieses Vorgehen lässt sich auch auf allgemeine dünnbesetzte lineare Gleichungssysteme übertragen (die nicht aus der Diskretisierung einer partiellen Differentialgleichung stammen müssen). In [13] (S. 200ff) wird der Ansatz der *kanonischen Erweiterung* des Gleichungssystems anhand eines Beispiels skizziert:
Ein Gleichungssystem, das wie in (69) angegeben in zwei Blöcke aufgeteilt wird, kann in folgender Form geschrieben werden:

$$\begin{pmatrix} D_1 & E_{1,g} & & \\ E_{g,1} & G_{1,1} & G_{1,2} & \\ & G_{2,1} & G_{2,2} & E_{g,2} \\ & & E_{2,g} & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ y_2 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ g_1 \\ g_2 \\ f_2 \end{pmatrix} \tag{75}$$

Einem Prozess werden hier jeweils die ersten beiden und die letzten beiden Zeilen des obigen Gleichungssystem zugeordnet. Die Vektoren f_1, g_1, f_2 und g_2 stellen die rechte Seite dar, die Vektoren x_1, y_1, x_2 und y_2 die Unbekannten.
Das System kann nun durch Hinzufügen zusätzlicher Gleichungen und Unbekannter erweitert werden:

$$\left(\begin{array}{ccc|ccc} D_1 & E_{1,g} & & & & \\ E_{g,1} & G_{1,1} & G_{1,2} & & & \\ & I & & -I & & \\ \hline & & -I & & I & \\ & & & G_{2,1} & G_{2,2} & E_{g,2} \\ & & & & E_{2,g} & D_2 \end{array}\right) \begin{pmatrix} x_1 \\ y_1 \\ \tilde{y}_2 \\ \tilde{y}_1 \\ y_2 \\ x_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ g_1 \\ 0 \\ 0 \\ g_2 \\ f_2 \end{pmatrix} \tag{76}$$

Die Lösungen für die Vektoren x_1, y_1, x_2 und y_2 sind offensichtlich identisch zu denen aus dem vorherigen Gleichungssystem. Für die hinzugekommenen Vektoren $\tilde{y}_1$ und $\tilde{y}_2$ ergibt sich $\tilde{y}_1 = y_1$ und $\tilde{y}_2 = y_2$.
Ein einfaches iteratives Verfahren für dieses Gleichungssystem erhält man mit den approximierten Lösungsvektoren x_1^k, y_1^k, $\tilde{y}_2^k$, x_2^k, y_2^k und $\tilde{y}_1^k$ aus dem Iterationsschritt k

durch wiederholtes Lösen der beiden Teilblöcke:

$$\begin{pmatrix} D_1 & E_{1,g} & \\ E_{g,1} & G_{1,1} & G_{1,2} \\ & I & \end{pmatrix} \begin{pmatrix} x_1^{k+1} \\ y_1^{k+1} \\ \tilde{y}_2^{k+1} \end{pmatrix} = \begin{pmatrix} f_1 \\ g_1 \\ \tilde{y}_1^{k} \end{pmatrix}$$
$$\begin{pmatrix} & I & \\ G_{2,1} & G_{2,2} & E_{g,2} \\ & E_{2,g} & D_2 \end{pmatrix} \begin{pmatrix} \tilde{y}_1^{k+1} \\ y_2^{k+1} \\ \tilde{x}_2^{k+1} \end{pmatrix} = \begin{pmatrix} \tilde{y}_2^{k} \\ g_2 \\ f_2 \end{pmatrix} \tag{77}$$

Dies ist eine einfache Form eines *Additive-Schwarz*-Verfahrens. Analog kann man zur Aufteilung eines Gleichungssystems in der Form von (69) in $m \in \mathbb{N}$ Blöcke vorgehen.

Der Grundgedanke der *Schwarz*-Verfahren für allgemeine lineare Gleichungssysteme liegt darin, das Gleichungssystem so um weitere Gleichungen zur Kopplung zu erweitern, dass es sich in möglichst unabhängige Teilprobleme aufteilen lässt. Dabei können sich die Teilprobleme auch „überlappen“, das bedeutet, dass Zeilen der Matrix des ursprünglichen Problems in mehreren Teilproblemen verwendet werden (dadurch kann die Qualität einer Schwarz-Vorkonditionierung verbessert werden, es steigt jedoch auch ihr Rechenaufwand). Welche Gleichungen zur Kopplung geeignet sind, hängt von den Eigenschaften des Problems ab.

5.1.4. Schur-Komplement-Verfahren

Einen etwas anderen Ansatz verfolgen *Schur-Komplement*-Methoden. Diese basieren auf einer Transformation des Gleichungssystems in zwei kleinere, nacheinander zu lösende Systeme. Betrachte dazu ein in Blöcke aufgeteiltes Gleichungssystem der Form

$$\begin{pmatrix} D & E \\ F & G \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{78}$$

mit den invertierbaren Matrizen $D \in \mathbb{C}^{n_x \times n_x}$ und $G \in \mathbb{C}^{n_y \times n_y}$, den Matrizen $E \in \mathbb{C}^{n_x \times n_y}$, $F \in \mathbb{C}^{n_y \times n_x}$ und den Vektoren der Unbekannten $(x, y)^T \in \mathbb{C}^{n_x + n_y}$ und der rechten Seite $(f, g)^T \in \mathbb{C}^{n_x + n_y}$ mit $n_x, n_y \in \mathbb{N}$.
Der linke untere Block F der Matrix kann mittels Multiplikation mit einer nicht singulären Matrix wie folgt elimiert werden:

$$\Leftrightarrow \qquad \begin{pmatrix} I & 0 \\ -FD^{-1} & I \end{pmatrix} \begin{pmatrix} D & E \\ F & G \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} I & 0 \\ -FD^{-1} & I \end{pmatrix} \begin{pmatrix} f \\ g \end{pmatrix} \tag{79}$$

$$\Leftrightarrow \qquad \begin{pmatrix} D & E \\ 0 & G - FD^{-1}E \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g - FD^{-1}f \end{pmatrix} \tag{80}$$

Die Lösung des Gleichungssystems ändert sich durch diese Transformation nicht. Definiert man das *Schur-Komplement* $S \in \mathbb{C}^{n_y \times n_y}, S := G - FD^{-1}E$ und den Vektor $g' \in \mathbb{C}^{n_y}, g' := g - FD^{-1}f$, erhält man die folgende zu dem System aus (78) äquivalente Formulierung:

$$\begin{aligned} Sy &= g' \\ Dx &= f - Ey \end{aligned} \tag{81}$$

Die obige Transformation kann man als eine Block-LU-Zerlegung interpretieren. Dadurch wird deutlich, dass eine LU-Zerlegung des ursprünglichen Gleichungssystems auch als

$$\begin{pmatrix} D & E \\ F & G \end{pmatrix} = \begin{pmatrix} L_D & 0 \\ FU_D^{-1} & L_S \end{pmatrix} \begin{pmatrix} U_D & L_D^{-1}E \\ 0 & U_S \end{pmatrix} \tag{82}$$

geschrieben werden kann. Hier bezeichnet $L_D U_D$ die LU-Zerlegung des Matrixblocks D und $L_S U_S$ die LU-Zerlegung des Schur-Komplements S.

Auf Basis der Schur-Komplement-Transformation lassen sich verschiedene Methoden zur Vorkonditionierung auf Parallelrechnern entwickeln. Wendet man diese beispielsweise auf ein verteilt gespeichertes Gleichungssystem der Form wie in (67) mit $D = \mathrm{diag}(D_1, \ldots, D_m)$ an, kann die Anwendung der Inversen D^{-1} auf einen Vektor effizient parallel auf m Prozessen berechnet werden (durch Lösen eines linearen Gleichungssystems auf jedem Prozess). Ebenso lassen sich Matrix-Vektor-Produkte mit verteilt gespeicherten Matrizen E und F effizient bestimmen. Damit fehlt nur noch eine Möglichkeit zur Bestimmung (oder Approximation) von y aus $Sy = g'$. Ein Verfahren, das auf diesem Ansatz beruht, wird im nächsten Kapitel vorgestellt.
Der wesentliche Unterschied zu den Schwarz-Verfahren liegt darin, dass bei diesen die Transformation des Gleichungssystems im Hinzufügen von Gleichungen und Variablen besteht, hier hingegen in der Elimination eines Matrix-Blocks. Weitere Zusammenhänge zwischen diesen beiden Ansätzen werden am Ende des folgenden Abschnittes dargestellt.

5.1.5. Mehrgitter- und Multilevel-Verfahren

Die Idee bei einem *Mehrgitterverfahren* liegt darin, Diskretisierungen des Problems (beispielsweise die Lösung eines Systems partieller Differentialgleichungen) auf unterschiedlich feinen Gittern zu nutzen, um damit einen optimalen Löser zu konstruieren (siehe S. 407ff in [10]), das heißt ein Lösungsverfahren, dessen (theoretischer) Aufwand mit der Auflösung der Diskretisierung skaliert.[5] Dabei müssen Zwischenergebnisse von einem feineren Gitter auf ein gröberes übertragen werden (*Restriktion*), beziehungsweise von einem gröberen auf ein feineres (*Prolongation*). Einen ausführlichen Überblick über verschiedene Mehrgitterverfahren findet man in [10] und insbesondere im Zusammenhang mit parallelen Methoden zur Lösung (elliptischer) partieller Differentialgleichungen in [12].

Algebraische Mehrgitterverfahren (AMG) übertragen diesen Ansatz auf lineare Gleichungssysteme, die nicht aus einer speziellen Diskretisierung stammen müssen. Anstelle von Problemen auf unterschiedlich feinen Gittern werden hier Gleichungssysteme auf verschiedenen Unterräumen betrachtet. Grundlegende Operationen sind dabei auch Projektionen zur Restriktion von einem „feineren“ Raum $X_h \subset \mathbb{C}^n$ auf einen „gröberen“

[5] Dies ist beispielsweise für ein (nicht vorkonditioniertes) CG-Verfahren zur Lösung einer diskretisierten Poisson-Gleichung nicht der Fall, da die Konditionszahl des Gleichungssystem bei feinerem Gitter steigt und damit ebenfalls die Zahl der benötigten Iterationen.

Raum $X_H \subset X_h$ und die Prolongation von X_H nach X_h:

$$\text{Restriktion:} \qquad I_h^H : X_h \to X_H \tag{83}$$

$$\text{Prolongation:} \qquad I_H^h : X_H \to X_h \tag{84}$$

Für ein gegebenes lineares Gleichungssystem

$$A_h x_h = f^h \qquad \text{mit } x_h, f_h \in X_h,\ A_h \in (X_h \times X_h) \tag{85}$$

wird dann ein „gröberes“ System

$$A_H x_H = f^H \qquad \text{mit } x_H, f_H \in X_H,\ A_H \in (X_H \times X_H) \tag{86}$$

konstruiert, für das gilt:

$$f^H = I_h^H f^h, \tag{87}$$

$$x_h \approx I_H^h x_H, \qquad \text{bzw. } x_h = I_H^h x_H + r_h, \quad r_h \in X_h. \tag{88}$$

Ein Beispiel für Systeme mit symmetrisch positiv definiter Matrix ist die *Galerkin-Projektion* [10, 12], bei der das auf einen gegebenen Unterraum X_H projizierte Gleichungssystem auf Basis des Prolongations-, beziehungsweise des Resktriktionsoperators definiert wird:

$$A_H := I_h^H A_h I_H^h \qquad \text{mit} \qquad I_H^h = (I_h^H)^T \tag{89}$$

Durch die rekursive Anwendung des obigen Ansatzes kann man mehrstufige Verfahren definieren, die auf Projektionen des Gleichungssystems auf verschachtelte Unterräume $X_l \subset X_{l-1} \subset \cdots \subset X_1 \subset X_0 = \mathbb{C}^n$, $l \in \mathbb{N}$ basieren.

Auch Schur-Komplement- und Schwarz-Verfahren können als spezielle zweistufige Verfahren in der obigen Form dargestellt werden:
Für Schur-Komplement-Verfahren in der Darstellung aus (81) ist dann das zu

$$A_h x_h = b_h : \qquad \begin{pmatrix} D & E \\ F & G \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

transformierte System auf X_H:

$$A_H x_H = b_H : \qquad Sy = g'$$

Als Restriktions- und Prolongationsoperator erhält man:

$$\text{Restriktion :} \quad I_h^H = \begin{pmatrix} -FD^{-1} & I \end{pmatrix}$$

$$\text{Prolongation :} \quad I_H^h = \begin{pmatrix} -D^{-1}E \\ I \end{pmatrix}$$

Zur Lösung des ursprünglichen Problems muss die interpolierte Lösung $I_H^h x_H$ noch korrigiert werden:

$$x_h = I_H^h x_H + r_h : \qquad \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -D^{-1}E \\ I \end{pmatrix} y + \begin{pmatrix} D^{-1}f \\ 0 \end{pmatrix}$$

Für den Fall einer symmetrischen Matrix A_h hat der Schur-Komplement-Ansatz die Form einer Galerkin-Projektion.
Die Iteration des Schwarz-Verfahrens aus (77) kann auch als das Lösen eines linearen Gleichungssystems $A_H x_H = b_H$ mit $x_H = (\tilde{y}_1, \tilde{y}_2)^T$ aufgefasst werden. Als Nebenprodukt wird in jeder Iteration auch die approximierte Lösung des ursprünglichen Systems

$$x_h = \left(x_1^{k+1}, y_1^{k+1}, y_2^{k+1}, x_2^{k+1}\right)^T \tag{90}$$

ermittelt (Prolongation und Korrektur).
Insbesondere gilt in dem oben betrachteten Beispiel $(\tilde{y}_1, \tilde{y}_2)^T = (y_1, y_2)^T$; das bedeutet, dass hier der gleiche Unterraum X_H wie bei dem obigen Schur-Komplement-Ansatz verwendet wird, auch wenn sich die Ansätze zur Konstruktion des System $A_H x_H = b_H$ unterscheiden.

6. Die DSC-Methode

Im folgenden wird eine spezielle Methode zur Vorkonditionierung mit Hilfe des Schur-Komplement-Ansatzes vorgestellt, die an eine Methode aus [11] angelehnt ist. Sie basiert auf folgender Darstellung eines dünnbesetzten linearen Gleichungssystems (vergleiche (69))

$$\begin{pmatrix} D_1 & E_1 & & & & & & \\ F_1 & G_1 & & G_{1,2} & \cdots & & G_{1,m} \\ & & D_2 & E_2 & & & \\ & G_{2,1} & F_2 & G_2 & \cdots & & G_{2,m} \\ & & & & & & \\ & \vdots & & \vdots & \ddots & \vdots & \\ & & & & & D_m & E_m \\ & G_{m,1} & & G_{m,2} & \cdots & F_m & G_m \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \\ \vdots \\ x_m \\ y_m \end{pmatrix} = \begin{pmatrix} f_1 \\ g_1 \\ f_2 \\ g_2 \\ \\ \vdots \\ f_m \\ g_m \end{pmatrix} \tag{91}$$

mit $D_i \in \mathbb{C}^{n_{x_i} \times n_{x_i}}$, $E_i \in \mathbb{C}^{n_{x_i} \times n_{y_i}}$, $F_i \in \mathbb{C}^{n_{y_i} \times n_{x_i}}$, $G_i \in \mathbb{C}^{n_{y_i} \times n_{y_i}}$ und $G_{i,j} \in \mathbb{C}^{n_{y_i} \times n_{y_j}}$ für $j = 1, \ldots, m \wedge j \neq i$ und $x_i, f_i \in \mathbb{C}^{n_{x_i}}$, $y_i, g_i \in \mathbb{C}^{n_{y_i}}$ für $i = 1, \ldots, m$.
Zur parallelen Lösung dieses Gleichungssystems können jedem Prozess i zwei aufeinander folgende Zeilen aus obiger Gleichung zugewiesen werden; dies kann man verdeutlichen, indem man das System in einen Satz von Gleichungssystemen umformt:

$$\begin{pmatrix} D_i & E_i \\ F_i & G_i \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j \neq i} G_{i,j}\, y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}, \qquad i = 1, \ldots, m \tag{92}$$

Im folgenden bezeichnet $A_i \in \mathbb{C}^{(n_{x_i}+n_{y_i}) \times (n_{x_i}+n_{y_i})}$ den Diagonalblock des Gleichungssystems auf dem jeweiligen Prozess i:

$$A_i := \begin{pmatrix} D_i & E_i \\ F_i & G_i \end{pmatrix}$$

Unter der Annahme, dass die Diagonalblöcke D_i der Matrix invertierbar sind, und durch Anwendung der Schur-Komplement-Transformation aus (79) erhält man mit $S_i \in \mathbb{C}^{n_{y_i} \times n_{y_i}}$, $S_i := G_i - F_i D_i^{-1} E_i$ und $g_i' \in \mathbb{C}^{n_{y_i}}$, $g_i' := g_i - F_i D_i^{-1} f_i$:

$$\begin{pmatrix} D_i & E_i \\ 0 & S_i \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} 0 \\ \sum_{j \neq i} G_{i,j}\, y_j \end{pmatrix} = \begin{pmatrix} f_i \\ g_i' \end{pmatrix}, \qquad i = 1, \ldots, m \tag{93}$$

Diese Formulierung ist äquivalent zu der Anwendung der Schur-Komplement-Transformation auf das gesamte Gleichungssystem (siehe (81)); das Schur-Komplement-System lässt sich auch in der Form $Sy = g'$ schreiben:

$$\begin{pmatrix} S_1 & G_{1,2} & \cdots & G_{1,m} \\ G_{2,1} & S_2 & \cdots & G_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m,1} & G_{m,2} & \cdots & S_m \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} g_1' \\ g_2' \\ \vdots \\ g_m' \end{pmatrix} \tag{94}$$

Die obige Formulierung zeigt, dass die Schur-Komplement-Transformation einer Block-LU-Zerlegung der Diagonalblöcke des Gleichungssystem entspricht:

$$\begin{pmatrix} D_i & E_i \\ F_i & G_i \end{pmatrix} = \begin{pmatrix} I & 0 \\ F_i D_i^{-1} & I \end{pmatrix} \begin{pmatrix} D_i & E_i \\ 0 & S_i \end{pmatrix} \tag{95}$$

Dadurch lassen sich die Diagonalblöcke S_i der Schur-Komplement-Matrix S mit Hilfe einer LU-Zerlegung darstellen:

$$\begin{pmatrix} D_i & E_i \\ F_i & G_i \end{pmatrix} = \begin{pmatrix} L_{D_i} & 0 \\ F_i U_{D_i}^{-1} & L_{S_i} \end{pmatrix} \begin{pmatrix} U_{D_i} & L_{D_i}^{-1} E_i \\ 0 & U_{S_i} \end{pmatrix} \tag{96}$$
$$\text{mit } L_{D_i} U_{D_i} = D_i \text{ und } L_{S_i} U_{S_i} = S_i$$

Hier wird angenommen, dass die LU-Zerlegungen der Diagonalblöcke existieren.
Ein Verfahren zur Lösung eines linearen Gleichungssystems in der Form von (91) mittels verteiltem Schur-Komplement (DSC, *distributed Schur complement*) ist damit:

Algorithmus 5. DSC-Verfahren

1: Berechne LU-Zerlegungen der Diagonalblöcke:
   $L_i U_i = A_i$, ergibt auch $L_{D_i} U_{D_i}$ und $L_{S_i} U_{S_i}$
2: $g_i' \leftarrow g_i - F_i (L_{D_i} U_{D_i})^{-1} f_i$
3: Löse das Gleichungssystem $Sy = g'$ aus (94)
4: $x_i \leftarrow (L_{D_i} U_{D_i})^{-1} (f_i - E_i y_i)$

Die Schritte 1, 2 und 4 können unabhängig voneinander parallel auf m Prozessen ausgeführt werden. Das Gleichungssystem aus Schritt 3 hat die Form:

$$(L_{S_i} U_{S_i}) y_i + \sum_{\substack{j=1,\dots,m \\ i \neq j}} G_{i,j} y_j = g_i', \qquad i = 1, \dots, m \tag{97}$$

Zur iterativen Lösung dieses Systems bietet sich eine Block-Jacobi-Vorkonditionierung mit der Matrix $\operatorname{diag}((L_{S_1} U_{S_1})^{-1}, \dots, (L_{S_m} U_{S_m})^{-1})$ an, da diese ohne zusätzlichen Aufwand durchgeführt werden kann.
Damit erhält man mit $\hat{g}_i := (L_{S_i} U_{S_i})^{-1} g_i'$ eine Iteration in dem Gleichungssystem:

$$y_i + (L_{S_i} U_{S_i})^{-1} \sum_{\substack{j=1,\dots,m \\ i \neq j}} G_{i,j} y_j = \hat{g}_i, \qquad i = 1, \dots, m \tag{98}$$

Wählt man als Startwert $y^0 = (\hat{g}_1^T, \dots, \hat{g}_m^T)^T$ ergibt sich eine weitere, interessante Formulierung des verteilten Schur-Komplement-Ansatzes:
Zur Darstellung ist es hilfreich, zunächst die Anwendung der Inversen der LU-Zerlegungen der Diagonalblöcke auf die Vektoren der rechten Seite zu betrachten (angewendet auf das gesamte Gleichungssystems entspricht das einer Block-Jacobi-Vorkonditionierung):

$$\begin{pmatrix} \hat{f}_i \\ \hat{g}_i \end{pmatrix} := (L_i U_i)^{-1} \begin{pmatrix} f_i \\ g_i \end{pmatrix} \tag{99}$$

Aus der obigen Formulierung der LU-Zerlegung ergibt sich:

$$\begin{aligned}
\begin{pmatrix} \hat{f}_i \\ \hat{g}_i \end{pmatrix} &= \left(\begin{pmatrix} L_{D_i} & 0 \\ F_i U_{D_i}^{-1} & L_{S_i} \end{pmatrix} \begin{pmatrix} U_{D_i} & L_{D_i}^{-1} E_i \\ 0 & U_{S_i} \end{pmatrix} \right)^{-1} \begin{pmatrix} f_i \\ g_i \end{pmatrix} \\
&= \begin{pmatrix} U_{D_i} & L_{D_i}^{-1} E_i \\ 0 & U_{S_i} \end{pmatrix}^{-1} \begin{pmatrix} L_{D_i} & 0 \\ F_i U_{D_i}^{-1} & L_{S_i} \end{pmatrix}^{-1} \begin{pmatrix} f_i \\ g_i \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} U_{D_i} & L_{D_i}^{-1} E_i \\ 0 & I \end{pmatrix}^{-1}}_{(U_i^b)^{-1}} \underbrace{\begin{pmatrix} I & 0 \\ 0 & U_{S_i} \end{pmatrix}^{-1} \begin{pmatrix} L_{D_i} & 0 \\ F_i U_{D_i}^{-1} & L_{S_i} \end{pmatrix}^{-1}}_{(L_i^b)^{-1}} \begin{pmatrix} f_i \\ g_i \end{pmatrix}
\end{aligned} \tag{100}$$

Offensichtlich ist $L_i^b U_i^b$ ebenfalls eine Zerlegung des Diagonalblocks A_i; einen Algorithmus zur Multiplikation der Inversen $(L_i^b)^{-1}$ mit einem Vektor erhält man durch Vorwärtseinsetzen mit L_i und Rückwärtseinsetzen mit dem unteren Block von U_i, Rückwärtseinsetzen mit dem oberen Block von U_i führt zur Multiplikation der Inversen $(U_i^b)^{-1}$ mit einem Vektor. Damit erhält man nun folgende Darstellung eines verteilten Schur-Komplement-Lösers:

Algorithmus 6. Iteratives DSC-Verfahren mit Block-Jacobi-Vorkonditionierung

1: Berechne LU-Zerlegungen der Diagonalblöcke:
$L_i U_i = A_i$, ergibt auch $L_{S_i} U_{S_i}$ und $L_i^b U_i^b$

2: $\begin{pmatrix} f_i' \\ y_i^0 \end{pmatrix} \leftarrow (L_i^b)^{-1} \begin{pmatrix} f_i \\ g_i \end{pmatrix}$

3: Approximiere iterativ mit k Schritten und Anfangswert $y^0 = ((y_1^0)^T, \ldots, (y_m^0)^T)^T$ die Lösung des Gleichungssystem $y_i + (L_{S_i} U_{S_i})^{-1} \sum_{j \neq i} G_{i,j}\, y_j = y_i^0, \quad i = 1, \ldots, m$

4: $\begin{pmatrix} x_i^k \\ y_i^k \end{pmatrix} \leftarrow (U_i^b)^{-1} \begin{pmatrix} f_i' \\ y_i^k \end{pmatrix}$

In diesem Algorithmus wird in die Lösung des Systems der Diagonalblöcke mittels Vorwärts- und Rückwärtseinsetzen (Schritt 2 und 4) eine globale Iteration mit den Nichtdiagonalblöcken $G_{i,j}$ des ursprünglichen Gleichungssystems eingeschoben.

Die bisher vorgestellten Verfahren ermöglichen die exakte Lösung des Gleichungssystems, benötigen aber dazu eine vollständige LU-Zerlegung der Diagonalblöcke. Zur Vorkonditionierung kann man an Stelle der vollständigen LU-Zerlegung aber auch eine unvollständige LU-Zerlegung verwenden (im Folgenden bezeichnet $\tilde{\cdot}$ jeweils eine Approximation von $\cdot$):

$$\begin{gathered}
A_i \approx \tilde{L}_i \tilde{U}_i = \begin{pmatrix} \tilde{L}_{D_i} & 0 \\ \widetilde{(F_i U_{D_i}^{-1})} & \tilde{L}_{S_i} \end{pmatrix} \begin{pmatrix} \tilde{U}_{D_i} & \widetilde{(L_{D_i}^{-1} E_i)} \\ 0 & \tilde{U}_{S_i} \end{pmatrix}, \\
\text{mit } \tilde{L}_{D_i} \tilde{U}_{D_i} \approx D_i \text{ und } \tilde{S}_i := \tilde{L}_{S_i} \tilde{U}_{S_i} \approx S_i,
\end{gathered} \tag{101}$$

Ebenfalls analog zu oben können damit $(\tilde{L}_i^b)^{-1}$ und $(\tilde{U}_i^b)^{-1}$ dargestellt werden. Dann erhält man ein *approximiertes verteiltes Schur-Komplement-Verfahren*:

Algorithmus 7. DSC-Vorkonditionierung

1: Berechne unvollständige LU-Zerlegungen der Diagonalblöcke:
$\tilde{L}_i\tilde{U}_i \approx A_i$, ergibt auch $\tilde{L}_{S_i}\tilde{U}_{S_i}$ und $\tilde{L}_i^b\tilde{U}_i^b$

2: $\begin{pmatrix}\tilde{f}_i' \\ \tilde{y}_i^0\end{pmatrix} \leftarrow (\tilde{L}_i^b)^{-1}\begin{pmatrix}f_i \\ g_i\end{pmatrix}$

3: Approximiere iterativ mit k Schritten und Anfangswert $\tilde{y}^0 = ((\tilde{y}_1^0)^T, \ldots, (\tilde{y}_m^0)^T)^T$ die Lösung des Gleichungssystem $\tilde{y}_i + (\tilde{L}_{S_i}\tilde{U}_{S_i})^{-1}\sum_{j\neq i} G_{i,j}\,\tilde{y}_j = \tilde{y}_i^0, \quad i = 1, \ldots, m$

4: $\begin{pmatrix}\tilde{x}_i^k \\ \tilde{y}_i^k\end{pmatrix} \leftarrow (\tilde{U}_i^b)^{-1}\begin{pmatrix}\tilde{f}_i' \\ \tilde{y}_i^k\end{pmatrix}$

Hier ist zu beachten, dass in Schritt 2 nicht das exakte Schur-Komplement-System aus (98) verwendet wird, sondern ein Ersatzsystem, das dieses approximiert. Eine genauere Lösung des Ersatzsystems führt jedoch nicht unbedingt zu einer besseren Approximation $((\tilde{x}_i^k)^T, (\tilde{y}_i^k)^T)^T$ der Lösung des ursprünglichen Gleichungssystems $(x_i^T, y_i^T)^T$. Wird in Schritt 3 ein Krylov-Unterraum-Verfahren verwendet, ist dieser Vorkonditionierer kein linearer Operator, sodass er sich nur zur Vorkonditionierung von *flexiblen* Verfahren wie FGMRES eignet.
Die Eigenschaften dieses Verfahrens zur Vorkonditionierung werden in Abschnitt 8 anhand empirischer Untersuchungen mit den Testfällen aus der Strömungstechnik diskutiert.

6.1. Vergleich mit einem in [11] vorgestellten Verfahren

In [11] wird eine ähnliche Methode vorgestellt (vergleiche Algorithmus 3.1 in [11]):

Algorithmus 8. Saad-DSC-Vorkonditionierung

1: Berechne unvollständige LU-Zerlegungen der Diagonalblöcke:
$\tilde{L}_i\tilde{U}_i \approx A_i$, ergibt auch $\tilde{L}_{S_i}\tilde{U}_{S_i}$

2: $\begin{pmatrix}\tilde{f}_i' \\ \tilde{y}_i^0\end{pmatrix} \leftarrow (\tilde{L}_i\tilde{U}_i)^{-1}\begin{pmatrix}f_i \\ g_i\end{pmatrix}$

3: Approximiere iterativ mit k Schritten und Anfangswert $\tilde{y}^0 = ((\tilde{y}_1^0)^T, \ldots, (\tilde{y}_m^0)^T)^T$ die Lösung des Gleichungssystem $\tilde{y}_i + (\tilde{L}_{S_i}\tilde{U}_{S_i})^{-1}\sum_{j\neq i} G_{i,j}\,\tilde{y}_j = \tilde{y}_i^0, \quad i = 1, \ldots, m$

4: $\begin{pmatrix}(\tilde{x}_i^k)' \\ (\tilde{y}_i^k)'\end{pmatrix} \leftarrow (\tilde{L}_i\tilde{U}_i)^{-1}\begin{pmatrix}\tilde{f}_i \\ \tilde{g}_i - \sum_{j\neq i} G_{i,j}\tilde{y}_j^k\end{pmatrix}$

Der Unterschied zwischen diesem Algorithmus und dem vorherigen besteht hauptsächlich in der Rücktransformation der Approximation für die Variablen y_i aus Schritt 3 zur Approximation von (x_i, y_i) in Schritt 4. Das geschieht hier durch Lösen des Gleichungssystems (vergleiche (92))

$$(\tilde{L}_i\tilde{U}_i)\begin{pmatrix}(\tilde{x}_i^k)' \\ (\tilde{y}_i^k)'\end{pmatrix} = \begin{pmatrix}f_i \\ g_i\end{pmatrix} - \begin{pmatrix}0 \\ \sum_{j\neq i} G_{i,j}\tilde{y}_j^k\end{pmatrix}. \tag{102}$$

Der Vorteil liegt darin, dass nicht auf einzelne Blöcke der unvollständigen LU-Zerlegung zugegriffen werden muss – anstelle einer unvollständigen LU-Zerlegung könnte so auch

ein anderes Verfahren zur approximativen Lösung des Gleichungssystem mit den Diagonalblöcken (Lösen von (92) für gegebene vorgegebene Werte für y_j) verwendet werden. Es wird dann jedoch eine andere Möglichkeit zur Darstellung des Schur-Komplements S_i benötigt.
Zum Vergleich der beiden Verfahren kann man (102) umformen:

$$
\begin{aligned}
\Leftrightarrow \qquad & (\tilde{L}_i^b \tilde{U}_i^b) \begin{pmatrix} (\tilde{x}_i^k)' \\ (\tilde{y}_i^k)' \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix} - \begin{pmatrix} 0 \\ \sum_{j \neq i} G_{i,j} \tilde{y}_j^k \end{pmatrix} \\
\Leftrightarrow \qquad & \begin{pmatrix} (\tilde{x}_i^k)' \\ (\tilde{y}_i^k)' \end{pmatrix} = (\tilde{L}_i^b \tilde{U}_i^b)^{-1} \left(\begin{pmatrix} f_i \\ g_i \end{pmatrix} - \begin{pmatrix} 0 \\ \sum_{j \neq i} G_{i,j} \tilde{y}_j^k \end{pmatrix} \right) \\
\Leftrightarrow \qquad & \begin{pmatrix} (\tilde{x}_i^k)' \\ (\tilde{y}_i^k)' \end{pmatrix} = (\tilde{U}_i^b)^{-1} \begin{pmatrix} \tilde{f}_i' \\ \tilde{y}_i^0 - (\tilde{L}_{S_i} \tilde{U}_{S_i})^{-1} \sum_{j \neq i} G_{i,j} \tilde{y}_j^k \end{pmatrix} \\
\Leftrightarrow \qquad & \begin{pmatrix} (\tilde{x}_i^k)' \\ (\tilde{y}_i^k)' \end{pmatrix} = (\tilde{U}_i^b)^{-1} \begin{pmatrix} \tilde{f}_i' \\ (\tilde{y}_i^k)' \end{pmatrix} \qquad \text{mit} \\
& \qquad\qquad (\tilde{y}_i^k)' = \tilde{y}_i^0 - (\tilde{L}_{S_i} \tilde{U}_{S_i})^{-1} \sum_{j \neq i} G_{i,j} \tilde{y}_j^k
\end{aligned}
\tag{103}
$$

Die Berechnung des Vektors $(\tilde{y}^k)'$ aus $\tilde{y}^k$ entspricht einer Iteration eines Block-Jacobi-Verfahrens für das approximierte Schur-Komplement-System. Der Unterschied zwischen der Methode aus [11] zu der aus dem vorherigen Abschnitt liegt also darin, dass nach der Anwendung eines (zu wählenden) iterativen Verfahrens (zur Bestimmung von $\tilde{y}^k$) eine zusätzliche Block-Jacobi-Iteration des Schur-Komplement-Systems durchgeführt wird.
Die Frage ist jedoch, ob es bei der Verwendung eines Krylov-Unterraum-Verfahrens zur Iteration des Schur-Komplement-Systems in Schritt 3 der obigen Algorithmen nicht günstiger ist, anstelle der Block-Jacobi-Iteration eine weitere Iteration des Krylov-Unterraum-Verfahrens (mit nahezu gleichem Aufwand) durchzuführen.
Zusätzlich werden in der Variante von [11] zwei Anwendungen von $(\tilde{L}_i \tilde{U}_i)^{-1}$ verwendet, während bei der hier vorgestellten DSC-Vorkonditionierung nur eine aufgeteilte Anwendung von $(\tilde{L}_i^b \tilde{U}_i^b)^{-1}$ und bei einer weiteren Iteration eine von $(\tilde{L}_{S_i} \tilde{U}_{S_i})^{-1}$ benötigt wird.

6.2. Iterative Verbesserung des approximierten Schur-Komplement-Systems

Die parallele Effizienz eines Verfahrens hängt wesentlich von dem Verhältnis des Rechenaufwandes zum Lösen der Teilprobleme auf den einzelnen Prozessen zu dem Aufwand der Kommunikation zwischen den Prozessen ab. Je mehr Prozesse verwendet werden, desto stärker fällt der Aufwand der Kommunikation ins Gewicht (bei gleichbleibender Problemgröße).
Daher kann der Gesamtaufwand des vorkonditionierten iterativen Verfahrens eventuell verringert werden, wenn der Rechenaufwand zur Vorkonditionierung auf jedem Prozess erhöht wird, um Iterationsschritte und damit Kommunikationsaufwand einzusparen. Eine Möglichkeit dazu besteht hier darin, die Approximation des Schur-Komplement-Systems zu verbessern (hier ist nicht eine genauere iterative Lösung des Schur-Komplement-

Systems gemeint, sondern das approximierte Gleichungssystem selbst).
Diese hängt in dem obigen Verfahren von der Qualität der unvollständigen LU-Zerlegung der Diagonalblöcke ($\tilde{L}_i\tilde{U}_i$) ab. Eine größere unvollständige LU-Zerlegung hat somit eventuell den gewünschten Effekt; dabei wird der Rechenaufwand auf allen Prozessen erhöht und eventuell die Approximation der Lösung der DSC-Vorkonditionierung bei gleichbleibender Zahl an Iterationen des Schur-Komplement-Systems verbessert.
Man kann jedoch auch das Ergebnis der Anwendung der Inversen einer unvollständigen LU-Zerlegung ($(\tilde{L}\tilde{U})^{-1}$ mit $A \approx \tilde{L}\tilde{U}$) iterativ verbessern (das heißt ein iteratives Verfahren mit dem Vorkonditionierer $(\tilde{L}\tilde{U})^{-1}$ einsetzen). Da dieser Ansatz nicht direkt auf die obige DSC-Vorkonditionierung übertragbar ist, wird im Folgenden eine mögliche Vorgehensweise skizziert:

Betrachte dazu zunächst wieder die allgemeine Form eines DSC-Verfahren:

Algorithmus 9. Allgemeines DSC-Verfahren (vgl. Algorithmus 5)

1: $g_i' \leftarrow g_i - F_i D_i^{-1} f_i$
2: Löse das Gleichungssystem $(G_i - F_i D_i^{-1} E_i) y_i + \sum_{j \neq i} G_{i,j} y_j = g_i', \quad i = 1, \ldots, m$
3: $x_i \leftarrow D_i^{-1}(f_i - E_i y_i)$

Der Algorithmus verdeutlicht, dass bei iterativer Lösung von Schritt 3 zur Berechnung der verteilten Schur-Komplement-Transformation nur folgende Operationen benötigt werden: die Multiplikation von Blöcken der Matrix mit Vektoren und das Lösen von Gleichungssystemen mit der Matrix D_i.
Analog zum Vorgehen bei Algorithmus 7 kann man auch in diesem Fall zunächst eine unvollständige LU-Zerlegung des Diagonalblocks A_i berechnen: Damit erhält man die Zerlegung $\tilde{L}_{D_i}\tilde{U}_{D_i}$, mit der ein iteratives Verfahren zur Lösung der Gleichungssysteme mit D_i vorkonditioniert werden kann. Ebenso kann man die Zerlegung $\tilde{L}_{S_i}\tilde{U}_{S_i}$ zur Vorkonditionierung des Gleichungssystems aus Schritt 3 mit der Matrix

$$\begin{pmatrix} (\tilde{L}_{S_1}\tilde{U}_{S_1})^{-1} & & \\ & \ddots & \\ & & (\tilde{L}_{S_m}\tilde{U}_{S_m})^{-1} \end{pmatrix} \tag{104}$$

verwenden.
Damit ergibt sich ein Verfahren, das wie die DSC-Vorkonditionierung aus Algorithmus 7 nur unvollständige LU-Zerlegungen der Diagonalblöcke benötigt, aber trotzdem das Schur-Komplement-System (im Rahmen der Rechengenauigkeit) mit einer vorgegebenen Genauigkeit darstellen kann (bei geeigneten Abbruchbedingungen des iterativen Verfahrens für D_i^{-1}). Daher wird es in den weiteren Abschnitten dieser Arbeit als *quasi-exakte DSC-Vorkonditionierung* bezeichnet.
Es kann folglich prinzipiell nicht nur zur Vorkonditionierung, sondern auch direkt als Verfahren zur Lösung des Gleichungssystems eingesetzt werden, der Aufwand dazu ist jedoch recht groß, da in jeder Iteration des Schur-Komplement-Systems aus Schritt 3 oben ein lineares Gleichungssystem mit D_i gelöst werden muss.
Wird dieses Verfahren zur Vorkonditionierung verwendet und dabei die Lösung der Gleichungssysteme mit D_i nur angenähert, dann erhält man ebenfalls nur eine Approximation für die Multiplikation von $S_i = G_i - F_i D_i^{-1} E_i$ mit einem Vektor. Ein Problem

ist jedoch, dass diese Approximation von S_i für unterschiedliche Vektoren keine konstante Matrix ergibt und man dadurch nicht von einer linearen Abbildung $\tilde{S}_i$ sprechen kann. Dies lässt sich nicht durch die Verwendung eines flexiblen iterativen Verfahrens zur Lösung des Schur-Komplement-Systems beheben, da nicht der Vorkonditionierer von Iterationsschritt zu Iterationsschritt variiert, sondern die Koeffizientenmatrix des zu lösenden Gleichungssystems.
Werden die Lösungen der Gleichungssysteme in D_i mit einer vorgegebenen Genauigkeit bestimmt, kann man das vereinfacht auch als ein Verfahren mit der richtigen Matrix bei einer entsprechenden Rechen(un)genauigkeit betrachten. Aus diesem Grunde kann man vermuten, dass die Lösungen der Gleichungssysteme in D_i ausreichend genau bestimmt werden müssen, um die Stabilität eines Krylov-Unterraum-Verfahrens für dieses Gleichungssystem zu gewährleisten. Als Heuristik kann man die Genauigkeit der Lösungen der Gleichungssysteme in D_i so wählen, dass sie um einige Größenordnungen kleiner ist als die angestrebte Genauigkeit der Lösung des Schur-Komplement-Systems.
Ein weiterer wesentlicher Nachteil gegenüber der DSC-Vorkonditionierung aus Algorithmus 7 besteht darin, dass sich hier zur Vorkonditionierung des Schur-Komplement-Systems mit der Matrix $\text{diag}((L_{S_1}U_{S_1})^{-1}, \ldots, (L_{S_m}U_{S_m})^{-1})$ ein zusätzlicher Aufwand ergibt.

Zusammenfassend lässt sich sagen, dass sich die Approximation der DSC-Vorkonditionierung mit dem hier skizzierten Verfahren durch lokale Iterationen auf jedem Prozess verbessern lässt, dies jedoch mit einem deutlich größeren Aufwand verbunden ist.
Einen kurzen empirischen Vergleich mit der einfacheren DSC-Vorkonditionierung aus Algorithmus 7 findet man in Abschnitt 8.

7. Implementierung in der DSC-Bibliothek des DLR

In der Programmbibliothek DSC-HPC[6] des DLR sind verschiedene Verfahren zur parallelen Lösung großer, dünnbesetzter linearer Gleichungssystem implementiert. Sie ist größtenteils in der Programmiersprache *Fortran* geschrieben und verwendet das *Message Passing Interface* (MPI)[7] zur parallelen Rechnung auf verteilten Computersystemen. Einige der verwendeten Algorithmen findet man in Anhang A.
Es werden nun einige Details der Implementierung erläutert, die zur Interpretation der Ergebnisse aus Abschnitt 8 notwendig sind; das ist zum Einen das Datenformat zur Darstellung des linearen Gleichungssystems und zum Anderen der Ablauf des gesamten Lösungsverfahrens sowie die Verknüpfung der einzelnen Verfahrensschritte.

7.1. Datenformat

Zur Darstellung des linearen Gleichungssystems wird ein verteiltes Block-CRS-Format (*compressed row storage*, siehe beispielsweise [10]) eingesetzt. Wie in Abschnitt 4 skizziert speichert jeder Prozess einen Satz von Zeilen des Gleichungssystems $Ax = b$, das heißt einen Teil der Matrix A und Teile der Vektoren x und b (und weiterer im Lösungsverfahren benötigter Vektoren).
Da die Matrix A dünnbesetzt ist, muss man nicht alle ihre Einträge speichern. Daher werden die Nichtnulleinträge, beziehungsweise Blöcke von Nichtnulleinträgen (hier der Größe 5×5), aus einer Zeile, beziehungsweise einem Block von Zeilen, hintereinander im Speicher abgelegt. Zur Adressierung benötigt man zusätzlich für jeden Eintrag beziehungsweise Block einen Spaltenindex (Block-CRS-Format). Weitere, im Laufe der Rechnung benötigte, dünnbesetzte Matrizen wie beispielsweise unvollständige LU-Zerlegungen werden ebenso dargestellt.

7.2. Ablauf des Lösungsverfahrens

Der folgende Algorithmus skizziert den Ablauf des in der DSC-Bibliothek implementierten Lösungsverfahrens. Hier werden einige Schritte vereinfacht oder ausgelassen und insbesondere nur die Verfahren erwähnt, die in dieser Arbeit untersucht werden (es können noch weitere Krylov-Unterraum-Verfahren zur Lösung des Gleichungssystems sowie verschiedene Vorkonditionierer ausgewählt werden).

Algorithmus 10. Löser der DSC-Bibliothek

1: Repartitioniere das lineare Gleichungssystem
2: Sortiere die Zeilen der Matrix für die DSC-Vorkonditionierung um
3: Permutiere Zeilen und Spalten der Diagonalblöcke der Matrix zur LU-Fill-In-Reduzierung um
4: Normalisiere Matrix

[6] Siehe http://software.dlr.de/p/dschpc
[7] Siehe http://www.mpi-forum.org/

5: Berechne unvollständige LU-Zerlegungen der Diagonalblöcke der Matrix

6: Löse das Gleichungssystem iterativ mit einem vorkonditionierten FGMRES-Verfahren, in jeder Iteration:

   1: Mögliche Vorkonditionierungen:

      - Block-Jacobi-ILU
      - DSC
      - Saad-DSC (eine DSC Variante aus [11])
      - Quasi-exakt-DSC

   2: Verteilte Matrix-Vektor-Multiplikation

   3: Orthogonalisieren der Basisvektoren des GMRES-Unterraumes, Givens-Rotation

7: Setze Ergebnisvektor zusammen

Zunächst kommen Schritte zur Vorbereitung des Lösers:
In Schritt 1 wird das Gleichungssystem so auf die Prozesse verteilt, dass der Kommunikationsaufwand bei der Matrix-Vektor-Multiplikation möglichst gering ist. Dies geschieht mit Hilfe eines Algorithmus zur Graph-Partitionierung aus der ParMETIS-Bibliothek[6]. Ausgangspunkt der Repartitionierung ist eine beliebige zeilenweise Verteilung des Gleichungssystems.
Bei der Verwendung einer DSC-Vorkonditionierung wird in Schritt 2 durch Vertauschungen von Zeilen auf den einzelnen Prozessen das Gleichungssystem in die benötigte Form (siehe (91)) gebracht.
Zur Reduzierung der Anzahl der Nichtnullelemente in der unvollständigen LU-Zerlegung werden mit Hilfe der METIS-Bibliothek[5] in Schritt 3 die Zeilen und Spalten der Diagonalblöcke umsortiert. Dabei wird beachtet, dass bei den verschiedenen DSC-Vorkonditionierungen die obige Form (91) des Gleichungssystems erhalten bleibt.
In Schritt 4 werden die Zeilen und Spalten der Matrix so skaliert, dass ihre Norm jeweils möglichst nahe an eins liegt; das reduziert mit geringem Aufwand die Konditionszahl der Matrix und verbessert dadurch die Ergebnisse der nachfolgenden Algorithmen.
Nun kann in Schritt 5 auf jedem Prozess eine unvollständige LU-Zerlegung des jeweiligen Diagonalblocks bestimmt werden.
Damit sind nun die Schritte zur Vorbereitung abgearbeitet und das vorkonditionierte Krylov-Unterraum-Verfahren zur Lösung des Gleichungssystems kann aufgerufen werden (Schritt 6). Da für die hier untersuchten Vorkonditionierer ein flexibles Verfahren benötigt wird, kommt ein FGMRES-Verfahren zum Einsatz.
Zum Schluss (Schritt 7) müssen aufgrund der Repartionierung noch die Teile des Lösungsvektors an die passenden Prozesse gesendet werden. Damit wird die ursprüngliche Reihenfolge der Lösungsvektorkomponenten wiederhergestellt.

Der Ablauf des FGMRES-Verfahrens ist hier nochmal grob skizziert: Es stehen vier verschiedene Vorkonditionierer zu Auswahl. In jedem Iterationsschritt wird dieser einmal aufgerufen und es muss jeweils eine Matrix-Vektor-Multiplikation berechnet sowie ein neuer Basisvektor des GMRES-Unterraumes zu den vorherigen orthogonalisiert werden.

7.3. Parameter des Lösers

In vielen Schritten des obigen Lösungsverfahrens werden zusätzliche Parameter benötigt. Einige wichtige Parameter werden nun vorgestellt:

Durch die Blockgröße wird angegeben, ob und wie die Nichtnullelemente der Matrix in Blöcken angeordnet sind. Dies beeinflusst zunächst das Datenformat, also wie auf die Matrixeinträge zugegriffen wird. Nur bei der unvollständigen LU-Zerlegung ergibt sich durch die Verwendung einer Block-Gauß-Eliminitions-Methode ein anderes Verfahren.
Die Größe der unvollständigen LU-Zerlegungen der Diagonalblöcke der Matrix wird durch zwei Parameter bestimmt: Der Wert von $lfil$ begrenzt die maximale Zahl der Nichtnulleinträge pro Zeile beziehungsweise Spalte (relativ zur Zahl der Nichtnulleinträge der ursprünglichen Matrix). Welche Werte der Gauß-Elimination verworfen werden, bestimmt eine durch den Wert von ϵ_{drop} spezifizierte Toleranzgrenze.
Ein weiterer Parameter ist die Zahl der Iterationen, nach der der FGMRES-Algorithmus neu gestartet wird (also die maximale Größe des betrachteten Krylov-Unterraumes). Eine Abbruchbedingung beendet das Verfahren, wenn eine gewünschte Genauigkeit der Lösung erreicht wurde (hier gemessen in der relativen Norm des Residuums).

Wie oben dargestellt, kann man verschiedene Methoden zur Vorkonditionierung nutzen, betrachtet werden hier eine Block-Jacobi-ILU-Vorkonditionierung, die in dieser Arbeit beschriebene DSC-Vorkonditionierung und quasi-exakte DSC-Vorkonditionierung. Zum Vergleich wird auch noch eine Variante der DSC-Vorkonditionierung von Saad aus [11] betrachtet.
Bei den DSC-Vorkonditionierungen kann das Krylov-Unterraum-Verfahren zur Lösung des Schur-Komplement-Systems ausgewählt (GMRES oder BICGSTAB) und die Zahl der Iterationen dieses Verfahrens festgelegt werden. Bei der quasi-exakten Variante der DSC-Vorkonditionierung kommt noch eine Abbruchbedingung (hier eine Schranke für die relative Norm des Residuums) für das Lösungsverfahren zur Darstellung des Schur-Komplement-Systems hinzu.

8. Empirische Untersuchung der DSC-Methode

In dieser Arbeit werden die Eigenschaften der DSC-Vorkonditionierung anhand der Anwendungsbeispiele aus Abschnitt 2 empirisch untersucht.
Frühere Ergebnisse der verteilten-Schur-Komplement-Vorkonditionierung findet man in [11] und [1]. Diese beziehen sich (unter anderem) auf die hier als Saad-DSC bezeichnete Variante. In [1] wird eine frühere Version der DSC-Bibliothek des DLR verwendet. Beide Artikel verwenden jedoch lineare Gleichungssysteme aus anderen Gebieten als der Strömungstechnik, deren Eigenschaften deutlich von den hier verwendeten Testfällen abweichen. Dadurch sind sie für die Betrachtung hier nur bedingt aussagekräftig.

Für die folgenden Untersuchungen werden die Gleichungssysteme der Testfälle TAU, TRACE-THD und TRACE-UHBR mit der DSC-Bibliothek mit verschiedenen Parametern auf unterschiedlich vielen Prozessorkernen gelöst. Hier wird die Iteration des vorkonditionierten FGMRES(m)-Verfahrens jeweils abgebrochen, sobald eine Reduzierung des Residuums um mindestens einen Faktor 10^{-5} erreicht ist.
Bei jeder Rechnung wird die benötigten Rechenzeit des gesamten Lösungsalgorithmus betrachtet und bei gegebener Anzahl an Prozessorkernen das Verfahren und die Parameter als optimal eingeschätzt, die zu der kleinsten Rechenzeit führen.
In vielen Fällen ist es ebenfalls interessant, welcher Teil des Lösers wie viel Rechenzeit benötigt; den Gesamtaufwand kann man aufteilen in den Aufwand zur Bestimmung der unvollständigen LU-Zerlegungen der Diagonalblöcke und den Aufwand der vorkonditionierten FGMRES-Iterationen, die Rechenzeit aller weiterer Schritte des Algorithmus ist im Verhältnis dazu klein und wird hier vernachlässigt.
Zusammen mit der benötigten Anzahl an Iterationen (bis zu einer vorgegebenen Reduzierung des Residuums) kann man den Aufwand und die Qualität der verwendeten Vorkonditionierer vergleichen.

Aufgrund der großen Zahl an Parametern des Lösers wurde hier folgende Vorgehensweise gewählt:
Als erstes werden anhand eines Beispiels die Auswirkungen der Blockgröße betrachtet.
Dann werden die Parameter der unvollständigen LU-Zerlegungen der Diagonalblöcke gewählt. Als Referenz dient hier (wie an anderen Stellen auch) die Block-Jacobi-ILU-Vorkonditionierung.
Als nächstes werden die Ergebnisse der DSC-Vorkonditionierung bei der Verwendung des GMRES- und des BICGSTAB-Verfahren gegenübergestellt.
Aufbauend auf den vorangegangenen Ergebnissen können zuletzt die Skalierbarkeit des gesamten Lösungsverfahrens und Aspekte der parallelen Rechnung auf vielen Prozessorkernen untersucht werden. Dazu wird die Lösung eines Testfalles fester Größe mit unterschiedlich vielen Kernen bestimmt und die Rechenzeiten verglichen (*strong scaling*).

Alle Rechnungen wurden auf dem 2011 neu eingerichteten Cluster der RWTH-Aachen durchgeführt. In Anhang B findet man weitere Informationen zu der hier verwendeten Hard- und Software, sowie die Ergebnisse zu allen Rechnungen, von denen hier nur einzelne Auszüge vorgestellt werden.

Hier soll noch angemerkt werden, dass die Anzahl der Iterationen eines vorkonditio-

nierten Verfahrens auch steigen kann, wenn sich der Vorkonditionierer verbessert (in dem Sinne, dass er die Anwendungen von A^{-1} besser approximiert). Die Überlegungen zu Eigenschaften verschiedener Vorkonditionierungen in den folgenden Abschnitten können folglich nur als Heuristik aufgefasst werden.

8.1. Berücksichtigung der Blockstruktur der Matrix

Zunächst soll hier angemerkt werden, dass in dieser Arbeit bisher zwar immer komplexe lineare Gleichungssysteme der Form $Ax = b$, $A \in \mathbb{C}^{n \times n}$, $x, b \in \mathbb{C}^n$ betrachtet wurden, diese jedoch auch äquivalent als reelle Gleichungssysteme formuliert werden können:

$$\begin{aligned} & Ax = b \\ \Leftrightarrow \quad & \begin{pmatrix} \mathrm{Re}(A) & -\mathrm{Im}(A) \\ \mathrm{Im}(A) & \mathrm{Re}(A) \end{pmatrix} \begin{pmatrix} \mathrm{Re}(x) \\ \mathrm{Im}(x) \end{pmatrix} = \begin{pmatrix} \mathrm{Re}(b) \\ \mathrm{Im}(b) \end{pmatrix} \end{aligned}$$

Die Matrix in der obigen reellen Form zu speichern ist offensichtlich nicht sinnvoll, da die Teile $\mathrm{Re}(A)$ und $\mathrm{Im}(A)$ (beziehungsweise $-Im(A)$) jeweils zweimal im Speicher abgelegt würden. Im Prinzip ist die Darstellung als komplexes System aber nur ein spezielle Anordnung des reellen Gleichungssystems in Blöcken der Dimension 2×2 der Form

$$\begin{pmatrix} \mathrm{Re}(a_{ij}) & -\mathrm{Im}(a_{ij}) \\ \mathrm{Im}(a_{ij}) & \mathrm{Re}(a_{ij}) \end{pmatrix},$$

die kompakt als jeweils eine komplexe Zahl a_{ij} geschrieben werden können.

Bei den hier untersuchten Testfällen ist (zusätzlich) eine Gruppierung der Nichtnulleinträge der Matrix in 5×5-Blöcken möglich.
Dadurch reduziert sich der Aufwand zur Adressierung der Nichtnullelemente erheblich. Des Weiteren werden eventuell auch die Gleitkommaoperationen schneller durchgeführt, da Rechenoperationen mit (dichten) kleinen Matrizen und Vektoren besser von Compiler und Prozessor optimiert werden können.

Block-Größe	ILU ϵ_{drop}	ILU-Fill-In	ILU Dauer [s]	Iterationen	Gesamtdauer [s]
1	0,001	0,8	17,90	77	37,88
5	0,005	0,8	2,77	59	7,49

Tabelle 2: Einfluss unterschiedlicher Blockgrößen (TAU-Testfall auf 12 Kernen mit Block-Jacobi-ILU-Vorkonditionierung)

Die obigen Ergebnisse zeigen eine Beschleunigung der gesamten Rechnung um einen Faktor 5. Besonders deutlich sinkt der Aufwand zur Berechnung der unvollständigen LU-Zerlegung der Diagonalblöcke der Matrix, die auch als spezielle Block-LU-Zerlegung für eine Matrix aus dichtbesetzten 5×5 Submatrizen formuliert werden kann. Dabei nutzt diese Formulierung den Umstand, dass die 5×5 Submatrizen auf der Diagonalen bei den hier betrachteten Anwendungen invertiertbar sind.

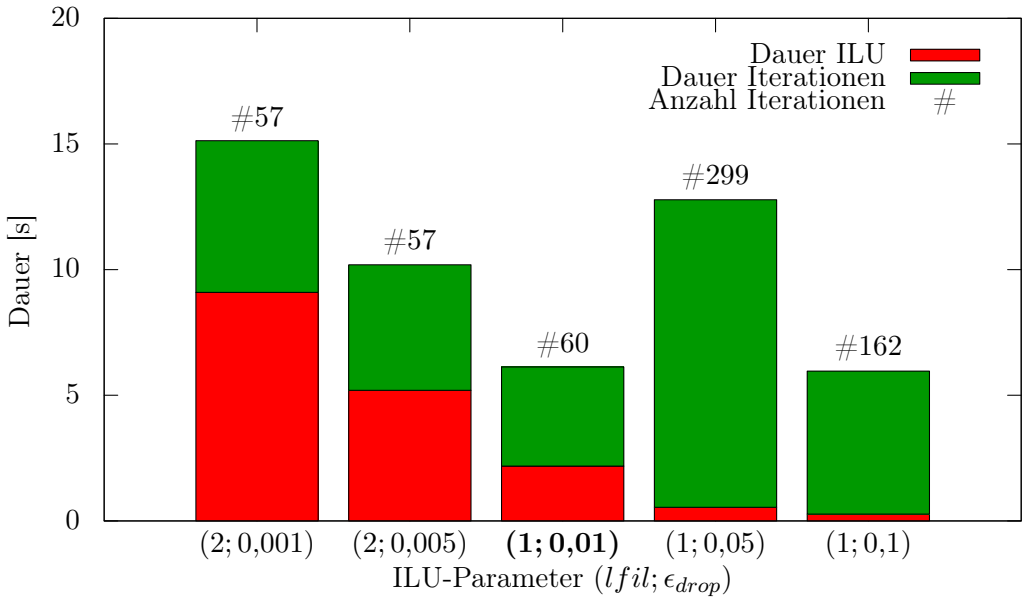
Zum Vergleich wurden die Parameter hier so gewählt, dass die Größe der unvollständigen LU-Zerlegungen ähnlich ist (ILU-Fill-In: bezeichnet hier das Verhältnis von der Anzahl hinzugekommener Nichtnulleinträge der Zerlegung zu den Nichtnulleinträgen der Matrix).
In diesem Beispiel sinkt auch die Anzahl der benötigten Iterationen durch die Verwendung der Block-ILU-Zerlegungen der Diagonalblöcke zur Block-Jacobi-Vorkonditionierung. Hier kann man nur vermuten, dass die unvollständigen Block-LU-Zerlegungen eine bessere Approximation der Diagonalblöcke ergeben, weil bei ihrer Berechnung die 5×5-Blöcke der ursprünglichen Matrix jeweils als Ganzes berücksichtigt oder verworfen werden. Dies gibt in gewissem Sinne die Struktur des Problems wieder; ein 5×5-Block wird jeweils durch die Abhängigkeiten von den Strömungsgrößen aus einer Zelle der Diskretisierung bestimmt.

Durch die Berücksichtigung der speziellen Blockstruktur des linearen Gleichungssystems ergeben sich demgemäß erheblich Vorteile. In allen folgenden Rechnungen werden daher Blöcke der Größe 5×5 verwendet.

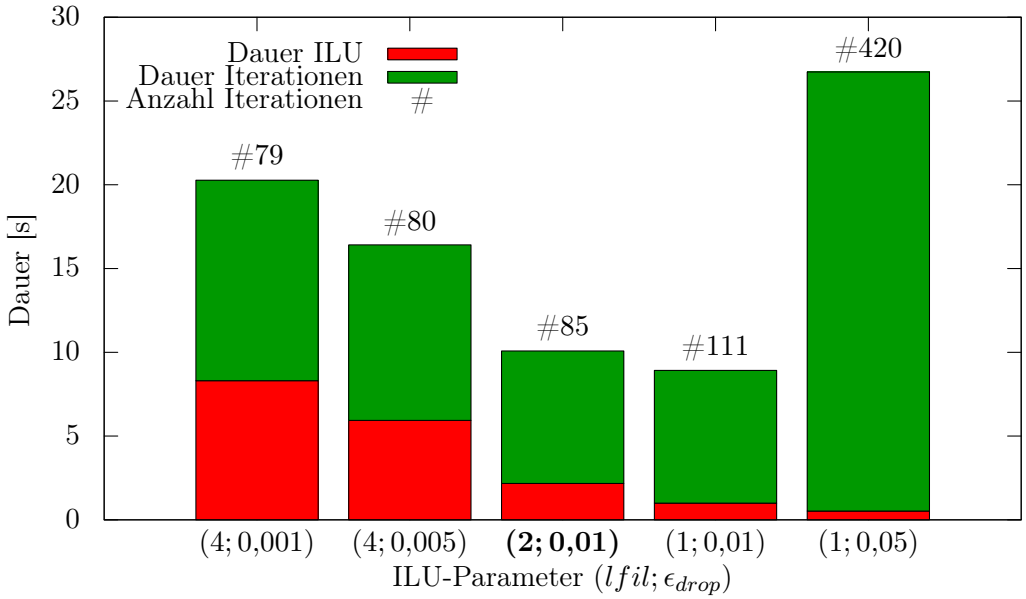
8.2. Wahl der Parameter der unvollständigen LU-Zerlegung

Die Parameter der unvollständigen LU-Zerlegungen der Diagonalblöcke bestimmen, welche Einträge während der Gauß-Elimination verworfen werden. Damit legen sie die Größe der unvollständigen LU-Zerlegungen im Sinne der Anzahl ihrer Nichtnulleinträge fest.
In vielen Fällen führen größere unvollständige LU-Zerlegungen zu einer besseren Vorkonditionierung, sodass die Anzahl der benötigten Iterationen des Verfahrens bei einer Block-Jacobi-ILU-Vorkonditionierung sinkt; es steigt jedoch der Aufwand zur Bestimmung der unvollständigen LU-Zerlegungen und der Aufwand der einzelnen vorkonditionierten Iterationsschritte.
Aus diesem Grund wurden hier für jeden Testfall zunächst Rechnungen mit unterschiedlichen ILU-Parametern auf einer festen Anzahl an Prozessorkernen durchgeführt, Auszüge der Ergebnisse dieser Rechnungen für die Testfälle TAU und TRACE-THD zeigt die Abbildung 6.
Die obigen Überlegungen spiegeln sich auch in den Rechenergebnissen wieder. Von links nach rechts verringert sich in Abbildung 6 die Größe der unvollständigen LU-Zerlegungen der Diagonalblöcke. Dadurch sinkt hier zunächst der gesamte Rechenaufwand, während die benötigte Iterationszahl nur leicht steigt. Ab einem gewissen Punkt steigt die Zahl der Iterationen deutlich an, wodurch auch der Gesamtaufwand wieder wächst.
Hier wurden jeweils die Parameter als am besten geeignet angesehen, die in der Nähe eines Minimums liegen, aber möglichst nicht knapp an den Punkt grenzen, an dem ein deutlicher Anstieg der Iterationszahlen zu beobachten ist.
Das sind für die Parameter $(lfil, \epsilon_{drop})$ (siehe Abschnitt 7.3) für TAU die Werte (1;0,01), für TRACE-THD (2;0,01) und für TRACE-UHBR (2;0,0005).

Die Wahl dieser Einstellungen der unvollständigen LU-Zerlegungen der Diagonalblöcke zur Block-Jacobi-ILU-Vorkonditionierung wurde stichprobenartig auf unterschiedlich vielen Prozessen überprüft: In den hier betrachteten Fällen sind die gleichen Parameter anscheinend bei der Verwendung unterschiedlich vieler Prozesse geeignet.



(a) TAU-Testfall



(b) TRACE-Testfall THD

Abbildung 6: Rechnungen mit unterschiedlichen ILU-Parametern auf 12 Kernen mit Block-Jacobi-ILU-Vorkonditionierung

Vorkonditionierung	ILU Dauer [s]	Iterationen	Gesamtdauer [s]
Block-Jacobi-ILU	**2,2**	**60**	**6,77**
DSC mit GMRES (0)	2,4	60	10,64
DSC mit GMRES (1)	2,4	40	9,14
DSC mit GMRES (2)	2,4	33	8,55
DSC mit GMRES (3)	**2,4**	**30**	**8,43**
DSC mit GMRES (4)	2,4	29	8,71
DSC mit GMRES (5)	2,4	29	9,08
DSC mit GMRES (6)	2,4	29	9,50
DSC mit GMRES (7)	2,4	29	10,18
DSC mit BICGSTAB (0)	2,4	60	11,49
DSC mit BICGSTAB (1)	**2,4**	**34**	**8,73**
DSC mit BICGSTAB (2)	2,4	32	9,27
DSC mit BICGSTAB (3)	2,4	46	13,41
DSC mit BICGSTAB (4)	2,4	45	13,82
DSC mit BICGSTAB (5)	2,4	51	16,73
DSC mit BICGSTAB (6)	2,4	35	13,79
DSC mit BICGSTAB (7)	2,4	55	20,32

Tabelle 3: Unterschiedliche Möglichkeiten zur Lösung des Schur-Komplement-Systems bei der DSC-Vorkonditionierung, in Klammern ist jeweils die Anzahl der verwendeten inneren Iterationen angegeben, die besten Ergebnisse sind fett markiert (TAU-Testfall auf 12 Kernen)

Für alle folgenden Untersuchungen (auch mit anderen als Block-Jacobi-ILU-Vorkonditionierungen) werden daher die Parameter der unvollständigen LU-Zerlegungen wie oben festgelegt.

8.3. Wahl des Verfahrens zur Lösung des Schur-Komplement-Systems

Bei den in dieser Arbeit betrachteten Schur-Komplement-Vorkonditionierungen kann man verschiedene iterative Verfahren zur Approximation der Lösung des (selbst ebenfalls approximierten) Schur-Komplement-Systems verwenden.
Hier werden daher die Ergebnisse der DSC-Vorkonditionierung mit einigen GMRES-Iterationen und einigen BICGSTAB-Iterationen des Schur-Komplement-Systems gegenübergestellt. Tabelle 3 zeigt beispielhaft die Ergebnisse des TAU-Testfalles auf 12 Prozessorkernen dazu.

Bei 0 Iterationen des GMRES- und des BICGSTAB-Verfahrens ist die DSC-Vorkonditionierung formal identisch zur Block-Jacobi-ILU-Vorkonditionierung. In der Implementierung hier ergeben sich jedoch zwei Unterschiede:
Erstens muss das Gleichungssystem zur DSC-Vorkonditionierung in einer bestimmtem Form vorliegen, das heißt die Unbekannten x_i auf jedem Prozess müssen vor die Unbekannten y_i sortiert werden. Dadurch kann das hier verwendete Verfahren zur Permuta-

tion der Matrix zur LU-Fill-In-Reduzierung nur auf einzelne Teile der Diagonalblöcke angewendet werden. Deshalb ergeben sich in diesem Fall größere unvollständige LU-Zerlegungen und ein erhöhter Aufwand zur Bestimmung dieser Zerlegungen.
Zweitens wird auch bei 0 vorgegebenen Iterationen eine Matrix-Vektor-Multiplikation zur Bestimmung des Residuums durchgeführt, was den deutlich höheren Aufwand gegenüber der Block-Jacobi-ILU-Vorkonditionierung erklärt.

Erhöht man schrittweise die Zahl der Iterationen des Verfahrens für das Schur-Komplement-System, kann man bei der GMRES-Variante im Prinzip Folgendes beobachten: Mit jedem Schritt steigt der Aufwand jeder einzelnen vorkonditionierten Iteration des äußeren FGMRES-Verfahrens, wobei die Anzahl der benötigten äußeren Iterationen bis zu einem gewissen Punkt sinkt. Dies entspricht den Erwartungen, da das Schur-Komplement-System selbst ebenfalls nur approximiert wird und daher zur Lösung des exakten Systems ein von der Anzahl der Iterationen unabhängiger, zusätzlicher Approximationsfehler hinzukommt.
Zu den obigen Ergebnissen mit dem BICGSTAB-Verfahren lässt sich Folgendes sagen: Zunächst sei hier angemerkt, dass eine BICGSTAB-Iteration vom Aufwand grob zwei GMRES-Iterationen entspricht, da dort in jeder Iteration zwei Matrix-Vektor-Multiplikationen benötigt werden.
Die Ergebnisse mit 1 und 2 Iterationen sind mit denen des GMRES-Verfahrens vergleichbar. Danach steigt in dem hier betrachteten Fall die benötigte Zahl der äußeren Iterationen wieder leicht an.
Insgesamt fällt auf, dass die Zahl der benötigten äußeren Iterationen in diesem Fall deutlich unregelmäßiger von der Zahl der in der Vorkonditionierung durchgeführten Iterationen abhängt als bei der Verwendung des GMRES-Verfahrens. Dies lässt sich dadurch erklären, dass das GMRES-Verfahren auf einer Minimierung des Residuums beruht und deshalb ein in diesem Sinne optimales Ergebnis bestimmt, was beim BICGSTAB-Verfahren nicht der Fall ist. Bei letzterem Verfahren kann man auch in anderen Fällen eine unregelmäßigere Konvergenz gemessen in der Norm des Residuums beobachten.
Weitere Überlegungen zur Verwendung von GMRES und BICGSTAB als inneres Verfahren einer GMRES⋆-Iteration findet man in [13].

Hier wird die Anzahl der inneren Iterationen für das approximierte Schur-Komplement-System als Parameter fest vorgegeben. Des Weiteren liegt in den betrachteten Fällen aus der Strömungstechnik der optimale Wert nur bei einigen wenigen Iterationen. In diesem Zusammenhang ist hier die Verwendung des GMRES-Verfahrens vorzuziehen, da man insbesondere für ähnliche Werte des Parameters für die Iterationszahl ähnliche Ergebnisse erwarten kann. Bei der Verwendung des BICGSTAB-Verfahrens ist dies nicht der Fall, sodass die Einstellung einer geeigneten Iterationszahl deutlich erschwert wird. In allen weiteren Untersuchungen wird daher zur inneren Iteration der Vorkonditionierung das GMRES-Verfahren eingesetzt.

8.4. Skalierung der Verfahren auf unterschiedlich vielen Prozessen

Wie aus Tabelle 3 ersichtlich ist, ergibt sich aus der Verwendung der Schur-Komplement-Iteration für den TAU-Testfall auf 12 Prozessorkernen kein Vorteil bei dem hier

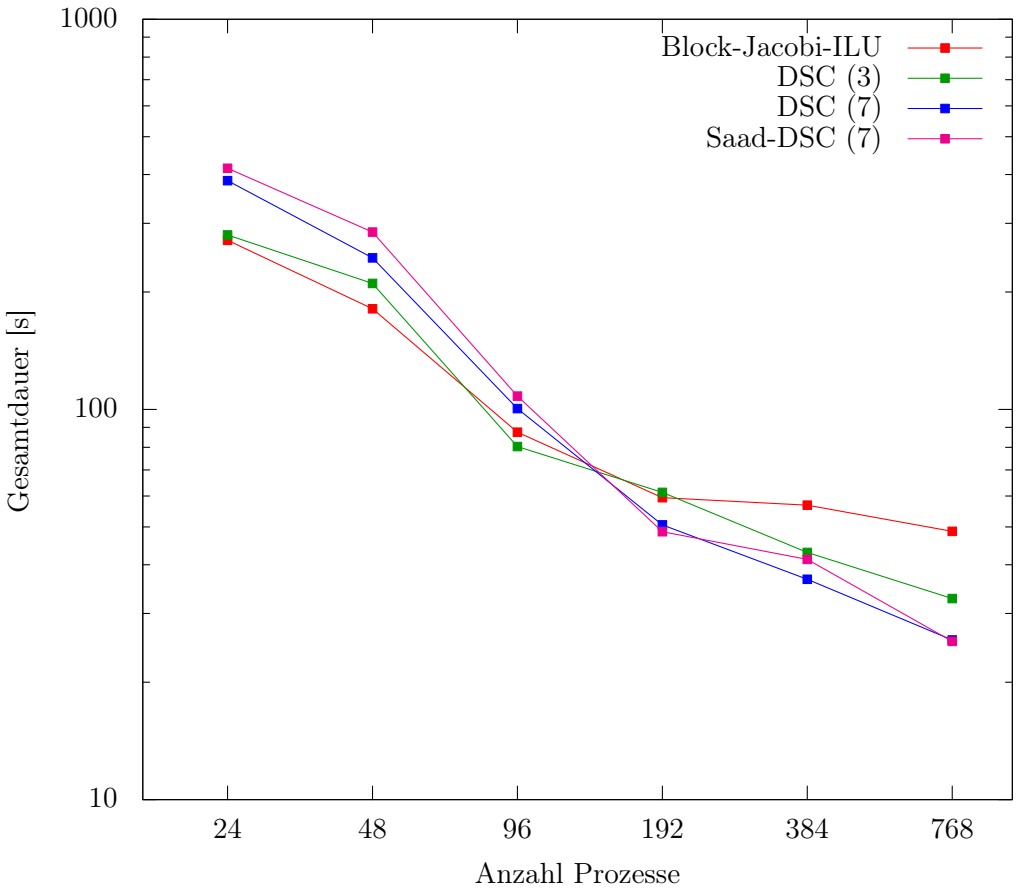


Abbildung 7: Rechenzeiten auf unterschiedlich vielen Kernen bei der Lösung des TRACE-Testfalls UHBR (logarithmisch aufgetragen)

verwendeten System. Vergleichbare Ergebnisse erhält man auch mit dem TRACE-THD-Testfall. Auf einem anderen System kann dies anders sein; das hängt insbesondere auch davon ab, wie lange jeweils die Bearbeitung der Rechen- und Kommunikationsschritte auf dem System dauert.[8] Um das Verhalten der Verfahren auf vielen Prozessorkernen zu untersuchen, wird deshalb im Folgenden der größere TRACE-UHBR-Testfall verwendet. Die benötigte Gesamtrechenzeit zur Lösung des Gleichungssystems mit Block-Jacobi-ILU- und DSC-Vorkonditionierung mit 3 und 7 Iterationen des Schur-Komplement-Systes wird in Abbildung 7 grafisch dargestellt. Weitere Informationen dazu findet man in Tabelle 4.

Zum Verständnis werden hier zunächst die Ergebnisse der Block-Jacobi-ILU-Vorkonditionierung betrachtet:
Auf je mehr Prozesse das Gleichungssystem verteilt wird, desto mehr Iterationen des

[8]Bei Tests auf einer Workstation des DLR ergab sich beispielsweise bei 8 Prozessen eine kleinere Gesamtdauer zur Lösung des TAU-Testfalls mit der DSC-Vorkonditionierung als mit der Block-Jacobi-ILU-Vorkonditionierung.

Vorkonditionierung	Prozesse	ILU Dauer [s]	Iterationen	Gesamtdauer [s]
Block-Jacobi-ILU	24	38,4	228	271,5
Block-Jacobi-ILU	48	20,1	294	181,2
Block-Jacobi-ILU	96	8,4	271	87,4
Block-Jacobi-ILU	192	3,6	360	59,4
Block-Jacobi-ILU	384	1,9	700	56,9
Block-Jacobi-ILU	768	0,9	1100	48,7
DSC (3)	24	40,0	100	280,0
DSC (3)	48	20,3	117	210,4
DSC (3)	96	8,6	97	80,4
DSC (3)	192	3,7	146	61,3
DSC (3)	384	2,1	173	43,0
DSC (3)	768	1,0	218	32,8
DSC (7)	24	40,0	95	385,9
DSC (7)	48	20,3	87	244,6
DSC (7)	96	8,4	76	100,5
DSC (7)	192	3,8	72	50,6
DSC (7)	384	2,1	85	36,7
DSC (7)	768	1,0	96	25,7
Saad-DSC (7)	24	40,0	100	415,2
Saad-DSC (7)	48	20,5	89	284,8
Saad-DSC (7)	96	8,2	80	108,2
Saad-DSC (7)	192	3,5	70	48,6
Saad-DSC (7)	384	2,0	80	41,3
Saad-DSC (7)	768	1,0	85	25,5
DSC (10)	768	1,0	73	24,6
DSC (20)	768	1,0	44	26,9
Quasi-exakt-DSC (10)	768	1,0	74	34,1
Quasi-exakt-DSC (20)	768	1,0	37	32,4

Tabelle 4: Rechenzeiten auf unterschiedlich vielen Kernen bei der Lösung des TRACE-Testfalls UHBR

Prozesse	$\min\left\{\frac{n_{y_i}}{n_{x_i}+n_{y_i}}\right\}$	$\max\left\{\frac{n_{y_i}}{n_{x_i}+n_{y_i}}\right\}$	$\frac{n_y}{n_x+n_y}$
24	19%	45%	25%
48	24%	58%	33%
96	25%	96%	41%
192	28%	100%	51%
384	36%	100%	61%
768	41%	100%	72%

Tabelle 5: Dimension des Schur-Komplement-Systems auf unterschiedlich vielen Prozessen im Verhältnis zum ursprünglichen Gleichungssystem

vorkonditionierten FGMRES-Verfahrens werden benötigt. Dies lässt sich dadurch erklären, dass die auf den einzelnen Prozessen gespeicherten Diagonalblöcke der Matrix durch die Aufteilung kleiner werden und auf mehr Prozessen ein größerer Teil der Matrix bei der Vorkonditionierung unberücksichtigt bleibt.
Wird die Zahl der verwendeten Prozessorkerne verdoppelt, können die unvollständigen LU-Zerlegungen der Diagonalblöcke mehr als zweimal so schnell berechnet werden. Auch die Zeit zur Berechnung der einzelnen Iterationen wird deutlich kleiner. Da jedoch die Anzahl der benötigten Iterationen steigt, sinkt die Gesamtrechenzeit bei der Verwendung von mehr als 192 Kernen nur noch geringfügig.

Bei der DSC-Vorkonditionierung verbessert sich in der Regel durch jeden Iterationsschritt des Schur-Komplement-Systems die Qualität der Vorkonditionierung, da der Einfluss der Matrixeinträge außerhalb der Diagonalblöcke auf den einzelnen Prozessen berücksichtigt wird. Dadurch sinkt die Zahl der benötigten Iterationen des äußeren Verfahrens.
Auf wenigen Prozessen (24 und 48) ist jedoch der zusätzliche Aufwand der Vorkonditionierung so groß, dass sich die benötigte Gesamtrechenzeit vergrößert. Erhöht man die Zahl der verwendeten Prozesse, sinkt die Qualität der Block-Jacobi-ILU-Vorkonditionierung, und es überwiegt der an äußeren Iterationsschritten eingesparte Aufwand.
Betrachtet man die optimale Anzahl an Schur-Komplement-Iterationen auf unterschiedlich vielen Prozessen für diesen Testfall, erhält man folgendes Resultat: Bei wenigen Prozessen (24 und 48) liegt sie bei Null, steigt ab einem bestimmten Punkt (96 Prozesse) langsam auf einige wenige Iterationen an (ungefähr 10 bei 768 Prozessen, siehe Anhang B).
Durch die DSC-Vorkonditionierung halbiert sich auf 768 Kernen ungefähr die Gesamtrechenzeit (gegenüber der Block-Jacobi-ILU-Vorkonditionierung).

Da die Dimension des Schur-Komplement-Systems bei der Rechnung auf mehr Prozessen höher ist (siehe Tabelle 5), steigt auch der Aufwand der DSC-Vorkonditionierung. Bei 768 Kernen liegt diese bei ungefähr 3/4 der Dimension des ursprünglichen Gleichungssystems. Das ist offensichtlich ein Nachteil der DSC-Vorkonditionierung bezüglich der Skalierbarkeit des Verfahrens.
Aus diesem Grunde könnte man annehmen, dass ab einer bestimmten Anzahl an Prozessen die Gesamtrechenzeit bei der Verwendung der DSC-Vorkonditionierung im Vergleich

zur Rechenzeit der Block-Jacobi-ILU-Vorkonditionierung wieder steigt, beziehungsweise die optimale Zahl an Iterationen des Schur-Komplement-Systems auf vielen Kernen sinkt. Das lässt sich hier jedoch nicht beobachten.
Für den Grenzfall, dass die Dimension des Schur-Komplement-Systems der des ursprünglichen Systems entspricht, erhält man eine Form der GMRES-Vorkonditionierung, das wäre hier eine äußere FGMRES-Iteration, die mit einem GMRES-Verfahren (mit Block-Jacobi-ILU-Vorkonditionierung) vorkonditioniert wird.

Die Tabelle 5 zeigt noch ein weiteres Problem der hier verwendeten Implementierung der DSC-Vorkonditionierung bezüglich des Load-Balancing:
Das ursprüngliche Gleichungssystem wird zwar gleichmäßig auf alle Prozesse aufgeteilt (also $n_{x_i} + n_{y_i}$ ist für alle Prozesse i ähnlich), diese Verteilung sorgt jedoch nicht dafür, dass dies auch für das Schur-Komplement-System gilt. Insbesondere existieren auf mehr als 192 Kernen Prozesse, auf denen alle Unbekannten von Unbekannten anderer Prozesse abhängen (das heißt, es gibt keine inneren Freiheitsgrade, also $n_{x_i} = 0$), und andere Prozesse, bei denen der Anteil der Variablen des Schur-Komplement-Systems unter 50% liegt. Dadurch müssen Prozesse bei der Berechnung der DSC-Vorkonditionierung gegebenenfalls länger aufeinander warten.

Auf 768 Kernen sinkt durch die Verwendung der quasi-exakten-DSC-Vorkonditionierung mit 20 Iterationen des Schur-Komplement-Systems zwar die benötigte Anzahl an äußeren Iterationen leicht, der Gesamtaufwand des Lösungsverfahrens ist jedoch höher als bei der DSC-Vorkonditionierung (siehe Tabelle 4).

Zusätzlich sind in Abbildung 7 und in Tabelle 4 zum Vergleich die Ergebnisse der hier als Saad-DSC-Vorkonditinierung bezeichneten Variante eingetragen. Man sieht deutlich, dass der Aufwand bei dieser Vorkonditionierung auf wenigen Prozessen (bis 96) höher ist als bei gleichen Einstellungen der DSC-Vorkonditionierung. Bei der Verwendung vieler Prozesse (ab 192) sind die Ergebnisse jedoch vergleichbar. Deshalb kann man vermuten, dass der Aufwand der gesamten Rechnung mit der hier beschriebene DSC-Vorkonditionierung in den Fällen geringer ist als bei der Saad-DSC Variante, in denen die optimale Zahl an Iterationen des Schur-Komplement-Systems klein ist.

GMRES Iterationen der Vorkond.	0	1	2	3	4	5	6	7
Äußere Iterationen mit DSC	800	478	200	173	143	115	94	85
Äußere Iterationen mit Saad-DSC	>2000	270	351	143	136	94	91	80

Tabelle 6: Iterationszahlen des äußeren FGMRES-Verfahrens beim UHBR-Testfall auf 384 Kernen mit DSC- und Saad-DSC-Vorkonditionierung mit unterschiedlich vielen Schur-Komplement-Iterationen

Tabelle 6 zeigt zur genaueren Gegenüberstellung der beiden Methoden anhand eines Beispiels die benötigten Iterationszahlen des äußeren Verfahrens mit DSC- und Saad-DSC-Vorkonditionierung. Man sieht, dass die zusätzliche Block-Jacobi-Iteration bei der Saad-DSC-Methode in manchen Fällen zu mehr äußeren Iterationen und in anderen zu weniger äußeren Iterationen führt. Der Aufwand der Saad-DSC-Methode mit k Iterationen ist auf vielen Kernen mit der DSC-Methode mit $k + 1$ Iterationen vergleichbar. In den

hier betrachteten Fällen verhält sich die Anzahl der benötigten äußeren Iterationen bei der Saad-DSC-Vorkonditionierung unregelmäßiger (bezüglich der vorgegebenen Anzahl an inneren Iterationen) als bei der DSC-Vorkonditionierung. Damit ist hier zur einfacheren Wahl der Parameter das DSC-Verfahren vorzuziehen (wie im Vergleich zwischen GMRES- und BICGSTAB-Verfahren zur Lösung des Schur-Komplent-Systems oben).

Da für diesen letzten Abschnitt nur die Ergebnisse eines Testfalles auf einem System betrachtet wurden, stellt sich die Frage, ob die hier dargelegten Beobachtungen auch auf die Lösung anderer linearer Gleichungssysteme aus TAU und TRACE oder anderen Anwendungen der Strömungstechnik übertragen werden können.
Für viele Phänomene konnten jedoch Erklärungen gefunden werden, die auch in anderen Fällen gültig sind. Daher kann man erwarten, dass das grundsätzliche Verhalten eines FGMRES-Verfahrens mit Block-Jacobi-ILU- und mit DSC-Vorkonditionierung auch bei der parallelen Lösung anderer linearer Probleme ähnlich ist. Die benötigte Zahl an FGMRES-Iterationen zum Erreichen einer gewissen Genauigkeit bei der Block-Jacobi-ILU-Vorkonditionierung steigt, wenn das Gleichungssystem auf mehr Prozesse verteilt wird; mit der DSC-Vorkonditionierung kann die Zahl der äußeren Iterationen gesenkt werden. In welchen Fällen dadurch der Gesamtaufwand sinkt und bei wie vielen Prozessen welche Zahl an Schur-Komplement-Iterationen optimal ist, lässt sich jedoch nicht allgemein angeben.

9. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit konnte die DSC-Bibliothek des DLR zur parallelen Lösung großer, dünnbesetzter linearer Gleichungssysteme in einigen Punkten weiterentwickelt werden. Insbesondere die mathematische Beschreibung der verwendeten Schur-Komplement-Methode zur Vorkonditionierung und die Gegenüberstellung von dieser und anderen Methoden führten zu Verbesserungen.

Zusammenfassend lässt sich sagen, dass Krylov-Unterraum-Verfahren zur Lösung der hier betrachteten Gleichungssysteme aus der Strömungstechnik auf Parallelrechnern geeignet sind. Die Vorkonditionierung der Gleichungssysteme trägt wesentlich zum Erfolg dieser Verfahren bei.
Mittels unvollständiger LU-Zerlegungen (ILU) lassen sich gute Vorkonditionierer konstruieren; Zerlegungen der ganzen Koeffizientenmatrix sind jedoch schwer parallelisierbar.
Bis zu einem gewissen Punkt erhält man in den untersuchten Fällen einen erfolgreichen Vorkonditionierer durch eine unvollständige LU-Zerlegung der Diagonalblöcke, die jeweils unabhängig voneinander behandelt werden können (Block-Jacobi-ILU-Vorkonditionierung). Bei der Aufteilung der Matrix auf viele Prozesse (und ebenso viele Diagonalblöcke) sinkt jedoch die Qualität dieses Vorkonditionierung deutlich.
Ein Ansatz zur Verbesserung dieser Vorkonditionierung liegt in der Berücksichtigung der Matrixeinträge außerhalb der Diagonalblöcke durch ein weiteres iteratives Verfahren zu berücksichtigen. Dies führt zu Schur-Komplement-Methoden. Auf vielen Prozessen ergibt sich dadurch eine deutlich bessere Vorkonditionierung; in den hier untersuchten Fällen beschleunigt sich die Lösung des Gleichungssystems um einen Faktor 2. Die Zunahme des Aufwandes der Vorkonditionierung bei einer Erhöhung der Anzahl der Prozesse begrenzt jedoch die Skalierung der Methode auf einem Parallelrechner.

Bei der in dieser Arbeit verwendeten Methode zur unvollständigen LU-Zerlegung könnten noch zwei Punkte verbessert werden:
Zum Einen könnten bessere Permutationen der Matrix zur Fill-In-Reduzierung der LU-Zerlegung der Diagonalblöcke bei Verwendung der Schur-Komplement-Vorkonditionierung berechnet werden.
Zum Anderen ermöglicht die Crout-Formulierung der unvollständigen LU-Zerlegung weitere, interessante Kriterien zur Bestimmung der Struktur der Nichtnulleinträge der Zerlegung, die beispielsweise auf einer Abschätzung der Norm der Inversen der Zerlegung beruhen (siehe [10]). Dadurch könnte die Qualität der unvollständigen LU-Zerlegung zur Vorkonditionierung bei gleichem Aufwand steigen.

Dünnbesetzte approximierte Inverse (*sparse approximated inverse*, SPAI) bilden eine weitere Klasse von Vorkonditionierungen, die sich gut parallelisieren lässt, hier aber nicht betrachtet wurden. Diese Verfahren lassen sich insbesondere auch mit Schur-Komplement-Methoden kombinieren (vergleiche [11]).

Ein grundlegendes Problem der in dieser Arbeit vorgestellten Schur-Komplement-Vorkonditionierung besteht darin, dass sich die Dimension des Schur-Komplement-Systems mit der Zahl der Prozesse erhöht und sich auf vielen Prozessen der Dimension des ursprünglichen Problems angleicht. Dadurch steigt der Aufwand dieser Vorkonditionie-

rung.
Da in den hier betrachteten Gleichungssystemen die Unbekannten jeweils von einer großen Zahl anderer Unbekannten abhängen (Abhängigkeitsbereich der Finite-Volumen-Diskretisierung von zwei Zellen je Richtung in TRACE), tritt dies bereits bei einer Zahl von Prozessen auf, die für die Anwendungen interessant sind. *Hybride* Parallelisierung könnte dieses Problem deutlich mindern. Im Folgenden wird diese Parallelisierungstechnik kurz erläutert. Das eingesetzte numerische Verfahren ändert sich dadurch jedoch nicht.

9.1. Hybride Parallelisierung der Schur-Komplement-Methode

In dieser Arbeit wurde vereinfacht davon ausgegangen, dass jeder Prozess Zugriff auf einen von ihm gespeicherten Teil des Gleichungssystems hat, einen seriellen Algorithmus ausführt und Daten zwischen Prozessen ausgetauscht werden können (*Distributed Memory* Rechnerarchitektur).
Das System, auf dem die Tests durchgeführt wurden, ist so aufgebaut, dass mehrere Prozessorkerne (hier 12) auf einen gemeinsamen Speicher zugreifen können. Dadurch kann man jeweils einem Satz von Kernen mit Zugriff auf den gemeinsamen Speicher (hier als Knoten bezeichnet) einen Teil des Gleichungssystems zuweisen. Bei der Schur-Komplement-Methode führt dann jeder Knoten die Berechnungen durch, die bisher von einem Kern auf dem Teildatensatz durchgeführt wurden.
Bei gleicher Anzahl an verwendeten Kernen muss das Gleichungssystem in weniger Teile aufgeteilt werden (hier beispielsweise um einen Faktor 12 weniger) und die Dimension des Schur-Komplement-Systems kann entsprechend verringert werden.
Man muss jedoch berücksichtigen, dass jeder Teil des Gleichungssystems nun von mehreren Kernen eines Knotens bearbeitet wird. Die verschiedenen eingesetzten Algorithmen (wie beispielsweise die Berechnung der unvollständigen LU-Zerlegungen der Diagonalblöcke), die bisher seriell durchgeführt wurden, müssen nun zusätzlich parallelisiert werden. Die Kerne eines Knotens können jedoch gemeinsam auf die Daten zugreifen (*Shared Memory* Rechnerarchitektur). Dies vereinfacht die Parallelisierung.

A. Algorithmen

Der folgende Algorithmus stammt aus [10] und wurde zur Rechnung mit Matrixblöcken erweitert.

Algorithmus 11. Block ILU Crout

Eingabe: $A \in \mathbb{C}^{(n_b n)\times(n_b n)}$ mit $A_{i,j} \in \mathbb{C}^{n_b \times n_b}$

1: $\tilde{L} \leftarrow 0, \tilde{U} \leftarrow 0$
2: **for** $k = 1 \rightarrow n$ **do**
3: Initialisiere Buffer für nächste Zeile: $Z_{k:n} \leftarrow A_{k,k:n}$
4: **for** $\{i | 1 \leq i < k \wedge \tilde{L}_{k,i} \neq 0\}$ **do**
5: **for** $j = k \rightarrow n$ **do**
6: $Z_j \leftarrow Z_j - \tilde{L}_{k,i}\, \tilde{U}_{i,j}$
7: **end for**
8: **end for**
9: Verwerfe nicht benötigte Blöcke in $Z_{k:n}$
10: $\tilde{U}_{k,k:n} \leftarrow Z_{k:n}$
11: Initialisiere Buffer für nächste Spalte: $W_{k+1:n} \leftarrow A_{k+1:n,k}$
12: **for** $\{i | 1 \leq i < k \wedge \tilde{U}_{i,k} \neq 0\}$ **do**
13: **for** $j = k+1 \rightarrow n$ **do**
14: $W_j \leftarrow W_j - \tilde{L}_{j,i}\, \tilde{U}_{i,k}$
15: **end for**
16: **end for**
17: Verwerfe nicht benötigte Blöcke in $W_{k+1:n}$
18: $\tilde{L}_{k,k} \leftarrow I$
19: $\tilde{L}_{k+1:n,k} \leftarrow W_{k+1:n}\, \tilde{U}_{k,k}^{-1}$
20: **end for**

Ausgabe: $\tilde{L}, \tilde{U}$

In Schritt 9 und 17 werden die Blöcke verworfen (also auf Null gesetzt), deren Norm $\|Z_j\|$ bzw. $\|W_j\|$ kleiner als $(\epsilon_{drop} \sum_{l=k}^{n} \|A_{k,l}\|)$ bzw. $(\epsilon_{drop} \sum_{l=k+1}^{n} \|A_{l,k}\|)$ ist. In einem zweiten Schritt werden jeweils alle bis auf die $lfil$ größten Blöcke in Z bzw. W auf Null gesetzt.

B. Ergebnisse

Erläuterungen

Alle Rechnungen wurden (größtenteils während der Testphase) auf dem RWTH-Bull-Cluster durchgeführt, das 2011 neu eingerichtet wurde.
Hardware: RWTH Bull Cluster, mit 2 Intel Westmere X5675 CPUs pro Node mit jeweils 6 Kernen a 3.06GHz
Software: Intel 12.1 Compiler, OpenMPI 1.5.3
FORTRAN-Compiler Flags: `-O3 -r8 -i4 -std03 -mkl=sequantial`
Die Rechnungen wurden in doppelter Genauigkeit durchgeführt (*double precision*).
Um den tatsächlichen Fehler bestimmen zu können, wurde die rechte Seite des Gleichungssystems so gewählt, dass die Lösung der Vektor $x = (1, \ldots, 1)^T$ ist.
Zur Lösung des Gleichungssystems wurde ein FGMRES(k)-Verfahren eingesetzt. Die Basisvektoren des Krylov-Unterraumes wurden mit dem klassischen Gram-Schidt-Verfahren (*classical gram-schmidt*, CGS) oder einer iterierten Version des klassischen Gram-Schidt-Verfahrens (*iterated classical gram-schmidt*, ICGS) orthogonalisiert. Das ICGS-Verfahren entspricht formal zwei Durchläufen des CGS-Verfahrens, ist jedoch stabiler.
Die FGMRES-Iteration wurde jeweils abgebrochen, sobald eine Reduktion des Residuums um einen Faktor 10^{-5} erreicht wurde (oder eine vorgegebene maximale Zahl an Iterationen). Die Dimension m des betrachteten Unterraumes bei der FGMRES(m)-Iteration wurde hier bei den Testfällen TAU und TRACE-THD auf 60 gesetzt, bei dem Testfall TRACE-UHBR auf 100.
Zur Ermittlung der Rechenzeit wurden alle Rechnungen mehrmals durchgeführt (5 bis 10 mal) und jeweils die kürzeste Ausführungszeit übernommen.

TAU-Testfall bei Blockgröße 1 und äußerer FGMRES(60)-Iteration mit CGS

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
12	1	1e-3	Block-Jacobi-ILU	-	-	-	7,01	1,64	1,80	17,90	77	4,35e-7	1,14e-4	12,98	37,88
12	1	5e-3	Block-Jacobi-ILU	-	-	-	6,95	1,05	1,25	8,28	85	4,61e-7	1,07e-4	11,48	26,71
12	1	1e-2	Block-Jacobi-ILU	-	-	-	7,21	0,75	0,94	4,96	407	4,64e-7	9,65e-5	47,43	59,59
12	1	5e-2	Block-Jacobi-ILU	-	-	-	7,25	0,22	0,31	1,40	601	1,24e+0	4,22e+1	49,89	58,54
12	1	1e-1	Block-Jacobi-ILU	-	-	-	7,22	0,11	0,13	0,90	601	1,48e+0	6,81e+1	45,33	53,45
12	2	1e-3	Block-Jacobi-ILU	-	-	-	7,34	2,23	2,71	44,49	72	4,96e-7	1,17e-4	15,51	67,35
12	2	5e-3	Block-Jacobi-ILU	-	-	-	6,96	1,12	1,40	10,51	88	4,58e-7	6,95e-5	12,45	29,93
12	2	1e-2	Block-Jacobi-ILU	-	-	-	6,93	0,77	1,03	5,57	601	1,35e-4	4,06e-2	72,45	84,95
12	2	5e-2	Block-Jacobi-ILU	-	-	-	7,07	0,22	0,31	1,41	601	9,98e-1	3,10e+0	50,37	58,84
12	2	1e-1	Block-Jacobi-ILU	-	-	-	6,97	0,11	0,13	0,90	601	3,08e+0	1,63e+2	46,00	53,88

TAU-Testfall bei Blockgröße 5 und äußerer FGMRES(60)-Iteration mit CGS

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
1	1	1e-3	Block-Jacobi-ILU	-	-	-	1,29	1,95	1,95	41,11	31	3,95e-7	7,34e-5	16,98	59,39
1	1	5e-3	Block-Jacobi-ILU	-	-	-	1,30	1,85	1,85	35,00	31	4,91e-1	1,08e-4	16,35	52,65
1	1	1e-2	Block-Jacobi-ILU	-	-	-	1,29	1,68	1,68	28,41	35	3,23e-7	4,87e-5	17,39	47,09
1	1	5e-2	Block-Jacobi-ILU	-	-	-	1,29	0,75	0,75	7,18	287	4,54e-7	3,08e-5	93,97	102,43
1	1	1e-1	Block-Jacobi-ILU	-	-	-	1,30	0,40	0,40	3,34	140	4,78e-7	6,70e-5	36,47	41,11
1	2	1e-3	Block-Jacobi-ILU	-	-	-	1,30	3,55	3,55	111,94	18	2,07e-7	3,11e-5	15,35	128,60
1	2	5e-3	Block-Jacobi-ILU	-	-	-	1,30	2,87	2,87	72,10	20	1,96e-7	3,52e-5	14,44	87,84
6	1	1e-3	Block-Jacobi-ILU	-	-	-	0,92	1,90	1,99	6,89	51	4,61e-7	7,35e-5	6,73	14,54
6	1	5e-3	Block-Jacobi-ILU	-	-	-	0,92	1,76	1,84	5,73	51	4,92e-7	8,82e-5	6,42	13,07
6	1	1e-2	Block-Jacobi-ILU	-	-	-	0,92	1,57	1,63	4,49	52	4,82e-7	1,21e-4	6,20	11,61
6	1	5e-2	Block-Jacobi-ILU	-	-	-	0,92	0,61	0,73	1,16	300	7,83e-7	2,55e-4	24,48	26,56
6	1	1e-1	Block-Jacobi-ILU	-	-	-	0,91	0,35	0,40	0,57	172	4,65e-7	6,98e-5	11,85	13,33
6	2	1e-3	Block-Jacobi-ILU	-	-	-	0,91	3,35	3,48	18,79	49	3,75e-7	1,02e-4	9,63	29,34
6	2	5e-3	Block-Jacobi-ILU	-	-	-	0,93	2,59	2,68	10,96	49	4,75e-7	8,01e-5	7,95	19,83
6	2	1e-2	Block-Jacobi-ILU	-	-	-	0,92	1,93	2,08	6,50	52	3,98e-7	8,14e-5	7,14	14,56
6	2	5e-2	Block-Jacobi-ILU	-	-	-	0,92	0,61	0,76	1,20	345	4,78e-7	2,16e-4	28,65	30,77
6	2	1e-1	Block-Jacobi-ILU	-	-	-	0,92	0,35	0,40	0,57	172	5,00e-7	7,89e-5	11,84	13,33
12	1	1e-3	Block-Jacobi-ILU	-	-	-	0,64	1,87	1,97	3,38	60	4,52e-7	2,02e-4	4,34	8,36
12	1	5e-3	Block-Jacobi-ILU	-	-	-	0,66	1,70	1,81	2,77	59	4,56e-7	1,63e-4	4,06	7,49
12	1	1e-2	Block-Jacobi-ILU	-	-	-	0,64	1,47	1,62	2,18	60	4,49e-7	1,34e-4	3,95	6,77
12	1	5e-2	Block-Jacobi-ILU	-	-	-	0,65	0,55	0,71	0,54	299	4,71e-7	2,61e-4	12,24	13,43
12	1	1e-1	Block-Jacobi-ILU	-	-	-	0,64	0,33	0,38	0,27	162	4,98e-7	2,11e-4	5,69	6,60
12	2	1e-3	Block-Jacobi-ILU	-	-	-	0,65	3,18	3,47	9,09	57	4,11e-7	1,82e-4	6,04	15,78

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
12	2	5e-3	Block-Jacobi-ILU	-	-	-	0,66	2,35	2,70	5,20	57	3,85e-7	1,32e-4	4,99	10,85
12	2	1e-2	Block-Jacobi-ILU	-	-	-	0,65	1,71	2,08	3,18	58	4,69e-7	1,79e-4	4,01	7,84
12	2	5e-2	Block-Jacobi-ILU	-	-	-	0,65	0,56	0,73	0,56	300	7,31e-7	4,89e-4	12,63	13,84
12	2	1e-1	Block-Jacobi-ILU	-	-	-	0,65	0,33	0,38	0,27	163	4,63e-7	1,93e-4	5,87	6,79
24	1	1e-2	Block-Jacobi-ILU	-	-	-	0,74	1,33	1,55	1,10	67	4,67e-7	1,08e-4	4,00	5,84
6	1	1e-2	DSC	BICGSTAB	0	-	0,91	1,58	1,68	4,61	54	4,62e-7	1,65e-4	8,44	13,97
6	1	1e-2	DSC	BICGSTAB	1	-	0,92	1,58	1,68	4,62	36	3,47e-7	3,55e-5	6,17	11,70
6	1	1e-2	DSC	BICGSTAB	2	-	0,92	1,58	1,68	4,63	37	3,21e-7	2,15e-5	6,85	12,40
6	1	1e-2	DSC	BICGSTAB	3	-	0,92	1,58	1,68	4,62	37	3,81e-7	1,89e-5	7,47	13,01
6	1	1e-2	DSC	BICGSTAB	4	-	0,92	1,58	1,68	4,62	73	4,45e-7	5,36e-5	13,62	19,16
6	1	1e-2	DSC	BICGSTAB	5	-	0,91	1,58	1,68	4,62	55	4,67e-7	7,31e-5	12,77	18,31
6	1	1e-2	DSC	BICGSTAB	6	-	0,92	1,58	1,68	4,62	67	4,56e-7	5,46e-5	14,53	20,07
6	1	1e-2	DSC	BICGSTAB	7	-	0,92	1,58	1,68	4,63	38	3,82e-7	1,78e-5	10,17	15,72
6	1	1e-2	DSC	GMRES	0	-	0,92	1,58	1,68	4,63	54	4,62e-7	1,65e-4	7,93	13,48
6	1	1e-2	DSC	GMRES	1	-	0,92	1,58	1,68	4,62	39	3,79e-7	1,13e-4	6,25	11,79
6	1	1e-2	DSC	GMRES	2	-	0,93	1,58	1,68	4,62	36	4,05e-7	9,99e-5	6,06	11,61
6	1	1e-2	DSC	GMRES	3	-	0,92	1,58	1,68	4,62	34	3,38e-7	4,32e-5	6,05	11,59
6	1	1e-2	DSC	GMRES	4	-	0,92	1,58	1,68	4,62	34	4,46e-7	2,40e-5	6,37	11,90
6	1	1e-2	DSC	GMRES	5	-	0,92	1,58	1,68	4,62	33	3,14e-7	2,07e-5	6,40	11,94
6	1	1e-2	DSC	GMRES	6	-	0,91	1,58	1,68	4,62	33	4,27e-7	2,64e-5	6,64	12,18
6	1	1e-2	DSC	GMRES	7	-	0,92	1,58	1,68	4,63	32	4,43e-7	2,51e-5	6,75	12,30
12	1	1e-2	DSC	BICGSTAB	0	-	0,67	1,47	1,63	2,40	60	6,51e-7	2,06e-4	8,42	11,49
12	1	1e-2	DSC	BICGSTAB	1	-	0,67	1,47	1,63	2,39	34	3,96e-7	8,45e-5	5,66	8,73
12	1	1e-2	DSC	BICGSTAB	2	-	0,66	1,47	1,63	2,40	32	3,87e-7	1,73e-5	6,21	9,27
12	1	1e-2	DSC	BICGSTAB	3	-	0,67	1,47	1,63	2,40	46	4,16e-7	3,64e-5	10,35	13,41
12	1	1e-2	DSC	BICGSTAB	4	-	0,67	1,47	1,63	2,40	45	2,95e-7	1,39e-5	10,75	13,82
12	1	1e-2	DSC	BICGSTAB	5	-	0,67	1,47	1,63	2,40	51	3,51e-7	1,43e-5	13,66	16,73
12	1	1e-2	DSC	BICGSTAB	6	-	0,67	1,47	1,63	2,40	35	4,31e-7	3,44e-5	10,71	13,79
12	1	1e-2	DSC	BICGSTAB	7	-	0,66	1,47	1,63	2,40	55	4,05e-7	5,46e-5	17,25	20,32
12	1	1e-2	DSC	GMRES	0	-	0,66	1,47	1,63	2,39	60	6,51e-7	2,06e-4	7,59	10,64
12	1	1e-2	DSC	GMRES	1	-	0,67	1,47	1,63	2,40	40	4,55e-7	5,43e-5	6,08	9,14
12	1	1e-2	DSC	GMRES	2	-	0,67	1,47	1,63	2,40	33	4,83e-7	9,05e-5	5,48	8,55
12	1	1e-2	DSC	GMRES	3	-	0,67	1,47	1,63	2,39	30	3,24e-7	4,93e-5	5,37	8,43
12	1	1e-2	DSC	GMRES	4	-	0,68	1,47	1,63	2,39	29	3,97e-7	2,38e-5	5,64	8,71
12	1	1e-2	DSC	GMRES	5	-	0,66	1,47	1,63	2,39	29	3,13e-7	3,42e-5	6,02	9,08
12	1	1e-2	DSC	GMRES	6	-	0,67	1,47	1,63	2,39	29	4,20e-7	5,39e-5	6,43	9,50
12	1	1e-2	DSC	GMRES	7	-	0,67	1,47	1,63	2,40	30	3,23e-7	4,65e-5	7,11	10,18
24	1	1e-2	DSC	GMRES	0	-	0,75	1,30	1,61	1,15	66	4,38e-7	1,36e-4	4,44	6,34
24	1	1e-2	DSC	GMRES	1	-	0,73	1,30	1,61	1,15	42	4,07e-7	9,78e-5	3,89	5,76
24	1	1e-2	DSC	GMRES	2	-	0,76	1,30	1,61	1,18	32	4,18e-7	9,20e-5	3,44	5,38
24	1	1e-2	DSC	GMRES	3	-	0,73	1,30	1,61	1,16	27	3,97e-7	7,79e-5	3,25	5,14
24	1	1e-2	DSC	GMRES	4	-	0,73	1,30	1,61	1,16	26	3,35e-7	4,88e-5	3,55	5,44

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
24	1	1e-2	DSC	GMRES	5	-	0,72	1,30	1,61	1,17	26	3,60e-7	4,68e-5	3,86	5,75
24	1	1e-2	DSC	GMRES	6	-	0,73	1,30	1,61	1,20	26	4,72e-7	7,25e-5	4,06	5,98
24	1	1e-2	DSC	GMRES	7	-	0,74	1,30	1,61	1,34	27	3,14e-7	3,93e-5	4,71	6,78
6	1	1e-2	Saad-DSC	GMRES	0	-	1,03	1,58	1,68	4,62	42	3,04e-7	2,14e-5	6,66	12,32
6	1	1e-2	Saad-DSC	GMRES	1	-	0,92	1,58	1,68	4,63	36	4,62e-7	3,06e-5	6,21	11,76
6	1	1e-2	Saad-DSC	GMRES	2	-	1,11	1,58	1,68	4,63	35	3,13e-7	2,33e-5	6,34	12,07
6	1	1e-2	Saad-DSC	GMRES	3	-	0,92	1,58	1,68	4,62	34	4,43e-7	2,80e-5	6,46	12,00
6	1	1e-2	Saad-DSC	GMRES	4	-	1,06	1,58	1,68	4,62	34	3,85e-7	2,13e-5	6,69	12,37
6	1	1e-2	Saad-DSC	GMRES	5	-	1,04	1,58	1,68	4,63	35	3,43e-7	3,72e-5	7,28	12,95
6	1	1e-2	Saad-DSC	GMRES	6	-	1,06	1,58	1,68	4,63	35	4,38e-7	2,24e-5	7,63	13,32
6	1	1e-2	Saad-DSC	GMRES	7	-	0,92	1,58	1,68	4,63	36	4,73e-7	2,47e-5	7,99	13,53
12	1	1e-2	Saad-DSC	GMRES	0	-	0,67	1,47	1,63	2,41	39	3,71e-7	3,79e-5	5,57	8,65
12	1	1e-2	Saad-DSC	GMRES	1	-	0,67	1,47	1,63	2,39	35	3,39e-7	7,20e-5	6,02	9,08
12	1	1e-2	Saad-DSC	GMRES	2	-	0,68	1,47	1,63	2,39	32	3,18e-7	2,16e-5	5,87	8,94
12	1	1e-2	Saad-DSC	GMRES	3	-	0,66	1,47	1,63	2,39	31	3,52e-7	2,21e-5	6,12	9,17
12	1	1e-2	Saad-DSC	GMRES	4	-	0,66	1,47	1,63	2,39	31	3,84e-7	3,18e-5	6,55	9,60
12	1	1e-2	Saad-DSC	GMRES	5	-	0,67	1,47	1,63	2,39	31	4,57e-7	5,31e-5	6,98	10,03
12	1	1e-2	Saad-DSC	GMRES	6	-	0,66	1,47	1,63	2,40	30	3,77e-7	1,96e-5	7,17	10,23
12	1	1e-2	Saad-DSC	GMRES	7	-	0,67	1,47	1,63	2,39	30	4,65e-7	3,73e-5	7,61	10,67
24	1	1e-2	Saad-DSC	GMRES	0	-	0,73	1,30	1,61	1,28	40	3,27e-7	2,48e-5	3,44	5,46
24	1	1e-2	Saad-DSC	GMRES	1	-	0,76	1,30	1,61	1,19	33	3,75e-7	8,40e-5	3,79	5,73
24	1	1e-2	Saad-DSC	GMRES	2	-	0,76	1,30	1,61	1,20	29	4,71e-7	1,21e-4	3,72	5,68
24	1	1e-2	Saad-DSC	GMRES	3	-	0,76	1,30	1,61	1,28	28	4,13e-7	4,55e-5	3,91	5,94
24	1	1e-2	Saad-DSC	GMRES	4	-	1,19	1,30	1,61	1,21	28	3,71e-7	2,23e-5	4,35	6,74
24	1	1e-2	Saad-DSC	GMRES	5	-	1,08	1,30	1,61	1,17	28	3,76e-7	2,08e-5	4,58	6,83
24	1	1e-2	Saad-DSC	GMRES	6	-	1,09	1,30	1,61	1,16	28	4,02e-7	2,13e-5	5,02	7,26
24	1	1e-2	Saad-DSC	GMRES	7	-	0,75	1,30	1,61	1,19	28	4,82e-7	2,68e-5	5,50	7,44

TRACE-THD-Testfall bei Blockgröße 5 und äußerer FGMRES(60)-Iteration mit CGS

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
1	1	1e-2	Block-Jacobi-ILU	-	-	-	0,53	1,87	1,87	11,98	84	4,89e-7	1,90e-5	33,04	45,56
1	2	1e-2	Block-Jacobi-ILU	-	-	-	0,53	3,20	3,20	27,05	40	4,12e-7	2,68e-5	22,35	49,93
6	1	1e-3	Block-Jacobi-ILU	-	-	-	0,41	1,80	1,89	2,24	97	4,34e-7	3,62e-5	7,89	10,55
6	1	5e-3	Block-Jacobi-ILU	-	-	-	0,41	1,77	1,87	2,03	96	4,78e-7	4,29e-5	7,73	10,17
6	1	1e-2	Block-Jacobi-ILU	-	-	-	0,41	1,73	1,83	1,87	98	4,40e-7	2,82e-5	7,83	10,11
6	1	5e-2	Block-Jacobi-ILU	-	-	-	0,41	1,25	1,34	1,02	288	4,66e-7	5,96e-5	20,33	21,76
6	2	1e-3	Block-Jacobi-ILU	-	-	-	0,41	3,32	3,46	5,77	60	6,96e-7	8,82e-5	7,19	13,36
6	2	5e-2	Block-Jacobi-ILU	-	-	-	0,42	1,31	1,45	1,12	284	4,97e-7	5,79e-5	20,83	22,37
12	1	1e-3	Block-Jacobi-ILU	-	-	-	0,28	1,78	1,88	1,19	109	4,72e-7	7,11e-5	7,89	9,36
12	1	5e-3	Block-Jacobi-ILU	-	-	-	0,28	1,74	1,86	1,09	110	4,42e-7	6,56e-5	7,93	9,29
12	1	1e-2	Block-Jacobi-ILU	-	-	-	0,27	1,69	1,83	1,00	111	4,52e-7	6,40e-5	7,92	9,19
12	1	5e-2	Block-Jacobi-ILU	-	-	-	0,28	1,16	1,33	0,53	420	6,51e-7	2,81e-4	26,21	27,01
12	1	1e-1	Block-Jacobi-ILU	-	-	-	0,27	0,78	0,98	0,33	601	1,25e+0	2,89e+1	33,51	34,11
12	2	1e-3	Block-Jacobi-ILU	-	-	-	0,28	3,25	3,45	3,02	82	4,93e-7	7,33e-5	8,35	11,65
12	2	5e-3	Block-Jacobi-ILU	-	-	-	0,28	2,96	3,27	2,57	82	4,99e-7	7,71e-5	8,02	10,87
12	2	1e-2	Block-Jacobi-ILU	-	-	-	0,28	2,63	3,02	2,17	85	4,51e-7	8,53e-5	7,91	10,36
12	2	5e-2	Block-Jacobi-ILU	-	-	-	0,28	1,19	1,44	0,57	358	4,91e-7	1,91e-4	22,80	23,66
12	2	1e-1	Block-Jacobi-ILU	-	-	-	0,28	0,79	1,15	0,42	601	1,48e+0	3,52e+1	33,81	34,52
12	4	1e-3	Block-Jacobi-ILU	-	-	-	0,27	5,46	6,08	8,31	79	4,86e-7	4,37e-5	11,97	20,55
12	4	5e-3	Block-Jacobi-ILU	-	-	-	0,29	4,13	5,11	5,93	80	4,27e-7	4,39e-5	10,48	16,71
12	4	1e-2	Block-Jacobi-ILU	-	-	-	0,29	3,14	4,02	3,77	82	4,64e-7	6,97e-5	9,10	13,15
12	4	5e-2	Block-Jacobi-ILU	-	-	-	0,28	1,19	1,44	0,57	358	4,86e-7	1,92e-4	22,96	23,81
12	4	1e-1	Block-Jacobi-ILU	-	-	-	0,27	0,79	1,41	0,66	601	2,61e+0	4,22e+2	35,80	36,73
24	1	1e-3	Block-Jacobi-ILU	-	-	-	0,40	1,76	1,85	0,65	106	4,98e-7	8,36e-5	4,99	6,04
24	1	5e-3	Block-Jacobi-ILU	-	-	-	0,43	1,72	1,81	0,62	107	4,36e-7	7,56e-5	4,77	5,82
24	1	1e-2	Block-Jacobi-ILU	-	-	-	0,41	1,65	1,79	0,50	106	4,93e-7	8,37e-5	4,43	5,34
24	1	5e-2	Block-Jacobi-ILU	-	-	-	0,41	1,01	1,34	0,27	229	4,55e-7	6,34e-5	8,68	9,36
24	2	1e-3	Block-Jacobi-ILU	-	-	-	0,44	3,17	3,33	1,42	87	4,64e-7	4,54e-5	5,32	7,18
24	2	5e-3	Block-Jacobi-ILU	-	-	-	0,41	2,77	3,17	1,28	87	4,85e-1	4,70e-5	5,27	6,97
24	2	1e-2	Block-Jacobi-ILU	-	-	-	0,47	2,38	2,95	1,26	87	4,88e-7	5,85e-5	5,00	6,73
24	2	5e-2	Block-Jacobi-ILU	-	-	-	0,44	1,02	1,41	0,37	230	4,58e-7	8,63e-5	8,21	9,01
6	1	1e-2	DSC	BICGSTAB	0	-	0,41	1,73	1,81	1,86	95	4,41e-7	2,38e-5	8,93	11,20
6	1	1e-2	DSC	BICGSTAB	1	-	0,41	1,73	1,81	1,86	70	4,45e-7	5,40e-5	8,49	10,76
6	1	1e-2	DSC	BICGSTAB	2	-	0,41	1,73	1,81	1,86	60	7,08e-7	8,91e-5	8,16	10,43
6	1	1e-2	DSC	BICGSTAB	3	-	0,41	1,73	1,81	1,86	54	4,93e-7	4,71e-5	8,01	10,28
6	1	1e-2	DSC	BICGSTAB	4	-	0,41	1,73	1,81	1,86	56	4,12e-7	4,38e-5	8,96	11,23
6	1	1e-2	DSC	BICGSTAB	5	-	0,42	1,73	1,81	1,86	55	4,79e-7	7,14e-5	9,64	11,92
6	1	1e-2	DSC	BICGSTAB	6	-	0,40	1,73	1,81	1,86	55	3,81e-7	4,49e-5	10,25	12,51
6	1	1e-2	DSC	BICGSTAB	7	-	0,41	1,73	1,81	1,86	56	4,04e-7	4,44e-5	11,21	13,47
6	1	1e-2	DSC	GMRES	0	-	0,41	1,73	1,81	1,86	95	4,41e-7	2,38e-5	8,90	11,17

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
6	1	1e-2	DSC	GMRES	1	-	0,41	1,73	1,81	1,86	70	4,45e-7	5,40e-5	8,52	10,79
6	1	1e-2	DSC	GMRES	2	-	0,41	1,73	1,81	1,86	60	7,08e-7	8,91e-5	8,15	10,42
6	1	1e-2	DSC	GMRES	3	-	0,40	1,73	1,81	1,86	54	4,93e-7	4,71e-5	8,02	10,29
6	1	1e-2	DSC	GMRES	4	-	0,41	1,73	1,81	1,86	56	4,12e-7	4,38e-5	9,06	11,33
6	1	1e-2	DSC	GMRES	5	-	0,43	1,73	1,81	1,86	55	4,79e-7	7,14e-5	9,70	11,99
6	1	1e-2	DSC	GMRES	6	-	0,41	1,73	1,81	1,86	55	3,81e-1	4,49e-5	10,38	12,64
6	1	1e-2	DSC	GMRES	7	-	0,41	1,73	1,81	1,86	56	4,04e-7	4,44e-5	11,24	13,51
12	1	1e-2	DSC	BICGSTAB	0	-	0,29	1,68	1,80	0,98	106	4,41e-7	1,09e-4	8,53	9,80
12	1	1e-2	DSC	BICGSTAB	1	-	0,28	1,68	1,80	0,98	74	4,85e-7	1,14e-4	8,18	9,45
12	1	1e-2	DSC	BICGSTAB	2	-	0,28	1,68	1,80	0,98	60	4,13e-7	1,07e-4	7,66	8,91
12	1	1e-2	DSC	BICGSTAB	3	-	0,28	1,68	1,80	0,98	56	4,26e-7	1,04e-4	7,94	9,21
12	1	1e-2	DSC	BICGSTAB	4	-	0,28	1,68	1,80	0,98	55	3,83e-7	9,96e-5	8,64	9,90
12	1	1e-2	DSC	BICGSTAB	5	-	0,28	1,68	1,80	0,98	53	4,48e-7	1,06e-4	9,22	10,48
12	1	1e-2	DSC	BICGSTAB	6	-	0,28	1,68	1,80	0,98	54	3,85e-7	8,70e-5	10,17	11,44
12	1	1e-2	DSC	BICGSTAB	7	-	0,29	1,68	1,80	0,98	54	3,46e-7	6,33e-5	11,04	12,32
12	1	1e-2	DSC	GMRES	0	-	0,28	1,68	1,80	0,98	106	4,41e-7	1,09e-4	8,83	10,09
12	1	1e-2	DSC	GMRES	1	-	0,28	1,68	1,80	0,98	74	4,85e-7	1,14e-4	8,18	9,44
12	1	1e-2	DSC	GMRES	2	-	0,30	1,68	1,80	0,98	60	4,13e-1	1,07e-4	7,65	8,93
12	1	1e-2	DSC	GMRES	3	-	0,27	1,68	1,80	0,98	56	4,26e-7	1,04e-4	7,95	9,20
12	1	1e-2	DSC	GMRES	4	-	0,29	1,68	1,80	0,98	55	3,83e-7	9,96e-5	8,69	9,96
12	1	1e-2	DSC	GMRES	5	-	0,28	1,68	1,80	0,98	53	4,48e-7	1,06e-4	9,26	10,53
12	1	1e-2	DSC	GMRES	6	-	0,29	1,68	1,80	0,98	54	3,85e-7	8,70e-5	10,16	11,43
12	1	1e-2	DSC	GMRES	7	-	0,27	1,68	1,80	0,98	54	3,46e-7	6,33e-5	11,05	12,30
24	1	1e-2	DSC	GMRES	0	-	0,42	1,64	1,77	0,71	109	4,68e-7	7,69e-5	5,54	6,67
24	1	1e-2	DSC	GMRES	1	-	0,42	1,64	1,77	0,48	76	4,28e-7	5,77e-5	5,56	6,46
24	1	1e-2	DSC	GMRES	2	-	0,42	1,64	1,77	0,53	57	4,09e-7	4,61e-5	5,04	5,98
24	1	1e-2	DSC	GMRES	3	-	0,42	1,64	1,77	0,61	51	3,89e-7	4,19e-5	5,21	6,24
24	1	1e-2	DSC	GMRES	4	-	0,41	1,64	1,77	0,54	49	3,24e-7	3,58e-5	5,42	6,37
24	1	1e-2	DSC	GMRES	5	-	0,40	1,64	1,77	0,53	48	3,71e-7	2,97e-5	5,98	6,91
24	1	1e-2	DSC	GMRES	6	-	0,43	1,64	1,77	0,48	47	4,98e-7	4,82e-5	6,46	7,36
24	1	1e-2	DSC	GMRES	7	-	0,41	1,64	1,77	0,51	47	4,13e-7	2,84e-5	6,69	7,62

TRACE-UHBR-Testfall bei Blockgröße 5 und äußerer FGMRES(100)-Iteration mit ICGS

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
24	2	5e-4	Block-Jacobi-ILU	-	-	-	2,80	2,75	3,58	38,40	228	4,75e-7	2,99e-4	230,26	271,46
48	2	1e-4	Block-Jacobi-ILU	-	-	-	2,03	2,94	3,83	24,04	294	4,74e-7	1,27e-3	162,10	188,17
48	2	5e-4	Block-Jacobi-ILU	-	-	-	2,02	2,65	3,69	20,12	294	4,75e-7	1,26e-3	159,02	181,16
48	2	1e-3	Block-Jacobi-ILU	-	-	-	1,99	2,37	3,59	17,14	294	4,74e-7	1,23e-3	159,60	178,74
48	2	5e-3	Block-Jacobi-ILU	-	-	-	2,00	1,26	3,20	8,46	387	4,82e-7	1,23e-3	188,59	199,05
48	2	1e-2	Block-Jacobi-ILU	-	-	-	1,89	0,79	2,89	7,06	1400	9,73e-7	2,00e-3	654,51	663,47
48	4	1e-4	Block-Jacobi-ILU	-	-	-	1,94	4,64	6,67	65,22	248	4,95e-7	8,97e-4	187,40	254,56
48	4	5e-4	Block-Jacobi-ILU	-	-	-	1,96	3,52	6,30	42,59	245	4,90e-7	9,87e-4	174,75	219,30
48	4	1e-3	Block-Jacobi-ILU	-	-	-	1,97	2,77	6,01	27,89	240	4,97e-7	9,72e-4	171,82	201,68
48	4	5e-3	Block-Jacobi-ILU	-	-	-	1,95	1,27	4,83	18,44	300	8,05e-7	1,23e-3	193,65	214,04
48	4	1e-2	Block-Jacobi-ILU	-	-	-	2,07	0,79	3,86	11,84	1300	7,52e-7	1,40e-3	677,57	691,48
48	8	1e-4	Block-Jacobi-ILU	-	-	-	2,05	5,64	10,88	119,82	221	4,80e-7	2,83e-4	228,01	349,88
48	8	5e-4	Block-Jacobi-ILU	-	-	-	1,84	3,61	9,75	73,56	218	5,00e-7	3,25e-4	211,34	286,75
48	8	1e-3	Block-Jacobi-ILU	-	-	-	1,95	2,78	9,08	65,02	218	4,87e-7	4,42e-4	203,82	270,80
48	8	5e-3	Block-Jacobi-ILU	-	-	-	1,92	1,27	5,96	29,68	300	5,65e-7	1,20e-3	203,05	234,64
48	8	1e-2	Block-Jacobi-ILU	-	-	-	2,26	0,79	4,10	13,96	1400	5,25e-7	1,55e-3	797,32	813,54
96	2	5e-4	Block-Jacobi-ILU	-	-	-	1,81	1,75	3,49	8,36	271	4,92e-7	5,94e-4	77,27	87,45
192	2	1e-4	Block-Jacobi-ILU	-	-	-	1,87	1,41	3,71	4,12	360	4,88e-7	6,01e-4	53,65	59,64
192	2	5e-4	Block-Jacobi-ILU	-	-	-	1,76	0,99	3,57	3,63	360	4,78e-7	5,91e-4	54,02	59,41
384	2	1e-4	Block-Jacobi-ILU	-	-	-	2,39	0,58	3,59	2,06	700	7,17e-1	1,56e-3	53,59	58,04
384	2	5e-4	Block-Jacobi-ILU	-	-	-	2,12	0,47	3,44	1,94	700	7,15e-1	1,53e-3	52,80	56,85
768	2	1e-4	Block-Jacobi-ILU	-	-	-	4,13	0,24	3,44	0,97	1100	5,23e-7	9,21e-4	45,26	50,36
768	2	5e-4	Block-Jacobi-ILU	-	-	-	4,17	0,21	3,34	0,90	1100	5,22e-7	8,97e-4	43,65	48,71
24	2	5e-4	DSC	GMRES	0	-	2,80	2,87	3,66	40,01	285	4,85e-7	7,68e-4	343,24	386,04
24	2	5e-4	DSC	GMRES	1	-	2,88	2,87	3,66	40,54	176	4,75e-7	6,16e-4	322,48	365,90
24	2	5e-4	DSC	GMRES	2	-	2,60	2,87	3,66	40,08	142	4,57e-7	6,10e-4	285,30	327,98
24	2	5e-4	DSC	GMRES	3	-	2,84	2,87	3,66	40,03	100	7,14e-7	6,51e-4	237,15	280,03
24	2	5e-4	DSC	GMRES	4	-	2,82	2,87	3,66	39,45	100	4,48e-7	3,93e-4	264,04	306,31
24	2	5e-4	DSC	GMRES	5	-	2,75	2,87	3,66	40,04	97	4,82e-7	4,75e-4	291,68	334,47
24	2	5e-4	DSC	GMRES	6	-	2,76	2,87	3,66	40,35	97	4,54e-7	3,73e-4	309,81	352,92
24	2	5e-4	DSC	GMRES	7	-	2,69	2,87	3,66	40,21	95	4,13e-7	3,63e-4	342,96	385,87
48	2	5e-4	DSC	GMRES	0	-	1,93	2,65	3,79	20,38	363	4,95e-7	1,02e-3	253,72	276,03
48	2	5e-4	DSC	GMRES	1	-	1,82	2,65	3,79	20,34	200	9,79e-1	6,71e-4	223,66	245,81
48	2	5e-4	DSC	GMRES	2	-	1,89	2,65	3,79	20,40	155	4,74e-1	7,18e-4	208,81	231,10
48	2	5e-4	DSC	GMRES	3	-	1,92	2,65	3,79	20,28	117	4,90e-7	1,64e-3	188,17	210,37
48	2	5e-4	DSC	GMRES	4	-	2,00	2,65	3,79	20,44	97	4,05e-7	9,05e-4	176,21	198,64
48	2	5e-4	DSC	GMRES	5	-	1,86	2,65	3,79	20,41	91	4,42e-7	7,45e-4	179,71	201,98
48	2	5e-4	DSC	GMRES	6	-	2,00	2,65	3,79	20,54	87	4,86e-7	5,04e-4	201,31	223,85
48	2	5e-4	DSC	GMRES	7	-	1,87	2,65	3,79	20,32	87	3,90e-7	3,83e-4	222,39	244,58
96	2	5e-4	DSC	GMRES	0	-	1,73	1,75	3,60	8,47	297	4,89e-7	6,13e-4	94,01	104,20

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
96	2	5e-4	DSC	GMRES	1	-	1,85	1,75	3,60	8,41	191	4,87e-7	6,20e-4	100,17	110,43
96	2	5e-4	DSC	GMRES	2	-	1,73	1,75	3,60	8,59	130	4,94e-7	6,69e-4	83,37	93,69
96	2	5e-4	DSC	GMRES	3	-	1,75	1,75	3,60	8,60	97	4,63e-7	6,95e-4	70,01	80,35
96	2	5e-4	DSC	GMRES	4	-	1,89	1,75	3,60	8,55	87	4,36e-7	6,22e-4	78,12	88,56
96	2	5e-4	DSC	GMRES	5	-	1,79	1,75	3,60	8,46	82	4,73e-7	5,75e-4	77,65	87,90
96	2	5e-4	DSC	GMRES	6	-	1,74	1,75	3,60	8,56	78	4,72e-7	4,33e-4	82,26	92,56
96	2	5e-4	DSC	GMRES	7	-	1,96	1,75	3,60	8,40	76	4,97e-7	3,14e-4	90,16	100,52
192	2	5e-4	DSC	GMRES	0	-	1,75	0,99	3,78	3,22	400	9,04e-7	2,21e-3	63,69	68,66
192	2	5e-4	DSC	GMRES	1	-	1,74	0,99	3,78	3,93	258	4,75e-7	8,33e-4	70,12	75,79
192	2	5e-4	DSC	GMRES	2	-	1,65	0,99	3,78	3,89	158	4,85e-7	8,53e-4	51,16	56,70
192	2	5e-4	DSC	GMRES	3	-	1,67	0,99	3,78	3,68	146	4,55e-7	4,20e-4	55,96	61,31
192	2	5e-4	DSC	GMRES	4	-	1,77	0,99	3,78	3,93	95	4,51e-7	7,22e-4	43,99	49,69
192	2	5e-4	DSC	GMRES	5	-	1,74	0,99	3,78	3,54	85	4,64e-7	8,75e-4	42,29	47,57
192	2	5e-4	DSC	GMRES	6	-	1,70	0,99	3,78	3,22	78	4,83e-7	9,29e-4	43,43	48,35
192	2	5e-4	DSC	GMRES	7	-	1,74	0,99	3,78	3,80	72	4,49e-7	7,83e-4	45,07	50,60
384	2	5e-4	DSC	GMRES	0	-	2,16	0,47	3,61	2,09	800	5,85e-7	5,37e-4	61,88	66,13
384	2	5e-4	DSC	GMRES	1	-	2,42	0,47	3,61	2,08	478	4,91e-7	8,96e-4	71,96	76,45
384	2	5e-4	DSC	GMRES	2	-	2,17	0,47	3,61	2,03	200	9,98e-7	1,46e-3	37,76	41,97
384	2	5e-4	DSC	GMRES	3	-	2,30	0,47	3,61	2,05	173	4,59e-7	5,95e-4	38,66	43,01
384	2	5e-4	DSC	GMRES	4	-	2,17	0,47	3,61	1,90	143	4,97e-7	6,95e-4	36,18	40,25
384	2	5e-4	DSC	GMRES	5	-	2,20	0,47	3,61	2,06	115	4,94e-7	5,56e-4	34,45	38,71
384	2	5e-4	DSC	GMRES	6	-	2,38	0,47	3,61	2,01	94	4,16e-7	2,98e-4	32,50	36,89
384	2	5e-4	DSC	GMRES	7	-	2,41	0,47	3,61	2,09	85	4,52e-7	3,82e-4	32,21	36,71
768	2	5e-4	DSC	GMRES	3	-	3,26	0,21	3,53	1,01	218	4,92e-7	1,02e-3	28,49	32,76
768	2	5e-4	DSC	GMRES	4	-	3,66	0,21	3,53	0,99	177	4,75e-7	4,47e-4	26,82	31,46
768	2	5e-4	DSC	GMRES	5	-	3,60	0,21	3,53	1,00	149	4,77e-7	4,10e-4	26,36	30,95
768	2	5e-4	DSC	GMRES	6	-	3,84	0,21	3,53	0,97	129	4,80e-7	5,17e-4	24,82	29,64
768	2	5e-4	DSC	GMRES	7	-	3,51	0,21	3,53	1,00	96	4,94e-7	4,54e-4	21,15	25,65
24	2	5e-4	Saad-DSC	GMRES	0	-	2,75	2,87	3,66	40,18	200	9,40e-7	5,69e-4	276,84	319,77
24	2	5e-4	Saad-DSC	GMRES	1	-	2,62	2,87	3,66	39,94	152	4,58e-7	8,80e-4	302,04	344,59
24	2	5e-4	Saad-DSC	GMRES	2	-	2,69	2,87	3,66	39,97	143	4,51e-7	4,36e-4	329,38	372,03
24	2	5e-4	Saad-DSC	GMRES	3	-	2,65	2,87	3,66	40,18	117	4,92e-7	1,21e-3	314,52	357,35
24	2	5e-4	Saad-DSC	GMRES	4	-	2,83	2,87	3,66	40,08	115	4,96e-7	1,18e-3	326,71	369,62
24	2	5e-4	Saad-DSC	GMRES	5	-	3,04	2,87	3,66	39,73	100	9,13e-7	1,46e-3	317,51	360,27
24	2	5e-4	Saad-DSC	GMRES	6	-	2,74	2,87	3,66	39,92	100	8,29e-7	1,44e-3	337,42	380,08
24	2	5e-4	Saad-DSC	GMRES	7	-	2,63	2,87	3,66	39,97	100	7,50e-7	1,24e-3	372,61	415,21
48	2	5e-4	Saad-DSC	GMRES	0	-	1,93	2,65	3,79	20,49	800	7,61e-7	3,33e-4	735,26	757,68
48	2	5e-4	Saad-DSC	GMRES	1	-	1,86	2,65	3,79	20,51	166	4,81e-7	9,65e-4	227,56	249,93
48	2	5e-4	Saad-DSC	GMRES	2	-	1,91	2,65	3,79	20,61	123	4,76e-7	9,29e-4	206,78	229,31
48	2	5e-4	Saad-DSC	GMRES	3	-	1,97	2,65	3,79	20,58	100	5,31e-7	1,01e-3	191,31	213,87
48	2	5e-4	Saad-DSC	GMRES	4	-	1,90	2,65	3,79	20,45	93	4,02e-7	5,57e-4	200,80	223,15
48	2	5e-4	Saad-DSC	GMRES	5	-	1,93	2,65	3,79	20,42	90	4,95e-7	4,38e-4	206,41	228,76

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
48	2	5e-4	Saad-DSC	GMRES	6	-	2,11	2,65	3,79	20,56	89	4,24e-7	1,47e-4	242,43	265,09
48	2	5e-4	Saad-DSC	GMRES	7	-	1,87	2,65	3,79	20,54	89	3,90e-7	3,08e-4	262,39	284,81
96	2	5e-4	Saad-DSC	GMRES	0	-	1,76	1,75	3,60	8,45	200	6,31e-7	6,50e-4	85,29	95,50
96	2	5e-4	Saad-DSC	GMRES	1	-	1,84	1,75	3,60	8,43	152	4,94e-7	8,81e-4	100,66	110,93
96	2	5e-4	Saad-DSC	GMRES	2	-	1,83	1,75	3,60	8,23	100	7,86e-7	7,90e-4	74,29	84,35
96	2	5e-4	Saad-DSC	GMRES	3	-	1,98	1,75	3,60	8,36	93	4,32e-7	6,89e-4	77,05	87,38
96	2	5e-4	Saad-DSC	GMRES	4	-	1,79	1,75	3,60	8,21	87	4,89e-7	5,49e-4	84,06	94,06
96	2	5e-4	Saad-DSC	GMRES	5	-	1,90	1,75	3,60	8,19	84	4,71e-7	4,27e-4	86,81	96,90
96	2	5e-4	Saad-DSC	GMRES	6	-	1,91	1,75	3,60	8,22	82	4,39e-7	4,92e-4	91,13	101,26
96	2	5e-4	Saad-DSC	GMRES	7	-	1,91	1,75	3,60	8,23	80	4,80e-7	4,68e-4	98,06	108,20
192	2	5e-4	Saad-DSC	GMRES	0	-	1,65	0,99	3,78	3,78	392	4,89e-7	1,33e-4	79,16	84,59
192	2	5e-4	Saad-DSC	GMRES	1	-	1,54	0,99	3,78	4,01	171	4,79e-1	1,35e-3	53,95	59,50
192	2	5e-4	Saad-DSC	GMRES	2	-	1,50	0,99	3,78	3,55	155	4,53e-7	2,60e-4	58,91	63,96
192	2	5e-4	Saad-DSC	GMRES	3	-	1,68	0,99	3,78	4,14	96	4,35e-7	1,25e-3	40,87	46,69
192	2	5e-4	Saad-DSC	GMRES	4	-	1,62	0,99	3,78	3,43	86	4,63e-7	1,22e-3	39,57	44,62
192	2	5e-4	Saad-DSC	GMRES	5	-	1,62	0,99	3,78	3,93	79	4,30e-7	1,18e-3	43,11	48,67
192	2	5e-4	Saad-DSC	GMRES	6	-	1,55	0,99	3,78	3,15	73	4,93e-7	1,27e-3	40,16	44,86
192	2	5e-4	Saad-DSC	GMRES	7	-	1,57	0,99	3,78	3,49	70	4,39e-7	1,22e-3	43,55	48,62
384	2	5e-4	Saad-DSC	GMRES	0	-	2,06	0,47	3,61	2,15	2001	1,78e-3	2,53e-1	278,46	282,66
384	2	5e-4	Saad-DSC	GMRES	1	-	2,15	0,47	3,61	2,12	270	4,89e-7	7,50e-4	59,53	63,80
384	2	5e-4	Saad-DSC	GMRES	2	-	2,15	0,47	3,61	1,88	351	4,97e-7	4,90e-4	90,20	94,24
384	2	5e-4	Saad-DSC	GMRES	3	-	2,52	0,47	3,61	2,01	143	4,67e-7	6,44e-4	43,02	47,55
384	2	5e-4	Saad-DSC	GMRES	4	-	2,09	0,47	3,61	2,08	136	4,71e-7	5,16e-4	46,31	50,49
384	2	5e-4	Saad-DSC	GMRES	5	-	2,13	0,47	3,61	2,14	94	4,50e-7	4,84e-4	35,19	39,46
384	2	5e-4	Saad-DSC	GMRES	6	-	2,19	0,47	3,61	1,86	91	4,57e-7	1,33e-3	37,64	41,70
384	2	5e-4	Saad-DSC	GMRES	7	-	2,12	0,47	3,61	2,04	80	4,88e-7	5,63e-4	37,11	41,28
768	2	5e-4	Saad-DSC	GMRES	3	-	3,78	0,21	3,53	0,96	171	4,93e-7	3,05e-4	26,08	30,82
768	2	5e-4	Saad-DSC	GMRES	4	-	3,84	0,21	3,53	1,00	300	9,97e-7	6,61e-4	50,51	55,36
768	2	5e-4	Saad-DSC	GMRES	5	-	3,29	0,21	3,53	1,00	115	4,92e-7	7,58e-4	22,70	26,99
768	2	5e-4	Saad-DSC	GMRES	6	-	3,80	0,21	3,53	1,01	95	4,72e-7	1,41e-3	20,91	25,72
768	2	5e-4	Saad-DSC	GMRES	7	-	3,79	0,21	3,53	1,00	85	4,61e-7	6,15e-4	20,66	25,45

TRACE-UHBR-Testfall bei Blockgröße 5 und äußerer FGMRES(100)-Iteration mit CGS

Prozesse	ILU		Vorkonditionierung				Repart.	rel. ILU Größe		ILU	vork. FGMRES				Gesamt-
	$lfil$	ϵ_{drop}	Typ	Löser	Iter.	ϵ	Dauer [s]	min.	max.	Dauer [s]	Iter.	rel. Res.	abs. Fehler	Dauer [s]	Dauer [s]
768	2	1e-4	Block-Jacobi-ILU	-	-	-	-	0,24	3,44	1,01	1100	5,23e-7	9,21e-4	38,32	43,35
768	2	5e-4	Block-Jacobi-ILU	-	-	-	-	0,21	3,34	0,91	1100	5,22e-7	8,97e-4	37,25	42,21
768	2	5e-4	DSC	GMRES	3	-	3,67	0,21	3,53	1,00	215	4,97e-7	1,11e-3	26,70	31,38
768	2	5e-4	DSC	GMRES	4	-	4,23	0,21	3,53	1,01	173	4,79e-7	3,44e-4	25,32	30,56
768	2	5e-4	DSC	GMRES	5	-	3,79	0,21	3,53	1,01	147	4,46e-7	4,60e-4	24,62	29,43
768	2	5e-4	DSC	GMRES	6	-	4,24	0,21	3,53	1,01	125	4,98e-7	8,62e-4	23,95	29,21
768	2	5e-4	DSC	GMRES	7	-	4,43	0,21	3,53	1,00	96	4,17e-7	4,69e-4	20,22	25,66
768	2	5e-4	DSC	GMRES	8	-	3,70	0,21	3,53	1,01	87	4,85e-7	5,12e-4	20,58	25,29
768	2	5e-4	DSC	GMRES	9	-	3,95	0,21	3,53	1,01	78	4,99e-7	4,65e-4	19,35	24,32
768	2	5e-4	DSC	GMRES	10	-	3,58	0,21	3,53	1,01	73	4,80e-7	4,59e-4	20,02	24,62
768	2	5e-4	DSC	GMRES	15	-	3,52	0,21	3,53	1,02	53	4,83e-7	3,53e-4	21,01	25,55
768	2	5e-4	DSC	GMRES	20	-	3,76	0,21	3,53	1,02	44	4,36e-7	3,97e-4	22,16	26,94
768	2	5e-4	Saad-DSC	GMRES	8	-	3,95	0,21	3,53	0,99	78	4,32e-7	5,18e-4	19,43	24,37
768	2	5e-4	Saad-DSC	GMRES	9	-	3,42	0,21	3,53	0,98	71	4,56e-7	5,04e-4	19,93	24,33
768	2	5e-4	Saad-DSC	GMRES	10	-	4,01	0,21	3,53	1,00	67	4,80e-7	5,69e-4	19,95	24,97
768	2	5e-4	Saad-DSC	GMRES	15	-	4,49	0,21	3,53	0,99	51	3,68e-7	3,45e-4	20,90	26,39
768	2	5e-4	Saad-DSC	GMRES	20	-	4,10	0,21	3,53	0,99	42	4,82e-7	4,29e-4	21,66	26,75
768	2	5e-4	Quasi-exakt-DSC	GMRES	10	0,01	3,47	0,21	3,53	1,01	74	4,38e-7	6,78e-4	31,42	35,90
768	2	5e-4	Quasi-exakt-DSC	GMRES	20	0,01	3,63	0,21	3,53	1,01	37	4,67e-7	7,17e-4	30,45	35,08
768	2	5e-4	Quasi-exakt-DSC	GMRES	5	0,01	3,48	0,21	3,53	1,01	168	4,56e-7	7,00e-4	37,63	42,11
768	2	5e-4	Quasi-exakt-DSC	GMRES	10	0,1	3,93	0,21	3,53	1,01	74	4,34e-7	6,46e-4	29,12	34,06
768	2	5e-4	Quasi-exakt-DSC	GMRES	20	0,1	3,89	0,21	3,53	1,02	37	4,50e-7	6,67e-4	27,50	32,41
768	2	5e-4	Quasi-exakt-DSC	GMRES	5	0,1	3,95	0,21	3,53	1,02	168	4,61e-7	7,07e-4	36,13	41,09

Literatur

[1] Basermann, A., U. Jaekel, M. Nordhausen und K. Hachiya: *Parallel iterative solvers for sparse linear systems in circuit simulation.* Future Generation Computer Systems, 21(8):1275 – 1284, 2005, ISSN 0167-739X.

[2] Dahmen, W. und A. Reusken: *Numerik für Ingenieure und Naturwissenschaftler.* Springer-Lehrbuch. Springer Berlin Heidelberg, zweite, korrigierte Auflage, 2008, ISBN 978-3-540-76492-2 (PRINT) 978-3-540-76493-9 (ONLINE).

[3] DLR: *TAU-Code User Guide Release 2011.2.0.* Institute of Aerodynamics and Flow Technology, Deutsches Zentrum für Luft- und Raumfahrt e.V., Braunschweig, 2011.

[4] DLR: *Technical Documentation of the DLR TAU-Code Release 2011.1.0.* Technischer Bericht, Institute of Aerodynamics and Flow Technology, Deutsches Zentrum für Luft- und Raumfahrt e.V., Braunschweig, 2011.

[5] Karypis, George und Vipin Kumar: *METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrice Version 4.0.* University of Minnesota, Department of Computer Science and Engineering / Army HPC Research Center, Minneapolis, 1998.

[6] Karypis, George, Kirk Schloegel und Vipin Kumar: *ParMETIS Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 3.1.* University of Minnesota, Department of Computer Science and Engineering / Army HPC Research Center, Minneapolis, 2003.

[7] Kersken, Hans Peter, C. Frex, C. Voigt und G. Ashcroft: *Time-linearized and Time-accurate 3D RANS Methods for Aeroelastic Analysis in Turbomachinery.* In: *ASME TurboExpo 2010: Power for Land, Sea and Air*, Nummer GT2010-22940, Glasgow, UK, June 2010.

[8] LeVeque, Randall J.: *Finite Volume Methods for Hyperbolic Problems.* Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002, ISBN 978-0-521-00924-9.

[9] Pope, Stephen B.: *Turbulent flows.* Cambridge University Press, 2000, ISBN 978-0-521-59886-6.

[10] Saad, Yousef: *Iterative methods for sparse linear systems.* SIAM, 2003, ISBN 978-0-89871-534-7.

[11] Saad, Yousef und Maria Sosonkina: *Distributed schur complement techniques for general sparse linear systems.* Siam Journal on Scientific Computing, 21, 1999.

[12] Smith, Barry F., Petter E. Bjørstad und William D. Gropp: *Domain decomposition: parallel multilevel methods for elliptic partial differential equations.* Cambridge University Press, New York, NY, USA, 1996, ISBN 978-0-521-49589-9.

[13] Vorst, H.A.: *Iterative Krylov methods for large linear systems.* Cambridge monographs on applied and computational mathematics. Cambridge University Press, 2003, ISBN 978-0-521-81828-5.