

Vorkonditionierung durch unvollständige
LU-Zerlegung beim
konjugierte-Gradienten-Verfahren

Diplomarbeit

Stephan Weber
Betreuer: Prof. Dr. U. Schendel
Fachbereich Mathematik
Freie Universität Berlin
1988

Inhalt

1	Einleitung	2
2	Kurzer Abriß des Relaxationsprinzips	5
2.1	Vorbemerkungen	5
2.2	Relaxation	5
2.3	Zur Konvergenz des Verfahrens	10
3	Das konjugierte-Gradienten-Verfahren	12
3.1	Das Verfahren	12
3.2	Konvergenzbetrachtungen	15
4	Vorkonditioniertes CG-Verfahren	24
4.1	Vorbemerkungen	24
4.2	Der Algorithmus	25
5	Eine Methode der Vorkonditionierung	29
6	Vorkonditionierung durch unvollständige Zerlegung	32
6.1	Vorbemerkungen	32
6.2	PCG mit unvollständiger <i>LU</i> -Zerlegung für symmetrische <i>M</i> -Matrizen	33
6.3	Unvollständige <i>LU</i> -Zerlegung nach der Methode von KERSHAW . . .	38
6.4	PCG mit modifizierter unvollständiger Zerlegung	39
7	Numerische Ergebnisse	48
7.1	Beschreibung des Programms und der Beispiele	48
7.2	Ergebnisse	53
7.3	Ausblicke	63
8	Anhang	64
9	Literaturverzeichnis	84

1 Einleitung

Bei der numerischen Lösung von Differentialgleichungen durch Umwandlung in Differenzgleichungen über einem diskretisierten Gebiet oder durch Finite-Elemente-Verfahren entstehen mehr oder weniger große lineare Gleichungssysteme. Eine wesentliche Eigenschaft dieser Systeme ist die Tatsache, daß die zugehörigen Koeffizientenmatrizen nur wenige von Null verschiedene Einträge haben ('sparse' sind).

Darüber hinaus weisen diese Matrizen oft noch spezielle Eigenschaften auf, wie bandedartige Struktur, Symmetrie, Definitheit, so daß direkte Lösungsverfahren wie z.B. der Gauß-Algorithmus weniger geeignet sind, da sie auf die spezielle Struktur wenig Rücksicht nehmen.

Speichert man z.B. aus Platzgründen bei einer großen Tridiagonalmatrix nur die Diagonalen ab, so benötigt man für die Dreiecksmatrizen, die beim Gauß-Algorithmus entstehen, doch wieder mehr Platz, weil zwischen den Diagonalen fill-in entsteht.

Ein weiterer wesentlicher Grund, statt direkter Verfahren iterative zu verwenden, besteht in folgendem: Bei den direkten Verfahren hat man, Lösbarkeit vorausgesetzt, die, relativ zur Kondition des Problems, exakte Lösung nach genau n Schritten, wobei n die Dimension des Vektorraumes des linearen Gleichungssystems ist. Bei den iterativen Verfahren hat man nicht unbedingt nach höchstens n Schritten die in obigem Sinne exakte Lösung, jedoch kann man oft erwarten, mit jedem Iterationsschritt eine bessere Näherung an die exakte Lösung zu erreichen. Oder anders formuliert: Man sucht nach iterativen Verfahren, bei denen die Näherungslösung nach relativ wenigen Iterationsschritten, d.h. mit relativ wenig Rechenaufwand, einen vorgegebenen relativen Fehler unterschreitet. Dabei spielt die numerische Stabilität des Algorithmus auch eine wichtige Rolle.

Ein Nachteil der iterativen Methoden besteht darin, daß bei der Lösung mehrerer Gleichungssysteme mit gleicher Koeffizientenmatrix, aber verschiedenen Konstantenvektoren ('rechte Seiten'), das Verfahren für jedes Gleichungssystem vollständig durchgeführt werden muß, während bei den direkten Methoden nur der jeweils andere Konstantenvektor eingesetzt werden muß.

Wie groß der Rechenaufwand nun jeweils ist, um eine bestimmte Verbesserung des Ergebnisses zu erhalten, ist eine Frage der Konvergenzeigenschaften des Verfahrens und damit auch der Eigenschaften der Koeffizientenmatrix. Die Konvergenzgeschwindigkeit hängt wesentlich von der Konditionszahl dieser Matrix sowie von der Verteilung ihrer Eigenwerte ab. Eine Idee der Konvergenzverbesserung ist nun die, die Koeffizientenmatrix A so zu transformieren, daß die Konditionszahl der transformierten Matrix \tilde{A} kleiner ist als die von A . Dann wird das transformierte System (schneller) gelöst und die Lösung zurücktransformiert. Sinn hat dies jedoch nur, wenn der zusätzliche Rechenaufwand zur Transformation nicht den Effekt der Konvergenzverbesserung wieder aufhebt. Diese Transformation bezeichnet man als

Vorkonditionierung.

In der vorliegenden Arbeit werden verschiedene Ansätze der Vorkonditionierung des Verfahrens der konjugierten Gradienten (CG-Verfahren) dargestellt. Das CG-Verfahren hat neben der Tatsache, daß man es als Iterationsverfahren verwenden kann noch den Vorteil, daß es parameterfrei ist, d.h. es ist nicht wie beim SOR-Verfahren und ähnlichen nötig, vorab Iterationsparameter zu berechnen oder zu schätzen. Dieses Verfahren selbst ist 1952 von M. Hestenes und E. Stiefel vorgestellt worden, verschiedene Ansätze zur Vorkonditionierung sind in den letzten zehn bis fünfzehn Jahren veröffentlicht worden.

Kapitel 2 enthält einen kurzen Abriß der Relaxationsverfahren, zu denen auch das CG-Verfahren gehört. Es wird eine hinreichende Bedingung für die Konvergenz von Relaxationsverfahren angegeben.

In Kapitel 3 wird das CG-Verfahren aus dem in Kapitel 2 Dargestellten hergeleitet. Weiter wird eine Abschätzung der Anzahl der Iterationsschritte, die nötig ist, um eine vorgegebene Fehlergrenze zu unterschreiten, angegeben. In dieser Abschätzung spielt die Kondition der Matrix eine Rolle.

In Kapitel 4 wird aufgrund der Abschätzung aus Kapitel 3 die Vorkonditionierung motiviert und der Algorithmus erläutert.

In Kapitel 5 werden kurz die stationären iterativen Verfahren dargestellt, aus welchen man bestimmte Vorkonditionierungsmatrizen erhält.

In Kapitel 6 werden Vorkonditionierungsmatrizen vorgestellt, welche aus der Koeffizientenmatrix durch unvollständige Zerlegung nach dem Gauß- oder Cholesky-Algorithmus gewonnen werden.

Kapitel 7 enthält Ergebnisse und Interpretationen eigener Rechnungen mit verschiedenen unvollständigen Zerlegungen.

Die vorliegende Diplomarbeit wurde im Zusammenhang mit dem Forschungsprojekt "Entwicklung und Strukturuntersuchung numerischer Algorithmen für Parallelrechner" (Projektleiter Prof. Dr. U. Schendel) erstellt.

Bezeichnungen und Definitionen

In der vorliegenden Arbeit werden ausschließlich reelle Matrizen behandelt.

Die Bezeichnungen A für die Koeffizientenmatrix, \tilde{A} für die transformierte Matrix, \mathbf{b} für die rechte Seite, $\hat{\mathbf{x}}$ für die Lösung von $A\mathbf{x} = \mathbf{b}$ und n für die Dimension des Raumes werden mit gleicher Bedeutung für die ganze Arbeit beibehalten. Die Menge $J \subset \{1, \dots, n\}^2$ enthält die Zeilen- und Spaltenindices, für die bei der (unvollständigen) Zerlegung von A Nichtnulleinträge zugelassen werden.

Vektoren werden mit fettgedruckten lateinischen Kleinbuchstaben bezeichnet.

(\cdot, \cdot) bedeutet das übliche Skalarprodukt, d.h. $(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n x_i y_i$,

$\|\cdot\|$ die euklidische Norm: $\|\mathbf{x}\| = \sqrt{(\mathbf{x}, \mathbf{x})}$.

Definition: Eine symmetrische Matrix $A \in \mathbf{R}^{n,n}$ heißt *positiv definit*, wenn für alle $\mathbf{x} \in \mathbf{R}^n \setminus \{\mathbf{0}\}$ gilt $(A\mathbf{x}, \mathbf{x}) > 0$.

Die Eigenwerte einer symmetrischen Matrix sind reell, diejenigen einer positiv definiten Matrix sind positiv.

$(\mathbf{x}, \mathbf{y})_A := (A\mathbf{x}, \mathbf{y}) = (\mathbf{x}, A\mathbf{y})$ ist ein Skalarprodukt, falls $A \in \mathbf{R}^{n,n}$ positiv definit ist, und $\|\mathbf{x}\|_A := \sqrt{(\mathbf{x}, \mathbf{x})_A}$ ist dann eine Norm.

Die Vektoren \mathbf{e}_i , $i = 1, \dots, n$ bezeichnen die kanonischen Basiseinheitsvektoren des \mathbf{R}^n . Begriffe wie Orthonormalbasis u.ä. beziehen sich auf die euklidische Norm.

Betragsstriche $|\cdot|$ und \leq bei Vektoren und Matrizen sind elementweise zu verstehen.

2 Kurzer Abriß des Relaxationsprinzips

2.1 Vorbemerkungen

Die meisten Probleme, die zu den hier behandelten linearen Gleichungssystemen führen, stammen aus physikalisch-technischen Anwendungen, wo Prozesse wie Strömungen, Balken- und Membranbiegung und -schwingung durch (partielle) Differentialgleichungen beschrieben werden. Bei der in der technischen Praxis bis vor einigen Jahren noch üblichen numerischen Lösung von Hand konnte man die Relaxationsverfahren so interpretieren, daß z.B. bei einem Balkentragwerk die Anfangsnäherungen der zu berechnenden Größen und während der Rechnung auftretende Residuen (s.u.) als der äußeren Belastung zusätzliche Zwangsgrößen aufgefaßt wurden. Der Rechengang bestand nun anschaulich darin, das Tragwerk sukzessive zu entspannen (to relax), d.h. am Ende der Rechnung waren die Zwangsgrößen ausgeglichen, so daß das System nicht mehr im Widerspruch zu seinen Elastizitäts- und Verformungsbedingungen stand. Dies bedeutet, daß die innere Energie des Systems minimiert wird (vgl. auch die Interpretation der Splines). Die Bezeichnung 'Relaxationsverfahren' geht auf SOUTHWELL zurück (vgl. SCHWARZ 1972, ZURMÜHL).

Aus den erwähnten Anwendungen ergibt sich oft, daß die Koeffizientenmatrix A des linearen Gleichungssystems symmetrisch und positiv definit ist, was bis auf weiteres vorausgesetzt sei.

2.2 Relaxation

Es sei also $A \in \mathbf{R}^{n,n}$ eine positiv definite Matrix und zu lösen sei das lineare Gleichungssystem

$$(1) \quad A\mathbf{x} = \mathbf{b},$$

mit dem konstanten Vektor $\mathbf{b} \in \mathbf{R}^n$.

Definiert man das Residuum

$$\mathbf{r} := \mathbf{r}(\mathbf{x}) := A\mathbf{x} - \mathbf{b},$$

dann besteht die Aufgabe also darin, den Vektor $\hat{\mathbf{x}}$ zu finden, für den $\mathbf{r}(\hat{\mathbf{x}}) = 0$ gilt. Dazu definiert man das Funktional

$$(2) \quad \begin{aligned} F(\mathbf{x}) &:= \frac{1}{2}(A\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x}) \\ &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n a_{ij} x_i x_j + \sum_{i=1}^n b_i x_i \end{aligned}$$

dessen partielle Ableitungen lauten

$$\begin{aligned}
 \frac{\partial}{\partial x_i} F(\mathbf{x}) &= \frac{1}{2} \left(\left(A \frac{\partial \mathbf{x}}{\partial x_i}, \mathbf{x} \right) + \left(A \mathbf{x}, \frac{\partial \mathbf{x}}{\partial x_i} \right) \right) - \left(\mathbf{b}, \frac{\partial \mathbf{x}}{\partial x_i} \right) \\
 &= \frac{1}{2} \left(\left(A \mathbf{e}_i, \mathbf{x} \right) + \left(A \mathbf{x}, \mathbf{e}_i \right) \right) - \left(\mathbf{b}, \mathbf{e}_i \right) \\
 &= \sum_{j=1}^n a_{ij} x_j - b_i \\
 &= r_i.
 \end{aligned}$$

Also gilt

$$(3) \quad \text{grad } F(\mathbf{x}) = (r_i)_{i=1 \dots n} = \mathbf{r} = A\mathbf{x} - \mathbf{b} =: F'(\mathbf{x}).$$

Weiterhin ist

$$\begin{aligned}
 \frac{\partial}{\partial x_j} (F'(\mathbf{x})) &= \frac{\partial}{\partial x_j} \left(\sum_{k=1}^n a_{ik} x_k - b_i \right)_{i=1 \dots n} \\
 &= (a_{ij})_{i=1 \dots n}
 \end{aligned}$$

Dies ist die j-te Spalte von A ; somit ist

$$(4) \quad (F_{x_j x_i})_{\substack{i=1, \dots, n \\ j=1, \dots, n}} = \left(\frac{\partial}{\partial x_j} \left(\frac{\partial}{\partial x_i} F(\mathbf{x}) \right) \right)_{\substack{i=1, \dots, n \\ j=1, \dots, n}} = A =: F''(\mathbf{x}).$$

Die hier interessierende Eigenschaft dieses Funktionals beschreibt der folgende

(5) **Satz:** F hat in $\hat{\mathbf{x}}$ ein eindeutiges Minimum genau dann, wenn $\hat{\mathbf{x}}$ Lösung von (1) ist.

Zum *Beweis* kann man sich mit (3) und (4) auf die Sätze der Analysis berufen (s. z.B. ERWE).

Ein anderer Weg ist folgender:

Schreibt man (2) mit Hilfe von $\mathbf{x} = \hat{\mathbf{x}} + \mathbf{x} - \hat{\mathbf{x}}$ und unter Berücksichtigung der Symmetrie von A aus, so erhält man:

$$\begin{aligned}
 F(\mathbf{x}) &= \frac{1}{2} (A(\hat{\mathbf{x}} + \mathbf{x} - \hat{\mathbf{x}}), \hat{\mathbf{x}} + \mathbf{x} - \hat{\mathbf{x}}) - (\mathbf{b}, \hat{\mathbf{x}} + \mathbf{x} - \hat{\mathbf{x}}) \\
 (6) \quad &= \frac{1}{2} (A\hat{\mathbf{x}}, \hat{\mathbf{x}}) + \frac{1}{2} (A(\mathbf{x} - \hat{\mathbf{x}}), \mathbf{x} - \hat{\mathbf{x}}) + (A\hat{\mathbf{x}}, \mathbf{x} - \hat{\mathbf{x}}) - (\mathbf{b}, \hat{\mathbf{x}}) - (\mathbf{b}, \mathbf{x} - \hat{\mathbf{x}}) \\
 &= F(\hat{\mathbf{x}}) + \frac{1}{2} (A(\mathbf{x} - \hat{\mathbf{x}}), \mathbf{x} - \hat{\mathbf{x}})
 \end{aligned}$$

wenn $\hat{\mathbf{x}}$ die Lösung von (1) ist.

Da A positiv definit ist, gilt $F(\mathbf{x}) > F(\hat{\mathbf{x}})$ f.a. $\mathbf{x} \neq \hat{\mathbf{x}}$. (Ist A nur positiv semidefinit, dann ist das Minimum nicht mehr eindeutig).

Hat nun umgekehrt F ein Minimum in $\hat{\mathbf{x}}$, dann ist notwendig

$$F'(\hat{\mathbf{x}}) = 0 = \mathbf{r}(\hat{\mathbf{x}}) = A\hat{\mathbf{x}} - \mathbf{b}$$

d.h. $\hat{\mathbf{x}}$ ist Lösung von (1). \square

Das Lösen des linearen Gleichungssystems (1) ist also gleichbedeutend damit, den Punkt $\hat{\mathbf{x}}$ zu finden, in dem das Funktional (2) sein Minimum annimmt.

Bemerkung: Der Wert des Funktional dort ist

$$F(\hat{\mathbf{x}}) = \frac{1}{2}(A\hat{\mathbf{x}}, \hat{\mathbf{x}}) - (\mathbf{b}, \hat{\mathbf{x}}) = \frac{1}{2}(\mathbf{b}, \hat{\mathbf{x}}) - (\mathbf{b}, \hat{\mathbf{x}}) = -\frac{1}{2}(\mathbf{b}, \hat{\mathbf{x}}).$$

Einige Bemerkungen zur Veranschaulichung des Verfahrens

Für jedes $c \in \mathbf{R}$ mit $c \geq F(\hat{\mathbf{x}})$ ist die Fläche 2. Ordnung $E_c := \{\mathbf{x} \in \mathbf{R}^n | F(\mathbf{x}) = c\}$ ein (verallgemeinertes) Ellipsoid, da A positiv definit ist. $\{\hat{\mathbf{x}}\}$ wird als entartetes Ellipsoid betrachtet (vgl. z.B. BRONSTEIN).

Für $n = 2$ ist dies eine Ellipse, für $n = 1$ zwei Punkte auf der reellen Zahlengeraden.

Das Zentrum dieser Ellipsen ist $\hat{\mathbf{x}}$, die Hauptachsenrichtungen sind durch die Eigenvektoren von A festgelegt, die Hauptachsenabschnitte sind proportional zu den Quadratwurzeln des Kehrwertes der zugehörigen Eigenwerte. Letzteres bedeutet also, daß die Form des Ellipsoiden immer mehr von der der Kugeloberfläche abweicht, je weiter die Eigenwerte von A gestreut sind.

Damit läßt sich das Problem, den Minimalpunkt von F zu finden, auch so ausdrücken, daß das gemeinsame Zentrum einer Ellipsoidenschar zu finden ist.

Definition: Der von $\hat{\mathbf{x}}$ ausgehende Strahl s schneide die Flächen E_α und E_β in \mathbf{x}_α bzw. \mathbf{x}_β ; $\alpha, \beta > F(\hat{\mathbf{x}})$. Ist dann $\varrho := |\mathbf{x}_\alpha - \hat{\mathbf{x}}|/|\mathbf{x}_\beta - \hat{\mathbf{x}}|$ nicht von der speziellen Wahl des Strahles s abhängig, dann heißen die beiden Flächen *ähnlich*.

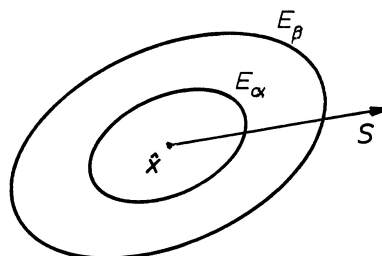


Bild 1

Lemma: Die Schar der E_c ist eine Schar ähnlicher Ellipsoide.

Beweis: Es seien E_α und E_β zwei solche Ellipsoide, $\alpha, \beta > F(\hat{\mathbf{x}})$, s ein Strahl, der von $\hat{\mathbf{x}}$ ausgeht und E_α und E_β in \mathbf{x}_α bzw. \mathbf{x}_β schneidet. Dann gibt es ein $\varrho \in \mathbf{R}^+$ so daß $\mathbf{x}_\beta - \hat{\mathbf{x}} = \sqrt{\varrho}(\mathbf{x}_\alpha - \hat{\mathbf{x}})$, weil $\mathbf{x}_\beta - \hat{\mathbf{x}}$ und $\mathbf{x}_\alpha - \hat{\mathbf{x}}$ die gleiche Richtung haben.

Nach (6) ist

$$\begin{aligned} F(\mathbf{x}_\beta) - F(\hat{\mathbf{x}}) &= \frac{1}{2}(A(\mathbf{x}_\beta - \hat{\mathbf{x}}), \mathbf{x}_\beta - \hat{\mathbf{x}}) \\ &= \frac{1}{2}\varrho(A(\mathbf{x}_\alpha - \hat{\mathbf{x}}), \mathbf{x}_\alpha - \hat{\mathbf{x}}) \\ &= \varrho(F(\mathbf{x}_\alpha) - F(\hat{\mathbf{x}})), \end{aligned}$$

also ist

$$\varrho = \frac{F(\mathbf{x}_\beta) - F(\hat{\mathbf{x}})}{F(\mathbf{x}_\alpha) - F(\hat{\mathbf{x}})} = \frac{\beta - F(\hat{\mathbf{x}})}{\alpha - F(\hat{\mathbf{x}})} > 0,$$

womit ϱ nicht von s , sondern nur von α und β abhängt. \square

Bemerkung: Dies bedeutet insbesondere, daß für $F(\hat{\mathbf{x}}) < \alpha < \beta$ das Ellipsoid E_α ganz 'innerhalb' des Ellipsoiden E_β liegt und der in Bild 2 dargestellte Fall nicht auftreten kann.

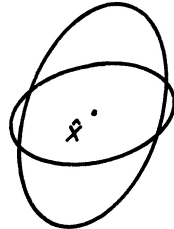


Bild 2

Zur iterativen Bestimmung von $\hat{\mathbf{x}}$ geht man nun wie folgt vor:

Man wählt einen Startvektor \mathbf{x} und eine Suchrichtung (Relaxationsrichtung) $\mathbf{p} \neq \mathbf{0}$. Im allgemeinen wird \mathbf{x} ungleich $\hat{\mathbf{x}}$ sein, so daß $F(\mathbf{x}) > F(\hat{\mathbf{x}})$ ist. Dann bestimmt man $t_0 \in \mathbf{R}$, so daß

$$F(\mathbf{x} + t_0\mathbf{p}) \leq F(\mathbf{x} + t\mathbf{p}) \text{ f.a. } t \in \mathbf{R},$$

d.h. man bestimmt auf der Geraden

$$G_{\mathbf{x}\mathbf{p}} := \{\mathbf{x} + t\mathbf{p} | t \in \mathbf{R}\}$$

den Punkt, für den F längs dieser Geraden minimal ist.

(7) **Lemma:** Ist $A \in \mathbf{R}^{n,n}$ positiv definit, dann existiert genau ein $t_0 \in \mathbf{R}$ mit dieser Minimumeigenschaft.

Beweis:

$$\begin{aligned} f(t) &:= F(\mathbf{x} + t\mathbf{p}) \\ &= \frac{1}{2}(A(\mathbf{x} + t\mathbf{p}), \mathbf{x} + t\mathbf{p}) - (\mathbf{b}, \mathbf{x} + t\mathbf{p}) \\ &= F(\mathbf{x}) + (A\mathbf{x} - \mathbf{b}, t\mathbf{p}) + \frac{1}{2}(At\mathbf{p}, t\mathbf{p}) \\ &= F(\mathbf{x}) + t(\mathbf{r}, \mathbf{p}) + \frac{1}{2}t^2(A\mathbf{p}, \mathbf{p}), \end{aligned}$$

$$\frac{d}{dt}f(t) = (\mathbf{r}, \mathbf{p}) + t(A\mathbf{p}, \mathbf{p}) = 0 \Rightarrow t_0 = -\frac{(\mathbf{r}, \mathbf{p})}{(A\mathbf{p}, \mathbf{p})};$$

da $\mathbf{p} \neq \mathbf{0}$ n.V., ist $(A\mathbf{p}, \mathbf{p}) > 0$, also

$$\frac{d^2}{dt^2}f(t_0) = (A\mathbf{p}, \mathbf{p}) > 0. \quad \square$$

Wird \mathbf{p} orthogonal zu \mathbf{r} gewählt, dann ist $t_0 = 0$, d.h. \mathbf{x} ist schon Minimalpunkt längs $G_{\mathbf{x}\mathbf{p}}$; ist $\mathbf{r} = \mathbf{0}$, dann ist $\mathbf{x} = \hat{\mathbf{x}}$ Lösung.

Aus formalen Gründen, damit das Verfahren nicht stehen bleibt, sei bis auf weiteres vorausgesetzt, daß \mathbf{p} nicht orthogonal zu \mathbf{r} gewählt wird.

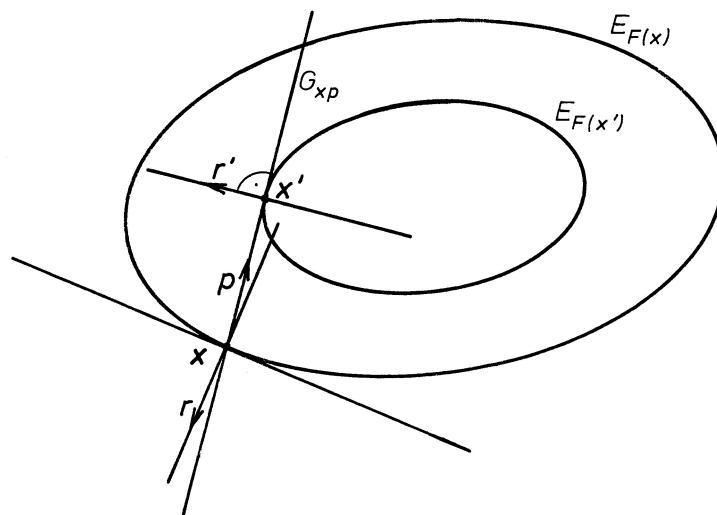


Bild 3

Setze nun $\mathbf{x}' := \mathbf{x} + t_0\mathbf{p}$;

da \mathbf{x}' eindeutiger Minimalpunkt längs $G_{\mathbf{x}\mathbf{p}}$ ist, ist $G_{\mathbf{x}\mathbf{p}}$ Tangente an $E_{F(\mathbf{x}')}$ in \mathbf{x}' .

Dies wird durch das folgende Lemma verdeutlicht.

(8) **Lemma:** $\mathbf{r}' = A\mathbf{x}' - \mathbf{b}$ ist orthogonal zu \mathbf{p} .

Beweis:

$$(\mathbf{r}', \mathbf{p}) = (A(\mathbf{x} + t_0\mathbf{p}) - \mathbf{b}, \mathbf{p}) = \left(\mathbf{r} - \frac{(\mathbf{r}, \mathbf{p})}{(A\mathbf{p}, \mathbf{p})} A\mathbf{p}, \mathbf{p} \right) = 0. \quad \square$$

(9) *Bemerkung:*

$$\begin{aligned} \Delta F := F(\mathbf{x}') - F(\mathbf{x}) &= \frac{1}{2} \left(A \left(\mathbf{x} - \frac{(\mathbf{r}, \mathbf{p})}{(A\mathbf{p}, \mathbf{p})} \mathbf{p} \right), \mathbf{x} - \frac{(\mathbf{r}, \mathbf{p})}{(A\mathbf{p}, \mathbf{p})} \mathbf{p} \right) \\ &\quad - \left(\mathbf{b}, \mathbf{x} - \frac{(\mathbf{r}, \mathbf{p})}{(A\mathbf{p}, \mathbf{p})} \mathbf{p} \right) - \left(\frac{1}{2}(A\mathbf{x}, \mathbf{x}) - (\mathbf{b}, \mathbf{x}) \right) \\ &= \dots = -\frac{1}{2} \frac{(\mathbf{r}, \mathbf{p})^2}{(A\mathbf{p}, \mathbf{p})} \\ &= -\frac{1}{2} \frac{(\mathbf{r}, \mathbf{p})^2 / \|\mathbf{p}\|^2}{(A\mathbf{p}, \mathbf{p}) / \|\mathbf{p}\|^2} = -\frac{1}{2} \frac{(\mathbf{r}, \mathbf{p}_e)^2}{(A\mathbf{p}_e, \mathbf{p}_e)} \leq 0, \end{aligned}$$

wobei $\|\mathbf{p}_e\| = 1$.

O.B.d.A. kann also hier $\|\mathbf{p}\| = 1$ vorausgesetzt werden, d.h. die Abnahme des Funktionalwertes ist nur von der Richtung von \mathbf{p} abhängig, was anschaulich klar ist.

Falls $\mathbf{r}' \neq \mathbf{0}$, dann wähle nun zu \mathbf{x}' eine Suchrichtung \mathbf{p}' mit $(\mathbf{r}', \mathbf{p}') \neq 0$.

Der Algorithmus lautet dann kurz zusammengefaßt:

- (10.1) $i := 0$, wähle Startvektor \mathbf{x}_i ,
 $\mathbf{r}_i = A\mathbf{x}_i - \mathbf{b}$,
- (10.2) falls $\mathbf{r}_i = \mathbf{0}$, dann ist $\mathbf{x}_i = \hat{\mathbf{x}}$ die Lösung,
- (10.3) sonst wähle eine Suchrichtung \mathbf{p}_i mit $(\mathbf{r}_i, \mathbf{p}_i) \neq 0$,
 $\mathbf{x}_{i+1} = \mathbf{x}_i - (\mathbf{r}_i, \mathbf{p}_i) / (A\mathbf{p}_i, \mathbf{p}_i) \mathbf{p}_i$,
 $i := i + 1$, weiter mit (10.2).

2.3 Zur Konvergenz des Verfahrens

Nach Satz (5) ist die Folge $(F_i)_{i \in \mathbf{N}}$ mit $F_i := F(\mathbf{x}_i)$, \mathbf{x}_i nach Algorithmus (10), durch $F(\hat{\mathbf{x}})$ nach unten beschränkt. Nach (9) ist $(F_i)_{i \in \mathbf{N}}$ monoton fallend, nach

(10.3) sogar streng monoton fallend, solange $\mathbf{x}_i \neq \hat{\mathbf{x}}$. Daher konvergiert die Folge gegen ein $F^* = F(\mathbf{x}^*)$, was wiederum bedeutet, daß nach (9) mit $\|\mathbf{p}_i\| = 1$ gilt:

$$\Delta F_i := F(\mathbf{x}_{i+1}) - F(\mathbf{x}_i) = -\frac{1}{2} \frac{(\mathbf{r}_i, \mathbf{p}_i)^2}{(A\mathbf{p}_i, \mathbf{p}_i)} \xrightarrow{i \rightarrow \infty} 0$$

also

$$(11) \quad (\mathbf{r}_i, \mathbf{p}_i) \xrightarrow{i \rightarrow \infty} 0.$$

(12) **Satz:** Existiert ein $\alpha > 0$, so daß für unendlich viele $i \in \mathbf{N}$ gilt

$$\left| \left(\frac{\mathbf{r}_i}{\|\mathbf{r}_i\|}, \mathbf{p}_i \right) \right| \geq \alpha, \quad \text{wobei} \quad \|\mathbf{p}_i\| = 1,$$

dann ist $\mathbf{x}^* = \hat{\mathbf{x}}$.

Beweis: Für die Teilfolge $(\mathbf{x}_{i_k})_{k \in \mathbf{N}}$, für die $|(\mathbf{r}_{i_k}/\|\mathbf{r}_{i_k}\|, \mathbf{p}_{i_k})| \geq \alpha$ gilt, ist nach (11) notwendig $\mathbf{r}_{i_k} \xrightarrow{k \rightarrow \infty} 0$, d.h. $F_{i_k} \xrightarrow{k \rightarrow \infty} F(\hat{\mathbf{x}})$. Da aber $(F_i)_{i \in \mathbf{N}}$ monoton ist, konvergiert auch $F_i \xrightarrow{i \rightarrow \infty} F(\hat{\mathbf{x}})$ und wegen der Eindeutigkeit des Minimalpunktes $\mathbf{x}_i \xrightarrow{i \rightarrow \infty} \hat{\mathbf{x}}$. \square

Dies sei hier nicht weiter vertieft, da mein Hauptthema das CG-Verfahren ist und dessen Konvergenz sich später anderweitig ergibt.

Die verschiedenen Relaxationsverfahren wie Einzelschrittverfahren, Methode des steilsten Abstiegs, Gesamtschrittverfahren, CG unterscheiden sich durch die jeweilige Wahl der Relaxationsrichtungen \mathbf{p}_i und dadurch, daß für $\mathbf{x}_{i+1} = \mathbf{x}_i + t\mathbf{p}_i$ nicht unbedingt $t = t_0$ nach Lemma (7) genommen wird. Für letzteren Fall gilt dann i.a. die Konvergenzaussage (12) nicht mehr.

3 Das konjugierte-Gradienten-Verfahren

3.1 Das Verfahren

Beim CG-Verfahren wird die Relaxationsrichtung \mathbf{p}_i in Abhängigkeit vom vorherigen Gradienten \mathbf{r}_{i-1} und der vorherigen Suchrichtung \mathbf{p}_{i-1} und damit von \mathbf{r}_{i-2} bestimmt.

Dazu wieder ein paar geometrische Bemerkungen:

Eine Ebene im \mathbf{R}^n schneidet einen Ellipsoiden in einer Ellipse oder gar nicht. Für den Fall, daß die Ebene den Ellipsoiden in einem Punkt berührt, betrachtet man den Berührungspunkt als entartete Ellipse. Die Schar der ähnlichen Ellipsoiden $\{E_{F(\mathbf{x})} | \mathbf{x} \in \mathbf{R}^n\}$ mit Zentrum $\hat{\mathbf{x}}$ wird dann von einer Ebene in einer Schar ähnlicher Ellipsen mit Zentrum \mathbf{x}^* geschnitten. Liegt $\hat{\mathbf{x}}$ in dieser Ebene, so ist $\mathbf{x}^* = \hat{\mathbf{x}}$, andernfalls ist \mathbf{x}^* Berührungspunkt der Ebene mit einem Ellipsoiden E_{min} .

Der geometrische Ort der Mittelpunkte aller Sehnen, die einem Ellipsendurchmesser d_1 parallel sind, ist wieder ein Durchmesser d_2 . d_2 ist eindeutig und heißt der zu d_1 konjugierte Durchmesser. (vgl. z.B. GROTEMEYER)

Umgekehrt ist dann natürlich auch d_1 konjugiert zu d_2 , so daß man von konjugierten Durchmessern spricht (s. Bild 4).

(13) Durch die positiv definite Matrix $A \in \mathbf{R}^{n,n}$ sei eine Ellipse definiert. Es sind $\mathbf{x} \in \mathbf{R}^n$ und $\mathbf{y} \in \mathbf{R}^n$ die Richtungen zweier konjugierter Durchmesser genau dann, wenn $(A\mathbf{x}, \mathbf{y}) = 0$ ist.

\mathbf{x} und \mathbf{y} heißen dann A -konjugiert, im folgenden kurz konjugiert.

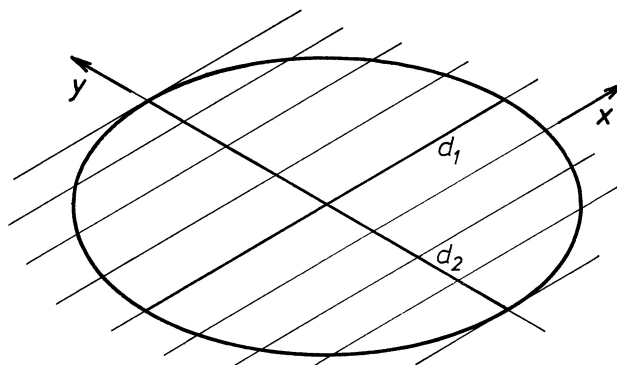


Bild 4

Für das CG-Verfahren sei nun ein Startvektor \mathbf{x}_0 gewählt und das Residuum $\mathbf{r}_0 = A\mathbf{x}_0 - \mathbf{b}$ berechnet. Wähle eine Relaxationsrichtung \mathbf{p}_0 , $(\mathbf{p}_0, \mathbf{r}_0) \neq 0$. Beim 'normalen' CG wählt man nun $\mathbf{p}_0 = -\mathbf{r}_0$; zwingend ist dies jedoch nicht.

Es wird für $G_{\mathbf{x}_0\mathbf{p}_0}$ t_0 bestimmt und $\gamma_0 := t_0$ gesetzt,

$$\mathbf{x}_1 = \mathbf{x}_0 + \gamma_0\mathbf{p}_0, \quad \mathbf{r}_1 = A\mathbf{x}_1 - \mathbf{b}.$$

Nach Lemma (8) gilt $(\mathbf{r}_1, \mathbf{p}_0) = 0$.

Für $k \geq 1$ sei \mathcal{E} die Ebene durch \mathbf{x}_k , die von \mathbf{r}_k und \mathbf{p}_{k-1} aufgespannt wird. Diese Ebene berührt E_{min} in \mathbf{x}^* (vgl. Vorbemerkungen). Von \mathbf{x}_k aus die beste Relaxationsrichtung in \mathcal{E} ist also nun die, die \mathbf{x}^* direkt erreicht, oder andersherum: Wird von \mathbf{x}_k aus \mathbf{p}_k so gewählt, daß \mathbf{x}^* auf $G_{\mathbf{x}_k \mathbf{p}_k}$ liegt, dann ist \mathbf{x}^* längs dieser Geraden und in dieser Ebene Minimalpunkt von F . Da \mathbf{x}_k schon als Berührungspunkt eines vorherigen Ellipsoiden $E_{min,v}$ bestimmt war, ist $G_{\mathbf{x}_k \mathbf{p}_{k-1}}$ Tangente an $E_{min,v}$ in \mathbf{x}_k und damit auch Tangente an eine Ellipse der durch \mathcal{E} ausgeschnittenen Ellipsenschar. Das gemeinsame Zentrum \mathbf{x}^* dieser Ellipsenschar liegt also auf der zu $G_{\mathbf{x}_k \mathbf{p}_{k-1}}$ konjugierten Geraden $G_{\mathbf{x}_k \mathbf{p}_k}$.

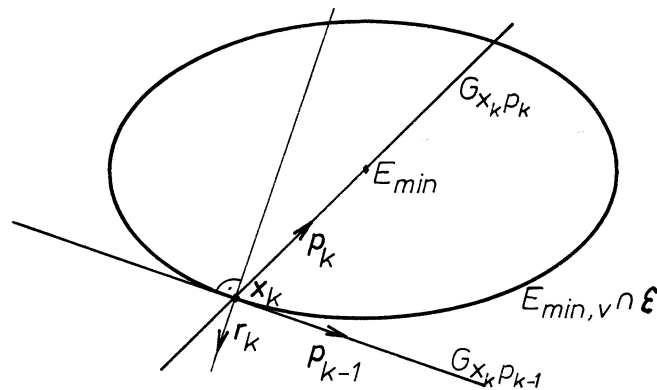


Bild 5

Nach (13) ist \mathbf{p}_k so zu bestimmen, daß $(A\mathbf{p}_k, \mathbf{p}_{k-1}) = 0$ gilt. Nach Konstruktion liegt \mathbf{p}_k in der durch \mathbf{r}_k und \mathbf{p}_{k-1} aufgespannten Ebene, ist also als Linearkombination darstellbar, wobei der Koeffizient von \mathbf{r}_k als ungleich null vorausgesetzt werden kann, denn das Gegenteil würde bedeuten, daß \mathbf{p}_k und \mathbf{p}_{k-1} parallel wären, also $(A\mathbf{p}_k, \mathbf{p}_{k-1}) \neq 0$. Man kann daher schreiben:

$$\mathbf{p}_k = -\mathbf{r}_k + \beta_{k-1}\mathbf{p}_{k-1}.$$

Damit ergibt sich

$$\begin{aligned} 0 &= (A\mathbf{p}_k, \mathbf{p}_{k-1}) = (\mathbf{p}_k, A\mathbf{p}_{k-1}) = (-\mathbf{r}_k + \beta_{k-1}\mathbf{p}_{k-1}, A\mathbf{p}_{k-1}) = \\ &= -(\mathbf{r}_k, A\mathbf{p}_{k-1}) + \beta_{k-1}(\mathbf{p}_{k-1}, A\mathbf{p}_{k-1}), \end{aligned}$$

also

$$(14) \quad \beta_{k-1} = \frac{(\mathbf{r}_k, A\mathbf{p}_{k-1})}{(A\mathbf{p}_{k-1}, \mathbf{p}_{k-1})}$$

und

$$(15) \quad \mathbf{p}_k = -\mathbf{r}_k + \frac{(\mathbf{r}_k, A\mathbf{p}_{k-1})}{(A\mathbf{p}_{k-1}, \mathbf{p}_{k-1})} \mathbf{p}_{k-1}.$$

Der Minimalpunkt \mathbf{x}_{k+1} auf $G_{\mathbf{x}_k \mathbf{p}_k}$ (Zentrum der Schnittellipsen) ist dann nach Lemma (7)

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{(\mathbf{r}_k, \mathbf{p}_k)}{(A\mathbf{p}_k, \mathbf{p}_k)} \mathbf{p}_k = \mathbf{x}_k + \gamma_k \mathbf{p}_k.$$

Damit ist der CG-Algorithmus gegeben; die Ausdrücke lassen sich aber noch vereinfachen. Die Residuenvektoren \mathbf{r}_k lassen sich rekursiv bestimmen, ohne die \mathbf{x}_k zu benutzen:

$$(16) \quad \mathbf{r}_{k+1} = A\mathbf{x}_{k+1} - \mathbf{b} = A\mathbf{x}_k + \gamma_k A\mathbf{p}_k - \mathbf{b} = \mathbf{r}_k + \gamma_k A\mathbf{p}_k$$

Da das Matrix-Vektor-Produkt $A\mathbf{p}_k$ sowieso benötigt wird zur Bestimmung von γ_k , spart man das Produkt $A\mathbf{x}_{k+1}$ und hat stattdessen eine Skalar-Vektor-Multiplikation durchzuführen.

Nach Lemma (8) gilt $(\mathbf{r}_k, \mathbf{p}_{k-1}) = 0$, so daß also

$$(\mathbf{r}_k, \mathbf{p}_k) = -(\mathbf{r}_k, \mathbf{r}_k) + \beta_{k-1}(\mathbf{r}_k, \mathbf{p}_{k-1}) = -(\mathbf{r}_k, \mathbf{r}_k)$$

und somit

$$\gamma_k = + \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(A\mathbf{p}_k, \mathbf{p}_k)} > 0 \text{ für } \mathbf{r}_k \neq \mathbf{0}$$

ist.

Da die Lösung gefunden ist, wenn $\mathbf{r}_{k+1} = \mathbf{0}$ ist, sei im folgenden $\mathbf{r}_k, \mathbf{r}_{k-1}, \dots, \mathbf{r}_0$ ungleich null vorausgesetzt. Damit erhält man aus (16)

$$A\mathbf{p}_k = \frac{1}{\gamma_k} (\mathbf{r}_{k+1} - \mathbf{r}_k).$$

Weiterhin ist aber nicht nur $(\mathbf{r}_{k+1}, \mathbf{p}_k) = 0$, sondern auch $(\mathbf{r}_{k+1}, \mathbf{y}) = 0$ für jeden in der betrachteten Ebene \mathcal{E} liegenden Vektor \mathbf{y} , also insbesondere auch für die beiden erzeugenden Vektoren \mathbf{r}_k und \mathbf{p}_{k-1} , denn \mathcal{E} ist ja Tangentialebene an E_{min} in \mathbf{x}_{k+1} . Es gilt daher

$$(17) \quad (\mathbf{r}_{k+1}, \mathbf{r}_k) = (\mathbf{r}_{k+1}, \mathbf{p}_{k-1}) = 0.$$

Damit wird

$$\begin{aligned}(\mathbf{r}_k, A\mathbf{p}_{k-1}) &= \frac{1}{\gamma_{k-1}}(\mathbf{r}_k, \mathbf{r}_k - \mathbf{r}_{k-1}) = \frac{1}{\gamma_{k-1}}(\mathbf{r}_k, \mathbf{r}_k) - \frac{1}{\gamma_{k-1}}(\mathbf{r}_k, \mathbf{r}_{k-1}) \\ &= \frac{1}{\gamma_{k-1}}(\mathbf{r}_k, \mathbf{r}_k),\end{aligned}$$

also

$$\begin{aligned}\beta_{k-1} &= \frac{1}{\gamma_{k-1}} \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(A\mathbf{p}_{k-1}, \mathbf{p}_{k-1})} = \frac{(A\mathbf{p}_{k-1}, \mathbf{p}_{k-1})}{(\mathbf{r}_{k-1}, \mathbf{r}_{k-1})} \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(A\mathbf{p}_{k-1}, \mathbf{p}_{k-1})} = \\ &= \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{r}_{k-1}, \mathbf{r}_{k-1})} > 0.\end{aligned}$$

Der CG-Algorithmus lautet dann:

- (18.1) $k := 0$, wähle Startvektor \mathbf{x}_k , $\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b}$, $\mathbf{p}_k = -\mathbf{r}_k$,
- (18.2) falls $\mathbf{r}_k = \mathbf{0}$, dann ist $\mathbf{x}_k = \hat{\mathbf{x}}$ die Lösung,
- (18.3) sonst berechne $A\mathbf{p}_k$,
 $\gamma_k = (\mathbf{r}_k, \mathbf{r}_k) / (A\mathbf{p}_k, \mathbf{p}_k)$,
 $\mathbf{r}_{k+1} = \mathbf{r}_k + \gamma_k A\mathbf{p}_k$,
 $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k \mathbf{p}_k$,
 $\beta_k = (\mathbf{r}_{k+1}, \mathbf{r}_{k+1}) / (\mathbf{r}_k, \mathbf{r}_k)$,
 $\mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$,
 $k := k + 1$, weiter mit (18.2).

Bemerkung: Die Berechnung von \mathbf{x}_k wird für die Durchführung des Algorithmus nicht benötigt.

3.2 Konvergenzbetrachtungen

Mit Satz (12) wurde im wesentlichen gezeigt, daß das allgemeine Relaxationsverfahren konvergiert, wenn der Winkel zwischen \mathbf{r}_i und \mathbf{p}_i mindestens um eine Konstante $\alpha \neq 0$ vom rechten Winkel abweicht (vgl. Bild 5). Diese Bedingung ist für das CG-Verfahren erfüllt, wobei α noch von der Exzentrizität der Ellipsoiden und damit von der Kondition der Matrix A , d.h. vom Verhältnis des größten Eigenwertes von A zum kleinsten abhängt. Das CG-Verfahren hat jedoch eine interessante Konvergenzeigenschaft:

Satz: Die $\mathbf{r}_i, i = 0, 1, \dots$ bilden ein Orthogonalsystem. Ist ferner m der kleinste Index, für den $\mathbf{r}_m = \mathbf{0}$ ist, also $\mathbf{x}_m = \hat{\mathbf{x}}$ Lösung ist, dann bilden die $\mathbf{p}_i, i = 0, 1, \dots, m-1$ ein System konjugierter Richtungen:

- a) $(\mathbf{r}_i, \mathbf{r}_j) = 0$ für $i \neq j$
- b) $(A\mathbf{p}_i, \mathbf{p}_j) = 0$ für $i \neq j; i, j < m$.

Beweis durch vollständige Induktion:

(i) Induktionsanfang, $k = 1$

- a) $(\mathbf{r}_0, \mathbf{r}_1) = 0$ nach (18.1) und (8)

Bemerkung: Bei einer anderen Wahl der Relaxationsrichtung \mathbf{p}_0 als $\pm \mathbf{r}_0$ gilt diese Aussage erst ab $k = 2$ nach (17).

$$\begin{aligned} \text{b) } (A\mathbf{p}_0, \mathbf{p}_1) &= (A\mathbf{p}_0, -\mathbf{r}_1 + \beta_1 \mathbf{p}_0) \\ &= -(A\mathbf{p}_0, \mathbf{r}_1) + \frac{(\mathbf{r}_1, A\mathbf{p}_0)}{(A\mathbf{p}_0, \mathbf{p}_0)} (A\mathbf{p}_0, \mathbf{p}_0) \\ &= 0 \end{aligned}$$

(ii) Induktionsvoraussetzung, es gelte:

- a) $(\mathbf{r}_i, \mathbf{r}_j) = 0$ für $0 \leq i \neq j \leq k; \mathbf{r}_0, \dots, \mathbf{r}_k \neq \mathbf{0}$
- b) $(\mathbf{p}_i, A\mathbf{p}_j) = 0$ für $0 \leq i \neq j < k; \mathbf{p}_0, \dots, \mathbf{p}_{k-1} \neq \mathbf{0}$

(iii) Induktionsschluß $k \rightarrow k + 1$ bzw. $k - 1 \rightarrow k$,
zu zeigen ist:

- a) $(\mathbf{r}_{k+1}, \mathbf{r}_j) = 0$ für $0 \leq j \leq k$
- b) $(\mathbf{p}_k, A\mathbf{p}_j) = 0$ für $0 \leq j < k$.

Für $j \leq k - 2$ gilt:

$$\begin{aligned} \text{b) } (\mathbf{p}_k, A\mathbf{p}_j) &= -(\mathbf{r}_k, A\mathbf{p}_j) + \beta_k (\mathbf{p}_{k-1}, A\mathbf{p}_j) \\ &= -(\mathbf{r}_k, A\mathbf{p}_j) \\ &= -\frac{1}{\gamma_{j+1}} (\mathbf{r}_k, \mathbf{r}_{j+1} - \mathbf{r}_j) \\ &= 0. \end{aligned}$$

Für $j = k - 1$ gilt $(\mathbf{p}_k, A\mathbf{p}_{k-1}) = 0$ nach Konstruktion (14), (15) analog zu (i) b).

- a) für $j = k$ gilt $(\mathbf{r}_{k+1}, \mathbf{r}_k) = 0$ nach (17),

für $1 \leq j < k$ gilt:

$$\begin{aligned}(\mathbf{r}_{k+1}, \mathbf{r}_j) &= (\mathbf{r}_k + \gamma_k A\mathbf{p}_k, \mathbf{r}_j) \\ &= \gamma_k (A\mathbf{p}_k, \mathbf{r}_j) \\ &= \gamma_k (A\mathbf{p}_k, -\mathbf{p}_j + \beta_j \mathbf{p}_{j-1}) \\ &= 0.\end{aligned}$$

Für $j = 0$ ist $(\mathbf{r}_{k+1}, \mathbf{r}_0) = \gamma_k (A\mathbf{p}_k, -\mathbf{p}_0) = 0$. \square

Folgerung: Es gilt $\mathbf{r}_m = \mathbf{0}$ für $m \leq n$, d.h. die exakte Lösung wird nach höchstens n Schritten erreicht, denn da die \mathbf{r}_i paarweise orthogonal sind, im \mathbf{R}^n aber nur eine Menge von höchstens n Vektoren linear unabhängig sein kann, können nicht mehr als n Residuenvektoren ungleich null sein.

Dieses Ergebnis ist aber nur theoretischer Natur, denn praktisch werden zwei Probleme auftreten:

1. $\mathbf{r}_0, \dots, \mathbf{r}_n \neq \mathbf{0}$ infolge Rundungsfehler,
2. bei großen Systemen möchte man nach relativ wenigen Schritten eine brauchbare Näherungslösung haben, d.h. man möchte das CG-Verfahren trotz seiner theoretischen Eigenschaft als endliches Verfahren eher als Iterationsverfahren benutzen. Man kann wegen Bemerkung (9) erwarten, daß dies möglich ist.

Beide Probleme haben mit der Kondition der Matrix A zu tun. Da man es in der Praxis mit großen Systemen zu tun hat, ist das zweite Problem das hier weiter interessierende, und man benötigt daher eine Abschätzung des Fehlers für $k \leq m$ Schritte.

Zur Herleitung einer Fehlerabschätzung

Wegen $\mathbf{r} = A\mathbf{x} - \mathbf{b} = A(\mathbf{x} - \hat{\mathbf{x}})$ ist $A^{-1}\mathbf{r} = \mathbf{x} - \hat{\mathbf{x}}$,

$$\begin{aligned}
 (19) \quad \|\mathbf{r}\|_{A^{-1}}^2 &= (A^{-1}\mathbf{r}, \mathbf{r}) = (\mathbf{x} - \hat{\mathbf{x}}, A(\mathbf{x} - \hat{\mathbf{x}})) = \|\mathbf{x} - \hat{\mathbf{x}}\|_A^2 \\
 &= (A\mathbf{x}, \mathbf{x}) - 2(A\mathbf{x}, \hat{\mathbf{x}}) + (A\hat{\mathbf{x}}, \hat{\mathbf{x}}) + (A\hat{\mathbf{x}}, \hat{\mathbf{x}}) - (A\hat{\mathbf{x}}, \hat{\mathbf{x}}) \\
 &= (A\mathbf{x}, \mathbf{x}) - 2(\mathbf{b}, \mathbf{x}) - ((A\hat{\mathbf{x}}, \hat{\mathbf{x}}) - 2(\mathbf{b}, \hat{\mathbf{x}})) \\
 &= 2(F(\mathbf{x}) - F(\hat{\mathbf{x}})).
 \end{aligned}$$

Bemerkung: Dies erklärt, warum $\|\cdot\|_A$ Energienorm genannt wird, denn bei Interpretation des Funktionals F als potentielle Energie (z.B. Höhe) gibt diese Norm ein Maß für die potentielle Energie im Punkt \mathbf{x} bezogen auf die potentielle Energie im Punkt $\hat{\mathbf{x}}$.

Der Vektor \mathbf{p}_k läßt sich nach (15) darstellen als Linearkombination der $\mathbf{r}_k, \dots, \mathbf{r}_0$, so daß nach (16) \mathbf{r}_k als Linearkombination von $\mathbf{r}_0, A\mathbf{r}_0, \dots, A^k\mathbf{r}_0$ darstellbar ist, also

$$\mathbf{r}_k = \mathbf{r}_0 + \sum_{i=1}^k \alpha_i^{(k)} A^i \mathbf{r}_0.$$

Die beiden folgenden Lemmata zeigen, daß \mathbf{r}_k eine Komponente in Richtung $A^k\mathbf{r}_0$ hat.

Lemma: Für den höchsten Koeffizienten gilt $\alpha_k^{(k)} = (-1)^k \prod_{i=0}^{k-1} \gamma_i$.

Beweis: Nach (18) gilt

$$\begin{aligned}
 \mathbf{r}_k &= \mathbf{r}_{k-1} + \gamma_{k-1} A \mathbf{p}_{k-1} \\
 &= \mathbf{r}_{k-1} + \gamma_{k-1} A(-\mathbf{r}_{k-1} + \beta_{k-2} \mathbf{p}_{k-2}) \\
 &= \mathbf{r}_{k-1} + \gamma_{k-1} A(-\mathbf{r}_{k-2} - \gamma_{k-2} A \mathbf{p}_{k-2} + \beta_{k-2} \mathbf{p}_{k-2}) \\
 &= \mathbf{r}_{k-1} + \underline{\gamma_{k-1} A(-\mathbf{r}_{k-2} - \gamma_{k-2} A(\mathbf{r}_{k-2} + \beta_{k-3} \mathbf{p}_{k-3}) + \beta_{k-2} \mathbf{p}_{k-2})}.
 \end{aligned}$$

Dieses wechselseitige Ersetzen von \mathbf{p}_i durch $-\mathbf{r}_i + \beta_{i-1} \mathbf{p}_{i-1}$ und darin \mathbf{r}_i durch $\mathbf{r}_{i-1} + \gamma_{i-1} A \mathbf{p}_{i-1}$ läßt sich fortführen, bis \mathbf{p}_0 durch $-\mathbf{r}_0$ zu ersetzen ist. Das Produkt der unterstrichenen Ausdrücke ist dann $(-1)^k \prod_{i=0}^{k-1} \gamma_i A^k \mathbf{r}_0$. Bei allen anderen Ersetzungen, die nötig sind, um aus dieser Gleichung die Polynomform zu erhalten, treten nur noch Potenzen von A auf, die kleiner als k sind. \square

Lemma: $\alpha_k^{(k)} \neq 0$.

Beweis: Solange die Lösung nicht gefunden ist, bricht der Algorithmus nicht ab, d.h. nach (18.3) sind die $\gamma_i \neq 0$. \square

Mit der linearen Hülle $S_k := \text{span} \{A\mathbf{r}_0, A^2\mathbf{r}_0, \dots, A^k\mathbf{r}_0\}$ und der Menge $M_k := \{\mathbf{r} \in \mathbf{R}^n \mid \mathbf{r} = \mathbf{r}_0 + \mathbf{h}, \mathbf{h} \in S_k\}$ ist dann $\mathbf{r}_k \in M_k$ mit einer Komponente in Richtung $A^k\mathbf{r}_0$.

Nach Bemerkung (9) ist F streng monoton fallend bezüglich k , so daß wegen (19) gilt:

$$(20) \quad \|\mathbf{x}_k - \hat{\mathbf{x}}\|_A = \|\mathbf{r}_k\|_{A^{-1}} = \min_{\mathbf{r} \in M_k} \|\mathbf{r}\|_{A^{-1}}.$$

Für die weiteren Betrachtungen sei \prod_k^1 die Menge der Polynome P_k vom Grad k , für die $P_k(0) = 1$ gilt und

$$\tilde{M}_k := \{\mathbf{r} \in \mathbf{R}^n \mid \mathbf{r} = P_k(A)\mathbf{r}_0, P_k \in \prod_k^1\} \subset M_k.$$

Da $\alpha_k^{(k)} \neq 0$ ist, ist $\mathbf{r}_k \in \tilde{M}_k$ und es folgt aus (20) weiter

$$(21) \quad \begin{aligned} \|\mathbf{r}_k\|_{A^{-1}} &= \min_{\mathbf{r} \in \tilde{M}_k} \|\mathbf{r}\|_{A^{-1}} = \min_{P_k \in \prod_k^1} \|P_k(A)\mathbf{r}_0\|_{A^{-1}} = \\ &= \min_{P_k \in \prod_k^1} (A^{-1}P_k(A)\mathbf{r}_0, P_k(A)\mathbf{r}_0)^{1/2}. \end{aligned}$$

(22) **Satz:** $M \subset \mathbf{R}$ enthalte alle Eigenwerte von A und für eine Schranke $s \geq 0$ und ein Polynom $\tilde{P}_k \in \prod_k^1$ gelte

$$\max_{\lambda \in M} |\tilde{P}_k(\lambda)| \leq s,$$

dann gilt:

$$\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A \leq s \|\mathbf{x}_0 - \hat{\mathbf{x}}\|_A.$$

Beweis: Seien $0 < \lambda_1 \leq \dots \leq \lambda_n$ die Eigenwerte von A . Da A symmetrisch ist, ist A normal und es existiert eine Orthonormalbasis des \mathbf{R}^n aus Eigenvektoren \mathbf{v}_i von A . Das erste Residuum hat dann eine Darstellung

$$\mathbf{r}_0 = \sum_{i=1}^n \alpha_i \mathbf{v}_i \text{ mit } \alpha_i = (\mathbf{v}_i, \mathbf{r}_0).$$

Damit ergibt sich

$$\begin{aligned} (A^{-1}P_k(A)\mathbf{r}_0, P_k(A)\mathbf{r}_0) &= \left(A^{-1}P_k(A) \sum_{i=1}^n \alpha_i \mathbf{v}_i, P_k(A) \sum_{i=1}^n \alpha_i \mathbf{v}_i \right) \\ &= \left(\sum_{i=1}^n \alpha_i A^{-1}P_k(A)\mathbf{v}_i, \sum_{i=1}^n \alpha_i P_k(A)\mathbf{v}_i \right) \\ &= \left(\sum_{i=1}^n \alpha_i A^{-1}P_k(\lambda_i)\mathbf{v}_i, \sum_{i=1}^n \alpha_i P_k(\lambda_i)\mathbf{v}_i \right) \\ &= \sum_{i=1}^n \alpha_i P_k(\lambda_i) \left(\sum_{j=1}^n \alpha_j P_k(\lambda_j) (A^{-1}\mathbf{v}_i, \mathbf{v}_j) \right) \\ &= \sum_{i=1}^n \alpha_i^2 (P_k(\lambda_i))^2 \frac{1}{\lambda_i}. \end{aligned}$$

Nach (21) ist dann

$$\|\mathbf{r}_k\|_{A^{-1}}^2 = \min_{P_k \in \prod_k^1} \sum_{i=1}^n \alpha_i^2 (P_k(\lambda_i))^2 \frac{1}{\lambda_i}$$

und für $P_k = \tilde{P}_k$

$$\begin{aligned}
\|\mathbf{r}_k\|_{A^{-1}}^2 &\leq \sum_{i=1}^n \alpha_i^2 (\tilde{P}_k(\lambda_i))^2 \frac{1}{\lambda_i} \\
&\leq \sum_{i=1}^n \alpha_i^2 s^2 \frac{1}{\lambda_i} \\
&= s^2 \sum_{i=1}^n (\mathbf{v}_i, \mathbf{r}_0) \alpha_i \frac{1}{\lambda_i} = s^2 \sum_{i=1}^n \left(\frac{1}{\lambda_i} \mathbf{v}_i, \mathbf{r}_0\right) \alpha_i \\
&= s^2 \sum_{i=1}^n (A^{-1} \mathbf{v}_i, \mathbf{r}_0) \alpha_i = s^2 \left(\sum_{i=1}^n A^{-1} \alpha_i \mathbf{v}_i, \mathbf{r}_0\right) \\
&= s^2 (A^{-1} \mathbf{r}_0, \mathbf{r}_0) = s^2 \|\mathbf{r}_0\|_{A^{-1}}^2,
\end{aligned}$$

also $\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A = \|\mathbf{r}_k\|_{A^{-1}} \leq s \|\mathbf{r}_0\|_{A^{-1}} = s \|\mathbf{x}_0 - \hat{\mathbf{x}}\|_A$. \square

Folgerung: Ist m die Anzahl der verschiedenen Eigenwerte von A , dann ist $\mathbf{x}_m = \hat{\mathbf{x}}$.

Beweis: Das Polynom $P_k^E(\lambda) = \prod_{j=1}^k (\lambda_j - \lambda)/\lambda_j$ hat den Grad k , die Nullstellen $0 \neq \lambda_j, j = 1, \dots, k$ und an der Stelle 0 den Wert 1, also $P_k^E \in \Pi_k^1$.

Es wurde oben gezeigt, daß $\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A^2 = \min_{P_k \in \Pi_k^1} \sum_{i=1}^n \alpha_i^2 (P_k(\lambda_i))^2 \frac{1}{\lambda_i}$ ist, mithin ist $\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A^2 \leq \sum_{i=1}^n \alpha_i^2 (P_k^E(\lambda_i))^2 \frac{1}{\lambda_i}$.

Seien $\lambda_j, j = 1, \dots, m$ die verschiedenen Eigenwerte von A . Falls $k < m$ ist, dann gibt es ein $i \in \{1, \dots, n\}$, z.B. $i = m$, so daß $P_k^E(\lambda_i) \neq 0$ ist. Ist $k \geq m$, dann ist für alle $i \in \{1, \dots, n\}$ $P_k^E(\lambda_i) = 0 = \sum_{j=1}^n \alpha_j^2 (P_k(\lambda_j))^2 \frac{1}{\lambda_j} \geq \|\mathbf{x}_k - \hat{\mathbf{x}}\|_A^2$. \square

O.B.d.A. kann man voraussetzen, daß A mindestens zwei verschiedene Eigenwerte besitzt, also $0 < \lambda_1 < \lambda_n$; andernfalls ist A diagonalähnlich zu einem Vielfachen der Einheitsmatrix, die Ellipsoide sind dann Sphären und die Lösung wird in einem Schritt gefunden.

Um eine gute Schranke s zu finden, sucht man nun ein Polynom $\tilde{P}_k \in \Pi_k^1$, dessen betragsgrößter Wert auf $[\lambda_1, \lambda_n]$ möglichst klein ist, d.h. \tilde{P}_k mit der Eigenschaft

$$(23) \quad \max_{\lambda \in [\lambda_1, \lambda_n]} |\tilde{P}_k(\lambda)| = \min_{P_k \in \Pi_k^1} \max_{\lambda \in [\lambda_1, \lambda_n]} |P_k(\lambda)|.$$

Lemma: Das Minimierungsproblem (23) wird durch

$$\tilde{P}_k(\lambda) = \frac{T_k((\lambda_n + \lambda_1 - 2\lambda)/(\lambda_n - \lambda_1))}{T_k((\lambda_n + \lambda_1)/(\lambda_n - \lambda_1))}$$

mit den Tschebyscheffpolynomen T_k vom Grad k

$$T_k(x) = \frac{1}{2} \left((x + \sqrt{x^2 - 1})^k + (x - \sqrt{x^2 - 1})^k \right)$$

gelöst.

Beweis: Für $x \in [-1, 1]$ hat T_k auch die Darstellung $T_k(x) = \cos(k \arccos x)$, so daß $|T_k(x)| \leq 1$ für $x \in [-1, 1]$ ist; für $x_i = \cos(i\pi/k)$, $i = 0, 1, \dots, k$ ist $T_k(x_i) = \cos(k \arccos(\cos(i\pi/k))) = \cos i\pi = (-1)^i$. Die Abbildung

$$T : [\lambda_1, \lambda_n] \rightarrow [-1, 1]$$

$$\lambda \mapsto \frac{\lambda_n + \lambda_1 - 2\lambda}{\lambda_n - \lambda_1}$$

ist eine Bijektion, so daß also für $\lambda \in [\lambda_1, \lambda_n]$ der Zähler von $\tilde{P}_k(\lambda)$ dem Betrag nach höchstens 1 ist und an $k + 1$ Punkten $\lambda^{(i)}$ abwechselnd 1 und -1; der Nenner von $\tilde{P}_k(\lambda)$ ist größer als 1. Sei nun $P_k \in \Pi_k^1$ mit

$$\max_{\lambda \in [\lambda_1, \lambda_n]} |P_k(\lambda)| < \max_{\lambda \in [\lambda_1, \lambda_n]} |\tilde{P}_k(\lambda)|$$

und

$$R(\lambda) := \tilde{P}_k(\lambda) - P_k(\lambda).$$

Mit dieser Voraussetzung an P_k hat dann $R(\lambda)$ an den $k+1$ Punkten $\lambda^{(i)}$ abwechselnd positives und negatives Vorzeichen und daher wegen der Stetigkeit k verschiedene Nullstellen zwischen den $\lambda^{(i)}$. Außerdem ist $R(0) = \tilde{P}_k(0) - P_k(0) = 0$, da $\tilde{P}_k, P_k \in \Pi_k^1$, somit hat R als Polynom vom Grad höchstens k $k + 1$ Nullstellen, ist also das Nullpolynom im Widerspruch zur Voraussetzung an P_k . \square

Es gilt also

$$\max_{\lambda \in [\lambda_1, \lambda_n]} |\tilde{P}_k| \leq 1/T_k((\lambda_n + \lambda_1)/(\lambda_n - \lambda_1)).$$

Satz (22) läßt sich dann so verschärfen:

$$(24) \quad \|\mathbf{x}_k - \hat{\mathbf{x}}\|_A \leq \frac{1}{T_k((\lambda_n + \lambda_1)/(\lambda_n - \lambda_1))} \|\mathbf{x}_0 - \hat{\mathbf{x}}\|_A.$$

Im nächsten Kapitel wird die Konditionszahl $\kappa(A)$ einer symmetrischen, positiv-definiten Matrix A definiert als $\kappa(A) = \lambda_n/\lambda_1 > 1$; jetzt wird mit κ aus (24) eine praktische Abschätzung hergeleitet.

Für $\kappa = \lambda_n/\lambda_1 > 1$ ist

$$(25) \quad T_k\left(\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1}\right) = T_k\left(\frac{\kappa + 1}{\kappa - 1}\right) =$$

$$\begin{aligned}
&= \frac{1}{2} \left(\left(\frac{\kappa+1}{\kappa-1} + \sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2 - 1} \right)^k + \left(\frac{\kappa+1}{\kappa-1} - \sqrt{\left(\frac{\kappa+1}{\kappa-1}\right)^2 - 1} \right)^k \right) \\
&= \frac{1}{2} \left(\left(\frac{\kappa+2\sqrt{\kappa}+1}{\kappa-1} \right)^k + \left(\frac{\kappa-2\sqrt{\kappa}+1}{\kappa-1} \right)^k \right) \\
&= \frac{1}{2} \left(\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^k + \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \right)^k \right) \\
&> \frac{1}{2} \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^k.
\end{aligned}$$

Aus der Reihenentwicklung für den Logarithmus erhält man

$$(26) \quad \ln \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right) > \frac{2}{\sqrt{\kappa}} \quad \text{für } \kappa > 1.$$

Damit läßt sich eine Art Umkehrung von Satz (22) formulieren:

(27) **Satz:** Es sei $0 < \varepsilon < 1$. Der kleinste Index k_ε , für den $\|\mathbf{x}_{k_\varepsilon} - \hat{\mathbf{x}}\|_A \leq \varepsilon \|\mathbf{x}_0 - \hat{\mathbf{x}}\|_A$ gilt, erfüllt die Ungleichung

$$k_\varepsilon \leq \frac{1}{2} \sqrt{\kappa(A)} \ln \left(\frac{2}{\varepsilon} \right) + 1.$$

Beweis: Es ist nach (24) k_ε so zu bestimmen, daß

$$\frac{1}{T_{k_\varepsilon} \left(\frac{\kappa+1}{\kappa-1} \right)} < \varepsilon,$$

also

$$T_{k_\varepsilon} \left(\frac{\kappa+1}{\kappa-1} \right) > \frac{1}{\varepsilon}.$$

Dies gilt nach (25), wenn

$$\frac{1}{2} \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1} \right)^{k_\varepsilon} > \frac{1}{\varepsilon},$$

also

$$k_\varepsilon > \frac{\ln(2/\varepsilon)}{\ln((\sqrt{\kappa}+1)/(\sqrt{\kappa}-1))},$$

mit (26) also erst recht für

$$k_\varepsilon > \frac{\ln(2/\varepsilon)}{2/\sqrt{\kappa}}.$$

Das bedeutet, daß k_ε unter den natürlichen Zahlen zu finden ist, die kleiner oder gleich $\frac{1}{2}\sqrt{\kappa} \ln(2/\varepsilon) + 1$ sind. \square

Der Satz gibt also an, wieviele Iterationen höchstens auszuführen sind, um einen vorgegebenen relativen Fehler zu unterschreiten.

Ist also zum Beispiel die Konditionszahl einer Matrix $\kappa = 100$, so hat man mit Sicherheit nach $k = 74$ Iterationsschritten einen relativen Fehler in der A -Norm von weniger als 10^{-6} : $\frac{\|\mathbf{x}_{74} - \hat{\mathbf{x}}\|_A}{\|\mathbf{x}_0 - \hat{\mathbf{x}}\|_A} < 10^{-6}$.

Bei der Durchführung des Algorithmus kann man aber weder die linke noch die rechte Seite der Identität $\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A = \|\mathbf{r}_k\|_{A^{-1}}$ bestimmen, da man weder die Lösung $\hat{\mathbf{x}}$ noch die Inverse A^{-1} zur Verfügung hat. Als Abbruchkriterium bleibt also nur $\|\mathbf{r}_k\| < \varepsilon$ (oder \mathbf{r}_k gemessen in einer anderen leicht zu bestimmenden Norm). Die numerischen Ergebnisse zeigen, daß die Iterationsanzahl k bis zum Erfüllen von $\|\mathbf{r}_k\| < \varepsilon$ meist ebenfalls die Ungleichung aus Satz (27) erfüllt.

4 Vorkonditioniertes CG-Verfahren

4.1 Vorbemerkungen

In der Konvergenzabschätzung (27) spielt die Kondition $\kappa = \kappa(A) = \lambda_n / \lambda_1$, λ_n größter, $\lambda_1 > 0$ kleinster Eigenwert von A , eine wesentliche Rolle in dem Sinne, daß durch eine Verkleinerung von κ auch eine Verminderung der nötigen Zahl von Iterationsschritten zum Unterschreiten eines vorgegebenen relativen Fehlers erreicht wird. Dies ist die Hauptmotivation der Vorkonditionierung, eine weitere ist folgende:

Bei der Durchführung eines Iterationsalgorithmus steht einem naheliegenderweise aber der Fehler $\mathbf{f}_k := \mathbf{x}_k - \hat{\mathbf{x}}$ in den Urbildern \mathbf{x} der linearen Abbildung A nicht zur Verfügung. Beim CG-Algorithmus (18) hat man jedoch den Fehler $\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b} = A\mathbf{f}_k$ in den Bildwerten, das Residuum, zur Verfügung. Allgemein gilt

$$(28) \quad \|\mathbf{f}\| = \|A^{-1}\mathbf{r}\| \leq \|A^{-1}\| \|\mathbf{r}\| = \frac{1}{\lambda_1} \|\mathbf{r}\|$$

für die euklidische Vektornorm und die mir ihr verträgliche Spektralnorm, was bedeutet, daß man ohne Kenntnis des kleinsten Eigenwertes nicht ohne weiteres von $\|\mathbf{r}_k\|$ auf die Güte der Näherung \mathbf{x}_k schließen kann, denn Ungleichung (28) ist scharf. Es ist nämlich durchaus folgender Fall möglich:

$\|\mathbf{r}_1\|_{A^{-1}} = \|\mathbf{x}_1 - \hat{\mathbf{x}}\|_A = \|\mathbf{x}_2 - \hat{\mathbf{x}}\|_A = \|\mathbf{r}_2\|_{A^{-1}}$ und $\|\mathbf{x}_1 - \hat{\mathbf{x}}\| \ll \|\mathbf{x}_2 - \hat{\mathbf{x}}\|$, aber $\|\mathbf{r}_2\| < \|\mathbf{r}_1\|$.

Dazu ein Beispiel:

$$A = \begin{pmatrix} 17 & -15 \\ -15 & 17 \end{pmatrix}, b = (49, -47)^T, \hat{\mathbf{x}} = (2, -1)^T, F(\hat{\mathbf{x}}) = -72.5,$$

die Eigenwerte sind $\lambda_1 = 2$, $\lambda_2 = 32$, die zugehörigen Eigenvektoren $\mathbf{v}_1 = (1, 1)^T$, $\mathbf{v}_2 = (1, -1)^T$. Betrachtet werden nun zwei Näherungen $\mathbf{x}_1 = (1.75, -0.75)^T$ und $\mathbf{x}_2 = (3, 0)^T$. Die zugehörigen Fehlernormen sind

$$\|\mathbf{f}_1\| = \sqrt{2}/4, \|\mathbf{r}_1\|_{A^{-1}} = 2, \|\mathbf{r}_1\| = 8\sqrt{2} \text{ bzw.}$$

$$\|\mathbf{f}_2\| = \sqrt{2}, \|\mathbf{r}_2\|_{A^{-1}} = 2, \|\mathbf{r}_2\| = 2\sqrt{2} \text{ und}$$

$F(\mathbf{x}_1) = F(\mathbf{x}_2) = -70.5$. Es ist also

$$\|\mathbf{f}_1\| < \frac{1}{\lambda_1} \|\mathbf{r}_1\|, \|\mathbf{f}_2\| = \frac{1}{\lambda_1} \|\mathbf{r}_2\| \text{ und generell } \|\mathbf{f}\| < \|\mathbf{r}\|, \text{ da } \lambda_1 > 1.$$

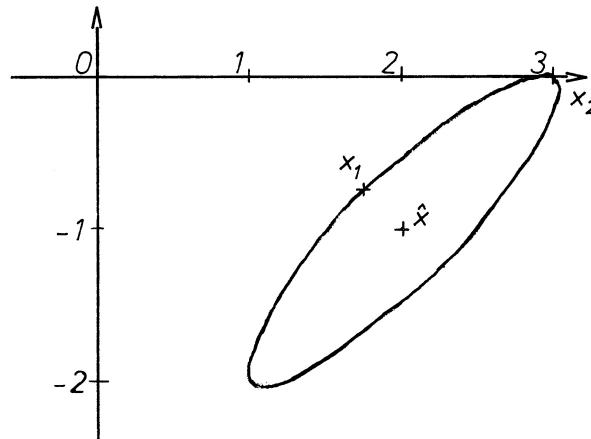


Bild 6

Setze nun $A' = \frac{1}{8}A$, $\mathbf{b}' = \frac{1}{8}\mathbf{b}$, dann hat das neue System die Eigenwerte $\lambda_1 = 1/4$, $\lambda_2 = 4$. Bild 6 ist auch hierfür gültig. Für die gleichen Näherungen \mathbf{x}_1 und \mathbf{x}_2 wird dann $\|\mathbf{r}_1\| = \sqrt{2}$, $\|\mathbf{r}_2\| = \sqrt{2}/4$, so daß also

$$\|\mathbf{f}_1\| = \sqrt{2}/4 < \sqrt{2} = \|\mathbf{r}_1\| \text{ und } \|\mathbf{f}_2\| = \sqrt{2} > \sqrt{2}/4 = \|\mathbf{r}_2\| \text{ ist.}$$

Die restlichen Relationen sind dieselben wie vorher. Das Beispiel zeigt auch, daß durch eine Verkleinerung der Kondition $\kappa(A)$, also eine Verminderung der Exzentrizität der Ellipsoiden, die Differenz in den möglichen Fehlern $\|\mathbf{f}_1\|$ und $\|\mathbf{f}_2\|$ und damit die Diskrepanz zwischen Energienorm und euklidischer Norm vermindert wird, d.h. man hat die Abschätzung (27) auch für \mathbf{x}_k und nicht nur für \mathbf{r}_k . Die Verminderung der Exzentrizität bedeutet aber auch, nach einer bestimmten Anzahl von Iterationsschritten einen kleineren Fehler \mathbf{r} zu haben.

Anschaulich besteht die Idee der Vorkonditionierung nun darin, das System so zu transformieren, daß die Ellipsoiden des neuen Systems weniger exzentrisch, also mehr sphärenähnlich sind, mit diesem System das CG-Verfahren durchzuführen und die Lösung zurückzutransformieren.

4.2 Der Algorithmus

Man geht also über zu dem Problem, die Lösung $\hat{\mathbf{y}}$ des Systems

$$(29) \quad \tilde{A}\mathbf{y} = \tilde{\mathbf{b}}$$

zu finden, wobei $\kappa(\tilde{A}) < \kappa(A)$ gelten soll.

Da die auftretenden Transformationen $\mathbf{y} \leftrightarrow \mathbf{x}$, $\tilde{\mathbf{b}} \leftrightarrow \mathbf{b}$ mit möglichst wenig Rechenaufwand zu bewerkstelligen sein sollen, sei also $L \in \mathbf{R}^{n,n}$ eine reguläre untere Dreiecksmatrix; die grundlegende Transformation ist dann

$$\mathbf{y} := L^T \mathbf{x}$$

und das Funktional \tilde{F} wird definiert zu

$$\begin{aligned} \tilde{F}(\mathbf{y}) &:= F(L^{-T} \mathbf{y}) = \frac{1}{2}(AL^{-T} \mathbf{y}, L^{-T} \mathbf{y}) - (\mathbf{b}, L^{-T} \mathbf{y}) \\ &= \frac{1}{2}(L^{-1}AL^{-T} \mathbf{y}, \mathbf{y}) - (L^{-1}\mathbf{b}, \mathbf{y}), \end{aligned}$$

so daß also

$$(30) \quad \tilde{A} = L^{-1}AL^{-T}, \tilde{\mathbf{b}} = L^{-1}\mathbf{b} \text{ ist.}$$

Mit A ist auch, da L regulär, \tilde{A} positiv-definit, so daß auf das System (29) das CG-Verfahren angewendet werden kann. Weiter gilt

$$(31) \quad L^{-T}\tilde{A}L^T = L^{-T}L^{-1}AL^{-T}L^T = L^{-T}L^{-1}A.$$

Definiert man $C := LL^T$; dann sind \tilde{A} und $C^{-1}A$ ähnlich, haben also die gleichen Eigenwerte, womit $\kappa(\tilde{A}) = \kappa(C^{-1}A)$ ist.

Das praktische Vorgehen bei der Vorkonditionierung ist nun umgekehrt, indem zuerst eine positiv-definite Matrix C gesucht wird und diese dann in Dreiecksfaktoren zerlegt wird. Dann besagt (31), daß, wenn auch \tilde{A} von der Zerlegung von C abhängt, die Konditionszahl von \tilde{A} davon unabhängig ist.

Für den Algorithmus ist es ungünstig, ihn auf der Basis der tatsächlich transformierten Matrix \tilde{A} durchzuführen, da diese Matrix im allgemeinen voll besetzt ist. Vielmehr ist es praktisch günstig, mit diesem vorkonditionierten Verfahren die Lösung $\hat{\mathbf{x}}$ des ursprünglichen Systems zu berechnen. Das folgende Lemma erlaubt dies.

(32) **Lemma:** Das vorkonditionierte CG-Verfahren (PCG) sei folgendermaßen definiert:

$$(32.1) \quad \gamma_k = (\mathbf{r}_k, \mathbf{h}_k) / (A\mathbf{p}_k, \mathbf{p}_k)$$

$$(32.2) \quad \mathbf{r}_{k+1} = \mathbf{r}_k + \gamma_k A\mathbf{p}_k$$

$$(32.3) \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k \mathbf{p}_k$$

$$(32.4) \quad \mathbf{h}_{k+1} = C^{-1}\mathbf{r}_{k+1}$$

$$(32.5) \quad \beta_k = (\mathbf{r}_{k+1}, \mathbf{h}_{k+1}) / (\mathbf{r}_k, \mathbf{h}_k)$$

$$(32.5) \quad \mathbf{p}_{k+1} = -\mathbf{h}_{k+1} + \beta_k \mathbf{p}_k$$

mit den Anfangsgrößen \mathbf{x}_0 , $\mathbf{r}_0 = A\mathbf{x}_0 - \mathbf{b}$, $\mathbf{h}_0 = C^{-1}\mathbf{r}_0$, $\mathbf{p}_0 = -\mathbf{h}_0$ und $C = LL^T$. Sind dann $\mathbf{y}_k, \tilde{\mathbf{r}}_k, \tilde{\mathbf{p}}_k, \tilde{\gamma}_k, \tilde{\beta}_k$ die Größen, die durch Anwendung des CG-Verfahrens auf das System (29), (30) entstehen und ist $\mathbf{y}_0 = L^T \mathbf{x}_0$, dann gilt für alle $k \in \mathbf{N}_0$

1. $\mathbf{y}_k = L^T \mathbf{x}_k$
2. $\tilde{\mathbf{r}}_k = L^{-1} \mathbf{r}_k = L^T \mathbf{h}_k$
3. $\tilde{\mathbf{p}}_k = L^T \mathbf{p}_k$.

Bemerkung: Dies besagt, daß das so definierte PCG eine Umformulierung des CG-Verfahrens bezüglich des $\tilde{\cdot}$ -Systems ist, d.h. die hier definierten Größen $\gamma_k, \mathbf{r}_k, \mathbf{x}_k, \beta_k, \mathbf{p}_k$ sind nicht dieselben wie diejenigen, die bei der Anwendung des CG-Verfahrens auf das ursprüngliche System entstehen.

Beweis durch Induktion über k :

$k = 0$:

1. $\mathbf{y}_0 = L^T \mathbf{x}_0$
2. $\tilde{\mathbf{r}}_0 = \tilde{A} \mathbf{y}_0 - \tilde{\mathbf{b}} = L^{-1} A L^{-T} L^T \mathbf{x}_0 - L^{-1} \mathbf{b} = L^{-1} \mathbf{r}_0 = L^{-1} L L^T \mathbf{h}_0 = L^T \mathbf{h}_0$
3. $\tilde{\mathbf{p}}_0 = -\tilde{\mathbf{r}}_0 = -L^T \mathbf{h}_0 = L^T \mathbf{p}_0$

$k \rightarrow k + 1$:

$$\begin{aligned}
 1. \quad \mathbf{y}_{k+1} &= \mathbf{y}_k + \tilde{\gamma}_k \tilde{\mathbf{p}}_k \\
 &= L^T \mathbf{x}_k + \gamma_k L^T \mathbf{p}_k \\
 &= L^T (\mathbf{x}_k + \gamma_k \mathbf{p}_k) \\
 &= L^T \mathbf{x}_{k+1}
 \end{aligned}$$

wegen

$$\tilde{\gamma}_k = \frac{(\tilde{\mathbf{r}}_k, \tilde{\mathbf{r}}_k)}{(\tilde{A} \tilde{\mathbf{p}}_k, \tilde{\mathbf{p}}_k)} = \frac{(L^{-1} \mathbf{r}_k, L^{-1} \mathbf{r}_k)}{(L^{-1} A L^{-T} L^T \mathbf{p}_k, L^T \mathbf{p}_k)} = \frac{(\mathbf{r}_k, C^{-1} \mathbf{r}_k)}{(A \mathbf{p}_k, \mathbf{p}_k)} = \frac{(\mathbf{r}_k, \mathbf{h}_k)}{(A \mathbf{p}_k, \mathbf{p}_k)} = \gamma_k.$$

$$\begin{aligned}
 2. \quad \tilde{\mathbf{r}}_{k+1} &= \tilde{\mathbf{r}}_k + \tilde{\gamma}_k \tilde{A} \tilde{\mathbf{p}}_k \\
 &= L^{-1} \mathbf{r}_k + \gamma_k L^{-1} A L^{-T} L^T \mathbf{p}_k \\
 &= L^{-1} (\mathbf{r}_k + \gamma_k A \mathbf{p}_k) \\
 &= L^{-1} \mathbf{r}_{k+1} = L^T \mathbf{h}_{k+1}
 \end{aligned}$$

$$\begin{aligned}
 3. \quad \tilde{\mathbf{p}}_{k+1} &= -\tilde{\mathbf{r}}_{k+1} + \tilde{\beta}_k \tilde{\mathbf{p}}_k \\
 &= -L^T \mathbf{h}_{k+1} + \beta_k L^T \mathbf{p}_k \\
 &= L^T (-\mathbf{h}_{k+1} + \beta_k \mathbf{p}_k) \\
 &= L^T \mathbf{p}_{k+1}
 \end{aligned}$$

wegen

$$\tilde{\beta}_k = \frac{(\tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{r}}_{k+1})}{(\tilde{\mathbf{r}}_k, \tilde{\mathbf{r}}_k)} = \frac{(\mathbf{r}_{k+1}, \mathbf{h}_{k+1})}{(\mathbf{r}_k, \mathbf{h}_k)} = \beta_k.$$

(vgl. HESTENES 1980) \square

Im Hinblick auf den durch die Vorkonditionierung zusätzlichen Rechenaufwand ist es wesentlich, daß das System (32.4) $C\mathbf{h}_k = \mathbf{r}_k$ leicht zu lösen ist, zumindest wesentlich leichter als das ursprüngliche $A\mathbf{x} = \mathbf{b}$. Dies ist bei einer Zerlegung von C in Dreiecksfaktoren natürlich gegeben. A priori muß L aber keine Dreiecksmatrix sein. Im übrigen ist es auch nicht mehr nötig, $C = LL^T$ zu zerlegen, da der Faktor L im Algorithmus nicht benötigt wird. Jede Zerlegung, die eine leichte Lösbarkeit von (32.4) ermöglicht, ist also anwendbar.

Aus dem vorangegangenen Lemma folgt $\hat{\mathbf{y}} = L^T \hat{\mathbf{x}}$ und

$$\|\mathbf{y}_k - \hat{\mathbf{y}}\|_A^2 = (L^{-1}AL^{-T}L^T(\mathbf{x}_k - \hat{\mathbf{x}}), L^T(\mathbf{x}_k - \hat{\mathbf{x}})) = \|\mathbf{x}_k - \hat{\mathbf{x}}\|_A^2,$$

so daß mit (27) gilt:

Ist k_ε der kleinste Index mit $\|\mathbf{x}_{k_\varepsilon} - \hat{\mathbf{x}}\|_A \leq \varepsilon \|\mathbf{x}_0 - \hat{\mathbf{x}}\|_A$, so ist

$$k_\varepsilon \leq \frac{1}{2} \sqrt{\kappa(\tilde{A})} \ln\left(\frac{2}{\varepsilon}\right) + 1.$$

Bei einer Wahl von C so, daß $\kappa(\tilde{A}) < \kappa(A)$, ist also zu erwarten, daß das PCG schneller konvergiert als das CG-Verfahren. In diesem Sinne optimal wäre die Wahl $C = \alpha A$, $\alpha \in \mathbf{R} \setminus \{0\}$, denn genau dann wäre $\kappa(\tilde{A}) = \kappa(C^{-1}A) = \kappa(\frac{1}{\alpha}\mathbf{I}) = 1$. Diese Wahl ist hier wenig sinnvoll, denn sie ist gleichbedeutend mit der direkten Lösung des Systems $A\mathbf{x} = \mathbf{b}$ im Schritt (32.4). Festzuhalten ist aber, daß $C^{-1}A$ von \mathbf{I} möglichst wenig abweichen sollte. Naheliegend ist es daher, C so zu wählen, daß A durch C in gewissem Sinne angenähert wird und, im Hinblick auf den Speicherplatzbedarf bei großen Systemen, die Faktoren von C nicht wesentlich mehr Platz benötigen als A selbst. Bei den Anwendungen handelt es sich meist um schwach besetzte Systeme, und eine vollständige Zerlegung von A würde unter Umständen ein großes fill-in ergeben.

5 Eine Methode der Vorkonditionierung

Im folgenden sollen Möglichkeiten zur Bestimmung einer Vorkonditionierungsmatrix C dargestellt werden. Dazu betrachtet man, im Unterschied zum Vorherigen,

$$(33) \quad A = M + R,$$

die additive Zerlegung (Splitting) von A , wobei M regulär sei. Damit läßt sich aus (1) ein Iterationsverfahren formulieren

$$M\mathbf{x}_{k+1} = -R\mathbf{x}_k + \mathbf{b}, \quad k \in \mathbf{N}_0,$$

beziehungsweise

$$(34) \quad \mathbf{x}_{k+1} = -M^{-1}R\mathbf{x}_k + M^{-1}\mathbf{b}$$

mit einem beliebigen Startvektor $\mathbf{x}_0 \in \mathbf{R}^n$.

Das Verfahren konvergiert genau dann, wenn

$$\varrho(M^{-1}R) = \max\{|\lambda| \mid \lambda \text{ Eigenwert von } M^{-1}R\} < 1 \text{ ist.}$$

Ein Splitting der Form (33) heißt regulär, wenn M regulär ist, $M^{-1} \geq 0$ und $R \leq 0$ ist.

Satz: Für ein reguläres Splitting konvergiert (34), wenn $A^{-1} \geq 0$ ist. (zum Beweis s. z.B. VARGA)

Aus (34) läßt sich erkennen, daß das Verfahren um so schneller konvergiert, je mehr M A annähert, d.h. je näher R der Nullmatrix ist. Wird als M die Diagonalmatrix gewählt, deren Diagonale gleich derjenigen von A ist, so ist (34) das Jacobi-Verfahren, während für die Wahl M gleich dem unteren Dreieck von A das Gauß-Seidel-Verfahren vorliegt.

Ist nun mit A auch M positiv definit, was z.B. für das Jacobi- bzw. JOR-Verfahren der Fall ist, für das Gauß-Seidel- bzw. SOR-Verfahren mangels Symmetrie aber nicht, so läßt sich M als Vorkonditionierungsmatrix C wählen. Wird M so gewählt, daß (34) konvergiert, d.h. daß die Eigenwerte μ_i von $B := M^{-1}R$ dem Betrag nach kleiner als 1 sind, dann lassen sich die Konvergenzgeschwindigkeiten von (34) und des mit M vorkonditionierten PCG vergleichen.

Für die Eigenwerte $\tilde{\lambda}_i$ von \tilde{A} , das sind die Eigenwerte von $C^{-1}A = M^{-1}A = M^{-1}(M + R) = \mathbf{I} + B$ gilt dann

$$\tilde{\lambda}_i = 1 + \mu_i, \quad i = 1, \dots, n;$$

sind die μ_i geordnet: $-1 < \mu_1 \leq \dots \leq \mu_n < 1$, so gilt $0 < \tilde{\lambda}_1 \leq \dots \leq \tilde{\lambda}_n < 2$, so daß also

$$(35) \quad \kappa(\tilde{A}) = \frac{\tilde{\lambda}_n}{\tilde{\lambda}_1} = \frac{1 + \mu_n}{1 + \mu_1} \leq \frac{1 + \varrho(B)}{1 - \varrho(B)}.$$

Sind die Eigenvektoren \mathbf{v}_i von B linear unabhängig, dann ist $\mathbf{x}_0 - \hat{\mathbf{x}}$ darstellbar als

$$\mathbf{x}_0 - \hat{\mathbf{x}} = \sum_{i=1}^n c_i \mathbf{v}_i.$$

Sei nun $\varepsilon > 0$, $\gamma := \sum_{i=1}^n |c_i| \|\mathbf{v}_i\|_A$ und k_1 und k_2 die jeweilige Mindestanzahl k die benötigt wird, damit $\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A < \varepsilon \gamma$ mit Verfahren (34) bzw. mit dem mit M vorkonditionierten PCG wird. Untere Schranken \hat{k}_1 und \hat{k}_2 für k_1 bzw. k_2 lassen sich folgendermaßen angeben:

Nach (34) gilt: $(\mathbf{x}_k - \hat{\mathbf{x}}) = (-B)^k (\mathbf{x}_0 - \hat{\mathbf{x}}) = (-B)^k \sum_{i=1}^n c_i \mathbf{v}_i = (-1)^k \sum_{i=1}^n c_i \mu_i^k \mathbf{v}_i$,
und

$$\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A = \left\| \sum_{i=1}^n c_i \mu_i^k \mathbf{v}_i \right\|_A \leq \sum_{i=1}^n |c_i| |\mu_i|^k \|\mathbf{v}_i\|_A \leq \varrho(B^k) \gamma.$$

Ist nun $\varrho(B^k) < \varepsilon$, d.h. $k > \ln \varepsilon / \ln \varrho(B)$, da $\varrho(B) < 1$ vorausgesetzt war, dann ist auch $\|\mathbf{x}_k - \hat{\mathbf{x}}\|_A < \varepsilon \gamma$, und man hat

$$\hat{k}_1 := \frac{\ln \varepsilon}{\ln \varrho(B)} = \frac{\ln(1/\varepsilon)}{\ln(1/\varrho(B))}.$$

Nach (28) ist $k_2 > \frac{1}{2} \sqrt{\kappa(\tilde{A})} \ln(2/\varepsilon) =: \hat{k}_2$.

Dann ist

$$\frac{\hat{k}_2}{\hat{k}_1} = \frac{1}{2} \frac{\sqrt{\kappa(\tilde{A})} \ln(2/\varepsilon) \ln(1/\varrho(B))}{\ln(1/\varepsilon)}.$$

Die Hilfsfunktion

$$f(x) = \ln \left(\frac{1}{x} \right) \sqrt{\frac{1+x}{1-x}}, \quad x \in (0, 1)$$

ist monoton fallend und für $x \rightarrow 1$ ist $f(x) \rightarrow 0$, d.h. mit Abschätzung (35) gilt

$$f(\varrho(B)) = \ln \left(\frac{1}{\varrho(B)} \right) \sqrt{\frac{1+\varrho(B)}{1-\varrho(B)}} \geq \ln \left(\frac{1}{\varrho(B)} \right) \sqrt{\kappa(\tilde{A})},$$

also

$$\frac{\hat{k}_2}{\hat{k}_1} \leq \frac{1}{2} f(\varrho(B)) \frac{\ln(2/\varepsilon)}{\ln(1/\varepsilon)}$$

mit $f(\varrho(B)) \rightarrow 0$ für $\varrho(B) \rightarrow 1$.

Dies bedeutet eine erhebliche Konvergenzbeschleunigung beim PCG gegenüber einem allgemeinen, symmetrischen Verfahren der Form (34), wie z.B. dem JOR- oder dem symmetrischen SOR-(SSOR-)-Verfahren, da bei diesen Verfahren bei feinerer Diskretisierung $\varrho(B)$ wächst (vgl. z.B. YOUNG). Dadurch wird die Wahl der Matrix M als Vorkonditionierungsmatrix motiviert, die sich beim SSOR-Verfahren ergibt. Für $\omega \in (0, 2)$ ist dort

$$M = \frac{1}{2 - \omega} \left(\frac{1}{\omega} D + L \right) \left(\frac{1}{\omega} D \right)^{-1} \left(\frac{1}{\omega} D + L \right)^T,$$

wobei D die Diagonale und L das Dreieck unterhalb der Diagonalen von A sind. In AXELSSON wird gezeigt, daß bei optimaler Wahl von ω gilt:

$$\kappa(\tilde{A}) \leq \sqrt{\left(\frac{1}{2} + \delta\right)\kappa(A)} + \frac{1}{2},$$

wobei $\delta := \max_{\mathbf{x} \neq \mathbf{0}} ((LD^{-1}L^T - \frac{1}{4}D)\mathbf{x}, \mathbf{x}) / (A\mathbf{x}, \mathbf{x}) \geq -\frac{1}{4}$.

Dieser Vorkonditionierungsansatz wird hier nicht weiter verfolgt; er hat auch den Nachteil, daß man wieder einen Iterationsparameter, nämlich ω berechnen oder schätzen muß. Im folgenden werde ich mich nur noch mit verschiedenen Vorkonditionierungsansätzen durch unvollständige Zerlegung beschäftigen.

6 Vorkonditionierung durch unvollständige Zerlegung

6.1 Vorbemerkungen

Für eine positiv definite Matrix A existiert immer die Zerlegung nach dem Gaußschen Algorithmus $A = LU$ (bzw. die Cholesky-Zerlegung $A = LL^T$) mit diagonalen Pivotwahl. Führt man jedoch nicht diese vollständige Zerlegung durch, sondern ändert man bei den einzelnen Eliminationsschritten die jeweiligen Faktoren $L^{(k)}, U^{(k)}$ ab, so erhält man, vorausgesetzt dieses Verfahren bricht nicht infolge Singulärwerdens des noch zu zerlegenden Teils ab, eine Restmatrix R , so daß gilt

$$(36) \quad A = LU + R,$$

oder umgekehrt interpretiert, LU ist die Faktorisierung von $A - R$.

Um eine Übereinstimmung von (36) und (33) zu haben, d.h. LU als Vorkonditionierungsmatrix C zu benutzen, ist diese unvollständige Zerlegung so durchzuführen, daß LU positiv definit bleibt, also insbesondere R symmetrisch ist.

Daraus ergeben sich zwei zusammenhängende Probleme: für welche Matrizenklasse und bei welcher Manipulation an der Zerlegung existiert ein Splitting (36) mit den angegebenen Eigenschaften?

Für die folgende formelmäßige Darstellung der unvollständigen Zerlegung sei der Form halber vorausgesetzt, daß eine solche Zerlegung existiert, damit man sinnvoll L, U, L^{-1}, U^{-1} usw. schreiben kann.

Bei einer vollständigen Zerlegung tritt bei den Faktoren L und U fill-in auf, d.h. L und U sind in der Summe im allgemeinen deutlich weniger *sparse* als A . Damit der Vorteil der Vorkonditionierung nicht durch den größeren Rechenaufwand durch das fill-in bei L und U wieder beseitigt wird, wird meist gefordert, daß in L und U dort null steht, wo auch A einen Nulleintrag hat. Der 'normale' Gauß-Algorithmus hat folgende Form:

$$(37.1) \quad A^{(1)} = A$$

$$(37.2) \quad A^{(k+1)} = L^{(k)} A^{(k)}, \quad k = 1, 2, \dots, n-1,$$

wobei $L^{(k)}$ untere Dreiecksmatrix ist mit

$$(37.3) \quad \begin{aligned} l_{ii}^{(k)} &= 1, & i &= 1, \dots, n \\ l_{ik}^{(k)} &= -a_{ik}^{(k)} / a_{kk}^{(k)}, & i &= k+1, \dots, n \\ l_{ij}^{(k)} &= 0 \text{ sonst.} \end{aligned}$$

Rekursiv ergibt dies die obere Dreiecksmatrix

$$(38) \quad U := A^{(n)} = L^{(n-1)} \dots L^{(1)} A.$$

Die Matrix

$$(39) \quad (L^{(n-1)} \dots L^{(1)})^{-1} =: L$$

ist dann eine untere Dreiecksmatrix mit 1-Diagonale. Sie hat unterhalb der Diagonalen die Einträge aus (37.3) mit umgekehrtem Vorzeichen. Die gebräuchliche unvollständige Zerlegung hat demgegenüber die Form

$$(40.1) \quad A^{(1)} = A$$

$$(40.2) \quad A^{(k+1)} = L^{(k)} A^{(k)} - R^{(k)}, \quad k = 1, \dots, n-1.$$

Dabei eliminiert $L^{(k)}$ wieder die k -te Spalte von $A^{(k)}$ unterhalb der Diagonalen, jedoch erhält $A^{(k+1)}$ nur an vorgegebenen Stellen Nichtnulleinträge; die dabei nicht berücksichtigten Werte, werden auf die entsprechenden Positionen in $R^{(k)}$ geschrieben. Damit wird

$$\begin{aligned} (L^{(n-1)} \dots L^{(1)})A &= A^{(n)} + (L^{(n-1)} \dots L^{(2)})R^{(1)} + \\ &\quad + (L^{(n-1)} \dots L^{(3)})R^{(2)} + \dots + L^{(n-1)}R^{(n-2)} + \\ &\quad + R^{(n-1)}. \end{aligned}$$

Mit U und L definiert formal wie in (38) bzw. (39) ist dann

$$\begin{aligned} (41) \quad A &= LU + (L^{(1)})^{-1}R^{(1)} + (L^{(2)}L^{(1)})^{-1}R^{(2)} + \\ &\quad + \dots + (L^{(n-1)} \dots L^{(1)})^{-1}R^{(n-1)} \\ &= LU + R. \end{aligned}$$

Vereinfachen läßt sich (41), indem Manipulationen an der exakten Zerlegung nur unterhalb der jeweiligen Pivotzeile zugelassen werden, d.h. $r_{ij}^{(k)} = 0$ für $i \leq k$. In diesem Fall gilt für $m \leq k$ $L^{(m)}R^{(k)} = R^{(k)}$ und in der Multiplikation $(L^{(k)} \dots L^{(1)})^{-1}R^{(k)}$ sind dann die $L^{(i)}$ neutral, daher

$$(42) \quad R = R^{(1)} + \dots + R^{(n-1)}.$$

Im weiteren wird es um die Wahl von R bzw. $R^{(i)}$ gehen.

6.2 PCG mit unvollständiger LU-Zerlegung für symmetrische M-Matrizen

Bei der Diskretisierung von Differentialgleichungen entstehen oft sogenannte M-Matrizen, so daß die Fragestellung nahe liegt, unter welchen Umständen für solche Matrizen eine unvollständige LU-Zerlegung existiert. Im folgenden wird daher der

Ansatz von MEIJERINK/VAN DER VORST, die die Durchführbarkeit fast jeder beliebigen unvollständigen Zerlegung für M-Matrizen gezeigt haben, dargestellt.

Definition: $A \in \mathbf{R}^{n,n}$ heißt *M-Matrix*, wenn A regulär, $a_{ij} \leq 0$ für $i \neq j$ und $A^{-1} \geq 0$ ist.

Die folgenden Lemmata geben Eigenschaften von M-Matrizen an.

(43) **Lemma:** Ist A eine M-Matrix, dann ist $a_{ii} > 0$ für $i = 1, \dots, n$.

Beweis: Sei A M-Matrix, $\mathbf{a}_i \neq \mathbf{0}$ die i -te Spalte von A und $a_{ii} \leq 0$, dann ist $A\mathbf{e}_i = \mathbf{a}_i \leq \mathbf{0}$ und, da $A^{-1} \geq 0$ ist, $\mathbf{0} \leq \mathbf{e}_i = A^{-1}\mathbf{a}_i \leq \mathbf{0}$, was ein Widerspruch ist, da $\mathbf{e}_i \neq \mathbf{0}$ ist. \square

(44) **Lemma:** Ist A symmetrische M-Matrix, dann ist A positiv definit.

Beweis: Als Hilfsmittel sei der **Satz** von Gerschgorin zitiert:

Die Eigenwerte λ_i einer komplexen Matrix $H = (h_{ij})$ liegen innerhalb oder auf dem Rand des Gebietes G der komplexen λ -Ebene, gebildet aus den n Kreisen K_i mit den Mittelpunkten h_{ii} und den Radien $\varrho = \sum_{k \neq i} |h_{ik}|$, also den Zeilensummen der Beträge der Nichtdiagonalelemente. Gleiches gilt für die Kreise K'_i der analogen Spaltensummen als Radien. Die Eigenwerte liegen also im Durchschnitt beider Gebiete (vgl. ZURMÜHL).

Für $\mathbf{e} := (1, \dots, 1)^T$ und jede Matrix $A \in \mathbf{R}^{n,n}$ sind die Komponenten des Vektors $A\mathbf{e}$ die Zeilensummen von A . Ist A M-Matrix, dann ist $\mathbf{x} := A^{-1}\mathbf{e} > 0$, da $A^{-1} \geq 0$.

Mit $X := \text{diag}(x_1, \dots, x_n)$ wird $X\mathbf{e} = \mathbf{x}$ und $AX\mathbf{e} = A\mathbf{x} = \mathbf{e}$, d.h. die Zeilensummen von AX sind positiv. Damit ist für alle $i \in \{1, \dots, n\}$

$$a_{ii}x_i > - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j$$

bzw.

$$a_{ii} > - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} \frac{x_j}{x_i} = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij} \frac{x_j}{x_i}|.$$

Letzteres besagt gerade, daß die Matrix $X^{-1}AX$, welche ähnlich zu A ist, strikt diagonal-dominant ist und daß keiner der Kreise nach dem Satz von Gerschgorin die Null enthält. Da A symmetrisch ist, sind alle Eigenwerte von A bzw. $X^{-1}AX$ reell und daher positiv, A also positiv definit. \square

Die Eigenschaft, M-Matrix zu sein, bleibt bei der Gauß-Zerlegung erhalten:

(45) **Satz:** Mit A ist auch $A^{(2)}$ eine M-Matrix, wobei $A^{(2)}$ die durch (37.2) definierte Matrix ist.

Beweis: Sei A M-Matrix und $L^{(1)}$ nach (37.3) eliminiere die erste Spalte von A . Mit $L^{(1)}$ und A ist auch $L^{(1)}A$ regulär und $a_{ij}^{(2)} = a_{ij} - a_{i1}a_{1j}/a_{11} \leq a_{ij}$, für $i, j > 1$, da

$a_{i1}, a_{1j} \leq 0, a_{11} > 0$. Daher ist $a_{ij}^{(2)} \leq a_{ij} \leq 0$ für $i \neq j$ und es bleibt $(L^{(1)}A)^{-1} \geq 0$ zu zeigen, was äquivalent ist zu $(L^{(1)}A)^{-1}\mathbf{e}_j \geq 0$ für alle $j = 1, \dots, n$. Es ist $(L^{(1)}A)\mathbf{e}_1 = a_{11}\mathbf{e}_1$, also $(L^{(1)}A)^{-1}\mathbf{e}_1 = \frac{1}{a_{11}}\mathbf{e}_1 \geq 0$. Für $j = 2, \dots, n$ ist $(L^{(1)}A)^{-1}\mathbf{e}_j = A^{-1}L^{(1)-1}\mathbf{e}_j = A^{-1}\mathbf{e}_j \geq 0$. \square

Folgerung: $U = A^{(n)}$ ist eine M-Matrix, wenn A eine M-Matrix ist.

(46) **Satz:** Sei A eine M-Matrix und $B \in \mathbf{R}^{n,n}$ eine Matrix, für die gilt

- a) $a_{ij} \leq b_{ij} \leq 0$ für $i \neq j$ und
- b) $0 < a_{ii} \leq b_{ii}$,

dann ist auch B eine M-Matrix.

Beweis: MEIJERINK/VAN DER VORST nach VARGA.

Damit ist es möglich, die Existenz einer gewünschten Zerlegung nach (36) für beliebige M-Matrizen zu zeigen. Die Matrix R wird dabei durch die Menge J definiert, die alle Indexpaare enthält, für die Nichtnulleinträge bei der Gauß-Zerlegung zugelassen wird. Im folgenden wird vorausgesetzt, daß die Diagonale in J enthalten ist, d.h.

$$(47) \quad \{(i, i) | i = 1, \dots, n\} \subset J.$$

Oft enthält J darüber hinaus noch alle Indexpaare, für die A Nichtnulleinträge enthält.

Satz: Zu jeder M-Matrix A und jeder Menge J von Indexpaaren nach (47) existiert eine untere Dreiecksmatrix L mit 1-Diagonale, eine obere Dreiecksmatrix U und eine Matrix R mit

$$\begin{aligned} l_{ij} &= 0 && \text{falls } (i, j) \notin J, \\ u_{ij} &= 0 && \text{falls } (i, j) \notin J, \\ r_{ij} &= 0 && \text{falls } (i, j) \in J, \end{aligned}$$

so daß das Splitting $A = LU + R$ regulär ist. L und U sind eindeutig.

Zum *Beweis* wird in Algorithmus (40) ein Zwischenschritt eingefügt:

$$(48) \quad \begin{aligned} A^{(1)} &= A \\ \tilde{A}^{(k)} &= A^{(k)} - R^{(k)}, \quad k = 1, \dots, n-1 \\ A^{(k+1)} &= L^{(k)}\tilde{A}^{(k)}. \end{aligned}$$

Dies ist so zu verstehen, daß im k -ten Schritt, zuerst diejenigen Elemente aus der k -ten Zeile und k -ten Spalte von $A^{(k)}$ in $R^{(k)}$ getan werden, welche im vorherigen

Eliminationsschritt als nicht zugelassenes fill-in auftreten und dann mit der so modifizierten Matrix ein üblicher Eliminationsschritt durchgeführt wird. $R^{(k)}$ hat hier also eine spezielle Gestalt und ist definiert durch

$$\begin{aligned} r_{kj}^{(k)} &= a_{kj}^{(k)} \text{ falls } (k, j) \notin J, \\ r_{ik}^{(k)} &= a_{ik}^{(k)} \text{ falls } (i, k) \notin J, \\ r_{ij}^{(k)} &= 0 \text{ sonst.} \end{aligned}$$

$L^{(k)}$ ist dann wie üblich definiert, d.h. die k -te Spalte lautet

$$(49) \quad \left(0, \dots, 0, 1, -\frac{\tilde{a}_{k+1,k}^{(k)}}{\tilde{a}_{kk}^{(k)}}, \dots, -\frac{\tilde{a}_{nk}^{(k)}}{\tilde{a}_{kk}^{(k)}} \right)^T$$

Da $A^{(1)} = A$ nach Voraussetzung eine M-Matrix ist, d.h. $a_{ij} \leq 0$ für $i \neq j$, ist wegen (47) dann $R^{(1)} \leq 0$. Nach (46) ist $\tilde{A}^{(1)} = A^{(1)} - R^{(1)}$ eine M-Matrix, weil die Elemente außerhalb der Diagonalen höchstens null sind, im übrigen $\tilde{A}^{(1)} \geq A^{(1)}$. Damit ist nach (45) auch $A^{(2)}$ eine M-Matrix. Nach (49) ist $L^{(1)} \geq 0$. Der Beweis für $k > 1$ verläuft identisch, so daß bisher gezeigt ist:

$$(50) \quad A^{(k)}, \tilde{A}^{(k)} \text{ sind M-Matrizen,} \\ L^{(k)} \geq 0, R^{(k)} \leq 0 \text{ für } k = 1, \dots, n-1.$$

Die $R^{(k)}$ aus (40.2) wurden in (48) gleich $L^{(k)}R^{(k)}$ gesetzt und die k -te Zeile dieser $R^{(k)}$ ist im allgemeinen nicht null. Mit diesen $R^{(k)}$ lauten die Summanden nach (41) dann

$$(L^{(k-1)} \dots L^{(1)})^{-1} R^{(k)}$$

so daß auch hier (42) gilt:

$$R = R^{(1)} + \dots + R^{(n-1)} \text{ und } R \leq 0.$$

Nach (50) ist $L^{-1} = (L^{(n-1)} \dots L^{(1)}) \geq 0$ und $U^{-1} = (A^{(n)})^{-1} \geq 0$ nach (45), also $(LU)^{-1} = U^{-1}L^{-1} \geq 0$.

Damit ist gezeigt, daß $A = LU + R$ ein reguläres Splitting ist. Die Eindeutigkeit ergibt sich aus der Eindeutigkeit der üblichen Gauß-Zerlegung. \square

Folgerung: Ist A eine symmetrische M-Matrix und wird J symmetrisch gewählt (d.h. $(i, j) \in J \Rightarrow (j, i) \in J$), dann ist LU positiv definit, d.h. $C = LU$ kann als Vorkonditionierungsmatrix genommen werden.

Beweis: Oben wurde $(LU)^{-1} \geq 0$ gezeigt. Außerdem ist außerhalb der Diagonalen LU nicht positiv, da R nur ein Nullsetzen von LU -Einträgen bewirkt. LU ist also eine symmetrische M-Matrix und damit nach Lemma (44) positiv definit. \square

Ein Problem bei der unvollständigen Zerlegung ist, daß die faktorisierte Matrix instabil in dem Sinne werden kann, daß die Pivotelemente klein oder negativ werden können, auch wenn A positiv definit ist. Für M-Matrizen haben MEIJERINK/VAN DER VORST mit den folgenden Sätzen gezeigt, daß die Faktorisierung immer relativ stabil ist.

(51) **Satz:** Wie in (46) sei A eine M-Matrix, $B \in \mathbf{R}^{n,n}$ eine Matrix, für die gilt

- a) $a_{ij} \leq b_{ij} \leq 0$ für $i \neq j$ und
- b) $0 < a_{ii} \leq b_{ii}$,

$A^{(2)}$ und $B^{(2)}$ seien die Matrizen, die aus $A = A^{(1)}$ bzw. $B = B^{(1)}$ durch den ersten Schritt der üblichen Gauß-Elimination entstehen, dann gilt

- 1) $a_{ij}^{(2)} \leq b_{ij}^{(2)} \leq 0$ für $i \neq j$,
 $0 < a_{ii}^{(2)} \leq b_{ii}^{(2)}$ und
- 2) $B^{(2)}$ ist eine M-Matrix.

Beweis: Nach a) ist $a_{i1}a_{1j}$ und $b_{i1}b_{1j}$ nicht negativ für $i, j > 1$ und $a_{i1}a_{1j} \geq b_{i1}b_{1j}$. Nach b) ist $a_{i1}a_{1j}/a_{11} \geq b_{i1}b_{1j}/b_{11} \geq 0$, so daß gilt

$$a_{ij}^{(2)} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}} \leq b_{ij} - \frac{b_{i1}b_{1j}}{b_{11}} = b_{ij}^{(2)} \text{ für } i, j > 1 \text{ und } b_{ij}^{(2)} \leq 0.$$

Nach (45) ist $A^{(2)}$ eine M-Matrix, d.h. $a_{ii}^{(2)} > 0$, also auch $b_{ii}^{(2)} \geq a_{ii}^{(2)} > 0$. Damit ist nach (46) auch $B^{(2)}$ eine M-Matrix. \square

Satz: Für eine M-Matrix ist die unvollständige Zerlegung (48) mindestens so stabil wie die vollständige Zerlegung ohne Pivotsuche.

Beweis: Zu zeigen ist, daß $\tilde{L}^{(k)}$ der unvollständigen Zerlegung dem Betrag nach nicht größer ist als $L^{(k)}$ der vollständigen Zerlegung, d.h.

$$\left| \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \geq \left| \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}} \right|,$$

wobei $a_{ik}^{(k)}$, $\tilde{a}_{ik}^{(k)}$ die Einträge von $A^{(k)}$ bzw. $\tilde{A}^{(k)}$ der vollständigen bzw. unvollständigen Zerlegung sind.

Sei also A eine M-Matrix, $\tilde{A}^{(1)}$ die Matrix nach (48), d.h. in der ersten Zeile oder ersten Spalte von $\tilde{A}^{(1)}$ sind möglicherweise Einträge null gesetzt, aber $\tilde{a}_{ii}^{(1)} \neq 0$; ansonsten sind $A = A^{(1)}$ und $\tilde{A}^{(1)}$ gleich. $L^{(1)}$ und $\tilde{L}^{(1)}$ seien die jeweiligen Eliminationsmatrizen. Da $a_{11} = \tilde{a}_{11}^{(1)} > 0$, sind $L^{(1)}$ und $\tilde{L}^{(1)}$ gleich bis auf höchstens die

Stellen in der ersten Spalte, wo $\tilde{a}_{i1}^{(1)} = 0$ gesetzt wurde; dort ist $\tilde{l}_{i1}^{(1)} = 0$. $L^{(1)}$ und $\tilde{L}^{(1)}$ haben außerhalb der Diagonalen nur nichtpositive Einträge, weil A M-Matrix ist. Somit gilt

$$|L^{(1)}| \geq |\tilde{L}^{(1)}|.$$

Nach (45) ist $A^{(2)} = L^{(1)}A^{(1)}$ M-Matrix, nach (51) ist $\tilde{A}^{(2)} = \tilde{L}^{(1)}\tilde{A}^{(1)}$ M-Matrix und es gilt

$$(52) \quad A^{(2)} \leq \tilde{A}^{(2)}.$$

Daher gilt dann auch $|A^{(2)} - \text{diag}(A^{(2)})| \geq |\tilde{A}^{(2)} - \text{diag}(\tilde{A}^{(2)})|$.

Was passiert längs der Diagonalen?

Für $A^{(2)}$ werden die Diagonalelemente nicht größer, für $\tilde{A}^{(2)}$ auch nicht, aber bei $\tilde{A}^{(2)}$ kann ein Diagonalelement seinen Wert behalten, wo es bei $A^{(2)}$ vermindert wird, nämlich dort, wo für $\tilde{A}^{(2)}$ ein Zeilen- oder Spaltenelement null gesetzt war. Die Diagonalelemente bleiben positiv nach (45), da $A^{(2)}$ M-Matrix. Also hat man

$$(53) \quad |A| = |A^{(1)}| \geq |\tilde{A}^{(1)}| \geq |\tilde{A}^{(2)}|.$$

Nach (51) und (52) gilt für den k -ten Schritt:

$$A^{(k)} \leq \tilde{A}^{(k)}, \text{ d.h. } a_{ij}^{(k)} \leq \tilde{a}_{ij}^{(k)} \leq 0 \text{ für } i \neq j, 0 < a_{ii}^{(k)} \leq \tilde{a}_{ii}^{(k)} \text{ und damit}$$

$$\frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \leq \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}} \leq 0 \text{ für } i > k,$$

also $L^{(k)} \leq \tilde{L}^{(k)}$ bzw. $|L^{(k)}| \geq |\tilde{L}^{(k)}|$ und nach (53) $|A| \geq |\tilde{A}^{(k)}|$. \square (vgl. (58))

Da die vollständige LU -Zerlegung für diagonaldominante M-Matrizen stabil ist, hat man für diesen praktisch auftretenden Fall auch die Stabilität der unvollständigen Zerlegung. Über die Konvergenzverbesserung des CG-Verfahrens im Sinne von (29) läßt sich hier keine Aussage machen. Die Modellrechnungen zeigen, daß je nach Wahl der Indexmenge J auch Konvergenzverschlechterung eintreten kann. Bei dem in dem bisher Bewiesenen ist in J mindestens nur die Diagonale enthalten. Bei den von MEIJERINK/VAN DER VORST dargestellten Beispielrechnungen sind in J aber mindestens alle Indexpaare (i, j) enthalten, für die $a_{ij} \neq 0$ ist.

6.3 Unvollständige LU -Zerlegung nach der Methode von KERSHAW

KERSHAW läßt die Voraussetzung, daß A M-Matrix sei, fallen; es wird vorausgesetzt, daß A positiv definit ist. Statt der LU -Zerlegung wie bei Meijerink/van der Vorst wird hier aber eine Cholesky-Zerlegung $A = L_C D L_C^T$, L_C untere Dreiecksmatrix, D Diagonalmatrix, vorgenommen. Aus der Gauß-Zerlegung $A = LU$ erhält man die Zerlegung $A = LU = L_C D L_C^T$, indem

$$D = \text{diag}(a_{11}^{(1)}, \dots, a_{nn}^{(n)}) = \text{diag}(u_{11}, \dots, u_{nn})$$

gesetzt wird. $L_C D^{\frac{1}{2}} =: \tilde{L}$ ergibt dann die übliche Form $A = \tilde{L} \tilde{L}^T$.

Hier wird jedoch die Zerlegung direkt aus der Bestimmungsgleichung $A = L_C D L_C^T$ bestimmt durch

$$(54) \quad l_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{jk} l_{ik} d_{kk}, \quad j = i, \dots, n$$

$$d_{ii} = 1/l_{ii}$$

Für die unvollständige Zerlegung wird die Indexmenge J so bestimmt, daß alle Indexpaare (i, j) in J sind, für die $a_{ij} \neq 0$ ist, d.h. L_C und D haben höchstens dort von null verschiedene Einträge, wo auch die Einträge von A ungleich null sind. In (54) wird dann $l_{ij} = 0$ gesetzt, falls $(i, j) \notin J$. Da A positiv definit ist, ist $(i, i) \in J$ für $i = 1, \dots, n$, wie schon früher bemerkt. Man erhält also eine unvollständige Zerlegung

$$A = L_C D L_C^T + R.$$

Während die vollständige Cholesky-Zerlegung für positiv definite Matrizen immer durchführbar ist, d.h. $l_{ii} > 0$ für $i = 1, \dots, n$, können die Fälle $l_{ii} = 0$ oder $l_{ii} \leq 0$ bei der unvollständigen Zerlegung durchaus auftreten (falls A M-Matrix ist, bleibt $l_{ii} > 0$). Im ersten Fall ist die Zerlegung nicht durchführbar, im zweiten Fall ist $A - R = L_C D L_C^T$ nicht mehr positiv definit, d.h. $L_C D L_C^T$ kann nicht als Vorkonditionierungsmatrix genommen werden.

Um diese Kalamitäten zu vermeiden, schlägt KERSHAW vor, falls in (54) $l_{ii} \leq 0$ ist für ein i , dann $l_{ii} > 0$ zu setzen, womit $r_{ii} \neq 0$ wird, und damit weiterzurechnen (KERSHAW setzt l_{ii} jedoch nicht beliebig, sondern $l_{ii} := \sum_{j=1, j \neq i}^n |l_{ij}|$). Da dies ein ad hoc - Verfahren ist, sind auch keine allgemeinen Aussagen möglich. Vielmehr ist durch praktische Versuche am jeweiligen Problem zu entscheiden, was noch geht und was nicht mehr.

6.4 PCG mit modifizierter unvollständiger Zerlegung

Der Ausgangspunkt ist bei dem Ansatz von AXELSSON/BARKER derselbe wie bei MEIJERINK/VAN DER VORST; die Menge J der Indices, für die Nicht-nulleinträge zugelassen werden, enthält mindestens die Diagonale, darüber hinaus ist J symmetrisch. Die Gauß-Elimination wird jedoch folgendermaßen modifiziert: ist $(i, j) \notin J$ im k -ten Eliminationsschritt ($i, j \geq k$), dann wird $a_{ij}^{(k+1)} = 0$ gesetzt und der Wert für $a_{ij}^{(k+1)}$, der sich durch die Elimination eigentlich ergäbe, wird zum Diagonalelement der i -ten Zeile addiert.

Damit läßt sich der Zerlegungsalgorithmus folgendermaßen beschreiben:

$$A^{(1)} = A,$$

für $k = 1, \dots, n-1$ und für $i = k+1, \dots, n$:

$$l_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)} \quad (55.1)$$

$$a_{ij}^{(k+1)} = \begin{cases} a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)} & \text{für } k+1 \leq j \leq n, (i, j) \in J, j \neq i & (55.2) \\ 0 & \text{für } k+1 \leq j \leq n, (i, j) \notin J & (55.3) \\ a_{ii}^{(k)} - l_{ik} a_{ki}^{(k)} + & \\ + \sum_{\substack{m=k+1 \\ (i,m) \notin J}}^n (a_{im}^{(k)} - l_{ik} a_{km}^{(k)}) & \text{für } j = i & (55.4) \\ 0 & \text{für } j = k & (55.5) \end{cases}$$

Ansonsten ist $a_{ij}^{(k+1)} = a_{ij}^{(k)}$ (55.6).

Im folgenden wird, wie es üblicherweise getan wird, vorausgesetzt, daß J alle Indexpaare enthält, für die die entsprechenden Einträge von A nicht null sind, d.h.

$$(56) \quad (i, j) \notin J \Rightarrow a_{ij} = 0.$$

Damit vereinfacht sich (55.4) wegen $a_{ij}^{(k)} = 0$ für $(i, j) \notin J$ zu

$$a_{ii}^{(k+1)} = a_{ii}^{(k)} - l_{ik} a_{ki}^{(k)} - \sum_{\substack{m=k+1 \\ (i,m) \notin J}}^n l_{ik} a_{km}^{(k)}$$

für alle $k = 1, \dots, n-1$.

Definiert man neben (55.1) die restlichen Einträge von L durch 1 auf der Diagonalen, 0 im oberen Dreieck und $U := A^{(n)}$, dann sind L und U in der üblichen Weise gegeben.

Für eine Klasse von Matrizen, die mit derjenigen der M-Matrizen verwandt ist, wird nun die Stabilität einer solchen unvollständigen Zerlegung nachgewiesen.

Definition: $A \in \mathbf{R}^{n,n}$ heißt schwach diagonaldominant, wenn für $i = 1, \dots, n$

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|.$$

Ist $a_{ii} \neq 0$ für $i = 1, \dots, n-1$, dann wird für diese i definiert

$$m(i) := \max\{j | i \leq j \leq n, a_{ij} \neq 0\}.$$

$m(i) \geq i$ und $a_{i,m(i)}$ ist der letzte Nichtnulleintrag in der i -ten Zeile.

(57) $A \in \mathbf{R}^{n,n}$ heißt \hat{M} -Matrix, wenn

a) $a_{ii} > 0$ für $i = 1, \dots, n-1$,

$$a_{nn} \geq 0,$$

b) $a_{ij} \leq 0$ für $i \neq j$,

c) $m(i) > i$ für $i = 1, \dots, n-1$.

$s_i^{(k)} := \sum_{j=1}^n a_{ij}^{(k)}$ ist die Summe der Einträge in der i -ten Zeile beim k -ten Eliminationsschritt, kurz *Zeilensumme*.

Bemerkungen: Wegen (43) ist eine M-Matrix, für die (57) c) gilt, eine \hat{M} -Matrix. Für das Splitting $A = LU + R$ bedeutet (55), daß die Zeilensummen von R null sind. Eine \hat{M} -Matrix ist genau dann schwach diagonaldominant, wenn alle Zeilensummen nicht negativ sind.

Die Eigenschaft 'schwach diagonaldominante \hat{M} -Matrix' bleibt bei der Gauß-Zerlegung erhalten:

(58) **Satz:** Es sei $A \in \mathbf{R}^{n,n}$ eine schwach diagonaldominante \hat{M} -Matrix, dann gilt: $A^{(2)}, A^{(3)}, \dots, A^{(n)}$ nach (55) sind ebenfalls schwach diagonaldominante \hat{M} -Matrizen, und

a) $a_{ij}^{(k+1)} \leq a_{ij}^{(k)} \leq 0$ für $i, j = k+1, \dots, n, i \neq j$,

b) $s_i^{(k+1)} \geq s_i^{(k)} \geq 0$ für $i = k+1, \dots, n$,

c) $0 < a_{ii}^{(k+1)} \leq a_{ii}^{(k)}$ für $i = k+1, \dots, n-1$,
 $0 \leq a_{nn}^{(k+1)} \leq a_{nn}^{(k)}$.

Beweis durch Induktion über k :

Es wird zuerst der Induktionsschritt $k \rightarrow k+1, k < n$ durchgeführt. Sei also $A^{(k)}$ eine schwach diagonaldominante \hat{M} -Matrix. In den ersten k Zeilen stimmen $A^{(k)}$ und $A^{(k+1)}$ überein, $A^{(k+1)}$ hat Nullen in den ersten k Spalten der letzten $n-k$ Zeilen. Daher ist $A^{(k+1)}$ genau dann eine schwach diagonaldominante \hat{M} -Matrix, wenn die Untermatrix aus den letzten $n-k$ Zeilen und Spalten von $A^{(k+1)}$ eine schwach diagonaldominante \hat{M} -Matrix ist. Letzteres wird nun gezeigt.

Zu a):

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{ik}^{(k)} a_{kj}^{(k)}}{a_{kk}^{(k)}}, \quad i, j = k+1, \dots, n, i \neq j,$$

nach (57) ist $a_{kk}^{(k)} > 0, a_{ik}^{(k)}, a_{kj}^{(k)} \leq 0$, also $a_{ik}^{(k)} a_{kj}^{(k)} / a_{kk}^{(k)} \geq 0$ und $a_{ij}^{(k+1)} \leq a_{ij}^{(k)}$. Ist $(i, j) \notin J$, dann ist $a_{ij}^{(k+1)} = a_{ij}^{(k)} = 0$, insgesamt also $a_{ij}^{(k+1)} \leq 0$ für $i \neq j$.

Zu b):

$$s_i^{(k+1)} = \sum_{j=k+1}^n a_{ij}^{(k+1)} \quad \text{für } i = k+1, \dots, n$$

$$\begin{aligned}
&= \sum_{j=k+1}^n \left(a_{ij}^{(k)} - \frac{a_{ik}^{(k)} a_{kj}^{(k)}}{a_{kk}^{(k)}} \right) \\
&= \sum_{j=k+1}^n a_{ij}^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \sum_{j=k+1}^n a_{kj}^{(k)} \\
&= \left(s_i^{(k)} - a_{ik}^{(k)} \right) - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \left(s_k^{(k)} - a_{kk}^{(k)} \right) \\
&= s_i^{(k)} - \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} s_k^{(k)}.
\end{aligned}$$

Nach (57) ist $a_{ik}^{(k)}/a_{kk}^{(k)} \leq 0$ und $s_i^{(k)}, s_k^{(k)} \geq 0$, da A schwach diagonaldominante \hat{M} -Matrix ist, also $s_i^{(k+1)} \geq s_i^{(k)} \geq 0$ und insgesamt $s_i^{(k+1)} \geq 0$ für $i = 1, \dots, n$.

Zu c):

$$a_{ii}^{(k+1)} = a_{ii}^{(k)} - \frac{a_{ik}^{(k)} a_{ki}^{(k)}}{a_{kk}^{(k)}} - \sum_{\substack{m=k+1 \\ (i,m) \notin J}}^n \frac{a_{ik}^{(k)} a_{km}^{(k)}}{a_{kk}^{(k)}} \text{ für } i = k+1, \dots, n.$$

Nach den gleichen Überlegungen wie bei a) gilt dann $a_{ii}^{(k+1)} \leq a_{ii}^{(k)}$. Nach a) ist

$$0 \leq s_i^{(k+1)} = a_{ii}^{(k+1)} - \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{ij}^{(k+1)}|,$$

also

$$\sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{ij}^{(k+1)}| \leq a_{ii}^{(k+1)},$$

d.h. $a_{ii}^{(k+1)} \geq 0$ für $i = k+1, \dots, n$ und $A^{(k+1)}$ ist schwach diagonaldominant. Damit folgt weiter

$$a_{ii}^{(k+1)} \geq |a_{i,m(i)}^{(k+1)}| \geq |a_{i,m(i)}^{(k)}| > 0 \text{ für } i = k+1, \dots, n-1,$$

$(m(i))$ ist der Spaltenindex des letzten Nichtnulleintrags in der i -ten Zeile der Ausgangsmatrix A , insgesamt also

$$\begin{aligned}
a_{ii}^{(k+1)} &> 0 \text{ für } i = 1, \dots, n-1 \\
a_{nn}^{(k+1)} &\geq 0 \\
m^{(k+1)}(i) &> i \text{ für } i = 1, \dots, n-1
\end{aligned}$$

Induktionsanfang:

$A^{(1)} = A$ ist nach Voraussetzung schwach diagonaldominante \hat{M} -Matrix. Wegen der Voraussetzung (56) an J gelten alle vorangehenden Beweisschritte auch für $k = 1$.
□

Damit läßt sich nun einfach die Stabilität der unvollständigen Zerlegung zeigen:

(59) **Satz:** Ist $A \in \mathbf{R}^{n,n}$ eine schwach diagonaldominante \hat{M} -Matrix, dann ist die Zerlegung (55) stabil in dem Sinne, daß

$$\max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}|$$

beschränkt ist.

Beweis: Da A schwach diagonaldominante \hat{M} -Matrix ist, gilt

$$a_{ii} \geq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \geq |a_{ij}| \text{ für } j = 1, \dots, n, j \neq i,$$

d.h. das betragsgrößte Element je Zeile ist das Diagonalelement. Also ist $\max_{i,j} |a_{ij}| = a_{mm}$ für ein $m \in \{1, \dots, n\}$ und nach (58) c) ist dann auch $\max_{i,j,k} |a_{ij}^{(k)}| = a_{mm}$, der Quotient von beiden also 1. □

Bemerkung: Da $a_{nn} = 0$ zugelassen ist, bedeutet (59) natürlich nicht, daß die Faktorisierung LU regulär ist.

Darstellung der Matrix R

Der Algorithmus (55) hat in Matrixschreibweise die Form (40), und daher ist $R = \sum_{k=1}^{n-1} R^{(k)}$. Für $R^{(k)}$ gilt

$$r_{ij}^{(k)} = 0 \text{ für } i \notin \{k+1, \dots, n\} \text{ oder } j \notin \{k+1, \dots, n\},$$

da $L^{(k)}A^{(k)}$ und $A^{(k+1)}$ dort übereinstimmen. Weiterhin gilt nach (55) für $i, j \in \{k+1, \dots, n\}$

$$(60) \quad r_{ij}^{(k)} = \begin{cases} 0 & \text{für } (i, j) \in J, i \neq j \\ -l_{ik}a_{kj}^{(k)} & \text{für } (i, j) \notin J \\ \sum_{\substack{m=k+1 \\ (i,m) \notin J}}^n l_{ik}a_{km}^{(k)} & \text{für } i = j. \end{cases}$$

Bemerkung: In der zweiten Zeile von (60) können auch Nullen auftreten, z.B. bei Tridiagonalmatrizen außerhalb der Nebendiagonalen.

Stillschweigend war bisher angenommen, daß der Eliminationsalgorithmus nicht abbricht, d.h. daß $a_{kk}^{(k)} \neq 0$ für $k = 1, \dots, n-1$ gilt. Für schwach diagonaldominante \hat{M} -Matrizen ist dies nach Satz (58) erfüllt. Im Gegensatz zu M -Matrizen, die nach

Definition regulär sind, kann eine schwach diagonaldominante \hat{M} -Matrix natürlich singulär sein; aber auch für eine reguläre schwach diagonaldominante \hat{M} -Matrix schließt Satz (59) nicht aus, daß $A^{(n)}$ singulär ist, d.h. $a_{nn}^{(n)} = 0$. Für eine Teilklasse der schwach diagonaldominanten \hat{M} -Matrizen wird nun gezeigt, daß auch $A^{(n)}$ noch regulär ist.

Definition: $A \in \mathbf{R}^{n,n}$ heißt halb Stark diagonaldominant, wenn A schwach diagonaldominant ist und für ein $i \in \{1, \dots, n\}$ gilt

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|.$$

(61) **Satz:** Ist $A \in \mathbf{R}^{n,n}$ eine halb Stark diagonaldominante \hat{M} -Matrix mit symmetrischer Struktur bezüglich der Nulleinträge, dann gilt

a) für alle $k \in \{1, \dots, n\}$ hat $A^{(k)}$ mindestens eine positive Zeilensumme in den Zeilen k, \dots, n ,

b) $a_{nn}^{(n)} > 0$.

Beweis: Da A halb Stark diagonaldominante \hat{M} -Matrix ist, sei $s_i > 0$ für $i < n$ (für $i = n$ ist der Beweis hier fertig). Nach (58) b) ist dann $s_i^{(k)} > 0$ für alle $k \in \{1, \dots, n\}$. Für $n \geq k \geq i$ erhält man aus der Formel für $s_i^{(k+1)}$ in (58) b), indem man k durch i und i durch $m(i)$ ersetzt

$$(62) \quad s_{m(i)}^{(i+1)} = s_{m(i)}^{(i)} - \frac{a_{m(i),i}^{(i)}}{a_{ii}^{(i)}} s_i^{(i)}.$$

$m(i) > i$ ist nach Voraussetzung erfüllt und damit ist wegen der symmetrischen Struktur von A $a_{m(i),i}^{(1)} = a_{m(i),i} < 0$ sowie nach (58) $a_{m(i),i}^{(i)} < 0$, $a_{ii}^{(i)} > 0$ und $s_{m(i)}^{(i)} \geq 0$, insgesamt ist dann $s_{m(i)}^{(i+1)} \geq 0$. Nach (58) ist also $s_{m(i)}^{(k)} > 0$ für $k = i + 1, \dots, n$.

Bemerkung: $s_{m(i)}^{(k)} > 0$ muß nicht für $k < i + 1$ gelten.

Für $k = 1, \dots, m(i)$ ist bis jetzt in $A^{(k)}$ ein Zeile mit Index mindestens k gefunden, deren Summe positiv ist. Ersetzt man nun in (62), falls $m(i) < n$ ist, i durch $i' = m(i)$ und $m(i)$ durch $m(i') = m(m(i))$, so erhält man die nächste Zeile (die $m(m(i))$ -te) mit positiver Summe. Da alle $A^{(k)}$ \hat{M} -Matrizen sind, ist diese Verfahren wiederholbar, solange $m(i') < n$ ist, und der Fall $m(i') = n$ tritt ebenfalls auf. Letzteres besagt, daß $s_n^{(n)} = a_{nn}^{(n)} > 0$ ist. \square

Ist nun A noch zusätzlich positiv definit, was bisher für das PCG vorausgesetzt war, dann ist nach (60) R symmetrisch und damit auch LU . Man definiert

$$D := \text{diag}(u_{11}, \dots, u_{nn}) = \text{diag}(a_{11}^{(1)}, \dots, a_{nn}^{(n)}).$$

Nach (61) und (58) ist D regulär. Mit

$$(63) \quad C := LDL^T$$

ist dann, da L regulär ist, die Vorkonditionierungsmatrix C positiv definit.

Zur Kondition von \tilde{A}

Zur Abschätzung der Iterationsanzahl nach Satz (27) ist die Kenntnis der Konditionszahl $\kappa(\tilde{A}) = \tilde{\lambda}_n/\tilde{\lambda}_1$ notwendig. Es zeigt sich, daß bei der unvollständigen Zerlegung mit Addition in der Diagonalen $\kappa(\tilde{A}) = \tilde{\lambda}_n$ ist. Nach (60) sind alle Zeilensummen von $R^{(k)}$ null und damit auch alle Zeilensummen von R , nach (58) ist $l_{ik}a_{kj}^{(k)} \geq 0$, d.h. $r_{ij}^{(k+1)} \leq 0$ sowie $r_{ij} \leq 0$ für $i \neq j$.

Lemma: $R \in \mathbf{R}^{n,n}$ sei symmetrisch, $r_{ij} \leq 0$ für $i \neq j$, $\sum_{j=1}^n r_{ij} = 0$ für alle $i \in \{1, \dots, n\}$, dann ist R positiv semidefinit.

Beweis: $r_{ii} = -\sum_{\substack{j=1 \\ j \neq i}}^n r_{ij} = \sum_{\substack{j=1 \\ j \neq i}}^n |r_{ij}| =: \varrho_i$, $i = 1, \dots, n$. Die Kreise um r_{ii} mit Radius ϱ_i schneiden die positive Halbachse also in 0 und $2r_{ii} \geq 0$. Da R symmetrisch ist, sind alle Eigenwerte von R reell und liegen somit nach dem Satz von Gerschgorin in $[0, 2 \max_i \{r_{ii}\}]$. \square

Da alle Zeilensummen von R null sind, ist für $\mathbf{e} := (1, \dots, 1)^T$

$$(64) \quad R\mathbf{e} = \mathbf{0}.$$

Damit und mit $A = C + R$ ergibt sich

$$(65) \quad \frac{(A\mathbf{x}, \mathbf{x})}{(C\mathbf{x}, \mathbf{x})} = 1 + \frac{(R\mathbf{x}, \mathbf{x})}{(C\mathbf{x}, \mathbf{x})} \geq 1.$$

Es gilt nun

Lemma: $\tilde{\lambda}_n = \max_{\mathbf{x} \neq \mathbf{0}} (A\mathbf{x}, \mathbf{x}) / (C\mathbf{x}, \mathbf{x})$, $\tilde{\lambda}_1 = \min_{\mathbf{x} \neq \mathbf{0}} (A\mathbf{x}, \mathbf{x}) / (C\mathbf{x}, \mathbf{x})$, wobei $\tilde{\lambda}_n$ und $\tilde{\lambda}_1$ größter bzw. kleinster Eigenwert von $C^{-1}A$ ist.

Beweis: Sei $\tilde{L} := LD^{\frac{1}{2}}$ nach (63), also $C = \tilde{L}\tilde{L}^T$. Nach (32) hat ja $C^{-1}A$ die gleichen Eigenwerte wie $\tilde{L}^{-1}A\tilde{L}^{-T} =: \tilde{A}$. \tilde{A} ist positiv definit, also berechnen sich $\tilde{\lambda}_n$ und $\tilde{\lambda}_1$ zu

$$\tilde{\lambda}_n = \max_{\mathbf{y} \neq \mathbf{0}} \frac{(\tilde{A}\mathbf{y}, \mathbf{y})}{(\mathbf{y}, \mathbf{y})}$$

beziehungsweise

$$\tilde{\lambda}_1 = \min_{\mathbf{y} \neq \mathbf{0}} \frac{(\tilde{A}\mathbf{y}, \mathbf{y})}{(\mathbf{y}, \mathbf{y})}.$$

Setzt man nun $\mathbf{x} := \tilde{L}^{-T}\mathbf{y}$, dann ist, weil \tilde{L} regulär ist, $\mathbf{y} = \mathbf{0}$ genau dann, wenn $\mathbf{x} = \mathbf{0}$, also

$$\tilde{\lambda}_n = \max_{\mathbf{x} \neq \mathbf{0}} \frac{(\tilde{L}\tilde{L}^{-1}A\tilde{L}^{-T}\tilde{L}^T\mathbf{x}, \mathbf{x})}{(\tilde{L}\tilde{L}^T\mathbf{x}, \mathbf{x})} = \frac{(A\mathbf{x}, \mathbf{x})}{(C\mathbf{x}, \mathbf{x})}$$

und $\tilde{\lambda}_1$ entsprechend. \square

Wegen (65) und (64) ist dann $\tilde{\lambda}_1 = 1$ und $\kappa(\tilde{A}) = \kappa(C^{-1}A) = \tilde{\lambda}_n$.

Für bestimmte Matrizenklassen, die bei der Approximation von partiellen Differentialgleichungen durch Differenzenverfahren oder Finite-Elemente-Methoden entstehen, läßt sich zeigen (vgl. z.B. GUSTAFSSON in EVANS 1983), daß gilt

$$\tilde{\lambda}_n = \kappa(\tilde{A}) = O(\sqrt{\kappa(A)}) \text{ für } n \rightarrow \infty,$$

d.h. unter dem Aspekt der Iterationsschritte nach (27) lohnt sich die Vorkonditionierung für solche Matrizen.

7 Numerische Ergebnisse

7.1 Beschreibung des Programms und der Beispiele

Programmbeschreibung

Zum Erzielen numerischer Ergebnisse mit den dargestellten Verfahren für verschiedene lineare Gleichungssysteme wurde ein Programm in FORTRAN 77 geschrieben. Es besteht in der Hauptsache aus Unterprogrammen, und zwar zum einen aus den Unterprogrammen zur LU-Zerlegung einer Matrix und zum CG-/PCG-Verfahren, zum anderen aus Serviceunterprogrammen wie Matrix-Vektor-Multiplikation, $L * U$ -Vektor-Multiplikation, Vektor- und Inversiteration zur Bestimmung des größten bzw. kleinsten Eigenwertes, direkte Lösung eines Dreieckssystems u.a.

Wie schon in der Einleitung erwähnt, werden mit dem PCG hauptsächlich schwach besetzte Gleichungssysteme gelöst, weswegen nicht die ganze Matrix A gespeichert wird, sondern nur die Nichtnulleinträge. Die Symmetrie der Koeffizientenmatrix wird dabei nicht berücksichtigt. Die Matrizen werden folgendermaßen gespeichert:

In einem INTEGER-Feld SN der Länge n_A , wobei n_A die Anzahl der Nichtnulleinträge von A ist, stehen die Spaltenindices der Nichtnulleinträge, und zwar zeilenweise geordnet. Parallel dazu stehen in einem REAL-Feld EI gleicher Länge die jeweiligen Einträge selbst. Da es sich hier nur um positiv-definite Matrizen handelt, ist in jedem Fall $SN(1) = 1$, $SN(n_A) = n$, $EI(1) = a_{11}$, $EI(n_A) = a_{nn}$. In einem weiteren INTEGER-Feld ZA der Länge $n + 1$ stehen die Indices von SN bzw. EI der jeweiligen Zeilenanfänge, d.h. fängt in SN bzw. EI an der Stelle k_i die i -te Zeile an, dann ist $ZA(i) = k_i$, und $ZA(1) = 1$ in jedem Fall. Um für jede Zeile, also auch die letzte, den gleichen Abzählmodus, z.B. bei der Matrix-Vektor-Multiplikation, zu haben, wird gesetzt $ZA(n + 1) = n_A + 1$. Dann sind nämlich die Nichtnulleinträge der i -ten Zeile diejenigen von $EI(ZA(i))$ bis $EI(ZA(i + 1) - 1)$, die zugehörigen Spaltenindices entsprechend. Darüberhinaus läßt sich bei einem BYTE-orientierten Rechner, dessen INTEGER-Zahlenbereich man auf vier oder zwei Bytes verkleinern kann, noch Platz sparen, indem man ein weiteres Strukturmerkmal vieler, bei der Diskretisation vorkommender Matrizen berücksichtigt. Diese haben nämlich oft nur relativ wenige verschiedene Einträge. In dem Fall kann man ein INTEGER-Feld der Länge n_A einfügen, in welchem die Indices der Einträge des REAL-Feldes EI stehen, welches dann nur noch n_E lang sein muß, n_E die Anzahl der verschiedenen Einträge.

Da die CDC CYBER 180-835, auf der das Programm hauptsächlich betrieben wurde, nicht BYTE-orientiert ist, sondern alle Konstanten und Variablen außer den CHARACTER-Größen in 64-bit-Worten speichert, wurde letztere Variante nicht benutzt. Für größere Matrizen wurde das Programm auf dem Vektorrechner CRAY X-MP/24 des Zentrums für Informationstechnik Berlin (ZIB) betrieben.

Bei dieser Speichertechnik ist ein unmittelbarer Zugriff auf ein Element a_{ij} nicht mehr möglich. Man muß sich dafür eine Hilfsfunktion definieren, die einem angibt, ob $a_{ij} \neq 0$ ist, also ob a_{ij} überhaupt gespeichert ist und wenn ja, an welcher Stelle von EI es zu finden ist. Ein solcher Zugriff wird aber nur bei der LU-Zerlegung benötigt, und dort erfüllt diese Hilfsfunktion gleichzeitig die Aufgabe, anzugeben, ob (i, j) in der Indexmenge J (s.u.) enthalten ist oder nicht.

Die Zerlegungsmatrizen L und U werden auf die gleiche Weise gespeichert, aber nicht einzeln, sondern zusammen in einer Matrix. Dabei entspricht das rechte obere Dreieck mit der Diagonalen U , der Rest L . Die 1-Diagonale von L wird bei den Rechnungen dann hinzugefügt.

Die Zerlegung wird folgendermaßen durchgeführt:

Zuerst wird die Indexmenge J festgelegt. Dafür stehen drei Möglichkeiten zur Auswahl.

1. Es wird kein fill-in zugelassen, d.h. J enthält nur die Indices der Nichtnull-einträge.
2. J enthält die Indices $(1, 1), \dots, (n, n)$, darüber hinaus können noch Nebendiagonalen oder einzelne Stellen eingegeben werden, wo Nichtnulleinträge zugelassen werden. J wird dabei immer symmetrisch gehalten.
3. Es wird jegliches fill-in zugelassen; dies entspricht der vollständigen Zerlegung, welche für o.a. Serviceprogramme benötigt wird.

Mit Hilfe dieser Eingaben wird nun die Matrix wie vorne beschrieben zerlegt, d.h. in den Fällen 1. und 2. besteht noch die Wahlmöglichkeit, das fill-in zeilenweise zum Diagonalelement zu addieren oder nicht. Für beide Zerlegungen wird zur Kontrolle die Kondition von $C^{-1}A$ berechnet.

Das CG- und das PCG-Verfahren wird in einem gemeinsamen Unterprogramm durchgeführt. Beim PCG muß nur in den Skalarprodukten jeweils einmal das Residuum \mathbf{r} durch einen Hilfsvektor $\mathbf{h} = C^{-1}\mathbf{r}$ ersetzt werden (und dafür natürlich dieses lineare Gleichungssystem gelöst werden). Als Startvektor wird immer der Nullvektor gewählt. Abbruchkriterium ist das Unterschreiten von einer vorgegebenen Schranke eps durch $\|\mathbf{r}\|$.

Beispiele

Positiv definite Matrizen treten zum einen bei technischen Berechnungen (z.B. Netzberechnungen, Baustatik u.a.) oft auf, wo die quadratische Form F z.B. die potentielle Energie eines Systems darstellt. Zum anderen führt die Diskretisierung von Differentialgleichungen häufig zu positiv definiten Matrizen.

Diskretisierungen

Es wurden verschiedene Diskretisierungen der partiellen Differentialgleichung

$$-\frac{\partial}{\partial x} \left(a_1 \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left(a_2 \frac{\partial u}{\partial y} \right) + cu = f \text{ in } \Omega := (0, 1)^2$$

mit der Dirichlet-Randbedingung $u = g$ untersucht. Hierbei sind $a_1, a_2, c, u, f, g : [0, 1]^2 \rightarrow \mathbf{R}$ Abbildungen mit $a_1, a_2 \geq \text{konst} > 0, c \geq 0$.

Setzt man $a_1(x, y) \equiv a_2(x, y) \equiv 1, c(x, y) \equiv 0$, dann erhält man die Poisson-Gleichung und damit das sogenannte Modellproblem. Werden dann die Seiten des Einheitsquadrates in $m + 1$ gleiche Teile geteilt und wird für das entstehende Gitter die Differentialgleichung durch die Differenzengleichung ersetzt, so entsteht die Modellmatrix der Ordnung $n = m^2$ (im folgenden $\text{mod}n$ genannt), die folgende Form hat,

$$A =$$

mit m Matrizen T und der Einheitsmatrix I mit jeweils der Ordnung m .

Bei der Diskretisierung durch finite Elemente wird das Gebiet Ω in Teilgebiete (Elemente), üblicherweise Rechtecke oder Dreiecke zerlegt; die Schnittpunkte der Ränder der Elemente heißen Knoten. Auf Ω werden dann Funktionen definiert, welche nur für wenige Elemente um jeweils einen Knoten herum Funktionswerte ungleich null haben (lokale Träger). Diese Funktionen können z.B. als stückweise Polynome höchstens ersten Grades folgendermaßen zusammengesetzt sein: Die Funktion φ_j ist auf allen Elementen, welche den Knoten p_j enthalten, durch Polynome ersten Grades dargestellt, welche in p_j den Wert 1 und an allen Nachbarknoten von p_j den Wert 0 haben. Außerhalb dieser Elemente ist $\varphi_j = 0$.

Mit einer Linearkombination dieser φ_j erhält man eine Approximation \tilde{u} , welche bei Verfeinerung des Gitters gegen u^* konvergiert, welches als sogenannte schwache Lösung der Differentialgleichung ein zugeordnetes Variationsproblem löst.

Existiert ein klassische Lösung \hat{u} , so löst \hat{u} das Variationsproblem. Ist u^* hinreichend glatt, dann ist u^* klassische Lösung. (vgl. HACKBUSCH, SCHWARZ 1984). Die

Bestimmung der Koeffizienten dieser Linearkombination führt dann auf ein lineares Gleichungssystem.

Diese Methode hat gegenüber derjenigen der finiten Differenzen (s.o) die Vorteile, daß man zum einen eine Näherungslösung nicht nur punktweise auf den Knoten erhält, sondern als Funktion auf dem ganzen Gebiet Ω , zum anderen fallen Aussagen über die Koeffizientenmatrix wie z.B. Definitheit mit ab. Diese zugehörige Matrix (im Folgenden smn genannt) ist wie die Modellmatrix strukturiert, falls die Ansatzfunktionen stückweise Polynome höchstens ersten Grades sind, was für die Beispielrechnungen gewählt wurde. Für $c \neq 0$ hat dann die Matrix noch zwei weitere Nebendiagonalen (beginnend bei $(m, 1)$ und $(1, m)$, m wie vor).

Bei einer Unterteilung der Seiten des Einheitsquadrates in $m + 1$ gleiche Teile ergibt sich als Ordnung der Matrix $n = m^2$, das ist die Anzahl der inneren Knoten. Verfeinert man die Unterteilung, indem man die gegebene Teilung halbiert und fängt mit der Unterteilung für $m = 1$ an (Mehrgitterverfahren), dann erhält man für die l -te Verfeinerung als Ordnung der Matrix $(2^{l+1} - 1)^2$, also $1, 3^2, 7^2, 15^2, \dots$

Die Matrizen

Es wurden Gleichungssysteme mit mod100, mod225, mod400 und mod1600 sowie sm49, sm225, sm961 jeweils mit $c = 0$ und $c = 1$ gerechnet. Für smn wurde $a_1 = 0.01$, $a_2 = 1$ und $f = -2.02$ gewählt. Für die Erstellung der sm -Matrizen wurde ein Programm von G. Dornscheidt benutzt. Die Matrizen T haben für $smn(c=0)$ statt 4 und -1 die Einträge 2.02 bzw. -0.01. Als "Mischung" beider Arten wurden msm100 und msm225 gerechnet, für die T die Einträge 4 bzw. -0.01 hat.

Neben diesen Gleichungssystemen wurde versucht, ein Bild davon zu gewinnen, wie sich das Verfahren verhält, wenn die Koeffizientenmatrix fast singulär ist. Dazu wurde die Hilbert-Matrix $H = (h_{ij})$, $h_{ij} = \frac{1}{i+j-1}$, $i, j = 1, \dots, n$ der Ordnung n genommen und mit $\text{mod}n$ "geschnitten". Um das "fast Singulärsein" zu variieren, wurde der Nenner noch mit einem Exponenten α , $\alpha = 1, 2, \dots$ versehen, so daß die Einträge von A die Form $\frac{4}{(2i-1)^\alpha}$, bzw. $\frac{-1}{(i+j-1)^\alpha}$ bzw. 0, $i, j = 1, \dots, n$ haben und A die gleiche Besetzungsstruktur wie $\text{mod}n$ hat. A wird im Folgenden $almn, \alpha$ genannt.

Darüber hinaus wurde ein Gleichungssystem der Ordnung 144 gerechnet, dessen Koeffizientenmatrix positiv definit und in einer Bandbreite von 17 voll besetzt war.

Bei den Matrizen $\text{mod}m^2$, $\text{smm}^2(c = 0)$, $\text{alm}m^2, \alpha$ und $\text{msm}m^2$ wurden in jedem Fall Nichtnulleinträge in den Diagonalen zugelassen, die bei $(m+1, 1)$, $(2, 1)$, $(1, 1)$, $(1, 2)$ und $(1, m+1)$ beginnen, für $\text{smm}^2(c = 1)$ zusätzlich noch in den $(m, 1)$ - und $(1, m)$ -Diagonalen. Dies entspricht bis auf die Stellen $(m+1, m)$ und $(m, m+1)$, $(2m, 2m+1)$ und $(2m+1, 2m)$ usw. der Besetzung der Matrix A . Die Bezeichnung "(0)" bezieht sich dann auf dieses fill-in; bei "(1)" werden zwei, bei "(2)" vier weitere Diagonalen für fill-in zugelassen. So wird z.B. bei "(1) 7" auf der $(7,1)$ - und $(1,7)$ -Diagonalen

zusätzlich fill-in zugelassen. Die Gesamtzahl der zugelassenen Diagonalen differiert daher bei den sm-Matrizen zwischen den Fällen $c = 0$ und $c = 1$ um 2, wodurch sich die geringere Anzahl der benötigten Iterationen bei ungefähr gleicher Konditionszahl erklärt.

7.2 Ergebnisse

Tab. 1: Anzahl der Iterationen bis zum Unterschreiten verschiedener Abbruchschranken eps durch $\|\mathbf{r}\|$

eps	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-12}		
CG	24	31	35	38	41	mod100, $\kappa = 48$	
	47	59	69	77	85	mod400, $\kappa = 178$	
	17	24	33	37	38	sm49($c = 0$), $\kappa = 25$	
	44	64	81	101	119	sm225($c = 0$), $\kappa = 102$	
PCG (0)	9	12	15	18	20	mod100, $\kappa = 5.1$	
	9	12	15	18	21	$\kappa = 3.0$	
	15	21	25	28	33	mod400, $\kappa = 16.5$	
	14	18	22	26	30	$\kappa = 5.9$	
	3	5	6	7	9	sm49, $\kappa = 1.1$	
	3	4	5	6	8	$\kappa = 1.0$	
	5	7	10	12	14	sm225, $\kappa = 1.7$	
	3	6	8	10	12	$\kappa = 1.7$	
PCG (1) m	7	9	10	12	14	mod100, $\kappa = 2.4$	
	7	9	11	13	15	$\kappa = 1.9$	
	10	14	17	19	21	mod400, $\kappa = 6.7$	
	10	13	16	19	21	$\kappa = 3.4$	
	2	3	3	4	4	sm49, $\kappa = 1.0$	
	2	2	3	4	4	$\kappa = 1.0$	
	3	4	5	6	7	sm225, $\kappa = 1.0$	
	2	3	4	5	6	$\kappa = 1.0$	
	PCG (3) $m - 2,$ $m - 1, m$	5	7	8	9	11	mod100, $\kappa = 1.7$
		5	7	9	10	11	$\kappa = 1.4$
7		10	12	15	17	mod400, $\kappa = 3.9$	
7		10	12	14	17	$\kappa = 2.3$	
2		2	3	4	4	sm49, $\kappa = 1.0$	
2		2	3	4	4	$\kappa = 1.0$	
2		3	4	5	6	sm225, $\kappa = 1.0$	
2		3	4	5	6	$\kappa = 1.0$	

Beim PCG bezieht sich die jeweils erste Zeile auf die Zerlegung ohne Addition in der Diagonalen, die zweite Zeile auf diejenige mit Addition.

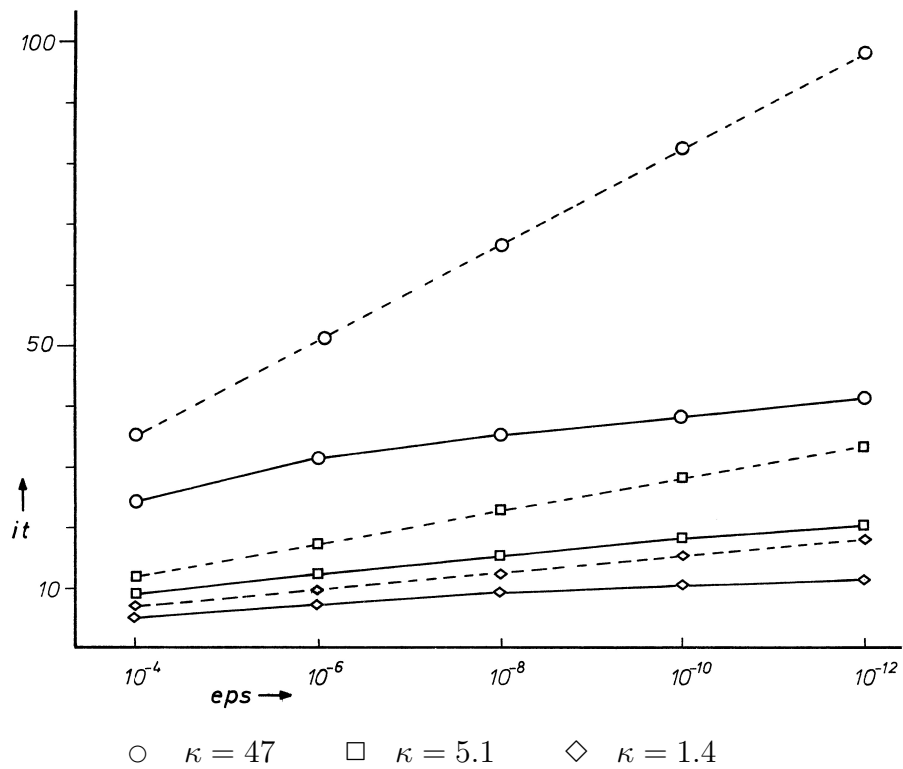


Bild 7: Anzahl der Iterationen in Abhängigkeit von der relativen Fehlerschranke eps für die Matrix mod100

Zeichenerklärung zu Bild 7 und Bild 8:

- CG ohne Vorkonditionierung
- PCG(0), ohne Addition in der Diagonalen
- ◇ PCG(3), mit Addition in der Diagonalen
- theoretische Schranke $t = \frac{1}{2}\sqrt{\kappa} \ln(2/\varepsilon) + 1$ bez. $\|\mathbf{r}\|_{A^{-1}}$
- tatsächlicher Wert bez. $\|\mathbf{r}\|$

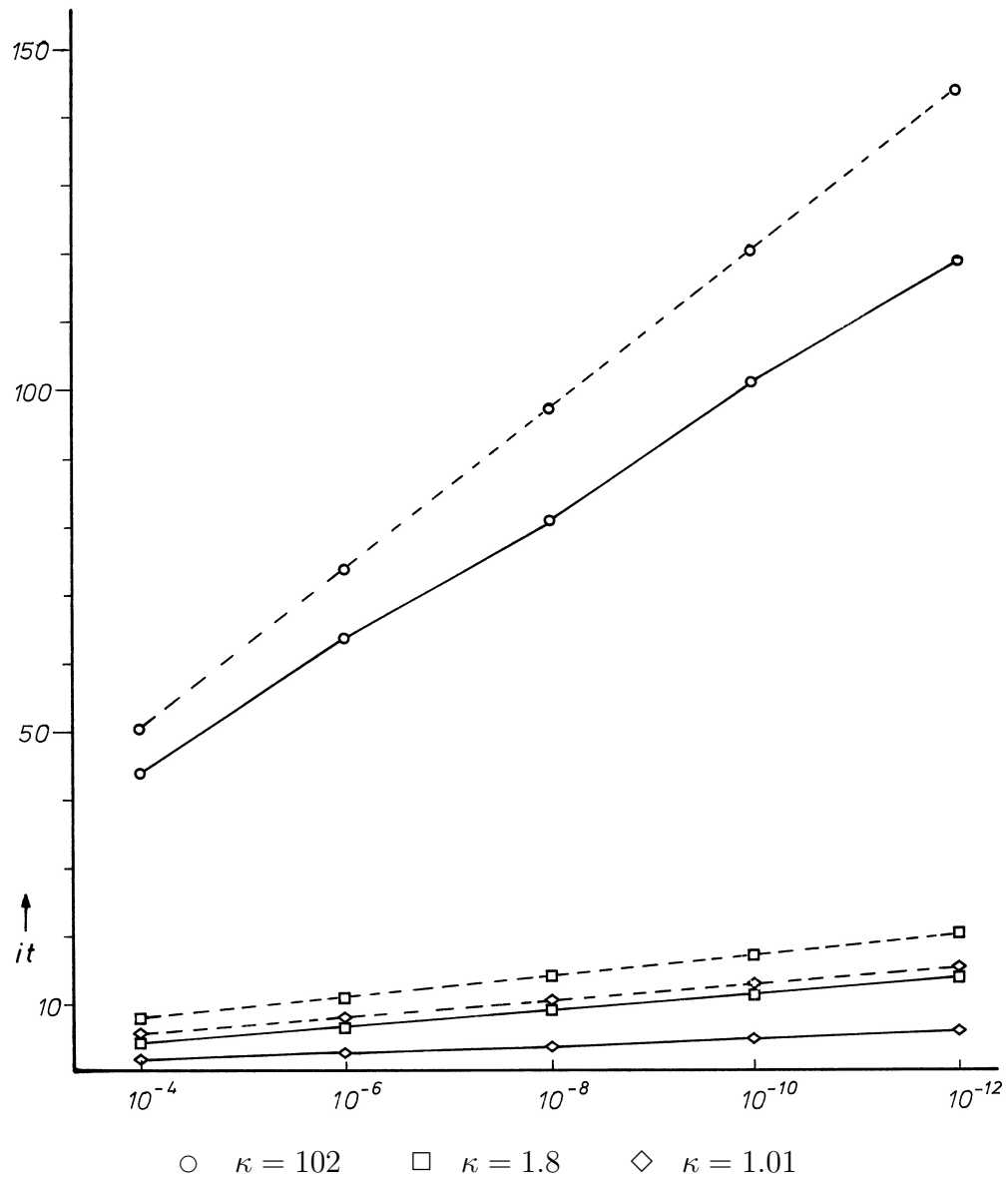


Bild 8: Anzahl der Iterationen in Abhängigkeit von der relativen Fehlerschranke eps für die Matrix $sm225(c=0)$

Tab. 2: Anzahl der Iterationen bis zum Unterschreiten von $eps = 10^{-6}$ durch $\|\mathbf{r}\|$

	CG	(0)	(1) 10		(3) 8-10	
mod100	31	12	9		7	
$\kappa = 48$		12	9		7	
			(1) 20	(2) 19,20	(3) 18-20	(4) 17-20
mod400	59	21	14	11	10	10
$\kappa = 178$		18	13	10	10	9
mod625	72	24				
$\kappa = 273$		20				
mod900	86	30				
$\kappa = 389$		22				
			(1) 40	(2) 3,40	(2) 39,40	(3) 38-40
mod1600	124	42	27	27	22	20
$\kappa = 681$		28	20	20	18	16
			(1) 7	(2) 3,7	(2) 6,7	(3) 5-7
sm49($c = 0$)	24	5	3	3	2	2
$\kappa = 25$		4	2	2	2	2
			(1) 6	(2) 3,6	(2) 5,6	(3) 4-6
sm49($c = 1$)	23	2	2	2	2	2
$\kappa = 25$		2	2	2	2	2
			(1) 15	(2) 3,15	(2) 14,15	(3) 13-15
sm225($c = 0$)	64	7	4	4	3	3
$\kappa = 102$		6	3	3	3	3
			(1) 14	(2) 3,14	(2) 13,14	(3) 12-14
sm225($c = 1$)	58	3	3	2	3	3
$\kappa = 92$		3	3	2	3	3
			(1) 31		(2) 30,31	(3) 29-31
sm961($c = 0$)	138	13	6		5	5
$\kappa = 409$		9	4		4	4
			(1) 30		(2) 29,30	(3) 28-30
sm961($c = 1$)	131	5	5		5	5
$\kappa = 372$		4	4		4	4

Beim PCG bezieht sich die jeweils erste Zeile auf die Zerlegung ohne Addition in der Diagonalen, die zweite Zeile auf diejenige mit Addition.

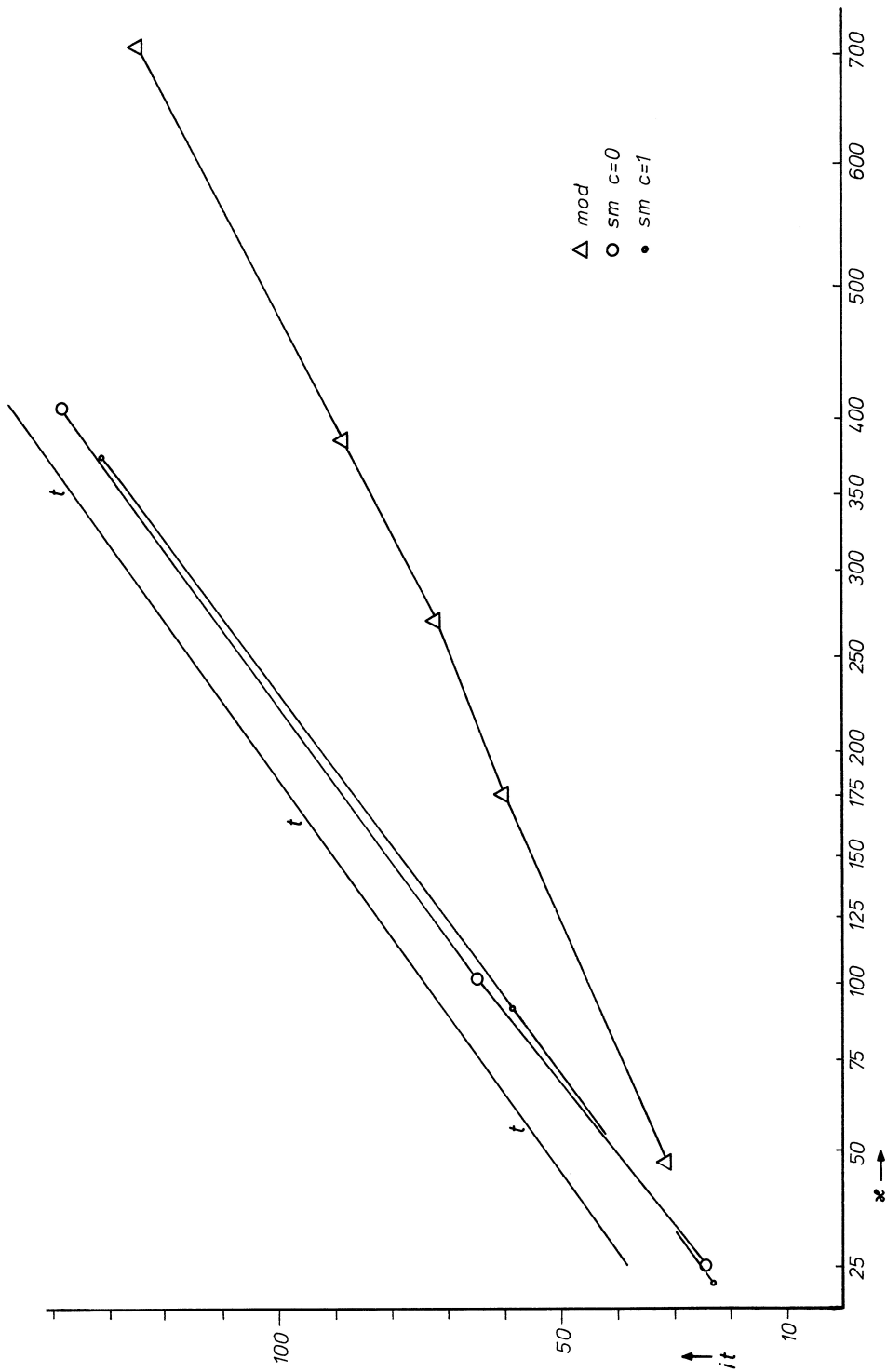


Bild 9: Anzahl der Iterationen in Abhängigkeit von der Kondition der Matrix A für $eps = 10^{-6}$, CG

Tab. 3: Anzahl der Iterationen und CPU-Zeiten

	mod100	msm100	mod225	msm225	sm225
CG	31	11	45	14	64
	0.164/0.045	0.061	0.532	0.165	0.206
	5.29/1.45	5.55	11.8	12.1	3.22
(0) o. Add.	12	3	17	3	7
	0.162/0.053	0.045	0.515	0.102	0.071
	13.5/4.42	15.0	30.3	34.0	10.1
(0) m. Add.	12	3	16	3	6
	0.163/0.053	0.046	0.481	0.101	0.061
	13.5/4.42	15.3	30.1	33.7	10.2
(1) o. Add.	9	2	11	2	4
	0.138/0.046	0.036	0.376	0.075	0.047
	15.3/5.11	18.0	34.2	37.5	11.8
(1) m. Add.	9	2	11	2	3
	0.140/0.046	0.036	0.375	0.076	0.036
	15.6/5.11	18.0	34.1	38.0	12.0
(3) o. Add.	7	2	8	2	3
	0.131/0.045	0.042	0.337	0.089	0.045
	18.7/6.43	21.0	42.1	44.5	15.0
(3) m. Add.	7	2	9	2	3
	0.131/0.045	0.043	0.375	0.091	0.045
	18.7/6.43	21.5	41.7	45.5	15.0

In der jeweils ersten Zeile steht die Anzahl der Iterationen für $eps = 10^{-6}$, in der zweiten die Rechenzeit in s , in der dritten die Rechenzeit je Iteration in ms . Bei mod100 sind die Zeiten für CDC/CRAY angegeben, bei sm225 für CRAY, sonst CDC.

Tab. 4: CPU-Zeiten der Rechner CDC und Cray in Sekunden für sm225($c = 0$)

	CDC	CRAY	speed-up
J f. vollst. Zerlegung	2.367	0.033	71.2
Zerlegung	11.075	2.435	4.5
CG	0.805	0.206	3.9
J (0)	0.009	0.001	9
Zerlegung i.M.	0.886	0.188	4.7
PCG o. Add.	0.274	0.071	3.9
PCG m. Add.	0.196	0.061	3.2
J (1), (2), (3) i. M.	2.64	0.033	80
Zerlegung	1.75	0.38	4.6
PCG	0.155	0.042	3.7

Tab. 5: Anzahl der Iterationen bis zum Unterschreiten von $eps = 10^{-6}$ durch $\|\mathbf{r}\|$

	CG	(0)		
alm9,0	5	5		
$\kappa = 5.8$		5		
alm9,1	9	6		
$\kappa = 35$		5		
alm9,2	10	5		
$\kappa = 466$		5		
alm9,3	12	6		
$\kappa = 7.1 \cdot 10^3$		5		
alm16,0	9	7		
$\kappa = 9.5$		7		
alm16,1	18	7		
$\kappa = 92$		7		
alm16,2	21	7		
$\kappa = 2.1 \cdot 10^3$		7		
alm16,3	26	7		
$\kappa = 5.5 \cdot 10^4$		7		
alm16,4	30	7		
$\kappa = 1.5 \cdot 10^6$		7		
	CG	(0)	(1) 10	(3) 8-10
alm100,1	83	12	8	7
$\kappa = 2.6 \cdot 10^3$		13	9	7
alm100,2	214	13	8	7
$\kappa = 3.4 \cdot 10^5$		14	9	7
alm100,3	525	13	8	7
$\kappa = 5.1 \cdot 10^7$		15	9	7
alm100,4	1233	13	8	7
$\kappa = 8.1 \cdot 10^9$		19	9	7

Tab. 6: Anzahl der Iterationen bis zum Unterschreiten von $eps = 10^{-6}$ durch $\|\mathbf{r}\|$ für eine Matrix der Ordnung 144

	it	κ
CG	37	355
PCG 9	38	337
PCG 8,9	38	311
PCG 6-9	38	274
PCG 2,5-9	32	163
PCG 2-4,8,9	25	114

Die Matrix ist folgendermaßen besetzt (erste Zeile):

32 -4 -4 -2 -2 -1 -1 -1 -1.

Beim PCG unterscheiden sich die Iterationsanzahlen nicht nach ohne/mit Addition in der Diagonalen, die Konditionszahlen unterscheiden sich nur wenig; angegeben ist der jeweilige Mittelwert.

PCG 8,9 z.B. bedeutet, daß für fill-in die Hauptdiagonale und die (9,1)-, (8,1)-, (1,8)- und (1,9)-Nebendiagonalen zugelassen werden.

Zusammenfassung der Ergebnisse

Die theoretische Schranke für die höchstens nötige Anzahl der Iterationen bezüglich der A^{-1} -Norm (Satz (27)) erwies sich auch als gültig für das Residuum \mathbf{r} , gemessen in der euklidischen Norm. Es zeigte sich aber, daß für bestimmte Matrizen und kleine relative Fehlerschranken eps oder große Konditionszahl κ diese Abschätzung sehr grob ist (vgl. Bild 7 und Bild 9).

Für die hier hauptsächlich untersuchten schwach besetzten Bandmatrizen ist das PCG deutlich besser als das CG-Verfahren im Hinblick auf die Anzahl der Iterationen, und zwar sowohl mit als auch ohne Addition in der Diagonalen. Für M-Matrizen ist es im allgemeinen wohl besser, das weggelassene fill-in zum Diagonalelement zu addieren.

Bei den jeweiligen Rechenzeiten ist eine differenziertere Betrachtung notwendig. So gibt es Matrizen, für die das PCG mehr Rechenzeit beansprucht als das CG-Verfahren (Modellmatrix) und solche, für die es umgekehrt ist (vgl. Tab. 3). Hier kommt es auf die Zeit pro Iterationsschritt an. Diese ist für das PCG wegen der zusätzlichen Lösung eines Gleichungssystems in jedem Fall größer als für das CG. Dieser zusätzliche Zeitbedarf sollte durch eine entsprechend geringe Anzahl von Iterationsschritten ausgeglichen werden. Unter diesem Aspekt der Rechenzeit für das PCG ist es für manche Matrizen also unter Umständen besser, mehrere Nebendiagonalen für fill-in zuzulassen. Die hier untersuchten Matrizen von der Art smn und $msmn$ sind solche, bei denen die Einträge in den äußeren Nebendiagonalen betragsmäßig deutlich größer sind als diejenigen der Nebendiagonalen, die nahe bei der Hauptdiagonalen liegen. Für diese Matrizen sind dann die Fehlermatrizen R "kleiner" als für die der Art $modn$, was in Übereinstimmung steht mit den theoretischen Überlegungen am Ende von Kapitel 4. Durch den höheren Zeitaufwand pro Iterationsschritt bei mehreren für fill-in zugelassenen Nebendiagonalen steigt die Rechenzeit wieder an, während die Anzahl der Iterationen nicht mehr wesentlich reduziert wird. Läßt man überhaupt Nebendiagonalen für fill-in zu, so scheint das Optimum bei einer oder zwei Nebendiagonalen zu liegen.

Für die unvollständige Zerlegung konnte bestätigt werden, daß das Zulassen von keinen oder nur wenigen Nebendiagonalen für fill-in schon gute Ergebnisse bringt (vgl. Tab. 1). In jedem Fall scheint es günstig zu sein, diese Nebendiagonalen direkt neben den äußersten ursprünglich besetzten Nebendiagonalen anzuordnen. Der Effekt von zugelassenem fill-in auf Nebendiagonalen, die nahe bei der Hauptdiagonalen liegen, ist sehr gering. Rechentechnisch ist es günstig, kein fill-in zuzulassen, weil dann statt der Felder BZA und BSN (s. Anhang) direkt AZA und ASN genommen werden können, was eine deutlich geringere Rechenzeit für die Zerlegung benötigt als das Vorsehen von weiteren Nebendiagonalen.

Für die schlecht konditionierten, fast singulären Matrizen $almn, \alpha$ ist das nicht vorkonditionierte CG-Verfahren kaum geeignet (vgl. Tab. 5). Es sind oft schon mehr

als n Iterationsschritte zur Lösung nötig, was im Widerspruch zu dem in Kapitel 3 erwähnten theoretischen Ergebnis steht. Das PCG bringt hier im Vergleich sehr gute Ergebnisse.

Bei ziemlich voll besetzten Matrizen (hier eine der Ordnung 144, vgl. Tab. 6) ist sowohl das CG-Verfahren wie das PCG wenig geeignet. Um eine deutliche Verringerung der Iterationsanzahl durch das PCG zu erreichen, ist ein großer Rechenaufwand nötig. Im Unterschied zu den bisherigen Matrizen verletzt hier auch jede unvollständige Zerlegung die Bedingung, daß für die Nichtnulleinträge wenigstens die Besetzungsstruktur von A zur Verfügung stehen soll.

7.3 Ausblicke

Als theoretisches Ergebnis interessant wäre sicherlich eine Abschätzung für $\tilde{\lambda}_n$ und damit von $\kappa(\tilde{A})$ in Abhängigkeit von $\kappa(A)$ beim PCG mit Addition in der Diagonalen, womit sich dann quantitative Aussagen über die Verbesserung durch Vorkonditionierung machen ließen. Für diese Vorkonditionierung wäre auch noch praktisch das theoretische Ergebnis aus der Literatur (vgl. z.B. GUSTAFSSON) zu bestätigen, wonach die Anzahl der Iterationen von der Ordnung $O(h^{-1})$ ist, wobei $h = \frac{1}{m+1}$, m wie im Abschnitt über Diskretisierungen beschrieben. Dafür ist die hier vorhandene Datenbasis zu klein.

Die Konvergenzverbesserung durch das PCG hängt von der Struktur der Matrix ab. Durch Zeilen- und Spaltenvertauschung läßt sich eine Matrix, für die das PCG einen großen Effekt bringt, in eine mit gleichen Eigenwerten transformieren, bei der der Gewinn durch das PCG geringer ist. Zu untersuchen wäre daher, für welche Matrizen dieser Effekt größer oder kleiner ist.

Ein weiterer interessanter Aspekt ist die Effektivierung des Verfahrens durch Anwendung von Techniken der parallelen Numerik. Wie Tab. 3 und Tab. 4 zeigen, ist zumindest in Teilbereichen eine deutliche Reduzierung der Rechenzeiten zu erwarten.

8 Anhang

Programmlisting und Dokumentation

```
PROGRAM PCGUA
* VERSION VOM 8.9.87, BEARBEITET 3.5.88
* SYSTEM A*X=B
  IMPLICIT LOGICAL (A-Z)
  INTEGER N1, N2, NSA, I, Z, J, DN, DSN, ZN2, ALLI
  PARAMETER (DN=250, DSN=2000, N2=7000)
  PARAMETER (ZN2=N2)
* N2 = GANZE HUELLE DER MATRIX
* N1: DIMENSION; NSA: ANZ. EINTR. IN ASN, AEI
* N2 BSN, BEI
  REAL XH1(DN), XH2(DN), VEKB(DN), VEKX(DN), XH3(DN), SEK
  REAL AEI(DSN), XH4(DN), BEI(N2), ZEIT, ZEI(ZN2), SECOND, ZAIT

  INTEGER AZA(DN+1), ASN(DSN), BZA(DN+1), BSN(N2)
  INTEGER ZZA(DN+1), ZSN(ZN2), BYTE(9), DIAB(DN), DIAZ(DN)
* BYTE(4): CG-PCG, (5): JOTT, (6): ZERL, (7): VEIN, (9): DRGL
* BYTE(1): VEKTOR DRUCKEN JA/NEIN
* BYTE(8): JOTT+ZERL, DAMIT BEIDES HINTEREINANDER AUSGEFUEHRT WIRD
* MATRIX A MUSS AUF 'MATR' IN ZEILENORIENTIERTER FORM STEHEN
* VEKTOR B AUF 'VEKB'
  DOUBLEPRECISION DPRO(DN)
  CHARACTER ST1*15, ST2*15, NAME*20
* FUER INPUT VON FILE 'DAT': /READ(*, /=/READ(24, /, A
*
  DO 15, I=1, DN
    VEKX(I)=0
    XH1(I)=0
    XH2(I)=0
    XH3(I)=0
    XH4(I)=0
    AZA(I)=0
    VEKB(I)=0
    BZA(I)=0
    ZZA(I)=0
    DIAB(I)=0
    DIAZ(I)=0
15  DPRO(I)=0
    AZA(N1+1)=0
    BZA(N1+1)=0
    ZZA(N1+1)=0
    DO 17, I=1, DSN
      ASN(I)=0
17  AEI(I)=0
    DO 19, I=1, N2
      BSN(I)=0
19  BEI(I)=0
    DO 20, I=1, ZN2
```

```

        ZSN(I)=0
20      ZEI(I)=0
        DO 21,I=1,15
21      BYTE(I)=0
*
*
        ZEIT=SECOND()
        ZAIT=ZEIT
        ST1='*****'
        ST2='????????????????'
        CALL MAEI(DN,DSN,N1,NSA,AZA,ASN,AEI,VEKB)
        PRINT*
        PRINT*, 'PARAMETER_SCHON_EINGESTELLT?'
        PRINT*, 'AKTUELL: N1', N1, 'NSA', NSA, 'N2', N2
        PRINT*, ST1, ST1
        PRINT*, 'NAME_DER_MATRIX_AUF_NOTIZ, MAX. 20 ZEICHEN'
        READ(*, '(A)') NAME
        PRINT*, 'MATRIX', NAME
        PRINT*, ST1, ST1
        OPEN (25, FILE='NOTIZ')
        OPEN (24, FILE='DAT')
        REWIND 24

        WRITE(25,*)
        WRITE(25,*) 'MATRIX', NAME
        WRITE(25,*) '*****'
        WRITE(25,*) 'ERSTE_ZEILE_VON_A, DIMENSION:', N1
        WRITE(25,*) (ASN(I), I=1, AZA(2)-1)
        WRITE(25,*) (AEI(I), I=1, AZA(2)-1)
        WRITE(25,*)
25      PRINT*
        SEK=SECOND()
        ZAIT=SEK-ZEIT
        IF (BYTE(1).EQ.0) THEN
            PRINT*, 'JETZT_OHNE_VEKTORAUSGABE, MIT: 1'
        ELSE
            PRINT*, 'JETZT_MIT_VEKTORAUSGABE, OHNE: 1'
        END IF
*      PRINT*, 'MATRIX-VEKTOR-MULTIPLIKATION: 2'
        PRINT*, 'POWERITERATION: 3'
        PRINT*, 'KONJUGIERTE_GRADIENTEN: 4'
        PRINT*, 'FILLIN_INDEXMENGE_J_EINGEBEN: 5'
        PRINT*, 'ZERLEGUNG: 6'
        PRINT*, 'INVERSITERATION: 7'
        PRINT*, 'VORKOND.CG: 8'
        PRINT*, 'DIREKTE_LOESUNG: 9'
        PRINT*, 'ABFRAGE_M-MATRIX: 11'
        PRINT*, 'AUSSTEIGEN: 0', ST2
*      SEK=SECOND()
*      ZAIT=SEK-ZEIT
        PRINT 26, ZAIT, 'SEK'
        WRITE(25,26) ZAIT, 'SEK'

```

```

26  FORMAT (F8.3,A)
    SEK=SECOND()
    ZAIT=SEK-ZEIT
    IF (ZAIT.GT.205) THEN
        WRITE(25,*) 'ABBRUCH_VOR_TIMEOUT'
        GOTO 99
    END IF
    READ(*,*) Z
*
    IF (Z.EQ.9) THEN
        BYTE(9)=0
        CALL DRGL(N1,N2,ZZA,ZSN,ZEI,DPRO,VEKB,XH1,BYTE,DIAZ)
        PRINT*, 'DIREKTE_LOESUNG(AUF"DIREKT)": '
        IF (BYTE(1).EQ.1) CALL PRIVEC(XH1,N1)
        OPEN (22,FILE='DIREKT')
        REWIND 22
        DO 27,I=1,N1
27  WRITE (22,*) '  ',I,'  ',XH1(I)
        CLOSE(22)
*
    ELSE IF (Z.EQ.2) THEN
*
        CALL MAVE(N1,AEI,ASN,AZA,XH1,AUS)
    ELSE IF (Z.EQ.6) THEN
        CALL ZERL(N1,N2,NSA,AZA,ASN,AEI,BYTE,BZA,BSN,BEI,
*ZZA,ZSN,ZEI,DPRO,XH1,XH2,XH3,XH4,DIAB,DIAZ)
    ELSE IF (Z.EQ.5) THEN
        CALL JOTT(N1,N2,NSA,AZA,ASN,BYTE,BZA,BSN)
    ELSE IF (Z.EQ.7) THEN
        CALL INVIT(N1,N2,ZZA,ZSN,ZEI,DPRO,XH1,XH2,XH3,BYTE,DIAZ)
    ELSE IF (Z.EQ.3) THEN
        CALL POIT(N1,NSA,N2,AEI,ASN,AZA,XH1,XH2,BYTE)
    ELSE IF (Z.EQ.1) THEN
        BYTE(1)=ABS(BYTE(1)-1)
    ELSE IF ((Z.EQ.4).OR.(Z.EQ.8)) THEN
        BYTE(4)=Z
        CALL KOGR(N1,VEKB,ST1,ST2,VEKX,XH1,XH2,XH3,XH4,AEI,ASN,
*AZA,DPRO,BZA,BSN,BEI,N2,NSA,BYTE,DIAB)
    ELSE IF (Z.EQ.0) THEN
        PRINT*, 'FILE_ "NOTIZ" IN_ SICHERHEIT_ BRINGEN!!! '
        GOTO 99
    ELSE IF (Z.EQ.11) THEN
        DO 28,I=1,N1
            CALL RINI(N1,1,XH1)
            XH1(I)=1
            BYTE(9)=1
            CALL DRGL(N1,N2,ZZA,ZSN,ZEI,DPRO,XH1,XH2,BYTE,DIAZ)
            DO 28,J=1,N1
                IF(XH2(J)) 29,28,28
28  CONTINUE
        PRINT*, 'A^-1_>=0'
        WRITE(25,*) '  '
        WRITE(25,*) '  A^-1_>=0'
        GOTO 30

```

```

29     PRINT*, 'A( ', J, ', ', ', I, ') _=_' , XH2(J)
       PRINT*, ' KEINE_M-MATRIX '
       WRITE(25,*) ' _ _ _ _ _ A^-1( ', J, ', ', ', I, ') _=_' , XH2(J)
       WRITE(25,*) ' _ _ _ _ _ KEINE_M-MATRIX '
30     CONTINUE
       ELSE
         PRINT*, ST2
         PRINT*, ' NUMMER_' , Z, ' _ NICHT _ IN _ DER _ LISTE ! '
         PRINT*, ST1
         GOTO 25
       END IF
       GOTO 25
99     CONTINUE
       CLOSE (24)
       CLOSE (25)
       END
*****
       SUBROUTINE KOGR(N1, B, ST1, ST2, X1, GR, DI, HA, HI, AEI, ASN,
*AZA, DPRO, BZA, BSN, BEI, N2, NSA, BYTE, DIAB)
* CG, PCG NACH AX/BARK 23+29
       IMPLICIT LOGICAL (A-Z)
       INTEGER N1, I, J, K, NSA, N2, SZ, NJO, BYTE(*)
       INTEGER ASN(NSA), AZA(N1+1), BZA(N1+1), BSN(N2), DIAB(*)
* BYTE(4)=4: CG, =8: PCG
       REAL EPS, BETA, TAU, DO, D1, D2, B(N1), BEI(N2), ZEIT, ZEITA
       REAL X1(N1), GR(N1), DI(N1), HA(N1), HI(N1), AEI(NSA), SKALP
       CHARACTER ST1*15, ST2*15, ZEI
       DOUBLE PRECISION DPRO(*)
       EXTERNAL SKALP
*
*
       CALL RINI(N1, 1, X1)
       CALL RINI(N1, 1, HI)
       IF (BYTE(4).EQ.4) THEN
         PRINT*, ' CG-VERFAHREN '
       ELSE
         IF (BYTE(8).EQ.1) THEN
           IF (BYTE(5).EQ.1) THEN
             PRINT*, ' OHNE_FILL-IN '
           ELSE IF (BYTE(5).EQ.2) THEN
             PRINT*, ' FILL-IN_AUF_NEBENDIAGONALEN '
           ELSE
             PRINT*, ' KEINE_VERNUENFTIGE_ZERLEGUNG_VORHANDEN '
             RETURN
           END IF
         ELSE
           PRINT*, ' MIT_"5"_UND_"6"_NOCHMAL_ZERLEGEN '
           RETURN
         END IF
       END IF
       ZEIT=SECOND()
       EPS=1.E-6

```

```

*      PRINT*, 'ABBRUCHSCHRANKE EPS=????'
*      PRINT*, ST2
*      READ(*,*) EPS
*
*
*      J=0
*      DO 5, I=1, N1
5      X1(I)=0
*      CALL MAVE(N1, AEI, ASN, AZA, X1, X1)
*      X1:=A*X1
*      STARTVEKTOR X1=0
*      DO 15, I=1, N1
15     GR(I)=-B(I)
*      HA(I)=GR(I)
*      IF (BYTE(4).EQ.8) THEN
*          BYTE(9)=2
*          CALL DRGL(N1, N2, BZA, BSN, BEI, DPRO, GR, HA, BYTE, DIAB)
*      END IF
*      H=C** -1 *G
*      DO 25, I=1, N1
25     DI(I)=-HA(I)
*      D=-H FALLS PCG, D=-G FALLS CG
*      DO=SKALP(GR, HA, N1)
*      D2=SKALP(GR, GR, N1)
*      IF (SQRT(D2)-EPS) 98, 98, 32
32     J=J+1
*      CALL MAVE(N1, AEI, ASN, AZA, DI, HI)
*      HILFSVEKTOR HI=A*DI
*      TAU=DO/SKALP(DI, HI, N1)
*      DO 35, I=1, N1
35     X1(I)=X1(I)+TAU*DI(I)
*      GR(I)=GR(I)+TAU*HI(I)
*      HA(I)=GR(I)
*      D2=SKALP(GR, GR, N1)
*      IF (SQRT(D2)-EPS) 98, 98, 42
42     CONTINUE
*      IF (BYTE(4).EQ.8) THEN
*          BYTE(9)=2
*          CALL DRGL(N1, N2, BZA, BSN, BEI, DPRO, GR, HA, BYTE, DIAB)
*      END IF
*      D1=SKALP(GR, HA, N1)
*      BETA=D1/DO
*      DO=D1
*      DO 45, I=1, N1
45     DI(I)=BETA*DI(I)-HA(I)
*      D=-G+BETA*D BZW D=-H+BETA*D
*      GOTO 32
98     ZEITA=SECOND()-ZEIT
*      PRINT*, ST1
*      PRINT*, 'IT', J, 'ZEIT', ZEITA
*      PRINT*, 'LOESUNG'
*      IF (BYTE(1).EQ.1) CALL PRIVEC(X1, N1)

```

```

      K=BYTE(4)/4+.01
      GOTO (110,120) K
110  PRINT*, 'FEHLER IN R', SQRT(D2)
      WRITE(25,*)
      WRITE(25,*) 'CG OHNE VORKONDITIONIERUNG', ZEITA
      WRITE(25,*) 'EPS', EPS, 'IT=', J, 'FEHLER', SQRT(D2)
      RETURN
120  PRINT*, 'FEHLER IN R', SQRT(D2)
      WRITE(25,*)
      WRITE(25,*) 'VORKONDITIONIERTES CG', ZEITA
      WRITE(25,*) 'EPS', EPS, 'IT=', J, 'FEHLER', SQRT(D2)
99   END
*****
      SUBROUTINE MAVE(N1, AEI, ASN, AZA, X, ERG)
      INTEGER N1, I, J, ASN(*), AZA(*)
      REAL AEI(*), X(N1), ERG(N1), SUM
* AEI: MATRIXEINTRAEGE
* ASN: SPALTENNUMMERN DER EINTRAEGE
* AZA: ZEILENANFAENGE (ERSTES NICHTNULLEL. IN DER ZEILE)
*
      DO 10, I=1, N1
          SUM=0
          DO 5, J=AZA(I), AZA(I+1)-1
5             SUM=SUM+AEI(J)*X(ASN(J))
10            ERG(I)=SUM
99           END
*****
      SUBROUTINE MAEI(DN, DSN, N1, NSA, AZA, ASN, AEI, KLB)
* DIE DIMENSIONEN MUESSEN AM ANFANG ALS @PARAMETER@ INS HAUPTPROGRAMM
* GESCHRIEBEN WERDEN!
      INTEGER N1, NSA, DN, DSN
      INTEGER AZA(DN+1), ASN(DSN)
      REAL AEI(DSN), KLB(DN)

      OPEN(22, FILE='MATR')
      REWIND 22
      READ(22,*) N1, NSA
      READ(22,*) (AZA(I), I=1, N1+1)
      READ(22,*) (ASN(I), I=1, NSA)
      READ(22,*) (AEI(I), I=1, NSA)
      CLOSE(22)
      OPEN(22, FILE='VEKB')
      REWIND 22
      READ(22,*) (KLB(I), I=1, N1)
      CLOSE(22)
99   END
*****
      SUBROUTINE POIT(N1, NSA, N2, AEI, ASN, AZA, X1, X2, BYTE)
* POWERITERATION SCHWARZ 176FF

      IMPLICIT LOGICAL (A-Z)
      INTEGER N1, I, J, K, KZ, NSA, N2

```

```

INTEGER ASN(NSA),AZA(N1+1),BYTE(*)
REAL AEI(N1+1),RAY,RAYV,RAY5,X1(N1),X2(N1)
DOUBLE PRECISION SP,SKPR
EXTERNAL SKPR

CALL RINI(N1,1,X2)
J=0
RAY=0
DO 5,I=1,N1
5   X1(I)=SIN(I*6.1/N1)
* ANFANGSVEKTOR
CALL MAVE(N1,AEI,ASN,AZA,X1,X2)
2   J=J+1
RAYV=RAY
SP=SKPR(X2,X2,N1)
SP=SQRT(SP)
IF (SP.EQ.0) STOP 'NULLVEKTOR;_STOP'
DO 25,I=1,N1
25  X1(I)=X2(I)/SP
* NORMIERT
CALL MAVE(N1,AEI,ASN,AZA,X1,X2)
* BEDEUTET: X1=X(K+1),X2=A*X(K+1)
RAY=SKPR(X2,X1,N1)
* RAYLEIGHQUOTIENT VON X(K+1) (EINHEITSVEKTOR)
IF (J.LE.5) THEN
RAY5=RAY
GOTO 2
END IF

IF (ABS(RAYV-RAY).GT.1.E-4*RAY5) GOTO 2
PRINT*, 'EIGENVEKTOR_'
IF (BYTE(1).EQ.1) CALL PRIVEC(X1,N1)
PRINT*, 'LAMBDA-N_=',RAY, ' _IT_=',J
WRITE(25,*)
WRITE(25,*) ' _LAMBDA-N_=',RAY, ' _IT_=',J
WRITE(25,*)
END

*****
SUBROUTINE IINI(N1,N2,X)
* INITIALISIERT EIN- ODER ZWEIDIMENSIONALE FELDER
INTEGER N1,N2,I
INTEGER X(N1*N2)

DO 1,I=1,N1*N2
1   X(I)=0
END
*****
SUBROUTINE RINI(N1,N2,X)
INTEGER N1,N2,I
REAL X(N1*N2)

```

```

DO 1, I=1, N1*N2
1   X(I)=0
END
*****
SUBROUTINE DINI(N1, N2, X)
INTEGER N1, N2, I
DOUBLE PRECISION X(N1*N2)

DO 1, I=1, N1*N2
1   X(I)=0
END
*****
DOUBLE PRECISION FUNCTION SKPR(X, Y, N)
* SKALARPRODUKT
INTEGER N, I
REAL X(N), Y(N)

SKPR=0
DO 5, I=1, N
5   SKPR=SKPR+DPROD(Y(I), X(I))
END
*****
REAL FUNCTION SKALP(X, Y, N)
INTEGER N, I
REAL X(N), Y(N)

SKALP=0
DO 5, I=1, N
5   SKALP=SKALP+Y(I)*X(I)
END
*****
INTEGER FUNCTION SPAL(ZA, SN, I, J)
* ERMITTELT AUS (I, J) DIE STELLUNG IN .SN
INTEGER I, J, N
INTEGER ZA(*), SN(*)

DO 5, N=ZA(I), ZA(I+1)-1
IF (J.EQ.SN(N)) THEN
SPAL=N
RETURN
END IF
5 CONTINUE
SPAL=0
END
*****
SUBROUTINE JOTT(N1, N2, NSA, AZA, ASN, BYTE, BZA, BSN)
IMPLICIT LOGICAL (A-Z)
INTEGER N1, N2, NSA, I, J, K, L, M, ANZDIA, UN, OB, SJ
PARAMETER (UN=-2000, OB=4000)
INTEGER AZA(*), ASN(*), JO(UN:OB), DIA(100), BZA(*), BSN(*), BYTE(*)

```

```

PRINT*, '***_FILLIN***'
5 PRINT*, 'FALLS_KEIN_ZUSAETZLICHES_FILLIN:_____1'
PRINT*, 'FALLS_DIAGONALE+_ANDERE_____2'
PRINT*, 'FALLS_VOLLSTAENDIGE_ZERLEGUNG:_____3'
PRINT*, 'AUSSTEIGEN:_____0'
PRINT*, '???'
READ(*,*) L
BYTE(5)=L
BYTE(8)=0
IF ((L.LT.0).OR.(L.GT.3)) THEN
    PRINT*, 'NR_',L,'_HAMWANICH'
    GOTO 5
END IF
WRITE(25,*) '*****'
WRITE(25,*)
GOTO (12,14,32)L
RETURN

12 WRITE(25,*) '___KEIN_ZUSAETZLICHES_FILLIN'

DO 131,I=1,N1+1
131 BZA(I)=AZA(I)

DO 132,J=1,NSA
132 BSN(J)=ASN(J)

GOTO 99

14 WRITE(25,*) '___DIAGONALE'
ANZDIA=0
17 PRINT*, 'ANFANGSSPALTE_D._OBEREN_NEBENDIAG_?;_FERTIG:_0_'
READ(*,*) I
IF(I.EQ.0) GOTO 55
IF ((I.LT.1).OR.(I.GT.N1)) THEN
    PRINT*, 'EINTRAG_(',I,')_HAMWANICH'
    GOTO 17
END IF
ANZDIA=ANZDIA+1
WRITE(25,*) '___PLUS_NEBENDIAG._(',I,',' ,1,')_UND_(',1,',' ,I,')'
DIA(ANZDIA)=I
GOTO 17

32 WRITE(25,*) '___GANZES_ENVELOPE'
ANZDIA=ASN(AZA(2))-1
PRINT*, 'DIE_RECHTESTE_NEBENDIAGONALE_MUSS_IN_DER_ERSTEN_ZEILE'
PRINT*, 'BESETZT_SEIN!'
PRINT*, 'ANDERNFALLS_"ANZDIA"_IN_JOTT_32_SO_SETZEN'

DO 33,J=1,ANZDIA
33 DIA(J)=J

```

```

55   BZA(1)=1
      DO 60, I=1,N1

          DO 56, L=UN,OB
56       JO(L)=0

          DO 57,L=1,ANZDIA
              JO(I+(DIA(L)-1))=1
57       JO(I-(DIA(L)-1))=1
              JO(I)=1
*   FILL DIAGONALEN

          IF (BYTE(5).NE.9) THEN
*       EIGENTLICH (BYTE(5).NE.2) FUER VOLLE MATRIZEN
              DO 58,L=AZA(I),AZA(I+1)-1
58       JO(ASN(L))=1
*       A EINTRAEGE
          END IF

          SJ=0
          DO 59,L=1,N1
              BSN(BZA(I)+SJ)=L
59       SJ=SJ+JO(L)

60   BZA(I+1)=BZA(I)+SJ

99   END
*****
      SUBROUTINE ZERL(N1,N2,NSA,AZA,ASN,AEI,BYTE,BZA,BSN,BEI,
*ZZA,ZSN,ZEI,DPRO,XH1,XH2,XH3,XH4,DIAB,DIAZ)
* (UNVOLLSTAENDIGE) GAUSS-ZERLEGUNG DER MATRIX A, ERGEBNIS IN B
      IMPLICIT LOGICAL (A-Z)
      INTEGER N1,N2,NSA,I,J,K,L,ZS,SPAL,KL
      INTEGER AZA(N1+1),ASN(NSA),BZA(N1+1),BSN(*)
      INTEGER ZZA(N1+1),ZSN(*),BYTE(*),DIAB(*),DIAZ(*)
      REAL AEI(NSA),BEI(*),RE1,RE2,ZEI(*)
      REAL XH3(N1),XH2(N1),XH1(N1),XH4(N1)
      DOUBLE PRECISION DPRO(N1)
      EXTERNAL SPAL

      BYTE(8)=1
      PRINT*
      ZS=0
      IF (BYTE(5).EQ.3) GOTO 20
* BEI GANZEM ENVELOPE KEINE ADDITION ZUR DIAG. MOEGLICH
15   PRINT*, 'ZERLEGUNG MIT ADDITION IN DER DIAGONALEN: 00001'
      PRINT*, 'OHNE SELBIGE ADD, ODER EINFACHE GAUSS: 00000000'
      READ(*,*) ZS
      IF ((ZS.LT.0).OR.(ZS.GT.1)) THEN
          PRINT*, 'NUR 0 ODER 1'

```

```

        GOTO 15
    END IF
    WRITE(25,*)
    IF (ZS.EQ.0) WRITE(25,*) 'OHNE ADDITION IN DER DIAGONALEN'
    IF (ZS.EQ.1) WRITE(25,*) 'MIT ADDITION IN DER DIAGONALEN'

20    DO 24, I=1, N1
        DO 24, K=BZA(I), BZA(I+1)-1
            J=BSN(K)
            L=SPAL(AZA, ASN, I, J)
            IF (L.EQ.0) THEN
                BEI(K)=0
            ELSE
                BEI(K)=AEI(L)
            END IF
24    CONTINUE

* ZERLEGUNG AX/BARK 41
    DO 38, L=1, N1
        K=SPAL(BZA, BSN, L, L)
        RE1=BEI(K)
        DO 38, I=L+1, N1
            K=SPAL(BZA, BSN, I, L)
            IF (K.GT.0) THEN
                RE2=BEI(K)/RE1
                BEI(K)=RE2
                DO 36, J=L+1, BSN(BZA(I+1)-1)
                    KL=SPAL(BZA, BSN, L, J)
                    IF (KL.GT.0) THEN
                        K=SPAL(BZA, BSN, I, J)
                        IF (K.GT.0) THEN
                            BEI(K)=BEI(K)-RE2*BEI(KL)
                        ELSE
                            K=SPAL(BZA, BSN, I, I)
                            BEI(K)=BEI(K)-ZS*RE2*BEI(KL)
                        END IF
                    END IF
                END IF
36        CONTINUE
            END IF
38    CONTINUE
    DO 41, I=1, N1
        DIAB(I)=SPAL(BZA, BSN, I, I)
        IF (BYTE(5).EQ.3) THEN
            DO 39, I=1, N1+1
                ZZA(I)=BZA(I)
            DO 40, I=1, BZA(N1+1)-1
                ZSN(I)=BSN(I)
                ZEI(I)=BEI(I)
            DO 42, I=1, N1
                DIAZ(I)=DIAB(I)
                BYTE(6)=1
            END IF

```

```

* OBERHALB DER DIAG STEHT DER FAKTOR U (RECHTS)
* UNTERHALB STEH L OHNE 1-DIAGONALE
  IF ((BYTE(6).EQ.1).AND.(BYTE(5).NE.3)) THEN
* FALLS VOLLSTAENDIGE ZERLEGUNG VORHANDEN
  BYTE(9)=1
  BYTE(7)=0
  CALL VEIN(N1,N2,BZA,BSN,BEI,DPRO,XH1,XH2,ZZA,ZSN,
*           ZEI,XH3,XH4,BYTE,DIAB,DIAZ)
  BYTE(7)=1
  CALL VEIN(N1,N2,ZZA,ZSN,ZEI,DPRO,XH1,XH2,BZA,BSN,
*           BEI,XH3,XH4,BYTE,DIAB,DIAB)
* MIT ADDITION (ZS=1) IST DER KL. EW. =1
  END IF
99  END
*****
SUBROUTINE INVIT(N1,N2,BZA,BSN,BEI,DX,XH1,XH2,XH3,BYTE,DIAZ)
  IMPLICIT LOGICAL (A-Z)
* INVERSITERATION ZUR BESTIMMUNG DES KLEINSTEN EW, SCHWARZ 180F
  INTEGER N1,N2,I,J,NJO,K
  INTEGER BZA(N1+1),BSN(N2),BYTE(*),DIAZ(*)
  REAL BEI(N2),XH1(N1),XH2(N1),RAY,RAYV,RAY5,XH3(N1)
  DOUBLE PRECISION SKPR,SP,DX(N1)
  EXTERNAL SKPR

  IF (BYTE(6).EQ.0) THEN
    PRINT*, 'KEINE VOLLST. ZERLEGUNG VON A VORHANDEN'
    RETURN
  END IF
  CALL DINI(N1,1,DX)
  CALL RINI(N1,1,XH1)
  CALL RINI(N1,1,XH2)
  CALL RINI(N1,1,XH3)
  J=0
  RAY=1
  DO 11,I=1,N1
11   XH1(I)=SIN(I*6.1/N1)

  BYTE(9)=1
  CALL DRGL(N1,N2,BZA,BSN,BEI,DX,XH1,XH2,BYTE,DIAZ)
* XH2:=A^-1*XH1
15   J=J+1
  RAYV=RAY
  SP=SKPR(XH2,XH2,N1)
  SP=SQRT(SP)
  IF (SP.EQ.0) STOP 'NULLVEKTOR, STOP!'
  DO 17,I=1,N1
17   XH1(I)=XH2(I)/SP

  BYTE(9)=1
  CALL DRGL(N1,N2,BZA,BSN,BEI,DX,XH1,XH2,BYTE,DIAZ)
* XH1=X(K+1), XH2=A^-1*X(K+1)

```

```

      RAY=SKPR(XH2 ,XH1 ,N1)
* RAYLEIGHQUOTIENT DES EINHEITSVEKTORS X(K+1)
  IF (J.LE.5) THEN
    RAY5=RAY
    GOTO 15
  END IF
  IF (ABS(RAYV-RAY).GT.1.E-4*RAY5) GOTO 15
  PRINT*, 'EIGENVEKTOR_'
  IF (BYTE(1).EQ.1) CALL PRIVEC(XH1 ,N1)
  PRINT*, 'LAMBDA1_=',1/RAY, 'IT=',J
  WRITE(25,*) 'LAMBDA-1_=',1/RAY, 'IT=',J
  WRITE(25,*)
  END
*****
      SUBROUTINE DRGL(N1 ,N2 ,BZA ,BSN ,BEI ,DX ,KLB ,ERG ,BYTE ,DIAG)
* AUFLOESEN EINES DOPPELTEN DREIECKSSYSTEMS (L,R)*ERG=KLB
* BEI AUFRUF ZU UEBERGEHEN: KLB=VEKB ,DX=DPRO
* MATRIX B BESTEHT AUS (L,R), WOBEI DIE 1-DIAGONALE VON L FEHLT
  INTEGER N1 ,N2 ,I ,J ,K ,S ,BYTE(*)
  INTEGER BZA(N1+1) ,BSN(*) ,DIAG(N1)
  REAL BEI(*) ,KLB(N1) ,ERG(N1) ,XH
  DOUBLE PRECISION SKPR ,DX(N1) ,DH
  EXTERNAL SKPR

  IF ((BYTE(6).EQ.0).AND.(BYTE(9).NE.2)) THEN
    PRINT*, 'F.DRGL_IST_VOLLST._ZERL._NOETIG!!!_ZURUECK!!!'
    RETURN
  END IF
  CALL DINI(N1 ,1 ,DX)
  DX(1)=KLB(1)
  DO 15 ,I=2 ,N1
    S=DIAG(I)-1
    DH=0
    DO 13 ,J=BZA(I) ,S
      XH=DX(BSN(J))
13      DH=DH+DPROD(BEI(J) ,XH)
15      DX(I)=KLB(I)-DH
* WG 1 IN DER DIAG VON L
* L*DX=KLB WAR DAS , JETZT R*ERG=DX

  DO 18 ,I=N1 ,1 ,-1
    S=DIAG(I)+1
    XH=BEI(S-1)
    IF (ABS(XH).LT.1.E-30) THEN
      PRINT*, 'DIAGONALEL_',I, '_=',XH, 'AB_MARKE_15_DRGL'
      PRINT*, 'WENN_ABBRUCH ,_DANN_O_EINGEBEN'
      READ(*,*) K
      IF (K.EQ.0) STOP 'ABBRUCH!_STOP!'
    END IF
    DH=0
    DO 16 ,J=S ,BZA(I+1)-1

```

```

16      DH=DH+DPROD(BEI(J),ERG(BSN(J)))
18      ERG(I)=(DX(I)-DH)/XH
99      END
*****
      SUBROUTINE MVLU(N1,N2,ZA,SN,EI,XH,X,ERG,DIAG)
      IMPLICIT LOGICAL(A-Z)
*   MULT. L*U*X DER LU-ZERLEGTEN MATR. IN B-FORM
      INTEGER N1,N2,I,J,K
      INTEGER ZA(N1+1),SN(*),DIAG(*)
      REAL EI(*),XH(N1),X(N1),ERG(N1),SUM

      DO 16, I=1,N1
        SUM=0
        K=DIAG(I)
        DO 15, J=K,ZA(I+1)-1
15          SUM=SUM+EI(J)*X(SN(J))
16          XH(I)=SUM

        ERG(I)=XH(I)
        DO 26, I=2,N1
          K=DIAG(I)-1
          SUM=XH(I)
          DO 25, J=ZA(I),K
25            SUM=SUM+EI(J)*XH(SN(J))
26            ERG(I)=SUM
99      END
*****72*
      SUBROUTINE VEIN(N1,N2,BZA,BSN,BEI,DX,X1,X2,ZZA,ZSN,ZEI,X3,
*   X4,BYTE,DIAFB,DIAFZ)
      IMPLICIT LOGICAL(A-Z)
*   VEKTOR- UND INVERSITERATION FUER C^-1 A
      INTEGER N1,I,J,KZ,N2
      INTEGER BZA(N1+1),BSN(*),ZZA(N1+1),ZSN(*),BYTE(*),DIAFB(*)
      INTEGER DIAFZ(*)
      REAL BEI(*),X1(N1),X2(N1),RAY,RAYV,RAY5,X3(N1),ZEI(*),X4(N1)
      DOUBLE PRECISION SKPR,SP,DX(N1)
      EXTERNAL SKPR

      CALL RINI(N1,1,X1)
      CALL RINI(N1,1,X2)
      CALL RINI(N1,1,X3)

      KZ=BYTE(7)
      J=0
      RAY=1
      DO 11, I=1,N1
11      X1(I)=SIN(I*6.1/N1)

15      J=J+1
      RAYV=RAY

```

```

      SP=SKPR(X1,X1,N1)
      SP=SQRT(SP)
      DO 16,I=1,N1
16      X1(I)=X1(I)/SP

      CALL MVLU(N1,N2,ZZA,ZSN,ZEI,X4,X1,X2,DIAFZ)
      CALL MVLU(N1,N2,BZA,BSN,BEI,X4,X1,X3,DIAFB)

      RAY=SKPR(X1,X2,N1)/SKPR(X1,X3,N1)
      IF (ABS(RAYV-RAY).LT.1.E-4) GOTO 30
      CALL DINI(N1,1,DX)
      CALL DRGL(N1,N2,BZA,BSN,BEI,DX,X2,X1,BYTE,DIAFB)
      GOTO 15

30     CONTINUE
      IF (KZ.EQ.0) THEN
          PRINT*, 'LAMBDA-N□=□',RAY,'□□IT□=□',J
          WRITE(25,*) '□□□GRO.□EW.□C-1A□',RAY,'□□□IT□=□',J
      ELSE
          PRINT*, 'LAMBDA-1□=□',1/RAY,'□□IT□=□',J
          WRITE(25,*) '□□□KLE.□EW.□C-1A□',1/RAY,'□□□IT□=□',J
      END IF
99     END
*****
      SUBROUTINE PRIVEC(V,N)
* VON GUENTER
      INTEGER N,J1,B1,INDEX,J2
      REAL V(N)

      WRITE(*,*)
      WRITE(25,*)
      B1=6
      J1=N/B1
      J2=J1*B1
      INDEX=0
      DO 20 J=1,J1
          WRITE(*,9999) (V(INDEX+K),K=1,B1)
          WRITE(25,9999) (V(INDEX+K),K=1,B1)
          INDEX=INDEX+B1
20     CONTINUE
      IF(N.GT.J2) THEN
          WRITE(*,9999) (V(INDEX+K),K=1,N-J2)
          WRITE(25,9999) (V(INDEX+K),K=1,N-J2)
          INDEX=INDEX+N-J2
      ENDIF
10     CONTINUE
9999  FORMAT(10(E12.5E2:'□'))
      END

```

Dokumentation

Das Programm ist interaktiv aufgebaut. Vor jeder Eingabe erscheint auf dem Bildschirm ein "Menu", aus dem die Eingabe zu wählen ist, oder es wird beschrieben, für welche Größe ein Zahlenwert einzugeben ist.

Vor Übersetzung und Start des Programmes müssen die drei Parameter DN , DSN und $N2$ in Zeile 6 geschrieben werden. Diese Werte sind durch Vorversuche zu ermitteln oder von vornherein so groß anzugeben, daß der damit reservierte Speicherplatz ausreicht (auch für mehrere Programmläufe mit unterschiedlichen Gleichungssystemen; dies spart dann Übersetzungszeit).

Es bedeuten:

DN Obergrenze der Dimension des Systems. Mit dieser Größe werden alle Vektoren und die Felder AZA , BZA und ZZA dimensioniert. Zur tatsächlichen Dimension des Systems s.u.

DSN Obergrenze der Anzahl der Nichtnulleinträge in A . Hiermit werden ASN und AEI dimensioniert.

$N2$ Obergrenze der Nichtnulleinträge, die bei der *vollständigen* Zerlegung von A auftreten. $N2$ ist dann für die volle Bandbreite von A zu bestimmen. Wird auf eine vollständige Zerlegung von A verzichtet (d.h. die Bestimmung von λ_1 und des größten und kleinsten Eigenwertes von $C^{-1}A$ ist nicht möglich), dann kann $N2$ entsprechend kleiner gewählt werden. Wird sogar kein zusätzliches fill-in zugelassen, dann kann $N2 = DSN$ gewählt werden.

Die Dimension $N1$ des Systems, die Anzahl NSA der Nichtnulleinträge von A und die Koeffizientenmatrix A in der zeilenorientierten Form AZA , ASN , AEI müssen in dieser Reihenfolge und geordnet (wie weiter vorne im Abschnitt Programmbeschreibung beschrieben) auf dem File mit dem Namen $MATR$ zur Verfügung stehen und werden dann mit Programmstart im Unterprogramm MAEI eingelesen. Dabei müssen $N1$ und NSA , AZA , ASN , AEI jeweils in einer Eingabezeile stehen. Mittels hier nicht dokumentierter Hilfsprogramme kann das File $MATR$ erzeugt werden. Ebenfalls durch MAEI wird die "rechte Seite" des Gleichungssystems vom File $VEKB$ eingelesen.

Die Ergebnisse werden sowohl auf den Bildschirm ausgegeben als auch auf das File $NOTIZ$. Dieses wird vom Programm nicht 'rewindet'; die Ergebnisse jedes neuen Programmdurchlaufes werden also hintereinander auf dieses File geschrieben.

Falls das Programm nicht interaktiv betrieben werden soll, kann das Inputfile mit einem Hilfsprogramm erzeugt werden.

Im "Hauptmenu" (Zeilen 75-83) können '3' und '4' ohne weiteres aufgerufen werden, für '7', '9' und '11' ist vermittels '5' und '6' die vollständige LU-Zerlegung vorher durchzuführen, für '8' ist zuerst ebenfalls mit '5' und '6' eine LU-Zerlegung durchzuführen.

Zu den wesentlichen Unterprogrammen:

KOGR ist der CG- bzw. PCG-Algorithmus, je nach Wert des Parameters K . Wird im "Hauptmenu" 'konjugierte Gradienten' aufgerufen, dann wird damit $K = 1$ gesetzt, für 'vorkond. CG' wird $K = 2$ gesetzt. Die relative Fehlerschranke eps ist mit 10^{-6} gesetzt, sie kann aber auch eingegeben werden (Entfernen des Kommentarszeichens in Zeile 176). Die Matrix A und die Zerlegung LU werden in zeilenorientierter Form $.ZA, .SN, .EI$ übergeben.

MAVE ist die Matrix-Vektor-Multiplikation für Matrizen in der zeilenorientierten Form. Das Unterprogramm wird nur intern aufgerufen.

POIT ist die einfache Vektoriteration (Poweriteration) zur Bestimmung des größten Eigenwertes von A .

JOTT bestimmt die Indexmenge J , für die bei der Zerlegung Nichtnulleinträge zugelassen werden. Damit werden auch die Felder BZA und BSN der LU-Zerlegung besetzt (Marke 55-60). Sollen Systeme gerechnet werden, bei denen die Besetzung von LU nicht mindestens die von A ist, dann ist in Zeile 431 die 9 durch eine 2 zu ersetzen. Dies bewirkt, daß für Nichtnulleinträge nur die Hauptdiagonale zugelassen wird, soweit nicht explizit noch Nebendiagonalen eingegeben werden.

ZERL führt die Zerlegung durch, d.h. es werden die Einträge BEI der Faktoren L und U bestimmt. Falls in JOTT die vollständige Zerlegung gewählt wurde, wird die Abfrage 'mit - ohne Addition in der Diagonalen' hier übergangen, da in dem Fall ja nichts zu addieren ist. Weiterhin werden in diesem Fall BZA, BSN, BEI nach ZZA, ZSN, ZEI kopiert. In $Z..$ steht also immer die vollständige Zerlegung in zeilenorientierter Form. Sie wird für die Bestimmung des größten und kleinsten Eigenwertes von $C^{-1}A$ durch das Unterprogramm VEIN, welches am Ende von ZERL aufgerufen wird, benötigt.

INVIT ist die einfache Inversiteration zur Bestimmung des kleinsten Eigenwertes von A .

DRGL löst ein lineares Gleichungssystem, dessen Koeffizientenmatrix schon vermittels JOTT und ZERL zerlegt ist. Das Unterprogramm wird von mehreren Stellen aus aufgerufen; es kann aber auch im "Hauptmenu" mit '9' zur direkten Bestimmung der Lösung aufgerufen werden.

MVLU ist die Matrix-Vektor-Multiplikation für Matrizen, die mit ZERL zerlegt sind. Diese Zerlegung liefert ja nicht das Produkt LU , sondern nur die beiden Faktoren. Der Unterschied ist wesentlich für die unvollständige Zerlegung. Das Unterprogramm wird nur von VEIN benötigt.

VEIN ist die Vektor- bzw. Inversiteration für $C^{-1}A$, je nach dem, welche Matrizen für $B..$ und $Z..$ übergeben werden. $C = LU$ und A werden dabei immer in der nach ZERL zerlegten Form übergeben. Daher kann für die Matrix-Vektor-Multiplikation einheitlich MVLU benutzt werden ohne Fallunterscheidung nach zu bestimmendem größten oder kleinsten Eigenwert von $C^{-1}A$. Diese Unterscheidung erfolgt außer

durch die Zuordnung der Matrizen beim Aufruf dadurch, daß im ersten Fall der Rayleighquotient, im zweiten dessen Kehrwert als Ergebnis genommen wird.

Das Programm ist optimierbar, indem z.B. auf DOUBLE PRECISION verzichtet wird, die Ausgaben sparsamer gestaltet werden sowie die Symmetrie der Koeffizientenmatrix berücksichtigt wird.

9 Literaturverzeichnis

- AXELSSON, O., BARKER, V.A.:** Finite Element Solution of Boundary Value Problems, New York: Acad. Pr. 1984
- BRONSTEIN, I.N., SEMENDJAJEW, K.A.:** Taschenbuch der Mathematik, Leipzig: Teubner 1979
- BUNSE, W., BUNSE-GERSTNER, A.:** Numerische Lineare Algebra, Stuttgart: Teubner 1984
- COLLATZ, L.:** Funktionalanalysis und numerische Mathematik, Berlin: Springer 1964
- DORNSCHEIDT, G., SCHENDEL, U.:** Lösung einer elliptischen Randwert-aufgabe mit dem Mehrgitterverfahren, Freie Univ. Berlin 1988
- ERWE, F.:** Differential- und Integralrechnung, Band 1, Mannheim: Bibliogr. Inst. 1962
- EVANS, D.J. (Hrsg.):** Preconditioning Methods, Theory and Applications, New York: Gordon and Breach Science Publ. 1983
- EVANS, D.J.:** The Use of Pre-conditioning in Iterative Methods for Solving Linear Equations with Symmetric Positive Definite Matrices, J. Inst. Maths. Applics. 4(1967):295-314
- GREENBAUM, A.:** Comparison of Splittings Used with the Conjugate Gradient Algorithm, Numer. Math. 33(1979):181-194
- GROTEMEYER, K.P.:** Analytische Geometrie, Berlin: de Gruyter 1969
- GUSTAFSSON, I.:** A Class of First Order Factorization Methods, BIT 18(1978): 142-156
- HACKBUSCH, W.:** Theorie und Numerik elliptischer Differentialgleichungen, Stuttgart: Teubner 1986
- HESTENES, M.:** Conjugate Direction Methods in Optimization, New York: Springer 1980
- HESTENES, M.R.:** The Conjugate-Gradient Method For Solving Linear Systems, Proc. Sympos. Appl. Maths. VI, Numerical Analysis:83-102 New York: Mc Graw-Hill 1956
- KERSHAW, D.S.:** The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations, J. Comp. Phys. 26(1978): 43-65

- MEIJERINK, J.A., VAN DER VORST, H.A.:** An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix, Math. Comp. 31(1977):148-162
- NICOLAIDES, R.A., CHOUDHURY, S.:** Iterative Methods for Elliptic Finite Element Equations on General Meshes, ICASE Report No. 86-78, Hampton: NASA Langley Research Center 1986
- REID, J.K. (Hrsg):** Large Sparse Sets of Linear Equations, London: Acad. Pr. 1971
- SCHENDEL, U.:** Einführung in die numerische Mathematik, Freie Univ. Berlin 1984
- SCHENDEL, U.:** Introduction to Numerical Methods for Parallel Computers, New York: John Wiley & Sons 1984
- SCHENDEL, U.:** Sparse-Matrizen, München: Oldenbourg 1977
- SCHWARZ, H.R.:** Methode der finiten Elemente, Stuttgart: Teubner 1984
- SCHWARZ, H.R., RUTISHAUSER, H., STIEFEL, E.:** Numerik symmetrischer Matrizen, Stuttgart: Teubner 1972
- STOER, J.:** Einführung in die Numerische Mathematik I, Berlin: Springer 1972
- SOUTHWELL, R.V.:** Relaxation Methods in Engineering Science, London: Oxford Univ. Pr. 1940
- VARGA, R.S.:** Matrix Iterative Analysis, Englewood Cliffs: Prentice-Hall 1962
- YOUNG, D.M.:** Iterative Solution of Large Linear Systems, New York: Acad. Pr. 1971
- ZURMÜHL, R.:** Matrizen und ihre technischen Anwendungen, Berlin: Springer 1964

Danksagung

Ich danke ganz herzlich

- Herrn Prof. Dr. U. Schendel für die Themenstellung, die fachliche Betreuung und wertvollen Hinweise,
- Herrn G. Dornscheidt für die vielen konstruktiven Diskussionen, zahlreichen Anregungen und die Zurverfügungstellung von Beispielen.

My eyes collide head-on with stuffed graveyards, false Gods, I scuff
at pettiness which plays so rough, walk upside-down inside handcuffs,
kick my legs to crash it off, say okay, I've had enough, what else can you
show me?

(Dylan)