Iterative Verfahren zur Lösung linearer Gleichungssysteme und deren Parallelisierung

Michael Krätschmer

8. Januar 2007

1 Einführung

In praktischen Anwendungen, darunter zählen z.B. Finite-Element oder Finite-Volumen Codes, entstehen sehr oft außerordentlich große Gleichungssysteme, deren Koeffizientenmatrix K allerdings von sehr spezieller Struktur ist; jede Zeile besitzt nur relativ wenige von null verschiedene Einträge. Die Lösung eines solchen Systems mit direkten Methoden (z.B. Gaußsches Eleminationsverfahren oder Cholesky) ist aufgrund des entstehenden Aufwands nicht mehr sinnvoll. Auch würden solche Algorithmen die Struktur des System nicht ausnutzen; ja sogar zerstören (fill-in).
Auch ist man oft gar nicht an der exakten Lsung interessiert, sondern gibt sich mit einer guten Näherung zufrieden, deren Fehler die gleiche Größenordung aufweist wie die im Laufe der Problembehandlung genutzten Diskritisierungs- und Approximationsmethoden.
In der Praxis werden in solchen Fällen beinahe ausschließlich iterative Verfahren genutzt, die so lange angwendet werden, bis sich der Lösungsvektor "nicht mehr wesentlich" ändert.

Die Grundidee bei der Gewinnung solcher Verfahren besteht in der additiven Zerlegung der Systemmatrix in einen "einfachen" und einen "schweren" Anteil:

$$K = E + S \ .$$

Mit dem einfaches Anteil der Matrix E löst man nun das lineare System

$$Eu^k = f \ ,$$

was auf ein Residuum

$$r^k := f - Ku^k$$

führt. Im Sinn einer iterativen Nachverbesserung wird nun eine Korrektur Δu nach

$$E\Delta u = r^k$$

errechnet. Umformung des Korrekturschritts führt dann schließlich auf die Iterationsvorschrift:

$$\begin{aligned} u^{k+1} &= u^k + \Delta u \\ &= u^k + E^{-1} r^k \\ &= u^k + E^{-1}(f - Ku^k) \\ &= E^{-1} f - E^{-1} S u^k \ . \end{aligned}$$

Für die Norm der Jacobimatrix J der Iteration gilt:

$$||J|| = ||E^{-1}S|| \leq \rho(J) + \varepsilon \quad (\varepsilon > 0 \text{ bel. klein}) \quad .$$

Dadurch wird ersichtlich, dass für die Konvergenz eines solchen Verfahren der Sprektralradius der Jacobimatrix $\rho(J)$ die entscheidende Größe darstellt. Es gilt sogar

Satz: Ein Iterationsverfahren nach obiger Bauart ist genau dann für beliebige Startwerte konvergent gegen die Lösung, wenn

$$\rho(E^{-1}S) < 1 \ .$$

Der Beweis dieses Satzes beruht im Wesentlichen auf dem Fixpunktsatz nach Banach.

Aus diesen Betrachtungen lassen sich folgende Forderungen für die Zerlegung der Systemmatrix:

- Der Matrixteil E sollte eine möglichst gute Approximation an A sein.
- Lineare System mit Systemmatrix E sollten leicht lösbar sein.

Da beide Kriterien nicht gleichzeitig optimal erfüllt werden können, muß ein Kompromiß gefunden werden, welcher je nach Verfahren unterschiedlich ausfällt.

2 Jacobi Verfahren

2.1 Sequentielle Variante

Das Jacobi-Verfahren nutzt nun die wohl einfachste Art der Approximation der Systemmatrix durch deren Diagonale. Diese sei mit D bezeichnet.

Wähle u^0
$r := f - Ku^0$
$\Delta u := D^{-1}r$
$\sigma := \sigma_0 :=< \Delta u, r >$
$k := 0$
while $\sigma > TOL \cdot \sigma_0$
    $k := k + 1$
    $u^k := u^{k-1} + \omega \cdot \Delta u$
    $r := f - Ku^k$
    $\Delta u := D^{-1}r$
    $\sigma :=< \Delta u, r >$
end

Wobei ω einen Relaxationsparameter bezeichnet.

Schreibt man dieses Verfahren komponentenweise aus, so erhällt man

$$u_i^k = u_i^{k-1} + \frac{1}{a_{ii}}(f_i - \sum_{j=1}^{n} a_{ij} u_i^{k-1}) \ .$$

Hierbei fällt auf, dass eine Komponente der neuen Näherung zur Berechnung nur die Komponenten der alten benötigt. Zwischen diesen n Rechnungen besteht keine Kommunikation, weswegen sich die Parallelisierung recht einfach gestaltet.

2.2 Paralleler Algorithmus

Die nachfolgende Beschreibung bezieht sich auf die bereits bekannten Definitionen. Vor allem auf die Begriffe der Typ-I/II Vektoren und Matrizen wird hierbei zurückgegriffen. Sämtliche Bezeichnungen beziehen sich auf eine Finite-Element-Diskritisierung.

Es wir im wesentlichen erläutert, wie die einzelnen Vektoren und Matrizen im Laufe des Verfahren gespeichert werden müssen, damit ein Minimum an Datenaustausch (sprich: Kommunikation) entsteht.

Die Speicherung der Koeffizientenmatrix K erfolgt nach dem Typ-II Prinzip (verteilte Speicherung), d.h. diese wird mit Hilfe der lokalen Koinzidenzmatrizen A_i (deren Werte entweder 1 oder 0 sein können) gespeichert:

$$K = \sum_{i=1}^{p} A^T K_i A_i \ .$$

Die K_i stellen hierbei die lokalen Steifigkeitsmatrizen dar; p entspricht demnach der Anzahl der Elemente. Mit Hilfe dieser Matrix muß zunächt im Iterationssetup die Inverse der Diagonalen D^{-1} gebildet werden, was Kommunikation nötig macht:

$$D^{-1} = diag^{-1}\left(\sum_{i=1}^{p} A_i^T K_i A_i\right) = \left(\sum_{i=1}^{p} A^T diag(K_i) A_i\right)^{-1}$$

Als nächstes kümmern wir uns um die Berechnung des Residuums $r = f - Ku^k$. Dies wird dadurch kommunikationslos realisiert, indem die rechte Seite f im Typ-II Schema gespeichert wird (d.h. in Form von Elementlast-

vektoren). Dann ergibt sich die Berechnung des Residuums zu

$$\begin{aligned} r &= f - \left(\sum_{i=1}^{p} A^T K_i A_i\right) u^k \\ &= \sum_{i=1}^{p} A_i^T f_i - \sum_{i=1}^{p} A^T K_i \underbrace{(A_i u^k)}_{:=u_i^k} \\ &= \sum_{i=1}^{p} (A_i^T \underbrace{[f_i - K_i u_i]}_{=r_i}) \end{aligned}$$

wobei mit $u_i := A_i u$ der "zum i-ten Element gehörende Teil" des Lösuungsvektor bezeichnet wird. Es fällt auf, dass dies gerade einer Typ-I Speicherung entspricht, während das Residuum im Typ-II Schema abgelegt wird.

Die Korrektur $\Delta u = D^{-1} \sum_{i=1}^{p} A_i^T r_i$ läßt dich, da es sich bei D um eine Diagonalmatrix handelt, aus

$$\Delta u = \sum_{i=1}^{p} A_i^T \underbrace{D_i^{-1} r_i}_{=\Delta u_i}$$

berechnen; mit $D_i := A_i D A_i^T$. Wobei nur die "lokalen Korrekturen" Δu_i berechnet werden. Aus diesen läßt sich dann das Skalarprodukts zum Zweck der Fehlerabschätzung bestimmen

$$\sigma =< \Delta u, r >= \sum_{i=1}^{p} < \Delta u_i . r_i > \ .$$

Kommunikation tritt hierbei nur in der Summation über den globalen Index auf.
Der Updateschritt erfolgt zunächst "lokal" über $u_i^{k+1} = u_i^k + \omega \cdot \Delta u_i$, bevor die Zellen zum Zweck des Zusammenbaus des neuen Iterationswerts u_i^{k+1} wegen

$$u^{k+1} = \sum_{i=1}^{p} A_i^T u_i^{k+1}$$

miteinander kommunizieren müssen.

$D^{-1} = \left(\sum_{i=1}^{p} A^T diag(K_i) A_i \right)^{-1}$
Wähle u^0
$r := f - Ku^0$
$\Delta u = \sum_{i=1}^{p} A_i^T D_i^{-1} r_i$
$\sigma := \sigma_0 :=< \Delta u, r >$
$k := 0$
while $\sigma > TOL \cdot \sigma_0$
    $k := k + 1$
    $u^k := u^{k-1} + \omega \cdot \Delta u$
    $r := f - Ku^k$
    $\Delta u = \sum_{i=1}^{p} A_i^T D_i^{-1} r_i$
    $\sigma :=< \Delta u, r >$
end

3 Gauß-Seidel Verfahren

3.1 Sequentielle Variante

Im Folgenden bezeichnen L und U den strikten unten bzw. oberen Dreiecksteil der Systemmatrix K (es gilt also $K = D + L + U$). Der Gauß-Seidel Algorithmus nutzt nun den unteren Matrixteil $D+L$ (Gauß-Seidel vorwärts) bzw. den oberen Matrixteil $D + U$ (Gauß-Seidel rückwärts) zur Lösung der im Laufe der iterativen Nachverbesserung auftretenden linearen Systeme. Wir betrachten exemplarisch nur die "Vorwärts-Variante":

Wähle u^0
$r := f - Ku^0$
$\sigma := \sigma_0 :=< r, r >$
$k := 0$
while $\sigma > TOL \cdot \sigma_0$
    $k := k + 1$
    $u^k := u^{k-1} + D^{-1} \cdot \left(f - Lu^k - (D + U) \cdot u^{k-1} \right)$
    $r := f - Ku^k$
    $\sigma :=< r, r >$
end

Komponentenweise lautet der oben angegebene Algorithmus:

$$u_i^k = u_i^{k-1} + \frac{1}{a_{ii}} (f_i - \sum_{j=1}^{i-1} a_{ij} u_i^k - \sum_{j=i}^{n} a_{ij} u_i^{k-1}) \ .$$

Hieran sieht man deutlich, wo das Problem bei einer Parallelisierung dieses Verfahrens liegt:
Eine Komponente der neuen Näherung benötigt neben Informationen aus der alten zusätzlich noch Daten aus allen vorigen Komponenten des aktuellen Schritts.

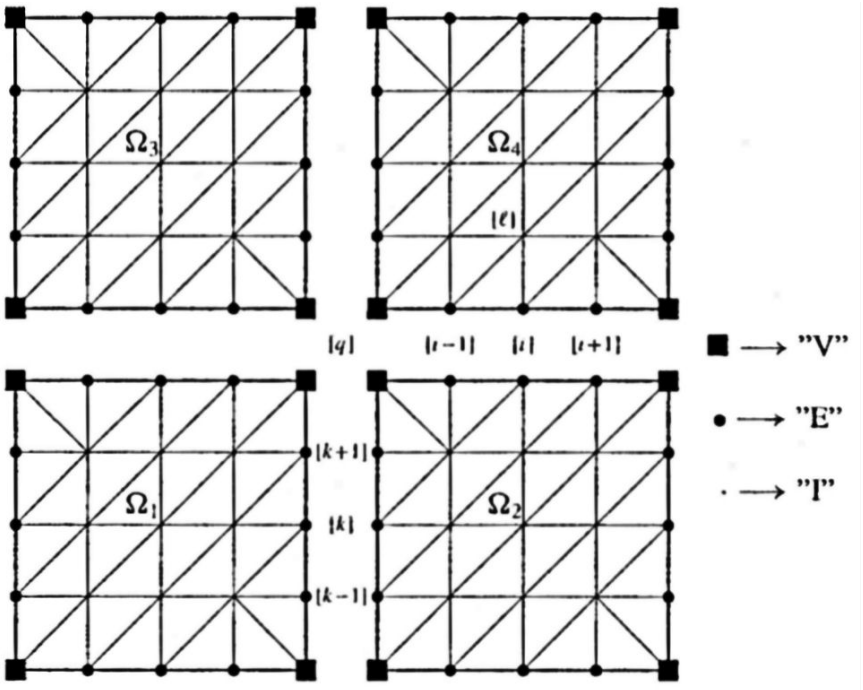
Dies führt dazu, dass eine komponentenweise Parallelisierung wie beim Jacobi-Verfahren nicht möglich ist.

3.2 Red-black Gauß-Seidel

Um diesem Problem Herr zu wird die Indexmenge in mindestens zwei disjunkte Teilmengen aufgeteilt: ω_{red} und ω_{black}. Diese Aufteilung erfolgt zu dem Zweck, dass die auf die Indexmengen eingeschränkte Systemmatrix K Diagonalgestalt besitzt und so die Komponenten des Lösungsvektors in diesem Block nicht gekoppelt sind, was die parallele Berechnung dieser ermöglicht.

3.3 Paralleler Algorithmus

Nun beschäftigen wir uns mit einer blockweisen Parallelisierung des Gauß-Seidel Verfahrens. Dieses Vorgehen sei anhand einer Blocktriangulierung eines quadratischen Problemgebiets veranschaulicht Der Lösungvektor soll



nun so numeriert werden, dass zunächst die Eintrage auftauchen, die zu den Ecken V gehören, dann die Einträge, die zu den Interfaces E und schließlich die inneren Knoten I. Dies resultiert in einem Gleichungssystem der folgenden Bauart:

$$\begin{pmatrix} K_V & K_{VE} & K_{VI} \\ K_{EV} & K_E & K_{EI} \\ K_{IV} & K_{IE} & K_I \end{pmatrix} \begin{pmatrix} u_V \\ u_E \\ u_I \end{pmatrix} = \begin{pmatrix} f_V \\ f_E \\ f_I \end{pmatrix} \quad .$$

Während der Matrixblock K_V Diagonalgestalt besitzt, ist der Block K_I i.a. voll besetzt, was nicht verwunderlich ist, da dieser Teil die Verbindung zwischen den einzelnen Blöcken herstellt und daher Kommuniktion auftreten muß. Der Knackpunkt ist allerdings der Matrixteil K_I. Dieser besitzt folgende Blockstruktur (für das Beispiel von vier inneren Zones $\Omega_1, \ldots, \Omega_4$):

$$K_I = \begin{pmatrix} K_{\Omega_1} & 0 & 0 & 0 \\ 0 & K_{\Omega_2} & 0 & 0 \\ 0 & 0 & K_{\Omega_3} & 0 \\ 0 & 0 & 0 & K_{\Omega_4} \end{pmatrix} \quad .$$

Dies führt auf einen *blockweise-parallelen* Gauß-Seidel Algorithmus. Dies stellt durchaus einen praktikablen Ansatz dar, da i.d.R. die Anzahl der inneren Knoten überwiegen (z.B. bei sog. blockstrukturierten Gittern).
Trotzdem hat man das Problem den Datanabhängigkeit des Verfahrens noch nicht ganz aus der Welt geräumt. Je mehr Prozessoren man einsetzen möchte, desto mehr innere Blöcke müssen kontruiert werden, was wiederum die Anzahl der Interfaceknoten und damit die Kommunikation erhöht.

3.4 Gauß-Seidel / Jacobi

Ein Ansatz dies zu vermeiden ist es die beiden besprochenen Verfahren so zu kombinieren, dass in den Teilen, in denen Kommunikation auftritt, das Jacobiverfahren zum Einsatz kommt, während im Inneren der Blöcke das Gauß-Seidel Verfahren verwendet wird. Durch die Kombination wird zwar die Konvergenz verglichen mit dem "reinen" Gauß-Seidel Verfahen schlechter, jedoch entsteht dabei deutlich weniger Kommunikation.