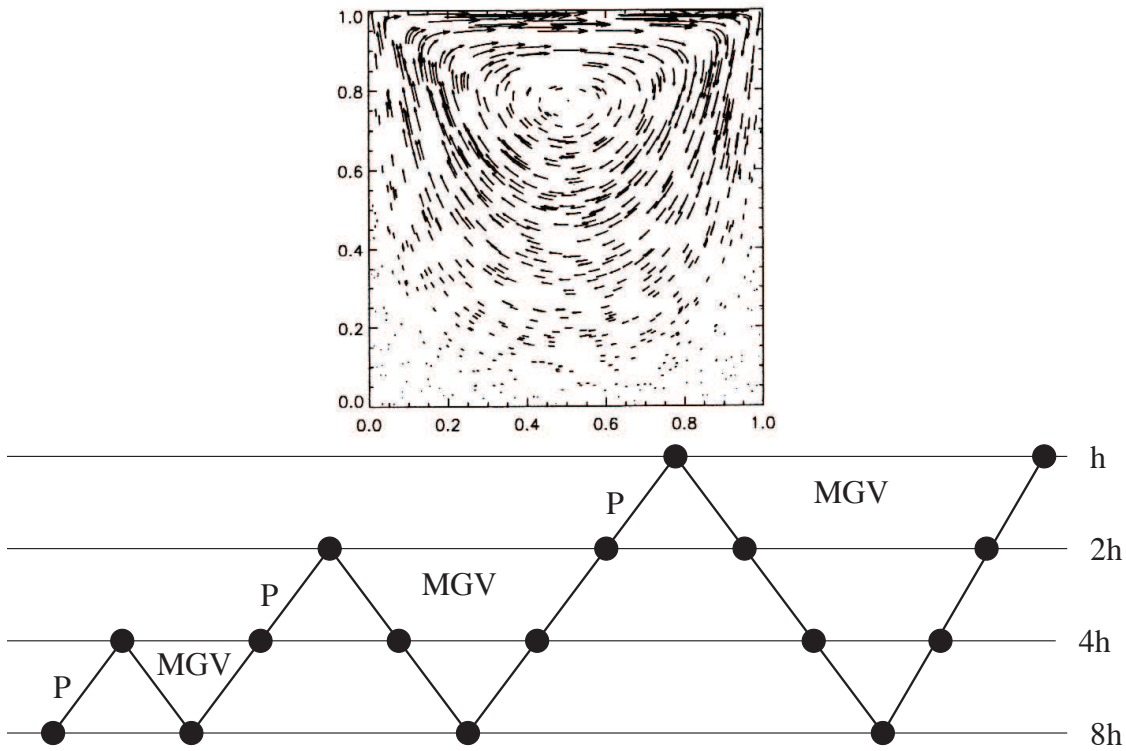
Karsten Urban

High Performance Computing
(Teil III)



SKRIPT, UNIVERSITÄT ULM, SOMMERSEMESTER 2013

Vorwort. Dieses Manuskript ist entstanden aus Mitschriften und Skripten verschiedener Vorlesungen, die ich seit 2002 an der Universität Ulm gehalten habe. Das vorliegende Manuskript bietet einen Rahmen für die Nachbearbeitung der Vorlesungen und der Vorbereitung auf Prüfungen. Dieses Manuskript kann keinesfalls das Studium von Lehrbüchern ersetzen. Eine entsprechende Liste von Lehrbüchern findet sich im Literaturverzeichnis und auf der Internet–Seite der Vorlesung. Jedes

Manuskript weist Fehler auf, sicher auch dieses. Wenn Sie Fehler, Druckfehler, sprachliche Unzulänglichkeiten oder inhaltliche Flüchtigkeiten finden, würde ich mich über einen entsprechenden Hinweis per Email freuen. Sie helfen damit zukünftigen Studierenden. Vielen Dank im Voraus.

Das letzte Kapitel ist als *Zusatzstoff* gedacht, es wird in der Vorlesung nicht mehr behandelt, zeigt aber, wie die zuvor erlernten Techniken in eine reale Anwendung einfließen können.

Danksagung. Eine Reihe von Personen haben bei der Erstellung geholfen. Ich danke Markus Bantle, Dr. Michael Lehn, Ralf Leidenderer, Daniel Nolte, Kristina Steih, Sebastian Singer, Alexander Stippler und Timo Tonn für zahlreiche Hinweise. Meinen Kollegen bzw. ehemaligen Kollegen Andreas Borchert, Stefan Funken und Alexander Keller danke ich für diverse Hinweise und die sehr gute Zusammenarbeit bei den Vorlesungen. Ganz besonderer Dank gebührt auch Frau Petra Hildebrand, die meine handschriftlichen Aufzeichnungen in LaTeX umgesetzt und zahlreiche Grafiken erstellt hat.

Copyright. Alle Rechte, insbesondere das Recht auf Vervielfältigung und Verbreitung sowie der Übersetzung, vorbehalten. Kein Teil des Werkes darf in irgendeiner Form ohne schriftliche Genehmigung des Autors reproduziert oder unter Verwendung elektronischer Systeme oder auf anderen Wegen verarbeitet, vervielfältigt oder verbreitet werden.

Stand. Ulm, Mai 2013.
Karsten Urban

Inhaltsverzeichnis

1 MEHRGITTER–VERFAHREN (MULTIGRID)

Mehrgitter–Verfahren gehören heute zu den schnellsten Verfahren für lineare Gleichungssysteme, die z.B. aus der Diskretisierung partieller Differential- oder Integralgleichungen entstehen. Mittlerweile sind Mehrgitter–Verfahren ein sehr umfangreiches Feld geworden, auf dem auch heute noch sehr aktiv geforscht wird. Es gibt an eine ganze Reihe von Veröffentlichungen zu dem Thema. Das Verfahren besteht aus einzelnen Komponenten, die allgemein beschrieben werden können und dann an das jeweilige Problem angepasst werden müssen. Wir beschreiben die grundlegenden Ideen und Eigenschaften an einem 1D–Beispiel.
In diesem Kapitel beschreiben wir zunächst die grundlegende Idee sowie den Algorithmus und gehen dann auf die Parallelisierung ein.

1.1 EINFÜHRENDES BEISPIEL UND GRUNDLEGENDE IDEE

1.1.1 Das Zwei–Punkt–Randwertproblem

Betrachte das Randwertproblem

$$-u''(x) = f(x), x \in (0,1) =: \Omega, \qquad u(0) = u(1) = 0. \tag{1.1.1}$$

Zunächst einige analytische Eigenschaften. Man nennt $\mu_k \in \mathbb{R}$ *Eigenwert* und $u_k \in C^2(0,1)$ *Eigenfunktion* des Differenzialoperators in (1.1.1), wenn

$$-u_k''(x) = \mu_k\, u_k(x), x \in (0,1) =: \Omega, \qquad u_k(0) = u_k(1) = 0. \tag{1.1.2}$$

Man kann leicht nachrechnen, dass

$$\mu_k = k^2\pi^2 \qquad \text{und} \qquad u_k(x) = \sin(k\pi x), \quad k \in \mathbb{N},$$

gilt. Mit steigendem $k \in \mathbb{N}$ steigt auch die Frequenz $k\pi$ der Schwingung. Insbesondere sind die Eigenfunktionen linear unabhängig, die Lösung des Randwertproblems besitzt eine Entwicklung in eine so genannte Sinus–Reihe (aufgrund der Randbedingungen fallen eventuelle Kosinus–Terme weg)

$$u(x) = \sum_{k\in\mathbb{N}} \alpha_k\, u_k(x),$$

wobei die α_k die Fourier–Koeffizienten sind. Die Lösung u hat also ein Spektrum, das aus allen ganzzahligen Frequenzen besteht. Man spricht von einem *Multiskalen–Problem*, da die Lösung Komponenten auf ganz unterschiedlichen (Längen-)Skalen besitzt. Man hat im Laufe der Jahrzehnte gelernt, dass besonders effiziente numerische Verfahren diese Mehrskalen–Struktur ausnutzen sollten.

1.1.2 Diskretisierung

Wir wollen nun sehen, in wie weit sich diese Multiskalen–Struktur des Problems auf die Diskretisierung auswirkt. Da sich dies nicht wesentlich zwischen den verschiedenen Diskretisierungen unterscheidet, betrachten wir die einfachste, nämlich Finite Differenzen (was in 1D bekanntermaßen identisch mit linearen Finiten Elementen auf einem äquidistanten Gitter ist).
Wir verwenden das äqudistante Gitter

$$\Omega_h := \{x_i^h := ih : \ i = 0, \ldots, N+1\}$$

mit der Schrittweite

$$h := \frac{1}{N+1}, \qquad N \in \mathbb{N}.$$

Wir erhalten das bekannte Tridiagonal–System

$$\frac{1}{h^2} \boldsymbol{A}_N \boldsymbol{u}_N = \boldsymbol{f}_N \tag{1.1.3}$$

mit $\boldsymbol{f}_N = (f(x_i))_{i=1}^N$ und

$$\boldsymbol{A}_N = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N \times N}.$$

Wir untersuchen Eigenwerte und -vektoren des diskretisierten Systems.

Lemma 1.1.1 *Eigenwerte und -vektoren von $\boldsymbol{A}_N$ sind gegeben durch*

$$\lambda_k = 2\big(1 - \cos(k\pi h)\big), \qquad \boldsymbol{v}^k := \left(\sin\left(\frac{k\pi i}{N+1}\right)\right)_{i=1}^N. \tag{1.1.4}$$

Beweis: Übung. □

Die ersten 4 Eigenvektoren sind in Abb. 1.1 dargestellt. Offenbar bilden die Vektoren $\boldsymbol{v}^k$, $k = 1, \ldots, N$, eine Eigenvektor–Basis des $\mathbb{R}^N$, also können wir die gesuchte Lösung in dieser Basis entwickeln:

$$\boldsymbol{u}_N = \sum_{k=1}^{N} \alpha_k \, \boldsymbol{v}^k.$$

Wenn wir uns Abb. 1.1 noch einmal genau ansehen, bemerken wir, dass $\boldsymbol{v}^k$ offenbar einer Schwingung mit der Frequenz $k\pi$ entspricht. Also ist auch die gesuchte diskrete Lösung eine Linearkombination verschiedener Frequenz–Anteile. Insbesondere besteht die Lösung aus niederfrequenten und hochfrequenten Anteilen. Das Mehrgitter–Verfahren nutzt nun diese Beobachtung aus.
Wir nennen Eigenfunktionen u_k bzw. die Eigenvektoren $\boldsymbol{v}^k$ mit $k > N/2$ *hochfrequent* (oder *oszillierend*), die übrigen *niederfrequent* (oder *glatt*).

Übung 1.1.2 *Betrachten Sie die Finite Differenzen–Diskretisierung des 2D–Poisson–Problems mit dem 5–Punkte–Stern. Bestimmen Sie Eigenwerte $\mu_{k,\ell}$ und -funktionen $u_{k,\ell}$ des kontinuierlichen Problems sowie die Eigenvektoren $\boldsymbol{v}_{k,\ell}$ des diskreten Problems.*

1.1.3 Das Richardson–Verfahren

Wir lösen (1.1.1) mit einem klassischen Iterationsverfahren, z.B. mit Gauß–Seidel oder —noch einfacher— mit dem Richardson–Verfahren. Dieses wollen wir ein wenig näher untersuchen als wir das in Numerik I gemacht haben, um die Wirkungsweise zu verstehen. Zur Erinnerung: Zur Lösung eines linearen Gleichungssystems $Ax = b$ im $\mathbb{R}^n$ und einem gegebenen Startwert $x^0 \in \mathbb{R}^n$ lautet das Verfahren

$$x^{k+1} := (I - \omega A)x^k + \omega\, b, \qquad k = 0, 1, 2, \ldots$$

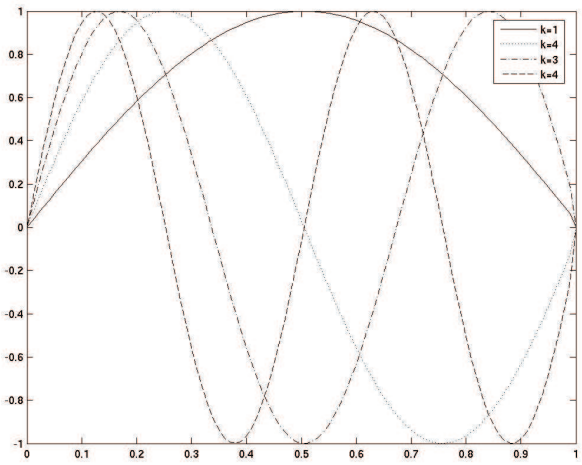


Abbildung 1.1: Eigenvektoren für $k = 1, \ldots, 4$.

mit einem *Dämpfungsparameter* $\omega \in \mathbb{R}$. Wir hatten für den Spektralradius der Iterationsmatrix

$$M(\omega) := I - \omega A$$

gezeigt, dass $\rho(M(\omega)) = \max\{|1 - \omega\lambda_{\text{max}}|, |1 - \omega\lambda_{\text{min}}|\}$ gilt. Also konvergiert das Verfahren für alle $\omega \in (0, 2/\lambda_{\text{max}})$. Wir hatten schon in der Numerik I gesehen, dass das Richardson–Verfahren oftmals sehr langsam konvergiert und deswegen benutzt man es nur sehr selten als Lösungsverfahren. Es hat jedoch die Eigenschaft, dass bestimmte Skalen–Anteile des *Anfangs–Residuums*

$$r^0 := b - Ax^0$$

sehr schnell gedämpft werden, andere jedoch nur extrem langsam verkleinert werden. Diese Eigenschaft nennt man *Glättungs–Eigenschaft*.
Für die Eigenwerte von $M(\omega)$ gilt $\mu_k = 1 - \omega\lambda_k$ mit den Eigenwerten λ_k von $\boldsymbol{A}_N$, also

$$\mu_k = 1 - 2\omega(1 - \cos(k\pi h)) = (1 - 2\omega) + 2\omega\cos(k\pi h), \quad h = \frac{1}{N+1}.$$

Sei x^* die exakte Lösung des Gleichungssystems $Ax = b$, dann gilt für den Fehler

$$e_k := x^k - x^*$$

des Richardson–Verfahrens im k-ten Schritt

$$\begin{aligned} e^{k+1} &= x^{k+1} - x^* = M(\omega)x^k + \omega b - x^* \\ &= (I - \omega A)x^k + \omega b - x^* + \omega(Ax^* - b) \\ &= M(\omega)(x^k - x^*) = M(\omega)e^k = \cdots = (M(\omega))^k e^0. \end{aligned}$$

Wir entwickeln nun den Startfehler e^0 in die Eigenvektorbasis $\{v^k : k = 1, \ldots, N\}$

$$e^0 = \sum_{k=1}^{N} \alpha_k v^k,$$

wobei aufgrund der Orthonormalität gilt

$$\|e^0\|^2 = \sum_{k=1}^{N} |\alpha_k|^2.$$

Nun gilt

$$\begin{aligned} e^1 &= M(\omega)e^0 = \sum_{k=1}^{N} \alpha_k \mu_k v^k \\ &= \sum_{k=1}^{N} \alpha_k ((1-2\omega) + 2\omega\cos(k\pi h))^2 v^k \end{aligned} \tag{1.1.5}$$

und wiederum wegen der Orthonromalität der v^k

$$\begin{aligned} \|e^1\|^2 &= \sum_{k=1}^{N} |\alpha_k|^2 ((1-2\omega) + 2\omega\cos(k\pi h))^2 \\ &\leq ((1-2\omega) + 2\omega\cos(\pi h))^2 \sum_{k=1}^{N} |\alpha_k|^2 \\ &= ((1-2\omega) + 2\omega\cos(\pi h))^2 \|e^0\|^2, \end{aligned}$$

also induktiv

$$\|e^k\| \leq ((1-2\omega) + 2\omega\cos(\pi h))^k \|e^0\| =: \rho^k \|e^0\|.$$

Wir entwickeln nun den Konvergenzfaktor in eine Taylor–Reihe und erhalten

$$\begin{aligned} \rho &= (1-2\omega) + 2\omega\cos(\pi h) \\ &= (1-2\omega) + 2\omega \left(1 - \frac{(\pi h)^2}{2} + \mathcal{O}(h^4)\right) \\ &= 1 - \mathcal{O}(h^2), \end{aligned}$$

also ist das Konvergenzverhalten umso schlechter, je kleiner h (also je feiner das Gitter Ω_h) ist.
Dies ist jedoch zu pessimistisch durch die grobe Abschätzung von $\cos(k\pi h)$ durch $\cos(\pi h)$. Nach (1.1.5) lautet der Fehler–Reduktionsfaktor von e^0 in Richtung v^k

$$1 - 2\omega + 2\omega\cos(k\pi h) =: \rho_k.$$

Wir betrachten nun ausschließlich den hochfrequenten Anteil, also

$$k \geq \frac{N+1}{2},$$

dann gilt einerseits wegen $\cos(x) \geq -1$

$$\rho_k \geq 1 - 4\omega$$

und andererseits wegen $\cos(k\pi h) = \cos\left(\frac{k\pi}{N+1}\right) \leq 0$ für $\frac{N+1}{2} \leq k \leq N+1$ die Abschätzung

$$\rho_k \leq 1 - 2\omega,$$

also

$$1 - 4\omega \leq \rho_k \leq 1 - 2\omega, \tag{1.1.6}$$

bzw.

$$|\rho_k| \leq \max\{(1-4\omega), (1-2\omega)\},$$

man erhält also die optimale Dämpfung $\rho^* = \frac{1}{3}$ für die Wahl $\omega = \frac{1}{3}$. Die hochfrequenten Anteile werden also mit $\rho^* = \frac{1}{3}$ gedämpft *unabhängig* von der Schrittweite h! Aufgrund der Dämpfung

hochfrequenter Anteile spricht man von einem *Glätter*. Andere beliebte Glätter (mit ähnlichen Eigenschaften) sind Jacobi- und Gauß–Seidel–Verfahren.
Wir verstehen nun auch besser, warum klassische Iterations–Verfahren so langsam konvergieren, nämlich weil niederfrequente Anteile nur extrem langsam reduziert werden.
Obige Analyse ist speziell für das Zwei–Punkt–Randwertproblem und Finite Differenzen. Man muss die Glättungs–Eigenschaft für jedes vorliegende Probelm und jede Diskretisierung nachweisen.
Wegen dieser Glättungs–Eigenschaft schreiben wir im Folgenden oft kurz

$$Sx_0$$

für eine Anwendung von $M(\omega)$ auf x_0. Dabei steht „*S*“ für smoothing (Glättung) und kann irgendein geeignetes Iterationsverfahren sein.

1.1.4 Ein Beispiel

Beispiel 1.1.3 *Wir betrachten das Randwertproblem (1.1.1) mit $f \equiv 0$, also der exakten Lösung $u \equiv 0$. Wir verwenden zwei äquidistante Gitter auf $[0, 1]$ mit Schrittweiten*

$$h_1 = \frac{1}{N_1},\ N_1 = 32\ , \quad h_2 = \frac{1}{N_2},\ N_2 = 64$$

und starten mit den diskreten Werten einer Funktion auf diesen Gittern:

$$u^{(0)}(x) := \sin(\omega_1 x) + \sin(\omega_2 x)$$

mit $\omega_1 = 27$ und $\omega_2 = 3$, also einer Überlagerung zweier Schwingungen. In Abbildung 1.2 ist links die Anfangsfunktion dargestellt, in der Mitte der Fehler (Residuum) nach 20 Iterationen für N_1 und rechts für N_2. Wir sehen, dass der hochfrequente Anteil von $u^{(0)}$ (also der bezüglich ω_1) schnell verringert wird, während der niederfrequente (bezüglich ω_2) sichtbar bleibt. Dieser Teil wird zwar von $N_1 = 32$ zu $N_2 = 64$ reduziert, ist aber immer noch klar erkennbar.

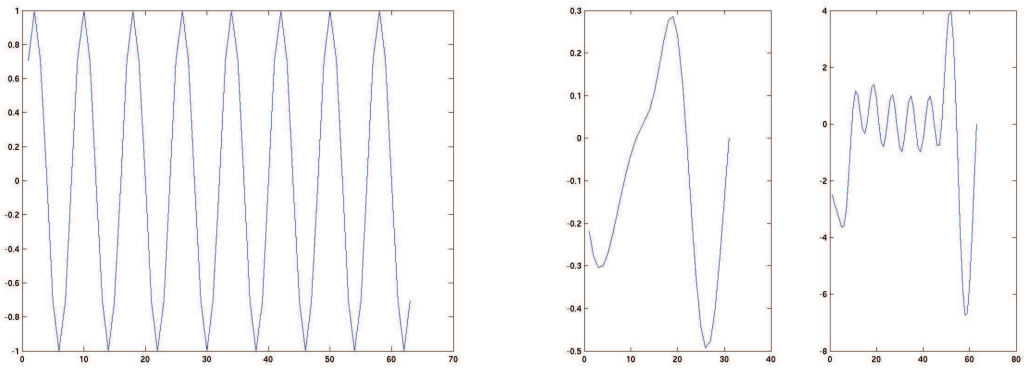


Abbildung 1.2: Startfunktion (links) und Residuum nach 20 Iterationen für $N_1 = 64$ (Mitte) und $N_2 = 32$ (rechts).

Bemerkung 1.1.4 *Eine wesentliche Beobachtung ist die, dass eine Funktion auf einem feinen Gitter Ω_h glatt sein kann, hingegen auf einem groben Gitter Ω_{2h} nicht (bzw. weniger) glatt ist, vgl. Abb. 1.3.*

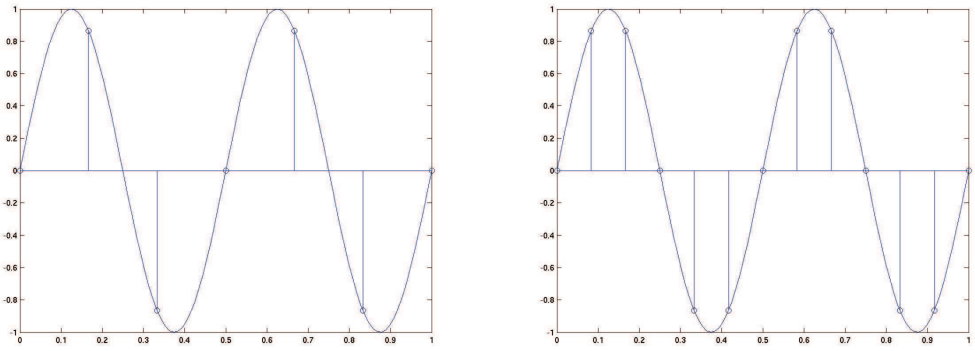


Abbildung 1.3: Sinus–Funktion auf grobem (links) und feinem (rechts) Gitter.

1.2 GITTER–HIERARCHIE

Die obigen Beobachtungen legen nahe, statt eines einzigen Gitters mehrere zu verwenden. Sinnvoll ist es weiterhin, diese als geschachtelt anzusehen, etwa

$$\Omega_h \subset \Omega_{2h}.$$

Wir werden im Allgemeinen nicht erwarten können, dass $\Omega_h \subset \Omega_H$ für $h < H$ gilt, wir werden aber einen gewissen Zusammenhang annehmen. Details dazu später.
Um die Gitter–Hierarchie zu beschreiben, bietet sich die Schreibweise

$$\boldsymbol{A}_h \boldsymbol{u}_h = \boldsymbol{f}_h$$

für das lineare Gleichungssystem auf Ω_h an. Zur Illustration des Vorgehens betrachten wir außerdem

$$\boldsymbol{A}_{2h} \boldsymbol{u}_{2h} = \boldsymbol{f}_{2h}$$

auf Ω_{2h}. Wir wollen 3 potentielle Vorteile ausnutzen:

1. Die Dimension von Ω_{2h} ist wesentlich kleiner als die von Ω_h, also ist das Gleichungssystem mit vergleichsweise niedrigem Aufwand numerisch zu lösen. Wir können eine (grobe) Approximation auf Ω_{2h} z.B. als Startwert für eine Iteration auf Ω_h verwenden.

2. Die Fehlerreduktion
$$1 - \mathcal{O}(4h^2) < 1 - \mathcal{O}(h^2)$$
ist (zumindest geringfügig) besser auf Ω_{2h}

3. Eine niederfrequente Funktion auf Ω_h kann auf Ω_{2h} hochfrequent sein. Diese Fehler–Bestandteile können also auf Ω_{2h} schnell reduziert werden.

Es kommt also auf das geschickte Zusammenspiel verschiedener Gitter an, wir brauchen geeignete Übergänge zwischen den Gittern, also Operatoren

$$P_h^{2h} : \Omega_{2h} \to \Omega_h,$$

Prolongation genannt und

$$R_{2h}^{h} : \Omega_h \to \Omega_{2h},$$

die sogenannte *Restriktion*. Ein Beispiel für die Prolongation ist lineare Interpolation, bei der die Werte an den Punkten aus Ω_{2h} übernommen werden und die Werte auf $\Omega_h \backslash \Omega_{2h}$ durch Mittelung entstehen, d.h.

$$u_{2i}^h := u_i^{2h} \ , \quad 1 \leq i \leq \frac{N-1}{2},$$
$$u_{2i+1}^h := \frac{1}{2}\left(u_i^{2h} + u_{i+1}^{2h}\right) \ , \quad 0 \leq i \leq \frac{N-1}{2},$$

bei der Nummerierung wie oben. Wir können dies abkürzen als

$$\boldsymbol{u}_h = \boldsymbol{P}_h^{2h} \boldsymbol{u}_{2h}$$

mit der Prolongationsmatrix

$$\boldsymbol{P}_h^{2h} = \frac{1}{2} \begin{pmatrix} 1 & & & & & \\ 2 & 1 & & & & \\ 1 & 2 & & & & \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & 2 & 1 & \\ & & & 1 & 2 & \\ & & & & 1 & \end{pmatrix} \in \mathbb{R}^{(N+2)\times\frac{N+1}{2}+1},$$

falls $h = \frac{N+1}{2}$ wie oben. Diese Prolongation ist in Abbildung 1.4 dargestellt.

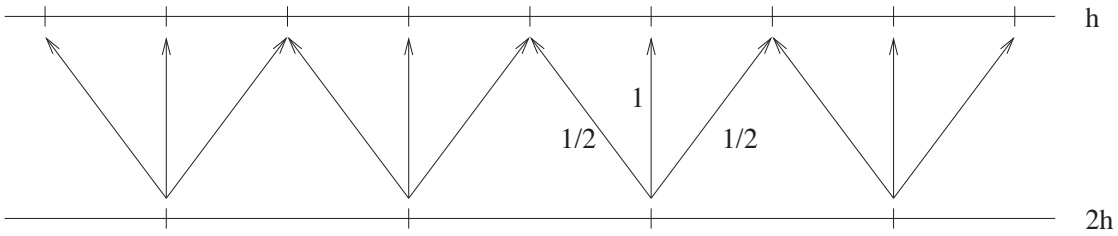


Abbildung 1.4: Prolongation durch lineare Interpolation in 1D.

Natürlich sind auch andere Prolongation denkbar und je nach Anwendung auch sinnvoll.

Ebenso gibt es für die Restriktion eine Reihe von Möglichkeiten. Die einfachste Möglichkeit besteht darin, die Werte an den zusätzlichen Punkten zu „vergessen“ und die anderen beizubehalten, d.h.

$$u_i^{2h} + u_{2i}^h \ , \quad 0 \leq i \leq \frac{N+1}{2},$$

vgl. Abbildung 1.5
Gebräuchlicher — und oftmals sinnvoller — ist die sogenannte *lineare Restriktion*, bei der ein gewichtetes Mittel der benachbarten Punkte gebildet wird

$$u_i^{2h} := \frac{1}{4}\left(u_{2i-1}^h + 2u_{2i}^h + u_{2i+1}^h\right), 1 \leq i \leq \frac{N-1}{2},$$

also

$$\boldsymbol{u}^{2h} = \frac{1}{2}(\boldsymbol{P}_h^{2h})^T \boldsymbol{u}^h =: \boldsymbol{R}_{2h}^h \boldsymbol{u}^h,$$

vgl. Abbildung 1.6.
Die Wahl von Restriktion und Prolongation beeinflusst natürlich wesentlich das Verhalten des Verfahrens. Die Konstruktion hängt von der jeweiligen Problemstellung ab.

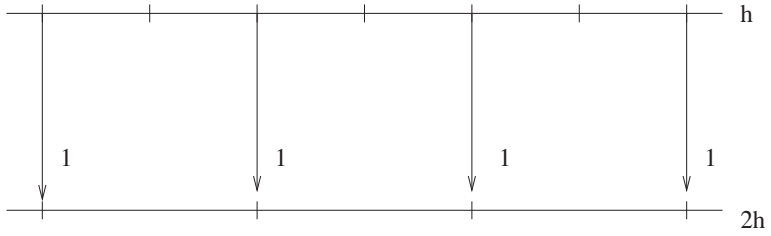


Abbildung 1.5: Einfache Restriktion in 1D.

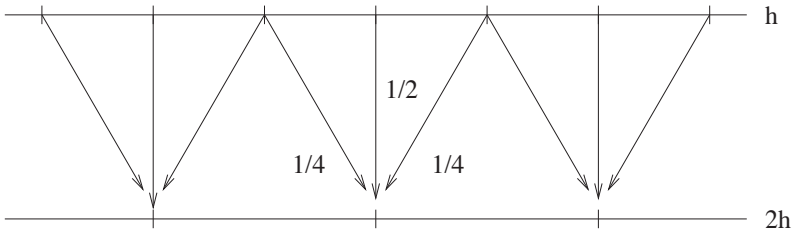


Abbildung 1.6: Lineare Restriktion in 1D.

Bemerkung 1.2.1 *Oftmals wird die Prolongation P_h^{2h} auch mit I_h^{2h} und die Restriktion R_{2h}^h mit I_{2h}^h bezeichnet.*

1.3 GROBGITTERKORREKTUR (COARSE GRID CORRECTION)

Wir wollen (wie der Name bereits sagt), die Näherung auf Ω_h durch einige Iterationen auf Ω_{2h} korrigieren und damit verbessern. Das grobe Vorgehen ist wie folgt:

1. Führe ν_1 Glättungsschritte auf Ω_h durch, erhalte v_h.
2. Bestimme die Restriktion des Residuums auf Ω_{2h}:
$$r^{2h} = R_{2h}^h(f_h - A_h u_h).$$
3. „Löse“ $A_{2h}e_{2h} = r^{2h}$ auf Ω_{2h}.
4. Korrektur:
$$v_h := v_h + P_h^{2h} e_{2h}$$
durch Prolongation der Näherungslösung.
5. Führe ν_2 Glättungsschritte auf Ω_h mit Startwert v_h durch.

Für $\nu_1 = 1$, $\nu_2 = 0$ erhält man die einfachste Grobgitterkorrektur, von der man jedoch zeigen kann, dass sie nicht konvergiert, da die entsprechende Iterationsmatrix einen nicht-trivialen Eigenvektor zum Eigenwert $\lambda = 1$ besitzt.

Es stellt sich natürlich die Frage, was man unter 3. mit „Löse“ zu verstehen hat. Wenn $\dim \Omega$ „klein“ ist, kann man das Gleichungssystem direkt lösen. Ansonsten kann man an dieser Stelle den Algorithmus rekursiv aufrufen. Man erhält den V–Zyklus.
Hierzu sei
$$A_\ell := A_{2^p-\ell h}, u_\ell := u_{2^p-\ell h}, f_\ell := f_{2^p-\ell h},$$

d.h., wir haben hierarchische Gitter

$$\Omega_h \supset \Omega_{2h} \supset \Omega_{4h} \supset \ldots \supset \Omega_{2^p h},$$

also entspricht $\ell = 0$ dem gröbsten und $\ell = p$ dem feinsten Gitter.

Algorithmus 1.3.1 (Mehrgitter–V–Zyklus) *function* $u_\ell = V$*–Zyklus* $(A_\ell, f_\ell, u_\ell^0, \ell)$

IF $\ell = 0$
  $d_0 := A_0^{-1} f_0$ *% exakte Lösung*
ELSE
  $u_\ell := S^{\nu_1}(A_\ell, f_\ell, u_\ell^0)$ *% Vor–Glättung*
  $r_\ell := f_\ell - A_\ell u_\ell$ *% Residuum*
  $r_{\ell-1} := R_{\ell-1}^{\ell} r_\ell$ *% Restriktion des Residuums*
  $d_{\ell-1} := V$*–Zyklus* $(A_{\ell-1}, r_{\ell-1}, 0, \ell-1)$ *% Rekursion*
  $u_\ell := u_\ell + P_\ell^{\ell-1} d_{\ell-1}$ *% Prolongation und Korrektur*
  $u_\ell := S^{\nu_2}(A_\ell, f_\ell, u_\ell)$ *% Nach–Glättung*
END

Wir können dieses Verhalten wie in Abbildung 1.7 darstellen. Hier bedeutet $S =$ Glättung (smoothing), $R =$ Restriktion, $P =$ Prolongation und $E =$ exakte Lösung.

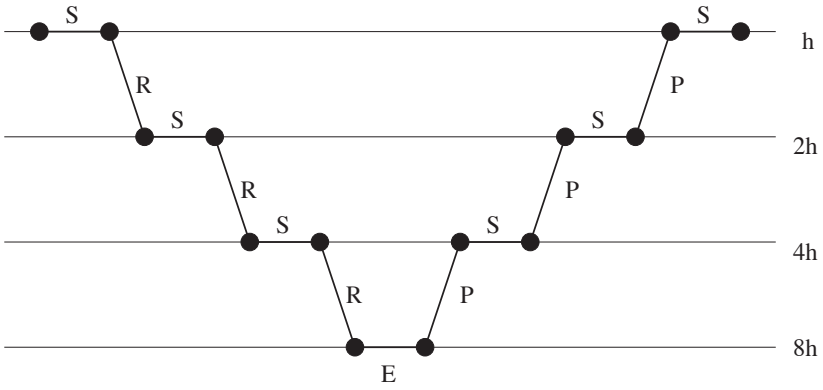


Abbildung 1.7: V–Zyklus mit 4 Gittern.

Eine Verallgemeinerung des V–Zyklus ist der sogenannte γ–Zyklus, bei dem die Rekursion γ-fach ($\gamma \in \mathbb{N}$) ausgeführt wird.

Algorithmus 1.3.2 (Mehrgitter–γ–Zyklus) *function* $u_\ell = \gamma$*–Zyklus* $(A_\ell, f_\ell, u_\ell^0, \ell, \gamma)$

IF $\ell = 0$
  $d_0 := A_0^{-1} f_0$ *% exakte Lösung*
ELSE
  $u_\ell := S^{\nu_1}(A_\ell, f_\ell, u_\ell^0)$ *% Vor–Glättung*
  $r_\ell := f_\ell - A_\ell u_\ell$ *% Residuum*
  $r_{\ell-1} := R_{\ell-1}^{\ell} r_\ell$ *% Restriktion des Residuums*
  $d_{\ell-1}^0 := 0$
  FOR $i = 1$ *to* γ *% Rekursion*
    $d_{\ell-1}^i := \gamma$*–Zyklus* $(A_{\ell-1}, r_{\ell-1}, d_{\ell-1}^{i-1}, \ell-1)$
  END
  $u_\ell := u_\ell + P_\ell^{\ell-1} d_{\ell-1}^\gamma$ *% Prolongation und Korrektur*
  $u_\ell := S^{\nu_2}(A_\ell, f_\ell, u_\ell)$ *% Nach–Glättung*
END

Für $\gamma = 1$ erhalten wir wieder den V–Zyklus, für $\gamma = 2$ den sogenannten W–Zyklus, vgl. Abbildung 1.8

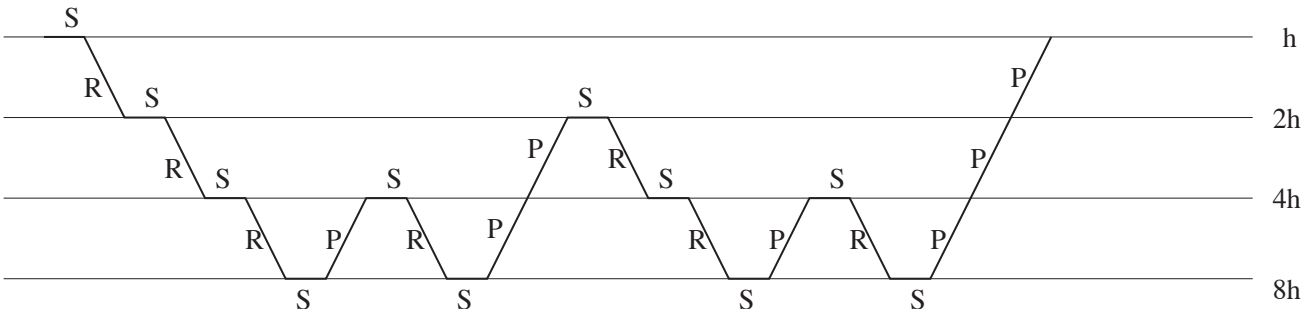


Abbildung 1.8: W–Zyklus mit 4 Gittern, zur Vereinfachung der Darstellung ohne Nach–Glättung, d.h. $\nu_2 = 0$.

Das Mehrgitter–Verfahren entsteht nun dadurch, dass man den V- oder W–Zyklus in jeder Iteration anwendet.

Algorithmus 1.3.3 (Mehrgitterverfahren) *1.) Wähle Startvektor* $u_h^0 \in \mathbb{R}^N$, *Stufenzahl* $p \in \mathbb{N}$, $k := 0$, $r_h^0 := f_h - A_h u_h^0$

2.) Falls Abbruchkriterium erfüllt: STOP

3.) $u_h^{k+1} :=$ *γ–Zyklus* $(A_h, f_h, u_h^k, p, \gamma)$

4.) $r_h^{k+1} := f_h - A_h u_h^{k+1}$

5.) $k := k + 1$, *gehe zu 2.)*

Bemerkung 1.3.1 *(a) Es gibt weitere Varianten des Mehrgitterverfahrens. Bestimmt man z.B. den Startwert durch eine Rekursion von grobem nach feinem Level, dann spricht man vom* vollständigen Mehrgitterverfahren *(full multigrid), vgl. Abbildung 1.9 für den V–Zyklus.*

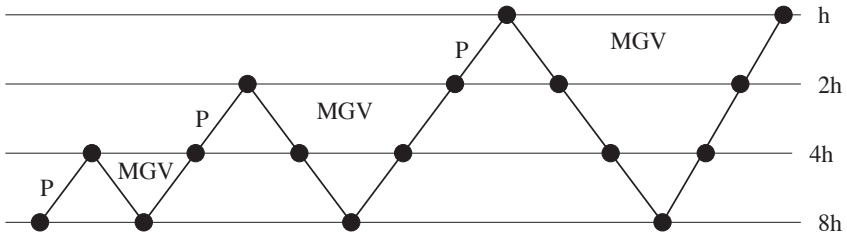


Abbildung 1.9: Vollständiges Mehrgitterverfahren mit einem V–Zyklus, MGV bedeutet Mehrgitterverfahren.

(b) Das einfachste Mehrgitterverfahren besteht darin, Approximationen auf groben Leveln als Startwerte für feinere Skalen zu verwenden ohne erneut auf die groben Skalen zu gehen. Dieses Verfahren ist auch als nested iteration *bekannt und in Abbildung 1.10 dargestellt. Allerdings kann der Fehler auf Ω_h hier immer noch Anteile von gröberen Skalen enthalten, was der Grund dafür ist, dass dieses Verfahren nicht unbedingt konvergiert.*

(c) Unter gewissen Annahmen an den Mehrgitter–Operator und den Prolongations–Operator kann man eine Fehlerabschätzung für das Mehrgitter–Verfahren beweisen, die in vielen Fällen optimal ist.

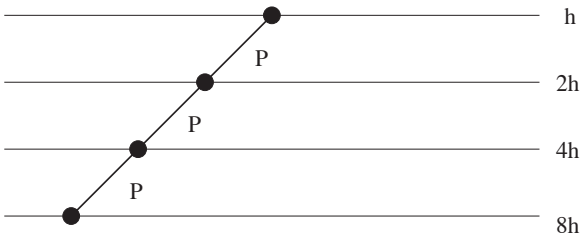


Abbildung 1.10: Nested Iteration — Methode mit 4 Leveln.

(d) Man kann das Mehrgitterverfahren auch als Vorkonditionierer im pcg–Verfahren verwenden.

(e) Wenn man keine natürlich Gitter–Hierarchie hat, dann kann man mit graphentheoretischen Methoden versuchen, aus der Struktur der Matrix A_h eine Partitionierung der Indexmenge zu gewinnen, um mittels dieser eine Matrix–Hierarchie zu erzeugen. Verwendet man diese in einem Mehrgitterverfahren, so spricht man von algebraischen Mehrgitterverfahren.

1.4 PROLONGATION UND RESTRIKTION IN 2D

Nun zum Übergang auf 2D. Der Übersichtlichkeit halber sei

$$\Omega = [0,1]^2$$

das Einheitsquadrat, welches sowohl in x- als auch in y-Richtung in N Teilintervalle unterteilt sei, d.h.

$$\Omega_h := \{(ih, jh) : 0 \le i, j \le N\}, \quad h = \frac{1}{N},$$

wobei die Werte an den äußeren Gitterpunkten

$$(0, jh), (1, jh), (ih, 0), (ih, 1), \quad 0 \le i, j \le N,$$

durch die Randbedingungen festgelegt sind. Wir wollen nun Beispiele für Prolongation P_h^{2h} und Restriktion R_{2h}^h angeben. Dazu sei $N = 2M$ gerade. Dann ist die Prolongation mittels Interpolation definiert durch

$$v^h = P_h^{2h} v^{2h}$$

und

$$\begin{aligned}
v^h_{2i,2j} &:= v^{2h}_{i,j}, && \text{für } 1 \le i,j \le \tfrac{N}{2} - 1,\\
v^h_{2i+1,2j} &:= \tfrac{1}{2}(v^{2h}_{i,j} + v^{2h}_{i+1,j}), && \text{für } 0 \le i \le \tfrac{N}{2} - 1,\ 1 \le j \le \tfrac{N}{2} - 1,\\
v^h_{2i,2j+1} &:= \tfrac{1}{2}(v^{2h}_{i,j} + v^{2h}_{i,j+1}), && \text{für } 1 \le i \le \tfrac{N}{2} - 1,\ 0 \le j \le \tfrac{N}{2} - 1,\\
v^h_{2i+1,2j+1} &:= \tfrac{1}{4}(v^{2h}_{i,j} + v^{2h}_{i+1,j} + v^{2h}_{i,j+1} + v^{2h}_{i+1,j+1}), && \text{für } 0 \le i,j \le \tfrac{N}{2} - 1,
\end{aligned}$$

vgl. Abbildung 1.11.
Für die Prolongation geben wir drei Möglichkeiten an

a) einfache Interpolation,

b) den *Half–Weighting–Operator*, und

c) den *Full–Weighting–Operator*,

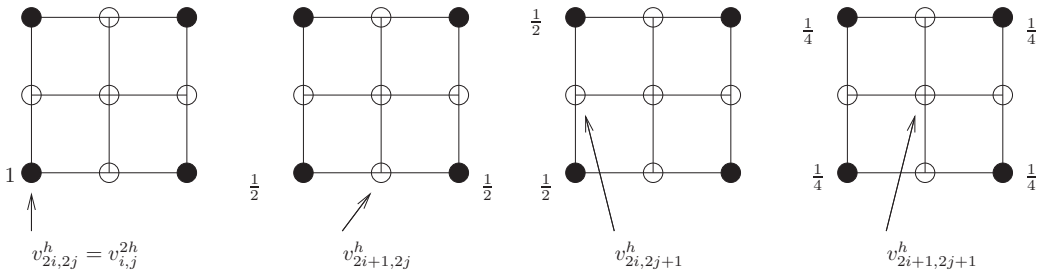


Abbildung 1.11: 2D–Prolongation durch Interpolation • sind Punkte in Ω_{2h}, ∘ die zusätzlichen Punkte in Ω_h. Die entsprechenden Gewichte sind angegeben.

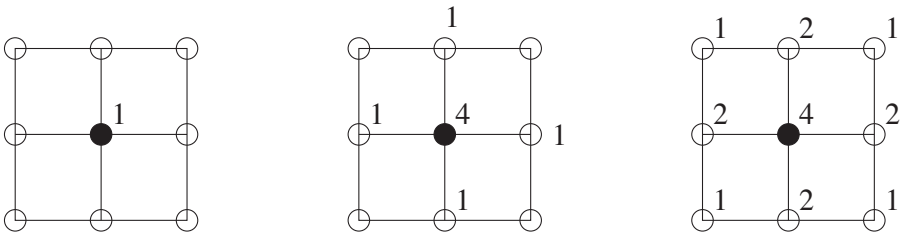


Abbildung 1.12: Verschiedene Möglichkeiten der Prolongation, einfache Prolongation (links), Half–Weighting (Mitte) und Full–Weighting (rechts).

vgl. Abbildung 1.12.
Diese sind definiert durch

a) $v^{2h}_{i,j} = v^h_{2i,2j}, \quad 1 \le i,j \le \frac{N}{2} - 1,$

b) $v^{2h}_{i,j} = \frac{1}{8}(4v^h_{2i,2j} + v^h_{2i-1,2j} + v^h_{2i+1,2j} + v^h_{2i,2j-1} + v^h_{2i,2j+1})$

c)
$$\begin{aligned} v^{2h}_{i,j} = \tfrac{1}{16} \quad & (4v^h_{2i,2j} \\ & +2(v^h_{2i-1,2j} + v^h_{2i+1,2j} + v^h_{2i,2j-1} + v^h_{2i,2j+1}) \\ & +(v^h_{2i-1,2j-1} + v^h_{2i+1,2j-1} + v^h_{2i-1,2j+1} + v^h_{2i+1,2j+1})) \end{aligned}$$

1.5 PARALLELISIERUNG VON MULTIGRID

Um das Mehrgitterverfahren zu parallelisieren, betrachten wir die verschiedenen Komponenten des Verfahrens, d.h.

1. Glättung
2. Restriktion
3. Prolongation

Für alle drei Komponenten verwendet man die Idee der Gebietszerlegung (domain decomposition), die wir zunächst beschreiben.

Die Hauptschwierigkeit bei der Parallelisierung des Mehrgitterverfahrens sind die Gitter unterschiedlicher Größe. Dies führt schnell zu Systemen immer kleinerer Granularität. Zudem müssen Zwischenergebnisse für u und das Residuum r gespeichert werden.

In den bisherigen Fällen war stets $\Omega_{2h} \subset \Omega_h$, d.h., Gitterpunkte auf dem groben Gitter sind gleichzeitig auch Punkte des feinen Gitters. Dies muss nicht unbedingt der Fall sein. Wenn es jedoch so ist, dann teilen wir das Gebiet Ω (bzw. das feinste Gitter Ω_h) in P Teilgebiete auf und weisen jedem Prozessor genau ein Teilgebiet (mit allen Vergröberungen) zu. Für $N = 16$ und $P = 16$ ist dies in Abbildung 1.13 dargestellt.

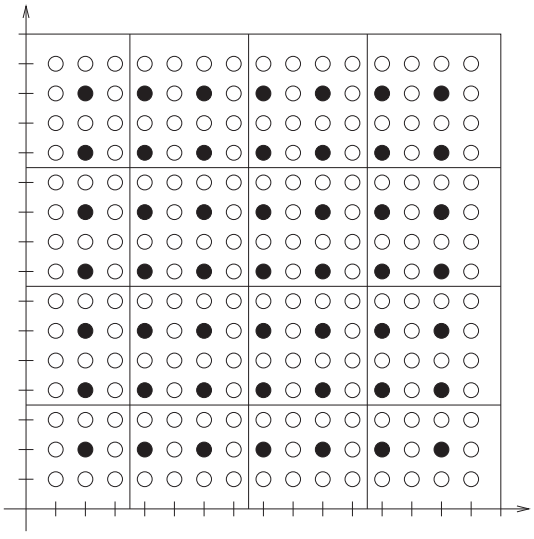


Abbildung 1.13: Unterteilung der Gitter Ω_h und Ω_{2h} in 16 Teilgitter für 16 Prozessoren (aus [1]).

Zur Berechnung von Näherungen an den Teilgittergrenzen benötigt man die Werte der Randpunkte der angrenzenden Teilgitter. Daher müssen diese Werte beim Übergang zwischen den Gittern ausgetauscht und synchronisiert werden. Wir beschreiben dies für verteilten und gemeinsamen Speicher für den V–Zyklus für Prozessoren $P(i)$, $1 \leq i \leq P$.

Verteilter Speicher

$u_0^{\text{loc}} := u^{\text{loc}},\ u_0^{(i)} := u^{(i)}$

Lokaler Datenaustausch (u_ℓ)
FOR $\ell := 0$ TO $L-1$ DO
  $u_\ell^{\text{loc}} := S^{\nu_1}(A_\ell^{\text{loc}}, f_\ell^{\text{loc}}, u_\ell)$
  $r_\ell^{\text{loc}} := f_\ell^{\text{loc}} - A_\ell^{\text{loc}} u_\ell^{\text{loc}}$

  Lokaler Datenaustausch (r_ℓ, u_ℓ)
  $f_{\ell+1}^{\text{loc}} := (P_\ell^{\ell+1}) r_\ell$
  $u_{\ell+1}^{\text{loc}} := 0$
END
IF $i = 1$ THEN $u_L^{\text{loc}} := (A_L^{\text{loc}})^{-1} f_L^{\text{loc}}$
Lokaler Datenaustausch (u_ℓ)
FOR $\ell := L-1$ DOWNTO 0 DO
  $u_\ell^{\text{loc}} := u_\ell^{\text{loc}} + (P_\ell^{\ell+1})^{\text{loc}} u_{\ell+1}$
  Lokaler Datenaustausch (u_ℓ)
  $u_\ell^{\text{loc}} := S^{\nu_2}(A_\ell^{\text{loc}}, f_\ell^{\text{loc}}, u_\ell)$
END
$u_{\text{loc}}^{k+1} := u_0^{\text{loc}}$
Globaler Datentausch

Gemeinsamer Speicher

FOR $i := 1$ TO P DO PARALLEL
  $u_0^{(i)} := u^{(i)},\ f_0^{(i)} := f^{(i)}$
BARRIER
FOR $\ell := 0$ TO $L-1$ DO
  FOR $i := 0$ TO P DO PARALLEL
    $u_\ell^{(i)} := S^{\nu_1}(A_\ell^{(i)}, f_\ell^{(i)}, u_\ell)$
    $r_\ell^{(i)} := f_\ell^{(i)} - A_\ell^{(i)} u_\ell^{(i)}$
    BARRIER
    $f_{\ell+1}^{(i)} := (P_\ell^{\ell+1}) r_\ell$
    $u_{\ell+1}^{(i)} := 0$
  END
END
IF $i = 1$ THEN $u_L^{(i)} := (A_L^{(i)})^{-1} f_L^{(i)}$
BARRIER
FOR $\ell := L-1$ DOWNTO 0 DO
  FOR $i := 1$ DO PARALLEL
    $u_\ell^{(i)} := u_\ell^{(i)} + (P_\ell^{\ell+1})^{(i)} u_{\ell+1}$
    BARRIER
    $u_\ell^{(i)} := S^{\nu_2}(A_\ell^{(i)}, f_\ell^{(i)}, u_\ell)$
    END
  $u^{k+1,i} := u_0^{(1)}$
  BARRIER

Man kann den Datenaustausch durch geschickte Strategien (geringfügig) verringern.

1.5.1 Parallelisierung des Glätters

Natürlich hängt die Parallelisierung sowohl von der Wahl des Glätters als auch von der Nummerierung der Indizes ab.

Jacobi–Verfahren

Für die Berechnung der Werte an den inneren Punkten benötigt man nur die Werte auf den inneren Gitterpunkten. Am Rand werden die unmittelbaren Nachbarn benötigt, die überlappend mit übergeben werden, vgl. Abbildung 1.14.

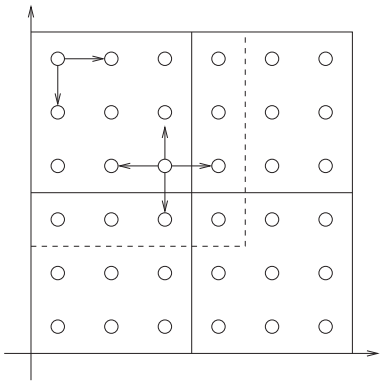


Abbildung 1.14: Index-Zugriffe beim Jacobi–Verfahren.

Gauß–Seidel mit Red–Black–Nummerierung
Wiederum wird ein Überlappungsbereich mit übergeben. Man führt den Glätter in zwei Halbschritten aus, zunächst für die roten, dann für die schwarzen Punkte.

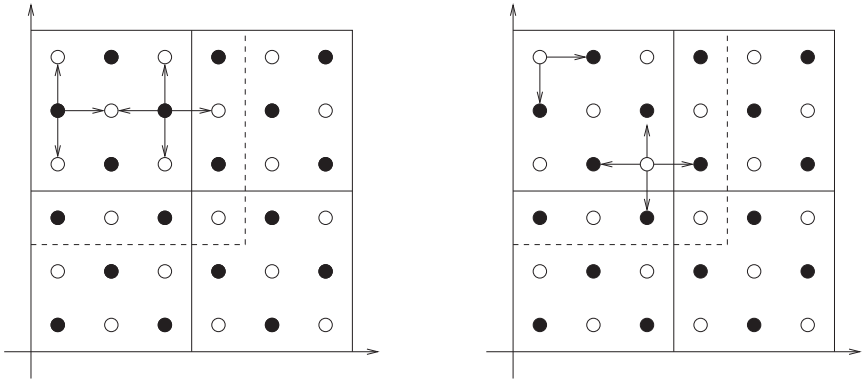


Abbildung 1.15: Index–Zugriffe bei Gauß–Seidel mit Red–Blck–Nummerierung.

Zunächst werden die Näherungen an den roten Punkten (links) paralle bestimmt. Diese Daten müssen an den Rändern synchronisiert werden, bevor die schwarzen Punkte bearbeitet werden können (rechts). Diese Daten werden dann wieder ausgetauscht, vgl. Abbildung 1.15.

Im Vergleich zu Jacobi werden die Daten doppelt so oft ausgetauscht, dafür aber nur die Hälfte der Daten.[1]

Gauß–Seidel mit Wavefront–Nummerierung
Durch die Nummerierung ist in jedem Schritt ein Datenaustausch notwendig, vgl. Abbildung 1.16.

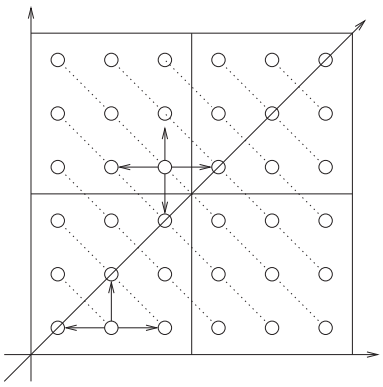


Abbildung 1.16: Index–Zugriffe bei Gauß–Seidel mit Wavefront–Nummerierung.

Effizienz
Für den Parallelisierungsgrad gilt

	Jacobi	G-S Red–Black	G-S Wavefront
par-deg (M)	$\#\Omega_h$	$\frac{1}{2}\#\Omega_h$	$\sqrt{\#\Omega_h}$

Zur Erinnerung: par-deg ist die Anzahl der parallel ausführbaren Operationen. Für $\#\Omega_h \geq 4$ ist also Gauß–Seidel mit Red–Black–Nummerierung besser als mit Wavefront–Nummeriereung.

[1] Dieses Verfahren ist derzeit nicht in der Software–Toolbox enthalten (Stand: 05/2013).

Wir betrachten nun den Speed-up in Abhängigkeit von Prozessoranzahl P und Problemgröße N, d.h.

$$S(P,N) = \frac{T_{\text{seriell}}(N)}{T_{\text{parallel}}(P,N)}$$

sowie die Effizienz

$$e(P,N) = \frac{S(P,N)}{P}.$$

Satz 1.5.1 *Wir setzen voraus*

1. *A_h bestehe — wie beim 5–Punkte–Stern — lediglich aus lokalen Abhängigkeiten.*
2. *Der Glätter sei hinreichend parallelisierbar, d.h. par-deg $(S) = \mathcal{O}(\#\Omega_h)$.*
3. *Die Anzahl der arithmetischen Operationen pro Gitterpunkt ist asymptotisch $(N \to \infty)$ für alle Teilgitter gleich.*

Dann gilt für festes P

$$\lim_{N\to\infty} e(P,N) = 1.$$

Beweis: Zunächst gilt

$$S(P,N) = \frac{T_{\text{seriell}}(N)}{\frac{1}{P}T_{\text{comp}} + T_{\text{comm}}(P)},$$

wobei T_{comp} die Rechenzeit und T_{comm} die Kommunikationszeit ist, also

$$S(P,N) = P\frac{T_{\text{seriell}}(N)}{T_{\text{seriell}}(N)}\left(1 + \frac{T_{\text{comm}}(P)}{T_{\text{comp}}(N)}\right)^{-1}.$$

Da sich die Kommunikation auf die Randpunkte der Teilgebiete beschränkt (diese seien mit $\partial\Omega_h(P)$ bezeichnet), gilt

$$\frac{T_{\text{comm}}(P)}{T_{\text{comp}}(N)} = \frac{\#\partial\Omega_h(P)}{\#\,\mathring{\Omega}_h\,(N)} \underset{N\to\infty}{\longrightarrow} 0$$

für festes P, wenn $\mathring{\Omega}_h\,(N)$ die Menge der inneren Punkte bezeichnet, also folgt

$$\lim_{N\to\infty} S(P,N) = P$$

für festes P und damit $\lim_{N\to\infty} e(P,N) = 1$. □

1.5.2 Parallelisierung der Restriktion

Nun betrachten wir die Parallelisierung durch Gebietszerlegung für Restriktionen. Diese hängt natürlich wieder von der Nummerierung ab. Da das Gauß–Seidel–Verfahren bessere Glättungs–Eigenschaften besitzt als das Jacobi–Verfahren, betrachten wir erstes, obwohl der Parallelisierungsgrad niedriger ist. Zur Erinnerung: Es gilt

$$\begin{aligned} u_r^k &= D_r^{-1}(-Eu_b^{k-1} + b_r) \\ u_b^k &= D_b^{-1}(-Eu_r^k + b_b) \end{aligned}$$

für die roten und schwarzen Teile u_r^k und u_b^k. Bei gleicher Dimension gilt $D_r = D_b$ und damit für

$$A = \begin{pmatrix} D & E \\ E^T & D \end{pmatrix}$$

$$\begin{aligned} r_b^k &= b_b - (Au^k)_b \\ &= b_b - E^T u_r^k - D u_b^k \\ &= b_b + E^T D^{-1} E u_b^{k-1} - E^T D^{-1} b_r + E^T u_r^k - b_b \\ &= E^T D^{-1}(E u_b^{k-1} - b_r) + E^T D^{-1}(b_r - (E u_b^{k-1}) \\ &= 0, \end{aligned}$$

d.h., das Residuum verschwindet an den schwarzen Gitterpunkten. Also muss der Defekt lediglich an den roten Gitterpunkten berechnet werden.

$$\begin{aligned} r_r^k &= b_r - (Au^k)_r \\ &= b_r - D u_r^k - E u_b^k. \end{aligned}$$

Aus r_r^k lässt sich leicht (je nach Definition) die Restriktion $R_{\ell-1}^{\ell} r^k = R_{\ell-1}^{\ell} r_r^k$ auf die roten Gitterpunkte berechnen. Insbesondere ist keine Kommunikation notwendig.

1.5.3 Parallelisierung der Prolongation

Nach der Aufdatierung in den Überlappungsbereichen am Ende des Glätters (am Ende eines Zyklus), liegen alle Daten für die Prolongation lokal vor. Weitere Kommunikation ist nicht notwendig.
Damit haben wir alle Bestandteile des Mehrgitterverfahrens in paralleler Weise zusammen.

2 LR- UND QR–ZERLEGUNG

Wir gehen kurz auf die Parallelisierung der direkten Lösungsverfahren für lineare Gleichungssysteme bzw. lineare Ausgleichsprobleme ein.

2.1 LR–ZERLEGUNG

Die LR–Zerlegung mit Zeilenpivotisierung lautet

$$PA = LR$$

und ein lineares Gleichungssystem $Ax = b$ wird dann durch zwei gestaffelte Systeme gelöst.

$$Ly = b, \quad Rz = y, \quad x = Pz.$$

Die serielle Version lautet bekanntlich:

Algorithmus 2.1.1 (LR–Zerlegung mit Zeilenpivotisierung)
$P := I,\ L := I, R := A$
FOR $k = 1$ *TO* $n - 1$ *DO*

Bestimme Index s *mit* $|u_{ks}| = \max\limits_{j=k,\ldots,n} |u_{kj}|$
Vertausche Spalten mit Index s *und* k *von* R *und* P
FOR $i = k + 1$ *TO* n *DO*

$\ell_{ik} := \frac{r_{ik}}{r_{kk}},\ r_{ik} := 0$
FOR $j = k + 1$ *TO* n *DO*
$r_{ij} := r_{ik} - \ell_{ik} r_{kj}$
END

END

END

Zur Parallelisierung der LR–Zerlegung will man natürlich erreichen, dass nicht die gesamte Matrix A auf alle Prozessoren $P_1, \ldots, P_p$ verteilt wird, sondern nur diejenigen Zeilen (Spalten), die benötigt werden.

Definition 2.1.1 *Eine Matrix* $A \in \mathbb{R}^{n \times m}$ *wird im Parallelrechner* zyklisch gespeichert *nach Zeilen (Spalten), falls* P_i *genau die Zeilen (Spalten)* j *von* A *mit*

$$j \bmod P = i$$

enthält.

Damit ergibt sich folgendes Verfahren.

Algorithmus 2.1.2 (Parallel LR–Zerlegung mit Zeilenpivotisierung)
myrows $:= \{j \in \mathbb{N} : j \bmod P = i,\ j \leq n\},\ 1 \leq i \leq P$
$P := I,\ L := I, R := A$ *(nur für Zeilen aus myrows)*
FOR $k = 1$ *TO* $n - 1$ *DO*

IF $k \in$ *myrows*

Bestimme Index s *mit* $|r_{ks}| = \max_{j=k,\dots,n} |r_{kj}|$
Versende $(r_{kk}, r_{k,k+1}, \dots, r_{kn})$ *und Index* s *an alle* P_i
$(t_k, t_{k+1}, \dots, t_n) := (r_{kk}, r_{k,k+1}, \dots, r_{k,n})$

ELSE

Empfange $(t_k, \dots, t_n)$ *und Index* s

END
Vertausche die Spalten mit Index s *und* k *von* R *und* P *(nur für die Zeilen aus myrows)*
Vertausche Elemente mit Index s *und* k *von* t
FOR $i = k+1$ *TO* n *DO*

IF $i \in$ *myrows*
$\ell_{ik} := \frac{r_{ik}}{t_k},\ r_{ik} := 0$
FOR $j = k+1$ *TO* n *DO*
$r_{ij} := r_{ij} - \ell_{ik} t_j$
END

END

END

Die Paralleleisierung der beiden gestaffelten Systeme ist mit viel Kommunikation verbunden, da jeweils die gesamte Information der vorherigen Stufen benötigt wird.
Da die Auflösung dieser Systeme bei vollbesetzten Matrizen $\mathcal{O}(n^2)$ Operationen benötigt (im Gegensatz zu $\mathcal{O}(n^3)$ bei der LR–Zerlegung), kann es sinnvoller sein, diese auf nur einem Prozessor durchzuführen.

Für spätere Referenz nennen wir obige ℓ_{ik} *Multiplikatoren*.

2.2 QR–Zerlegung mit Householder–Spiegelungen

Wie aus Numerik I bekannt, lautet die Householder–Spiegelmatrix

$$H = Id - 2\frac{vv^T}{v^T v},$$

wobei v so gewählt wird, dass

$$Ha = \alpha e_1,$$

wenn a die erste Spalte von A ist. Man rechnet leicht nach, dass

$$v = a - \alpha e_1, \quad \alpha = \pm \|a\|_2$$

gilt. Zur Erinnerung noch einmal der Algorithmus:

Algorithmus 2.2.1 (Householder–QR)
FOR $k = 1$ *TO* n *DO*

$\alpha_k = -$ *sign* $(a_{kk}) \sqrt{a_{kk}^2 + \dots + a_{mk}^2}$
$v_k = (0, \dots, 0, a_{kk}, \dots, a_{mk})^T - \alpha_k e_k$
$\beta_k = v_k^T v_k$
IF $\beta_k = 0$ *THEN CONTINUE WITH NEXT* k *END*

FOR $j = k$ *TO* n *DO*
$\gamma_i = v_k^T a_j$
$a_j = a_j - (2\gamma_j/\beta_k)v_k$
END

END.

Bemerkung 2.2.1 *Bezüglich der Parallelisierung ist Householder sehr ähnlich zu LR, der Vektor* v_k *ist ähnlich zum Vektor der Multiplikatoren mit dem Unterschied der Orientierung (Zeile im Gegensatz zu Spalte). Der restliche Algorithmus ist bezüglich Schleifen–Orientierung und Index–Zugriff vollkommen analog, so dass wir nicht in die Details gehen.*

2.3 QR–ZERLEGUNG MIT GIVENS–ROTATION

Die Givens–Rotation führt sukzessive Drehungen mit Matrizen der Form

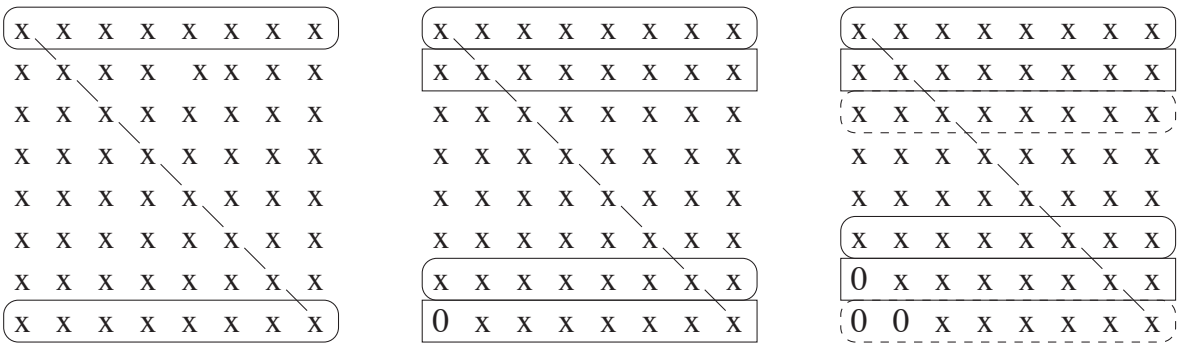
$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, \qquad c = \cos\alpha = \frac{a_1}{\sqrt{a_1^2 + a_2^2}}, \quad s = \sin\alpha = \frac{a_2}{\sqrt{a_1^2 + a_2^2}},$$

bezogen auf einen Vektor $\boldsymbol{a} = [a_1, a_2]^T$ aus (jeweils eingebettet in den $\mathbb{R}^n$). Die Reihenfolge der Drehungen spielt dabei keine Rolle, solange diese nur auf Nicht–Null–Einträge angewandt werden.

Um die Parallelisierbarkeit zu untersuchen, stellen wir zunächst fest, dass $G_{ij}A$, also die Drehung der Komponente x_j auf x_i wie folgt wirkt

Lese–Zugriff	Schreib–Zugriff
Zeile i, j	Zeilen i, j

Damit ist klar, dass eine spaltenweise Aufteilung von A viel Kommunikation erzeugt. Wir untersuchen daher eine zeilenweise Aufteilung von A. In der folgenden Abbildung beschreiben wir die Zugriffe bei einer (8×8)–Matrix.



1. Schritt 2. Schritt 3. Schritt

Dies führt im Schritt 5 zu einem Konflikt, also muss man die Reihenfolge stärker versetzen. Außerdem ist der 2. Schritt nur dann parallel ausführbar, wenn die zweite Zeile bereits gedreht wurde, was im

Allgemeinen nicht der Fall ist. Damit ergibt sich folgendes Schema, das zeigt, in welchem Schritt welcher Eintrag eliminiert werden kann.

$$\begin{bmatrix}
\times & & & & & & & \\
4 & \times & & & & & & \\
6 & 8 & \times & & & & & \\
5 & 7 & 9 & \times & & & & \\
4 & 6 & 8 & 10 & \times & & & \\
3 & 5 & 7 & 9 & 11 & \times & & \\
2 & 4 & 6 & 8 & 10 & 12 & \times & \\
1 & 3 & 5 & 7 & 9 & 11 & 13 & \times
\end{bmatrix}$$

Der höchste Parallelisierunggrad ist offenbar $\frac{n}{2}$ im Schritt $n-1$.

Bemerkung 2.3.1 *(a) Nach jedem Schritt müssen die Zeilen ausgetauscht werden, dies kann jedoch reduziert werden, indem man Block–Matrizen bildet und die entsprechenden Einträge vollständig eliminiert.*

(b) Zahlreiche spezielle Strategien sind denkbar, je nach Besetztheitsmuster.

3 EIGENWERT–PROBLEME

Wir kennen aus Numerik 1 (Numerische Lineare Algebra) die Vektor–Iteration, inverse Vektor–Iteration und das QR–Verfahren (mit und ohne shifts). Wir werden diese Verfahren hier teilweise wiederholen, stets unter dem Aspekt der Parallelisierbarkeit.

3.1 ÄHNLICHKEITS–TRANSFORMATIONEN

Zunächst bringt man $A \in \mathbb{R}^{n\times n}$ mittels Ähnlichkeits–Transformationen auf eine Gestalt, die einfacher zu handhaben ist.

Definition 3.1.1 *Zwei Matrizen $A, B \in \mathbb{R}^{n\times n}$ heißen* ähnlich*, wenn es eine reguläre Matrix $T \in \mathbb{R}^{n\times n}$ gibt mit*

$$B = T^{-1}AT. \tag{3.1.1}$$

Ähnlichkeits–Transformationen erhalten das Spektrum: Sei y ein Eigenvektor von B zum Eigenwert λ,

$$By = \lambda y,$$

dann gilt

$$T^{-1}ATy = \lambda y, \text{ also } A(Ty) = \lambda(Ty),$$

also ist Ty Eigenvektor zum Eigenwert λ, d.h.

$$\sigma(A) = \sigma(T^{-1}AT) \tag{3.1.2}$$

für das *Spektrum*

$$\sigma(A) := \{\lambda \in \mathbb{R}\backslash\{0\} : \exists \in \mathbb{R}^n\backslash\{0\} : Av = \lambda v\}. \tag{3.1.3}$$

Die Eigenvektoren können analog bestimmt werden.
Die folgende Tabelle zeigt, welche Formen durch welche Transformationen unter welchen Bedingungen erreichbar sind.

A	T	B
paarweise verschiedene Eigenwerte	regulär	diagonal
reell symmetrisch	orthogonal	reell diagonal
complex hermitesch	unitär	reell diagonal
normal	unitär	diagonal
beliebig reell	orthogonal	reell Block–Dreieck (Schur)
beliebig	unitär	obere Dreieck (Schur)
beliebig	regulär	Jordan

Typischerweise verwendet man QR zur Transformation auf eine „angenehmere“ Gestalt. Zu beachten ist jedoch, dass die Transformationen von links und rechts anzuwenden sind, um die Ähnlichkeit zu erhalten (also auf Zeilen und Spalten).

3.2 PARALLELE ALGORITHMEN

Zunächst benötigen wir parallele Routinen für Standard–Operationen, die wir bereits betrachtet haben.

- Vektor–Aufdatierung
- innere Produkte (Skalarprodukte)
- Matrix–Vektor- und Matrix–Matrix–Multiplikation
- Lösung von gestaffelten Systemen
- QR–Zerlegungen

Daraus ergibt sich sofort eine parallele Version der Vektoriteration:

Algorithmus 3.2.1 (Vektor–Iteration)
$x^{(0)} \neq 0$ *beliebiger Startvektor*
FOR $k = 1, 2, \ldots$ *DO*
$y^{(k)} = Ax^{(k-1)}$
$x^{(k)} = y^{(k)}/\|y^{(k)}\|$
END

Man kann obigen Algorithmus auch auf $A - \sigma I$ mit einem *shift* $\alpha \in \mathbb{R}$ anwenden. Die Parallelisierung von Algorithmus 3.2.1 ist klar.

Durch Ersetzen von $y^{(k)} = Ax^{(k-1)}$ durch

$$\text{Löse } Ay^{(k)} = x^{(k-1)}$$

(evtl. mit shift) erhält man bekanntlich die *inverse Vektoriteration*. Man bestimmt vorab parallel eine Zerlegung von A und löst dann in jedem Schritt ein gestaffeltes System.

Bei der sogenannten *simultanen Iteration* führt man die Vektoriteration parallel auf q Startvektoren aus und erhält eine Matrix $X^{(k)} \in \mathbb{R}^{n \times q}$, so dass span $(X^{(k)})$ gegen den invarianten Teilraum konvergiert, der durch die q betragsgrößten Eigenwerte von A gegeben ist, falls

$$|\lambda_q| > |\lambda_{q+1}|.$$

In Numerik I hatten wir das *QR–Verfahren* eingeführt (hier ohne shifts).

Algorithmus 3.2.2 (QR–Verfahren)
$A^{(0)} = A$
FOR $k = 1, 2, \ldots$

Bestimme QR–Zerlegung $A^{(k-1)} = Q^{(k)} R^{(k)}$
$A^{(k)} = R^{(k)} Q^{(k)}$

END

Wir hatten gezeigt, dass $A^{(k)}$ für $k \to \infty$ gegen eine obere Dreiecksmatrix konvergiert, auf deren Diagonale die Eigenwerte von A stehen.

Wir benötigen also pro Schritt eine QR–Zerlegung und eine Matrix–Matrix–Multiplikation, beides haben wir auch in paralleler Form. Der Aufwand pro Schritt ist $\Theta(n^3)$, der reduziert werden kann

- auf $\Theta(n^2)$, falls A Hessenberg–Form hat,
- auf $\Theta(n)$, falls A symmetrisch und tridiagonal ist.

Zur Erinnerung: eine Matrix $H = (h_{ij}) \in \mathbb{K}^{m\times n}$, $m \geq n$, besitzt *obere Hessenberg–Form*, wenn H die Gestalt

$$\left[\begin{array}{ccccc} h_{11} & h_{12} & h_{13} & \cdots & h_{1n} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2n} \\ 0 & h_{32} & h_{33} & \cdots & h_{3n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & h_{n,n-1} & h_{nn} \\ 0 & \cdots & \cdots & 0 & h_{n+1,n} \\ \hline & & 0 & & \end{array}\right]$$

besitzt, also wenn $h_{ij} = 0$ für alle i und j mit $j < i - 1$. Die Reduktion auf Hessenberg–Form wird mit QR gemacht und ist ein $\Theta(n^3)$-Prozess. Ist A symmetrisch, so ergibt die Hessenberg–Form automatisch eine Tridiagonal–Gestalt. Die Reduktion benötigt $\Theta(n^2)$ und jeder Iterationsschritt dann nur noch $\Theta(n)$.

Bemerkung 3.2.1 *Es gibt eine Reihe von speziellen Verfahren zu Bestimmung von Eigenwerten von symmetrischen Tridiagonalmatrizen: Bisektion, Multisektion und Divide-and-Conquer [1], 129–138, auf die wir hier jedoch nicht näher eingehen.*

Es gibt aber mittlerweile neuere Verfahren speziell für symmetrische Matrizen, die wir im Folgenden beschreiben.

3.3 DAS JACOBI–VERFAHREN

Von nun an sei $A \in \mathbb{K}^{n\times n}$ hermitesch, d.h. $A = A^*$. Wir betrachten die Zerlegung

$$A = D - R - R^* \tag{3.3.1}$$

mit $D =$ diag (A) und der oberen Dreiecks–Matrix R. Wir betrachten die *Frobenius–Norm*

$$\|A\|_F := \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}. \tag{3.3.2}$$

Man stellt dann fest, dass

$$S(A) := \|R + R^*\|_F^2 \tag{3.3.3}$$

ein Maß dafür ist, wie gut die Diagonalelemente von A die Eigenwerte von A approximieren:

Lemma 3.3.1 *Sei A hermitesch und d_{ii} ein beliebiges Diagonalelement. Dann existiert ein $\lambda \in \sigma(A)$ mit*

$$|d_{ii} - \lambda| \leq \sqrt{S(A)}. \tag{3.3.4}$$

Beweis: Nach dem Satz von Bauer und Fike (vgl. Numerik 1, Kapitel über Eigenwert–Probleme) gilt

$$|\lambda - \hat{\lambda}| \leq \|E\|_2,$$

wenn $\lambda \in \sigma(A+E)$ und $\hat{\lambda} \in \sigma(A)$ ist. Da $d_{ii} \in \sigma(D)$, wende diesen Satz auf $E = -(R+R^*)$ an, also

$$|d_{ii} - \lambda| \leq \|R + R^*\|_2 \leq \|R + R^*\|_F = \sqrt{S(A)},$$

also die Behauptung. □

Die Idee besteht nun darin, das Maß $S(A)$ mit unitären Ähnlichkeitstransformationen

$$A^{(k+1)} = (Q^{(k)})^* A^{(k)} (Q^{(k)}),\ k = 0, 1, 2, \ldots, A^{(0)} = A$$

sukzessive zu reduzieren.
Von nun an beschränken wir uns auf reell-symmetrische Matrizen $A \in \mathbb{R}^{n\times n}$.
Zur Vorbereitung:

Lemma 3.3.2 *Sei $A \in \mathbb{R}^{n\times n}$ beliebig und $Q \in \mathbb{R}^{n\times n}$ orthogonal, dann gilt:*

(a) $\|A\|_F = \|A^T\|_F$.

(b) $\|QA\|_F = \|AQ\|_F = \|A\|_F$.

Beweis: [1], 119. □

Wir bezeichnen mit

$$G(i, j, c, s)$$

die Givens–Rotation mit den Indizes i, j und den Einträgen c, s. Weiter sei

$$G(i, j, \theta) := G(i, j, \cos\theta, \sin\theta).$$

Damit gilt::

Satz 3.3.3 *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und $i, j \in \{1, \ldots, n\}$ zwei feste, aber beliebige Indizes mit $i \neq j$, $a_{ij} \neq 0$. Es sei*

$$\tau := \frac{a_{ij} - a_{ii}}{2a_{ij}}, \quad \tan\theta := \frac{\mathit{sign}\,\tau}{|\tau| + \sqrt{1+\tau^2}}, \quad |\theta| \leq \frac{\pi}{4} \tag{3.3.5}$$

und

$$G := G(i, j, \theta)$$

sowie

$$\tilde{A} := G^T A G = (\tilde{a}_{pq}).$$

Dann gilt:

(a) $\tilde{a}_{ij} = 0$.

(b) $S(\tilde{A}) = S(A) - 2a_{ij}^2$.

Beweis: [1], 119–121. □

Die Dreh–Matrizen $G(i, j, \theta)$ werden auch *Jacobi–Matrizen* genannt, daher stammt auch der Name der Verfahrens.

Satz 3.3.3 besagt lediglich, dass das Jacobi–Verfahren konvergiert, mit etwas mehr Aufwand zeigt man, dass das Verfahren quadratisch konvergiert, [3], 242–243.
Nun kann man noch die Reihenfolge der Paare (i, j) wählen.

1. *Klassische Wahl:* maximales $|a_{ij}|$.
2. *Zyklische Wahl:* Reihenfolge, bei der alle Elemente a_{ij} genau einmal zu Null gesetzt werden, z.B. $(2,1),(3,1),(3,2),(4,1),(4,2),(4,3),\ldots,(n,n-1)$, vgl. Algorithmus 3.3.1.
3. *Threshold–Verfahren:* Wie bei 2., jedoch wird die Rotation nur dann ausgeführt, wenn

$$|a_{ij}| > \text{thresh}$$

mit einem Threshold–Parameter.

Algorithmus 3.3.1 (Jacobi–Verfahren, zeilenzyklisch, ein Durchgang)
FOR $i = 2$ *TO* n *DO*
  FOR $j = 1$ *TO* $i-1$ *DO*
    IF $a_{ij} \neq 0$ *THEN*
      Berechne $\tan\theta$ *gemäß (3.3.5)*
      Berechne $A := G(i,j,\theta)^T A G(i,j,\theta)$
        $Q := QG(i,j,\theta)$
    END
  END
END

Nun zur Parallelisierung. Wir müssen die Parameter–Paare so trennen, dass diese parallel verarbeitet werden können, also

$$\{i,j\} \cap \{p,q\} = \emptyset.$$

In diesem Fall werden unterschiedliche Zeilen bzw. Spalten verändert, also können die betreffenden Operationen unabhängig voneinander ausgeführt werden. Wir können maximal

$$\frac{n}{2}$$

verschiedene Index–Paare finden. Also sei o.B.d.A. n gerade.
Man sucht also $n-1$ Mengen von $\frac{n}{2}$ Index–Paaren, um einen möglichst hohen Parallelisierungsgrad zu erzeilen. Dies leistet der sogenannte *Karussell–Algorithmus*.

Beispiel: für $n = 8$

Schritt 1:

$$\begin{array}{ccccccc} 1 & & 3 & \to & 5 & \to & 7 \\ & \nearrow & & & & & \downarrow \\ 2 & \leftarrow & 4 & \leftarrow & 6 & \leftarrow & 8 \end{array}$$

$\Rightarrow (2,1),(4,3),(6,5),(8,7)$

Schritt 2:

$$\begin{array}{ccccccc} 1 & & 2 & \to & 3 & \to & 5 \\ & \nearrow & & & & & \downarrow \\ 4 & \leftarrow & 6 & \leftarrow & 8 & \leftarrow & 7 \end{array}$$

$\Rightarrow (4,1),(6,2),(8,3),(7,5)$

Schritt 3:

$$\begin{array}{ccccccc} 1 & & 4 & \to & 2 & \to & 3 \\ & \nearrow & & & & & \downarrow \\ 6 & \leftarrow & 8 & \leftarrow & 7 & \leftarrow & 5 \end{array}$$

$\Rightarrow (6,1),(8,4),(7,2),(5,3)$

Schritt 4:

$$\begin{array}{ccccccc} 1 & & 6 & \to & 4 & \to & 2 \\ & \nearrow & & & & & \downarrow \\ 8 & \leftarrow & 7 & \leftarrow & 5 & \leftarrow & 3 \end{array}$$

$\Rightarrow (8,1),(7,6),(5,4),(3,2)$

Schritt 5:

$$\begin{array}{ccccccc} 1 & & 8 & \to & 6 & \to & 4 \\ & \nearrow & & & & & \downarrow \\ 7 & \leftarrow & 5 & \leftarrow & 3 & \leftarrow & 2 \end{array}$$

$\Rightarrow (7,1),(8,5),(6,3),(4,2)$

Schritt 6:

$$\begin{array}{ccccccc} 1 & & 7 & \to & 8 & \to & 6 \\ & \nearrow & & & & & \downarrow \\ 5 & \leftarrow & 3 & \leftarrow & 2 & \leftarrow & 4 \end{array}$$

$\Rightarrow (5,1),(7,3),(8,2),(6,4)$

Schritt 7:

$$\begin{array}{ccccccc} 1 & & 5 & \to & 7 & \to & 8 \\ & \nearrow & & & & & \downarrow \\ 3 & \leftarrow & 2 & \leftarrow & 4 & \leftarrow & 6 \end{array}$$

$\Rightarrow (3,1),(5,2),(7,4),(8,6)$

Schritt 8:

$$\begin{array}{ccccccc} 1 & & 3 & \to & 5 & \to & 7 \\ & \nearrow & & & & & \downarrow \\ 2 & \leftarrow & 4 & \leftarrow & 6 & \leftarrow & 8 \end{array}$$

$\Rightarrow (2,1),(4,3),(6,5),(8,7)$

Algorithmus 3.3.2
new_top$(1) =$ *top*(1), *new_top*$(2) =$ *bottom*(1).
new_bottom$(n/2) =$ *top*$(n/2)$.
FOR $k = 3$ *TO* $n/2$
    new_top$(k) =$ *top*$(k-1)$.
FOR $k = 1$ *TO* $n/2 - 1$
    new_bottom$(k) =$*bottom*$(k+1)$.
top = new_top, bottom = new_bottom.

Initialisierung der Felder vor dem ersten Aufruf durch:
top $= (1, 3, \ldots, n-1)$, *bottom* $= (2, 4, \ldots, n)$.

Bemerkung 3.3.4 *Bei dieser Reihenfolge der Indizes ist die Konvergenz* nicht *gesichert.*

Die Parallelisierung auf $P = \frac{n}{2}$ Prozessoren ist nun klar, jeder Prozessor bearbeitet zwei Spalten von A und Q. Wir erhalten

Algorithmus 3.3.3 (Paralleles Jacobi–Verfahren für Prozesor P_k)
Repeat
    FOR $s = 1$ *TO* $n-1$
        $p =$ *top*(k), $a =$ *bottom*(k)
        Berechne $\tan\theta_k$ *für das Paar* (p,q)
        aus den entsprechenden Elementen von A_{loc}.
        Berechne $A_{\mathrm{loc}} := A_{\mathrm{loc}} J_k$, $Q_{\mathrm{loc}} := Q_{\mathrm{loc}} J_k$.
        Verschicke $\tan\theta_k$ *an alle Prozessoren.*
        FOR $m = 1$ *TO* $n/2$

IF $m \neq k$ *empfange* $\tan\theta_k$ *von Prozessor* P_m.
$p' =$ *top*(m), $q' =$ *bottom*(m)
$(A_{\mathrm{loc}})_{\substack{i=p',q' \\ j=1,2}} = J_m^T (A_{\mathrm{loc}})_{\substack{i=p',q' \\ j=1,2}}$

Ringtausch $(A_{\mathrm{loc}}, Q_{\mathrm{loc}})$
Karussell (top, bottom)

UNTIL $\|A\|'_F < \epsilon$

Die Prozedur Ringtausch gibt die Spalten auf die gleiche Weise auf verschiedene Prozessoren weiter wie im Karussell in Algorithmus 3.3.2.

Der Rechenaufwand pro Prozessor und Schleife ist $\mathcal{O}(n)$ mit $\frac{n}{2}$ Kommunikations–Operationen. Dieses schlechte Verhältnis lässt sich durch Übertragung von mehr Spalten je Prozessor verbessern.

4 ∗ Eine Anwendung: Turbulente Nischen–Strömungen

Wir wollen im abschließenden Kapitel ein Beispiel betrachten, zu dessen numerischer Simulation wir nahezu alle zuvor besprochenen Komponenten zusammenfügen möchten. Wie bereits im Vorwort erwähnt, enthält dieses Kapitel zusätzlichen Stoff.

4.1 Die Navier–Stokes–Gleichungen

Zunächst behandeln wir *laminare* Strömungen *viskoser, inkompressibler* Fluide, die durch die sogenannten Navier–Stokes–Gleichungen mathematisch beschrieben werden. Das Existenz–Problem für diese Gleichungen ist eines der 10 Claymath–Probleme.

Sei $\Omega \subset \mathbb{R}^N$, $N = 2, 3$, das Strömungsgebiet, welches bezüglich der Zeit $t \in [0, t_{\text{end}}] =: I$ zunächst unabhängig sei. Bei örtlich konstanter Dichte

$$\rho(x, t) = \rho_\infty = \text{const.}$$

wird die (inkompressible) Strömung durch folgende Größen charakterisiert

- $\boldsymbol{u} : \Omega \times I \to \mathbb{R}^N$ Geschwindigkeitsfeld,
- $p : \Omega \times I \to \mathbb{R}$ Druck.

Die Navier–Stokes–Gleichungen sind ein System partieller Differentialgleichungen und lauten

$$\begin{aligned} \frac{\partial}{\partial t}\boldsymbol{u} - \frac{1}{Re}\Delta\boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p &= \boldsymbol{g}, \text{ (Impulsgleichung)} && (4.1.1)\\ \text{div } \boldsymbol{u} &= 0, \text{ (Kontinuitätsgleichung)}, && (4.1.2) \end{aligned}$$

wobei $Re \in \mathbb{R}$ die sogenannte dimensionslose *Reynolds–Zahl* und $\boldsymbol{g} : \Omega \times I \to \mathbb{R}^N$ die gegebene äußere Kraft ist (z.B. die Erdanziehung). Grob gesprochen: Je größer Re ist, je turbulenter ist die Strömung, je kleiner, desto viskoser.
Zunächst wollen wir (4.1.1) richtig verstehen. Da $\boldsymbol{u}$ ein Vektorfeld ist, sind (4.1.1) N Gleichungen. Die Terme

$$\frac{\partial}{\partial t}\boldsymbol{u} = \left(\frac{\partial}{\partial t}u_1, \ldots, \frac{\partial}{\partial t}u_N\right)^T$$

für $\boldsymbol{u} = (u_1, \ldots, u_N)^T$ sowie

$$\begin{aligned} \Delta\boldsymbol{u} &= (\Delta u_1, \ldots, \Delta u_N)^T,\\ \nabla p &= \left(\frac{\partial}{\partial x_1}p, \ldots, \frac{\partial}{\partial x_N}p\right)^T,\\ \boldsymbol{g} &= (g_1, \ldots, g_N)^T \end{aligned}$$

sind relativ einfach. Die Kurz–Schreibweise $(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}$ für den nicht-linearen Teil bedeutet Folgendes:

$$(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = \sum_{j=1}^{N} u_j \frac{\partial}{\partial x_j}\boldsymbol{u} = \left(\sum_{j=1}^{N} u_j \frac{\partial}{\partial x_j}u_i\right)_{i=1,\ldots N},$$

also z.B. für $N = 2$

$$(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} = \begin{pmatrix} u_1 \frac{\partial}{\partial x_1} u_1 + u_2 \frac{\partial}{\partial x_2} u_1 \\ u_1 \frac{\partial}{\partial x_1} u_2 + u_2 \frac{\partial}{\partial x_2} u_2 \end{pmatrix}.$$

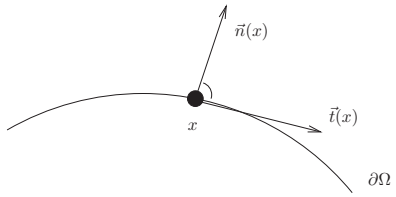
Der Divergenz–Operator ist wie üblich definiert

$$\text{div } \boldsymbol{u} = \nabla \cdot \boldsymbol{u} = \sum_{i=1}^{N} \frac{\partial}{\partial x_i} u_i.$$

Wir benötigen noch Anfangs- und Randbedingungen. Typischerweise setzt man

$$\boldsymbol{u}(x, 0) = \boldsymbol{u}_0(x), \quad x \in \Omega, \tag{4.1.3}$$

für eine gegebene Anfangs–Strömung $\boldsymbol{u}_0 : \Omega \to \mathbb{R}^N$. Für die Randbedingungen benötigen wir noch einige Bezeichnungen.



Für $x \in \partial\Omega$ bezeichne $\vec{n}(x)$ den Normal- und $\vec{t}(x)$ den Tangential–Vektor. Damit seien

- φ_n die Geschwindigkeitskomponente senkrecht zu $\partial\Omega$,
- φ_t die Geschwindigkeitskomponente parallel zu $\partial\Omega$,
- $\frac{\partial \varphi_n}{\partial n}, \frac{\partial \varphi_t}{\partial n}$ die jeweilige Ableitung in Normalenrichtung.

Für jedes $x \in \partial\Omega$ sind folgende Randbedingungen möglich:

1.) **Haftbedingungen (no-slip):** $\varphi_n(x) = \varphi_t(x) = 0,\ x \in \partial\Omega$.

2.) **Rutschbedingungen (free slip):** $\varphi_n(x) = 0,\ \frac{\partial \varphi_t}{\partial n}(x) = 0,\ x \in \partial\Omega$.

3.) **Einströmbedingungen (inflow):** $\varphi_n(x) = \varphi_n^0,\ \varphi_t(x) = \varphi_t^0$ mit gegebenen φ_n^0, φ_t^0.

4.) **Ausströmbedingungen (outflow):** $\frac{\partial \varphi_n}{\partial n}(x) = \frac{\partial \varphi_t}{\partial n}(x) = 0,\ x \in \partial\Omega$.

5.) **Periodische Randbedingung:** Bei Achsen-parallelen Problemen sind φ_n, φ_t und p periodisch.

Bemerkung 4.1.1 *Bei Dirichlet–Bedingungen $\boldsymbol{u} = \boldsymbol{f}$ auf $\partial\Omega$ ergibt sich eine Zusatz–Bedingung an $\boldsymbol{f}$. Aus dem Gauß'schen Integralsatz folgt nämlich*

$$\int_{\partial\Omega} \boldsymbol{f} \cdot \vec{n}\, ds = \int_{\partial\Omega} \boldsymbol{u} \cdot \vec{n}\, ds = \int_{\Omega} \text{div } \boldsymbol{u}\, dx = 0$$

aufgrund der Kontinuitätsgleichung (4.1.2).

Beispiel 4.1.2 *Wir betrachten in 2D die Nischen–Strömung, auch bekannt als* Driven Cavity. *Hier ist*

$$\Omega = [0, 1]^2$$

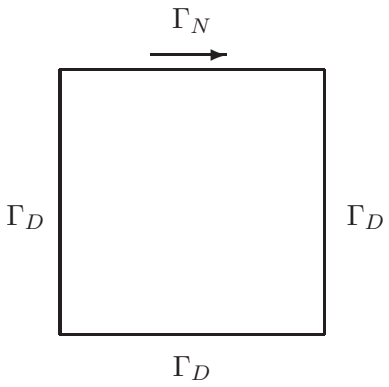
und

$$\boldsymbol{u} = 0 \ \textit{auf}\ \Gamma_D := \partial\Omega \cap \{\{x = 0\} \cup \{x = 1\} \cup \{y = 1\}\}$$

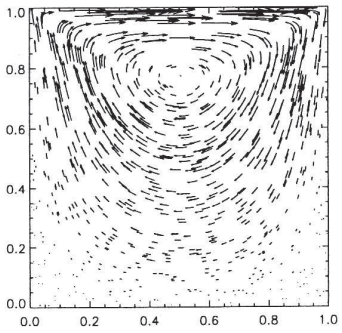
sowie

$$\boldsymbol{u} = \bar{u} \cdot \vec{e}_1 \ \textit{auf}\ \Gamma_N := \partial\Omega \cap \{y = 1\}$$

mit vorgegebenem $\bar{u} \in \mathbb{R}$ *und* $\vec{e}_1 = (1, 0)^T$.



Man beachte, dass dieses Problem streng genommen nicht korrekt gestellt ist, da die Randbedingungen an den Punkten $(0, 1)$ *und* $(1, 1)$ *springen. Die Vorstellung besteht aus einem mit Flüssigkeit gefüllten Topf über den ein Band mit fester Geschwindigkeit gezogen wird.*



4 Wände mit Haftbedingungen.
Die obere Wand bewegt sich mit konstanter Geschwindigkeit nach rechts,
d.h. $\varphi_t = \varphi_t^0 = \text{const.}$ am oberen Rand, im Gegensatz zu $\varphi_t = 0$ bei gewöhnlichen Haftbedingungen.

Abbildung 4.1: Nischenströmung (Geschwindigkeitsfeld), aus [2].

4.2 MODELLIERUNG: DIE HERLEITUNG DER NAVIER–STOKES–GLEICHUNGEN

Wissenschaftliches Rechnen bezeichnet die interdisziplinäre Verknüpfung von Modellierung, Analyse, Simulation und Visualisierung. Wir zeigen nun ein Beispiel für die Modellierung, also der Übersetzung eines Anwendungsproblems in mathematische Gleichungen. Oftmals kann die Modellierung nur interdisziplinär erfolgreich gestaltet werden, wenn z.B. Ingenieure, Mediziner oder Naturwissenschaftler einerseits, Informatiker und Mathematiker andererseits, zusammenarbeiten.
In diesem Abschnitt wollen wir kurz die Herkunft der Navier–Stokes–Gleichungen für inkompressible Strömungen erläutern. Detaillierte Herleitungen findet man in Lehrbüchern über Strömungsmechanik oder mathematische Physik. An diesen Arbeiten haben wir uns hier orientiert.
Grob gesagt, resultieren die Navier–Stokes–Gleichungen aus dem dritten Newtonschen Gesetz und dem Prinzip der Massenerhaltung. Wir werden dies nun näher erläutern.
Zunächst stellt sich die Frage nach einem geeigneten Koordinatensystem zur Beschreibung einer instationären Strömung. Eine Möglichkeit besteht darin, das Koordinatensystem mit den strömenden

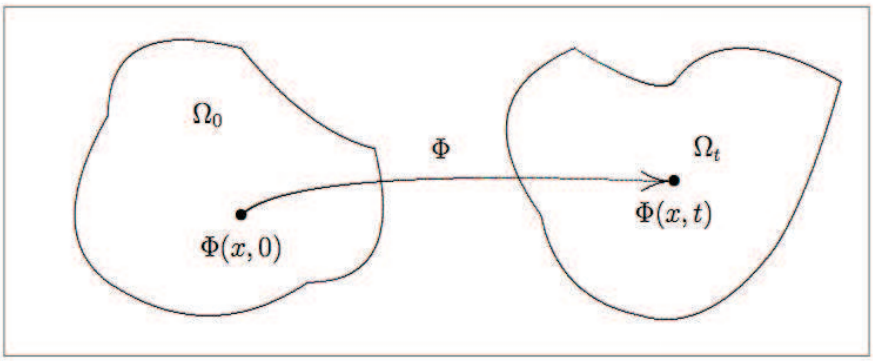


Abbildung 4.2: Bewegung eines Teilchens in lokaler Darstellung.

Teilchen mitzubewegen. Dies sind die *Lagrange–Koordinaten* oder auch die *substantielle Darstellung*. Für unsere Zwecke ist eine ortsfeste Beschreibung der Strömung geeigneter. Diese *lokale Darstellung* ist vor allem unter dem Namen *Euler–Koordinaten* bekannt. Hierbei sei

$$\Phi : \Omega \times [0,\infty) \to \Omega \tag{4.2.1}$$

die Koordinatenfunktion, die einem Teilchen, das sich zum Zeitpunkt $t = 0$ am Ort $x = \Phi(x,0) \in \Omega$ befunden hat, seine Position zu jeder Zeit $t \in [0,\infty)$ zuordnet. Die Geschwindigkeit $\boldsymbol{u}$ ist bekanntermaßen durch die Ableitung des Weges nach der Zeit gegeben, d. h.

$$\boldsymbol{u}\big(\Phi(x,t),t\big) = \frac{d}{dt}\Phi(x,t). \tag{4.2.2}$$

Um nun die physikalischen Relationen beschreiben zu können, betrachten wir instationäre Volumina. Sei $\Omega_0 \subset \Omega$ eine Menge von Teilchen zur Zeit $t = 0$. Dann ist

$$\Omega_t := \Phi(\Omega_0,t) = \{\Phi(x,t) \in \Omega : \ x \in \Omega_0\} \tag{4.2.3}$$

das Bild von Ω_0 unter der Abbildung $\Phi(\cdot,t)$. Die Abbildung 4.2 verdeutlicht diesen Sachverhalt. Mit der *Dichte* ρ ist dann die *Masse* durch den Ausdruck

$$m(t) := \int_{\Omega_t} \rho(x,t)\, dx \tag{4.2.4}$$

gegeben. Das Prinzip der Massenerhaltung besagt, daß die Masse konstant ist, also folgt

$$\frac{d}{dt}m(t) = 0. \tag{4.2.5}$$

Wir haben nun das Problem, das Integral über einem instationären Volumen berechnen zu müssen. Dabei hilft der folgende Satz:

Satz 4.2.1 (Transportsatz) *Für* $h(\cdot,\cdot) \in C^1\big(\Omega \times [0,\infty)\big)$ *gilt die Beziehung*

$$\frac{d}{dt}\int_{\Omega_t} h(x,t)\, dx \ = \ \int_{\Omega_t} \Big(\frac{\partial}{\partial t}h + \operatorname{div}\,(h\boldsymbol{u})\Big)(x,t)\, dx. \tag{4.2.6}$$

Zum Beweis des Transportsatzes benötigen wir eine Vorbereitung. Sei

$$A : I \to \mathbb{R}^{n\times n}\ ,\ I \subset \mathbb{R},$$

eine Matrix-wertige Funktion. Man nennt eine Matrix-Funktion $Y : I \to \mathbb{R}^{n\times n}$ *Wronski–Matrix*, falls

$$\frac{d}{dt}Y(t) = A(t)Y(t). \tag{4.2.7}$$

Also enthält $Y(t)$ als Spalten die Fundamental–Lösungen der homogenen, linearen Differentialgleichung

$$y'(t) = A(t)y(t).$$

Für die Determinante, die *Wronski–Determinante* gilt

Lemma 4.2.2 *Falls $Y(t)$ (4.2.7) erfüllt, gilt*

$$\frac{d}{dt}(\det Y(t)) = (tr\ A(t))(\det Y(t)) \tag{4.2.8}$$

mit der Spur $tr\ A := \sum_{i=1}^{n} a_{ii}$.

Beweis: Es sei $Y_i(t) := (y_{i,1}(t), \ldots, y_{i,n}(t))$ die i-te Zeile von $Y(t)$, dann gilt mit (4.2.7)

$$\frac{d}{dt}Y_i(t) = \sum_{j=1}^{n} a_{i,j}(t)Y_j(t),\ i = 1, \ldots, n. \tag{4.2.9}$$

Wir benutzen nun die allgemeine Definition der Determinanten

$$\det A = \sum_{\sigma\in s_n} (\text{sgn}\ \sigma)a_{1,\sigma_1} \ldots a_{n,\sigma_n}$$

und wenden die Produktregel an

$$\begin{aligned}
\frac{d}{dt}\det Y(t) &= \sum_{i=1}^{n}\sum_{\sigma\in s_n} (\text{sgn}\ \sigma)y'_{i,\sigma_i}(t)\prod_{\substack{j=1\\ j\neq i}}^{n} y_{j,\sigma_j}(t)\\
&= \sum_{i=1}^{n} \det(Y_1(t), \ldots, Y_{i-1}(t), \frac{d}{dt}Y_i(t), Y_{i+1}(t), \ldots, Y_n(t))^T\\
&\overset{(4.2.9)}{=} \sum_{i,j=1}^{n} a_{i,j}(t)\underbrace{\det(Y_1(t), \ldots, Y_{i-1}(t), Y_j(t), Y_{i+1}(t), \ldots, Y_n(t))^T}_{=0 \text{ für alle } j\neq i}\\
&= \sum_{i=1}^{n} a_{i,i}(t)\det Y(t) = (tr\ A(t))(\det Y(t)).
\end{aligned}$$

□

Zum Beweis des Transportsatzes wenden wir das Lemma wie folgt an: Aus (4.2.2) folgt mit der Kettenregel

$$\frac{d}{dx}\frac{d}{dt}\Phi(x,t) = \boldsymbol{u}_x(\Phi(x,t),t)\ \Phi_x(x,t),$$

also

$$\frac{d}{dt}\underbrace{\Phi_x(x,t)}_{=Y(t)} = \underbrace{\boldsymbol{u}_x(\Phi(x,t),t)}_{A(x)}\Phi_x(x,t)$$

und damit für $J(x,t) = \det \Phi_x(x,t)$

$$\begin{aligned}
\frac{d}{dt}J(x,t) &= (tr\ \boldsymbol{u}_x(\Phi(x,t))J(x,t)\\
&= (\text{div}\ \boldsymbol{u}(\Phi(x,t))J(x,t).
\end{aligned} \tag{4.2.10}$$

Beweis von Satz 4.2.1: Wir verwenden die Transformation $\Omega_t = \Phi(\Omega_0, t)\ x(t) = \Phi(\hat{x}, t)$ in (4.2.3) und erhalten

$$\int\limits_{\Omega_t} h(x,t)dt = \int\limits_{\Omega_0} h(\Phi(\hat{x},t),t)J(\hat{x},t)d\hat{x}$$

mit $J(\hat{x},t) = \det(\Phi(\hat{x},t)$ wie oben. Dann gilt mit der Produktregel

$$\begin{aligned}
\frac{d}{dt}\int\limits_{\Omega_t} h(x,t)dx &= \int\limits_{\Omega_0} \frac{\partial}{\partial t} h(\Phi(\hat{x},t),t)J(\hat{x},t)d\hat{x} \\
&\qquad + \int\limits_{\Omega_0} h(\Phi(\hat{x},t),t)\frac{\partial}{\partial t}J(\hat{x},t)dt \\
&= \int\limits_{\Omega_0} \left(h_t(\Phi(\hat{x},t),t) + \nabla h(\Phi(\hat{x},t),t) \cdot \frac{d}{dt}\Phi(\hat{x},t) \right) J(\hat{x},t)d\hat{x} \\
&\qquad + \int\limits_{\Omega_0} h(\Phi(\hat{x},t),t)(\text{div } \boldsymbol{u}(\Phi(\hat{x},t),t)J(\hat{x},t)d\hat{x}
\end{aligned}$$

mit (4.2.10). Fassen wir dies zusammen, erhalten wir mit $\frac{d}{dt}\Phi = \boldsymbol{u}$

$$\begin{aligned}
\frac{d}{dt}\int\limits_{\Omega_t} h(x,t)dx &= \int\limits_{\Omega_0} (h_t + \underbrace{\nabla h \cdot \boldsymbol{u} + h \text{ div } \boldsymbol{u}}_{\text{div } (h\boldsymbol{u})})(\Phi(\hat{x},t),t)J(\hat{x},t)d\hat{x} \\
&= \int\limits_{\Omega_t} \left(\frac{\partial}{\partial t} h + \text{ div } (h\boldsymbol{u}) \right)(x,t)dx
\end{aligned}$$

wiederum mit obiger Transformation. □

Die Glattheitsbedingung im Transportsatz wird hier stets als gültig vorausgesetzt. Wenn also die Dichte als Funktion des Ortes und der Zeit genügend glatt ist, folgt aus dem Prinzip der Erhaltung der Masse $(4.2.5)$:

$$0 \;=\; \frac{d}{dt}m(t) \;=\; \frac{d}{dt}\int_{\Omega_t} \rho(x,t)\ dx \;=\; \int_{\Omega_t} \Big(\frac{\partial}{\partial t}\rho + \text{div } (\rho\boldsymbol{u}) \Big)(x,t)\ dx. \tag{4.2.11}$$

Wenn man zusätzlich noch annimmt, daß der Integrand in $(4.2.11)$ glatt ist, folgt

$$\frac{\partial}{\partial t}\rho + \text{div } (\rho\boldsymbol{u}) \;=\; 0. \tag{4.2.12}$$

Dieses Prinzip nennt man *Lokalisierung*, und $(4.2.12)$ ist als *Kontinuitätsgleichung* bekannt. Inkompressible Strömungen sind dadurch gekennzeichnet, daß die Dichte räumlich und zeitlich konstant ist. Somit erhalten wir in diesem Fall aus $(4.2.12)$ die Gleichung

$$\text{div } \boldsymbol{u} \;=\; 0. \tag{4.2.13}$$

Impulsgleichung
Das dritte Newtonsche Gesetz besagt allgemein, daß die Impulsänderung eines materiellen Körpers gleich der Summe der an ihn angreifenden Kräfte ist. Diese physikalischen Größen führen wir zunächst ein. Der *Impuls* ist durch

$$\boldsymbol{Imp}(t) := \int_{\Omega_t} \big(\rho\boldsymbol{u}\big)(x,t)\ dx \tag{4.2.14}$$

definiert. Man nimmt für die Modellierung der *Kraft* $\boldsymbol{F}$ stets an, daß sich diese in eine *Volumenkraft* $\boldsymbol{F}_V$ (wie z. B. die Gravitationskraft) und eine *Randkraft* $\boldsymbol{F}_R$ aufteilen läßt. Dabei sei die Volumenkraft durch

$$\boldsymbol{F}_V(t) := \int_{\Omega_t} \big(\rho \boldsymbol{f}\big)(x,t)\, dx \tag{4.2.15}$$

gegeben, wobei $\boldsymbol{f}$ ein gegebenes äußeres Kraftfeld (wie z. B. das Erdschwerefeld) ist. Die Randkraft ist definiert durch

$$\boldsymbol{F}_R(t) := \int_{\partial\Omega_t} \big(\underline{\boldsymbol{\sigma}}\, \boldsymbol{n}\big)(x,t)\, dS, \tag{4.2.16}$$

wobei $\boldsymbol{n}$ der äußere Normalenvektor von $\partial\Omega_t$ und $\underline{\boldsymbol{\sigma}}$ der *Spannungstensor*

$$\underline{\boldsymbol{\sigma}} := \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)_{i,j=1,\ldots,n} \tag{4.2.17}$$

ist. Der Spannungstensor liefert eine vollständige Beschreibung des Spannungszustandes eines Körpers. Damit besagt das dritte Newtonsche Gesetz in unserem Fall

$$\frac{d}{dt}\int_{\Omega_t} \big(\rho \boldsymbol{u}\big)(x,t)\, dx \;=\; \int_{\Omega_t} \big(\rho \boldsymbol{f}\big)(x,t)\, dx + \int_{\partial\Omega_t} \big(\underline{\boldsymbol{\sigma}}\, \boldsymbol{n}\big)(x,t)\, dS. \tag{4.2.18}$$

Wir betrachten diesen vektorwertigen Ausdruck nun komponentenweise. Nehmen wir die Glattheit von $\rho\boldsymbol{u}$ an, so können wir auf die Komponenten der linken Seite von $(4.2.18)$ den Transportsatz anwenden und erhalten:

$$\frac{d}{dt}\int_{\Omega_t} \big(\rho u_j\big)(x,t)\, dx \;=\; \int_{\Omega_t} \Big(\frac{\partial}{\partial t}(\rho u_j) + \operatorname{div}\,(\rho u_j \boldsymbol{u})\Big)(x,t)\, dx. \tag{4.2.19}$$

Mit Hilfe der Kettenregel folgt hieraus für inkompressible Fluide (d. h. $\rho \equiv const.$)

$$\frac{d}{dt}\int_{\Omega_t} \big(\rho u_j\big)(x,t)\, dx \;=\; \int_{\Omega_t} \Big(\rho\frac{\partial}{\partial t} u_j + \rho \operatorname{div}\,(u_j \boldsymbol{u})\Big)(x,t)\, dx. \tag{4.2.20}$$

Die Komponenten des zweiten Integrals auf der rechten Seite von $(4.2.18)$ formen wir zunächst mit dem Gaußschen Integralsatz um:

$$\int_{\partial\Omega_t} \big(\boldsymbol{\sigma}_j\, \boldsymbol{n}\big)(x,t)\, dS \;=\; \int_{\Omega_t} \operatorname{div}\, \boldsymbol{\sigma}_j(x,t)\, dx, \tag{4.2.21}$$

wobei $\boldsymbol{\sigma}_j$ die j–te Zeile des Spannungstensors bezeichnet. Setzen wir alles zusammen, so erhalten wir aus $(4.2.18)$, $(4.2.20)$ und $(4.2.21)$

$$\int_{\Omega_t} \Big(\rho\frac{\partial}{\partial t} u_j + \rho \operatorname{div}\,(u_j \boldsymbol{u})\Big)(x,t)\, dx \;=\; \int_{\Omega_t} \Big((\rho f_j) + \operatorname{div}\, \boldsymbol{\sigma}_j\Big)(x,t)\, dx \tag{4.2.22}$$

und wiederum mit Lokalisierung

$$\rho\frac{\partial}{\partial t} u_j + \rho \operatorname{div}\,(u_j \boldsymbol{u}) \;=\; \rho f_j + \operatorname{div}\, \boldsymbol{\sigma}_j \tag{4.2.23}$$

oder in Vektorenschreibweise

$$\rho\frac{\partial}{\partial t} \boldsymbol{u} + \rho\,(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} \;=\; \rho\boldsymbol{f} + \operatorname{div}\, \underline{\boldsymbol{\sigma}}. \tag{4.2.24}$$

Der Ausdruck „$\operatorname{div}\, \underline{\boldsymbol{\sigma}}$“ heißt in der Physik *Tensordivergenz*, und für diese Größe kann man folgende Beziehung zum Druck p und dem Inversen ν der Reynoldszahl zeigen:

$$\int_{\Omega_t} \operatorname{div}\, \underline{\boldsymbol{\sigma}} \;=\; -\int_{\Omega_t} \operatorname{grad}\, p(x,t)\, dx + \nu\,\rho \int_{\Omega_t} \Delta\boldsymbol{u}(x,t)\, dx. \tag{4.2.25}$$

Wiederum durch Lokalisation erhalten wir aus $(4.2.24)$ und $(4.2.25)$ die Gleichung

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - \nu\Delta\boldsymbol{u} + \frac{1}{\rho}\operatorname{grad}\, p = \boldsymbol{f}, \tag{4.2.26}$$

die zusammen mit der Kontinuitätsgleichung $(4.2.13)$ und den entsprechenden Rand- und Anfangsbedingungen das Navier–Stokes–Problem für inkompressible Strömungen bildet.

4.3 FINITE–DIFFERENZEN–DISKRETISIERUNG

Wir nehmen zunächst an, dass Ω ein rechteckiges Gebiet ist

$$\Omega = [0, a] \subset \mathbb{R}^N,\ a = (a_1, \ldots, a_N)^T \in \mathbb{R}^N,$$

über das wir ein regelmäßigs Gitter Δ_h mit Gitterweite $h = (h_1, \ldots, h_N)^T$ und

$$h_i := \frac{a_i}{n_i},$$

legen. Wir diskretisieren $\Delta \boldsymbol{u}$ bezüglich Δ_h mit zentralen Differenzen wie üblich. Der nichtlineare Term $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$ ist ein *Transport-* oder *Konvektions–Term*, der mit besonderen Methoden diskretisiert werden muss. Dazu betrachten wir das 1D–Besipiel

$$\begin{cases} -u''(x) + ku'(x) = f(x), \quad x \in (0,1), \\ u(0) = u(1) = 0, \end{cases} \tag{4.3.1}$$

eine *Konvektions–Diffusions–Gleichung*. Um weiterhin eine symmetrische Steifigkeitsmatrix zu erhalten, verwendet man für $u'(x)$ den zentralen Differenzenquotienten

$$u'(x_i) \approx \frac{1}{2\Delta x}(u(x_i + \Delta x) - u(x_i - \Delta x)). \tag{4.3.2}$$

Falls k groß ist $(k \gg 1)$, nennt man (4.3.1) ein *konvektions-dominiertes* Problem. In diesem Fall muss

$$\Delta x \leq \frac{2}{|k|} \tag{4.3.3}$$

gewählt werden, damit das Problem stabil gelöst werden kann. Dies kann man anhand der Eigenwerte der Steifigkeitsmatrix sehen.

Aufgabe:

(a) Wie lauten die Einträge der Steifigkeitsmatrix von (4.3.1) mit zentralen Differenzenquotienten?

(b) Bestimmen Sie die Eigenwerte.

(c) Welche Bedeutung hat Bedingung (4.3.3)?

Offenbar führt (4.3.3) (besonders in hohen Dimensionen) zu extrem großen Systemen.

Eine Möglichkeit zur stabilen Approximation konvektiver Terme ist das *Donor–Cell-Schema*. Dabei sei k nun eine Funktion $k = k(x)$, die an den Mittelpunkten der Intervalle gegeben ist. Dies ist analog zur Fluß–Diskretisierung auf einem versetzten Gitter („staggered grid“) bei Problemen mit variablen Koeffizienten.

Man beachtet dann das Vorzeichen und setzt

$$\left[\frac{d}{dx}(ku)\right]_i^{dc} := \frac{1}{\Delta x}(k_r u_r - k_\ell u_\ell) \tag{4.3.4}$$

mit

$$u_r := \begin{cases} u_i, & k_r > 0, \\ u_{i+1}, & k_\ell < 0, \end{cases} \qquad u_\ell := \begin{cases} u_{i-1}, & k_\ell > 0, \\ u_i, & k_\ell < 0, \end{cases} \tag{4.3.5}$$

oder ohne die Fallunterscheidung

$$\left[\frac{d}{dx}(ku)\right]_i^{dc} = \frac{1}{2\Delta x}\{k_r(u_i+u_{i+1}) - k_\ell(u_{i-1}+u_i) + |k_r|(u_i-u_{i+1}) - |k_\ell|(u_{i-1}-u_i)\}. \tag{4.3.6}$$

4.4 DISKRETISIERUNG DES DRUCKS

Bei den Navier–Stoks–Gleichungen haben wir es mit einem *System* zu tun, bei dem die Variablen $\boldsymbol{u}$ und p gekoppelt sind. Dies hat u.a. auch zur Folge, dass wir die beiden Variablen nicht beliebig diskretisieren dürfen, die Diskretisierungen von $\boldsymbol{u}$ und p müssen zueinander passen.

Wir zeigen dies am 2D–Beispiel mit homogenen Dirichlet–Bedingungen und $\boldsymbol{g} \equiv 0$. Dann ist die Lösung

$$\boldsymbol{u} \equiv 0, \quad p = \text{ const.}$$

Verwendet man zentrale Differenzen für $\boldsymbol{u}$ und p auf Δ_h, dann löst

$$\boldsymbol{u}_h \equiv 0, \quad p_{i,j} = \begin{cases} P_1, & \text{für } i+j \text{ gerade}, \\ P_2, & \text{für } i+j \text{ ungerade} \end{cases}$$

mit beliebigen $P_1, P_2 \in \mathbb{R}$ das diskrete Problem, denn

$$\nabla p(x_{ij}) \approx \frac{1}{2\Delta x}\{p_{i+1,j} - p_{i-1,j} + p_{i,j+1} - p_{i,j-1}\} = 0$$

da die Summe der Indizes $i+j \pm 1$ in allen vier Fällen entweder gerade oder ungerade ist. Also kann der Druck beliebig stark oszillieren.

+	−	+	−	+	−
−	+	−	+	−	+
+	−	+	−	+	−
−	+	−	+	−	+
+	−	+	−	+	−
−	+	−	+	−	+

Daher betrachtet man, wie bei variablen Koeffizienten, für den Druck ein vesetztes Gitter (engl. *staggered grid*), bei dem der Druck an den Mittelpunkten der Zellen gesucht wird, sowie die erste und zweite Geschwindigkeitskomponente vertikal bzw. horizontal versetzt sind.

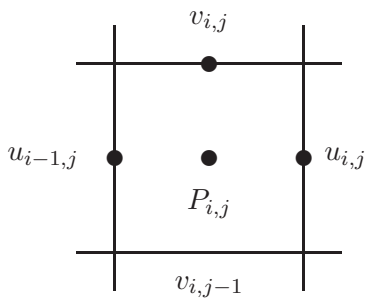
4.5 VOLLSTÄNDIGE ORTS–DISKRETISIERUNG

Nun können wir die Diskretisierung bezüglich des Ortes $x \in \Omega$ kompletieren. Die Kontinuitätsgleichung (4.1.2) wird an den Zell–Mittelpunken diskretisiert

$$\frac{1}{\Delta x}(u_{i,j} - u_{i-1,j}) + \frac{1}{\Delta y}(v_{i,j} - v_{i-1,j}), \tag{4.5.1}$$

wenn $h = (\Delta x, \Delta y)$ und $\boldsymbol{u} = (u, v)$ im 2D-Fall.

Zur Impulsgleichung:



- Auswertung bzgl. u an den Mittelpunkten der senkrechten Kanten.
- Auswertung bzgl. v an den Mittelpunkten der waagerechten Kanten.
- Diffusive Terme: $\frac{\partial^2}{\partial x^2}u$, $\frac{\partial^2}{\partial y^2}u$, $\frac{\partial^2}{\partial x^2}v$, $\frac{\partial^2}{\partial y^2}v$ werden durch zentrale Differenzen approximiert, z.B.

$$\frac{d^2}{dx^2}u(x_{i,j}) \approx \frac{1}{\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}).$$

- Ortsableitungen des Drucks durch zentrale Differenzenquotienten mit halber Schrittweite.
- die konvektiven Terme sind etwas schwieriger. Diese lauten in 2D

$$(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = \begin{pmatrix} u\frac{\partial}{\partial x}u + v\frac{\partial}{\partial y}u \\ u\frac{\partial}{\partial x}v + v\frac{\partial}{\partial y}v \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x}(u^2) + \frac{\partial}{\partial y}(uv) \\ \frac{\partial}{\partial x}(uv) + \frac{\partial}{\partial y}(v^2) \end{pmatrix},$$

  denn wegen der Ketten- und Produktregel und der Kontinuitätsgleichung gilt

$$\begin{aligned} \frac{\partial}{\partial x}(u^2) + \frac{\partial}{\partial y}(uv) &= 2u\frac{\partial}{\partial x}u + u\frac{\partial}{\partial y}v + v\frac{\partial}{\partial y}u \\ &= u\frac{\partial}{\partial x}u + v\frac{\partial}{\partial y}u + u(\text{div } \boldsymbol{u}) \\ &= u\frac{\partial}{\partial x}u + v\frac{\partial}{\partial y}u = ((\boldsymbol{u}\cdot\nabla)\boldsymbol{u})_1 \end{aligned}$$

  und analog für die zweite Komponente.

Wir haben also die konvektiven Terme

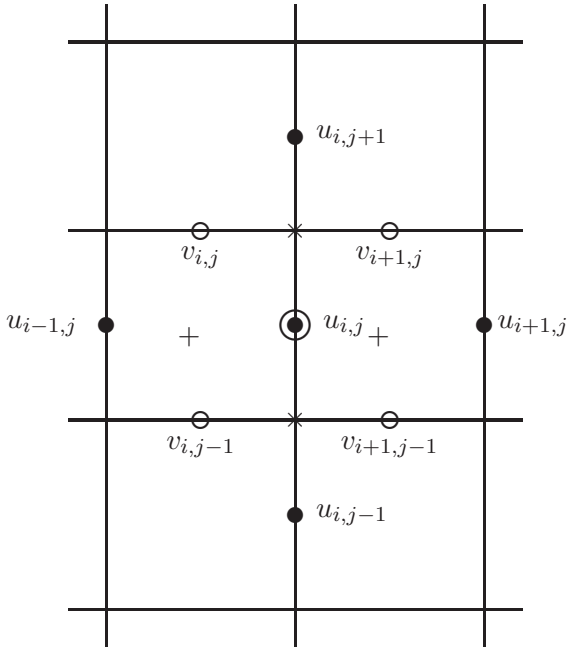
$$\frac{\partial}{\partial x}(u^2),\ \frac{\partial}{\partial y}(v^2),\ \frac{\partial}{\partial x}(uv),\ \frac{\partial}{\partial y}(uv)$$

zu diskretisieren.

Als Beispiel betrachte $\frac{\partial}{\partial x}(uv)$:
Im Punkt für $u_{i,j}$ (gekennzeichnet durch $\odot$) benötigen wir die Werte für das Produkt uv in vertikaler Richtung. Dort gibt es keine Punkte, an denen simultan u und v vorliegen.

Idee:

- verwende die mit $\times$ gekennzeichneten Punkte,
- mittle u aus den vertikal benachbarten Punken,



- mittle v aus den horizontal benachbarten Punkten.

$$\left(\frac{\partial}{\partial y}(uv)\right)_{i,j} \approx \frac{1}{\Delta y}\Big(\frac{1}{2}(v_{i,j}+v_{i+1,j})\cdot\frac{1}{2}(u_{i,j}+u_{i,j+1}) - \frac{1}{2}(v_{i,j-1}+v_{i+1,j-1})\cdot\frac{1}{2}(u_{i,j}+u_{i,j-1})\Big)$$

Der Term $\frac{\partial}{\partial x}(u^2)$ wird analog mit den mit $+$ markierten Punkten diskretisiert.

Nun sind all diese Terme konvektiv, daher müssen wir die Diskretisiertung wie in Abschnitt 4.3 anpassen. Dies tun wir, indem wir zu den Termen wie oben das γ-fache Vielfache ($\gamma \in [0,1]$) der Donor–Cell–Diskretisierung addieren (für $\gamma = 1$ ergibt sich die Donor–Cell–Form). Für obiges Beispiel $\frac{\partial}{\partial x}(uv)$ verwenden wir

$$k_r := \frac{1}{2}(v_{i,j}+v_{i+1,j}), \quad k_\ell := \frac{1}{2}(v_{i,j-1}+v_{i+1,j-1})$$

und erhalten

$$\begin{aligned}\left(\frac{\partial}{\partial y}(uv)\right)_{i,j} \approx\ & \frac{1}{\Delta y}\left\{\frac{1}{4}(v_{i,j}+v_{i+1,j})(u_{i,j}+u_{i,j+1}) - \frac{1}{4}(v_{i,j-1}+v_{i+1,j-1})(u_{i,j}+u_{i,j-1})\right\}\\ & +\gamma\frac{1}{\Delta y}\left\{\frac{1}{4}(v_{i,j}+v_{i+1,j})(u_{i,j}+u_{i,j+1}) - (v_{i,j-1}+v_{i+1,j-1})(u_{i,j-1}-u_{i,j})\right\}\end{aligned}$$

und analog für alle anderen konvektiven Terme.

Schließlich müssen die Randbedingungen adäquat diskretisiert werden, vgl. [2], 30-33.

4.6 DIE ZEIT–DISKRETISIERUNG

Bislang haben wir uns ausschließlich mit der Diskretisierung bezüglich des Ortes beschäftigt, (4.1.1) ist aber eine instationäre Gleichung mit der Zeit–Ableitung

$$\frac{\partial}{\partial t}\boldsymbol{u}.$$

Wie bei der Numerik gewöhnlicher Differentialgleichungen wird eine zeitartige (gewöhnliche) Differentialgleichung mittels einer Iteration approximiert. Dazu benötigt man zunächst Startwerte

$$u_{i,j}^{(0)},\ v_{i,j}^{(0)},$$

z.B.

$$(u_{i,j}^{(0)}, v_{i,j}^{(0)})^T = \boldsymbol{u}_0(x_{i,j})$$

mit der Funktion $\boldsymbol{u}_0$ aus der Anfangs–Bedingung (4.1.3). Das einfachste Verfahren ist das explizite Euler–Verfahren. Dies lautet für das System von Gleichungen

$$\frac{\partial}{\partial t}\boldsymbol{u}_{i,j} = f(\boldsymbol{U}, P),\ \boldsymbol{U} = (\boldsymbol{u}_{i,j})_{i,j},\ P = (p_{i,j})_{i,j}$$

mit dem Startwert $\boldsymbol{u}_{i,j}^{(0)}$ und der Zeit–Schrittweite Δt

$$\boldsymbol{u}_{i,j}^{(n+1)} = \boldsymbol{u}_{i,j}^{(n)} + \Delta t\ f(\boldsymbol{U}^{(n)}, P^{(n)}).$$

In unserem Fall ergibt sich f aus der Orts–Diskretisierung der Navier–Stokes–Gleichungen.

Zunächst ist dieses Vorgehen scheinbar einfach, weil man $\boldsymbol{u}^{(n+1)}$ aus $\boldsymbol{u}^{(n)}$ einfach durch Auswertung von f, also Auswertung der Finiten Differenzen, erhält. Leider ist die Situation nicht so einfach:

- es kann mit dieser Iteration leicht passieren, dass die Kontinuitätsgleichung div $\boldsymbol{u}^{(n)} = 0$ verletzt ist,
- der Druck wird nicht iteriert. Es gibt keine Vorschrift zur Berechnung von $P^{(n)}$ aus $P^{(n+1)}$.

Daher müssen wir anders vorgehen.

Wir diskretisieren zunächst nur die Zeitableitung in der Impulsgleichung

$$\begin{aligned} \boldsymbol{u}^{(n+1)} &= \boldsymbol{u}^{(n)} + \Delta t \left\{ \frac{1}{Re}\Delta\boldsymbol{u} - (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \boldsymbol{g} \right\} - \Delta t \nabla p \\ &=: F - \Delta t \nabla p. \end{aligned} \tag{4.6.1}$$

Um nun den Druck mit in die Iteration einzubeziehen, diskretisiert man *explizit* in der Geschwindigkeit und *implizit* im Druck, d.h.

$$\boldsymbol{u}^{(n+1)} = F^{(n)} - \Delta t \nabla p^{(n+1)}. \tag{4.6.2}$$

Dies hat zur Folge, dass der Druck bekannt sein muss, wozu man die Kontinuitätsgleichung heranzieht:

$$0 = \text{ div } \boldsymbol{u}^{(n+1)} = \text{ div } F^{(n)} - \Delta t \Delta p^{(n+1)},$$

also

$$\Delta p^{(n+1)} = \frac{1}{\Delta t} \text{ div } F^{(n)}. \tag{4.6.3}$$

Damit haben wir folgendes allgemeine Vorgehen:

1.) Berechne $F^{(n)} = \boldsymbol{u}^{(n)} + \Delta \left\{ \frac{1}{Re}\Delta\boldsymbol{u}^{(n)} - (\boldsymbol{u}^{(n)}\cdot\nabla)\boldsymbol{u}^{(n)} + \boldsymbol{g} \right\}$ aus der Diskretisierung der Geschwindigkeitsausdrücke.

2.) Löse die Poisson–Gleichung (4.6.3) für den Druck $p^{(n+1)}$.

3.) Berechne $\boldsymbol{u}^{(n+1)}$ aus (4.6.2).

Für die numerische Lösung von (4.6.3) brauchen wir noch Randbedingungen. Bei der *Projektionsmethode von Chorin* verwendet man die Impulsgleichung, um eine Neumann–Randbedingung herzuleiten. Mit der äußeren Normalen $\vec{n}$ auf $\Gamma = \partial\Omega$ gilt

$$\nabla p^{(n+1)} \cdot \vec{n} = -\frac{1}{\Delta t}(\boldsymbol{u}^{(n+1)} - F^{(n)}) \cdot \vec{n}$$

und $\boldsymbol{u}^{(n+1)}$ ist auf $\partial\Omega$ durch die Randbedingung der Navier–Stokes–Gleichungen geben. Schließlich wird die Divergenz in (4.6.3) mit rückwärtigen Differenzen diskretisiert:

$$\text{div } F^{(n)} \approx \frac{1}{\Delta x}(F^{(n)}_{i,j} - F^{(n)}_{i-1,j}) + \frac{1}{\Delta y}(F^{(n)}_{i,j} - F^{(n)}_{i,j-1}).$$

Zur Lösung der Druck–Gleichung (4.6.3) verwendet man z.B. das Mehrgitter–Verfahren.

4.7 EINIGE BEMERKUNGEN ZUR PARALLELISIERUNG

Wir betrachten wiederum Gebiets–Zerlegung zur Parallelisierung. Für die Druck–Gleichung muss dabei eine Schicht der Druck–Randwerte übertragen werden, für die Berechnung von $F^{(n)}$ eine Randschicht von Geschwindigkeitswerten. Ansonsten haben wir alle weiteren Komponenten bereits parallelisiert.

Literaturverzeichnis

[1] G. Alefeld, I. Leonhardt, H. Obermaier, Parallele Numerische Verfahren, Springer, 2002.

[2] M. Griebel, T. Dornseifer, T. Neunhoeffer, Numerische Simulation in der Strömungsmechanik, Vieweg 1995.

[3] M. Hanke–Bourgeois, Grundlagen der Numerischen Mathematik und des wissenschaftlichen Rechnens, Teubner 2006.