Numerische Verfahren

zur

geodätischen Optimierung

Wolf-Dieter Schuh

Institut für Theoretische Geodäsie
Friedrich-Wilhelms-Universität Bonn

Stand von 16. Oktober 2003

Vorwort

Ein Hauptanwendungsgebiet der linearen Algebra in der Geodäsie bildet die Ausgleichsrechnung bzw. mathematische Statistik. Die auftretenden Systeme haben vielfach sehr gutmütige und stabile Eigenschaften. Ein Großteil der Berechnungen konzentriert sich auf positiv definite, symmetrische Systeme, die durch die Assemblierung der Normalgleichungen zustande kommen oder von Kovarianzmatrizen stammen. Diese meist statischen Systeme weisen oft eine beträchtliche Größe auf, wobei das Auftreten von Nullelementen (dünne Besetztheit) oder Regelmäßigkeit die Berechnungen erleichtern oder auch erst ermöglichen. Aber auch dynamische Systeme treten bei Optimierungsverfahren, Filtertechniken und robusten statistischen Methoden auf und müssen daher in dieser Zusammenstellung Berücksichtigung finden. In dieser Arbeit wird versucht moderne Rechenverfahren zu erschließen, das Prinzip zu durchleuchten und die Scheu der Verwendung zu nehmen. Viele der dargestellten Verfahren sind vielfach wohlbekannt und weitverbreitet. Manche Berechnungsmethoden stellen sich aus der Sicht eines Geodäten in einem anderen Licht dar, da spezielle Situationen eintreten, die für andere Anwendungen nur geringe Bedeutung haben. Sich daraus ergebende Konsequenzen müssen durchdacht und eventuell auftretende Vorteile genützt werden.

Vielfach ergeben sich auf Grund der Aufgabenstellung geometrische Zusammenhänge, die sowohl für Kontrollzwecke als auch Effizienzsteigerungen und Plausibilitätskontrollen verwendbar sind. Oft können vorweg (a priori) Informationen zur Kontrolle und Genauigkeitsabschätzung Verwendung finden. In manchen Situationen vor allem bei hochpräzisen Arbeiten läßt sich eine bestmögliche rechengestützte Fehlerabschätzung mit Hilfe statistischer Methoden nicht umgehen. Die dazu benötigten Glieder der Inversen beschränken sich auf wenige Stück. Meist ist die Hauptdiagonale mit einigen nahe der Hauptdiagonale liegenden Gliedern und wenigen weiter entfernten Gliedern erforderlich. In den wenigsten Fällen rechtfertigt sich dafür die Berechnung und auch Speicherung der gesamten Inversen. Notwendig wird die Inversenberechnung bei dynamischen Systemen, wo durch nichtvorhersagbares Hinzufügen und Wegnehmen von Unbekannten oder Unbekanntengruppen das System oftmals verändert werden muß. Dieser speziellen Situation ist der erste Abschnitt gewidmet. Der zweite Abschnitt beschäftigt sich mit Auflösungstechniken von symmetrischen Systemen, der im dritten Abschnitt weiter spezialisiert wird auf Systeme vom Type $\boldsymbol{A}^T\boldsymbol{A}$. Besonderer Augenmerk wird im folgenden Abschnitt dünnbesetzten symmetrische Systeme gewidmet, da viele geodätische Anwendungen mit Hilfe dieser Techniken sehr effizient zu bearbeiten sind. Im fünften Abschnitt stehen die Strategien zur Lösung von großen und sehr großen Systemen im Vordergrund. Durch die Automatisation der Meßtechnik werden immer

größere Datenmengen verfügbar, die in einem *„Guß“* behandelt werden. Bedingt durch feste Meßintervalle treten dabei vielfach regelmäßige Strukturen auf. Die numerische Behandlung von Systemen mit regelmäßigen Strukturen erfolgte im sechsten Abschnitt.

Grundelemente des Pseudocodes

Die Leistungsfähigkeit der Berechnungen werden mit Hilfe des benötigten Speicherplatzes und der Anzahl und der Ordnung der Rechenoperationen abgeschätzt.

Der Begriff der Operationen soll hier kurz etwas näher erläutert werden. Unter einer Rechenoperation wird im Weiteren immer eine Multiplikation (oder Division) gefolgt von einer Addition (oder Subtraktion) betrachtet, da diese Folge bei Berechnungen der linearen Algebra häufig zusammen auftreten. Bei allen nachfolgenden Betrachtungen über die Anzahl der Rechenoperationen wird nicht so sehr auf die genaue Anzahl wertgelegt, sondern es soll nur eine Größenordnung abgeschätzt werden. Von Bedeutung ist das Wachstum (Ordnung) mit der die Anzahl der Rechenoperationen steigt. Eine wesentliche Komponente bei Behandlung von großen Datenmengen stellt die Speicherung der Daten dar. In dieser Darstellung wird in der Regel von einer internen Speicherung im Arbeitsspeicher ausgegangen. Bei externer Speicherung stellt die Anzahl der Zugriffe und Blockwechsel sicher eine beachtenwerte Größe dar. Bedingt durch virtuelle Adressierung können Rechenanlagen beträchtliche Datenmengen *„scheinbar“* intern verwalten. Bei der physikalischen Realisierung stößt man wieder auf die Zugriffe auf externe Dateien. Als Grundprinzip hat bei der internen Speicherung daher zu gelten, daß es der Rechnerarchitektur sehr entgegenkommt, wenn man benachbarte Speicheradressen benützt. Große Sprünge innerhalb des Speichers verursachen einen Mehraufwand, der sich in längeren Rechenzeit bemerkbar macht, und sollten daher vermieden werden. Bei Berechnungen in FORTRAN ist darauf zu achten, daß die Speicherung einer Matrix beginnend mit dem ersten Index erfolgt. Im zweidimensionalen Fall ergibt sich damit eine spaltenweise Speicherung. Verwendet man eine Matrix zeilenweise, so ist jeweils eine gesamte (dimensionierte) Spalte zu überspringen um zum nächsten Element zu kommen. Der spaltenweise Zugriff ist hingegen unmittelbar benachbart. Prinzipiell kann zur Programmierung von Algorithmen angemerkt werden, daß versucht werden soll, ein möglichst dem logischen Ablauf entsprechendes Programm zu erzeugen. In vielen Fällen kann besonders *„raffiniertes“* Programmieren von den optimierenden Compilern nicht aufgelöst werden, wodurch längere Rechenzeiten entstehen. All diese Bemerkungen veranschaulichen, daß die Anzahl der Operationen nicht immer signifikant sein muß, wohl aber die Ordnung mit der der Speicherplatz und die Rechenzeit mit zunehmender Datenmenge wächst.

Zeichenerklärung

Zeichen	Definition	Beschreibung
$\mathrm{int}(a)$		größter ganzzahliger Teil von a
$a \bmod b$	$r = a - \mathrm{int}\left(\frac{a}{b}\right) b$	die Modulofunktion gibt den Divisionsrest r von $\frac{a}{b}$ an, mit $b \neq 0$.

16. Oktober 2003

Wolf-Dieter Schuh
ITGB, Uni Bonn

Inhaltsverzeichnis

Kapitel 1

Dynamische Gleichungssysteme

Bei den meisten Darstellungen der numerischen Techniken der linearen Algebra wird von statischen Gleichungssystemen ausgegangen. Dies bedeutet, daß vor dem Start des Lösungsvorganges das das Gleichungssystem sowohl in seiner Gestalt als auch in seinem Inhalt festgelegt ist. Nach der Berechnung werden die Ergebnisse analysiert und allenfalls ein weiterer Lösungsschritt mit demselben Gleichungssystem berechnet. Bei der Abänderung der Problemstellung und damit der Änderung des Gleichungssystems erfolgt zumeist eine vollkommene Neuberechnung der Lösung des abgeänderten Systems ohne auf die vorhergehende Berechnung zurückzugreifen. Die Strategien zeichnen sich zumeist durch gute Strukturierung und große Flexibilität aus. Die Effizienz bleibt dabei zumeist auf der Strecke.

Oftmals bereiten dynamische Veränderungen während des Rechenprozesses große Probleme. Die effiziente Meisterung dieser Aufgaben stellt meist eine hohe Anforderung dar. Dieser Abschnitt behandelt die dazu nötigen grundlegenden Operationen und wird deswegen hier an den Beginn gestellt, da diese Operationen auch bei anderen Methoden der nachfolgenden Abschnitte Verwendung finden. Der Ausdruck dynamisch bezieht sich auf zwei mögliche Änderungen des Gleichungssystems. Einerseits kann sich die Anzahl und Bedeutung der Variablen verändern, wobei das Hinzufügen von neuen Variablen oder Variablengruppen, das Entfernen beziehungsweise der Übergang von einer abhängigen zu einer unabhängigen Variablen (*Austauschschritt*) als Grundoperationen anzusehen sind (Abschnitt 1.1). Eine zweite Möglichkeit der Dynamik ist durch Änderung der Koeffizienten gegeben, wobei strenge Berechnungen bei gravierenden Umwandlungen, die die Struktur des Systems verändern, und näherungsweise Berechnungen bei kleinen Abänderungen verwendet werden (Abschnitt 1.7). Die Anwendung von iterativen Verfahren (Abschnitt 1.8) ermöglicht vielfach eine sehr effiziente Lösung von dynamischen Systemen. Da der Übergang von abhängigen zu unabhängigen Variablen (Austauschschritt) eine Grundoperation bei vielen Auflösungs- und Iterationstechniken darstellt, soll mit dieser Operation begonnen werden.

1.1 Austauschschritt

Betrachtet man ein System von abhängigen und unabhängigen Variablen

$$\begin{aligned} y_1 &= f_1(x_1, x_2, x_3, x_4) \\ y_2 &= f_2(x_1, x_2, x_3, x_4) \\ y_3 &= f_3(x_1, x_2, x_3, x_4) \ , \end{aligned} \tag{1.1}$$

wobei x_1, x_2, x_3 und x_4 die **unabhängigen** und y_1, y_2 und y_3 die **abhängigen** Variablen darstellen. Die hier behandelten Funktionen f weisen linearen Charakter auf, wodurch die Beziehungen durch Faktoren $a_{11}, \ \ldots \ , a_{34}$ dargestellt werden können

$$\begin{aligned} y_1 &= a_{11}\ x_1 + a_{12}\ x_2 + a_{13}\ x_3 + a_{14}\ x_4 \\ y_2 &= a_{21}\ x_1 + a_{22}\ x_2 + a_{23}\ x_3 + a_{24}\ x_4 \\ y_3 &= a_{31}\ x_1 + a_{32}\ x_2 + a_{33}\ x_3 + a_{34}\ x_4 \quad . \end{aligned} \tag{1.2}$$

Aufgabenstellung ist es nun, eine abhängige (gebundene) Variable durch eine unabhängige (ungebundene) auszutauschen. Wir versuchen, die abhängige Variable y_2 durch die unabhängige Variable x_3 auszutauschen. Dazu dividiert man die mittlere Zeile (*Pivotzeile*) durch a_{23} (*Pivotelement*) und isoliert x_3 auf der linken Seite

$$x_3 = -\frac{a_{21}}{a_{23}}\ x_1 - \frac{a_{22}}{a_{23}}\ x_2 + \frac{1}{a_{23}}\ y_2 - \frac{a_{24}}{a_{23}}\ x_4 \ . \tag{1.3}$$

Setzt man nun Formel (1.3) in die erste und dritte Zeile von (1.2) ein und ordnen nach den unabhängigen Variablen, so erreicht man

$$\begin{aligned} y_1 = &\left(a_{11} - \tfrac{a_{21}a_{13}}{a_{23}}\right) x_1 + \left(a_{12} - \tfrac{a_{22}a_{13}}{a_{23}}\right) x_2 + \tfrac{a_{13}}{a_{23}}\ y_2 + \left(a_{14} - \tfrac{a_{24}a_{13}}{a_{23}}\right) x_4 \\ x_3 = &- \tfrac{a_{21}}{a_{23}}\ x_1 - \tfrac{a_{22}}{a_{23}}\ x_2 + \tfrac{1}{a_{23}}\ y_2 - \tfrac{a_{24}}{a_{23}}\ x_4 \\ y_3 = &\left(a_{31} - \tfrac{a_{21}a_{33}}{a_{23}}\right) x_1 + \left(a_{32} - \tfrac{a_{22}a_{33}}{a_{23}}\right) x_2 + \tfrac{a_{33}}{a_{23}}\ y_2 + \left(a_{34} - \tfrac{a_{24}a_{33}}{a_{23}}\right) x_4 \end{aligned} \tag{1.4}$$

Die dritte Spalte wird als *Pivotspalte* bezeichnet. Die Regel zur Durchführung eines Austauschschrittes kann in vier Schritten zusammengefaßt werden (siehe *E. Stiefel (1976)[84]*, Seite 13-14):

1. Das Pivotelement geht in seinen reziproken Wert über,
2. Die übrigen Elemente der Pivotspalte sind durch das Pivotelement zu dividieren,
3. Ein Element im Rest der Matrix (Matrix ohne Pivotspalte und Pivotzeile) wird transformiert, indem man von ihm das Produkt subtrahiert, gebildet aus dem entsprechenden Element der neu ermittelten Pivotspalte (Element in der gleichen Zeile) und dem entsprechenden Element der Pivotzeile (Element in der gleichen Spalte) und
4. Die übrigen Elemente der Pivotzeile sind durch das negative Pivotelement zu dividieren.

Algorithmus 1.1 : Austauschschritt; Austausch einer abhängigen Variablen y_i mit einer unabhängigen Variablen x_j.

Aufgabenstellung: Gegeben ist ein Gleichungssystem mit m Zeilen und n Spalten. Gesucht ist das resultierende System, wenn die abhängigen Variable y_i durch die unabhängige Variable x_j getauscht wird.

Anmerkung: Die Berechnung erfolgt *am Platz*, sodaß die eingegebenen Koeffizienten $a_{k\ell}$ durch die modifizierten Koeffizienten $\bar{a}_{k\ell}$ ersetzt werden.

Schnittstellen:

Eingabe:	`a(m,n)`	... Koeffizienten der Matrix $\boldsymbol{A}$
	`m`	... Anzahl der Zeilen des Systems
	`n`	... Anzahl der Spalten des Systems
	`i`	... Pivotzeile
	`j`	... Pivotspalte
Ausgabe:	`a(m,n)`	... modifizierte Koeffizienten der Matrix $\boldsymbol{A}$
Felder:	`a(m,n)`	skalare Version
	`a(m,n), hilf(n)`	vektorielle Version

Ausgangssituation:

	x_1	x_2	...	x_j	...	x_n
y_1	a_{11}	a_{12}	...	a_{1j}	...	a_{1n}
y_2	a_{21}	a_{22}	...	a_{2j}	...	a_{2n}
.	.			.		.
.	.			.		.
.	.			.		.
y_i	a_{i1}	a_{i2}	...	a_{ij}	...	a_{in}
.	.			.		.
.	.			.		.
.	.			.		.
y_m	a_{m1}	a_{m2}	...	a_{mj}	...	a_{mn}

Endsituation:

	x_1	x_2	...	y_i	...	x_n
y_1	$\bar{a}_{11}$	$\bar{a}_{12}$	...	$\bar{a}_{1j}$	...	$\bar{a}_{1n}$
y_2	$\bar{a}_{21}$	$\bar{a}_{22}$	...	$\bar{a}_{2j}$	...	$\bar{a}_{2n}$
.	.			.		.
.	.			.		.
.	.			.		.
x_j	$\bar{a}_{i1}$	$\bar{a}_{i2}$	...	$\bar{a}_{ij}$	...	$\bar{a}_{in}$
.	.			.		.
.	.			.		.
.	.			.		.
y_m	$\bar{a}_{m1}$	$\bar{a}_{m2}$	...	$\bar{a}_{mj}$	...	$\bar{a}_{mn}$

Wolf-Dieter Schuh
ITGB, Uni Bonn

Lösungsweg: Pseudocode (skalar)

1. *Überprüfe das Pivotelement, speichere den Reziprokwert und initialisiere das Element:*
   `IF ABS(a(i,j))≤num_Null GOTO Ende:` 'Austausch nicht möglich'.
   `pivot = 1.0/a(i,j) ; a(i,j) = 1.0 .`
2. *Alle Elemente der Pivotspalte werden durch das Pivotelement dividiert:*
   `DO k=1 TO m ; a(k,j)=pivot*a(k,j) ; END DO k .`
3. *Berechne alle restlichen Elemente spaltenweise:*

```
DO ℓ=1 TO n
  IF ℓ ≠ j THEN
     hilf=a(i,ℓ) ; a(i,ℓ)=0.0
     IF hilf≠rech_Null THEN
       DO k=1 TO m ; a(k,ℓ)=a(k,ℓ)-hilf*a(k,j) ; END DO k.
     END IF
  END IF
END DO ℓ
```

`Ende:` 'o.k. Ende '.

Lösungsweg: Pseudocode (vektoriell)

1. *Überprüfe das Pivotelement, speichere den Reziprokwert und initialisiere das Element:*
   `IF ABS(a(i,j))≤num_Null GOTO Ende:` 'Austausch nicht möglich'.
   `pivot = 1.0/a(i,j) ; a(i,j) = 1.0 .`
2. *Alle Elemente der Pivotspalte werden durch das Pivotelement dividiert:*
   `a(1:m,j)=pivot*a(1:m,j) .`
3. *Berechne alle restlichen Elemente blockweise:*

```
hilf(1:j-1)=a(i,1:j-1) ; a(i,1:j-1)=0.0
a(1:m,1:j-1)=a(1:m,1:j-1)-a(1:m,j)*hilf(1:j-1)
hilf(j+1:n)=a(i,j+1:n) : a(i,j+1:n)=0.0
a(1:m,j+1:n)=a(1:m,j+1:j-1)-a(1:m,j)*hilf(j+1:n)
```

`Ende:` 'o.k. Ende '.

Ressourcen: Anzahl der Operationen: mn bzw. 2 $\mathcal{MAT}(\frac{1}{2}mn)$
Platzbedarf: mn

Pro Austauschschritt bei m abhängigen Variablen y_i und n unabhängigen Variablen x_j sind somit nm Operationen erforderlich. Zusätzlicher Speicherplatz für Zwischenspeicherungen wird nur im geringem Ausmaß benötigt. Nachfolgend werden die aufgezeigten Schritte im Algorithmus 1.1 zusammengefaßt, wobei eine skalare und vektorielle Formulierung aufgezeigt ist.

Solange das Element a_{ij} ungleich Null ist, kann mit Hilfe des Algorithmus 1.1 beliebige abhängige Variable y_i durch unabhängige Variable x_j getauscht werden.

Geometrisch bedeutet ein Austauschschritt einen Basiswechsel. Den abhängigen Variablen y_i (*Basisvariablen*) sind die *natürlichen Basisvektoren* $\boldsymbol{e}^{(i)}$ zugeordnet, die nur mit einer Eins im i-ten Element und ansonsten mit Nullen besetzt ist. Diese spannen die *natürliche Basis* im $\mathbb{R}^m$ auf.

$$\begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} y_1 + \begin{bmatrix} 0 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} y_2 + \ \ldots \ + \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} y_m \ = \begin{bmatrix} a_{11} \\ a_{21} \\ \cdot \\ \cdot \\ \cdot \\ a_{m1} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ a_{22} \\ \cdot \\ \cdot \\ \cdot \\ a_{m2} \end{bmatrix} x_2 + \ \ldots \ + \begin{bmatrix} a_{1n} \\ a_{2n} \\ \cdot \\ \cdot \\ \cdot \\ a_{mn} \end{bmatrix} x_n \tag{1.5}$$

In kompakterer Form kann dieser Zusammenhang mit Hilfe der Vektoren $\boldsymbol{e}^{(i)}$ und $\boldsymbol{a}^{(j)}$ dargestellt werden,

$$\boldsymbol{e}^{(1)}y_1 + \boldsymbol{e}^{(2)}y_2 + \ \ldots \ + \boldsymbol{e}^{(2)}y_m = \boldsymbol{a}^{(1)}x_1 + \boldsymbol{a}^{(2)}x_2 + \boldsymbol{a}^{(3)}x_3 + \ \ldots \ + \boldsymbol{a}^{(n)}x_n \ . \tag{1.6}$$

Die Vektoren $\boldsymbol{a}^{(j)}$ zu den unabhängigen Variablen x_j (*Nichtbasisvariablen*) bilden ebenfalls Vektoren im $\mathbb{R}^m$ und werden als *Nichtbasisvektoren* bezeichnet. Ein Austauschritt bedeutet somit einen Wechsel innerhalb der Basis. Der Nichtbasisvektor $\boldsymbol{a}^{(j)}$ wird in die Basis aufgenommen und ersetzt den Basisvektor $\boldsymbol{e}^{(i)}$. Die durch diesen Schritt zu modifiziertenden Vektoren $\bar{\boldsymbol{a}}^{(j)}$ drücken das Verhältnis der Nichtbasisvektoren bezüglich der neuen Basis aus. Formel (1.7) zeigt diesen Basiswechsel bezüglich des Pivotelementes a_{23}. Die getauschten Variablen sind dabei speziel hervorgehoben.

$$\boldsymbol{e}^{(1)}y_1 + \boldsymbol{e}^{(2)}\boxed{x_3} + \ \ldots \ + \boldsymbol{e}^{(m)}y_m = \bar{\boldsymbol{a}}^{(1)}x_1 + \bar{\boldsymbol{a}}^{(2)}x_2 + \bar{\boldsymbol{a}}^{(3)}\boxed{y_2} + \ \ldots \ + \bar{\boldsymbol{a}}^{(n)}x_n \tag{1.7}$$

So können beliebige Basiswechsel durchgeführt werden. Probleme entstehen nur dann, wenn das Pivotelement sehr klein wird oder verschwindet:

$a_{ij} = 0$ Ein Nullelement bei einem der Komponenten a_{ij} bedeutet, daß eine Änderung der unabhängigen Variable x_j die abhängige Variable y_i nicht beeinflußt. Da die Vektoren der gebundenen Variablen eine vollständige Basis im $\mathbb{R}^m$ aufspannen, bedeutet dies, daß die entsprechende ungebundene Variable x_j eine Richtung repräsentiert, die auf die Richtung der gebundenen Variable y_i orthogonal steht. Da ein vollständige Basis aufgespannt wird, impliziert dies, daß die unabhängige Variable x_j linear abhängig von den restlichen abhängigen Variablen y sein muß. Der Austauschschritt kann somit nicht erfolgen, da eine linear abhängige Variable in die Basis aufgenommen würde. Aufgrund des resultierenden Rangdefektes wird die Darstellung alle Spaltenvektoren $\bar{\boldsymbol{a}}^{(j)}$ vereitelt.

$a_{ij} \approx 0$ Der Austausch von zwei Variablen, wobei das entsprechende Pivotelement klein ist, bedeutet die Einführung eines neuen Basisvektors, der beinahe von den anderen Basisvektoren abhängig ist. Das gesamte Basissystem verliert an Stabilität, wodurch numerische Probleme auftreten.

Das Universalwerkzeug bei der Behandlung von linearen Gleichungssystemen

$$\boldsymbol{y} = \boldsymbol{A}\,\boldsymbol{x} \tag{1.8}$$

stellt das *Gauß-Jordan-Verfahren* (siehe z. B. *H. Sünkel (1993)[89]*, Seite 46ff) dar. Durch Ausnutzung der Invarianzeigenschaften der Lösung - genauer Lösungsmannigfaltikeit - gegenüber den Operationen

- Multiplikation einer Gleichung (Zeile) mit einem Faktor $\neq 0$,
- Vertauschung zweier Gleichungen und
- der Linearkombination von zwei Gleichungen

werden die Spalten $\boldsymbol{a}^{(j)}$ der Matrix $\boldsymbol{A}$ sukzessive in eine Einheitsmatrix $\boldsymbol{I}$ verwandelt. Vor der genaueren Analyse verdeutlichen wir uns zunächst der Äquivalenz der Formulierung der Matrizengleichung (1.8) mit der vektoriellen Formulierung (1.6).

Ausgehend von der vektoriellen Formulierung (1.6) kann mit Hilfe der sukzessiven Anwendung der drei Grundoperationen eine beliebige Spalte $\boldsymbol{a}^{(j)}$ in eine Spalten $\boldsymbol{e}^{(i)}$ verwandelt werden (Bedingung: $a_{ij} \neq 0$). Das Ergebnis diese Operation, wenn beispielsweise a_{23} als Pivotelement verwendet wird, ergibt sich mit

$$\boldsymbol{e}^{(1)}y_1 + \boxed{\hat{\boldsymbol{a}}^{(3)}}\, y_2 + \ \ldots\ + \boldsymbol{e}^{(2)}y_m = \hat{\boldsymbol{a}}^{(1)}x_1 + \hat{\boldsymbol{a}}^{(2)}x_2 + \boxed{\boldsymbol{e}^{(2)}}\, x_3 + \ \ldots\ + \hat{\boldsymbol{a}}^{(n)}x_n\ . \tag{1.9}$$

Im Gegensatz zu (1.7) sind hier nicht die Variablen y_2 und x_3 hervorgehoben, sondern die zugeordneten Vektoren, die die Plätze getauscht haben. Es gilt:

$$\hat{\boldsymbol{a}}^{(j)} = \begin{bmatrix} -\frac{a_{1j}}{a_{ij}} \\ -\frac{a_{2j}}{a_{ij}} \\ \cdot \\ \cdot \\ \cdot \\ \frac{1}{a_{ij}} \\ \cdot \\ \cdot \\ \cdot \\ -\frac{a_{mj}}{a_{ij}} \end{bmatrix}, \qquad \begin{matrix} \hat{\boldsymbol{a}}^{(\ell)} \\ \ell = 1,\ldots,n \\ \ell \neq j \end{matrix} = \begin{bmatrix} a_{1\ell} - \frac{a_{i\ell}a_{1j}}{a_{ij}} \\ a_{2\ell} - \frac{a_{i\ell}a_{2j}}{a_{ij}} \\ \cdot \\ \cdot \\ \cdot \\ \frac{a_{i\ell}}{a_{ij}} \\ \cdot \\ \cdot \\ \cdot \\ a_{m\ell} - \frac{a_{i\ell}a_{mj}}{a_{ij}} \end{bmatrix} \tag{1.10}$$

Durch Umordnung der Variablen und einem Vorzeichenwechsel in der j-ten Zeile (Gleichung) ist das gleiche Ergebnis wie beim Austauschschritt (siehe Formel (1.7)) zu erreichen. Der Austauschschritt bildet somit eine platzoptimierte Variante des Gauß-Jordan-Verfahrens.

Die Darstellung als Austauschschritt eignet sich besonders bei dynamischen Systemen, wo ein ständiger Basiswechsel stattfindet. Auch bei Verfahren, wo die Basisvariablen nicht vorweg festgelegt sind, sondern erst erst durch eine iterative Strategie bestimmt

werden, ist ein Einsatzgebiet für den Austauschschritt. Er bildet die Grundoperation des Standardverfahren der linearen Optimierung, des *Simplex-Verfahrens* (siehe *G. Dantzig (1966)[21]*, Seite 110ff, oder *R.E. Burkard (1972)[15]*, Seite 23ff).

Im nachfolgenden Abschnitt wird durch sukzessive Anwendung des Austauschschrittes die Berechnung der Inversen einer regulären, quadratischen Matrix $\boldsymbol{A}$ *am Platz* dargestellt.

Literatur:

[15] BURKARD Rainer E. (1972): Methoden der Ganzzahligen Optimierung. Springer Verlag, Wien-New York, Seite 23ff.

[21] DANTZIG G. B. (1966): Lineare Programmierung und Erweiterungen. Springer Verlag, Berlin-Heidelberg-New York, Seite 110ff.

[84] STIEFEL Eduard (1976): Einführung in die numerische Mathematik. Teubner Studienbücher Mathematik, 5. Auflage, Stuttgart, Seite 13-14.

[89] SÜNKEL Hans (1993): Skriptum aus Ausgleichsrechnung I. Mathematische Geodäsie und Geoinformatik, TU Graz, Seite 47ff.

1.2 Inversion

Zunächst beschränken wir uns auf eine reguläres Gleichungssystem

$$\boldsymbol{y} = \boldsymbol{A}\ \boldsymbol{x} \tag{1.11}$$

mit der quadratischen Matrix $\boldsymbol{A}$ der Dimension n, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. Durch sukzessiven Anwendung von Austauschschritten können die Basisvariablen $\boldsymbol{y}$ durch die Nichtbasisvariablen $\boldsymbol{x}$ getauscht werden, sodaß

$$\boldsymbol{x} = \bar{\boldsymbol{A}}\ \boldsymbol{y} \tag{1.12}$$

gebildet wird. Aus dieser Darstellung ist unmittelbar erkennbar, daß die ermittelte Matrix $\bar{\boldsymbol{A}}$ die *Kehrmatrix* oder *Inverse* $\boldsymbol{A}^{-1}$ der Ausgangsmatrix $\boldsymbol{A}$ darstellt.

Für die Inversion eines eindeutig bestimmten Systems (regulären Systems) kann ein Algorithmus gefunden werden, der die Inversion ohne Verwendung zusätzlichen Speicherplatzes - also *am Platz* - durchführt. Die Auswahl der Pivotelemente erfolgt auf Grund der Größe aller Koeffizienten der unabhängigen (noch nicht getauschten) Variablen. Da sowohl die Zeilen als auch die Spalten nach dem größten Element durchsucht werden, spricht man von einer *vollständigen Pivotsuche*. Diese gewährleistet größtmögliche numerische Stabilität und erlaubt außerdem bei singulären Systemen eine Aussage über den Rang der Matrix. Die vollständige Pivotsuche und die organisatorischen Maßnahmen bedingt durch die gezielte Pivotauswahl verursacht einen erheblicher Mehraufwand.

Liegen zusätzliche Informationen über das Gleichungssystem vor - zum Beispiel starke Dominanz der Diagonale - so kann die Berechnung entweder ohne Pivotsuche durchgeführt werden, oder mit vereinfachter Pivotstrategien gearbeitet werden. Diese Strategien kommen vielfach bei großen Gleichungssystemen zum Einsatz, wo das betragsgrößte Element der nachfolgenden Zeile (*Zeilenpivotsuche*), Spalte (*Spaltenpivotsuche*) oder Blockes als Pivotelement festgelegt wird. Einerseits wird dadurch die Anzahl der Vergleiche reduziert, andererseits muß nicht immer die gesamte Matrix, von der Teile ausgelagert sein können, besucht werden.

Der im folgenden zusammengefaßte Algorithmus - 1.2 - vollführt die Inversion einer quadratischen Matrix $\boldsymbol{A}$ ohne zusätzlichen Speicheraufwand bei vollständiger Pivotierung. Um eine effiziente Berechnung zu ermöglichen wird die Reihenfolge der Spalten und Zeilen während der Berechnung so verändert, daß die Pivotelemente nacheinander in der Diagonale angeordnet werden. Zur Kennzeichnung und Wiederherstellung der ursprünglichen Reihenfolge werden Hilfsvektoren (*Indexvektoren*) benötigt. Zur Überprüfung der Singularität wird jeweils das Verhältnis des aktuellen zum bisher größten Pivotelement herangezogen. Wenn das neue Pivotelement im Verhältnis zum bisher größten Pivotelement numerisch Null ergibt, wird die Berechnung der Inversen abgebrochen, wobei der Rang und die linearen Abhängigkeiten als Ergebnis vorliegen. Alle mit Einsen gekennzeichnet Spalten (Zeilen) bilden eine Gruppe von linear unabhängig Vektoren, während die durch Nullen gekennzeichneten Spalten (Zeilen) von dieser Gruppe linear abhängig sind.

Algorithmus 1.2 : Inversion einer quadratischen Matrix $\boldsymbol{A}$ am Platz mit vollständiger Pivotsuche.

Aufgabenstellung: Inversion einer quadratischen Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.

Anmerkung: Die Berechnung erfolgt *am Platz*, sodaß die eingegebenen Koeffizienten $a_{k\ell}$ durch die modifizierten Koeffizienten $\bar{a}_{k\ell}$ ersetzt werden.

Schnittstellen:

Eingabe: `a(n,n)` ... Koeffizienten der Matrix $\boldsymbol{A}$
`n` ... Anzahl der Zeilen und Spalten des Systems
Ausgabe: `a(n,n)` ... modifizierte Koeffizienten der Matrix $\boldsymbol{A}$
`ind_sp` ... Indexvektor Spalten
`ind_z` ... Indexvektor Zeilen
Felder: `a(n,n),ind_sp(n),ind_z(n)`

Hilfsproceduren:

`i,j = MAXLOC(a)`	... Zeile i und Spalte j des größten Koeffizienten der $n\times n$-Matrix $\boldsymbol{A}$.
`TAUSCHE_SPALTEN(a,i,j)`	... Vertausche die Spalten i und j in der $n\times n$-Matrix $\boldsymbol{A}$.
`TAUSCHE_ZEILEN(a,i,j)`	... Vertausche der Zeilen i und j in der $n\times n$-Matrix $\boldsymbol{A}$.
`AUSTAUSCHSCHRITT(a,,s,s)`	... siehe Algorithmus 1.1.

Ausgangssituation:

	x_1	x_2	...	x_j	...	x_n	`ind_z`
y_1	a_{11}	a_{12}	...	a_{1j}	...	a_{1n}	$\star$
y_2	a_{21}	a_{22}	...	a_{2j}	...	a_{2n}	$\star$
.	.			.		.	.
.	.			.		.	.
.	.			.		.	.
y_i	a_{i1}	a_{i2}	...	a_{ij}	...	a_{in}	$\star$
.	.			.		.	.
.	.			.		.	.
.	.			.		.	.
y_n	a_{n1}	a_{n2}	...	a_{nj}	...	a_{nn}	$\star$
`ind_sp`	$\star$	$\star$	...	$\star$	...	$\star$	

`ind_sp, ind_z` ... INTEGER Hilfsvektoren
($\star$ ... beliebige Werte)

Wolf-Dieter Schuh
ITGB, Uni Bonn

Endsituation: reguläre Matrix, $Rang(\boldsymbol{A}) = n$.

	y_1	y_2	...	y_j	...	y_n	`ind_z`
x_1	$\bar{a}_{11}$	$\bar{a}_{12}$	...	$\bar{a}_{1j}$	...	$\bar{a}_{1n}$	$\star$
x_2	$\bar{a}_{21}$	$\bar{a}_{22}$	...	$\bar{a}_{2j}$	...	$\bar{a}_{2n}$	$\star$
.	.			.		.	.
.	.			.		.	.
.	.			.		.	.
x_i	$\bar{a}_{i1}$	$\bar{a}_{i2}$	...	$\bar{a}_{ij}$	...	$\bar{a}_{in}$	$\star$
.	.			.		.	.
.	.			.		.	.
.	.			.		.	.
x_n	$\bar{a}_{n1}$	$\bar{a}_{n2}$	...	$\bar{a}_{nj}$	...	$\bar{a}_{nn}$	$\star$
`ind_sp`	$\star$	$\star$	...	$\star$	...	$\star$	

Endsituation (beispielhaft)**:** singuläre Matrix, $Rang(\boldsymbol{A}) = 3$.

	y_2	y_4	y_1	x_1	x_5	x_6	`ind_z`
x_3	$\bar{a}_{23}$	$\bar{a}_{24}$	$\bar{a}_{22}$	$\bar{a}_{21}$	$\bar{a}_{25}$	$\bar{a}_{26}$	1
x_4	$\bar{a}_{43}$	$\bar{a}_{44}$	$\bar{a}_{42}$	$\bar{a}_{41}$	$\bar{a}_{45}$	$\bar{a}_{46}$	1
x_2	$\bar{a}_{13}$	$\bar{a}_{14}$	$\bar{a}_{12}$	$\bar{a}_{11}$	$\bar{a}_{15}$	$\bar{a}_{16}$	0
y_3	$\bar{a}_{33}$	$\bar{a}_{34}$	$\bar{a}_{32}$	0	0	0	1
y_5	$\bar{a}_{53}$	$\bar{a}_{54}$	$\bar{a}_{51}$	0	0	0	0
y_6	$\bar{a}_{63}$	$\bar{a}_{64}$	$\bar{a}_{61}$	0	0	0	0
`ind_sp`	0	1	1	1	0	0	

Die linear unabhängigen Spalten (Zeilen) sind durch Einsen im Hilfsvektor `ind_sp` (`ind_z`) erkennbar.

Lösungsweg: Pseudocode

1. *Initialisiere Austauschschleife und das Element zur Abschätzung von num. Null:*
   `s=0; ele_max=0 .`
   Initialisiere die Hilfsvektoren:
   `ind_sp(1:n)=0; ind_z(1:n)=0 .`
2. *Schleife über einen Austauschschritt:* `s=s+1`
   `IF s>n GOTO 10: .`
3. *Suche Position des Pivotelements:*
   `i_local,j_local=MAXLOC(ABS(a(s:n,s:n))`
   `i=i_local+s-1; j=j_local+s-1` *(i Pivotzeile, j Pivotspalte)*
   `IF ABS(a(i,j)) > ele_max THEN ele_max = ABS(a(i,j)) .`

Wolf-Dieter Schuh
ITGB, Uni Bonn

4. *Überprüfe das Pivotelement auf num. Null:*
   `IF ABS(a(i,j))/ele_max < num_Null GOTO` `13:` .
5. *Bringe Pivotspalte j in Spalte s durch Vertauschen:*
   `IF s≠j THEN TAUSCHE_SPALTEN(a(1:n,1:n),j,s)` .
6. *Bringe Pivotzeile i in Zeile s durch Vertauschen:*
   `IF s≠i THEN TAUSCHE_ZEILEN(a(1:n,1:n),i,s)` .
7. *Führe Austauschschritt mit Pivot a_{ss} durch:*
   `AUSTAUSCHSCHRITT(a(1:n,1:n),,s,s)`
8. *Markiere den Inhalt der s-ten Zeile und Spalte:*
   `ind_sp(s)=i; ind_z(s)=j` .
9. *Nächster Austauschschritt:*
   `GOTO` `2:` .
10. *Rücksortierung der Spalten bei vollem Rang:*
    `DO k = n-1 TO 1 STEP -1`
    `j=ind_sp(k)`
    Tausche Spalte j und k:
    `IF k≠j THEN TAUSCHE_SPALTEN(a(1:n,1:n),j,k)`
    `END DO k` .
11. *Rücksortierung der Zeilen:*
    `DO ℓ=n-1 TO 1 STEP -1`
    `i=ind_z(ℓ)`
    `IF ℓ≠i THEN TAUSCHE_ZEILEN(a(1:n,1:n),i,ℓ)`
    `END DO ℓ` .
12. `Ende:` 'O.K. ENDE, VOLLER RANG'.
13. *Rücksortierung der Indexvektoren bei Rangdefekt:*
    `DO k=s-1 TO 1 STEP -1`
    Behandle Spalte:
    `j=ind_sp(k); ind_sp(k)=ind_sp(j); ind_sp(j)=1`
    Behandle Zeile:
    `i=ind_z(k); ind_z(k)=ind_z(i); ind_z(i)=1`
    `END DO k` .
14. `Ende:` 'O.K. ENDE, KEIN VOLLER RANG, RANG = S-1' .

Ressourcen: Anzahl der Operationen: n `AUSTAUSCH` $\Longrightarrow$ n^3
Platzbedarf: n^2

Den Hauptanteil des Rechenaufwandes wird durch die n Austauschschritte hervorgerufen. Da jeder skalare Austauschschritt in n^2 Operationen benötigt, sind n^3 Operationen zur vollständigen Durchführung erforderlich. Bei Anwendung der vollständigen Pivotsuche sind außerdem n^3 Vergleichsoperationen notwendig.

Liegt kein reguläres System vor, so bricht der Algorithmus 1.2 ab und liefert den Rang der Matrix $\boldsymbol{A}$. Außerdem wird eine vollständige Basis sowohl im Zeilenraum ($\mathcal{Z}(\boldsymbol{A})$) als auch im Spaltenraum ($\mathcal{S}(\boldsymbol{A})$) der Matrix $\boldsymbol{A}$ gekennzeichnet. Wegen der vollständigen

Pivotierung bilden die markierten Zeilen(Spalten)vektoren im Sinne der numerischen Stabilität die *bestmögliche* Kombination von linear unabhängigen Vektoren.

Diese Informationen können für eine genaue Analyse der Problemstellung dienlich sein. Durch weiter Modifikationen (siehe Abschnitt 1.5) des Algorithmus kann aber auch die *verallgemeinerte Inverse* $\boldsymbol{A}^{-}$ mit Hilfe des Austauschverfahrens dargestellt werden. Bevor wir diese Modifikationen vornehmen, wollen wir den Austauschschritt noch etwas näher durchleuchten.

1.3 Verallgemeinerte Austauschschritt

Für die weiteren Betrachtungen ist es interessant das Austauschverfahren mit folgender Ausgangssituation (*Ausgangstableau*)

(1.13)

	$\boldsymbol{x}_1$	$\boldsymbol{x}_2$	
$\boldsymbol{y}_1$	$\boldsymbol{A}_{11}$	$\boldsymbol{A}_{12}$	$-\boldsymbol{a}_1$
$\boldsymbol{y}_2$	$\boldsymbol{A}_{21}$	$\boldsymbol{A}_{22}$	$-\boldsymbol{a}_2$
	$\boldsymbol{z}_1$	$\boldsymbol{z}_2$	$-z$

zu beginnen. Die skalaren Koeffizienten a_{ij} von Abschnitt 1.1 werden dabei durch Matrizen ersetzt, wobei $\boldsymbol{A}_{11}$ zunächst als regulär vorausgesetzt wird. Das System wird ferner um einen Spaltenvektor $\boldsymbol{a}$ und einen Zeilevektor $\boldsymbol{z}$ erweitert. Die analog der Blockeinteilung gegliederten Randelemente beinhalten gemeinsam den skalaren Wert z. Natürlich können diese Randelemente an Stelle von Vektoren auch Matrizen darstellen. Wesentlich ist, daß diese Randelemente während des Austauschschrittes gleich behandelt werden, wie alle anderen Elemente, selbst jedoch nie als Pivotspalte oder Pivotzeile in Erscheinung treten.

Die Durchführung der Austauschschritte erfolgt analog zu Abschnitt 1.1 unter Beachtung der Reihenfolge der Matrizenmultiplikationen. Ein elementarer Austauschschritt kann wieder in vier Schritte zusammengefaßt werden:

1. Die Pivotmatrix geht in ihre Inverse über.
2. Die übrigen Matrizen der Pivotspalte werden von rechts mit der Inversen der Pivotmatrix multipliziert.
3. Eine Matrix im verbleibenden Teil (System ohne Pivotspalte und Pivotzeile) wird transformiert, indem man von ihm das Produkt subtrahiert, gebildet aus der entsprechenden Matrix der neu ermittelten Pivotspalte (Matrix in der gleichen Zeile) und der entsprechenden Matrix der Pivotzeile (Element in der gleichen Spalte).
4. Die übrigen Matrizen der Pivotzeile sind mit dem negativen inversen Pivotmatrix von links zu multiplizieren.

Widmen wir uns zunächst dem analytischen Zugang und erarbeiten die mathematischen Darstellung bei einem vollständigen Basistausch. Der Tausch der Variablengruppe $\boldsymbol{x}_1$

mit $\boldsymbol{y}_1$ liefert zunächst

(1.14)

	$\boldsymbol{y}_1$	$\boldsymbol{x}_2$	
$\boldsymbol{x}_1$	$\boldsymbol{A}_{11}^{-1}$	$-\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$	$\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1$
$\boldsymbol{y}_2$	$\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}$	$\boldsymbol{A}_{22}-\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$	$-\boldsymbol{a}_2+\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1$
	$\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}$	$\boldsymbol{z}_2-\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}$	$-z+\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1$

und der nachfolgende Tausch von $\boldsymbol{x}_2$ und $\boldsymbol{y}_2$ führt zu

(1.15)

	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	
$\boldsymbol{x}_1$	$\boldsymbol{A}_{11}^{-1}+\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\left(\square\right)^{-1}\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}$	$-\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\left(\square\right)^{-1}$	...
$\boldsymbol{x}_2$	$-\left(\square\right)^{-1}\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}$	$\left(\square\right)^{-1}$	...
	$\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}-\left(\boldsymbol{z}_2-\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\right)\left(\square\right)^{-1}\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}$	$\left(\boldsymbol{z}_2-\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\right)\left(\square\right)^{-1}$	...

...	
...	$\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1-\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\left(\square\right)^{-1}\left(\boldsymbol{a}_2-\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1\right)$
...	$\left(\square\right)^{-1}\left(\boldsymbol{a}_2-\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1\right)$
...	$-z+\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1+\left(\boldsymbol{z}_2-\boldsymbol{z}_1\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\right)\left(\square\right)^{-1}\left(\boldsymbol{a}_2-\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1\right)$

wobei bedingt durch Platzprobleme das zweite Pivotelement durch

$$\left(\square\right)=\boldsymbol{A}_{22}-\boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \tag{1.16}$$

dargestellt wird. Die hier gewonnene analytische Darstellungen werden in den nachfolgenden Abschnitten des öfteren verwendet, wo Bezug auf diesen Abschnitt genommen wird.

Neben der theoretischen Bedeutung des allgemeinen Austauschschrittes existieren eine Reihe von numerischen Anwendungen. Sehen wir den inneren Teil des Ausgangstableaus (1.13) als eine blockstrukturierte Matrix, wobei wir uns nicht auf eine 2×2 Struktur beschränken müssen, so kann mit Hilfe der Austauschschritte eine beliebige Matrix bearbeitet werden. Dieser verallgemeinerte Austauschschritt kann dort eingesetzten werden, wo nur beschränkt Arbeitsspeicher zur Verfügung steht und daher die Matrizenblöcke extern gespeichert werden müssen.

Die Unterstützung von schnellen Matrix-Operationen durch spezielle Hardware (Vektorprozessoren) oder Software (FORTRAN90, BLAS) verleihen dieser Methode auch bei genügend Speicher und virtueller Adressierung erneut Aktualität und Attraktivität.

1.4 Lösung eines Gleichungungssystems

Eine Möglichkeit zur Lösung eines linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{a} \tag{1.17}$$

haben wir schon im Abschnitt 1.2 kennengelernt. Durch die Bildung der Inversen $\boldsymbol{A}^{-1}$ kann die gesuchte Lösung $\boldsymbol{x}$ sofort durch

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{a} \tag{1.18}$$

berechnet werden, wenn $\boldsymbol{A}$ eine reguläre Matrix darstellt. Wir gehen zunächst von der Tatsache aus, daß vordringleich die Lösung $\boldsymbol{x}$ des regulären Systems gesucht ist und die Inverse selbst nicht benötigt wird. Den singuläre Fall bewahren wir uns für den nächsten Abschnitt auf. Starten wird ein Austauschverfahren mit dem Ausgangstableau

(1.19)

	$\boldsymbol{x}$	
$\boldsymbol{y}$	$\boldsymbol{A}$	$-\boldsymbol{a}_1$

und führen den Basiswechsel zwischen allen $\boldsymbol{x}$ und $\boldsymbol{y}$ durch, so enden wird nach Formel (1.14) mit dem Tableau

(1.20)

	$\boldsymbol{y}$	
$\boldsymbol{x}$	$\boldsymbol{A}^{-1}$	$\boldsymbol{A}^{-1}\boldsymbol{a}$

Anstelle der zusätzlich mitgeführten Spalte $-\boldsymbol{a}$ können wird sofort die Lösung des Systems ablesen. Das im Zentrum des Endtableaus die Inverse $\boldsymbol{A}^{-1}$ erzeugt wird, ist somit nur dann von Bedeutung, wenn diese Matrix auch wirklich für weitere Analysen benötigt wird. Ist nur die Lösung des Systems mit einer oder mehreren rechten Seiten erforderlich, so kann diese durch *Mitreduktion* ermittelt werden. Wenn die Inverse für weitere Analysen nicht benötigt wird, so kann man sich sehr einfach überlegen, daß deren vollständige Berechnung auch zur Ermittlung des Lösungsvektors unnotwendig ist. Aus der genaueren Analyse des Austauschprozesses geht hervor, daß jegliche Berechnung mit schon getauschten Pivotspalten unterbleiben kann, da diese keinen Einfluß auf die weitere Auswertung haben. Außerdem kann natürlich die Rückordnung der Spalten unterbleiben. Obwohl nur sehr wenige Änderungen an Algorithmus 1.2 vorzunehmen sind, wird auf Grund der Bedeutung dieser Aufgabenstellung trotzdem der geänderte Algorithmus hier abgedruckt.

Algorithmus 1.3 : Lösung eines quadratischen Gleichungssystems $\boldsymbol{Ax} = \boldsymbol{a}$ am Platz mit vollständiger Pivotsuche.

Aufgabenstellung: Lösung eines regulären Gleichungssystems $\boldsymbol{Ax} = \boldsymbol{a}$ ($\in \mathbb{R}^{n\times n}$).

Anmerkung: Die Berechnung erfolgt *am Platz*, sodaß die eingegebenen Koeffizienten $a_{k\ell}$ durch modifizierten Koeffizienten ersetzt werden.

Schnittstellen:

Eingabe:
- `a(n,n)` ... Koeffizienten der Matrix $\boldsymbol{A}$
- `n` ... Anzahl der Zeilen und Spalten der quadratischen Matrix $\boldsymbol{A}$
- `a(n,n+1:n+c)` ... ein oder mehrere Konstantenvektoren $\boldsymbol{a}$ (Achtung: Vorzeichen)
- `c` ... Anzahl der rechten Seiten

Ausgabe:
- `a(n,n+1:n+c)` ... Lösungen des Gleichungssystems
- `ind_z` ... Indexvektor Zeilen

Felder: `a(n,n+c),ind_z(n)`

Hilfsproceduren:
- `i,j = MAXLOC(a)` ... Zeile i und Spalte j des größten Koeffizienten der $n\times n$-Matrix $\boldsymbol{A}$.
- `TAUSCHE_SPALTEN(a,i,j)` ... Vertausche die Spalten i und j in der $n\times n$-Matrix $\boldsymbol{A}$.
- `TAUSCHE_ZEILEN(a,i,j)` ... Vertausche der Zeilen i und j in der $n\times n$-Matrix $\boldsymbol{A}$.
- `AUSTAUSCHSCHRITT(a,s,s)` ... siehe Algorithmus 1.1.

Ausgangssituation:

	$\boldsymbol{x}$	
$\boldsymbol{y}$	$\boldsymbol{A}$	$-\boldsymbol{a}_1$

Endsituation: reguläre Matrix

	$\boldsymbol{y}$	
$\boldsymbol{x}$	$\boldsymbol{A}^{-1}$	$\boldsymbol{A}^{-1}\boldsymbol{a}$

Endsituation: singuläre Matrix
Die linear unabhängigen Zeilen sind durch Einsen im Hilfsvektor `ind_z` erkennbar.

Lösungsweg: Pseudocode

1. *Initialisiere Austauschschleife und das Element zur Abschätzung von num. Null:*
   `s=0; ele_max=0 .`

Wolf-Dieter Schuh
ITGB, Uni Bonn

Initialisiere die Hilfsvektoren:
`ind_z(1:n)=0 .`

2. *Schleife über einen Austauschschritt:* `s=s+1`
   `IF s>n GOTO 10: .`
3. *Suche Position des Pivotelements:*
   `i_local,j_local=MAXLOC(ABS(a(s:n,s:n))`
   `i=i_local+s-1; j=j_local+s-1` *(i Pivotzeile, j Pivotspalte)*
   `IF ABS(a(i,j)) > ele_max THEN ele_max = ABS(a(i,j)) .`
4. *Überprüfe das Pivotelement auf num. Null:*
   `IF ABS(a(i,j))/ele_max < num_Null GOTO 12: .`
5. *Bringe Pivotspalte j in Spalte s durch Vertauschen:*
   `IF s≠j THEN TAUSCHE_SPALTEN(a(1:n,1:n+c),j,s) .`
6. *Bringe Pivotzeile i in Zeile s durch Vertauschen:*
   `IF s≠i THEN TAUSCHE_ZEILEN(a(1:n,s:n+c),i,s) .`
7. *Führe Austauschschritt mit Pivot a_{ss} durch:*
   `AUSTAUSCHSCHRITT(a(1:n,s:n+c),n,n,s,1)`
8. *Markiere den Inhalt der s-ten Zeile:*
   `ind_z(s)=j .`
9. *Nächster Austauschschritt:*
   `GOTO 2: .`
10. *Rücksortierung der Zeilen in der Lösung:*
    `DO ℓ=n-1 TO 1 STEP -1`
    `  i=ind_z(ℓ)`
    `  IF ℓ≠i THEN TAUSCHE_ZEILEN(a(1:n,n+1:nn),n,n,i,ℓ)`
    `END DO ℓ .`
11. `Ende:` 'O.K. ENDE, VOLLER RANG'.
12. *Rücksortierung der Indexvektoren bei Rangdefekt:*
    `DO k=s-1 TO 1 STEP -1`
    `  Behandle Zeile:`
    `  i=ind_z(k); ind_z(k)=ind_z(i); ind_z(i)=1`
    `END DO k .`
13. `Ende:` 'O.K. ENDE, KEIN VOLLER RANG, RANG = S-1' .

Ressourcen:

Anzahl der Operationen:	n `AUSTAUSCH`	$\Longrightarrow n^2\left(\frac{1}{2}(n+1)+c\right)$
Platzbedarf:	$n \times nn$	

Im Gegensatz zu Algorithmus 1.2 werden die Austauschschritte nicht immer für die vollständige $n \times nn$ Matrix durchgeführt, sondern für den Teil links der Pivotspalte unterdrückt. Die Anzahl der Operationen für den Austauschschritt senkt sich somit von $n \times nn$ auf $n \times nn - n + 1$. Die Gesamtzahl der Operationen beträgt somit $\sum_{s=1}^{n}(n+c-s+1)n$ bzw. $n^2\left(+\frac{1}{2}(n+1)+c\right)$. Gegenüber der vollständigen Inversion wird die Anzahl der Operationen somit halbiert.

Wolf-Dieter Schuh
ITGB, Uni Bonn

1.5 Verallgemeinerte Inverse

Wolf-Dieter Schuh
ITGB, Uni Bonn

1.6 Änderung der Dimension durch Vergrößerung und Verkleinerung

Wenden wir uns zunächst der ersten Aufgabenstellung zu. Von dem Gleichungssystem

$$\boldsymbol{A}_{11}\boldsymbol{x}_1 = \boldsymbol{a}_1 \tag{1.21}$$

liegen die Lösung $\bar{\boldsymbol{x}}_1$ und die Inverse $\boldsymbol{A}_{11}^{-1}$ vor. Gesucht ist die Lösung beziehungsweise die Inverse des erweiterten Systems

$$\begin{aligned} \boldsymbol{A}_{11} \quad \boldsymbol{x}_1 + \boldsymbol{A}_{12} \quad \boldsymbol{x}_2 &= \boldsymbol{a}_1 \\ \boldsymbol{A}_{21} \quad \boldsymbol{x}_1 + \boldsymbol{A}_{22} \quad \boldsymbol{x}_2 &= \boldsymbol{a}_2 \end{aligned} \tag{1.22}$$

Für die Berechnung der vollständigen Inversen wird auf Abschnitt 1.2 verwiesen. Hier soll die Lösung des erweiterten Systems $\boldsymbol{x}$ im Vordergrund stehen. Unter Zuhilfenahme von 1.15 errechnet sich $\boldsymbol{x}$ formal durch

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1 - \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\left(\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\right)^{-1}\left(\boldsymbol{a}_2 - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1\right) \\ \left(\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\right)^{-1}\left(\boldsymbol{a}_2 - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1\right) \end{bmatrix} \tag{1.23}$$

Berücksichtigt man die schon vorliegende Lösung $\bar{\boldsymbol{x}}_1$ des Systems (1.21), so errechnet sich $\boldsymbol{x}_2$ vereinfacht durch

$$\boldsymbol{x}_2 = \left(\boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\right)^{-1}\left(\boldsymbol{a}_2 - \boldsymbol{A}_{21}\bar{\boldsymbol{x}}_1\right) \ . \tag{1.24}$$

Die Lösung $\boldsymbol{x}_2$ kann somit als Ergebnis des Systems

$$\bar{\boldsymbol{A}}_{22}\boldsymbol{x}_2 = \bar{\boldsymbol{a}}_2 \tag{1.25}$$

mit den modifizierten Matrizen

$$\bar{\boldsymbol{A}}_{22} = \boldsymbol{A}_{22} - \boldsymbol{A}_{21}\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12} \tag{1.26}$$

$$\bar{\boldsymbol{a}}_2 = \boldsymbol{a}_2 - \boldsymbol{A}_{21}\bar{\boldsymbol{x}}_1 \tag{1.27}$$

gewonnen werden. Setzt man $\boldsymbol{x}_2$ für die Berechnung von $\boldsymbol{x}_1$ ein, so kann (1.23) umgeformt werden zu

$$\boldsymbol{x}_1 = \boldsymbol{A}_{11}^{-1}\boldsymbol{a}_1 - \boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\boldsymbol{x}_2 \tag{1.28}$$

Die erste Unbekanntengruppe $\boldsymbol{x}_1$ kann somit als Ergebnis des Systems

$$\boldsymbol{A}_{11}\boldsymbol{x}_1 = \bar{\boldsymbol{a}}_1 \tag{1.29}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

mit der modifizierten Matrix

$$\bar{\boldsymbol{a}}_1 = \boldsymbol{a}_1 - \boldsymbol{A}_{12}\boldsymbol{x}_2 \tag{1.30}$$

errechnet werden.

Bezüglich der zweiten Aufgabestellung - der Verkleinerung eines bereits berechneten Systems - starten wir mit von der inversen Ausgangssituation, wo $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ die unabhängigen und $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ die abhängigen Variablen repräsentieren.

(1.31)

	$\boldsymbol{y}_1$	$\boldsymbol{y}_2$	
$\boldsymbol{x}_1$	$\boldsymbol{A}_{11}^{(-1)}$	$\boldsymbol{A}_{12}^{(-1)}$	$\hat{\boldsymbol{x}}_1$
$\boldsymbol{x}_2$	$\boldsymbol{A}_{21}^{(-1)}$	$\boldsymbol{A}_{22}^{(-1)}$	$\hat{\boldsymbol{x}}_2$

Die Matrizen $\boldsymbol{A}_{ij}^{(-1)}$, $i = 1, 2$, $j = 1, 2$ bilden die Inverse zur Ausgangssituation

$$\begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}_{11}^{(-1)} & \boldsymbol{A}_{12}^{(-1)} \\ \boldsymbol{A}_{21}^{(-1)} & \boldsymbol{A}_{22}^{(-1)} \end{bmatrix} \tag{1.32}$$

Der Konstantenvektor beinhaltet die Lösung des erweiterten Systems $\hat{\boldsymbol{x}}_1$ und $\hat{\boldsymbol{x}}_2$. Zu ermitteln ist die Lösung $\bar{\boldsymbol{x}}_1$ beziehungsweise die Inverse des Systems

$$\boldsymbol{A}_{11}\boldsymbol{x}_1 = \boldsymbol{a}_1 \tag{1.33}$$

Formel (1.14) entnimmt man, daß nach dem Tausch der Unbekanntengruppe $\boldsymbol{x}_2$ und $\boldsymbol{y}_2$ das gewünschte Ergebnis in der ersten Zeile vorliegt.

	$\boldsymbol{y}_1$	$\boldsymbol{x}_2$	
$\boldsymbol{x}_1$	$\boldsymbol{A}_{11}^{(-1)} - \boldsymbol{A}_{12}^{(-1)}\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}\boldsymbol{A}_{12}^{(-1)}$	$\boldsymbol{A}_{12}^{(-1)}\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}$	$\hat{\boldsymbol{x}}_1 - \boldsymbol{A}_{12}^{(-1)}\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}\hat{\boldsymbol{x}}_2$
$\boldsymbol{y}_2$	$-\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}\boldsymbol{A}_{21}^{(-1)}$	$\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}$	$-\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}\hat{\boldsymbol{x}}_2$

(1.34)

Identifiziert man die Glieder in (1.15) und (1.34) so liefert die Gegenüberstellung des ersten Gliedes in der ersten Zeile und Spalte die Inverse

$$\boldsymbol{A}_{11}^{-1} = \boldsymbol{A}_{11}^{(-1)} - \boldsymbol{A}_{12}^{(-1)}\left(\boldsymbol{A}_{22}^{(-1)}\right)^{-1}\boldsymbol{A}_{21}^{(-1)} \ . \tag{1.35}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

Das erste Glied in der Konstantenspalte von (1.34) zeigt somit die gesuchte Lösung $\bar{\boldsymbol{x}}_1$

$$\bar{\boldsymbol{x}}_1 = \hat{\boldsymbol{x}}_1 - \boldsymbol{A}_{12}^{(-1)} \left(\boldsymbol{A}_{22}^{(-1)} \right)^{-1} \hat{\boldsymbol{x}}_2 \ . \tag{1.36}$$

auf. Gewinnt man die Lösung $\bar{\boldsymbol{x}}_2$ aus dem System

$$\boldsymbol{A}_{22}^{(-1)} \bar{\boldsymbol{x}}_2 = \hat{\boldsymbol{x}}_2 \tag{1.37}$$

so errechnet sich die gesuchte Lösung des verkleinerten Systems durch

$$\bar{\boldsymbol{x}}_1 = \hat{\boldsymbol{x}}_1 - \boldsymbol{A}_{12}^{(-1)} \bar{\boldsymbol{x}}_2 \ . \tag{1.38}$$

Sowohl das Hinzufügen oder Vielfach auch die Wirkung einer hinzuzufügenden Variable aber auch der Übergang auf weniger Variablen stellen wesentliche Aufgabenstellungen der dynamischen Optimierung dar.

1.7 Änderung der Koeffizienten

Bei vielen Anwendungen (Fehlersuche, Netzoptimierung) ändern sich die Koeffizienten des Gleichungssystems, wobei zwei Fälle unterschieden werden können. Für Änderungen, die die gesamte Matrix betreffen, eignen sich zwei Verfahren, die auch einander ergänzend, nacheinander angewandt werden können. Mit Hilfe der Neumann Serienentwicklung kann eine Änderung $\Delta \boldsymbol{A}$ der Ausgangsmatrix $\boldsymbol{A}_0$ ohne neue Inversion über geführt werden in eine Änderung der Inversen $\Delta \boldsymbol{A}^{(-1)}$, die einen Zuschlag zur ursprünglichen Inversen $\boldsymbol{A}_0^{-1}$ von $\boldsymbol{A}_0$ widerspiegelt

$$\begin{aligned}
\boldsymbol{A}^{-1} &= (\boldsymbol{A}_0 + \Delta \boldsymbol{A})^{-1} = \\
&= \left(\boldsymbol{A}_0 \left(\boldsymbol{I} + \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A}\right)\right)^{-1} = \\
&= \left(\boldsymbol{I} + \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A}\right)^{-1} \boldsymbol{A}_0^{-1} \approx \\
&\approx \left(\boldsymbol{I} - \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} + \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} - \ \ldots\right) \boldsymbol{A}_0^{-1} \approx \\
&\approx \boldsymbol{A}_0^{-1} - \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} + \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} - \ \ldots
\end{aligned} \tag{1.39}$$

Durch Zerlegung von

$$\boldsymbol{A}^{-1} = \boldsymbol{A}_0^{-1} + \Delta \boldsymbol{A}^{(-1)} \tag{1.40}$$

kann der Zuschlag $\Delta \boldsymbol{A}^{(-1)}$ dargestellt werden durch

$$\Delta \boldsymbol{A}^{(-1)} = -\boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} + \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} - \ \ldots \ . \tag{1.41}$$

Eine Iterationsvorschrift kann man erlangen, wenn man (1.40) umschreibt in

$$\boldsymbol{A}^{-1} = \boldsymbol{A}_0^{-1} \left(\boldsymbol{I} - \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} + \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} - \ \ldots\right) \ . \tag{1.42}$$

Mit der Ausgangsbedingung

$$\boldsymbol{H}^{(0)} = \boldsymbol{I} - \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} \tag{1.43}$$

erhält man die Iterationsvorschrift

$$\boldsymbol{H}^{(i)} = \boldsymbol{I} - \Delta \boldsymbol{A} \boldsymbol{A}_0^{-1} \boldsymbol{H}^{(i-1)} \ . \tag{1.44}$$

Die Inverse errechnet sich somit aus

$$\boldsymbol{A}^{-1} = \boldsymbol{A}_0^{-1} \boldsymbol{H}^{(i)} \ . \tag{1.45}$$

Eine andere Möglichkeit, die veränderte Inverse zu erlangen beziehungsweise eine näherungsweise berechnete Inverse $\boldsymbol{A}_0$ zu verbessern, ist ein iteratives Verfahren. Aus der nicht exakten Erfüllung der Definitionsgleichung

$$\boldsymbol{I} - \boldsymbol{A} \boldsymbol{A}_0^{-1} = \boldsymbol{R}_0 \neq \boldsymbol{0} \tag{1.46}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

ergibt sich eine Residuenmatrix $\boldsymbol{R}_0$. Durch Multiplikation mit der unbekannten, zu berechnenden Inversen $\boldsymbol{A}^{-1}$ und entsprechender Umformung erlangt man

$$\boldsymbol{A}^{-1} = \boldsymbol{A}_0^{-1} + \boldsymbol{A}^{-1}\boldsymbol{R}_0 \tag{1.47}$$

Verwendet man zur Berechnung des Korrekturgliedes anstelle von $\boldsymbol{A}^{-1}$ die Näherungsinverse $\boldsymbol{A}_0^{-1}$ und errechnet damit eine verbesserte Inverse $\boldsymbol{A}_1^{-1}$ so lautet die Iterationsvorschrift

$$\boldsymbol{A}_{i+1}^{-1} = \boldsymbol{A}_0^{-1} + \boldsymbol{A}_i^{-1}\boldsymbol{R}_0 \tag{1.48}$$

Wird während der Iteration auch die Matrix der Residuen neu ermittelt durch

$$\boldsymbol{R}_i = \boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}_i^{-1} \tag{1.49}$$

und anstelle von $\boldsymbol{R}_0$ in (1.48) verwendet

$$\boldsymbol{A}_{i+1}^{-1} = \boldsymbol{A}_i^{-1} + \boldsymbol{A}_i^{-1}\boldsymbol{R}_i \tag{1.50}$$

so tritt eine Konvergenzverbesserung ein.

Eine strenge, jedoch nicht inversionsfreie, Möglichkeit der Berechnung der geänderten Inversen ist dann gegeben, wenn sich die Änderung $\Delta\boldsymbol{A}$ durch

$$\Delta\boldsymbol{A} = \boldsymbol{U}^T\boldsymbol{V} \tag{1.51}$$

mit $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{k\times m}$ darstellen läßt. Die Inverse des geänderten Systems ist durch die *Sherman-Morrison-Woodburg* Formel gegeben

$$\left(\boldsymbol{A} + \boldsymbol{U}^T\boldsymbol{V}\right)^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{U}^T\left(\boldsymbol{I} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}^T\right)^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}\ , \tag{1.52}$$

die für geodätische Anwendungen etwas modifiziert

$$\left(\boldsymbol{A} \pm \boldsymbol{U}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{V}\right)^{-1} = \boldsymbol{A}^{-1} \mp \boldsymbol{A}^{-1}\boldsymbol{U}^T\left(\boldsymbol{\Sigma} \pm \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}^T\right)^{-1}\boldsymbol{V}\boldsymbol{A}^{-1}\ , \tag{1.53}$$

darstellbar ist, wobei $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{k\times m}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{k\times k}$ und $\boldsymbol{A}$, $\boldsymbol{\Sigma}$ und $\left(\boldsymbol{\Sigma} \pm \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}^T\right)$ regulär sind. Wie *P. Meissl (1967)[50]* zeigte, behält (1.53) auch bei verallgemeinerten Inversen bei nichtregulären Matrizen $\boldsymbol{A}$ ihre Gültigkeit.

Der Aufwand für die Inversion von $\left(\boldsymbol{\Sigma} \pm \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U}^T\right)$ ist abhängig von der Dimension k. Ist eine Änderung mit geringer Dimension k darstellbar, wie zum Beispiel Änderungen einzelner Beobachtungen beim vermittelnden Ausgleich, so ist durch (1.53) eine effiziente und strenge Neuberechnung möglich.

1.8 Iterative Verfahren

Neben den direkten Lösungsverfahren existiert eine Fülle von iterativen Methoden, bei denen die Lösung durch einen schrittweisen Vorgang angenähert wird. Obwohl die einzelnen Schritte oft durch einfache Operationen zu bewerkstelligen sind und wenig Rechen- und Speicheraufwand erfordern, sind die iterativen Verfahren nicht sehr weit verbreitet. Die Abschätzung des verbleibenden Restfehlers und der Konvergenzgeschwindigkeit erfordert eine genaue Kenntnis der Eigenschaften des linearen Gleichungssystems. Kennt man dieses Verhalten aufgrund von Spektralanalysen oder sich ähnelnder Aufgabenstellungen (z. B. bei finiten Netzen, Normalgleichungsmatrizen bei geodätischen Netzen, aus Kovarianzfunktionen abgeleitete Matrizen, konsistent geordneter Matrizen) so lassen sich für viele Probleme sehr effiziente Verfahren finden.

In diesem Abschnitt wird ein Überblick über die klassischen Iterationsverfahren (Jacobi, Gauß-Seidel) gegeben, wo nur wenig Information über die Eigenschaften des zu lösenden Systems notwendig ist. Die Darstellung des Prinzips der sukzessiven Überrelaxation rundet dieses Kapitel ab. Parallel zu den direkten Methoden werden in den nachfolgenden Abschnitten die iterativen Verfahren für spezielle Aufgabenstellungen adaptiert.

Die klassischen iterativen Verfahren zur Lösung des linearen Gleichungssystems

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{a} \tag{1.54}$$

bedienen sich der Aufspaltung der Matrix $\boldsymbol{A}$ in zwei zunächst beliebig wählbare Anteile

$$\boldsymbol{A} = \boldsymbol{S} - \boldsymbol{T}. \tag{1.55}$$

Der Übergang von der Näherungslösung $\boldsymbol{x}^{(i)}$ zur Lösung $\boldsymbol{x}^{(i+1)}$ wird durch die Iterationsvorschrift

$$\boldsymbol{S}\boldsymbol{x}^{(i+1)} = \boldsymbol{T}\boldsymbol{x}^{(i)} + \boldsymbol{a} \tag{1.56}$$

festgelegt. Da die Berechnung von $\boldsymbol{x}^{(i+1)}$ nach (1.56) nur von der letzten Näherungslösung $\boldsymbol{x}^{(i)}$ beeinflußt werden und ferner die Matrizen $\boldsymbol{S}$ und $\boldsymbol{T}$ unabhängig vom Iterationsschritt i sind, wird diese Iterationsvorschrift auch als *lineares stationäres Einschrittverfahren* bezeichnet. Die allgemeinste Iterationsvorschrift ist durch eine beliebige Funktion Φ_i in Abhängigkeit von mehreren Näherungslösungen gegeben,

$$\boldsymbol{x}^{(i+1)} = \Phi_i \left(\boldsymbol{x}^{(0)}, ..., \boldsymbol{x}^{(i)}, \boldsymbol{A}, \boldsymbol{a} \right). \tag{1.57}$$

Der Restfehler $\boldsymbol{e}^{(i)}$ der $i-$ten Iteration ist definiert durch die Differenz zwischen der Näherungslösung $\boldsymbol{x}^{(i)}$ und der exakten Lösung $\boldsymbol{x}$,

$$\boldsymbol{e}^{(i)} = \boldsymbol{x}^{(i)} - \boldsymbol{x}. \tag{1.58}$$

Da die exakte Lösung $\boldsymbol{x}$ durch die Iterationsvorschrift (1.56) wegen (1.54) und (1.55) reproduziert wird - Fixpunkteigenschaft der Lösung - kann durch

$$\boldsymbol{S}\left(\boldsymbol{x}^{(i)}-\boldsymbol{x}\right)=\boldsymbol{T}\left(\boldsymbol{x}^{(i-1)}-\boldsymbol{x}\right) \tag{1.59}$$

der Restfehler der $i-$ten Iteration aus dem Restfehler des vorangegangenen Schrittes durch

$$\boldsymbol{e}^{(i)}=\left(\boldsymbol{S}^{-1}\boldsymbol{T}\right)\boldsymbol{e}^{(i-1)} \tag{1.60}$$

errechnet werden. Eine rekursive Anwendung ermöglicht die Abschätzung des Restfehlers $\boldsymbol{e}^{(i)}$ bezüglich des Fehlers der Startlösung $\boldsymbol{x}^{(0)}$,

$$\boldsymbol{e}^{(i)}=\left(\boldsymbol{S}^{-1}\boldsymbol{T}\right)^{i}\boldsymbol{e}^{(0)} \tag{1.61}$$

Somit ist die Konvergenz für jeden beliebigen Startvektor $\boldsymbol{x}^{(0)}$ garantiert (notwendige und hinreichende Bedingung), wenn und nur wenn alle Eigenwerte λ_j (im allgemeinsten Fall die Singulärwerte) von $\boldsymbol{S}^{-1}\boldsymbol{T}$ die Bedingung

$$\mid \lambda_j \mid < 1, \qquad j=1,2,...,n \tag{1.62}$$

erfüllen (Beweis: siehe z. B. *W. Törnig et al. (1988)[92]*, Seite 163-164). Die Konvergenzgeschwindigkeit wird durch den betragsmäßig größten Eigenwert (Singulärwert) bestimmt. Diesen Wert wird als Spektralradius ρ von $\boldsymbol{S}^{-1}\boldsymbol{T}$ bezeichnet,

$$\rho=\rho\left(\boldsymbol{S}^{-1}\boldsymbol{T}\right)=\mathtt{max}\mid \lambda_j \mid . \tag{1.63}$$

Die Differenz der durch die Iterationsvorschrift (1.56) errechneten Näherungsvektoren

$$\boldsymbol{x}^{(i+1)}-\boldsymbol{x}^{(i)}=\boldsymbol{S}^{-1}\boldsymbol{T}\left(\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(i-1)}\right) \tag{1.64}$$

definiert unter Beachtung von (1.61) beziehungsweise $\rho<1$ eine kontrahierende Folge. Für die Vektornorm gilt somit

$$\mid \boldsymbol{x}^{(i+1)}-\boldsymbol{x}^{(i)} \mid \leq \rho \mid \boldsymbol{x}^{(i)}-\boldsymbol{x}^{(i-1)} \mid \leq \rho^{i} \mid \boldsymbol{x}^{(1)}-\boldsymbol{x}^{(0)} \mid . \tag{1.65}$$

Erweitert man diese Darstellung für ein positives ganzzahliges p so ergibt sich

$$\begin{aligned} \mid \boldsymbol{x}^{(i+p)}-\boldsymbol{x}^{(i)} \mid &\leq \sum_{\ell=1}^{p} \mid \boldsymbol{x}^{(i+\ell)}-\boldsymbol{x}^{(i+\ell-1)} \mid \\ &\leq \left(1+\rho+...+\rho^{\ell-1}\right) \mid \boldsymbol{x}^{(i+1)}-\boldsymbol{x}^{(i)} \mid \\ &\leq \tfrac{1-\rho^{p}}{1-\rho} \mid \boldsymbol{x}^{(i+1)}-\boldsymbol{x}^{(i)} \mid \end{aligned} \tag{1.66}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

Für $p \to \infty$ konvergiert die Folge gegen die exakte Lösung $\boldsymbol{x}$, sodaß eine Abschätzung der Fehler wegen

$$\mid \boldsymbol{x}^{(i+1)} - \boldsymbol{x} \mid = \mid \boldsymbol{e}^{(i)} \mid \leq \frac{1}{1-\rho} \mid \boldsymbol{x}^{(i+1)} - \boldsymbol{x}^{(i)} \mid \tag{1.67}$$

durch Anwendung von (1.65) ermöglicht wird. Die als *A-posteriori Fehlerabschätzung* bezeichnet Form erlaubt die Abschätzung des Restfehlers auf Grund der Zuschläge der letzten Iteration

$$\mid \boldsymbol{x}^{(i+1)} - \boldsymbol{x} \mid \leq \frac{\rho}{1-\rho} \mid \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i-1)} \mid \tag{1.68}$$

beziehungsweise die Form

$$\mid \boldsymbol{x}^{(i+1)} - \boldsymbol{x} \mid \leq \frac{\rho^i}{1-\rho} \mid \boldsymbol{x}^{(i)} - \boldsymbol{x}^{(0)} \mid \tag{1.69}$$

ermöglicht eine voraussichtliche, *A-priori Abschätzung* des Fehlers nach der $i-$ten Iteration auf Grund der Zuschläge des ersten Schrittes.

Eine zuverlässige Schätzung der Genauigkeit der Startlösung $\boldsymbol{e}^{(0)}$ erlaubt eine Abschätzung der Anzahl der notwendigen Iterationen um den Betrag des Restfehlers um einen Faktor 10^{-d} - also um d Ziffern - zu verkleinern. Wegen

$$\frac{\mid \boldsymbol{e}^{(i)} \mid}{\mid \boldsymbol{e}^{(0)} \mid} \leq \rho \left(\boldsymbol{S}^{-1} \boldsymbol{T} \right)^i \leq 10^{-d} \tag{1.70}$$

kann die maximale Anzahl der notwendigen Iterationen i_{max} um eine um d Ziffern gesteigerte Genauigkeit bezüglich der Startlösung zu erlangen durch

$$i_{max} \approx \frac{d}{\log_{10} \rho \left(\boldsymbol{S}^{-1} \boldsymbol{T} \right)} \tag{1.71}$$

abgeschätzt werden.

Die Problematik besteht nun in der Tatsache, daß manchmal die Genauigkeit der Startlösung recht gut abgeschätzt werden kann, selten jedoch Informationen über den Spektralradius verfügbar sind. Die einzige leicht verfügbare Information über die Güte der Lösung stellt der Residuenvektor

$$\boldsymbol{r}^{(i)} = \boldsymbol{A}\boldsymbol{x}^{(i)} - \boldsymbol{a} \tag{1.72}$$

dar. Da der Residuenvektor der wahren Lösung verschwindet, sind starke Auslöschungserscheinungen bei der numerischen Berechnung zu erwarten. Eine Beziehung zwischen dem wahren Fehler $\boldsymbol{e}^{(i)}$ und dem Residuenvektor $\boldsymbol{r}^{(i)}$ kann durch Erweiterung der obigen Gleichung mit der wahren Lösung hergestellt werden, woraus sich

$$\boldsymbol{e}^{(i)} = \boldsymbol{A}^{-1} \boldsymbol{r}^{(i)} \tag{1.73}$$

beziehungsweise

$$| \boldsymbol{e}^{(i)} | \leq | \boldsymbol{A}^{-1} | \, | \boldsymbol{r}^{(i)} | \tag{1.74}$$

ableitet. Für die Abschätzung der Norm von $\boldsymbol{A}^{-1}$ stehen verschiedene Möglichkeiten zur Verfügung. Bei finiten Netzen ermöglicht die Kenntnis des zu erwartenden, maximalen mittleren Fehlers und des Verlaufes des Fehlers eine Abschätzung der Norm (siehe dazu *W.D. Schuh (1984)[73]*, Seite 54-57). Diese Aussagen können von einem geübten Bearbeiter schon vor der Berechnung vorgenommen werden. Bei der Anwendung von iterativen Verfahren bei Kovarianzmatrizen grenzt die Größe des zufälligen statistischen Anteils (Rauschen) die Eigenwerte der Matrix $\boldsymbol{A}$ nach unten hin ab, wodurch eine zuverlässige Obergrenze der Norm von $\boldsymbol{A}^{-1}$ verfügbar ist.

Ein untere Grenze für die Norm von $\boldsymbol{A}^{-1}$ kann auch aus dem Iterationsprozeß gewonnen werden. Betrachtet man die Differenz der Residuenvektoren von zwei Iterationsschritten i und j

$$\boldsymbol{r}^{(j)} - \boldsymbol{r}^{(i)} = \boldsymbol{A}\boldsymbol{x}^{(j)} - \boldsymbol{A}\boldsymbol{x}^{(i)} \tag{1.75}$$

so erlangt man

$$\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)} = \boldsymbol{A}^{-1}\left(\boldsymbol{r}^{(j)} - \boldsymbol{r}^{(i)}\right). \tag{1.76}$$

Durch den Übergang auf die Norm und die Isolierung von $| \boldsymbol{A}^{-1} |$ kann

$$| \boldsymbol{A}^{-1} | \geq \frac{\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(i)}}{\boldsymbol{r}^{(j)} - \boldsymbol{r}^{(i)}} \tag{1.77}$$

gefunden werden.

Ziel der zunächst beliebigen Aufspaltung von $\boldsymbol{A}$ in die Matrizen $\boldsymbol{S}$ und $\boldsymbol{T}$ sollte es sein, den Spektralradius $\rho\left(\boldsymbol{S}^{-1}\boldsymbol{T}\right)$ möglichst klein zu halten. Die auf den ersten Blick offensichtlich beste Wahl $\boldsymbol{S} = \boldsymbol{A}, \boldsymbol{T} = \boldsymbol{0}$ minimiert den Spektralradius $\rho = 0$ und führt daher zur Lösung nach dem ersten Schritt. Dieser Schritt benötigt die sehr rechenaufwendige Bestimmung von der Inversen von $\boldsymbol{A}$ (bzw. genauer die Lösung des linearen Gleichungssystems). Die Äquivalenz zu direkten Verfahren ist augenscheinlich. Zur Umsetzung der Idee der iterativen Verfahren sind zwei Bedingungen bei der Wahl von $\boldsymbol{S}$ und $\boldsymbol{T}$ zu erfüllen: Einerseits muß $\boldsymbol{S}$ eine reguläre Matrix sein und andererseits soll die Iterationsvorschrift (1.56) leicht auswertbar sein. Dies ist durch Wahl einer Diagonal- oder Dreiecksstruktur für $\boldsymbol{S}$ gegeben.

Zur einheitlichen Darstellung des Nachfolgenden spalten wir die Matrix $\boldsymbol{A}$ in drei Teile

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}, \tag{1.78}$$

wobei $\boldsymbol{L}$ ($\boldsymbol{U}$) den unteren (oberen) Dreiecksteil der Matrix $\boldsymbol{A}$ ohne die Diagonale symbolisiert. Die Diagonale von $\boldsymbol{A}$ wird durch $\boldsymbol{D}$ dargestellt. Weitverbreitet sind folgende Annahmen für $\boldsymbol{S}$:

- $\boldsymbol{S} = \boldsymbol{D}$, $\quad \boldsymbol{T} = -(\boldsymbol{L} + \boldsymbol{U})$ $\quad$ Jacobi-Verfahren
- $\boldsymbol{S} = \boldsymbol{L} + \boldsymbol{D}$, $\quad \boldsymbol{T} = -\boldsymbol{U}$ $\quad$ Gauß-Seidel Verfahren

die im Nachfolgenden genauer dargestellt werden.

1.8.1 Jacobi-Verfahren

Die einfachste Wahl der Matrix $\boldsymbol{S}$ ist durch die Diagonalmatrix $\boldsymbol{S} = \boldsymbol{D}$ gegeben. Die Iterationsvorschrift ist somit definiert durch

$$\boldsymbol{D}\boldsymbol{x}^{(i+1)} = -\left(\boldsymbol{L} + \boldsymbol{U}\right)\boldsymbol{x}^{(i)} + \boldsymbol{a}. \tag{1.79}$$

Die vollständige Auswertung dieser Formel wird als $i-$ter Iterationsschritt bezeichnet. Pro Iterationsschritt erfährt jede Unbekannte $\boldsymbol{x}^{(i+1)}$ eine Veränderung, wobei die Informationen (genäherten Unbekannte) des vorangegangenen Schrittes $\boldsymbol{x}^{(i)}$ zu Grunde gelegt werden. Dieses Verfahren wird daher auch als *Gesamtschrittverfahren* bezeichnet und ist auf Grund der zuvor erwähnten Unabhängigkeit von den aktuell berechneten Unbekannten für die Paralellverarbeitung geeignet. Explizit ergibt sich die Rechenvorschrift zur Berechnung der einzelnen Unbekannten $x_\ell^{(i+1)}$ im $i+1-$ten Schritt mit

$$\boldsymbol{x}_\ell^{(i+1)} = \frac{1}{a_{\ell\ell}}\left(-\sum_{k=1}^{\ell-1} a_{\ell k}\boldsymbol{x}_k^{(i)} - \sum_{k=\ell+1}^{n} a_{\ell k}\boldsymbol{x}_k^{(i)} + a_\ell\right) \tag{1.80}$$

für $\ell = 1, 2, ..., n$, wobei vorausgesetzt wird, daß alle Diagonalelemente $a_{\ell\ell}$ ungleich Null sind, was bei regulärem $\boldsymbol{A}$ durch Zeilentausch gewährleistet ist. Der Spektralradius ρ kann durch die maximale Zeilensummennorm

$$\rho\left(\boldsymbol{S}^{-1}\boldsymbol{T}\right) = \mid \boldsymbol{D}^{-1}\left(\boldsymbol{L} + \boldsymbol{U}\right) \mid_\infty = \max_\ell \sum_{\substack{k=1 \\ \ell \neq k}}^{n} \frac{\mid a_{\ell k} \mid}{\mid a_{\ell\ell} \mid} \tag{1.81}$$

abgeschätzt werden. Diese Abschätzung dient einerseits dem Konvergenznachweis und liefert andererseits eine Aussage über die Konvergenzgeschwindigkeit. Eine notwendige Bedingung für die globale Konvergenz stellt die Forderung nach einem Spektralradius kleiner als Eins dar. Um dies zu gewährleisten ist eine strikt diagonaldominante Matrix, möglicherweise nach einer Vertauschung der Reihenfolge der Zeilen, unbedingt erforderlich.

1.8.2 Gauß-Seidel-Verfahren

Die Wahl einer Dreiecksmatrix ermöglicht ebenfalls eine rasche und einfache Berechnung der Iterationsvorschrift (1.56). Wählt man zur Iteration

$$\left(\boldsymbol{L} + \boldsymbol{D}\right)\boldsymbol{x}^{(i+1)} = -\boldsymbol{U}\boldsymbol{x}^{(i)} + \boldsymbol{a}. \tag{1.82}$$

so wird zur Berechnung jeder einzelnen Unbekannten x_ℓ im $(i+1)-$ten Iterationsschritt die gesamte bisher erarbeitete Information des Schrittes i und des Schrittes $i+1$ für

die Unbekannten kleiner als ℓ verwendet. Da ein elementweises Vorgehen bei dieser Methode erforderlich ist, wird es auch als *Einzelschrittverfahren* bezeichnet. Die explizite Iterationsvorschrift für $\ell = 1, 2, ..., n$ ergibt sich somit durch

$$\boldsymbol{x}_\ell^{(i+1)} = \frac{1}{a_{\ell\ell}} \left(-\sum_{k=1}^{\ell-1} a_{\ell k} \boldsymbol{x}_k^{(i+1)} - \sum_{k=\ell+1}^{n} a_{\ell k} \boldsymbol{x}_k^{(i)} + a_\ell \right) \tag{1.83}$$

Eine Aussage über den Spektralradius kann hier nicht unmittelbar abgeleitet werden. Für die Anwendung bei symmetrischen, positiv definiten Matrizen ist die Konvergenz immer gewährleistet (Beweis: siehe z. B. *W. Törnig et al. (1988)[92]*, Seite 174-175).

1.8.3 Sukzessive Überrelaxation

Geleitet vom oft zu beobachtenden Konvergenzverhalten - monotone oder alternierende Annäherung zur exakten Lösung - wird versucht durch eine Über- oder Unterkompensation eine Beschleunigung des Iterationsvorganges herbeizuführen. Dazu wird die Rechenvorschrift des Gauß-Seidel-Verfahrens in der Fixpunktform durch Multiplikation mit eine Skalar $\omega > 1$ zu

$$(\omega \boldsymbol{L} + \omega \boldsymbol{D})\, \boldsymbol{x} = -\omega \boldsymbol{U} \boldsymbol{x} + \omega \boldsymbol{a} \tag{1.84}$$

modifiziert, wobei der Anteil der Diagonale aufgespalten wird,

$$(\omega \boldsymbol{L} + \boldsymbol{D})\, \boldsymbol{x} = (1-\omega)\, \boldsymbol{D}\boldsymbol{x} - \omega \boldsymbol{U} \boldsymbol{x} + \omega \boldsymbol{a}. \tag{1.85}$$

Die explizite Iterationsvorschrift ergibt sich mit

$$\begin{aligned} \boldsymbol{x}_\ell^{(i+1)} &= \tfrac{1}{a_{\ell\ell}} \left(-\omega \sum_{k=1}^{\ell-1} a_{\ell k} \boldsymbol{x}_k^{(i+1)} + (1-\omega) a_{\ell\ell} \boldsymbol{x}_\ell^{(i)} - \omega \sum_{k=\ell+1}^{n} a_{\ell k} \boldsymbol{x}_k^{(i)} + \omega a_\ell \right) = \\ &= (1-\omega) \boldsymbol{x}_\ell^{(i)} + \tfrac{\omega}{a_{\ell\ell}} \left(-\sum_{k=1}^{\ell-1} a_{\ell k} \boldsymbol{x}_k^{(i+1)} - \sum_{k=\ell+1}^{n} a_{\ell k} \boldsymbol{x}_k^{(i)} + a_\ell \right) \end{aligned} \tag{1.86}$$

Der Überrelaxationsfaktor ω sollte so gewählt werden, daß der Spektralradius

$$\rho\left(\boldsymbol{S}^{-1}\boldsymbol{T}\right) = \rho\left((\omega \boldsymbol{L} + \boldsymbol{D})^{-1} \left((1-\omega)\, \boldsymbol{D} - \omega \boldsymbol{U} \right) \right) \tag{1.87}$$

möglichst klein wird. Wenn das Gauß-Seidel-Verfahren konvergiert ist die Konvergenz des Verfahrens der sukzessiven Unterrelaxation gewährleistet, wenn $0 < \omega \leq 1$ erfüllt ist. Für symmetrische, positiv definite Systeme garantiert die Wahl $0 < \omega \leq 2$ die Konvergenz (Beweis: siehe z. B. *W. Törnig et al. (1988)[92]*, Seite 174-176).

Eine optimale Wahl erfordert genaue Kenntnis des Spektrums der Problemstellung. Für spezielle Matrizen und Anwendungen sind bei *R. Varga (1962)[94]* und *D. Young (1971)[98]* Funktionen und Tabellen zu finden, die bei der Wahl des Relaxationfaktors behilflich sind.

Kapitel 2

Symmetrische, positiv definite Systeme

Die große Bedeutung von symmetrischen Matrizen in der geodätischen Optimierung wird durch Normalgleichungs- und Kovarianzmatrizen begründet. Vorwiegend werden die Lösungen dieser Systeme benötigt. Für statistische Fehleraussagen sind manchmal auch Teile der Inversen erforderlich. In seltenen Fällen muß die gesamte Inverse ermittelt werden. Für die exakte Analyse der Zuverlässigkeit und Stabilität werden Netze mit Hilfe von spektralen Verfahren durchleuchtet. Aus diesem Grund wird im Abschnitt 2.1 die spektrale Zerlegung behandelt. Im engen Zusammenhang damit stehen Stabilitäts- und Sensibilitätsbetrachtungen der Lösung bzw. abgeleiteter Größen die im Abschnitt 2.2 besprochen werden. Möglichkeiten zur Verbesserung der Kondition stehen im Mittelpunkt des dritten Abschnittes. Die folgenden zwei Abschnitte widmen sich den hauptsächlich verwendeten Methoden der direkten und indirekten Lösungstechniken. Als Hauptvertreter werden dabei die Methode nach Cholesky (Abschnitt 2.4.1) und das Verfahren der konjugierten Gradienten (Abschnitt 2.5) detailliert besprochen.

2.1 Spektralzerlegung

Um die speziellen Eigenheiten von geodätischen Netzen zu studieren, werden die auftretenden Gleichungssysteme mit Hilfe von Spektralmethoden untersucht. Da das System der Normalgleichungen und Kovarianzmatrix eine reelle, symmetrische Matrizen N bildet, existiert eine orthogonale Matrix V , sodaß

$$\boldsymbol{V}^T\boldsymbol{N}\boldsymbol{V} = \boldsymbol{\Lambda} \tag{2.1}$$

gilt, wobei Λ eine reelle Diagonalform hat (siehe *R. Zurmühl (1964)[99]*, Seite 190, Satz 3). Diese reellen Diagonalelemente von $\boldsymbol{\Lambda}$ sind die charakteristischen Wurzeln oder Eigenwerte von $\boldsymbol{N}$. Die Matrix $\boldsymbol{V}$ wird *Spektralmatrix* bezeichnet. Die Spalten der orthogonalen Matrix $\boldsymbol{V}$, die *Modalmatrix* genannt wird, bilden die Eigenvektoren $\boldsymbol{v}_i$. Da normierte Eigenvektoren vorausgesetzt werden, gilt

$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I} \ , \tag{2.2}$$

wobei $\boldsymbol{I}$ die Einheitsmatrix bezeichnet. Auf Grund der Orthonormiertheit entspricht die transponierte Matrix den Eigenvektoren gleich seiner Inversen

$$\boldsymbol{V}^T = \boldsymbol{V}^{-1} \tag{2.3}$$

beziehungsweise

$$\left(\boldsymbol{V}^T\right)^{-1} = \boldsymbol{V} \ . \tag{2.4}$$

Mit Hilfe der Beziehungen (2.3) und (2.4) läßt sich aus der Gleichung (2.1) die Matrix $\boldsymbol{N}$ ausdrücken durch

$$\boldsymbol{N} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T \ . \tag{2.5}$$

Die Inverse dieser Matrix ergibt sich, wegen der Gültigkeit von

$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \tag{2.6}$$

bei regulären Matrizen $\boldsymbol{A}$ und $\boldsymbol{B}$, mit

$$\boldsymbol{N}^{-1} = \left(\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T\right)^{-1} = \left(\boldsymbol{V}^T\right)^{-1}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^{-1} \ . \tag{2.7}$$

Berücksichtigt man die Orthonormiertheit von $\boldsymbol{V}$ so folgt

$$\boldsymbol{N}^{-1} = \boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^T \ . \tag{2.8}$$

Betrachtet man die einzelnen Eigenvektoren, spaltet also die Modalmatrix $\boldsymbol{V}$ in die Spaltenvektoren $\boldsymbol{v}^{(i)}$ auf, und beachtet, daß die Spektralmatrix $\boldsymbol{\Lambda}$ Diagonalform hat,

so kann die symmetrische Matrix $\boldsymbol{N}$ durch die Summe von dyadischen Produkten dargestellt werden,

$$\boldsymbol{N} = \lambda_1 \boldsymbol{v}^{(1)} \boldsymbol{v}^{(1)^T} + \lambda_2 \boldsymbol{v}^{(2)} \boldsymbol{v}^{(2)^T} + \ldots + \lambda_n \boldsymbol{v}^{(n)} \boldsymbol{v}^{(n)^T} = \sum_{i=1}^{n} \lambda_i \boldsymbol{v}^{(i)} \boldsymbol{v}^{(i)^T} \tag{2.9}$$

Die reguläre Inverse (2.8) ergibt sich durch

$$\boldsymbol{N}^{-1} = \frac{1}{\lambda_1} \boldsymbol{v}^{(1)} \boldsymbol{v}^{(1)^T} + \frac{1}{\lambda_2} \boldsymbol{v}^{(2)} \boldsymbol{v}^{(2)^T} + \ldots + \frac{1}{\lambda_n} \boldsymbol{v}^{(n)} \boldsymbol{v}^{(n)^T} = \sum_{i=1}^{n} \frac{1}{\lambda_i} \boldsymbol{v}^{(i)} \boldsymbol{v}^{(i)^T} \tag{2.10}$$

Liegt ein Rangdefekt d vor, sodaß die Matrix $\boldsymbol{N}$ nur $r = n - d$ nicht verschwindende Eigenwerte besitzt, so kann die Pseudoinverse (siehe Abschnitt 1.5) durch

$$\boldsymbol{N}^{+} = \frac{1}{\lambda_1} \boldsymbol{v}^{(1)} \boldsymbol{v}^{(1)^T} + \frac{1}{\lambda_2} \boldsymbol{v}^{(2)} \boldsymbol{v}^{(2)^T} + \ldots + \frac{1}{\lambda_n} \boldsymbol{v}^{(r)} \boldsymbol{v}^{(r)^T} \ , \qquad r = \mathrm{Rg}(\boldsymbol{N}) \tag{2.11}$$

angegeben werden. Für diese und weitere Darstellungen wird ohne Beschränkung der Allgemeinheit angenommen, daß die ersten r Eigenwerte die Nichtverschwindenden sind und daß die Eigenwerte in der Größe absteigend sortiert sind,

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_r \ , \qquad \lambda_{r+1} = \ldots = \lambda_n = 0 \tag{2.12}$$

Der zum kleinsten Eigenwert gehörende Eigenvektor wird für die weitere Darstellung als *kleinster Eigenvektor* bezeichnet. An Hand der Gleichungen (2.10) bzw. (2.11) kann der Einfluß jedes einzelnen Eigenwertes auf die Inverse unmittelbar abgelesen werden.

Ergänzend sei hier noch folgende Berechnungsmöglichkeit zur Berechnung der Pseudoinversen erwähnt, die bei Kenntnis einer orthonormierten Basis in $S \perp (N)$ gewinnbringend sein kann. Sind bei einer singulären, symmetrischen Matrix die Eigenvektoren der verschwindenden Eigenwerte λ_i, $i = r+1, \ldots, n$ bekannt, so läßt sich die Pseudoinverse nach

$$\boldsymbol{N}^{+} = \left(\boldsymbol{N} + \boldsymbol{v}^{(r+1)} \boldsymbol{v}^{(r+1)^T} + \ldots \boldsymbol{v}^{(n)} \boldsymbol{v}^{(n)^T} \right)^{-1} - \boldsymbol{v}^{(r+1)} \boldsymbol{v}^{(r+1)^T} - \ldots \boldsymbol{v}^{(n)} \boldsymbol{v}^{(n)^T} \tag{2.13}$$

berechnen (*H. Pelzer (1980)[62]*, Seite 20-23).

Die sehr rechenaufwendige numerische Berechnung der Eigenwerte und Eigenvektoren kann mit Hilfe mehrerer Verfahren durchgeführt werden. Eine Übersichtdarstellung ist bei *K. Oreschnik (1985)[60]*, Seite 15-29 gegeben. Detailliertere Ausführungen mit genauen Rechenvorschriften sind zum Beispiel bei *H.R. Schwarz (1968)[74], (1980)[76] und (1988)[78]* zu finden. FORTRAN Implementierungen sind in *H.R. Schwarz (1981)[77]* zusammengestellt. Eine Darstellung der Lanczos Methode, die für die Berechnung der Eigenwerte bei schwach besetzten Systemen geeignet ist, ist bei *G.H. Golub et.al. (1983)[38]*, Seite 322-351 nachzulesen, wo auch umfangreiche Literatur angegeben wird.

Literatur:

[38] GOLUB Gene H., Charles F. van LOAN (1983): Matrix Computations. North Oxford Academic, Oxford.

[60] ORESCHNIK Kurt (1985): Analyse von zweidimensionalen Netzen mit Hilfe von Eigenwerten und Eigenvektoren. Diplomarbeit, TU Graz.

[62] PELZER H. (1980): Geodätische Netze in Landes- und Ingenieurvermessung. Vermessungswesen Band 5, Konrad Wittwer.

[74] SCHWARZ Hans Rudolf (1968): Numerik symmetrischer Matrizen. Teubner, Stuttgart.

[77] SCHWARZ Hans Rudolf (1981): FORTRAN - Programme zur Methode der finiten Elemente. Teubner Studienbücher für Mathematik.

[78] SCHWARZ Hans Rudolf (1988): Numerische Mathematik. Teubner, Stuttgart, 2.Auflage.

Der Einfluß der einzelnen Eigenwerte und Eigenvektoren auf die Lösung eines zunächst regulären Systems

$$\boldsymbol{N}\boldsymbol{x} = \boldsymbol{b} \tag{2.14}$$

kann durch die Darstellung der Inverse $\boldsymbol{N}^{-1}$ als Summe von dyadischen Produkten (2.10) mit

$$\boldsymbol{x} = \boldsymbol{n}^{-1}\boldsymbol{b} = \sum_{i=1}^{n} \frac{1}{\lambda_i} \boldsymbol{v}^{(i)} \boldsymbol{v}^{(i)^T} \boldsymbol{b} \tag{2.15}$$

angegeben werden. Faßt man die skalare Größe, die durch das Skalarprodukt $\boldsymbol{v}^{(i)^T}\boldsymbol{b}$ entsteht, mit dem entsprechendem Eigenwert λ_i zusammen

$$\boldsymbol{x} = \sum_{i=1}^{n} \frac{\boldsymbol{v}^{(i)^T}\boldsymbol{b}}{\lambda}_i \boldsymbol{v}^{(i)} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}^{(i)} \tag{2.16}$$

so ist eine Form gefunden, die den Einfluß α_i jedes Eigenwertes bzw. Eigenvektors, unter Beachtung des Informationsgehaltes des Eigenvektors, widerspiegelt. Für Systeme, bei denen ein Rangdefekt d vorliegt, ist die Lösung (2.16) zu erweitern und lautet in der allgemeinsten Form

$$\boldsymbol{x} = \sum_{i=1}^{r} \alpha_i \boldsymbol{v}^{(i)} + \sum_{i=r+1}^{n} \beta_i \boldsymbol{v}^{(i)} \tag{2.17}$$

mit den Einflußfaktoren α_i, $i = 1, \ldots, r$ und den freien Parametern β_i, $i = r+1, \ldots, n$. Die der Bildung der Pseudoinversen $\boldsymbol{N}^{+}$ äquivalente Forderung nach einer minimalen Länge des Lösungsvektors führt auf die Wahl $\beta_i = 0$, $i = r+1, \ldots, n$.

Vielfach ist die Herkunft der Systeme und auch das Ziel der Berechnungen zur Abschätzung der Genauigkeit und Zuverlässigkeit der erzielten Ergebnisse zu berücksichtigen. Bei geodätischen Fragestellungen handelt es sich oft um überbestimmte, nicht konsistente Systeme

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\ell} + \boldsymbol{v} \ , \tag{2.18}$$

die mit Hilfe eines Zuschlages (Verbesserung $\boldsymbol{v}$) in eine konsistente Form gebracht werden. Die Minimierung der Funktion $\Phi(\boldsymbol{x})$

$$\Phi(\boldsymbol{x}) = \boldsymbol{v}^T\boldsymbol{v} \ \ldots \ \min \tag{2.19}$$

stellt somit das Ziel der Berechnung dar. Begründet durch diese Forderung, errechnet sich die optimale Lösung $\tilde{\boldsymbol{x}}$ aus dem symmetrischen, positiv definiten Gleichungssystem

$$\boldsymbol{N}\tilde{\boldsymbol{x}} = \boldsymbol{b} \qquad \text{mit} \qquad \boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}^T\boldsymbol{\ell} \tag{2.20}$$

(*Normalgleichungen*). Die zu minimierende Funktion $\Phi(\boldsymbol{x})$ (2.19) kann unter Verwendung der *Beobachtungsgleichungen* (2.18) assembliert werden. Formt man die Zielfunktion (2.19) unter Verwendung von (2.18) und (2.20) um so erlangt man die Form

$$\Phi(\boldsymbol{x}) = \boldsymbol{v}^T\boldsymbol{v} = -\boldsymbol{\ell}^t\boldsymbol{v} \ . \tag{2.21}$$

Bedingt durch die Verwendung der Standardbezeichungen ist im folgenden etwas Vorsicht geboten. Der Vektor der Verbesserungen wird mit $\boldsymbol{v}$ bezeichnet und ist durch die Kleinschreibung (Vektor) und das Fehlen des Index eindeutig identifizierbar. Die Spalten der Modalmatrix $\boldsymbol{V}$ (Eigenvektoren) werden wie zuvor durch $\boldsymbol{v}^{(i)}$ gekennzeichnet. Setzt man in die Zielfunktion (2.21) für die Verbesserungen $\boldsymbol{v}$ die Beobachtungsgleichungen (2.18) ein

$$\Phi(\boldsymbol{x}) = \boldsymbol{\ell}^T\boldsymbol{\ell} = -\boldsymbol{\ell}^t\boldsymbol{A}\tilde{\boldsymbol{x}} \tag{2.22}$$

und ersetzt $\tilde{\boldsymbol{x}}$ durch die Normalgleichungen (2.20), so ergibt sich die modifizierte Zielfunktion (*„Gewinn der Ausgleichung“*)

$$\Phi(\boldsymbol{x}) = \boldsymbol{\ell}^T\boldsymbol{\ell} - \boldsymbol{b}^T\boldsymbol{N}^+\boldsymbol{b} \ . \tag{2.23}$$

Die Verwendung der Spektraldarstellung (2.11) ermöglicht, bedingt durch das Auftreten der Diagonalmatrix Λ, die Summenform

$$\Phi(\boldsymbol{x}) = \sum_{i=1}^{beo} \ell_i^2 - \sum_{i=1}^{r} \frac{1}{\lambda_i} h_i^2 \ , \quad \text{mit} \quad h_i = \boldsymbol{v}^{(i)^T}\boldsymbol{b} \ . \tag{2.24}$$

Die erste Summe *beo* bezieht sich auf die Anzahl der Beobachtungsgleichungen. Der Rang r der Matrix $\boldsymbol{N}$ zeichnet für die zweite Summation verantwortlich. Analog wie in (2.16) der Einfluß eines Eigenwertes auf die Lösung, kann mit Hilfe von (2.24) der Einfluß jedes Eigenwertes auf die zu minimierende Funktion berechnet werden. Die Abschätzung des Fehlers, der durch die Wahl einer Schranke hervorgerufen wird, erleichter die nur problemorientiert zu treffende Entscheidung.

Wolf-Dieter Schuh
ITGB, Uni Bonn

2.2 Stabilität und Sensibilität der Lösung

Eine schwierige und problemabhängige Aufgabe stellt die Beantwortung der Frage nach der gewünschten bzw. erzielten Rechengenauigkeit dar. Welche maschinen- oder implemtierungsbedingte Rechengenauigkeit ist zu erwarten? Welche Folgen bewirkt die Einführung einer Schranke (*„numerisch Null“*) unter der sehr keine Eigenwerte zu Null gesetzt werden?

Zur Beantwortung dieses Fragenkomplexes muß man sich bewußt machen, daß verschiedene Fehlerquellen existieren. Die einem numerischen Prozeß zur Verfügung gestellten Daten sind meist durch komplexe Berechnungen bestimmt worden und weisen eine bestimmte Genauigkeit auf. Vielfach repräsentieren die Daten eine vereinfachte Darstellung eines komplizierten Modells, wo Diskretisierungen und Linearisierungen vor genommen werden müssen, um rechenbare funktionale Modelle zu gewinnen. Aber auch stochastische Modelle sind vielfach von Abschätzungen und empirischen Verfahren geprägt. Die daraus resultierenden Unsicherheiten in den Eingangsgrößen werden als Datenfehler bezeichnet. Eng mit diesem Fehler verbunden, ist die Fragestellung, wie sich eine Störung (kleine Veränderung) der Eingangsdaten auf das berechnete Ergebnis auswirkt (Störeffekte). Als stabiles oder gut konditioniertes Problem bezeichnet man eine Aufgabenstellung, wo kleine Datenfehler nur kleine Änderungen im Ergebnis bewirken. Führen kleine Datenfehler zu großen Verschiebungen des Ergebnisses so spricht man von schlecht konditionierten Systemen.

Eine zweite Gruppe von Fehlern werden unter dem Begriff Verfahrensfehler zusammengefaßt. Bei Näherungsverfahren und iterativen Verfahren aber auch bei direkten Verfahren wird der Bearbeiter vielfach mit der Aufgabe konfrontiert, das Verfahren bis zu einer bestimmten Genauigkeit voranzutreiben. Durch die Wahl von Abbruchskriterien wird ein Restfehler hervorgerufen der dem Verfahren und der Aufgabenstellung eigen ist. Aber auch die zuvor gestellte Frage nach der Wahl der Schranke für verschwindende Eigenwerte oder die Festlegung eines Rangdefektes ist dieser Gruppe zuzuordnen.

Vollkommen unabhängig von diesen beiden Gruppen sind die bei den numerischen Prozessen auftretenden Rundungsfehler. Lediglich von der Wahl und der Implementierung der Rechenverfahren abhängig, werden diese Fehler bei der digitalen Darstellung der Zahlen durch eine beschränkte Anzahl von signifikanten Ziffern hervorgerufen. Der durch Runden oder Abschneiden hervorgerufene maximale Fehler bei der Darstellung einer Zahl im Rechner wird als relative Maschinengenauigkeit bezeichnet. Abgesehen von wenigen Ausnahmen erfolgt die Darstellung einer Zahl in digitaler Form immer bezüglich einer Basis β mit einer bestimmten, aber festen Anzahl von Mantissenstellen τ (Ausnahme z.B. APL). Die relative Maschinengenauigkeit ε ist dadurch definiert, daß keine Unterscheidung der Größen x im Bereich von $x \pm \varepsilon$ im Rechner erfolgen kann. Somit ist ε durch

$$\varepsilon = \frac{1}{2}\beta^{-\tau+1} \geq \frac{|x - x_M|}{|x|} \tag{2.25}$$

definiert, wobei x die darzustellende Zahl und x_M die Maschinenzahl in digitaler Form angibt. Die Abschätzung der einem numerischen Verfahren zugrunde liegenden Rundungsfehler ist unabhängig von der Stabilität der Aufgabenstellung. Die Auswirkung

von Rundungsfehler kann jedoch bei schlecht konditionierten Problemen die Ergebnisse unbrauchbar machen.

Zwei vollkommen verschiedene Zugänge zur Analyse aller auftretenden Fehler und deren Einfluß auf die folgenden Operationen sind üblich. Die Methode der *Vorwärtsanalyse* versucht die Entstehung der Fehler schrittweise nachzuvollziehen und die Auswirkungen auf das Ergebnis abzuschätzen. Die vielfach einfachere Methode ergibt sich durch die Umkehrung der Fragestellung. Das errechnete verfälschte Ergebnis $\hat{\boldsymbol{x}}$ wird als strengen Ergebnis einer gestörten Aufgabenstellung

$$(\boldsymbol{N} + \Delta\boldsymbol{N})\,\hat{\boldsymbol{x}} = \boldsymbol{b} \tag{2.26}$$

interpretiert. Die Abschätzung der Größenordnung der entstandenen Störeffekte $\Delta\boldsymbol{N}$ wird wird als *Rückwärtanalyse* bezeichnet. Diese Störeffekte werden wie Datenfehler behandelt, wodurch die Beeinflussung des Ergebnisses abgeschätzt werden kann.

Eine zentraler Punkt der Fehlerabschätzung betrifft die Ermittlung der Störeffekte, also der Abschätzung des Einflusses einer Änderung der Matrix $\boldsymbol{N}$ auf den Lösungsvektor $\boldsymbol{x}$. Stellt man eine diese Änderung der Matrix N als kleine Änderung ϵ ($\lambda_r \geq \epsilon \geq 0$) in Richtung eines beliebigen Eigenvektors $\boldsymbol{v}^{(i)}$ dar, so ergibt sich die modifizierte Matrix $\boldsymbol{N}_\epsilon$ durch

$$\boldsymbol{N}_\epsilon = \boldsymbol{N} + \epsilon\ \boldsymbol{v}^{(i)}\boldsymbol{v}^{(i)^T}\ , \tag{2.27}$$

Der geänderte Lösungsvektor $\boldsymbol{x}_\epsilon$ errechnet sich durch

$$\boldsymbol{x}_\epsilon = \left(\boldsymbol{N} + \epsilon\ \boldsymbol{v}^{(i)}\boldsymbol{v}^{(i)^T}\right)^{-1}\boldsymbol{b}\ . \tag{2.28}$$

Die Inverse kann durch Anwendung der *Sherman-Morrison-Woodburg* Formel (1.53) in

$$\boldsymbol{x}_\epsilon = \boldsymbol{x} - \epsilon\frac{1}{\lambda_i}\,\boldsymbol{v}^{(i)}\left(1 + \epsilon\frac{1}{\lambda_i}\right)^{-1}\boldsymbol{v}^{(i)^T}\boldsymbol{x} \tag{2.29}$$

aufgespalten werden. Der verbleibende skalare Ausdruck kann in eine Reihe entwickelt werden. Unter Beachtung der Normiertheit der Eigenvektoren errechnet sich

$$\boldsymbol{x}_\epsilon = \boldsymbol{x} - \frac{\epsilon}{\lambda_i}\left(1 - \frac{\epsilon}{\lambda_i} + \left(\frac{\epsilon}{\lambda_i}\right)^2 - \ldots\right)^{-1}\boldsymbol{x} \tag{2.30}$$

Die Änderung des Lösungsvektors $\Delta\boldsymbol{x}_\epsilon = \boldsymbol{x} - \boldsymbol{x}_\epsilon$ auf Grund der Änderung in Richtung des kleinsten Eigenvektors $\boldsymbol{v}^{(r)}$ ergibt sich näherungsweise durch

$$\Delta\boldsymbol{x}_\epsilon \approx \frac{\epsilon}{\lambda_r}\,\boldsymbol{x}\ , \qquad \lambda_r \gg \epsilon\ . \tag{2.31}$$

Beachtet man, daß die Änderung maximal ist, wenn sie in der Richtung des kleinsten Eigenvektors zu liegen kommt, so kann (2.16) als oberer Schranke genommen werden, wenn eine gleichgroße Änderung in einer beliebigen Richtung vorliegt,

$$\Delta \boldsymbol{x}_\epsilon \leq \frac{\epsilon}{\lambda_r} \boldsymbol{x} \ , \qquad \lambda_r \gg \epsilon \ . \tag{2.32}$$

Durch diese Formel kann eine Genauigkeitsabschätzung des absoluten Fehlers in der Lösung auf Grund eines absoluten Fehlers in der Matrix angegeben werden. Man erkennt dabei, daß große Koeffizienten im Lösungsvektor große absolute Fehler verursachen. Betrachtet man die erste Berechnung als vorläufige Lösung oder führt eine Näherungslösung $\boldsymbol{x}_0$ so kann die Abweichung $\Delta \boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_0$ zur exakten Lösung $\boldsymbol{x}$ aus

$$\boldsymbol{N} \left(\boldsymbol{x}_0 + \Delta \boldsymbol{x} \right) = \boldsymbol{b} \tag{2.33}$$

durch Umformung und Modifikation des Konstantenvektors

$$\boldsymbol{N} \Delta \boldsymbol{x} = \boldsymbol{b} - \boldsymbol{N} \boldsymbol{x}_0 = \boldsymbol{b} - \boldsymbol{b}_0 = \Delta \boldsymbol{b} \tag{2.34}$$

errechnet werden. Die Differenz $\Delta \boldsymbol{b}$ des Konstantenvektors $\boldsymbol{b}$ minus der aus der Näherungslösung $\boldsymbol{x}_0$ berechneten Werten $\boldsymbol{b}_0$ weist auf die Nichterfüllung des Gleichungssystems hin. Diese wegen der zu erwartenden Ziffernauslöschung numerisch schwierig zu ermittelnden Größen sind bis auf das Vorzeichen ident mit den Residuen $\boldsymbol{r}$. Da nur kleine Zuschläge zur Näherungslösung erwartet werden und (2.32) auf diese Zuschläge anzuwenden ist, kann durch diesen Vorgang, der als Nachiteration bezeichnet wird, eine erhöhte absolute Genauigkeit der Lösung erreicht werden.

Wenn die Voraussetzung $\lambda_r \gg \epsilon$ nicht erfüllt ist, ist zu beachten, daß (2.32) nur näherungsweise Gültigkeit hat. Sobald ϵ ungefähr gleich λ_r wird, ist die Berechnung sinnlos, da die Fehler der Lösung signifikant werden. Nach der Überprüfung des Modells ist die Entscheidung, ob die Berechnung mit erhöhter Genauigkeit (Senkung von ϵ) durchgeführt wird oder ob ein Rangdefekt vorliegt, unumgänglich.

Eine Möglichkeit der Abschätzung des relativen Fehlers ist durch den Ansatz

$$\left(\boldsymbol{N} + \Delta \boldsymbol{N} \right) \left(\boldsymbol{x} + \Delta \boldsymbol{x} \right) = \boldsymbol{b} + \Delta \boldsymbol{b} \tag{2.35}$$

möglich, wobei mit Δ die absoluten Fehler der einzelnen Größen gekennzeichnet sind. Durch Umformung erlangt man

$$\Delta \boldsymbol{x} = -\boldsymbol{N}^{-1} \left(\Delta \boldsymbol{N} \boldsymbol{x} - \Delta \boldsymbol{b} \right) + \mathcal{O}(\Delta^2) \tag{2.36}$$

Betrachtet man die normierte Form

$$|\Delta \boldsymbol{x}| \leq \left| \boldsymbol{N}^{-1} \right| \left(|\Delta \boldsymbol{N}| \, |\boldsymbol{x}| + |\Delta \boldsymbol{b}| \right) \tag{2.37}$$

und erweitert rechts um $\frac{|\boldsymbol{N}|}{|\boldsymbol{N}|}$, so ergibt sich

$$|\Delta \boldsymbol{x}| \leq |\boldsymbol{N}| \left|\boldsymbol{N}^{-1}\right| \left(\frac{|\Delta \boldsymbol{N}|}{|\boldsymbol{N}|} |\boldsymbol{x}| + \frac{|\Delta \boldsymbol{b}|}{|\boldsymbol{N}|} \right) \ . \tag{2.38}$$

Geht man auf den relativen Fehler in $\boldsymbol{x}$ über und beachtet

$$|\boldsymbol{b}| \leq |\boldsymbol{N}| \, |\boldsymbol{x}| \tag{2.39}$$

(*Cauchy-Schwartz'sche Ungleichung*) erhält man

$$\frac{|\Delta \boldsymbol{x}|}{|\boldsymbol{x}|} \leq |\boldsymbol{N}| \left|\boldsymbol{N}^{-1}\right| \left(\frac{|\Delta \boldsymbol{N}|}{|\boldsymbol{N}|} + \frac{|\Delta \boldsymbol{b}|}{|\boldsymbol{b}|} \right) \ . \tag{2.40}$$

Die Form $|\boldsymbol{N}| \left|\boldsymbol{N}^{-1}\right|$ wird zusammengefaßt und als *Konditionszahl* $\kappa(\boldsymbol{N})$ bezeichnet. Bei Anwendung der L_2-Norm errechnet sich $\kappa(N)$

$$\kappa_2(\boldsymbol{N}) = |\boldsymbol{N}|_2 \left|\boldsymbol{N}^{-1}\right|_2 = \frac{\lambda_1}{\lambda_n} = \frac{\lambda_{max}}{\lambda_{min}} \tag{2.41}$$

durch die Division des größten (λ_{max}) durch den kleinsten (λ_{min}) Eigenwert. Wenn die Konditionszahl sehr groß ist, bezeichnet man das System als *„schlecht konditioniert"*. Die relative Genauigkeit der Lösung wird bestimmt durch die relativen Genauigkeiten der Matrix und der Konstantenspalte multipliziert mit der Konditionszahl $\kappa(\boldsymbol{N})$. Die für praktische Anwendungen ausreichende Gleichung (2.40) stellt wegen der Vernachlässigung der Glieder 2. Ordnung nur eine Näherung der exakten Abschätzung

$$\frac{|\Delta \boldsymbol{x}|}{|\boldsymbol{x}|} \leq \frac{\kappa(\boldsymbol{N})}{1 - \kappa(\boldsymbol{N}) \frac{|\Delta \boldsymbol{N}|}{|\boldsymbol{N}|}} \left(\frac{|\Delta \boldsymbol{N}|}{|\boldsymbol{N}|} + \frac{|\Delta \boldsymbol{b}|}{|\boldsymbol{b}|} \right) \ . \tag{2.42}$$

dar (siehe z.B. *H.R. Schwarz (1988)[78]*, Seite 33-34).

2.3 Konditionsverbesserung

Wie aus den Betrachtungen des Abschnittes 2.2 hervorgeht, hängt die Stabilität der Lösung wesentlich von der Konditionszahl $\kappa(\boldsymbol{N})$ ab. Die relative Genauigkeit der Eingangsgrößen wird mit der Konditionszahl multipliziert und legt somit die relative Genauigkeit der Ausgangsgrößen fest. Eine Verbesserung der Kondition entspricht einer Verringerung der Konditionszahl, das bedeutet, daß das Verhältnis des größten zum kleinsten Eigenwert verkleinert wird. Für die direkten Lösungsmethoden bewirkt dies eine erhöhte Genauigkeit bei der Berechnung, wodurch sich eine Nachiteration erübrigen kann. Bei den iterativen Methoden kann vor allem bei schlecht konditionierten

Systemen eine wesentliche Beschleunigung der Konvergenz herbeigeführt werden. Die zugrunde liegende Idee bei der Konditionsverbesserung beruht darauf, das Gleichungssystem

$$\boldsymbol{N}\boldsymbol{x} = \boldsymbol{b} \tag{2.43}$$

durch eine lineare Abbildung zu stabilisieren. Diese Abbildung wird durch eine Linksmultiplikation mit der Inversen einer regulären Matrix $\boldsymbol{C}$ bewerkstelligt

$$\boldsymbol{C}^{-1}\boldsymbol{N}\boldsymbol{x} = \boldsymbol{C}^{-1}\boldsymbol{b} \; . \tag{2.44}$$

Die Matrix $\boldsymbol{C}$ gilt es nun so zu wählen, daß eine die Konditionszahl des umgeformten Systems (2.44) gegenüber dem Originalsystem (2.43) möglichst klein wird. Die rechenintensivste aber idealste Wahl wäre $\boldsymbol{C} = \boldsymbol{N}$, denn das zu lösende Gleichungssystem wäre somit ident der Einheitsmatrix und besäße folglich eine Konditionszahl $\kappa(\boldsymbol{C}^{-1}\boldsymbol{N}) = 1$. Ziel ist es, eine möglichst einfach zu berechnende Matrix C zu finden, wobei sich bei symmetrischen, positiv definiten Matrizen die Zerlegung in die Hilfsmatrizen H und ihre Transponierte H^T anbietet

$$\boldsymbol{C} = \boldsymbol{H}^T\boldsymbol{H} \tag{2.45}$$

Setzt man für $\boldsymbol{C}^{-1}$ in (2.44) ein, so erlangt man

$$\boldsymbol{H}^{-1}\left(\boldsymbol{H}^T\right)^{-1}\boldsymbol{N}\boldsymbol{x} = \boldsymbol{H}^{-1}\left(\boldsymbol{H}^T\right)^{-1}\boldsymbol{b} \; . \tag{2.46}$$

Eliminiert man $\boldsymbol{H}^{-1}$ durch Linksmultiplikation mit $\boldsymbol{H}$ und erweitert die linke Seite um $\boldsymbol{H}^{-1}\boldsymbol{H} = \boldsymbol{I}$, so errechnet sich

$$\left(\boldsymbol{H}^T\right)^{-1}\boldsymbol{N}\boldsymbol{H}^{-1}\boldsymbol{H}\boldsymbol{x} = \left(\boldsymbol{H}^T\right)^{-1}\boldsymbol{b} \; . \tag{2.47}$$

Durch Einführung von

$$\hat{\boldsymbol{N}} = \left(\boldsymbol{H}^T\right)^{-1}\boldsymbol{N}\boldsymbol{H}^{-1} \tag{2.48}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x} \tag{2.49}$$

$$\hat{\boldsymbol{b}} = \left(\boldsymbol{H}^T\right)^{-1}\boldsymbol{b} \tag{2.50}$$

kann (2.47) vereinfacht geschrieben werden mit

$$\hat{\boldsymbol{N}}\hat{\boldsymbol{x}} = \hat{\boldsymbol{b}} \; , \tag{2.51}$$

wobei die Symmetrie wegen (2.48) erhalten bleibt. Es muß noch bewiesen werden, daß die Umformung (2.46) auf (2.47) die Eigenwerte der Matrix $\boldsymbol{C}^{-1}\boldsymbol{N}$ nicht verändert

hat, daß also $\boldsymbol{N}$ und $\boldsymbol{C}^{-1}N$ die gleichen Eigenwerte besitzen. Durch Erweiterung von (2.44) um die Einheitsmatrix $\boldsymbol{I} = \boldsymbol{H}^{-1}\boldsymbol{H}$ errechnet sich

$$\boldsymbol{C}^{-1}\boldsymbol{N} = \boldsymbol{H}^{-1}\left(\boldsymbol{H}^{T}\right)^{-1}\boldsymbol{N} = \boldsymbol{H}^{-1}\left(\boldsymbol{H}^{T}\right)^{-1}\boldsymbol{N}\boldsymbol{H}^{-1}\boldsymbol{H} \ . \tag{2.52}$$

Verwendet man (2.48) so erlang man

$$\boldsymbol{C}^{-1}\boldsymbol{N} = \boldsymbol{H}^{-1}\hat{\boldsymbol{N}}\boldsymbol{H} \tag{2.53}$$

und führt eine Spektralzerlegung der einzelnen Matrizen von (2.53) durch, so erhält man pro Matrix eine Summe von dyadischen Produkten (siehe (2.9)). Wegen der Kommutativität der skalaren Multiplikation eliminiert sich der Einfluß von $\boldsymbol{H}^{-1}$ und $\boldsymbol{H}$, wodurch gezeigt wird, daß $\boldsymbol{C}^{-1}\boldsymbol{N}$ und $\boldsymbol{N}$ zueinander ähnlich sind, also die gleichen Eigenwerte besitzen. Bei der Wahl von $\boldsymbol{H}$ und damit von $\boldsymbol{C}$ steht die Verbesserung der Kondition im Vordergrund, die dann erreicht wird, wenn $\boldsymbol{C}$ die Eigenschaften von $\boldsymbol{N}$ möglichst widerspiegelt. Der Rechenaufwand hält sich in Grenzen, wenn eine leichte Berechenbarkeit der Formeln (2.48) bis (2.50) gewährleistet wird. Neben der Berechnung von (2.48) ist zu beachten ist, daß zwei Systeme mit der Hilfsmatrix $\boldsymbol{H}$ gelöst werden müssen. Zum einen muß der modifizierte Konstantenvektor $\hat{\boldsymbol{b}}$ aus (2.50) berechnet werden und zum anderen erfordert die Umrechnung der aus (2.51) berechneten modifizierten Lösung $\hat{\boldsymbol{x}}$ zum Lösungsvektor $\boldsymbol{x}$ die Lösung von (2.49). Man erkennt somit, daß eine Dreiecks-, Band- oder Diagonalmatrix eine rechengünstige Wahl bedeutet. Wählt man das einfachere Modell der Diagonalmatrix, so spricht man von *Skalierung*. Das aufwendigere Modell mit Matrizen allgemeineren Typs (Dreiecks-, Bandmatrizen) wird als *Vorkonditionierung* bezeichnet und vor allem bei iterative Methoden verwendet, da die Berechnung von (2.48) dabei nicht explizit erfolgen muß (siehe Abschnitt 2.3.2 bzw. 2.5.1.3).

2.3.1 Skalierung

Eine optimale Skalierung kann so gewählt werden, daß die Zeilen und Spalten der Matrix äquilibriert sind. Dies bedeutet, daß die euklidischen Normen von allen Zeilen und Spalten gleich Eins sind. Diese optimalen Skalierungsfaktoren ergeben sich aus den nichtlinearen Bedingungen

$$\left| \frac{1}{h_{ii}^2} \sum_{j=1}^{n} n_{ij}^2 \frac{1}{h_{jj}^2} \right| = 1 \ , \qquad i = 1, \ldots, n \tag{2.54}$$

$$\left| \frac{1}{h_{jj}^2} \sum_{i=1}^{n} n_{ij}^2 \frac{1}{h_{ii}^2} \right| = 1 \ , \qquad j = 1, \ldots, n \tag{2.55}$$

Die Ermittlung der Faktoren stellt somit eine rechenintensive Aufgabe dar. Für praktische Zwecke begnügt man sich mit einer einfach zu realisierenden Wahl der Skalierungsfaktoren

$$h_{ii} = \sqrt{n_{ii}} \ . \tag{2.56}$$

Die skalierte Matrix $\hat{\boldsymbol{N}}$ besitzt damit nur Diagonalelemente $\hat{n}_{ii} = 1$. Starke Größenunterschiede in den Diagonalelementen bewirken unweigerlich eine große Konditionszahl, da für den größten (kleinsten) Eigenwert das größte (kleinste) Diagonalenelement als untere (obere) Schranken gilt. Für die Konditionszahl $\kappa_2(\boldsymbol{N})$ hat somit die Abschätzung

$$\kappa_2(N) = \frac{\lambda_{max}}{\lambda_{min}} \geq \frac{(n_{ii})_{max}}{(n_{ii})_{min}} \tag{2.57}$$

Gültigkeit. Hinzuweisen wäre noch darauf, daß es vielfach nicht notwendig und oft auch nicht sinnvoll ist, die skalierte Matrix $\hat{\boldsymbol{N}}$ explizit zu berechnen. Die Skalierungsfaktoren h_{ii} können oft implizit in den Rechenvorgang übernommen werden und z.B. bei der Pivotierung berücksichtigt werden. Variieren trotz der Skalierung mit (2.56) die Zeilen- und Spaltennormen sehr stark, so weist auch die skalierte Matrix keine sehr stabile Form auf.

Geometrisch bedeutet die Rechtsmultiplikation von $\boldsymbol{N}$ mit der Diagonalmatrix $\boldsymbol{H}^{-1}$ eine Einführung von unterschiedlichen Maßstäben für die einzelnen Spaltenvektoren. Diese Spaltenvektoren spannen den Raum auf, durch dessen Linearkombination der Vektor $\hat{\boldsymbol{b}} = \boldsymbol{H}^{-1}\boldsymbol{b}$ dargestellt werden soll. Eine bestmögliche numerische Darstellung wäre möglich, wenn diese Vektoren (Spalten) die gleiche Länge hätten und den Raum bestmöglich *ausleuchten*. Die erste Forderung ist durch die Skalierung erfüllt. Die optimale Erfüllung der zweite Forderung würde bedeuten, daß alle linear unabhängigen Spaltenvektoren paarweise aufeinander orthogonal sein sollten. Diese Forderung bleibt bei der Skalierung unberücksichtigt.

2.3.2 Vorkonditionierung

Die Mängel, die durch das vereinfachte Modell der Diagonalmatrix bei der Skalierung auftreten, können durch rechenaufwendigere Modelle mit Dreiecksmatrizen beseitigt werden. Die Wahl der Konditionierungsmatrix $\boldsymbol{C}$ ist geprägt durch die Forderungen, daß $\boldsymbol{C}$ möglichst die Eigenschaften von $\boldsymbol{N}$ widerspiegeln soll und die Berechnung (2.48), (2.49) und (2.50) einfach sein soll. Durch die Kenntnis des Verhaltens der Inversen von $\boldsymbol{N}$ durch theoretische Voruntersuchungen (Studium von regelmäßigen Netzen, Kriteriumsmatrizen) können problemorientiert zufriedenstellende Konditionierungsmatrizen gefunden werden. Liegen keine Vorinformationen über das Verhalten der Matrix $\boldsymbol{N}$ vor, so kann eine näherungsweise Berechnung zum Beispiel durch Berücksichtigung eines Bandes erfolgen.

Bei schwachbesetzten Matrizen verwendet man eine *partielle Cholesky-Zerlegung*, um eine geeignete Matrix $\boldsymbol{C}$ zu berechnen. Die obere Dreiecksmatrix $\boldsymbol{H}$ wird dabei so berechnet, daß eine Cholesky-Zerlegung durchgeführt wird, wobei alle Füllelemente unberücksichtigt bleiben (siehe Kapitel 4). Die Dreiecksmatrix $\boldsymbol{H}$ weist somit dieselbe Besetzungsstruktur wie die gegebene Matrix $\boldsymbol{N}$ auf und kann in analoger Weise komprimiert gespeichert werden. Bei der Durchführung der partiellen Cholesky-Zerlegung kann es zu Problemen kommen, da nicht gewährleistet ist, daß die partielle Zerlegung

existiert. Tritt ein negativer Radikand auf, wird empfohlen, vor Beginn der Zerlegung die Außendiagonalelemente durch Multiplikation mit $(1-\alpha)$, $0 \leq \alpha \leq 1$ zu verkleinern. Die Güte der Konditionsverbesserung ist abhängig von der Wahl des Parameters α. Je kleiner der Parameter gewählt ist, umso besser ist die Wirkung der Konditionsverbesserung (siehe *H.R. Schwarz (1981)[77]*, Seite 114).

Die häufige Verwendung der Vorkonditionierung bei iterativen Verfahren hat zwei Ursachen. Einerseits ist bei vielen Problemstellungen eine enorme Konvergenzsteigerung zu beobachten und andererseits kann durch die Möglichkeit, den Vorkonditionierungsprozeß in den Iterationsprozeß einzubinden eine sehr ökonomische Berechnung gefunden werden. Im Abschnitt 2.5.1.3 wird diese Einbindung an Hand der Methode der konjugierten Gradienten aufgezeigt.

2.4 Direkte Methoden

Die im Abschnitt 1 bearbeitet Verfahren, das Austauschverfahren in Standardform und in der verallgemeinerten Form sind geeignete Verfahren für die Berechnung der gesamten Inversen beziehungsweise der Pseudoinversen. Hier sollen in erster Linie Verfahren aufgezeigt werden, die sich speziell für die Lösung also nicht notwendigerweise die Inversion von symmetrischen, positiv definiten, linearen Gleichungssystemen eignen. Das Grundprinzip des Großteils der Verfahren ist ident. Das aufzulösende Gleichungssystem wird in leicht zu berechnende und lösende Dreiecksmatrizen und Diagonalmatrizen zerlegt. Durch Anwendung des Gauß'schen Eliminationsverfahrens nur unterhalb der Hauptdiagonale kann die Matrix in eine untere (linke,left) und obere (rechte,upper) Dreiecksmatrix zerlegt werden (LR- oder LU-Verfahren). Für symmetrische, reguläre Systeme kann eine Dreieckszerlegung durchgeführt werden, wobei die untere Dreiecksmatrix L die Transponierte der oberen Dreiecksmatrix $\boldsymbol{R}$ darstellt ($\boldsymbol{R}^T\boldsymbol{R}$- oder $\boldsymbol{L}\boldsymbol{L}^T$-Verfahren). Dieses *Verfahren von Cholesky* benötigt für die Berechnung die Quadratwurzeln der Diagonalglieder und wird daher auch vielfach als *Quadratwurzelverfahren* bezeichnet. Bei nicht positiv definiten Systemen verursacht die Berechnung der Quadratwurzeln einen Mehraufwand, da bei negativen Diagonalgliedern die Beachtung und Mitführung der imaginären Werte erforderlich ist. Um die Verwendung von imaginären Werten zu umgehen, existieren spezielle Algorithmen für nicht positiv definite Systeme. Anstatt einer $\boldsymbol{R}^T\boldsymbol{R}$-Zerlegung wird von *Parlett, Reid und Aasen* eine Zerlegung in $\boldsymbol{R}^T\boldsymbol{T}\boldsymbol{R}$ mit der tridiagonalen Matrix T konstruiert (siehe *W. Bunse et.al. (1985)[14]*, Seite 69-72).

Für symmetrische und positiv definite Systeme, wie sie in der Ausgleichsrechnung vorliegen, sind die Quadratwurzeln nur aus positiven Werten zu berechnen, sodaß keine imaginären Teile auftreten (siehe *H.R. Schwarz (1988)[78]*, Satz 1.1, Seite 38). Die einfache und rasche Berechnung und die guten Stabilitätseigenschaften sind die Hauptargumente dafür, daß sich dieses Verfahren für symmetrische, positiv definite Matrizen als Standardverfahren durchgesetzt hat.

2.4.1 Verfahren von Cholesky

2.4.1.1 Grundprinzip

Gesucht ist die Lösung $\boldsymbol{x}$ des positiv definiten, symmetrischen Gleichungssystems

$$\boldsymbol{N}\boldsymbol{x} = \boldsymbol{b} \; . \tag{2.58}$$

Die Matrix N läßt sich in zwei Dreiecksmatrizen R zerlegen

$$\boldsymbol{N} = \boldsymbol{R}^T\boldsymbol{R} \; , \tag{2.59}$$

wobei $\boldsymbol{R}$ eine obere Dreiecksmatrix bildet. Damit geht das lineare Gleichungssystem (2.58) in das System

$$\boldsymbol{R}^T\boldsymbol{R}\boldsymbol{x} = \boldsymbol{b} \tag{2.60}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

über. Durch Substitution von

$$\boldsymbol{R}\boldsymbol{x} = \boldsymbol{y} \tag{2.61}$$

in Gleichung (2.60) erlangt man ein System mit unterer Dreiecksstruktur (siehe Abb. 2.1)

$$\boldsymbol{R}^T\boldsymbol{y} = \boldsymbol{b}\ . \tag{2.62}$$

Wegen der Dreiecksstruktur kann, bei bekannten Koeffizienten der Dreiecksmatrix $\boldsymbol{R}$ und des Vektors $\boldsymbol{b}$, leicht nach dem unbekannten Hilfsvektor $\boldsymbol{y}$ in auf steigender Reihenfolge entwickelt werden. Von der ersten Zeile beginnend errechnen sich die Unbekannten y_1, y_2 usw. durch

$$y_1 = b_1/r_{11} \tag{2.63}$$

$$y_2 = (b_2 - r_{12}y_1)\,/r_{22} \tag{2.64}$$

$$y_3 = (b_3 - r_{13}y_1 - r_{23}y_2)\,/r_{33}\ . \tag{2.65}$$

Allgemein formuliert ergibt sich

$$y_i = \left(b_i - \sum_{k=1}^{i-1} r_{ki}y_k\right)/r_{ii}\ , \qquad i = 1,\ldots,n\ . \tag{2.66}$$

Da die Berechnung in der ersten Zeile begonnen wird, wird dieser Teilschritt der Prozeß des *Vorwärtseinsetzens* genannt.

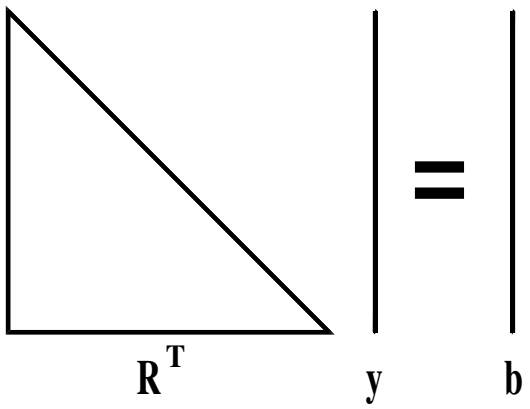


Abbildung 2.1: Darstellung der Struktur des Gleichungssystems (2.62)

Nach der Berechnung des Hilfsvektors $\boldsymbol{y}$ können mit Hilfe der Formel (2.61) die Elemente des Unbekanntenvektors x_i sehr einfache Weise ermittelt werden. Da das System

eine obere Dreiecksstruktur aufweist (siehe Abb. 2.2) beginnt man am besten bei der letzten Zeile, woraus sich

$$x_n = y_n / r_{nn} \tag{2.67}$$

ergibt. Benützt man die soeben berechneten Koeffizienten und setzt in die jeweils vorhergehende Zeile ein, so erlangt man

$$x_{n-1} = \left(y_{n-1} - r_{n-1,n} x_n\right) / r_{n-1,n-1} \tag{2.68}$$

$$x_{n-2} = \left(y_{n-2} - r_{n-2,n-1} x_{n-1} - r_{n-2,n} x_n\right) / r_{n-2,n-2} \tag{2.69}$$

beziehungsweise allgemein

$$x_i = \left(y_i - \sum_{k=i+1}^{n} r_{ik} x_k\right) / r_{ii} \tag{2.70}$$

Aufgrund der vom Ende beginnenden Berechnung wird dieser Prozeß als *Rückwärtseinsetzen* bezeichnet.

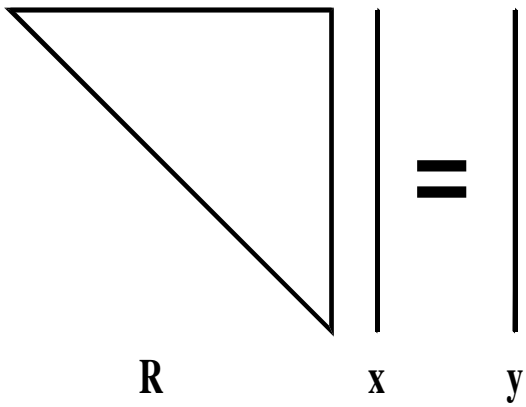


Abbildung 2.2: Darstellung der Struktur des Gleichungssystems (2.61)

Die Glieder der Dreiecksmatrix $\boldsymbol{R}$ lassen sich aus der Identität (2.59) errechnen. Diese Identität - in (2.71) ausführlich dargestellt

$$\begin{bmatrix} n_{11} & n_{12} & \dots & n_{1n} \\ n_{12} & n_{22} & \dots & n_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ n_{1n} & n_{2n} & \dots & n_{nn} \end{bmatrix} = \begin{bmatrix} r_{11} & & & \\ r_{12} & r_{22} & & \\ \vdots & \vdots & \ddots & \\ r_{1n} & r_{2n} & \dots & r_{nn} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix} \tag{2.71}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

führt nach der Ausmultiplikation auf n^2 - beziehungsweise aus Symmetriegründen auf $\frac{1}{2}n(n+1)$ unabhängige - Gleichungen, aus denen die Koeffizienten der Dreiecksmatrix ermitteln lassen. Beginnend mit

$$\begin{array}{lllllll}
n_{11} = r_{11}^2 & & & & & & [1,1] \\
n_{12} = r_{11}r_{12} & & & & & & [1,2] \\
n_{22} = r_{12}^2 & + & r_{22}^2 & & & & [2,2] \\
n_{13} = r_{11}r_{13} & & & & & & [1,3] \\
n_{23} = r_{12}r_{13} & + & r_{22}r_{23} & & & & [2,3] \\
n_{33} = r_{13}^2 & + & r_{23}^2 & + & r_{33}^2 & & [3,3] \\
\quad \ldots & & & & & &
\end{array} \tag{2.72}$$

errechnen sich aus $[1,1], [1,2], \ldots$ usw.

$$\begin{array}{lll}
[1,1] & \Longrightarrow & r_{11}{=}\sqrt{n_{11}} \\
[1,2] & \Longrightarrow & r_{12}{=}n_{12}/r_{11} \\
[2,2] & \Longrightarrow & r_{22}{=}\sqrt{n_{22} - r_{12}^2} \\
[1,3] & \Longrightarrow & r_{13}{=}n_{13}/r_{11} \\
[2,3] & \Longrightarrow & r_{23}{=}\left(n_{23} - r_{12}r_{13}\right)/r_{22} \\
[3,3] & \Longrightarrow & r_{33}{=}\sqrt{n_{33} - r_{13}^2 - r_{23}^2}
\end{array} \tag{2.73}$$

Dies führt zur allgemeinen Darstellung

$$n_{ij} = \sum_{k=1}^{i} r_{ki}r_{kj} \tag{2.74}$$

beziehungsweise

$$n_{ii} = \sum_{k=1}^{i} r_{ki}^2 \tag{2.75}$$

woraus sich

$$\begin{array}{ll}
r_{ij} = \left(n_{ij} - \sum\limits_{k=1}^{i-1} r_{ki}r_{kj}\right)/r_{ii} & i = 1, \ldots, n \\
 & j = i+1, \ldots, n \\
r_{ii} = \sqrt{n_{ii} - \sum\limits_{k=1}^{i-1} r_{ki}^2} & i = 1, \ldots, n
\end{array} \tag{2.76}$$

rückrechnen lassen. Der Übergang von der symmetrischen Matrix $\boldsymbol{N}$ auf die obere Dreiecksmatrix wird im folgenden als *Cholesky-Reduktion* bezeichnet. Für weiter Betrachtungen, insbesondere die Anwendungen bei schwach besetzten Systemen (Abschnitt 4.5.1) und bei großen Systemen (Abschnitt 5.2), ist es notwendig die Bildung der Choleskyreduktion durch die Formeln (2.76) näher zu durchleuchten. Zur Berechnung des

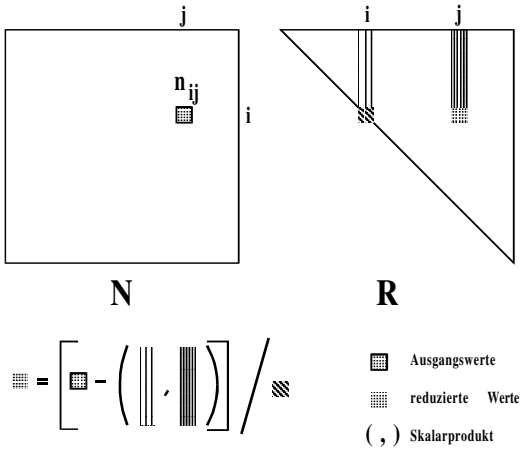


Abbildung 2.3: Darstellung des Berechnungsvorganges eines Nichtdiagonalgliedes r_{ij}, $i < j$ bei der Cholesky-Reduktion. Formel (2.76/1)

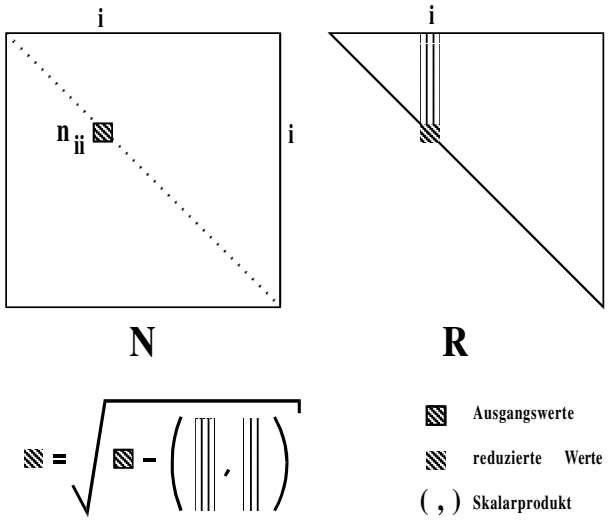


Abbildung 2.4: Darstellung des Berechnungsvorganges eines Diagonalgliedes r_{ii} bei der Cholesky-Reduktion. Formel (2.76/2)

Gliedes r_{ij}, $i < j \leq n$ wird das unreduzierte Glied n_{ij}, das innere Produkt der reduzierten Spalte i und j vom ersten bis zum $(i-1)$-ten Glied und das reduzierte Diagonalelement r_{ii} benötigt (Abb. 2.3).

Zur Reduktion des Diagonalgliedes r_{ii} benötigt man das unreduzierte Glied n_{ii} und das innerer Produkt der reduzierten Spalte i vom ersten bis zum $(i-1)$-ten Glied mit sich selbst.

Man erkennt, daß lediglich ein nichtreduziertes Glied nämlich n_{ij} bzw. n_{ii} zur Reduktion benötigt wird. Alle anderen Glieder müssen schon in reduzierter Form vorliegen. Beginnt man von links oben und reduziert spalten- oder zeilenweise fortschreitend nach

rechts unten, so sind diese Vorgaben immer erfüllt (siehe Abb. 2.5).

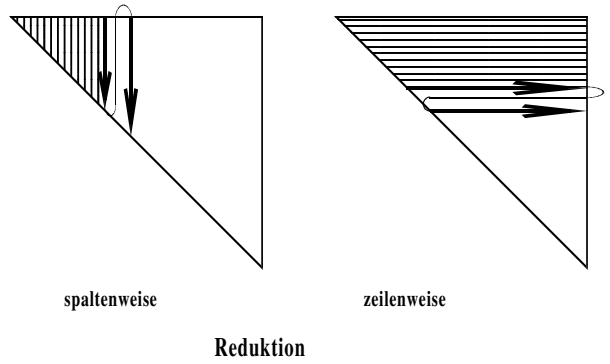


Abbildung 2.5: Darstellung des Ablauf der Choleskyreduktion

In seiner seiner klassischen Form kann das Cholesky-Verfahren in folgender Form zusammengefasst werden:

$$
\begin{aligned}
&DO \;\; i = 1, ..., n \\
&\qquad r_{ii} = \sqrt{n_{ii} - \sum_{k=1}^{i-1} r_{ki}^2} \\
&\qquad DO \;\; j = i+1, ..., n \\
&\qquad\qquad r_{ij} = (n_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj}) / r_{ii} \\
&\qquad END \; DO \;\; j \\
&END \; DO \;\; i
\end{aligned}
$$

Abbildung 2.6: Cholesky-Faktorisierung; klassische Form.

Die Reduktion kann somit ohne Probleme am Ort (in der oberen Dreieckshälfte) vorgenommen werden. Vielfach wird bei der numerischen Berechnung die Ähnlichkeit der Formeln (2.70) und (2.76/1) ausgenützt, indem der Konstantenvektor $\boldsymbol{b}$ als $(n+1)$-te Spalte an die Matrix N angefügt wird. Das Vorwärtseinsetzen erfolgt somit in einem Guß mit der Reduktion. Für den Prozeß des Rückwärtseinsetzens ist es vorteilhaft, nicht die Formel (2.70) direkt für jede Unbekannte x_i auszuwerten, sondern den Anteil jeder berechneten Unbekannten x_i in allen anderen Unbekannten x_k, $1 \leq k \leq i-1$ sofort zu eliminieren, da somit ein spaltenorientiertes Vorgehen ermöglicht wird.

Algorithmus 2.1 : Auflösung eines in einer Matrix gespeicherten, symmetrischen, positiv definiten Gleichungssystems durch das Verfahren von Cholesky.

Aufgabenstellung: Gegeben ist ein symmetrisches, positiv definites Gleichungssystem $\boldsymbol{N} \in \mathbb{R}^{n \times n}$, der Konstantenvektor $\boldsymbol{b}$ ist in der $(n+1)$-ten Spalte der Matrix $\boldsymbol{N}$ gespeichert. Gesucht ist die Lösung $\boldsymbol{x}$ des Systems, wenn das Gleichungssystem positiv definit ist. Andernfalls wird Verfahren abgebrochen.

Schnittstellen:

Eingabe:	`n(1:nn,1:nn)`	... Koeffizienten der symmetrischen Matrix **N**, nur Elemente $i \leq j$ (obere Dreiecksmatrix) erforderlich
	`nn`	... Anzahl der Zeilen und Spalten des Systems
	`n(1:nn,nn+1:nn+c)`	... ein oder mehrere Konstantenvektoren **b**
	`c`	... Anzahl der rechten Seiten
Ausgabe:	`n(1:nn,1:nn)`	... modifizierte Koeffizienten der Matrix $\boldsymbol{N}$
	`n(1:nn,nn+1:nn+c)`	... Lösungen des Gleichungssystems
Felder:	`n(nn,nn+c)`	

Hilfsproceduren: —

Ausgangssituation:

Matrix N				rechte Seiten b		
n_{11}	n_{12}	$\dots$	n_{1n}	b_{11}	$\dots$	b_{1c}
$\star$	n_{22}	$\dots$	n_{2n}	b_{21}	$\dots$	b_{2c}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$		$\vdots$
$\star$	$\star$	$\dots$	n_{1n}	b_{11}	$\dots$	b_{1c}

Anmerkung:
Es wird nur das obere Dreieck verwendet, der Teil unterhalb der Diagonale bleibt unverändert.

Endsituation:

Matrix N				rechte Seiten b		
r_{11}	r_{12}	$\dots$	r_{1n}	x_{11}	$\dots$	x_{1c}
$\star$	n_{22}	$\dots$	r_{2n}	x_{21}	$\dots$	x_{2c}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$		$\vdots$
$\star$	$\star$	$\dots$	r_{1n}	x_{11}	$\dots$	x_{1c}

Lösungsweg: Pseudocode

1. *Initialisiere den Spaltenindex für die Choleskyreduktion bzw. das Vorwärtseinsetzen (spaltenweise Reduktion):*
   `j=0 .`
2. *Erhöhe den Spaltenindex und initialisiere den Zeilenindex:*
   `j=j+1; i=0 .`

Wolf-Dieter Schuh
ITGB, Uni Bonn

3. *Erhöhe den Zeilenindex:*
   `i=i+1 .`
4. *Beginne die Reduktion des Gliedes in der i-ten Zeile und j-ten Spalte (Anmerkung: Da die Reduktion am Platz erfolgt, gilt $r_{k\ell} = n_{k\ell}$, für $k < i$, $\ell < j$):*
   `n(i,j) = n(i,j) - n(1:i-1,i) n(1:i-1,j) .`
5. *Überprüfe ob das zu reduzierende Glied ein Diagonalglied ist oder nicht und setze entsprechend fort:*
   `IF i=j GOTO 8: .`
6. *Reduziertes Glied stellt kein Diagonalglied dar. Vervollständige die Berechnung durch Berücksichtigung der Division:*
   `n(i,j) = n(i,j)/n(i,i) .`
7. *Beende entsprechend dem Zeilenindex den Vorgang der Reduktion bzw. des Vorwärtseinsetzens oder setze die Reduktion beim nächsten Glied dieser Spalte fort:*
   `IF (i=nn AND j=nn+c) GOTO 10:`
   `IF (i=nn AND j<nn+c) GOTO 2:`
   `GOTO 3: .`
8. *Reduziertes Glied ist ein Diagonalglied. Überprüfe das Diagonalglied auf positive Definitheit:*
   `IF n<num_Null GOTO Ende:` 'Keine positiv definite Matrix' .
9. *Vervollständige die Berechnung des Diagonalgliedes durch Berücksichtigung der Wurzel:*
   `n(i,i) =` $\sqrt{\texttt{n(i,i)}}$
   `GOTO 2: .`
10. *Starte den Prozeß des Rückwärtseinsetzens, wobei wieder spaltenweise vorgegangen wird. Berechne die i-ten Unbekannten:*
    `n(i,nn+1:nn+c) = n(i,nn+1:nn+c) / n(i,i) .`
11. *Überprüfe ob alle Unbekannten bearbeitet wurden:*
    `IF i=1 GOTO Ende:` 'Ende o.K.' .
12. *Berücksichtige der Anteile der i-ten Unbekannten bei allen Gleichungen von 1 bis i-1:*
    `n(1:i-1,nn+1:nn+c) = n(1:i-1,nn+1:nn+c) - n(1:i-1,i)`
    `n(i,nn+1:nn+c) .`
13. *Verringere den Schleifenzähler und berechne die nächstniedere Unbekannte:*
    `i=i-1`
    `GOTO 10: .`

Ende:

Ressourcen: Anzahl der Operationen: $\frac{1}{6}n^3 + n^2c + \mathcal{O}(n^2, nc)$
Platzbedarf: n^2, nc

Dieser Algorithmus löst das Gleichungssystem in der oberen Dreieckshälfte. Der Anteil unterhalb der Diagonale wird weder verwendet noch verändert und behält somit die ursprünglichen Information. Wenn die Matrix N für weitere Berechnungen benötigt wird,

ist daher vor dem Start von Algorithmus 2.1 die Diagonale zu sichern. Der Algorithmus 2.1 gibt, da keine umständlichen Adreßrechnungen vorhanden sind, die Grundstruktur des Ablaufs bei der Choleskyreduktion sehr schön wieder und dient daher als Grundgerippe für die folgenden Spezialanwendungen. Für die Wahl des Abbruchkriteriums wegen der nicht positiv Definitheit (Schritt 8) wird auf die Anmerkung am Schluß von Abschnitt 2.4.2 verwiesen, wo eine anschauliche Interpretation des Abbruchgliedes gegeben wird.

Die Anzahl der Rechenoperationen während der Choleskyreduktion ist durch ist geprägt von den i Operationen zur Berechnung des Skalarproduktes (Schritt 4). Unter der vereinfachten Annahme, daß alle Operationen (Multiplikation, Division und Berechnung der Quadratwurzel) gleich gezählt werden sind für die Choleskyreduktion pro Element i Grundoperationen (Addition und Multiplikation) auszuführen. Für die gesamte Dreieckszerlegung werden somit

$$\Gamma = \sum_{j=1}^{n} \sum_{i=1}^{j} i = \frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n \tag{2.77}$$

benötigt. Die Anzahl von Divisionen $\frac{1}{2}n(n-1)$ bzw. der Berechnungen von Quadratwurzeln n ist von untergeordneter Bedeutung. Für die (Vorwärts- und Rückwärts-) Einsetzprozesse sind jeweils

$$\Pi = \frac{1}{2}n(n+1) \tag{2.78}$$

Operationen erforderlich. Die Gesamtanzahl an Operation für die Berechnung von c Lösungen eines Gleichungssystems nach der Methode von Cholesky akkumuliert sich somit zu

$$\Gamma + 2c\Pi = \frac{1}{6}n^3 + \frac{1+2c}{2}n^2 + \frac{1+3c}{3}n \tag{2.79}$$

Als Faustregel kann die Anzahl der Rechenoperationen des Lösungsverfahrens ist mit ungefähr $\frac{1}{6}n^3 + cn^2 + \mathcal{O}(n^2, cn)$ abgeschätzt werden.

Bei der Beschäftigung mit dynamischen Systemen ist es interessant auf welche Elemente des reduzierten Systems sich eine Änderung einer Spalte oder Zeile bzw. eines Elementes n_{ij} auswirkt. Diese Frage läßt sich anschaulich in Zusammenhang mit den Abb. 2.3 und 2.4 beantworten. Eine Änderung des Elements n_{ij} bewirkt eine Änderung der Spalte i ab der Zeile j in der reduzierten Form und eine Änderung aller reduzierten Glieder $r_{\ell k}$, für die $\ell \geq j$ und $k \geq j$ gilt (siehe Abb. 2.7). Abhängig vom Ort der Änderung müssen alle Glieder oder nur eine Spalte neu berechnet werden.

2.4.1.2 Rundungsfehler beim Verfahren von Cholesky

Bei der Auswahl des Verfahrens wurden zwei Beweggründe angeführt. Die einfache, rasche und platzsparende Berechnung unter voller Nutzung der Symmetrie und die

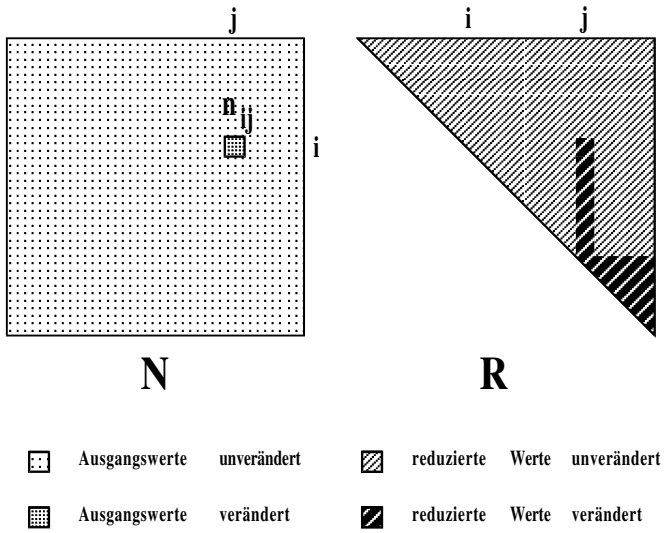


Abbildung 2.7: Darstellung der Auswirkung einer Änderung des Elementes n_{ij}, $i \leq j$ im reduzierten System.

gute numerische Stabilität. Angesichts der großen Anzahl der Rechenoperationen ist es vor allem in Hinblick auf große Systeme erforderlich, das Verhalten von Rundungsfehler zu studieren: wie in der Betrachtung über Stabilität und Sensibilität (Abschnitt 2.2) erläutert, sind oft mehrere Zugänge zur Abschätzung der entstehenden Fehler durch Rundungseinflüsse möglich. *G.W. Stewart (1973)[82]* benützt die Methode der Rückwärtsanalyse. Die Untersuchung gliedert sich in die Analyse der Dreieckszerlegung, des Vorwärts- und Rückwärtseinsetzens. Die für diesen Abschnitt wesentlichen Ergebnisse sind hier kurz zusammengefaßt.

Wie bei der Rückwärtsanalyse üblich, wird die mit Rundungsfehlern behaftete Lösung als strenge Lösung eines gestörten Systems betrachtet. Der Einfluß von Störungen in den Eingangsdaten auf die Fehler in der Lösung kann durch (2.40) abgeschätzt werden. Bei der Dreieckszerlegung bewirken die Rundungsfehler eine Störung ΔN^D

$$\left(\boldsymbol{N} + \Delta \boldsymbol{N}^D\right) \boldsymbol{x} = \boldsymbol{b}, \tag{2.80}$$

deren Koeffizienten Δn_{ij}^D nach *G.W. Stewart (1973)[82]*, Th. 3.5.2 folgendermaßen

$$\left|\Delta n_{ij}^D\right| \leq c\ n\ a\ g\ \varepsilon \tag{2.81}$$

beschränkt sind. Darin bedeuten

c ... Konstante der Größenordnung 1
n ... Dimension des Systems
a ... betragsgrößtes Element in den Eingangsdaten $\max\limits_{i,j} |n_{ij}|$
g ... Wachstumsfaktor $\frac{1}{2} \max\limits_{i,j,k} \left|n_{ij}^{(k)}\right| / \max\limits_{i,j} |n_{ij}|$
$n_{ij}^{(k)}$ ... alle Zahlen (Summen, Produkte) die während des Prozesses der Dreieckszerlegung auftreten (im weiteren auch während des Vorwärts- und Rückwärtseinsetzens).
ε ... Maschinengenauigkeit $\frac{1}{2}\beta^{-\tau+1}$ (siehe (2.26)

Die Koeffizienten des Störfeldes $\Delta \boldsymbol{N}^E$ bei den (Vorwärts- und Rückwärts-) Einsetzprozessen sind durch

$$\left|\Delta n_{ij}^E\right| \leq c\ (n+1)\ \ |r_{ij}|\ \ \varepsilon \tag{2.82}$$

zu beschränken *G. W. Stewart (1973)[82]*, Th. 3.5.1, Seite ???). Bei der Zusammenfassung aller notwendigen Vorgänge akkumulieren sich die Störeinflüße zu ΔN

$$(\boldsymbol{N} + \Delta \boldsymbol{N})\, \boldsymbol{x} = \boldsymbol{b}, \tag{2.83}$$

mit den oberen Schranken

$$|\Delta n_{ij}| \leq \left(c_1 n + 2c_2 n^2 + c_2^2 n^3 \varepsilon\right) a\ g\ \varepsilon \tag{2.84}$$

für die Größe der Koeffizienten Δn_{ij} des Störfeldes (c_1,c_2 ... Konstanten der Größenordnung 1, n, a, g, ε wie (2.81). Die Unsicherheit der Lösung $\Delta \boldsymbol{x}$ kann aus (2.40) mit

$$\frac{|\Delta \boldsymbol{x}|}{|\boldsymbol{x}|} \leq \kappa_2(\boldsymbol{N}) \left(c_1 n + 2c_2 n^2 + c_2^2 n^3 \varepsilon\right) g\ \varepsilon \tag{2.85}$$

abgeleitet werden. Die Konditionszahl $\kappa_2(\boldsymbol{N})$ wirkt somit wieder als Multiplikator bei der Fortpflanzung des Einflusses der Rundungsfehler. Die Hauptkomponente des zweiten Terms sind quadratisch mit der Dimension n des Gleichungssystems gekoppelt. Der kubische Faktor wird durch die Multiplikation mit ε stark abgeschwächt. Diese Ergebnisse gelten für beliebige Dreieckszerlegungen von regulären Systemen. Abschätzungen bezüglich des Wachstumsfaktors zeigen den Vorteils des Cholesky-Verfahrens sehr deutlich. Ohne Pivotierung kann für eine LR-Zerlegung der Wachstumsfaktor g nicht eingegrenzt werden. Bei einer Spaltenpivotsuche (partiellen Pivotisierung) gilt die Aussage $g \leq 2^b$ bei einer Bandbreite b. Für eine LR-Zerlegung mit vollständige Pivotisierung wird ein Faktor $g \leq n$ vermutet. Bei symmetrischen Matrizen kann der Wachstumsfaktor sowohl für die LR-Zerlegung mit Pivotisierung als auch für die Choleskyreduktion ohne Pivotisierung mit $g \leq 1$ eingeschränkt werden. Zusammenfassend gilt daher die Aussage, daß die Auswirkungen der Rundungsfehler quadratisch mit der Dimension des Gleichungssystems ansteigen.

Mit der Fehlerabschätzung bei großen geodätischen Netzen mit ungefähr 400.000 Unbekannten befaßt, überlegte sich *P. Meissl (1980)[54]* einen vollkommen anderen Zugang. Er betrachtet die durch die beschränkte Ziffernzahl auftretenden Fehler als statistische Größen. Abgeleitet von einer angenommenen Gleichverteilung der auftretenden Einzelfehler, werden für zwei Rechnertypen Fehleraussagen zum Gesamtproblem abgeleitet. Er untersuchte sowohl einen Rechner bei dem die Ergebnisse aller Einzeloperationen in gerundeter Form vom Rechenregister in den Speicher übernommen werden (Fehler=$\pm\varepsilon$, CDC 6600, $\beta = 2, \tau = 48$) als auch eine Maschinen deren Ergebnisse mit Vernachlässigung der nachlaufenden Stellen übergeben werden (Fehler=2ε, IBM 360, $\beta = 16$, $\tau = 14$). Seine Ergebnisse für die Dreieckszerlegung mit Vorwärts- und Rückwärtseinsetzen werden im folgenden zusammengefaßt.

Stellt man die Ergebnisse der Einzeloperationen zusammen und bringt sie in eine einheitliche Notation zum zuvor Gesagtem, so erlangt man für die Dreieckszerlegung inklusive der Reduktion der rechten Seite zu folgenden Aussagen über den Erwartungswert $E\{\Delta x\}$

$$E\{\Delta \boldsymbol{x}\} \begin{cases} = 0 & \textit{rundend} \\ \leq d \left|\boldsymbol{N}^{-1}\right| (|x|\,\Gamma + \Pi)\, a\; g\; \varepsilon & \textit{abschneidend} \end{cases} \tag{2.86}$$

und die Varianz $\sigma\{\Delta \boldsymbol{x}\}$ des Lösungsvektors,

$$\sigma\{\Delta \boldsymbol{x}\} \leq d \left|\boldsymbol{N}^{-1}\right| \sqrt{\frac{1}{6}\,|\boldsymbol{x}|^2\,\Gamma + \frac{1}{12}\Pi}\; a\; g\; \varepsilon. \tag{2.87}$$

Die Größen Γ entspricht der Anzahl der notwendigen Operationen zur Berechnung der Choleskyreduktion (siehe (2.77)) und die Größe Π legt die Anzahl der Operationen beim Vorwärts- und Rückwärtseinsetzen fest (siehe (2.78)). Die Größen a, g, ε sind gleich wie für (2.81) definiert. Im Gegensatz zu *P. Meissl* wird wieder eine Multiplikation mit nachfolgender Addition als eine Operation aufgefaßt und der Übergang auf die relative Maschinengenauigkeit ε (2.26) durchgeführt. Die Konstante d weist somit die Größe $\frac{2}{5}$ auf. Analoge Umformungen führen zum Fehler beim Rückwärtseinsetzen, die durch

$$E\{\Delta \boldsymbol{x}\} \begin{cases} = 0 & \textit{rundend} \\ \leq d \left|\boldsymbol{R}^{-1}\right| \Pi a\; g\; \varepsilon & \textit{abschneidend} \end{cases} \tag{2.88}$$

$$\sigma\{\Delta \boldsymbol{x}\} \leq d \left|\boldsymbol{R}^{-1}\right| \sqrt{\frac{1}{12}\Pi}\; a\; g\; \varepsilon. \tag{2.89}$$

gegeben sind. Durch das Fortpflanzungsgesetz für die Erwartungswerte und die Varianzen, folgt bei angenommener Unkorreliertheit

$$E\{\Delta \boldsymbol{x}\} \begin{cases} = 0 & \textit{rundend} \\ \leq d\kappa_2(\boldsymbol{N})\, (|\boldsymbol{x}|\,\Gamma + 2\Pi)\, g\; \varepsilon & \textit{abschneidend} \end{cases} \tag{2.90}$$

$$\sigma\{\Delta \boldsymbol{x}\} \leq d\,\kappa_2(\boldsymbol{N})\sqrt{\frac{1}{6}\,|\boldsymbol{x}|^2\,\Gamma + \frac{1}{6}\Pi}\;g\;\varepsilon \tag{2.91}$$

eine Schätzung für die Erwartungswerte $E\{\Delta x\}$ des Rundungsfehler bei *rundender* und *abschneidender* Berechnung als auch deren Varianz $\sigma\{\Delta x\}$. Da nach (2.77) die Größe Γ - im Gegensatz zur quadratischen Ordnung von Π (siehe (2.78)) - mit der dritten Ordnung der Anzahl der Unbekannten n wächst, hat dieser Term bei steigender Anzahl von Unbekannten die größte Bedeutung. Interpretiert man den Erwartunswert $E\{\Delta x\}$ als eine Verzerrung des Ergebnisses, so bestehen bei Berechnungen, wo gerundet wird, keine Bedenken. Werden die Zwischenergebnisse durch Vernachlässigung der nachlaufenden Ziffern weiterverarbeitet, so ist große Vorsicht geboten, da die Verzerrung proportional mit der dritten Ordnung wächst. Bemerkenswert ist auch die Tatsache, daß die Varianz der Rundungsfehler (2.91) nur mit der Ordnung $n^{\frac{2}{3}}$ ansteigt. Vergleiche dazu die quadratische Ordnung in der Abschätzung (2.85).

Generell können zur Verringerung des Einflußes von Rundungsfehler mehrere Hilfmittel eingesetzt werden. Die Senkung der Norm von $|x|$ durch Einführung von Näherungskoordinaten oder durch eine Nach iteration stellt eine Möglichkeit zur Steigerung der absoluten Genauigkeit dar. Dazu betrachtet man die berechnete, von Rundungsfehlern verfälschte Lösung als Näherungslösung $\hat{x}$. Der Residuenvektors r

$$r = A\hat{x} - b \tag{2.92}$$

weist auf die Fehler dieser Lösung hin. Die absolute Genauigkeit der Lösung kann nun gesteigert werden, indem die verfälschte Lösung als Näherungslösung aufgefaßt wird und der Zuschlag δx zu dieser Näherungslösung aus

$$A\delta x = -r \tag{2.93}$$

errechnet wird. Die endgültige Lösung x errechnet sich aus

$$x = \hat{x} + \delta x \tag{2.94}$$

Für die zweite Berechnung kann auf die schon erfolgte Cholesky-Reduktion zurückgegriffen werden. Bei diesem Vorgang der *Nachiteration* ist nur mehr mit kleinen Zuschlägen zu rechnen. Der Multiplikator $|\delta x|$ der an die Stelle von $|x|$ rückt, stellt somit in (2.85) und (2.91) einen kleinen Wert dar.

Eine weitere Möglichkeit den Einfluß von Rundungsfehlern zu verringern, ist durch eine erhöhte Genauigkeit bei der Berechnung gegeben. Bei vielen technischen Programmiersprachen sind neben einfacher und doppelter Genauigkeit auch Berechnungen mit vierfacher Genauigkeit Berechnungen möglich. Dies erhöht den Zeitaufwand und vor allem den Aufwand an Speicherplatz. Konzentriert man sich auf den Rechenkern des Verfahrens von Cholesky, die Bildung des Skalarproduktes (Schritt 4), so stellt die doppelt genaue Akkumulation des Skalarproduktes eine einfaches aber wirksames Mittel

zur Senkung der Rundungsfehler dar. Bei Nutzung dieser Methode kann die Fehlerabschätzung (2.84) umgeformt werden zu

$$|\Delta n_{ij}| \leq \left(c_1 + 2c_2 n + c_2^2 n\varepsilon\right) a\ g\ \varepsilon \tag{2.95}$$

(siehe *G.W. Stewart (1973)[82]*). Für den stochastischen Ansatz ist zu beachten, daß für die Anzahl der Operationen mit Rundungsfehlern nicht mehr Γ und Π einzusetzten ist. Durch das doppelt genaue Skalarprodukt wird die Ordnung der Operatoren mit Rundungsfehlern auf eine quadratische Ordnung $\mathcal{O}(n_2)$ gesenkt. Eine weitere extreme Senkung der Anzahl der Operationen wird durch die Beachtung von schwacher Besetztheit bei vielen Problemstellungen erreicht. Diesen Methoden ist Kapitel 4 gewidmet.

2.4.1.3 Cholesky-Reduktion mit effizienter Speicherung

Der in Abschnitt 2.4.1.1 dargestellte Algorithmus 2.1 zur Lösung eines symmetrischen Gleichungssystems, nutzt die Symmetrie nur bei der Berechnung nicht jedoch bei der Speicherung. Eine effizientere Speicherung ist dann möglich, wenn die Matrix N anderweitig zwischengespeichert wird oder nicht mehr benötigt wird. Speichert man die obere Dreiecksmatrix spaltenweise in einem Vektor, so ist eine gute Ausnutzung des Arbeitsspeichers, eine einfachere dynamische Dimensionierung und eine günstigere, eng gepackte Speicherbelegung gegeben (Abb. 2.8).

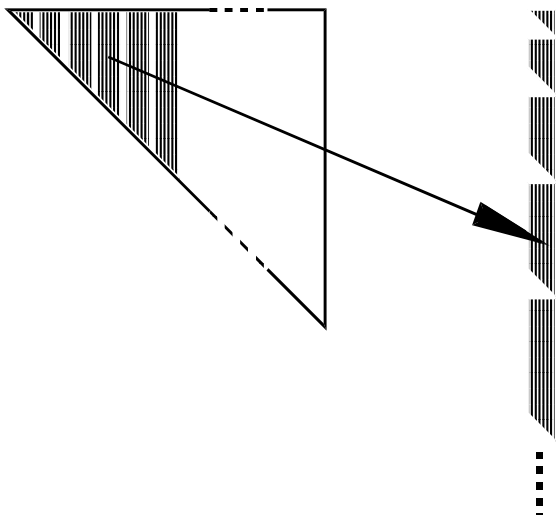


Abbildung 2.8: Darstellung der gepackten Speicherbelegung der oberen Dreiecksmatrix in einem Vektor.

Die Adressierung der Elemente erfolgt durch

$$n_{ij} = \text{Vek(k)} \qquad \text{mit} k = j(j-1)/2 + i, \quad i \leq j\ . \tag{2.96}$$

Algorithmus 2.2 stellt eine modifizierte Form des Algorithmus 2.2 dar, der diese Art der Speicherung berücksichtigt und der Aufwand an Adreßrechnung minimiert. Da

der Konstantenvektor als letzte Spalte der Matrix gespeichert wird, beträgt der zur Speicherung notwendige Platzbedarf für das Gleichungssystem nur mehr $\frac{1}{2}(n+1)(n+2)$ Speicherplätze.

Algorithmus 2.2 : Auflösung eines als Vektor gespeicherten, symmetrischen, positiv definiten Gleichungssystems durch das Verfahren von Cholesky.

Aufgabenstellung: Gegeben ist ein symmetrisches, positiv definites Gleichungssystem $\mathbf{N} \in \mathbb{R}^{n \times n}$, der Konstantenvektor $\mathbf{b}$. Zur Speicherung der Matrix wird der Vektor **VEK** verwendet, wo die oberer Dreiecksmatrix spaltenweise gespeichert wird. Der Konstantenvektor wird wie die $(n+1)$-te Spalte behandelt. Gesucht ist die Lösung des Systems, wenn das Gleichungssystem positiv definit ist. Andernfalls ist das Verfahren mit einer entsprechenden Meldung zu unterbrechen.

Schnittstellen:

Eingabe:
- `VEK(1:n(n+1)/2)` ... Koeffizienten der symmetrischen Matrix **N**, nur Elemente $i \leq j$ (obere Dreiecksmatrix) spaltenweise gespeichert,
- `VEK(n(n+1)/2+1:n(n+1)/2+n)` ... Konstantenvektor
- `n` ... Anzahl der Zeilen und Spalten des Systems

Ausgabe:
- `VEK(1:n(n+1)/2)` ... modifizierte Koeffizienten der Matrix **N**
- `VEK(n(n+1)/2+1:n(n+1)/2+n)` ... Lösungen des Gleichungssystems

Felder: `n(1:n(n+1)/2+n)`

Hilfsproceduren: —

Ausgangssituation:

n_{11} n_{12} n_{22} n_{13} n_{23} n_{33} ... n_{nn} b_1 ... b_n

Endsituation:

r_{11} r_{12} r_{22} r_{13} r_{23} n_{33} ... r_{nn} x_1 ... x_n

Wolf-Dieter Schuh
ITGB, Uni Bonn

Lösungsweg: Pseudocode

1. *Initialisiere den Spaltenindex, die Spaltenadresse, die Adresse des als zu bearbeitenden Elements und die Adresse des Konstantenvektors (Unbekanntenvektors) für die Cholesky-Reduktion bzw. das Vorwärtseinsetzen (spaltenweise Reduktion):*
   `j=0; adr_j=0; adr_akt=0; adr_unb=n(n+1)/2` .
2. *Bearbeite nächsten Spalte; rechne neue Adresse des Spaltenbeginns und erhöhe den Spaltenindex:*
   `adr_j=adr_j+j; j=j+1` .
   Initialisiere den Zeilenindex und die Zeilenadresse:
   `i=0, adr_i=0` .
3. *Bearbeite nächstes Element; rechne neue Adresse, erhöhe den Zeilenindex und die Adresse des zu bearbeitenden Elements:* `adr_i=adr_i+i; i=i+1; adr_akt=adr_akt+1` .
4. *Beginne die Reduktion des Gliedes in der i-ten Zeile und j-ten Spalte:*
   `VEK(adr_akt) = VEK(adr_akt) - VEK(adr_i+1:adr_i+i-1) VEK(adr_j+1:adr_j+i-1)` .
5. *Überprüfe ob das zu reduzierende Glied ein Diagonalglied ist oder nicht und setze entsprechend fort:*
   `IF i=j GOTO` `8:` .
6. *Reduziertes Glied stellt kein Diagonalglied dar. Vervollständige die Berechnung durch Berücksichtigung der Division:*
   `VEK(adr_akt) = VEK(adr_akt) / VEK(adr_i+i)` .
7. *Beende entsprechend dem Zeilenindex den Vorgang der Reduktion bzw. des Vorwärtseinsetzens oder setze die Reduktion beim nächsten Glied dieser Spalte fort:*
   `IF i=n GOTO` `10:`
   `GOTO` `3:` .
8. *Reduziertes Glied ist ein Diagonalglied. Überprüfe das Diagonalglied:*
   `IF VEK(adr_akt) < num_Null GOTO` `Ende:` 'Keine positiv definite Matrix' .
9. *Vervollständige die Berechnung des Diagonalgliedes durch Berücksichtigung der Wurzel:*
   `VEK(adr_akt) =` $\sqrt{\texttt{VEK(adr\_akt)}}$
   `GOTO` `2:` .
10. *Starte den Prozeß des Rückwärtseinsetzens, wobei wieder spaltenweise vorgegangen wird. Berechne die i-te Unbekannte:*
    `VEK(adr_unb+i) = VEK(adr_unb+i) / VEK(adr_i+i)` .
11. *Überprüfe ob alle Unbekannten bearbeitet wurden:*
    `IF i=1 GOTO` `Ende:` 'Ende o.K.' .

Wolf-Dieter Schuh
ITGB, Uni Bonn

12. *Berücksichtige den Anteil der i-ten Unbekannten bei allen Gleichungen von 1 bis i-1:*
`VEK(adr_unb+1:adr_unb+i-1) = VEK(adr_unb+1:adr_unb+i-1) - -`
`VEK(adr_i+1:adr_i+i-1) / VEK(adr_unb+i) .`

13. *Verringere den Schleifenzähler, rechne die neue Adresse und berechne die nächstniedere Unbekannte:*
`i=i-1; adr_i=adr_i-i`
`GOTO 10:` .

Ende:

Ressourcen: Anzahl der Operationen: $\frac{1}{6}n^3 + \frac{1}{2}n^2c + \wr(n^2, nc)$
Platzbedarf: $\frac{1}{2}n(n+3)$

2.4.1.4 Verallgemeinerte Cholesky-Reduktion mit Blockmatrizen

Eine symmetrische, positiv definite Matrix N wird in vier Blöcke

$$\boldsymbol{N} = \begin{bmatrix} \boldsymbol{N}_{11} & \boldsymbol{N}_{12} \\ \boldsymbol{N}_{12}^T & \boldsymbol{N}_{22} \end{bmatrix} \tag{2.97}$$

geteilt, wobei $\boldsymbol{N}_{11}$ und $\boldsymbol{N}_{22}$ quadratische Matrizen darstellen. In analoger Weise wird eine obere Dreiecksmatrix R ein

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ & \boldsymbol{R}_{22} \end{bmatrix} \tag{2.98}$$

unterteilt, wobei $\boldsymbol{R}_{11}$ und $\boldsymbol{R}_{22}$ ebenfalls obere Dreiecksmatrizen bilden, $\boldsymbol{R}_{12}$ hingegen vollbesetzt ist. Aus der Identität

$$\boldsymbol{N} = \boldsymbol{R}^T\boldsymbol{R} \tag{2.99}$$

können durch Ausmultiplikation und anschließenden Koeffizientenvergleich die Beziehungen

$$\boldsymbol{N}_{11} = \boldsymbol{R}_{11}^T\boldsymbol{R}_{11} \tag{2.100}$$

$$\boldsymbol{N}_{12} = \boldsymbol{R}_{11}^T\boldsymbol{R}_{12} \tag{2.101}$$

$$\boldsymbol{N}_{22} = \boldsymbol{R}_{12}^T\boldsymbol{R}_{12} + \boldsymbol{R}_{22}^T\boldsymbol{R}_{22} \tag{2.102}$$

hergeleitet werden. Isoliert man die Größen der oberen Dreiecksmatrix und deutet durch $()^C$ die nach Cholesky reduzierte Form an, so ergeben sich aus (2.100) bis (2.102) die Relationen

$$\boldsymbol{R}_{11} = \boldsymbol{N}_{11}^C \tag{2.103}$$

$$\boldsymbol{R}_{12} = \left(\boldsymbol{R}_{11}^T\right)^{-1} \boldsymbol{N}_{12} \tag{2.104}$$

$$\boldsymbol{R}_{22} = \left(\boldsymbol{N}_{22} - \boldsymbol{R}_{12}^T \boldsymbol{R}_{12}\right)^C \tag{2.105}$$

Der Ausdruck (2.105) kann durch Anwendung von (2.104) unter Beachtung von

$$\boldsymbol{R}_{11}^{-1}\left(\boldsymbol{R}_{11}^{-1}\right)^T = \left(\boldsymbol{R}_{11}^T \boldsymbol{R}_{11}\right)^{-1} = \boldsymbol{N}_{11}^{-1} \tag{2.106}$$

umgeformt werden zu

$$\boldsymbol{R}_{22} = \left(\boldsymbol{N}_{22} - \boldsymbol{N}_{12}^T \boldsymbol{N}_{11}^{-1} \boldsymbol{N}_{12}\right)^C \quad . \tag{2.107}$$

2.4.1.5 Generalisierte Form der Cholesky-Reduktion

To analyze this algorithm, we reorder the steps and concentrate the computational work on the inner loop. First we define $N_{ij}^{(k)}$ as an update of $n_{ij}^{(k-1)}$,

$$n_{ij}^{(k)} := n_{ij}^{(k-1)} - r_{ki} r_{kj} \tag{2.108}$$

with $n_{ij}^{(0)} = n_{ij}$ as the initial value. The fully updated variable $n_{ij}^{(i-1)}$ is denoted by $\bar{n}_{ij}$. Now we are able to summarize three different operations with the help of one defining equation,

$$r_{ki} r_{kj} = n_{ijk} \qquad \text{with} \qquad n_{ijk} = \begin{cases} \bar{n}_{ij} & \text{for} \quad i = k \\ n_{ij}^{(k-1)} - n_{ij}^{(k)} & \text{for} \quad i \neq k \ . \end{cases} \tag{2.109}$$

A corresponding matrix form is defined by

$$\boldsymbol{R}_{ki}^T \boldsymbol{R}_{kj} = \boldsymbol{N}_{ijk} \qquad \text{with} \qquad \boldsymbol{N}_{ijk} = \begin{cases} \bar{\boldsymbol{A}}_{ij} & \text{for} \quad i = k \\ \boldsymbol{N}_{ij}^{(k-1)} - \boldsymbol{N}_{ij}^{(k)} & \text{for} \quad i \neq k \ . \end{cases} \tag{2.110}$$

If $k = i = j$, equation 2.109 becomes $r_{ii}^2 = n_{ii}^{(i-1)} = \bar{n}_{ii}$. Because of the fully updated variable $\bar{n}_{ii}$ on the right, the left unknown r_{ii} can be computed by

$$r_{ii} = \sqrt{\bar{n}_{ii}}. \tag{2.111}$$

Having in mind to exploit these steps also for matrix operations, let us call this step a *Cholesky-step* because the fully updated sub-matrix $\bar{n}_{ii}$ can be split into two triangular matrices R_{ii}^T and R_{ii} by a Cholesky-factorization,

$$\boldsymbol{R}_{ii}^T \boldsymbol{R}_{ii} = \bar{\boldsymbol{N}}_{ii}. \tag{2.112}$$

For $k = i$ but $i \neq j$ the defining equation 2.109 becomes $r_{ii}r_{ij} = \bar{n}_{ij}$ and the variable r_{ij} is defined by

$$r_{ij} = \bar{n}_{ij}/r_{ii}. \tag{2.113}$$

Looking at the matrix case, the solution R_{ij} of the linear equation system

$$\boldsymbol{R}_{ii}^T \boldsymbol{R}_{ij} = \bar{\boldsymbol{N}}_{ij} \tag{2.114}$$

can be performed very easily by a *substitution-step*, because R_{ii} defines an upper triangular matrix. The third operation, if $k \neq i$ and $k \neq j$, was already defined in equation (2.108) with the definition of $n_{ij}^{(k)}$. Let us define the matrix formulation with

$$\boldsymbol{N}_{ij}^{(k)} := \boldsymbol{N}_{ij}^{(k-1)} - \boldsymbol{R}_{ki}^T \boldsymbol{R}_{kj} \tag{2.115}$$

and call this step *update-step*.

$$\begin{array}{l}
DO \;\; i = 1, ..., n \\
\quad DO \;\; j = i, ..., n \\
\quad\quad DO \;\; k = 1, ..., i \\
\quad\quad\quad r_{ki}r_{kj} = n_{ijk} \quad \Rightarrow \\
\quad\quad \Rightarrow \left\{ \begin{array}{ll}
k = i = j & r_{ii} = \sqrt{n_{ii}^{(k-1)}} \\
 & CHOLESKY \\
k = i, i \neq j & r_{ij} = n_{ij}^{(k-1)}/r_{ii} \\
 & SUBSTITUTION \\
k \neq i & n_{ij}^{(k)} = n_{ij}^{(k-1)} - r_{ki}r_{kj} \\
 & UPDATE
\end{array} \right. \\
\quad\quad END \;\; DO \;\; k \\
\quad END \;\; DO \;\; j \\
END \;\; DO \;\; i
\end{array}$$

Abbildung 2.9: Cholesky-Factorization; scalar form. Three nested loops with a concentrated computational part(body).

We collect the three tasks of the scalar defining equation (2.109) and the matrix defining equation (2.110), respectively,

- *Cholesky-step* (2.111), or (2.112),
- *substitution-step* (2.113), or (2.114) and

- *update-step* (2.108), or (2.115)

and call these steps the '*body*' of the process. Algorithm 2.9 illustrates the whole routine for scalars.

The implementation of the algorithm in this form will be very simple. But this is not the main aspect of this reformulation. We want to demonstrate only the fact that the sequence of the three loops can be changed arbitrarily. The expert has free scope to optimize the memory access for a special hardware. Subsection 2.4.1.6 shows some reflections on this topic. For the further treatment we choose, by the way of example, the sequence ijk for the loop nesting.

We try now to remove from Algorithm 2.9 as many steps as possible from the inner loop (body) and form independent vector and matrix operations. For a simple representation of the formulas we use an index notation which is compatible with FORTRAN 90 standard[1].

The Cholesky-step is only necessary, if $k = i = j$ and is therefore situated within the first loop. The substitution-step can be done for the whole remaining row

$$\begin{aligned} &DO \;\; j = i+1, ..., n \\ &\quad r_{ij} = \bar{n}_{ij}/r_{ii} \\ &END \; DO \;\; j \end{aligned} \tag{2.116}$$

and can be performed by one vector-operation

$$r_{i,i+1:*} = \bar{n}_{i,i+1:*}/r_{ii} \; . \tag{2.117}$$

These two steps (Cholesky-step, substitution-step) imply the condition that $\bar{n}_{i,i:*}$ is computed beforehand, that means the fully updated i^{th} row has to exist. Therefore, the next row of the matrix has to be updated completely by

$$\begin{aligned} &DO \;\; j = i+1, ..., n \\ &\quad DO \;\; k = 1, ..., i-1 \\ &\quad\quad n_{ij}^{(k)} = n_{ij}^{(k-1)} - r_{ki} r_{kj} \\ &\quad END \; DO \;\; k \\ &END \; DO \;\; j \; . \end{aligned} \tag{2.118}$$

The kernel of this update-step consists of a multiplication of the sub-vector $r_{1:i-1,i}$ with the sub-matrix $r_{1:i-1,i+1:*}$. The compact formulation

$$\bar{n}_{i,i+1:*} = n_{i,i+1:*} - r_{1:i-1,i} r_{1:i-1,i+1:*} \tag{2.119}$$

replaces the j and k loops and performs a complete update of the $(i+1)^{th}$ row.

[1] $n_{i,j}$ means the element $A(i,j)$ of a matrix. $n_{l:u,j}$ denotes a part from j^{th}-column of $\boldsymbol{A}$, starting with the element in row l and ending with the element in row u. If a star '$*$' or blank ' ' marks the lower (upper) border, instead of l (u), the first (last) element is defined. No element is matched, if a lower border is greater than the corresponding upper border. This notation can be used to describe arbitrary sub-matrices. The above example $n_{l:u,j}$ denotes a column-vector with $u - l + 1$ elements. $n_{i,g:h}$ or $n_{i,1:*}$ characterize row-vectors. The notation $n_{l:u,g:h}$ marks a sub-matrix. $n_{*:*,*:*}$ or $n_{:,:}$ defines the whole two-dimensional matrix $\boldsymbol{A}$.

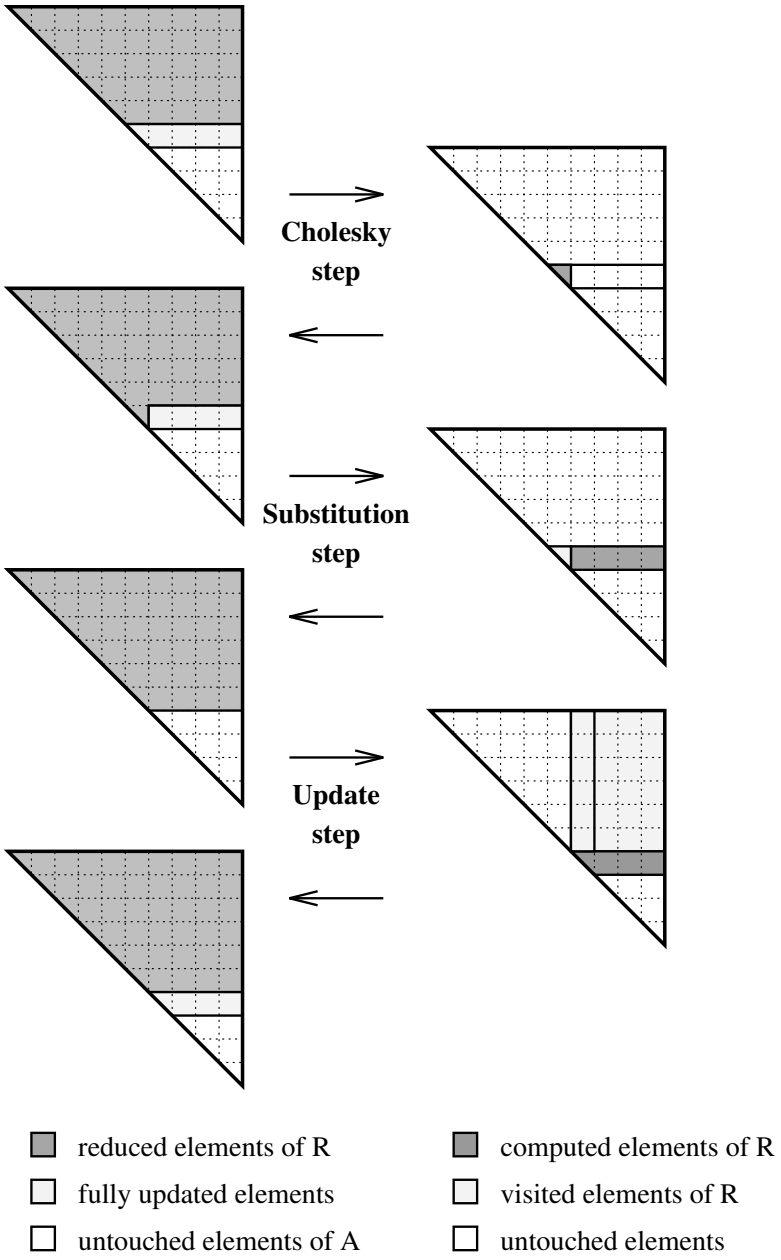


Abbildung 2.10: Cholesky-Factorization; memory and access situation during an i loop. On the left side the memory situation is illustrated. The right side shows the necessary accesses during the three basic operations.

Algorithm 2.11 shows a compact form of algorithm 2.9, the **vector form**. The memory situation and also all necessary accesses during the operations are illustrated in Fig. 2.10. This figure also implies the well-known fact that one triangular system can hold the original elements of $\boldsymbol{N}$ as well as the reduced elements of $\boldsymbol{R}$. The factorization can be performed *on place*.

$$
\begin{array}{ll}
DO \;\; i = 1, ...n & \\
\quad r_{ii} = SQRT(n_{ii}) & CHOLESKY \\
\quad r_{i,i+1:*} = n_{i,i+1:*}/r_{ii} & SUBSTITUTION \\
\quad \bar{n}_{i+1,i+1:*} = n_{i+1,i+1:*} - r_{1:i,i+1} r_{1:i,i+1:*} & \\
END \; DO \;\; i & UPDATE
\end{array}
$$

Abbildung 2.11: Cholesky-Factorization; vector form.

$$
\begin{array}{ll}
N = \lceil n/b \rceil & \\
DO \;\; i = 1, ...N & \\
\quad R_{ii}^T R_{ii} = N_{ii} & \\
\quad\quad \Rightarrow \; R_{ii} & CHOLESKY \\
\quad R_{ii}^T R_{i,i+1:*} = N_{i,i+1:*} & \\
\quad\quad \Rightarrow \; R_{i,i+1:*} & SUBSTITUTION \\
\quad R_{1:i,i+1}^T R_{1:i,i+1:*} = N_{i+1,i+1:*} - \bar{N}_{i+1,i+1:*} & \\
\quad\quad \Rightarrow \; \bar{N}_{i+1,i+1:*} & UPDATE \\
END \; DO \;\; i &
\end{array}
$$

Abbildung 2.12: Cholesky-Factorization; matrix (blocked) form.

If we now describe this algorithm not only for scalar elements but also for a block structured matrix, we end up with the algorithm 2.12, the **matrix form**. This algorithm needs the vector form (cf. Alg. 2.11) to solve the Cholesky-step. But all the other steps use `matrix×matrix` operations. If we denote b as the number of rows per block, then N means the round-up of n/b to the next integer and fixes the number of blocks. All blocks, with the exception of the remaining blocks on the right and low border, have the same size $b * b$. The factor b is called *block-size* and can be chosen arbitrarily. The optimal choice for this factor depends mainly on the software and hardware platform of the target system.

2.4.1.6 ijk-Formen der Cholesky-Reduktion

The compiler optimizes the sequence of algorithmic steps. The processor, therefore, works very efficiently. High clock rates of the processors and pre-fetch and pre-decode within the instruction cache allow a lot of operations. As a consequence, the bottleneck of memory access appears. With a hierarchical structure of memory (registers, cache memory, local memory, shared memory) this critical point seems to be removed. Since

the high-speed memory is often 5 to 10 times faster than the next level, the extensive use of the higher level memory components can substantially reduce the memory access time. But the success of hierarchy is then attributed to locality of reference. In order to improve the performance of algorithms, we must use the high-speed memory whenever possible. Therefore, we must consider not only the total number of memory references, but also the pattern of memory accesses.

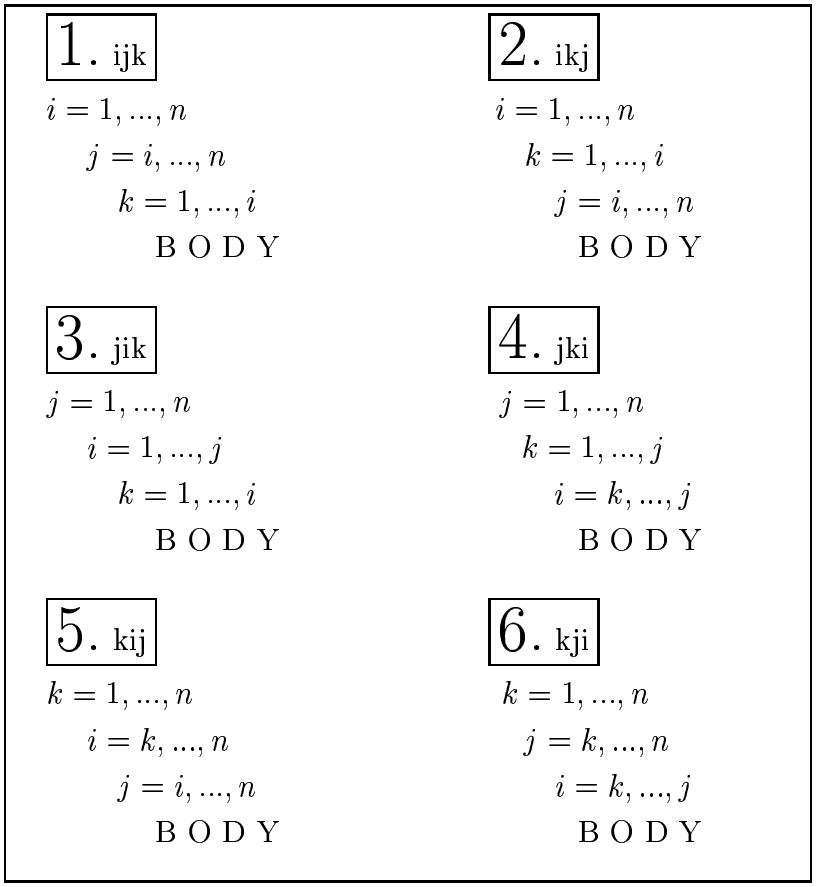


Abbildung 2.13: Cholesky-Factorization; scalar form. Six possibilities of loop nesting with one unique body including the tasks Cholesky, Substitution and Update.

The design of software can consider this fact with two strategies. First, an optimal loop nesting is necessary. The Cholesky-Factorization in scalar form (Alg. 2.9) demonstrates the possibility to nest the three loops in six different ways (cf. Alg. 2.13).

Depending on the loop variable different memory accesses are performed. We distinguish between column or row oriented versions. On the other hand different operations (scalar-product or vector-sum) build the kernel of the computation and affect both speed and achievable accuracy. Another distinction between these algorithms is the kind of updating. Some of these algorithms update the whole remaining part immediately after the computation of an element (right-looking form) or compute all the updates just before the reduction of a special element (left-looking form). A detailed

description of all these forms can be found e.g. in *GEORGE (1987)[33]*, *GALLIVAN et al. (1990)[30]*, or *MAREL (1989)[49]*.

It now depends on the storage scheme of the matrix, which type of loop nesting optimizes the memory access. If the matrix is stored column by column the jik form represents an optimal solution for a scalar processor. This version guarantees a numerical stable behavior, because the kernel consists of a scalar product, which can be done with high precision. The memory access takes place column by column and therefore, higher level fast memory partitions are fully utilized. If the algorithm works on long, successively stored vectors, an optimal speed is achievable.

The speed of this algorithm mainly depends on the implementation of the kernel step. This code can be optimized with an assembler routine for a special hardware architecture. But this is inconsistent with the demand for compatibility over different hardware platforms. The use of standardized routines, which are available on different computers, helps us to overcome this problem. For numerical problems within linear algebra the **B**asic **L**inear **A**lgebra **S**ubprograms (BLAS) are defined by *LAWSON et al. (1979)[47]* and *DONGARRA et al. (1988)[24]* and *(1990)[25]*. The definition covers three types of operations. Level 1 BLAS includes `vector`×`vector` operations, Level 2 BLAS deals with `vector`×`matrix` operations and Level 3 BLAS defines `matrix`×`matrix` operations. In addition to the public domain source code, which is available via *netlib* (E-mail to `netlib@ornl.gov` or *ftp* with username `anonymous` to `netlib.att.com`), there exist a lot of special implementations for different hardware platforms. Within these subprograms a full reuse of data in the cache or local memory is provided. This approach avoids excessive exchange of data to and from memory and allows an efficient processor use.

If an algorithm can be formulated such that the kernel part is done with these optimized BLAS routines, a very high performance is achievable. The vector form of the Cholesky-Factorization (Alg. 2.11) fulfils this condition. This algorithm can be implemented with Level 1 and Level 2 BLAS routines. The benefit of the BLAS routines increases if large parts of vectors or matrices are treated in one step. The matrix form (Alg. 2.12) with the internal loop nesting ijk and a column by column storage scheme allows the largest blocks (cf. *DONGARRA et al. (1990)[25]*,*GALLIVAN et al. (1990)[30]*).

2.4.1.7 Unvollständige Cholesky-Reduktion

Bei der Berechnung von großen Systemen werden Zerlegungsverfahren verwendet, wo ein Teil der Unbekannten aus dem Gesamtsystem durch Reduktion eliminiert wird (*Helmertsche Blockzerlegung*). Neben dem üblichen Weg über Blockinversionsverfahren oder verallgemeinerte Austauschschritte kann auch eine unvollständige Cholesky-Reduktion durchgeführt werden. Dazu wird nicht die gesamte Matrix in eine Dreiecksmatrix zerlegt, sondern man begnügt sich mit der Reduktion auf das System

$$\bar{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ & \bar{\boldsymbol{N}}_{22} \end{bmatrix} \tag{2.120}$$

wobei $\bar{\boldsymbol{N}}_{22}$ eine quadratische, vollbesetzte Matrix darstellt. Das ursprüngliche System $\boldsymbol{N}$ (2.97) wird somit in zwei ungleiche Teile $\bar{\boldsymbol{L}}$ und $\bar{\boldsymbol{R}}$ zerlegt. Die Reduktion wird also nur für die Zeilen und Spalten des ersten Blocks durchgeführt, was sich in der Matrix

$$\bar{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{R}_{11}^T & \\ \boldsymbol{R}_{12}^T & \boldsymbol{I} \end{bmatrix} \tag{2.121}$$

widerspiegelt. Aus der Identität

$$\boldsymbol{N} = \bar{\boldsymbol{L}}\bar{\boldsymbol{R}} \tag{2.122}$$

beziehungsweise

$$\begin{bmatrix} \boldsymbol{N}_{11} & \boldsymbol{N}_{12} \\ \boldsymbol{N}_{12}^T & \boldsymbol{N}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{11}^T & \\ \boldsymbol{R}_{12}^T & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ & \bar{\boldsymbol{N}}_{22} \end{bmatrix} \tag{2.123}$$

errechnet sich

$$\boldsymbol{N}_{11} = \boldsymbol{R}_{11}^T \boldsymbol{R}_{11} \tag{2.124}$$

$$\boldsymbol{N}_{12} = \boldsymbol{R}_{11}^T \boldsymbol{R}_{12} \tag{2.125}$$

$$\boldsymbol{N}_{22} = \boldsymbol{R}_{12}^T \boldsymbol{R}_{12} + \bar{\boldsymbol{N}}_{12} \tag{2.126}$$

woraus

$$\boldsymbol{R}_{11} = \boldsymbol{N}_{11}^C \tag{2.127}$$

$$\boldsymbol{R}_{12} = \left(\boldsymbol{R}_{11}^T\right)^{-1} \boldsymbol{N}_{12} \tag{2.128}$$

$$\bar{\boldsymbol{N}}_{22} = \boldsymbol{N}_{22} - \boldsymbol{R}_{12}^T \boldsymbol{R}_{12} \ . \tag{2.129}$$

abgeleitet wird. Die Rechenvorschrift (siehe auch Abb. 2.14) zur Berechnung der Elemente $\bar{n}_{ij}$, $\ell \le i \le n$ bzw. $\ell \le j \le n$ der Matrix $\bar{\boldsymbol{N}}_{22}$ lautet

$$\bar{n}_{ij} = n_{ij} - \sum_{k=1}^{\ell} r_{ki} r_{kj} \qquad i,j = \ell+1, \ldots, n \tag{2.130}$$

womit wieder eine symmetrische Matrix entsteht, wenn $\boldsymbol{N}_{22}$ als symmetrisch vorausgesetzt wird.

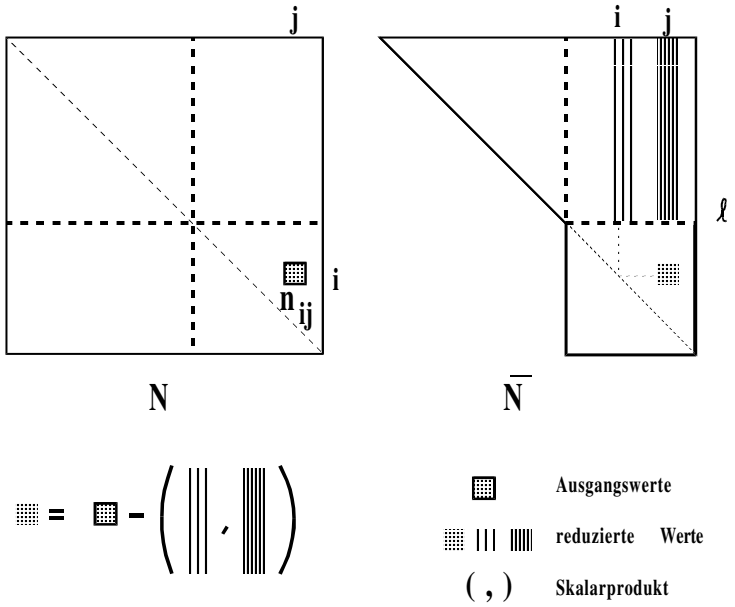


Abbildung 2.14: Grafische Darstellung der Berechnung des Blocks in der zweiten Zeile und Spalte bei der unvollständigen Cholesky-Reduktion (2.130).

Setzt man $\boldsymbol{R}_{12}$ (2.130) in $\bar{\boldsymbol{N}}_{22}$ (2.129) ein so erlangt man

$$\bar{\boldsymbol{N}}_{22} = \boldsymbol{N}_{22} - \boldsymbol{N}_{12}^T \boldsymbol{N}_{11}^{-1} \boldsymbol{N}_{12} \; . \tag{2.131}$$

Der nichtreduzierte Block dieser unvollständigen Cholesky-Reduktion ist ident mit dem Ausdruck für den Block der zweiten Zeile und Spalte des Austauschschrittes (1.14). Das Ergebnis zeigt somit die Veränderung der Beziehungen zwischen den Unbekannten der zweiten Gruppe an, die durch die Elimination der ersten Unbekannten verursacht wird.

Die Anzahl der notwendigen Rechenschritte setzt sich zusammen aus der Berechnung des Dreieckteils der Dimensions ℓ, der Reduktion der $(n-\ell)$ Spalten und der Auswertung von (2.130)

$$\Gamma(\ell) + (n-\ell)\Pi(\ell) + \frac{1}{2}(n-\ell)(n-\ell+1)\ell \; . \tag{2.132}$$

Zusammengefaßt ergeben sich

$$\Gamma(\ell) + \frac{1}{2}(n-\ell)\ell(2+n) \tag{2.133}$$

Rechenoperationen.

2.4.2 Berechnung der Inversen bei Vorliegen einer Cholesky-reduzierten Form

Im Abschnitt 1.2 werden allgemeine Verfahren zur Berechnung der Inversen bzw. der verallgemeinerten Inversen und Pseudoinversen besprochen. Der Algorithmus 1.2 zur Inversion einer regulären, quadratischen Matrix am Platz kann ohne große Probleme so adaptiert werden, das durch die Ausnutzung der Symmetrie nahezu eine Halbierung der Rechenzeit ermöglicht wird. Dieses Verfahren kam vielfach bei beschränktem Arbeitsspeicher zum Einsatz. Anstelle einer elementweisen Berechnung, wird das Verfahren auf Matrizenblöcke verallgemeinert, und mit Hilfe von raschen, hardwarenahen Matrizenoperationen eine blockweise Inversion berechnet. Der dazu notwendige Formelapparat kann aus der Formel (1.15) durch Spezialisierung auf symmetrische Matrizen abgeleitet werden.

Der hier eingeschlagene Weg zur Berechnung der Inversen eines symmetrischen Systems beruht auf einer bereits durchgeführten Cholesky-Reduktion einer symmetrischen Matrix. Darauf aufbauend kann die Inverse sehr einfach berechnet werden. Neben dem für Genauigkeitsabschätzungen interessanten Zusammenhang zwischen der nach Cholesky reduzierten Form und der Inversen, bietet dieses Verfahren vor allem bei schwach besetzten Systemen große Vorteile (siehe Abschnitt 4.5).

Eine allgemein übliche Darstellung - ein hochgestelltes '(-1)' - kennzeichnet die einzelnen Blöcke der Inversen der Gesamtmatrix N.

$$\boldsymbol{N}^{-1} = \begin{bmatrix} \boldsymbol{N}_{11}^{(-1)} & \boldsymbol{N}_{12}^{(-1)} \\ \left(\boldsymbol{N}_{12}^{(-1)}\right)^T & \boldsymbol{N}_{22}^{(-1)} \end{bmatrix} \tag{2.134}$$

Die Inverse eines Blockes $\boldsymbol{N}_{11}$ wird wie immer durch ein hochgestelltes '-1' bezeichnet. Die Abhängigkeiten der Inversenblöcke von den Blöcken der Ausgangsmatrix kann durch den verallgemeinerten Austauschschritt (siehe Abschnitt 1.3) erhaltenen werden. Die Formeln (1.15) legen diese Beziehungen fest,

$$\boldsymbol{N}_{11}^{(-1)} = \boldsymbol{N}_{11}^{-1} + \boldsymbol{N}_{11}^{-1}\boldsymbol{N}_{12}\boldsymbol{N}_{22}^{(-1)}\boldsymbol{N}_{12}^T\boldsymbol{N}_{11}^{-1} \tag{2.135}$$

$$\boldsymbol{N}_{12}^{(-1)} = -\boldsymbol{N}_{11}^{-1}\boldsymbol{N}_{12}\boldsymbol{N}_{22}^{(-1)} \tag{2.136}$$

$$\boldsymbol{N}_{22}^{(-1)} = \left(\boldsymbol{N}_{22} - \boldsymbol{N}_{12}^T\boldsymbol{N}_{11}^{-1}\boldsymbol{N}_{12}\right)^{-1}. \tag{2.137}$$

Ziel der folgenden Umformungen ist es, ein Formelsystem zu finden, bei welchem die Blöcke der Inversen $\boldsymbol{N}^{-1}$ nur durch die Blöcke der oberen Dreiecksmatrix $\boldsymbol{R}$ ausgedrückt sind. Am leichtesten ist der untere Diagonalblock $\boldsymbol{N}_{22}^{(-1)}$ zu ermitteln. Vergleicht man das Glied $\boldsymbol{N}_{22}^{(-1)}$ in der Darstellung (2.137) mit dem Glied $\boldsymbol{R}_{22}$ aus Formel (2.107), so erkennt man, daß der Ausdruck innerhalb der Klammer derselbe ist. Will man das

Glied $\boldsymbol{R}_{22}$ in das Glied $\boldsymbol{N}_{22}^{(-1)}$ überführen, so kann entweder die Cholesky-Reduktion dieses Blockes unterlassen werden, oder man muß die durchgeführte Reduktion rückgängig machen. Durch die Inversion der quadratischen Form von $\boldsymbol{R}_{22}$ ist $\boldsymbol{N}_{22}^{(-1)}$ zu bestimmen,

$$\boldsymbol{N}_{22}^{(-1)} = \left(\boldsymbol{R}_{22}^T \boldsymbol{R}_{22}\right)^{-1} . \tag{2.138}$$

Zur Berechnung des Gliedes $\boldsymbol{N}_{12}^{(-1)}$ aus (2.136) kann zunächst $\boldsymbol{N}_{11}^{-1}$ aus (2.104) ersetzt werden. Verwendet man für $\boldsymbol{N}_{11}^{-1}$ die Beziehung (2.106) so erhält man

$$\boldsymbol{N}_{12}^{(-1)} = -\boldsymbol{R}_{11}^{-1} \boldsymbol{R}_{12} \boldsymbol{N}_{22}^{(-1)} \tag{2.139}$$

Das Glied $\boldsymbol{N}_{11}^{(-1)}$ kann durch Einsetzen in (2.135) für $\boldsymbol{N}_{12}$ (2.136) und $\boldsymbol{N}_{11}^{-1}$ (2.106) errechnet werden,

$$\boldsymbol{N}_{11}^{(-1)} = \left(\boldsymbol{R}_{11}^T \boldsymbol{R}_{11}\right)^{-1} - \boldsymbol{N}_{12}^{(-1)} \boldsymbol{R}_{12}^T \boldsymbol{R}_{11}^{-1} . \tag{2.140}$$

Die Formeln (2.138) bis (2.140) erlauben die Rückrechnung der Inversen aus der reduzierten Dreiecksmatrix. Auf den ersten Blick scheint diese Berechnung nicht sehr einfach. Wenn man jedoch nicht die ganze Inverse in einem Rechengang ermittelt, sondern schrittweise vorgeht, gelangt man zu folgender Berechnungsmöglichkeit. Ausgehend vom letzte Glied $r_n n$ der Dreiecksmatrix R und versucht die vorletzte Zeile der Inversen rückzurechnen. Aus (2.138) folgt zunächst

$$n_{nn}^{(-1)} = \frac{1}{r_{nn}^2} \tag{2.141}$$

Wegen (2.139) errechnet sich das Nebendiagonalglied mit

$$n_{n-1,n}^{(-1)} = -\frac{r_{n-1,n}}{r_{n-1,n-1}} \, n_{nn}^{(-1)}. \tag{2.142}$$

Das vorletzte Diagonalglied der Inversen kann aus (2.140)

$$n_{n-1,n-1}^{(-1)} = \frac{1}{r_{n-1,n-1}^2} - \frac{r_{n-1,n}}{r_{n-1,n-1}} \, n_{n-1,n}^{(-1)} \tag{2.143}$$

hergeleitet werden. Verwendet den soeben berechneten untersten 2x2 Block der Inversen zur Ermittlung der drittletzten Zeile und fährt in dieser Art zeilenweise fort, so können aus (2.139) und (2.140) die allgemeinen Formeln zur Rückrechnung der Inversenglieder abgeleitet werden,

$$\left.\begin{aligned} n_{ij}^{(-1)} &= -\tfrac{1}{r_{ii}} \sum_{k=i+1}^{n} r_{ik} n_{kj}^{(-1)}, \qquad j = i+1, \ldots, n \\ n_{ii}^{(-1)} &= \tfrac{1}{r_{ii}^2} - \tfrac{1}{r_{ii}} \sum_{k=i+1}^{n} r_{ik} n_{ik}^{(-1)} \; . \end{aligned}\right\} \quad i = n, \ldots, 1 \tag{2.144}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

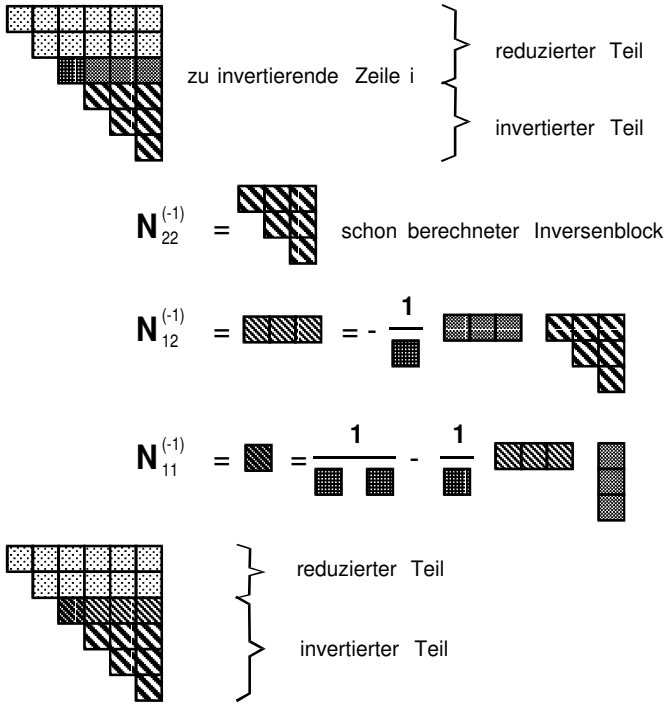


Abbildung 2.15: Grafische Darstellung des Inversionsverfahrens; Einzelschritt.

Die grafische Darstellung eines Berechnungsschrittes (Abb. 2.15) zeigt, daß die Abspeicherung der oberen Dreieckshälfte ausreicht um die Invertierung am Platz durchzuführen.

Als Wermutstropfen muß allerdings angeführt werden, daß sowohl ein zeilenweiser als auch ein spaltenweiser Zugriff unumgänglich ist. Die Anzahl der Rechenoperation ist mit

$$\frac{1}{3}\left(n^3 + 8n\right) \tag{2.145}$$

festgelegt. Wenn nur der obere Dreiecksteil verwendet werden darf, wird ein Vektor der Länge n als Zwischenspeicher benötigt. Kann auch der untere Dreiecksteil verwendet werden, so ist kein zusätzlicher Speicherbedarf erforderlich. Algorithmus 2.3 bietet die Möglichkeit analog zu Algorithmus 2.1 die Grundstruktur des Inversionsverfahren zu studieren, da die Verwendung der gesamten Matrix ein sehr klares Bild des Rechenablaufs ermöglicht.

Algorithmus 2.3 : Vollständige Inversion eines symmetrischen, positiv definiten Gleichungsystems, welches in einer Cholesky-reduzierten Form vorliegt und in einer Matrix gespeichert ist.

Aufgabenstellung:

Gegeben ist ein symmetrisches, positiv definites Gleichungssystem $\mathbf{N} \in \mathbb{R}^{n \times n}$, der Konstantenvektor $\mathbf{b}$ ist in der $(n+1)$-ten Spalte der Matrix $\mathbf{N}$ gespeichert. Das Gleichungssystem wird einer Cholesky-Reduktion unterzogen (z.B. Algorithmus 2.1), wobei die obere Dreiecksmatrix in reduzierter Form vorliegt. Gesucht ist die Inverse der Matrix N.

Schnittstellen:

Eingabe: `n(1:nn,1:nn)` ... Cholesky-reduzierte Koeffizienten der symmetrischen Matrix $\mathbf{N}$, nur Elemente $i \leq j$ (obere Dreiecksmatrix) erforderlich
`nn` ... Anzahl der Zeilen und Spalten des Systems
Ausgabe: `n(1:nn,1:nn)` ... Koeffizienten der Matrix N^{-1}
Felder: `n(nn,nn), hilf(nn)`

Hilfsproceduren:

`s = DOT_PRODUCT(vek1,vek2)` ... Skalarprodukt von `vek1` mit `vek2`.

Ausgangssituation:

$$\begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ \star & r_{22} & \dots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \star & \star & \dots & r_{1n} \end{bmatrix}$$

Anmerkung:
Es wird nur das obere Dreieck verwendet, der Teil unterhalb der Diagonale bzw. etwaige rechte Seiten bleiben unverändert.

Endsituation:

$$\begin{bmatrix} n_{11}^{(-1)} & n_{12}^{(-1)} & \dots & n_{1n}^{(-1)} \\ \star & n_{22}^{(-1)} & \dots & n_{2n}^{(-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \star & \star & \dots & n_{1n}^{(-1)} \end{bmatrix}$$

Lösungsweg: Pseudocode

1. *Berechnung des letzten Gliedes der Inversen:*
   `n(nn,nn) = 1 / n(nn,nn)`^{2} .
2. *Initialisiere Zeilenindex:*
   `i=nn` .

Wolf-Dieter Schuh
ITGB, Uni Bonn

3. *Vermindere Zeilenindex und überprüfe auf Schleifenende:*
   `i=i-1,`
   `IF i=1 GOTO Ende:` 'ENDE O.K.' .
4. *Berechne die Elemente (i,j) der i-ten Inversenzeile für die $i < j$ gilt und speichere sie in einem Hilfsvektor ab.*
   `hilf(i+1:nn) = - n(i,i+1:nn) n(i+1:nn,i+1:nn) / n(i,i) .`
5. *Berechnung des Diagonalgliedes (i,i):*
   `n(i,i) = 1 / n(i,i)`^{2} `- DOT_PRODUCT(n(i,i+1:nn),hilf(i+1:nn)) .`
6. *Umspeicherung der Elemente der Inversenzeile:*
   `n(i,i+1:nn) = hilf(i+1:nn)`
7. *Invertiere nächste Zeile:*
   `GOTO 3:` .

`Ende:`

Ressourcen: Anzahl der Operationen: $\frac{1}{3}(n^3 + 8n)$
Platzbedarf: $\frac{1}{2}n(n+1), n$

Die Vorteile dieses Verfahrens treten dann besonders hervor, wenn schwach besetzte Gleichungssysteme in der Form einer Band- oder Profilmatrix invertiert werden und nur bestimmte Elemente von Interesse sind (siehe dazu Abschnitt 4.5.2)

Anmerkung: Gleichung (2.141) liefert den Zusammenhang zwischen dem letzten Diagonalglied der Cholesky-Reduktion und dem Inversenglied, wodurch ein unmittelbarer Zugang zur Abschätzung der Größenordnung möglich wird. Da in der Anwendung in der Ausgleichsrechnung die Inverse Aussagen über Genauigkeiten liefert, kann durch die Gleichung (2.141) die Verbindung zwischen möglicher Genauigkeit und Signifikanz des Diagonalelementes gegenüber numerisch Null hergestellt werden.

2.5 Iterative Lösungsverfahren

Iterativen Lösungsverfahren oder Relaxationsmethoden liegen Rechenvorschriften zugrunde, bei denen eine Näherung schrittweise verbessert wird, um eine Lösung zu erhalten. Ausgehend von einer symmetrischen, positiv definiten Matrix N und dem Konstantenvektor b soll die Lösung x ermittelt werden

$$Nx = b \tag{2.146}$$

Ziel jeder Relaxationsmethode ist es, von einer beliebigen Startlösung $x^{(0)}$, die Lösung solange zu verbessern $x^{(1)}$, $x^{(2)}, \ldots, x^{(k)}$ bis der Residuenvektor $r^{(i)}$

$$r^{(i)} = Nx^{(i)} - b \tag{2.147}$$

verschwindet oder dessen Betrag unter einer bestimmten Schranke zu liegen kommt.

Eine vollkommen anderen Zugang erhält man, wenn man die quadratische Funktion $F(x)$,

$$\Phi(x) = \frac{1}{2}x^T Nx - x^T b \tag{2.148}$$

durch Veränderung des Vektors x zu minimieren versucht. Durch Differentiation nach der Veränderlichen gelangt man zur Extremwertbedingung

$$\frac{\partial \Phi(x)}{\partial x} = \text{grad}F = Nx - b. \tag{2.149}$$

Die Auflösung des symmetrischen, positiv definiten Gleichungssystems (2.146) ist daher gleichbedeutend mit der Aufgabe, das Minimum der quadratischen Funktion (2.148) zu bestimmen. Der Extremwert wird erreicht, wenn das Gleichungssystem (2.147) widerspruchslos erfüllt wird. Somit sind die Unbekannten x so zu bestimmen, daß alle Residuen r und damit alle partiellen Ableitungen von $\Phi(x)$ verschwinden, womit die Funktion $\Phi(x)$ für diese Werte x stationär wird. Wenn man positive Definitheit der quadratischen Form $\Phi(x)$ voraussetzt, muß diese Funktion genau eine stationäre Stelle und zwar ein Minimum besitzen. Die quadratische Funktion stellt ein Hyperparaboloid dar, wobei die offene Seite in Richtung der positiven $\Phi(x)$ Achse weist. Bei semidefiniten Systemen wird die Funktion $\Phi(x)$ auch stationär, nur liegt dann kein Minimum sondern ein Sattelpunkt vor.

2.5.1 Grundprinzip der Relaxation

Der Begriff Relaxation bedeutet Nachlassen, Erschaffen und meint ein sukzessives und systematisches Verkleinern der Reste (*R. Zurmühl (1964)[99]*, Seite ???). Ausgehend

von einer Startlösung $x^{(0)}$ wählt man einen vom Nullvektor verschiedenen Richtungsvektor p (*Relaxationsrichtung*), in dessen Richtung der Startvektor um einen Betrag τ korrigiert wird.

$$x^{(0)} = x^{(1)} + \tau p \tag{2.150}$$

sodaß der Wert der quadratischen Form (2.148) abnimmt. Verallgemeinert man diesen Schritt, so gelangt man zu der Iterationsformel

$$x^{(i+1)} = x^{(i)} + \tau^{(i)} p^{(i)} \tag{2.151}$$

Der Lösungsvektor $x^{(i)}$, des i-ten Schrittes wird übergeführt in einen verbesserten Vektor $x^{(i+1)}$, indem man in der Relaxationsrichtung $p^{(i)}$ ein Stück $\tau^{(i)}$ entlangwandert. Die Berechnung des Längenfaktors $\tau^{(i)}$ bei festem $x^{(i)}$ und $p^{(i)}$ ergibt sich durch die Minimierung der Funktion

$$\Phi\left(x^{(i+1)}\right) = \Phi\left(x^{(i)} + \tau^{(i)} p^{i}\right). \tag{2.152}$$

Setzt man in die quadratische Form (2.148) ein und ordnet das Ergebnis in bezug auf den Längenfaktor, erhält man unter Einbeziehung von (2.147)

$$\Phi\left(x^{(i+1)}\right) = \frac{1}{2}\tau^{(i)^2} p^{(i)^T} N p^{(i)} + \tau^{(i)} r^{(i)^T} p^{(i)} + \Phi\left(x^{(i)}\right). \tag{2.153}$$

Der Parameter $\tau^{(i)}$ ist so zu bestimmen, daß die Funktion $\Phi\left(x^{(i+1)}\right)$ in bezug auf die Richtung $p^{(i)}$ minimal wird,

$$\frac{d\Phi\left(x^{(i+1)}\right)}{d\tau^{(i)}} = \tau^{(i)} p^{(i)^T} N p^{(i)} + r^{(i)^T} p^{(i)} = 0. \tag{2.154}$$

Daraus ergibt sich

$$\tau^{(i)}_{min} = -\frac{r^{(i)^T} p^{(i)}}{p^{(i)^T} N p^{(i)}} \tag{2.155}$$

Der Beweis, daß $\tau^{(i)}_{min}$ wirklich die Funktion minimiert, geht direkt aus der zweiten Ableitung der Funktion $\Phi(x)$ hervor, wenn man berücksichtigt, daß $p^{(i)^T} N p^{(i)}$ wegen der positiven Definitheit von N und wegen $p^{(i)} \neq 0$ größer als Null ist.

Da jeder Schnitt des Hyperparaboloids parallel zur $\Phi(x)$-Achse eine nach oben offene Parabel darstellt, erkennt man, daß der Funktionswert $\Phi\left(x^{(i+1)}\right)$ für alle Werte von $\tau^{(i)}$ zwischen 0 und $2\tau^{(i)}$ kleiner als der Funktionswert $\Phi\left(x^{(i)}\right)$ wird.

Die Abnahme der quadratischen Funktion $\Phi(x)$ beim Übergang von $x^{(i)}$ zu $x^{(i+1)}$ folgt aus

$$\Delta\Phi = \Phi\left(x^{(i+1)}\right) - \Phi\left(x^{(i)}\right) = -\frac{1}{2}\frac{\left(p^{(i)^T} p^{(i)}\right)^2}{p^{(i)^T} N p^{(i)}} \leq 0\,. \tag{2.156}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

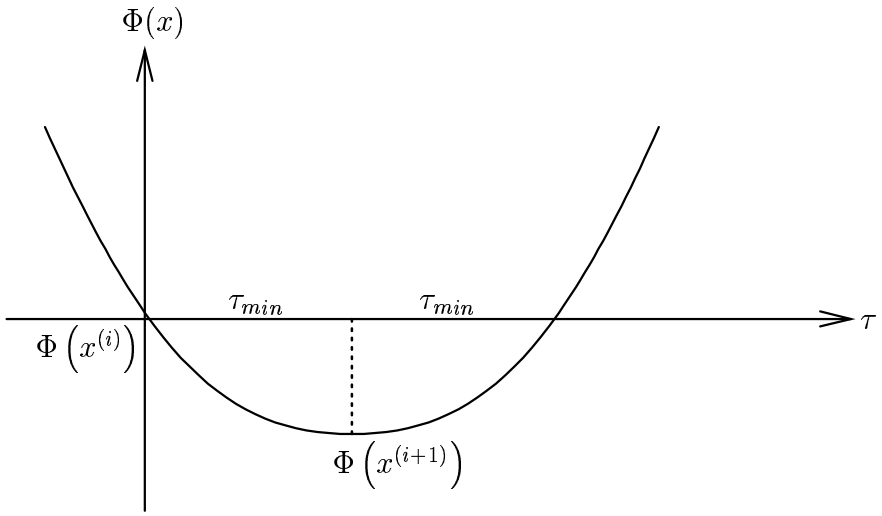


Abbildung 2.16: $\Phi\left(x^{(i+1)}\right)$ in Abhängigkeit des Längenfaktors $\tau^{(i)}$.

Die einzige Möglichkeit, keine Verringerung des Funktionswertes zu erhalten, ist dann gegeben, wenn die Relaxationsrichtung $p^{(i)}$ orthogonal auf den Residuenvektor $r^{(i)}$ steht. Diese Wahl ist daher zu vermeiden, da man somit am Näherungspunkt verweilen würde.

Abbildung 2.17 veranschaulicht die optimale Wahl des Längenfaktors $\tau_{min}^{(i)}$ für den zweidimensionalen Fall. Das Hyperparaboloid geht über in ein Paraboloid. Die Schnitte mit $\Phi(x) = const.$ stellen Ellipsen dar. Wenn man entlang der gewählten Relaxationsrichtung $p^{(i)}$ bis zum Punkt mit minimalen Funktionswert fortschreitet, so bedeutet dies, daß die Relaxationsrichtung $p^{(i)}$ in diesen Punkt zur Tangente der Niveaulinie wird.

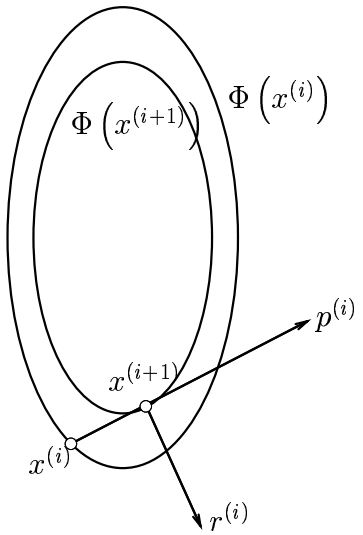


Abbildung 2.17: Optimale Wahl von $\tau^{(i)}$.

Da $p^{(i)}$ die Tangente an die Niveauellipse $\Phi\left(x^{(i+1)}\right)$ darstellt, muß der Gradientenvektor im Punkt $x^{(i+1)}$ - $\texttt{grad}\left(\Phi\left(x^{(i+1)}\right)\right) = r^{(i+1)}$, orthogonal zur Relaxationsrichtung $p^{(i)}$ stehen. Die formelmäßige Darstellung ergibt sich aus (2.147) unter Verwendung von (2.151)

$$r^{(i+1)} = Nx^{(i+1)} - b = r^{(i)} + \tau^{(i)} N p^{(i)} \tag{2.157}$$

und der Multiplikation mit $p^{(i)}$

$$p^{(i)^T} r^{(i+1)} = p^{(i)^T} r^{(i)} + \tau^{(i)} p^{(i)^T} N p^{(i)} \tag{2.158}$$

Setzt man für $\tau^{(i)}$ die optimale Wahl $\tau^{(i)}_{min}$ (2.155) ein, so erlangt man

$$p^{(i)^T} r^{(i+1)} = 0, \quad \text{für} \quad \tau^{(i)} = \tau^{(i)}_{min} \tag{2.159}$$

Bedingt durch unterschiedliche Aufgabenstellung und damit Gleichungssysteme mit verschiedenen Eigenheiten wurden mehrere Verfahren für schnelle und rechengünstige Lösungen entwickelt. Die Verfahren unterscheiden sich durch die Wahl der Relaxationsrichtung und des Längenfaktors.

Einzelschrittverfahren:	Handrelaxation (Jacobi-Verfahren)
	Überrelaxation
Gradientenverfahren:	Methode des stärksten Abstiegs
	Gesamtschrittverfahren
	Methode der konjugierten Gradienten

2.5.1.1 Gradientenmethoden

Auf Grund von (2.149) stellt der Residuenvektor $r^{(i)}$ (2.147) den Gradienten der zu minimierenden Funktion $\Phi(x)$ im Punkt $x^{(i)}$ dar und weist somit in die Richtung, in welcher $\Phi(x)$ lokal am stärksten abnimmt. Es ist somit nur natürlich, den Gradienten zur Festlegung der Relaxationsrichtung $p^{(i)}$ zu benützen. Relaxationsmethoden, welche den Residuenvektor $r^{(i)}$ oder bereits zuvor auftretende Residuenvektoren $r^{(k)}$, $\leq k < i$ zur Festlegung der Relaxationsrichtung verwenden, nennt man *Gradientenverfahren.*

Versucht man in der lokalen Situation das Beste zu machen, verwendet die Gradientenrichtung als Relaxationsrichtung und benutzt (2.155) für die optimale Festlegung des Längenfaktors $\tau^{(i)}$, so erhält man die Methode des stärksten Abstieges.

```
r = N x - b
p = -r
3: rtr = DOT_PRODUCT(r,r)
hilf = N r
tau = rtr / DOT_PRODUCT(r,hilf)
x = x + tau p
Abbruckskriterium: √rtr < num_Null
GOTO 3
```

Nach der Durchführung des i-ten Relaxationsschrittes steht der neue Residuenvektor $r^{(i+1)}$ orthogonal zum vorhergehenden Residuenvektor $r^{(i)}$ (siehe dazu (2.159) bzw. Abbildung 2.17). Während der Relaxation beschreibt man geometrisch betrachtet, im n-dimensionalen euklidischen Raum einen stückweise geradlinigen, rechtwinkeligen Weg, der schließlich im Mittelpunkt der Ellipsen (Ellipsoide) endet, wo der Funktionswert minimal ist.

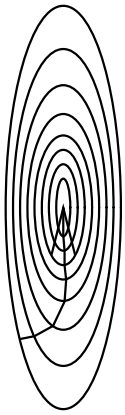


Abbildung 2.18: Methode des stärksten Abstiegs.

Schwierig ist dabei die Wahl des Abbruchskriteriums, da durch die Länge des Residuenvektors kein direkter Rückschluß auf die Genauigkeit der Unbekannten möglich ist. Verwendet man (**??**) so ist eine Abschätzung

$$\frac{|\Delta x|}{|x|} \leq \kappa_2(N) \frac{\left|r^{(i)}\right|}{|r^{(0)}|} \tag{2.160}$$

möglich, wobei die Konditionszahl $\kappa_2(N)$ das Verhältnis des größten zum kleinsten Eigenwert festlegt. Die globale Konvergenz ist abhängig vom Verhältnis der größten zur kleinsten Halbachse des Ellipsoides und kann durch die Umgleichung

$$\Phi\left(x^{(i)}\right) + \frac{1}{2} b^T N^{-1} b \leq \left(1 - \frac{\lambda_{min}}{\lambda_{max}}\right)\left(\Phi\left(x^{(i-1)}\right) + \frac{1}{2} b^T N^{-1} b\right) \tag{2.161}$$

(*G.H. Golub et.al. (1983)[38]*, Seite 363) abgeschätzt werden.

Bei der Berechnung nach dieser Methode erkennt man (siehe Abbildung 2.18), daß die Wahl der "besten" Strategie in einem individuellen Punkt nicht immer den besten Weg zur Lösung des Gesamtproblems darstellt.

Die Methode kann vereinfacht werden, indem nicht bei jedem Schritt der Längenfaktor berechnet wird, sondern über mehrere Schritt hinweg konstant gehalten wird. Für die Konvergenz dieser Methode als notwendige und hinreichende Bedingung muß

$$0 < \tau < \frac{2}{\lambda_{max}} \tag{2.162}$$

erfüllt sein (*E. Stiefel (1952)[83]*, Seite 20). Diese Bedingung kann man mit Hilfe der Gleichung

$$\tau_{min} = \frac{r^{(i)^T} r^{(i)}}{r^{(i)^T} N r^{(i)}} = \frac{1}{\mathtt{R}(\mathtt{r}^{(\mathtt{i})})} \tag{2.163}$$

und Abbildung 2.16 veranschaulichen. Die Gleichung stellt den Reziprok wert des *Rayheigh'schen Quotienten* dar, der durch $\lambda_{min} \leq R(x) \leq \lambda_{max}$ abgeschätzt werden kann. Durch die bei Abbildung 2.16 angestellten Betrachtungen ergibt sich eine Senkung des Funktionswertes nur im Bereich $0 \leq \tau \leq 2\tau_{min}$.

2.5.1.2 Methode der konjugierten Gradienten

Eine Verallgemeinerung der lokal besten Strategie des stärksten Abstiegs wird durch die geometrische Tatsache motiviert, daß jede Ebene normal zur Paraboloidachse (Funktionsachse) geschnitten mit der quadratischen Funktion als Schnittfigur eine Ellipse aufweist und somit die Menge aller Schnitte in einer Ellipsoide beschreibt. Betrachtet man die Situation in einem Näherungspunkt $x^{(i)}$ mit dem entsprechenden Schnittellipsoid, die die Niveaufläche von $\Phi(x)$ durch den Punkt $x^{(i)}$ darstellt. Die '*alte*' Relaxationsrichtung $p^{(i-1)}$ ist Tangente an das Schnittellipsoid, wenn der Längenfaktor $\tau^{(i-1)}$ durch (2.155) bestmöglich berechnet wird. Da alle Niveauflächen $\Phi(x) = const.$ konzentrische und ähnliche Ellipsoide bilden, liegt der Minimalpunkt der Funktion $\Phi(x)$ im gemeinsamen Zentrum. Wählt man die neue Relaxationsrichtung $p^{(i)}$ in der Form, daß $p^{(i)}$ und $p^{(i-1)}$ konjugiert in Bezug auf die Ellipsen $\Phi(x) = const.$ sind,

$$p^{(i)^T} N p^{(i-1)} = 0 \tag{2.164}$$

dann liegt das Zentrum der Ellipsoide auf diesem Strahl.

Da die Auflösung des homogenen Gleichungssystems (2.164) nach $p^{(i)}$ einen Rechenaufwand bedeuten würde, der einer direkten Lösung entspricht, sind zusätzliche Vorgaben notwendig, um die Rechnung zu vereinfachen. Dies geschieht durch Vorgabe einer Ebene, aufgespannt durch die Relaxationsrichtung $p^{(i-1)}$ und den Residuenvektor $r^{(i)}$, in der sich die neue Relaxationsrichtung $p^{(i)}$ befinden soll. $p^{(i)}$ läßt sich somit als Linearkombination von $r^{(i)}$ und $p^{(i-1)}$ darstellen, wobei der Koeffizient von $r^{(i)}$ von Null verschieden sein muß, um eine Verringerung des Funktionswertes zu erreichen. Er kann im Sinne einer Normierung mit -1 festgelegt werden, wodurch die eingeschränkte Relaxationsrichtung $p^{(i)}$ folgendermaßen dargestellt ist

$$p^{(i)} = -r^{(i)} + \varepsilon^{(i)} p^{(i-1)}. \tag{2.165}$$

Der unbekannte, skalare Faktor $\varepsilon^{(i)}$ wird durch die Bedingung der konjugierten Richtung (2.164) errechnet, indem die Darstellung (2.165) vorgegeben wird.

$$varepsilon^{(i)} = \frac{r^{(i)^T} N p^{(i-1)}}{p^{(i-1)^T} N p^{(i-1)}} \tag{2.166}$$

In der Relaxationsrichtung $p^{(i)}$ geht man bis zum Punkt mit dem kleinsten Funktionswert. Dies erfolgt durch Verwendung von (2.151) und (2.155).

$$x^{(i+1)} = x^{(i)} + \tau^{(i)} p^{(i)} \tag{2.151}$$

$$\tau_{min}^{(i)} = -\frac{r^{(i)^T} p^{(i)}}{p^{(i)^T} N p^{(i)}} \tag{2.155}$$

Nach dem durchgeführten $(i-1)$-ten Relaxationsschritt ist $x^{(i)}$ Minimalpunkt in der von $r^{(i-1)}$ und $p^{(i-1)}$ beziehungsweise $r^{(i-1)}$ und $p^{(i-1)}$ (siehe (2.165)) aufgespannten Ebene. Der neue Residuenvektor $r^{(i)}$ ist wegen der besten Wahl des Längenfaktors $\tau(i)$ orthogonal auf die Ebene von $r^{(i-1)}$ und $p^{(i-1)}$.

$$r^{(i)^T} r^{(i-1)} = 0 \tag{2.167}$$

$$r^{(i)^T} p^{(i-1)} = 0 \tag{2.168}$$

$$r^{(i)^T} p^{(i-2)} = 0 \tag{2.169}$$

Durch vollständige Induktion kann gezeigt werden, daß alle Residuenvektoren $r^{(i)}$, $i = 0, \ldots, k$ aufeinander orthogonal sind (siehe *H.R. Schwarz (1968)[74]*, Seite 73), wodurch folgende Orthogonalitätsbeziehungen gelten

$$r^{(i)^T} r^{(k)} = 0 \qquad \text{für} k = 0, 1, \ldots, i-1 \tag{2.170}$$

Durch Ausnützung dieser Beziehungen können vereinfachte Rechenformeln für den Längenfaktor $\tau^{(i)}$

$$\tau^{(i)} = \frac{r^{(i)^T} r^{(i)}}{p^{(i)^T} N p^{(i)}} \tag{2.171}$$

den skalaren Faktor $\varepsilon^{(i)}$ (2.166)

$$\varepsilon^{(i)} = \frac{r^{(i)^T} r^{(i)}}{r^{(i-1)^T} r^{(i-1)}} \tag{2.172}$$

und der Residuenvektor $r^{(i)}$

$$r^{(i)} = r^{(i-1)} + \tau(i-1) N p^{(i-1)}. \tag{2.173}$$

Da die Residuenvektoren $r^{(i)}$ aufeinander orthogonal sind (2.170), ist der von den Spalten von N aufgespannte Raum nach $\texttt{Rang}(N)$ Schritten durchwandert, womit die Lösung errechnet sein muß, da kein neuer Residuenvektor gebildet werden kann, der orthogonal auf alle vorher gehenden ist und im Spaltenraum von N liegt. Das Verfahren der konjugierten Gradienten stellt somit theoretisch einen endlichen Prozeß dar.

Bei der numerischen Durchführung zeigen sich Abweichungen der Orthogonalitäten der Residuenvektoren durch numerische Instabilitäten. Da die Lösung bei jedem Schritt verbessert wird, kann die Iteration über die theoretische Grenze von $\texttt{Rang}(N)$ Schritte geführten werden und so die Genauigkeit der Lösung gesteigert werden. Die Konvergenz ist durch diese Relaxationsmethode gewährleistet, da sich der Funktionswert $\Phi(x)$ bei jedem Schritt verringert. (Siehe (2.156).

The adjustment of observation equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{\ell} + \boldsymbol{v} \tag{2.174}$$

relies on the minimization of the sum of squared observation residuals,

$$\Phi(\boldsymbol{x}) = \boldsymbol{v}^T\boldsymbol{v} = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} - 2\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{\ell} + \boldsymbol{\ell}^T\boldsymbol{\ell}\ . \tag{2.175}$$

From the partial derivatives of the function $\Phi(\boldsymbol{x})$ with respect to the unknowns $\boldsymbol{x}$ we get the normal equations

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{\ell} \qquad \text{or} \qquad \boldsymbol{N}\boldsymbol{x} = \boldsymbol{n}\ . \tag{2.176}$$

The solution of the normal equations can be interpreted as the search for the minimal function value of the (hyper)paraboloid given by (2.175). This interpretation uses some iterative procedures to solve symmetric linear equation systems.

All iterative techniques have in common a stepwise improvement of the approximate solution vector

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + q_i\,\boldsymbol{p}^{(i)}. \tag{2.177}$$

This stepwise improvement of the approximate solution $\boldsymbol{x}^{(i)}$ relies on a (relaxation) direction $\boldsymbol{p}^{(i)}$ and a corresponding scale factor q_i. Depending on the choice of this factor and on the relaxation direction, several iterative procedures can be distinguished.

When the best local strategy at the approximate point $\boldsymbol{x}^{(i)}$ is applied, the direction of steepest descent (gradient) serves as the relaxation direction $\boldsymbol{p}_G^{(i)}$,

$$\boldsymbol{p}_G^{(i)} = \text{grad}\ \Phi\left(\boldsymbol{x}\right) = -\boldsymbol{r}^{(i)} \qquad \text{with} \qquad \boldsymbol{r}^{(i)} = \boldsymbol{N}\boldsymbol{x}^{(i)} - \boldsymbol{n}\ . \tag{2.178}$$

The factor q_i is chosen in such a way that the resulting new approximate point $\boldsymbol{x}^{(i+1)}$ yields a minimal function value of $\Phi(\boldsymbol{x})$ along this direction,

$$q_i = \frac{\boldsymbol{r}^{(i)^T}\boldsymbol{r}^{(i)}}{\boldsymbol{p}^{(i)^T}\boldsymbol{N}\boldsymbol{p}^{(i)}}\ . \tag{2.179}$$

This approach, known as the *Gradient* technique, can be improved by using the special shape of the level lines, which are concentric ellipses. Using this information, the new relaxation direction $\boldsymbol{p}_{CG}^{(i)}$ can be chosen as the conjugate diameter in the plane spanned by $\boldsymbol{p}_{CG}^{(i-1)}$ and $\boldsymbol{r}^{(i)}$. The direction of the conjugate diameter can be computed by a linear-combination of the old relaxation direction $\boldsymbol{p}_{CG}^{(i-1)}$, which is a tangent to the ellipse $\Phi\left(\boldsymbol{x}^{(i)}\right) = \text{const.}$, and the residual vector $\boldsymbol{r}^{(i)}$ [2].

$$\boldsymbol{p}_{CG}^{(i)} = -\boldsymbol{r}^{(i)} + e_i\,\boldsymbol{p}_{CG}^{(i-1)} \tag{2.180}$$

[2] Due to a local optimal scale factor q_{i-1}, the residual vector $\boldsymbol{r}^{(i)}$ is orthogonal to $\boldsymbol{p}_{CG}^{(i-1)}$ (cf. Fig. 2.19).

The factor e_i of the linear combination is determined by the condition of conjugate diameters

$$\boldsymbol{p}_{CG}^{(i)^T} \boldsymbol{N} \boldsymbol{p}_{CG}^{(i-1)} = 0 \tag{2.181}$$

and yields

$$e_i = \frac{\boldsymbol{r}^{(i)^T} \boldsymbol{r}^{(i)}}{\boldsymbol{r}^{(i-1)^T} \boldsymbol{r}^{(i-1)}} \quad . \tag{2.182}$$

The common center $\boldsymbol{x}^{(i+1)}$ of the concentric ellipses is found immediately if we look for the minimal function value along the direction $\boldsymbol{p}_{cg}^{(i)}$. The scale factor q_i computed by (2.179) again defines the optimal step-size.

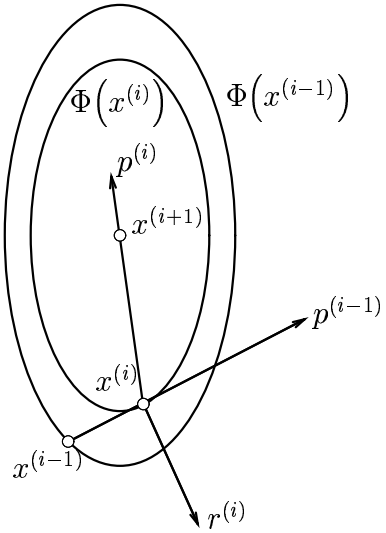


Abbildung 2.19: Conjugate Gradients.

The quality of the solution $\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + q_i \, \boldsymbol{p}^{(i)}$ can be estimated by the norm of the residuals $\boldsymbol{r}^{(i+1)}$, which can be efficiently computed[3] by the update formula

$$\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} + q_i \, \boldsymbol{N} \boldsymbol{p}^{(i)} \; . \tag{2.183}$$

The algorithm which is based on these formulas is called *Conjugate Gradient Method* (CG) and was introduced by *HESTENES et al. (1952) [42]*. It can be found in any textbook of numerical algebra (e.g. *GOLUB et al. (1983)[38]*, Sec. 10.2, *NASH (1979)[57]*, Sec. 16) as well as in dedicated books (*HESTENES (1980)[43]*) and reports (e.g. *ASH-KENAZI et al. (1973)[1]*, *GRÜNDIG (1980)[41]*, *SCHUH (1984)[73]*). *D'AZEVEDO et al. (1993)[22]* concentrate their attention on the adaptations of the conjugate gradient technique to a distributed memory multiprocessors architecture.

Alg. 2.4 summarizes the required computational steps and introduces the notation.

[3] The product $\boldsymbol{N} \boldsymbol{p}^{(i)}$ is already necessary to compute q_i according to equation (2.179).

Input/Output Parameters:

N	$[i,=]$	... normal matrix
n	$[i,=]$	... right hand side of the normals
$\#iter$	$[i,=]$	... number of iterations
$x^{(i)}$	$[i,o]$	... $x^{(0)}$ ... initial solution
		... $x^{(i)}$ ... solution at step $\#i$
rtr	$[o]$	... squared norm of the residuals $r^T r$

Algorithm:

$r^{(0)} = N * x^{(0)} - n$
$rtr = (r^{(0)})^T * r^{(0)}$
$p^{(0)} = -r^{(0)}$

! Iterations

$DO \;\; i = 0, \ldots, \#iter$
  $IF \; i > 0 \; THEN$
    $(rtr)_{old} = rtr$
    $rtr = (r^{(i)})^T * r^{(i)}$
    $e_i = rtr \; / \; (rtr)_{old}$
    $p^{(i)} = -r^{(i)} + e_i * p^{(i-1)}$
  $END \; IF$
  $h^{(i)} = N * p^{(i)}$
  $q_i = rtr \; / \; \left((p^{(i)})^T * h^{(i)} \right)$
  $x^{(i+1)} = x^{(i)} + q_i * p^{(i)}$
  $r^{(i+1)} = r^{(i)} + q_i * h^{(i)}$
$END \; DO \;\; i$

Algorithmus 2.4: Conjugate Gradient (CG) Algorithm.

Again, one important issue is the measurement of the convergence rate. The spectral radius ρ_{CG} is defined by (cf. *GOLUB et al. (1983)[38]*, p. 371, Eq. (10.2-16))

$$\rho_{CG} = \left(\frac{1 - \sqrt{\kappa_2(\boldsymbol{N})}}{1 + \sqrt{\kappa_2(\boldsymbol{N})}} \right)^2 , \tag{2.184}$$

where $\kappa_2(\boldsymbol{N})$ represents the condition number of $\boldsymbol{N}$, which is defined by the ratio between the largest and smallest singular value,

$$\kappa_2(\boldsymbol{N}) = \frac{\lambda_{max}}{\lambda_{min}} . \tag{2.185}$$

If we investigate the spectrum of the SGG data set (Fig. **??**), but also the SST data set (Fig. **??**) and the combined system (Fig. **??**), then we may expect a very slow convergence rate. Therefore, we stabilize the system by another matrix, which has the same characteristics like the normal equations, but can be computed more easily.

2.5.1.3 Vorkonditionuerte konjugierten Gradienten

For the stabilization of the normal equation system $\boldsymbol{N}\boldsymbol{x} = \boldsymbol{n}$ we have to look for an appropriate matrix, which reflects the characteristics of $\boldsymbol{N}$ as closely as possible, and is easy to handle at the same time. Because of the close relation to $\boldsymbol{N}$ this matrix is denoted a *representative matrix* $\boldsymbol{N}_{\oplus}$. To reduce the condition number and to accelerate the iterative procedure, the system $\boldsymbol{N}\boldsymbol{x} = \boldsymbol{n}$ is replaced by an auxiliary system

$$\begin{aligned} \bar{\boldsymbol{N}}\boldsymbol{x} = \bar{\boldsymbol{n}} \quad &\text{with} \quad \bar{\boldsymbol{N}} = \boldsymbol{N}_{\oplus}^{-1}\boldsymbol{N} \\ &\text{and} \quad \bar{\boldsymbol{n}} = \boldsymbol{N}_{\oplus}^{-1}\boldsymbol{n} \end{aligned} \tag{2.186}$$

by multiplying the original system with the inverse $\boldsymbol{N}_{\oplus}^{-1}$ of the representative matrix $\boldsymbol{N}_{\oplus}$. This step is never performed explicitly because it is impossible to compute this step for large systems. The preconditioned system (2.186) is only integrated into the iterative procedure.

To perform this preconditioning step in an integrated manner, we factorize the regular symmetric matrix $\boldsymbol{N}_{\oplus}$ by a product of a regular matrix $\boldsymbol{H}$ with its transposed

$$\boldsymbol{N}_{\oplus} = \boldsymbol{H}^T\boldsymbol{H} \ . \tag{2.187}$$

This can be done[4], e.g., by a *Cholesky-Factorization* (cf. App. **??**). The linear system $\boldsymbol{N}\boldsymbol{x} = \boldsymbol{n}$ then is extended to

$$\boldsymbol{H}^{-T}\boldsymbol{N}\boldsymbol{H}^{-1}\boldsymbol{H}\boldsymbol{x} = \boldsymbol{H}^{-T}\boldsymbol{n} \qquad \text{note:} \quad \boldsymbol{H}^{-T} = \left(\boldsymbol{H}^T\right)^{-1} \tag{2.188}$$

and is shortly denoted by

$$\tilde{\boldsymbol{N}}\tilde{\boldsymbol{x}} = \tilde{\boldsymbol{n}} \tag{2.189}$$

with the definitions

$$\tilde{\boldsymbol{N}} = \boldsymbol{H}^{-T}\boldsymbol{N}\boldsymbol{H}^{-1}, \qquad \tilde{\boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x} \qquad \text{and} \qquad \tilde{\boldsymbol{n}} = \boldsymbol{H}^{-T}\boldsymbol{n} \ . \tag{2.190}$$

[4] This factorization is only necessary to derive the formulas of the algorithm. During the computation this factorization can be used to support the solution step, but it is not obligatory to perform this factorization step explicitly. The background for this formulation is a preconditioning technique for sparse matrices $\boldsymbol{N}$, known as *Incomplete Cholesky Strategy* (cf. *KERSHAW (1978)[45], SCHWARZ (1981)[77]*). By neglecting all fill-ins during the Cholesky factorization, a sparse matrix $\boldsymbol{N}$ is split into two triangular matrices $\boldsymbol{H}$. Because of these simplifications the result of $\boldsymbol{H}^T\boldsymbol{H}$ produces a matrix which is similar to $\boldsymbol{N}$ and may be used as the representative matrix $\boldsymbol{N}_{\oplus}$.

The matrix $\boldsymbol{N}_{\oplus}$ and $\boldsymbol{H}$, resp., should be chosen such, that the condition number $\kappa_2\left(\boldsymbol{N}_{\oplus}^{-1}\boldsymbol{N}\right)$ becomes as small as possible. It is easy to show that the problems defined by (2.189) and (2.186), resp., are equivalent from the numerical point of view, because $\tilde{\boldsymbol{N}}$ and $\bar{\boldsymbol{N}}$ are *similar* matrices, which means their eigenvalues $\boldsymbol{\Lambda}$ are identical. Since $\boldsymbol{H}$ represents a nonsingular matrix, the product $\boldsymbol{H}^{-1}\tilde{\boldsymbol{N}}\boldsymbol{H}$ shares the eigenvalues with $\tilde{\boldsymbol{N}}$. But this product $\boldsymbol{H}^{-1}\tilde{\boldsymbol{N}}\boldsymbol{H}$ equals $\bar{\boldsymbol{N}}$, because

$$\boldsymbol{H}^{-1}\tilde{\boldsymbol{N}}\boldsymbol{H} = \boldsymbol{H}^{-1}\boldsymbol{H}^{-T}\boldsymbol{N}\boldsymbol{H}^{-1}\boldsymbol{H} = \boldsymbol{N}_{\oplus}^{-1}\boldsymbol{N} = \bar{\boldsymbol{N}} \ . \tag{2.191}$$

Depending on (2.189) the following relations between the preconditioned and original quantities exist:

$$\tilde{\boldsymbol{r}}^{(i)} = \boldsymbol{H}^{-T}\boldsymbol{r}^{(i)}, \qquad \tilde{\boldsymbol{p}}^{(i)} = \boldsymbol{H}^{-T}\boldsymbol{p}^{(i)}, \qquad \text{and} \qquad \tilde{\boldsymbol{x}}^{(i)} = \boldsymbol{H}\boldsymbol{x}^{(i)} \tag{2.192}$$

Therefore, the parameter update

$$\tilde{\boldsymbol{x}}^{(i+1)} = \tilde{\boldsymbol{x}}^{(i)} + \tilde{q}_i\,\tilde{\boldsymbol{p}}^{(i)} \tag{2.193}$$

can be rewritten by

$$\boldsymbol{H}\boldsymbol{x}^{(i+1)} = \boldsymbol{H}\boldsymbol{x}^{(i)} + \tilde{q}_i\,\boldsymbol{H}^{-T}\boldsymbol{p}^{(i)} \tag{2.194}$$

and after a left-multiplication with $\boldsymbol{H}^{-1}$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \tilde{q}_i\,\boldsymbol{H}^{-1}\boldsymbol{H}^{-T}\boldsymbol{p}^{(i)} \ , \tag{2.195}$$

and considering $\boldsymbol{N}_{\oplus}^{-1} = \boldsymbol{H}^{-1}\boldsymbol{H}^{-T}$ this results in

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \tilde{q}_i\,\boldsymbol{N}_{\oplus}^{-1}\boldsymbol{p}^{(i)} \ . \tag{2.196}$$

The same can be done for

$$\tilde{\boldsymbol{r}}^{(i+1)} = \tilde{\boldsymbol{r}}^{(i)} + \tilde{q}_i\,\tilde{\boldsymbol{N}}\tilde{\boldsymbol{p}}^{(i)} \tag{2.197}$$

and yields

$$\boldsymbol{r}^{(i+1)} = \boldsymbol{r}^{(i)} + \tilde{q}_i\,\boldsymbol{N}\boldsymbol{N}_{\oplus}^{-1}\boldsymbol{p}^{(i)} \ . \tag{2.198}$$

As a next step the relaxation direction $\tilde{\boldsymbol{p}}^{(i)}$

$$\tilde{\boldsymbol{p}}^{(i)} = -\tilde{\boldsymbol{r}}^{(i)} + \tilde{e}_i\,\tilde{\boldsymbol{p}}^{(i-1)} \tag{2.199}$$

is transformed into

$$\boldsymbol{p}^{(i)} = -\boldsymbol{r}^{(i)} + \tilde{e}_i\,\boldsymbol{p}^{(i-1)} \ . \tag{2.200}$$

Input/Output Parameters:

N	$[i,=]$	... normal matrix
n	$[i,=]$	... right hand side of the normals
$N_\oplus$	$[i,=]$	... representative part of the normal matrix
$\#iter$	$[i,=]$	... number of iterations
$x^{(i)}$	$[i,o]$	... $x^{(0)}$ ... initial solution
		... $x^{(i)}$ ... solution at step $\#i$
$rt\rho$	$[o]$	... squared norm of the residuals

Algorithm:

$r^{(0)} = N * x^{(0)} - n$
$\rho^{(0)} = SOLVE\left(N_\oplus, r^{(0)}\right)$
$rt\rho = (r^{(0)})^T * \rho^{(0)}$
$\Pi^{(0)} = -\rho^{(0)}$

! Iterations

$DO \;\; i = 0, \ldots, \#iter$
  $IF \; i > 0 \; THEN$
    $(rt\rho)_{old} = rt\rho$
    $rt\rho = (r^{(i)})^T * \rho^{(i)}$
    $\bar{e}_i = rt\rho \;/\; (rt\rho)_{old}$
    $\Pi^{(i)} = -\rho^{(i)} + \bar{e}_i * \Pi^{(i-1)}$
  $END \; IF$
  $h^{(i)} = N * \Pi^{(i)}$
  $\bar{q}_i = rt\rho \;/\; \left((\Pi^{(i)})^T * h^{(i)}\right)$
  $x^{(i+1)} = x^{(i)} + \bar{q}_i * \Pi^{(i)}$
  $r^{(i+1)} = r^{(i)} + \bar{q}_i * h^{(i)}$
  $\rho^{(i+1)} = SOLVE\left(N_\oplus, r^{(i+1)}\right)$
$END \; DO \;\; i$

Algorithmus 2.5: Preconditioned Conjugate Gradient (PCG) Algorithm.

The scalar products can be rewritten in terms of

$$\tilde{\boldsymbol{r}}^{(i)^T}\tilde{\boldsymbol{r}}^{(i)} = \boldsymbol{r}^{(i)^T}\boldsymbol{H}^{-1}\boldsymbol{H}^{-T}\boldsymbol{r}^{(i)} = \boldsymbol{r}^{(i)^T}\boldsymbol{N}_\oplus^{-1}\boldsymbol{r}^{(i)} \tag{2.201}$$

and

$$\tilde{\boldsymbol{p}}^{(i)^T}\tilde{\boldsymbol{N}}\tilde{\boldsymbol{p}}^{(i)} = \boldsymbol{p}^{(i)^T}\boldsymbol{H}^{-1}\boldsymbol{H}^{-T}\boldsymbol{N}\boldsymbol{H}^{-1}\boldsymbol{H}^{-T}\boldsymbol{p}^{(i)} = \boldsymbol{p}^{(i)^T}\boldsymbol{N}_\oplus^{-1}\boldsymbol{N}\boldsymbol{N}_\oplus^{-1}\boldsymbol{p}^{(i)} \; . \tag{2.202}$$

To simplify the computation,

$$\boldsymbol{\rho}^{(i)} = \boldsymbol{N}_{\oplus}^{-1}\boldsymbol{r}^{(i)} \tag{2.203}$$

is introduced. In addition the equations for $\boldsymbol{x}^{(i+1)}$ and $\boldsymbol{r}^{(i+1)}$ can be simplified by introducing

$$\boldsymbol{\Pi}^{(i)} = \boldsymbol{N}_{\oplus}^{-1}\boldsymbol{p}^{(i)} \tag{2.204}$$

which can be easily updated (cf. (2.200)) by

$$\boldsymbol{\Pi}^{(i)} = -\boldsymbol{\rho}^{(i)} + \tilde{e}_i\boldsymbol{\Pi}^{(i-1)} \ . \tag{2.205}$$

With this simplifications the scalar products can be computed by

$$\tilde{\boldsymbol{r}}^{(i)^T}\tilde{\boldsymbol{r}}^{(i)} = \boldsymbol{r}^{(i)^T}\boldsymbol{\rho}^{(i)} \tag{2.206}$$

and

$$\tilde{\boldsymbol{p}}^{(i)^T}\tilde{\boldsymbol{N}}\tilde{\boldsymbol{p}}^{(i)} = \boldsymbol{\Pi}^{(i)^T}\boldsymbol{N}\boldsymbol{\Pi}^{(i)} \ . \tag{2.207}$$

Algorithm 2.5 lists the basic steps of the *Preconditioned Conjugate Gradient* (PCG) algorithm. In connection with algorithm 2.4 the algorithm 2.5 serves as a basis of the subsequent considerations. Both algorithms allow several variations and enable the processing of large data sets which may be even heterogeneous.

Kapitel 3

Gleichungssysteme vom Type $\mathbf{A}^T\ \mathbf{A}$

3.1 Lineare Ausgleichsprobleme und damit verbundene Rechentechniken

3.2 Direkte Methoden beim Type $\mathbf{A}^T$ $\mathbf{A}$

Wolf-Dieter Schuh
ITGB, Uni Bonn

3.3 Iterative Methoden beim Type $\mathbf{A}^T$ $\mathbf{A}$

3.4 Solution Techniques using the Observation Equations

When we estimated the storage requirements (cf. Tab. **??**), we concluded that it is nearly impossible to assemble and store the full normal equations for high degree models. Therefore, we prefer to use the observation equations as such to perform the least squares harmonic analysis. This modification can be done within both iterative strategies. First the *Block-Jacobi* (BJ) algorithm is adapted and then the *Conjugate Gradient* (CG) algorithm is modified.

Because of the expected size of the matrices, special attention is necessary in order to implement correlated measurements. Therefore, this section is divided into two parts. In the first part an approach for uncorrelated, homogeneous measurements is elaborated. In the second approach correlated measurements are taken into account, and special emphasis is put on a frequency selective metric.

3.4.1 Uncorrelated Homogeneous Observations

The only part which has to be changed within the Block-Jacobi algorithm is the recomputation of the residuals. This requires the following substitutions in the BJ-algorithm:

$$r^{(i+1)} = r^{(i)} + N * p^{(i)} \quad \Longleftrightarrow \quad \left\{ \begin{array}{l} v^{(i+1)} = v^{(i)} + A * p^{(i)} \\ r^{(i+1)} = A^T * v^{(i+1)} \end{array} \right. . \tag{3.1}$$

The same step has to be performed also for the residuals $\boldsymbol{r}^{(0)}$. With the initial solution $\boldsymbol{x}^{(0)}$ the observation corrections $\boldsymbol{v}^{(0)}$ can be computed and with the orthogonality relation $\boldsymbol{A}^T\boldsymbol{v} = \boldsymbol{O}$ the first command can be exchanged by

$$r^{(0)} = N * x^{(0)} - n \qquad \Longleftrightarrow \quad \left\{ \begin{array}{l} v^{(0)} = A * x^{(0)} - \ell \\ r^{(0)} = A^T * v^{(0)} \end{array} \right. . \tag{3.2}$$

Of course, these computations need two matrix-vector products instead of just one, but the very time-consuming assembling of the full normal equations is avoided. We have to compute only those parts of the representative matrix $N_{\oplus}$ which reflect a sparse behavior. Algorithm 3.1 summarizes this approach step by step.

It should be mentioned that these two matrix-vector products can be performed within one cycle if the observation equations $\boldsymbol{A}$ are made available row by row $\boldsymbol{A}_{k,:}$. First a dot-product fixes the component v_k of the vector $\boldsymbol{v}$ [1]. Immediately afterwards an update step assembles this information in the vector $\boldsymbol{r}$ (cf. Alg. 3.2). If the observation equations have to be recomputed for each access, then the benefit of this implementation is obvious. But anyway, the minimization of memory access can accelerate the computations.

[1] The matrix-vector operation $\boldsymbol{A}_{k,:}\boldsymbol{y}$ corresponds to a synthesis of the spherical harmonics with respect to arbitrary coefficients $\boldsymbol{y}$.

Wolf-Dieter Schuh
ITGB, Uni Bonn

Input/Output Parameter:

A	$[i,=]$	... design matrix
ℓ	$[i,=]$	... observations
$N_\oplus$	$[i,=]$	... representative part of the normal equation matrix
$\#iter$	$[i,=]$	... number of iterations
$x^{(i)}$	$[i,o]$	... $x^{(0)}$ ... starting point solution
		... $x^{(i)}$ ... solution
rtr	$[o]$	... squared norm of the residuals $r^T r$
$v^{(i)}$	$[o]$	... observation residuals

Algorithm:

$v^{(0)} = A * x^{(0)} - \ell$

$r^{(0)} = A^T * v^{(0)}$

! Iterations

$DO \;\; i = 0, \ldots, \#iter$

$rtr = (r^{(i)})^T * r^{(i)}$

$p^{(i)} = -SOLVE\left(N_\oplus, r^{(i)}\right)$

$x^{(i+1)} = x^{(i)} + p^{(i)}$

$v^{(i+1)} = v^{(i)} + A * p^{(i)}$

$r^{(i+1)} = A^T * v^{(i+1)}$

$END \; DO \;\; i$

Algorithmus 3.1: Block-Jacobi Adjustment (BJA) Algorithm.

The same strategy can be used for the Conjugate Gradient approach. This yields the following changes within the CG-algorithm (Alg. 2.4),

$$\left.\begin{array}{l} h^{(i)} = N * p^{(i)} \\ q_i = rtr \;/\; \left((p^{(i)})^T * h^{(i)}\right) \\ x^{(i+1)} = x^{(i)} + q_i * p^{(i)} \\ r^{(i+1)} = r^{(i)} + q_i * h^{(i)} \end{array}\right\} \Longleftrightarrow \left\{\begin{array}{l} g^{(i)} = A * p^{(i)} \\ q_i = rtr \;/\; \left((g^{(i)})^T * g^{(i)}\right) \\ x^{(i+1)} = x^{(i)} + q_i * p^{(i)} \\ v^{(i+1)} = v^{(i)} + q_i * g^{(i)} \\ r^{(i+1)} = A^T * v^{(i+1)} \quad . \end{array}\right. \tag{3.3}$$

The initial steps have to be changed in the same way as before. Before we insert these steps in the algorithm we look again at the matrix-vector multiplications. These products appear in the first and last step of the changed part. At first sight it seems to be impossible to merge these steps and to reduce the access to the matrix $\boldsymbol{A}$ to a single row-by-row approach. But if we go into detail and substitute the update of the

$$v^{(i+1)} = v^{(i)} + A * p^{(i)}$$
$$r^{(i+1)} = A^T * v^{(i+1)}$$

Substituted by

$$r^{(i+1)} = 0$$
$$DO \quad k = 1, \ldots, \#n\_obs$$
$$\quad v_k^{(i+1)} = v_k^{(i)} + A(k,:) * p^{(i)}$$
$$\quad r^{(i+1)} = r^{(i+1)} + A(k,:)^T * v_k^{(i+1)}$$
$$END\ DO \quad k$$

Utility Algorithm 3.2: Implementation of the updates of $\boldsymbol{v}^{(i+1)}$ and $\boldsymbol{r}^{(i+1)}$ for a row-wise access to the uncorrelated homogeneous observation equations.

residuals

$$r^{(i+1)} = A^T * v^{(i+1)} \quad \Longleftrightarrow \quad \left\{ \begin{array}{l} h^{(i)} = A^T * g^{(i)} \\ r^{(i+1)} = r^{(i)} + q_i * h^{(i)} \end{array} \right. \tag{3.4}$$

we are able to reorder the computation to

$$\left. \begin{array}{l} h^{(i)} = N * p^{(i)} \\ q_i = rtr \ / \ \left((p^{(i)})^T * h^{(i)} \right) \\ x^{(i+1)} = x^{(i)} + q_i * p^{(i)} \\ r^{(i+1)} = r^{(i)} + q_i * h^{(i)} \end{array} \right\} \Longleftrightarrow \left\{ \begin{array}{l} g^{(i)} = A * p^{(i)} \\ h^{(i)} = A^T * g^{(i)} \\ q_i = rtr \ / \ \left((g^{(i)})^T * g^{(i)} \right) \\ x^{(i+1)} = x^{(i)} + q_i * p^{(i)} \\ v^{(i+1)} = v^{(i)} + q_i * g^{(i)} \\ r^{(i+1)} = r^{(i)} + q_i * h^{(i)} \end{array} \right. \tag{3.5}$$

and use the same strategy as documented in the utility algorithm 3.2 to compute the two matrix-vector products within one cycle. These steps are summarized in the utility algorithm 3.3.

It should be mentioned that almost the same changes can be applied to the PCG-algorithm. An algorithmic form of the CG- and PCG-Algorithm suited for the direct solution of observation equations is summarized in algorithm 3.4 (*Conjugate Gradient Adjustment* (CGA)) and 3.5 (*Preconditioned Conjugate Gradient Adjustment* (PCGA)). All these algorithms are suited for uncorrelated homogeneous observations. Only small changes are necessary to adapt these strategies also for correlated data sets.

3.4.2 Correlated, Band-limited Observations

From the mathematical point of view the extension to correlated observations is only a small step, but a large one from the computational point of view. The computation of

$g^{(i)} = A * p^{(i)}$
$h^{(i)} = A^T * g^{(i)}$

Substituted by

$h^{(i)} = 0$
$DO \quad k = 1, \ldots, \#n\_obs$
$\qquad g_k^{(i)} = A(k,:) * p^{(i)}$
$\qquad h^{(i)} = h^{(i)} + A(k,:)^T * g_k^{(i)}$
$END\ DO \quad k$

Utility Algorithm 3.3: Implementation of the computation of $\boldsymbol{g}^{(i)}$ and $\boldsymbol{h}^{(i)}$ for a row-wise access to the uncorrelated homogeneous observation equations.

the normal equations $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ has to be substituted by $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{M}\boldsymbol{A}$. The insertion of the metric $\boldsymbol{M}$ causes only small changes within the algorithms. We perform these changes exemplarily on the CG-algorithm, but the same strategies can be also applied to the other procedures.

For simplicity we demonstrate the changes with respect to the basic CG-algorithm, but having in mind that the new procedure only represents a mutation of the CGA-algorithm for homogeneous data sets,

$$\left.\begin{array}{l} h^{(i)} = N * p^{(i)} \\ q_i = rtr \;/\; \left((p^{(i)})^T * h^{(i)}\right) \\ x^{(i+1)} = x^{(i)} + q_i * p^{(i)} \\ r^{(i+1)} = r^{(i)} + q_i * h^{(i)} \end{array}\right\} \iff \left\{\begin{array}{l} g^{(i)} = A * p^{(i)} \\ f^{(i)} = M * g^{(i)} \\ h^{(i)} = A^T * f^{(i)} \\ q_i = rtr \;/\; \left((g^{(i)})^T * f^{(i)}\right) \\ x^{(i+1)} = x^{(i)} + q_i * p^{(i)} \\ v^{(i+1)} = v^{(i)} + q_i * g^{(i)} \\ r^{(i+1)} = r^{(i)} + q_i * h^{(i)} \quad . \end{array}\right. \tag{3.6}$$

If the metric $\boldsymbol{M}$ can be split into $\boldsymbol{F}^T\boldsymbol{F}$, the observation matrix $\boldsymbol{A}$ is substituted by the homogeneous counterpart $\hat{\boldsymbol{A}} = \boldsymbol{F}\boldsymbol{A}$, $\hat{\boldsymbol{\ell}} = \boldsymbol{F}\boldsymbol{\ell}$ and $\hat{\boldsymbol{v}} = \boldsymbol{F}\boldsymbol{v}$, and the standard CGA-algorithm can be used again. Of course, we can optimize the number of operations[2] avoiding the matrix-matrix operation $\boldsymbol{F}\boldsymbol{A}$ and preferring matrix-vector operations,

$$\left.\begin{array}{l} \hat{g}^{(i)} = F * A * p^{(i)} \\ h^{(i)} = A^T * F^T * \hat{g}^{(i)} \end{array}\right\} \iff \left\{\begin{array}{l} \hat{g}^{(i)} = F * (A * p^{(i)}) \\ h^{(i)} = A^T * (F^T * \hat{g}^{(i)}) \quad . \end{array}\right. \tag{3.7}$$

[2] if $\hat{\boldsymbol{A}}$ is too large to be stored

Input/Output Parameter:

A	$[i,=]$	... design matrix
ℓ	$[i,=]$	... observations
$\#iter$	$[i,=]$	... number of iterations
$x^{(i)}$	$[i,o]$	... $x^{(0)}$ ... starting point solution
		... $x^{(i)}$ ... solution
rtr	$[o]$	... squared norm of the residuals $r^T r$
$v^{(i)}$	$[o]$	... observation residuals

Algorithm:

$v^{(0)} = A * x^{(0)} - \ell$

$r^{(0)} = A^T * v^{(0)}$

$rtr = (r^{(0)})^T * r^{(0)}$

$p^{(0)} = -r^{(0)}$

! Iterations

$DO \;\; i = 0, \ldots, \#iter$

$IF \; i > 0 \; THEN$

$(rtr)_{old} = rtr$

$rtr = (r^{(i)})^T * r^{(i)}$

$e_i = rtr \; / \; (rtr)_{old}$

$p^{(i)} = -r^{(i)} + e_i * p^{(i-1)}$

$END \; IF$

$g^{(i)} = A * p^{(i)}$

$h^{(i)} = A^T * g^{(i)}$

$q_i = rtr \; / \; \left((g^{(i)})^T * g^{(i)} \right)$

$x^{(i+1)} = x^{(i)} + q_i * p^{(i)}$

$v^{(i+1)} = v^{(i)} + q_i * g^{(i)}$

$r^{(i+1)} = r^{(i)} + q_i * h^{(i)}$

$END \; DO \;\; i$

Algorithmus 3.4: Conjugate Gradient Adjustment (CGA) Algorithm.

From the mathematical point of view the whole problem is solved. We can choose the stochastic behavior of our data set and use one of the iterative strategies to determine the solution. In section **??** we elaborated three different approaches to model the frequency selective behavior of data sets:

- frequency selective filter (Sec. **??**),
- frequency selective target function in a time-domain approach (Sec. **??**) and

Input/Output Parameter:

A	$[i,=]$	... design matrix
ℓ	$[i,=]$	... observations
$N_\oplus$	$[i,=]$	... representative part of the normal equation matrix
$\#iter$	$[i,=]$	... number of iterations
$x^{(i)}$	$[i,o]$	... $x^{(0)}$ ... starting point solution
		... $x^{(i)}$ ... solution
$rt\rho$	$[o]$	... squared norm of the residuals
$v^{(i)}$	$[o]$	... observation residuals

Algorithm:

$v^{(0)} = A * x^{(0)} - \ell$
$r^{(0)} = A^T * v^{(0)}$
$\rho^{(0)} = SOLVE\left(N_\oplus, r^{(0)}\right)$
$rt\rho = (r^{(0)})^T * \rho^{(0)}$
$\Pi^{(0)} = -\rho^{(0)}$

! Iterations

$DO\ \ i = 0, \ldots, \#iter$
  $IF\ i > 0\ THEN$
    $(rt\rho)_{old} = rt\rho$
    $rt\rho = (r^{(i)})^T * \rho^{(i)}$
    $\bar{e}_i = rt\rho\ /\ (rt\rho)_{old}$
    $\Pi^{(i)} = -\rho^{(i)} + \bar{e}_i * \Pi^{(i-1)}$
  $END\ IF$
  $h^{(i)} = A * \Pi^{(i)}$
  $\bar{q}_i = rt\rho\ /\ \left((h^{(i)})^T * h^{(i)}\right)$
  $x^{(i+1)} = x^{(i)} + \bar{q}_i * \Pi^{(i)}$
  $v^{(i+1)} = v^{(i)} + \bar{q}_i * h^{(i)}$
  $r^{(i+1)} = A^T v^{(i+1)}$
  $\rho^{(i+1)} = SOLVE\left(N_\oplus, r^{(i+1)}\right)$
$END\ DO\ \ i$

Algorithmus 3.5: Preconditioned Conjugate Gradient Adjustment (PCGA) Algorithm.

- frequency selective target function in a frequency-domain approach (Sec. **??**).

All these approaches can also handle huge data sets. The frequency selective filter

is based on very simple difference equations with only a small number of constant coefficients. In contrast to that the two last approaches take advantage of the efficiency of the discrete fast Fourier transform (DFFT). But these FFT based approaches have a severe disadvantage: they need the whole information of a vector at once to perform the transition to the frequency-domain. In our approach we presume that the observation equations are available only row-by-row or in small blocks, but it seems to be impossible or very inefficient to store the whole matrix or to provide a column-wise access.

Therefore, only the first approach allows a step-by-step computation, which can be performed by (cf. **??**)

$$\bar{\ell}[n] = \sum_{l=0}^{nb} \bar{b}_l \; \ell[n-l] - \sum_{l=1}^{na} \bar{a}_l \; \bar{\ell}[n-l] \qquad \text{with} \quad \bar{a}_l = \frac{a_l}{a_0} \; , \quad \bar{b}_l = \frac{b_l}{a_0} \; . \tag{3.8}$$

With the help of signal-flow graphs (cf. App. **??**), this filter implementation can be optimized with respect to operations, storage requirements and numerical stability. Algorithm 3.6 shows an efficient implementation which is characterized by a minimum number of delay elements. Writing the algorithm, we have for the sake of convenience assumed that $na = nb$. If this is not the case, some of the update steps have to be slightly modified. The *memory* of this filter is reduced from two delay sequences $\ell[n-1], \ldots, \ell[n-nb]$ and $\bar{\ell}[n-1], \ldots, \bar{\ell}[n-nb]$ to one sequence $reg_1, \ldots, reg_{na}$ of registers only[3].

```
! Computational Step to compute ℓ̄[n]
ℓ̄[n] = b_0 ℓ[n] + reg_1
! Update Step to compute reg_i
DO  i = 1, ..., #na − 1
      reg_i = b_i ℓ[n] + reg_{i+1} − a_i ℓ̄[n]
END DO  i
reg_na = b_na ℓ[n] − a_na ℓ̄[n]
```

Utility Algorithm 3.6: $\bar{\ell}[n] = FILTER(\ell[n])$; efficient filter implementation with minimized number of registers.

The initial state condition that past inputs as well as outputs are zero ($\bar{\ell}[i] = 0$, $\ell[i] = 0$, $i < 0$), corresponds to a '*zero memory*' ($reg_1, \ldots, reg_{na} = 0$) of the digital filter. Introducing this filter into the iterative procedures, we have to perform the following typical combination of operations

$$\begin{aligned} \hat{g}^{(i)} &= F * A * p^{(i)} \\ h^{(i)} &= A^T * F^T * \hat{g}^{(i)} \; . \end{aligned} \tag{3.9}$$

[3] A detailed discussion of the intermediate steps can be found in appendix **??**.

In section **??** we have shown that the application of a digital filter $\bar{\boldsymbol{\ell}} = FILTER(\boldsymbol{\ell})$ is equivalent to the matrix-vector operation $\bar{\boldsymbol{\ell}} = \boldsymbol{F}\boldsymbol{\ell}$. In the same manner we can filter each column of the matrix $\boldsymbol{A}$. But we can also combine the input to a multi-input stream and construct a filter which acts on this multiple input in a parallel mode, $\hat{\boldsymbol{A}} = FILTER(\boldsymbol{A})$. Applying this filter to the formulas (3.9) we get

$$\begin{aligned} \hat{A} &= FILTER(A) \\ \hat{g}^{(i)} &= \hat{A} * p^{(i)} \\ h^{(i)} &= \hat{A}^T * \hat{g}^{(i)} \ . \end{aligned} \tag{3.10}$$

This results in the utility algorithm 3.7 if we consider a row-wise access to the matrix $\boldsymbol{A}$.

$g^{(i)} = A * p^{(i)}$
$h^{(i)} = A^T * g^{(i)}$

Substituted by

INIT_FILTER(#u)
$h^{(i)} = 0$
$DO \;\; k = 1, \ldots, \#n\_obs$
$\quad \hat{A}(k,:) = FILTER(A(k,:))$
$\quad \hat{g}_k^{(i)} = \hat{A}(k,:) * p^{(i)}$
$\quad h^{(i)} = h^{(i)} + \hat{A}(k,:)^T * \hat{g}_k^{(i)}$
$END \; DO \;\; k$

Utility Algorithm 3.7: Implementation of the computation of $\boldsymbol{g}^{(i)}$ and $\boldsymbol{h}^{(i)}$ for a row-wise access to equations of a band-limited data set.

3.5 Combined Systems and Parallel Computations

The last changes demonstrate in a very impressive way the variability of these iterative procedures: Let us expand the algorithms by including more than one group of (uncorrelated or correlated) observations, and at the same time add normal equations from another data set (e.g. a low degree and order data set). Because of the condition

$$\boldsymbol{N} = \sum_s \boldsymbol{A}_s^T \boldsymbol{A}_s + \sum_s \boldsymbol{A}_s^T \boldsymbol{M}_s \boldsymbol{A}_s + \sum_s \boldsymbol{A}_s^T \boldsymbol{F}_s^T \boldsymbol{F}_s \boldsymbol{A}_s + \sum_s \hat{\boldsymbol{A}}_s^T \hat{\boldsymbol{A}}_s + \sum_t \boldsymbol{N}_t \tag{3.11}$$

for uncorrelated groups of data, the residual vector can also be summed up individually. This enables the transfer of this summation process to different processing units and

to speed up this time-consuming part of the algorithm. Within the final procedures these parallel parts are documented by brackets { } and sums. If these computations are performed on a massive parallel system with distributed memory, each node is responsible for a special part of the observations and normals. Therefore, it might be possible to recompute the observation equations within each iteration if the hard-disk storage capacity is too small to hold the whole system. The final algorithms show that within each step this computation of the normals has to be done only once, because the two operations $\boldsymbol{Au}$ and $\boldsymbol{A}^T\boldsymbol{v}$ can be done simultaneously (cf. utility algorithms within section 3.4).

3.6 Gross Error Detection

Let us assume that we have a combined system with normal equations and observation equations. Therefore we are free to compute the observational corrections $v^{(i)}$ within each iteration step. This can be done individually within each block. If the observation corrections v are available, we can apply statistical techniques such as variance estimation for each group of observation components or individual tests for each observation. Robust estimation techniques and re-weighting procedures can be directly incorporated into these iterative procedures (cf. SCHUH (1989)[**?**]).

Wolf-Dieter Schuh
ITGB, Uni Bonn

Wolf-Dieter Schuh
ITGB, Uni Bonn

Kapitel 4

Schwach besetzte Gleichungssysteme

Bei vielen Anwendungen kommt es, sobald die Dimension des Gleichungssystems ansteigt, zu einem speziellem Typ von Matrizen, da keine direkten Beziehungen zwischen allen Unbekannten vorhanden sind. Die einzelnen Unbekannten haben nur auf wenige andere Variable einen direkten Einfluß. Die Verbindung zu allen anderen Variablen sind indirekter Natur, also nur über Zwischenvariable gegeben. Solche Strukturen können durch Netzwerke oder Graphen veranschaulicht werden.

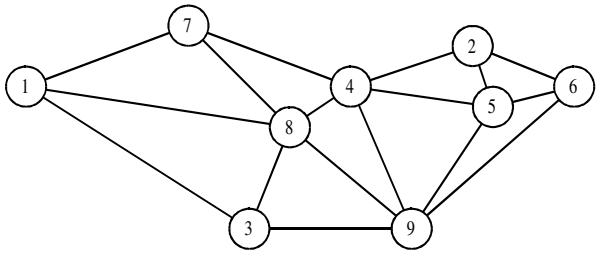


Abbildung 4.1: Knoten-Kanten-Struktur

Die Unbekannten stellen die Knoten des Netzes dar. Direkte Beziehungen zwischen den einzelnen Knoten sind durch Linien gekennzeichnet, die man als Kanten bezeichnet. Die aus solchen Strukturen resultierenden Gleichungssysteme weisen sehr viele Koeffizienten mit dem Wert Null (Nullelemente) auf und werden daher *schwach besetzte Gleichungssysteme* genannt. Durch geeignete Vorkehrungen kann dieser Umstand genützt werden um einerseits die Berechnungen zu beschleunigen und andererseits Speicherplatz zu sparen. Als Standardwerk über schwach besetzte Matrizen kann das Buch von *GEORGE et al. (1981)[34]* betrachtet werden. Darstellungen für spezielle Anwendungen und Verfahren bei finiten Elementansätzen sind bei *SCHWARZ (1980)[76]* und *(1981)[77]* zu finden. Umfangreichere Überblicksarbeiten mit Anwendungen wurden u.a. von *CUTHILL (1972)[19]*, *SCHEK et al. (1977)[71]* und *SCHUH (1981)[72]* veröffentlicht. Hier werden Verfahren dargestellt, die den Lösungsprozeß von schwach besetzten Gleichungssystemen in sehr effizienter Art ermöglichen. Zuerst wird auf die

bestmöglicher Erhaltung der schwachbesetzten Struktur während des Eliminationsprozesses eingegangen. Durch die Umordnung der Reihenfolge der Knoten wird versucht, möglichst viele Nullelemente beizubehalten. Die Algorithmen die diese Umordnung bewerkstelligen werden im Abschnitt 4.4 erarbeitet. Die sich ergebenden Konsequenzen bei der Durchführung von direkten Lösungsverfahren werden an Hand der Cholesky-Reduktion aufgezeigt, wobei speziell auf die Speichertechnik, die Lösung und die Rückrechnung von Gliedern der Inversen eingegangen wird (Kap. 4.5). Auf mögliche Auswirkungen der schwachen Besetztheit bei indirekten Methoden wird im Kapitel 4.6 hingewiesen, wobei speichertechnische Aspekte und die Vorkonditionierung durch eine unvollständige Cholesky-Reduktion im Vordergrund stehen.

Literatur:

[19] CUTHILL E. (1972): Several Strategies for Reducing the Bandwidth of Matrices. ROSE D.J. and R.A. WILLOUGHBY (1972): (Editors) 'Sparse Matrices and their Applications', Academic Press, New York-London.

[34] GEORGE Alan and Joseph W-H. LIU (1981): Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

[71] SCHEK Hans-Jörg, Franz STEIDLER und Ulrich SCHAUER (1977): Ausgleichung grosser geodätischer Netze mit Verfahren für schwach besetzte Matrizen. Deutsche Geodätische Kommission (DGK), Reihe A, Heft 87.

[72] SCHUH Wolf-Dieter (1981): Programmierung rationeller Algorithmen zur Umordnung, Auflösung und Inversion der Normalgleichungen geodätischer Netze. Diplomarbeit, TU Graz.

[76] SCHWARZ Hans Rudolf (1980): Methode der finiten Elemente. Teubner Studienbücher für Mathematik, Band 47.

[77] SCHWARZ Hans Rudolf (1981): FORTRAN - Programme zur Methode der finiten Elemente. Teubner Studienbücher für Mathematik.

4.1 Einführung

Zur Untersuchung der Eigenschaften schwach besetzter Matrizen können sehr vorteilhaft einige Grundbegriffe aus der Graphentheorie verwendet werden. Der Zusammenhang kann durch Identifikation der k-ten Unbekannten mit dem **Knoten** x_k und des Matrizenelementes n_{ij} mit einer **ungerichtete Kante** (x_i, x_j) hergestellt werden. Ein Graph setzt sich aus der Menge von n Knoten $x_1, x_2, \ldots, x_n$, die eine durchlaufende willkürliche Numerierung aufweisen, und aus der Menge der ungericheten Kanten, die eine Teilmenge aller möglichen Kanten bilden, zusammen. Wenn eine Kante keinen der Knoten als Anfangs- oder Endknoten speziell ausweist, spricht man von einer ungerichteten Kante. Gehört die Kante (x_i, x_j) zum Graph, so ist auch (x_j, x_i) ein Element des Graphen. Ein Graph enthält definitionsgemäß keine sogenannten Schleifen (x_i, x_i). Aus dem Graph geht somit nicht hervor, ob die Diagonalelemente gleich oder ungleich Null sind. Im Hinblick auf die Anwendung bei positiv definiten Matrizen ist dies bedeutungslos, da die Diagonalglieder immer positiv sind.

Mögliche Formen der Abspeicherung sind in Abb. 4.2 dargestellt. Neben einer ungeordneten Abspeicherung sind vor allem eine knotenweise geordnete (statische) Speicherung und eine knotenweise gekettete (dynamische) Speicherung von Bedeutung.

Durch die Äquivalenz der beiden Darstellungsformen kann jede Matrix durch einen Graph repräsentiert werden und umgekehrt. Eine spezielle Form, die nur die Struktur der Kanten veranschaulicht, stellt die Verknüpfungsmatrix darstellen (Abb. 4.3).

Um für das Nachfolgende eine gemeinsame Sprache zu finden, sind einige Vereinbarungen notwendig. Zwei Knoten x_i und x_j eines Graphen heißen **benachbart**, falls sie durch eine Kante direkt verbunden sind. Alle Nachbarn eines Knotens bilden die Menge der benachbarten Knoten. Unter dem **Grad eines Knotens** versteht man die Anzahl der Nachbarn, also die Anzahl der Kanten an denen dieser Knoten beteiligt ist. Alle benachbarten Knoten einer Gruppe von Knoten werden in der **Menge der Anrainer** zusammengefaßt, wobei eine direkte Verbindung zu einem Knoten der Gruppen für die Zugehörigkeit zu den Anrainern ausreichend ist. Für die Menge der Anrainer der Anrainer wird der Begriff der **Zweitanrainer** geprägt. Durch die Einführung eines Abstandsbegiffes zwischen zwei Knoten können diese Gruppen besser beschrieben werden. Als **Abstand** zwischen zwei Knoten wird die geringste Anzahl von Kanten verstanden, die notwendig ist um von einem Knoten zum anderen zu kommen. Zwei benachbarte Knoten weisen somit den Abstand 1 auf. In analoger Form läßt sich ein Abstandsbegriff zwischen einer Gruppe von Knoten und einem einzelnen Knoten beschreiben als die geringste Anzahl von Kanten die notwendig ist, um von einem beliebigen Knoten der Gruppe zum speziellen Knoten zu gelangen. Alle Anrainer sind somit durch den Abstand 1 gekenntzeichnet. Die Zweitanrainer einer Gruppe von Knoten weisen den Abstand 2 auf. Der größtmögliche Abstand in einem Graphen wird als **Durchmesser** bezeichnet. All diese Begriffe beziehen sich sich auf einen **zusammenhängenden** Graphen, der dadurch festgelegt ist, daß zwischen zwei beliebigen Knoten eine Folge von Kanten (Kantenzug) existiert. Knoten zwischen denen ein Kantenzug existiert, nennt man **verbunden**.

Von jedem Graphen kann durch Spezifizierung einer Gruppe von Knoten und Kanten

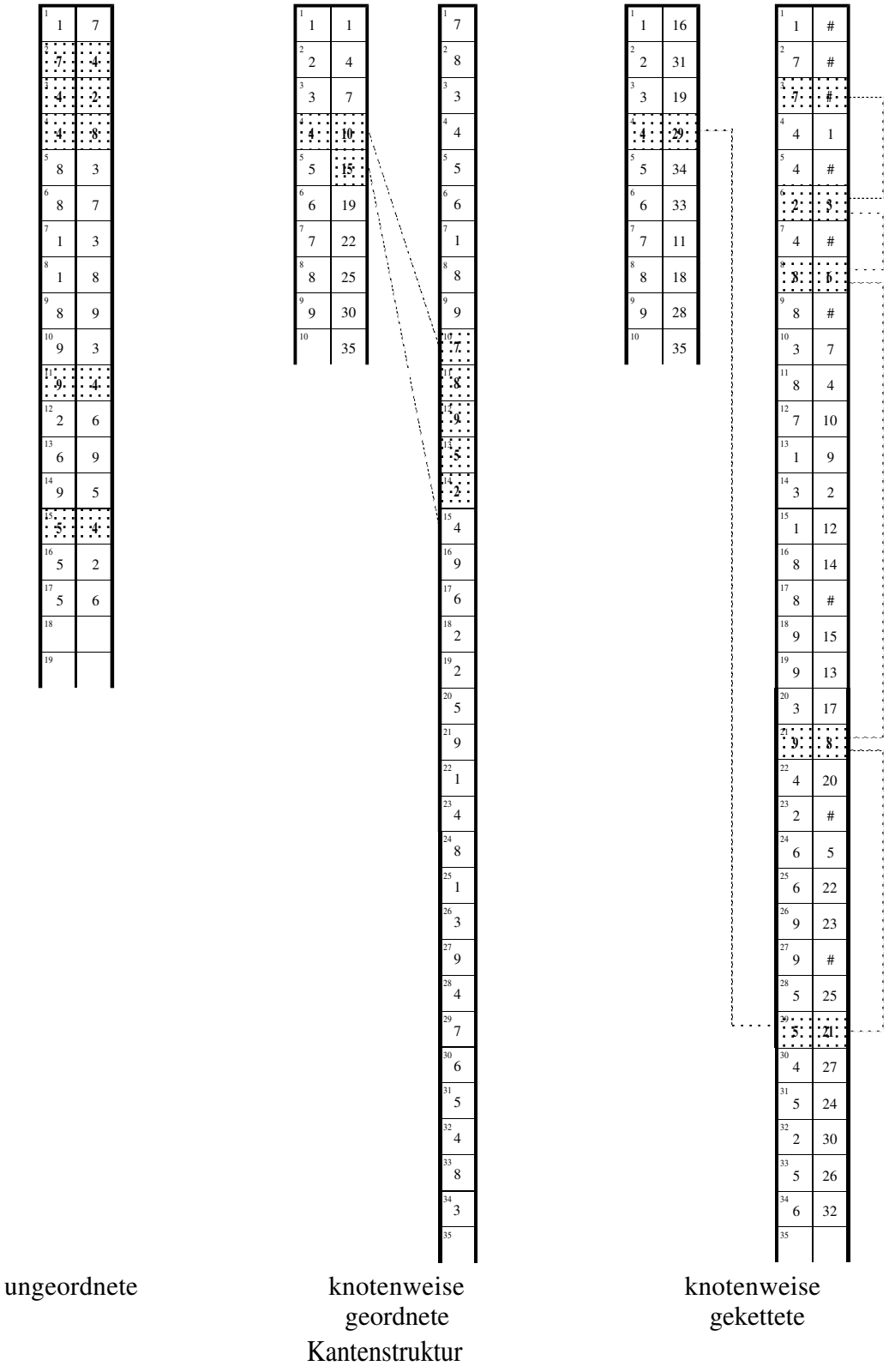


Abbildung 4.2: Speicherung eines Graphen

	1	2	3	4	5	6	7	8	9
1	×		×				×	×	
2		×		×	×	×			
3	×		×					×	×
4		×		×	×		×	×	×
5		×		×	×	×			×
6		×			×	×			×
7	×			×			×	×	
8	×		×	×			×	×	×
9			×	×	×	×		×	×

Abbildung 4.3: Verknüpfungsmatrix

ein **Untergraph** (Teilmenge des Graphen) gebildet werden, der selbst wieder Graph ist. Durch Herausnahme eines Untergraphen aus dem ursprünglichen Graphen kann ein verbleibender (komplemetärer) Untergraph erzeugt werden, der zusammenhängend sein kann, aber auch in mehrere **isolierte** (nicht zusammenhängende) Gruppen zerfallen kann. Hat ein Graph einen Knoten mehr als Kanten und besitzt er keine isolierte Knoten (d.h. jeder Knoten ist mit jedem verbunden) so spricht man von einem **Baum**. Ein **gespannter Baum** eines Graphen ist ein Untergraph, welcher zugleich Baum ist und alle Knoten des Graphen enthält.

Mit der Graphendarstellung läßt sich sehr anschaulich die Auswirkung eines Eliminationsschrittes auf die Stuktur der Restmatrix darstellen, wobei folgende Regel gilt (siehe Abb. 4.4):

Eliminationsregel:

Nach der Elimination eines Knotens x_i aus einem System, dem eine Verknüpfungsmatrix $\boldsymbol{N}_i$ zugeordnet ist, entsteht eine neue Verknüpfungsmatrix $\boldsymbol{N}_{i+1}$. Diese neue Matrix entspricht ein neuer Graph, der sich vom ursprünglichen Graphen folgendermaßen unterscheidet:

1. Durch das Fehlen des Knotens x_i und aller mit ihm verbundener Kanten.
2. Durch Verbindungen aller in dem Knoten x_i benachbarten Knoten untereinander. Jede Verbindung zwischen zwei Knoten, die bisher nicht verbunden waren, erzeugt eine neue Kante, beziehungsweise ein neues Nichtnullelement (**Füllelement**) in der Verknüpfungsmatrix.

Ein vollkommen anderer Zugang ist durch eine symbolische Elimination der Spalte und Zeile in der Verknüpfungsmatrix möglich. Bei einer symbolischen Elimination liegt das Interesse nicht an den numerischen Werten, sondern nur daran, ob eine Wert Null ist, Null bleibt oder ungleich Null wird. Dies kann kann Hand eines Austauschschrit-

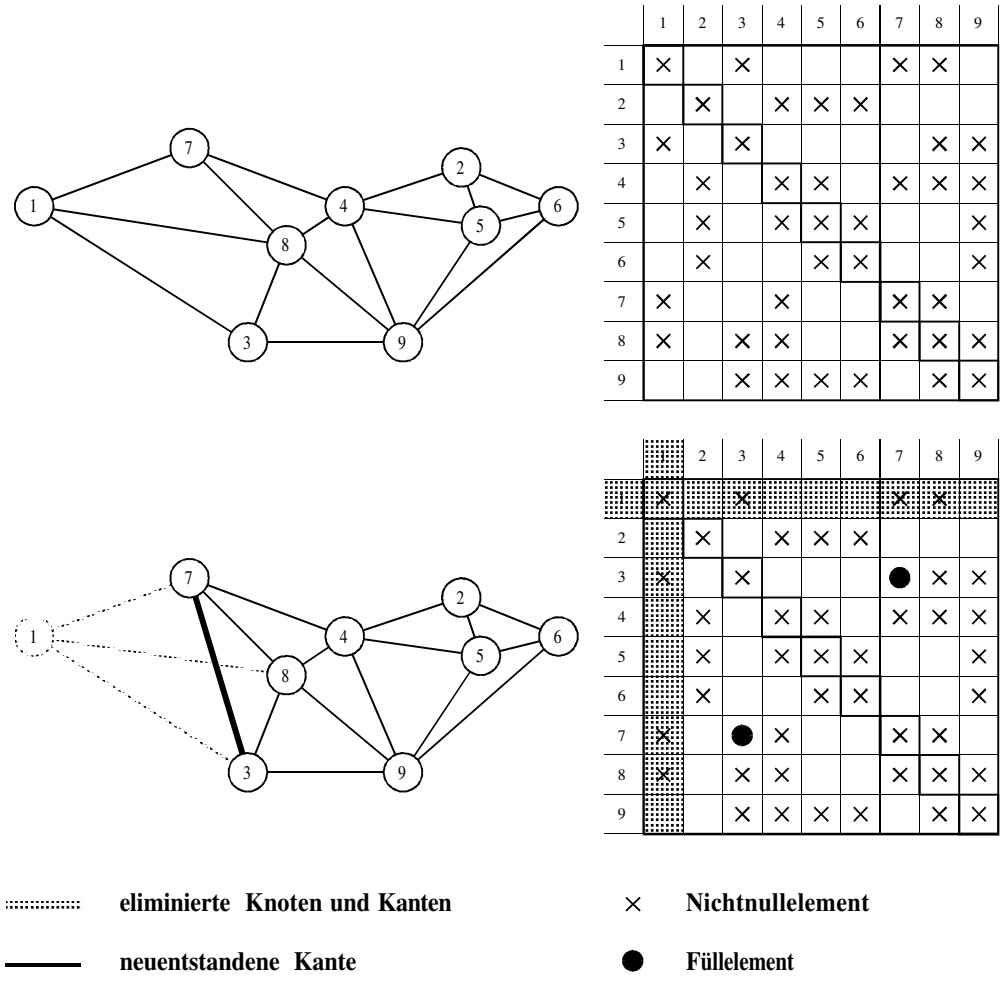


Abbildung 4.4: Auswirkung eines Elimintionsschrittes auf den Graphen und die Verknüpfungsmatrix

tes demonstriert werden. Ruft man sich die in im Abschnitt 1.1 erarbeiteten Schritte in Erinnerung, so kann folgendes über die Entstehung eines Füllelementes ausgesagt werden.

> In den Schritten 1, 2 und 4 entstehen keine neuen Elemente, da nur bestehende Elemente dividiert werden.
> Schritt 3 ist veranwortlich für die Entstehung von Füllelementen. Ist ein Nullelement in der i-ten Zeile und j-ten Spalte vorhanden, so ist das i-te Element der Pivotspalte und das j-te Element der Pivotzeile entscheidend, ob das Element weiterhin null bleibt. Sind beide Elemente ungleich null, so entsteht ein neues Füllelement.

Man versucht die Knoten in einer solchen Reihenfolge zu eliminieren, daß möglichst wenige neue Kanten entstehen. Die Reihenfolge der Elimination wird durch Umordnung der Knoten, was einer Permutation entspricht, gesteuert.

Neben der Anzahl der Füllelemente beeinflußt auch der Platz der Entstehung von neuen Kanten die Effizienz der Elimination. Aus speichertechnischen Gründen bevorzugt man Band- oder Profilstrukturen, auch variable Bandstrukturen, hüllenorientierte oder

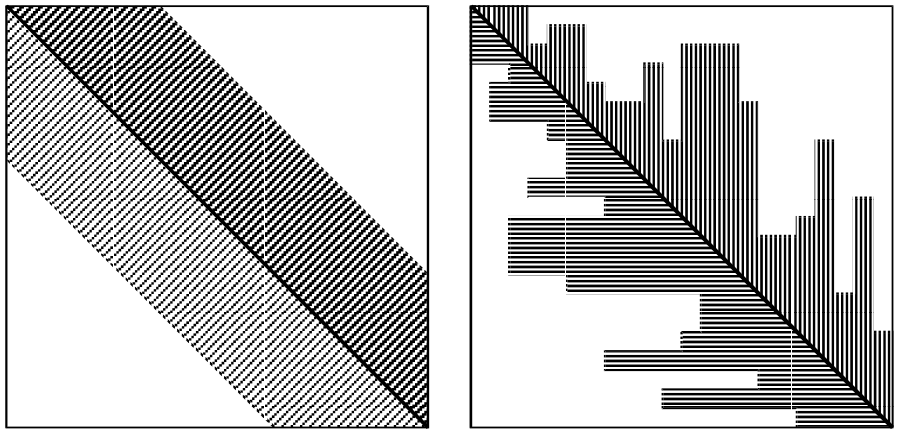


Abbildung 4.5: Bandstrukuren und hüllenorientierte Strukturen

skyline Strukturen genannt.

4.2 Regelmäßige, mehrdimensionale Netzstrukturen

Regelmäßige
Netze treten in den verschiedensten Formen auf. Bei der zwei(drei)dimensionalen Anwendung von finiten Elementen stößt man auf Netze, die durch Dreiecke (Tetraeder) oder Vierecke (Würfel) aufgebaut sind. Unregelmäßigkeiten entstehen einerseits durch unregelmäßige Randflächen und andererseits durch exaktere Modellierung bestimmter Gebiete. Zweidimensionale Approximationsaufgaben unter Verwendung von örtlich beschränkten Grundfunktionen und der Vorgabe eines streng oder nahezu regelmäßigen Stützpunktgitters erzeugen ähnliche Strukturen.

Die Aufgabe dieses Kapitels ist es, die Verknüpfungsmatrizen darzu stellen und Auswirkungen der unterschiedlichsten Numerierungsmethoden aufzuzeigen. Prinzipiell ist diese Anwendung nur für regelmäßige Netze mit "kleinen Störungen" gedacht. Diese kleinen Störungen beziehen sich auf die oben erwähnten Abweichungen vom Gitter, unregelmäßige Rand flächen und andere die exakte Regelmäßigkeit störenden Einflüsse. Für streng regelmäßige Strukturen existieren viel leistungsfähigere Ver fahren, die im Abschnitt 6 behandelt sind.

Manchmal empfehlenswert ist vorweg die Berechnung des ungestörten Netzes und die ausschließende Behandlung der Unregelmäßigkeiten. Diese kann man mit Hilfe der im Abschnitt 1.6 bzw. 1.7 aufgezeigten Verfahren bewerkstelligen. Eine konkrete Anwendung sind im Kapitel 6.1.4 erarbeitet und im Abschnitt 6.1.5 bei der Lösung von Band-Toeplitz Matrizen aufgezeigt. Die nachträgliche Behandlung von Unregelmäßigkeiten kann aber auch durch iterative Verfahren durchgeführt werden. Gute Näherungswerte und gute Informationen über das Spektrum können enorme Rechenvorteile bieten

(siehe Abschnitt 1.8 bzw. 2.5).

Um sich auf wesentliche Aspekte konzentrieren zu können, wird wegen der besseren Übersichtlichkeit und um den Umfang nicht zu sprengen, hier nur auf zweidimensionale Ausdehnungen eingegangen. Die Struktur der Verknüpfungsmatrix wird durch lokal beschränkte homogene isotrope Funktionen aufgebaut. Jeder Knoten übt demnach auf alle in seinem Umkreis liegenden Knoten eine Wirkung aus. Der Radius des Kreises, wo eine Wirkung besteht, wird als **Träger** bezeichnet. Als Grundstruktur für die Anordnung der Knoten wird ein quadratischer Raster vorausgesetzt. Die Erweiterung auf mehrdimensionale Ausdehnungen, bei anderen Grundmustern und nicht isotrope Funktionen sollte kein Problem dar stellen.

Der Aufbau der Verknüpfungsmatrix und das Verhalten während der Auf lösung soll an einem einfachen Beispiel demonstriert werden. Wir verwenden dafür ein regelmäßiges Gitter mit 49 auf einem quadratischen Raster angeordneten Knoten, wobei der Knotenabstand als eine Einheit festgelegt wird.

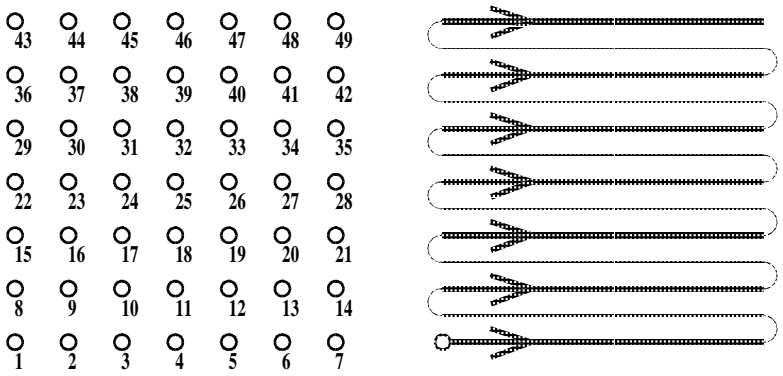


Abbildung 4.6: Knoten auf einem quadratischen Raster;
Numerierungsbeispiel 1, Zeilennumerierung

Jedem Knoten entspricht eine Zeile und Spalte in der Verknpüfungs matrix. Besteht eine direkte Wechselwirkung zwischen allen Knoten, so ist die Verknüpfungsmatrix ein voll besetztes System. Ist die Wirkung eines Knotens auf andere Knoten lokal beschränkt, so entsteht ein schwach besetztes System. Im Numerierungsbeispiel 1 (Abb. 4.6), wo ein Träger von 1.5 Einheiten angenommen ist, sind die Koeffizienten vom Knoten 25 zu den Knoten 17, 18, 19, 24, 26, 31, 32 und 33 ungleich Null.

Für dieses Beispiel ergibt sich ein Gleichungssystem, welches in Abb. 4.7 symbolisiert dargestellt ist. Alle Werte ungleich Null sind durch Kreuze gekennzeichnet. Bei der Reduktion des Systems entstehende Füllelemente sind im rechten Teil der Abb. 4.7 aufgezeigt.

Das Gleichungssystem mit 49 Unbekannten hätte bei voller Speicherung einen Platzbedarf von 2401 Elementen. Bei Beachtung der Symmetrie und Speicherung der oberen (oder unteren) Hälfte ergibt sich ein Platzbedarf von 1225 Elementen. Speichert man hingegen in komprimierter Form ab, so bieten sich zwei Formen an; erstens eine Abspeicherung in Bandform und zweitens eine hüllenorientierte Abspeicherung. Es

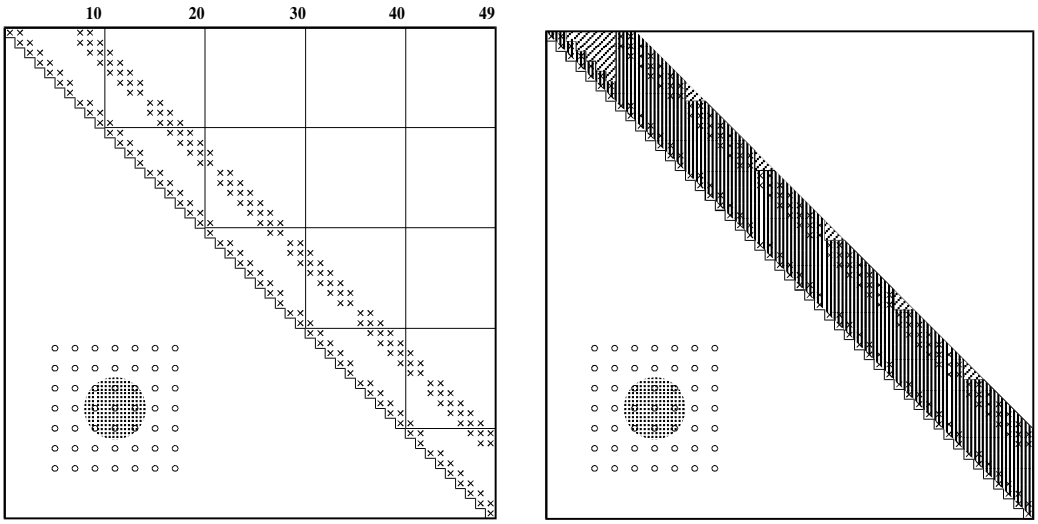


Abbildung 4.7: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 1, Träger 1.5
(links) vor der Reduktion (rechts) nach der Reduktion

ergeben sich 405 bzw. 385 abzuspeichernde und auch zu berechnende Elemente. Der Rechen aufwand bei voller Abspeicherung steigt mit der dritten Potenz der Anzahl der Unbekannten. Bei einer bandorientierten (hüllenorientierten) Speicherungsform ist der Rechenaufwand proportional der Anzahl der Unbekannten und steigt quadratisch mit der (mittleren) Bandbreite (siehe *SCHUH (1981)[72], S. 6*).

Vergrößert man den Träger der Grundfunktion, so ändert sich die Band breite der Verknüpfungsmatrix. Für die Träger von 2.5 und 4.5 zeigt die Figur 4.8 die Struktur auf.

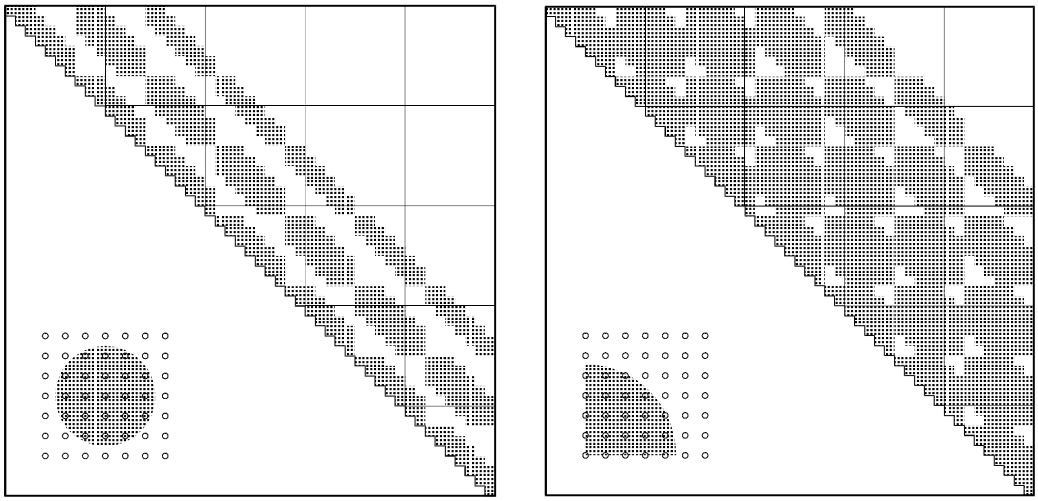


Abbildung 4.8: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 1,
(links) Träger 2.5 (rechts) Träger 4.5

Wie schon oben erwähnt, ist die Struktur der Verknüpfungsmatrix wesentlich von der Anordnung der Unbekannten abhängig. Die Abb. 4.10 bis 4.11 zeigen für die gleichen Beispiele wie oben die Strukturen der Verknüpfungsmatrizen.

Mit Ausnahme des Trägers 1.5 erweisen sich die beiden Numerierungsarten als gleichwertig. Die Verwendung von Bandalgorithmen ist bei einem Träger von 1.5 die erste Art der Numerierung zu bevorzugen.

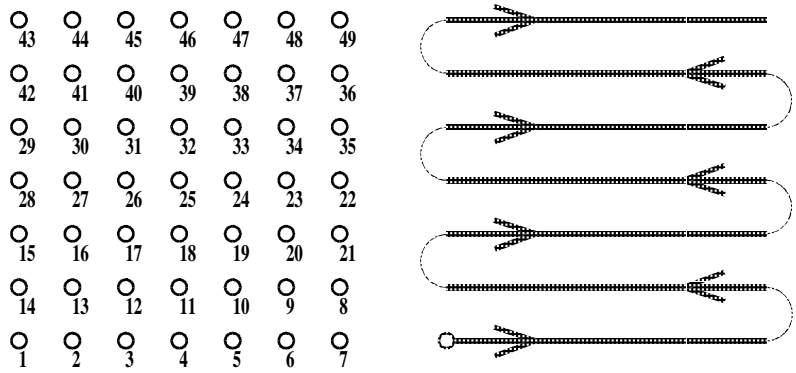


Abbildung 4.9: Knoten auf einem quadratischen Raster;
Numerierungsbeispiel 1, Meandernumerierung

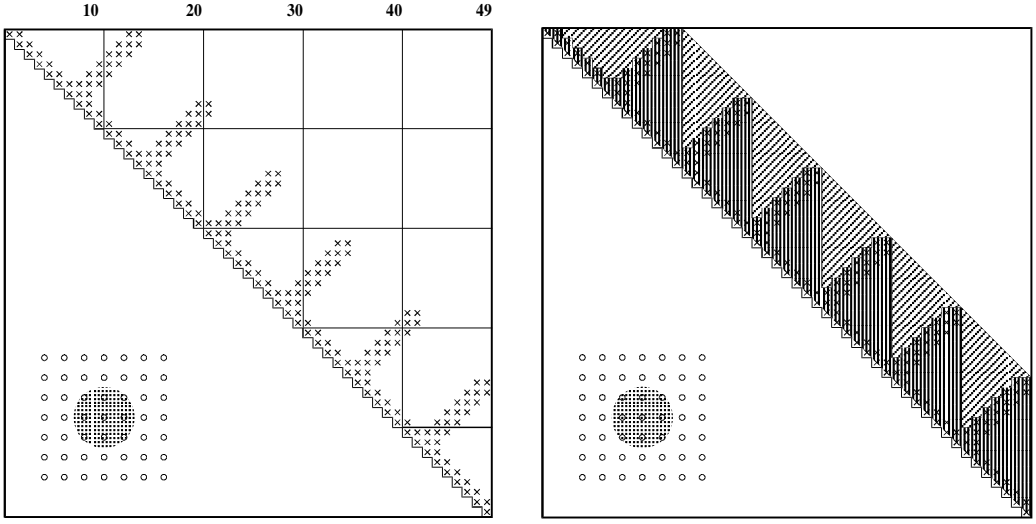


Abbildung 4.10: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 2, Träger 1.5
(links) vor der Reduktion (rechts) nach der Reduktion

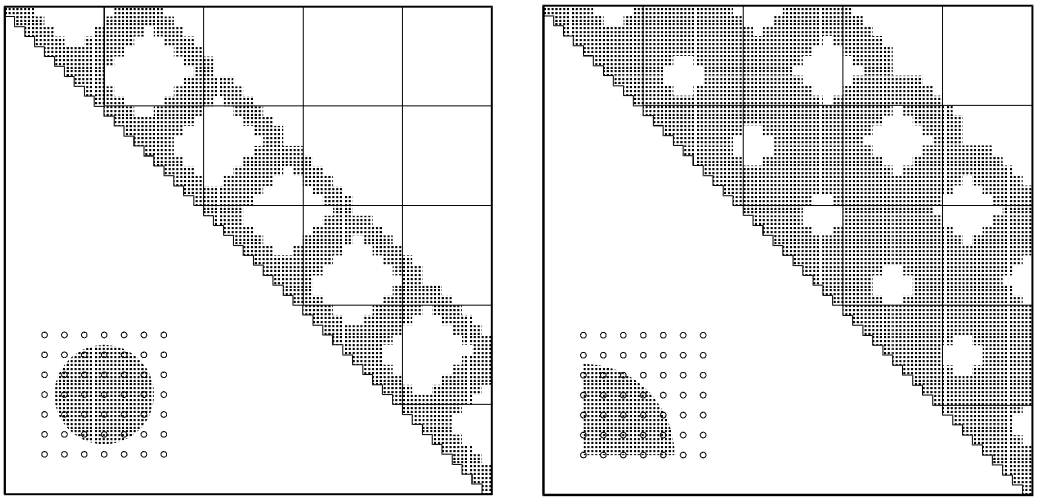


Abbildung 4.11: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 2,
(links) Träger 2.5 (rechts) Träger 4.5

Zur Vervollständigung der Möglichkeiten einer systematischen Numerierung werden zwei weitere Varianten ausgeführt, die in Diagonalenrichtung orientiert sind. Zum einen wird die Numerierung pro Diagonale von rechts unten nach links oben untersucht (Abb. 4.12). Die zweite Variante beruht auf einer diagonalorientierten Meandernumerierung (Abb. 4.14). Die Verknüpfungsmatrizen für einen Träger von 1.5 (Abb. 4.13 und Abb. 4.15) weisen zunächst keine Vorteile dieser Numerierungsart auf.

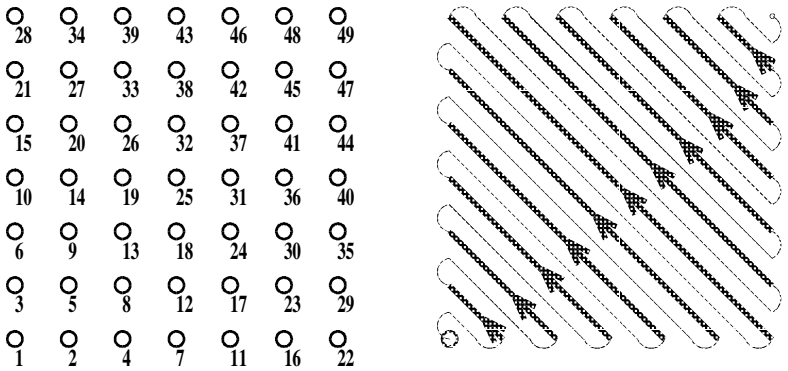


Abbildung 4.12: Knoten auf einem quadratischen Raster;
Numerierungsbeispiel 3, diagonalorientierte Zeilennumerierung

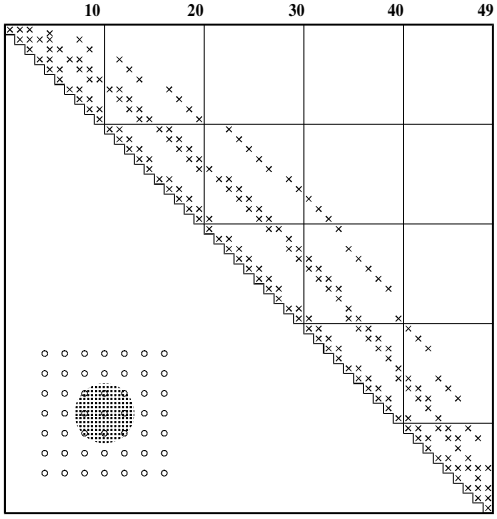


Abbildung 4.13: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 3,
Träger 1.5, vor der Reduktion

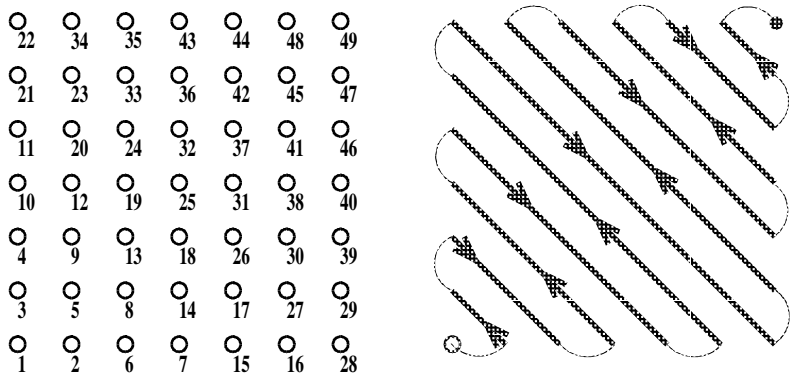


Abbildung 4.14: Knoten auf einem quadratischen Raster; Numerierungsbeispiel 4, diagonalorientierte Meandernumerierung

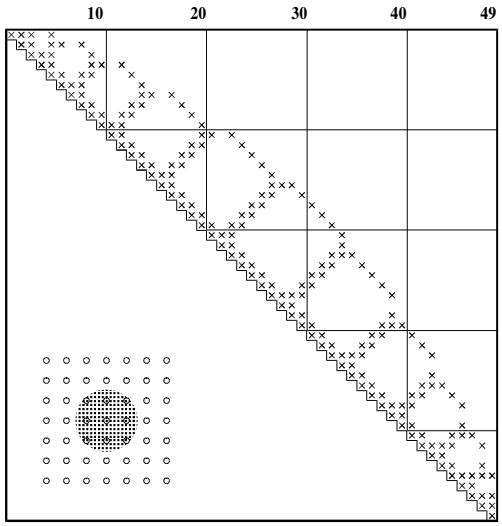


Abbildung 4.15: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 4, Träger 1.5, vor der Reduktion

Verringert man die Länge des Trägers, sodaß die Verknüpfung zwischen diagonal liegenden Knoten verschwindet (nur quadratisches Netz), so ergeben sich durch diese Numerierungsart Vorteile, die aus Abb. 4.16 unmittelbar ersichtlich sind. Während bei der Numerierungsart 1 die Bandbreite konstant gleich acht bleibt, variiert sie bei Numerierungsart 3 in einem Bereich zwischen drei und acht.

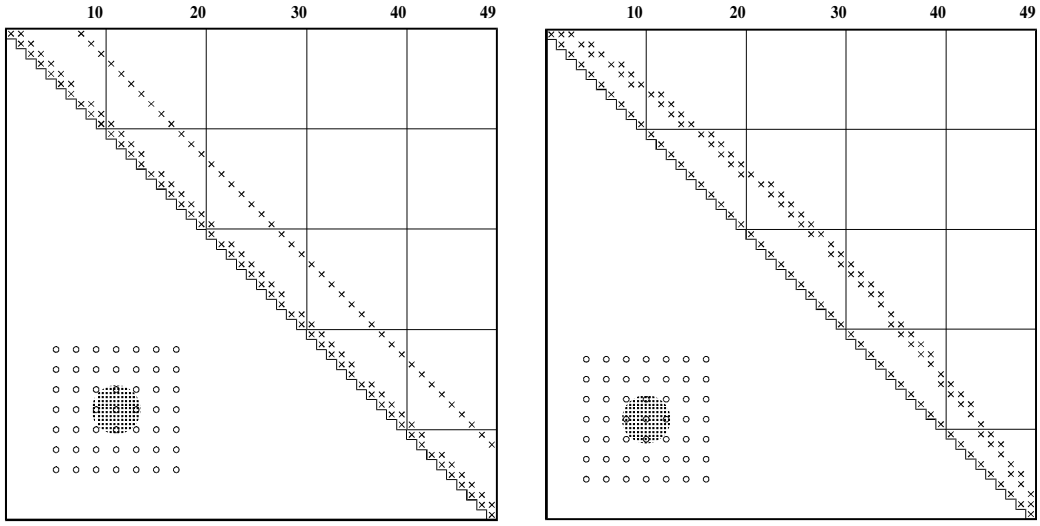


Abbildung 4.16: Struktur der Verknüpfungsmatrix; Träger 1.3,
(links) Numerierung 1 (rechts) Numerierung 3

Eine andere Numerierungsart, verbunden mit dem Begriff **Nested Dissection** wird in Abb. 4.17 dargestellt.

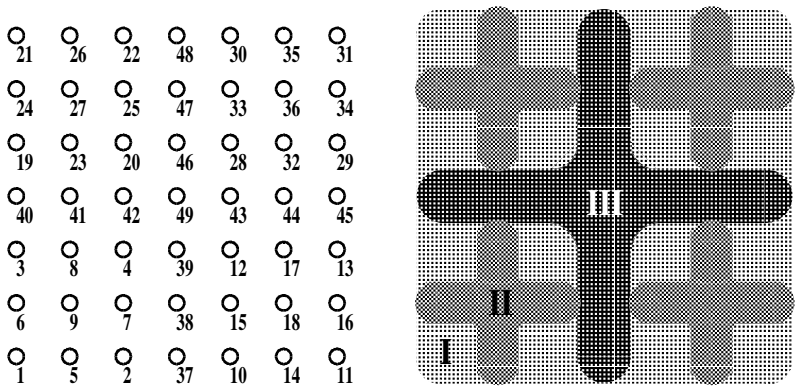


Abbildung 4.17: Knoten auf einem quadratischen Raster;
Numerierungsbeispiel 5, Nested Dissection

Die Einteilung der Bereich ist dabei so vorzunehmen, daß jeder Knoten nur einen Einfluß auf Knoten im eigenen Bereich und im unmittelbar benachbarten Bereich aufweist. Für die in Abb. 4.17 aufgezeigte Einteilung kann der Träger so groß sein, daß z.B. zwischen den Knoten 2 und 10 keine Verknüpfung besteht. Ein Träger mit einer Länge kleiner als zwei Einheiten ist somit sinnvoll.

Die Verknüpfungsmatrix weist bei dieser Numerierungsart zunächst eine scheinbar ungünstige Struktur auf (Abb. 4.18)

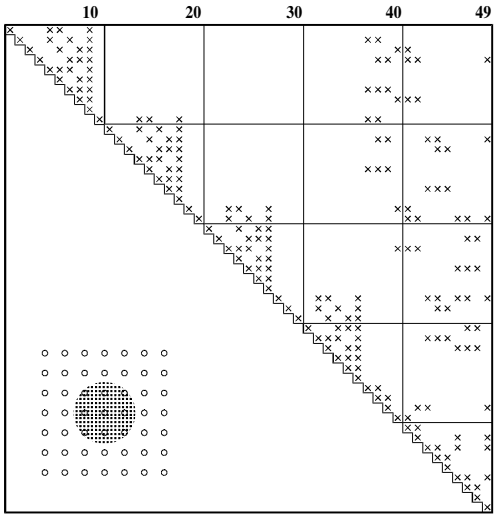


Abbildung 4.18: Struktur der Verknüpfungsmatrix;
Numerierungsbeispiel 5, Träger 1.5, vor der Reduktion

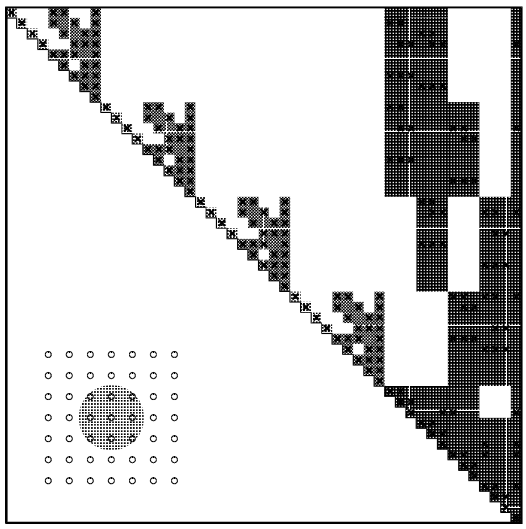


Abbildung 4.19: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 5, Träger 1.5,
Darstellung der wiederkehrenden (rekursiven) Blockstrukturen

Betrachtet man die Struktur nicht im Detail sondern in großen Blöcken, so erkennt man eine immer wiederkehrende Blockstruktur. Sie besteht aus vier an der Diagonalen angeordneten voneinander unabhängigen Blöcken, aus anschließenden vier leicht besetzten Spalten und einer voll besetzten Spalte (Abb. 4.19)

Bei der Auflösung des Systems unter Verwendung der Blöcke als Bearbeitungseinheit ist mit einem Füllelement (*'Füllblock'*) zu rechnen. Analysiert man einen der vier an der Diagonale angeordneten Block, so erkennt man, daß er die gleiche Struktur im kleinen widerspiegelt. Die Verkleinerung der Dimension ist abhängig von der Gebietsgröße. Betrachtet man einen quadratischen Bereich von insgesamt n rasterartig angeordneten Knoten, so ist für die Verkleinerung die halbe Ausdehnung $\frac{1}{2}\sqrt{n}$ als Rechenfaktor anzusetzen, wenn die Länge des Trägers im Vergleich zur Gebietsgröße als vernachlässigbar angenommen wird. Die ersten vier Blöcke und damit die ersten vier Zeilen repräsentieren jeweils ein Viertel des ungeteilten Gebietes und weisen somit eine Ausdehnung von $\frac{1}{2}\sqrt{n}$ im Quadrat auf und beinhalten somit $\frac{1}{4}n$ Knoten. Von Interesse ist die Dimension der acht rechteckigen Blöcke, die sich mit $\frac{1}{4}n \times \frac{1}{2}\sqrt{n}$ errechnet. Diese Dimension stellt die höchste Ordnung bei allen Speicheranforderungen dar, da zu beachten ist, daß die ersten vier Blöcke an der Diagonalen rekursiv aus der nächstkleineren Struktur berechenbar sind. Das asymptotische Verhalten für eine große Unbekanntenanzahl ist demnach proportional der Ordnung $n\sqrt{n}$ und zeigt somit das gleiche Wachstum wie die Numerierungsart 1. Auf Grund der höheren Koeffizienten der Elemente niederer Ordnungen erweist sich die Numerierungsart 5 bei blockorientierter Anwendung als speicherplatzintensiver. Für das angegebene Beispiel ergeben sich bei blockorientierter Auflösung 249 Füllelemente, was über der Anzahl der Füllelemente bei hüllenorientierter Speicherung und der Verwendung einer Bandform liegt (180 bzw. 200 Füllelemente). Vorteile kann die blockorientierte Berechnung nur in Zusammenhang mit gleichzeitig ablaufenden Rechenoperationen bringen. Können die Blöcke mit Hilfe von Vektorprozessoren (gleichzeitige Bearbeitung von mehreren Komponenten eines Vektors) verarbeitet werden, so erweist sich diese Numerierungsart als vorteilhaft.

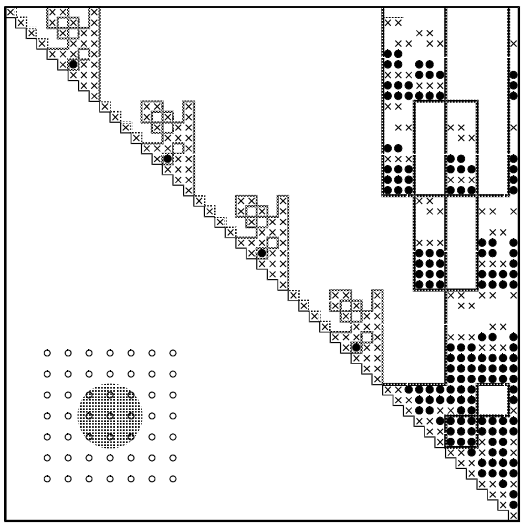


Abbildung 4.20: Struktur der Verknüpfungsmatrix; Numerierungsbeispiel 5, Träger 1.5, Füllelemente und Füllblöcke nach der Reduktion

Geht man bei der Numerierungsart 5 weiter ins Detail und betrachtet nicht die Blöcke als Bearbeitungseinheit, so erkennt man, daß nach der Auflösung auch innerhalb der Blöcke viele Nullelemente erhalten bleiben.

Nützt man all diese Plätze, so verursacht die Reduktion nur 156 Füllelemente. Dies liegt unter der Anzahl aller zuvor besprochenen Verfahren. Allerdings kann der Platz, wo ein Füllelement auftritt, erst während der Berechnung ermittelt werden, was eine dynamische Speicherung erfordert (siehe Kap. 4.1). Eine andere Möglichkeit besteht, indem man eine symbolische Auflösung zur Ermittlung der Füllelemente durchführt und so die Speicherzuordnung festlegt. Bei dieser Numerierungs- und Berechnungsart ist der Speicheraufwand für einen quadratischen Raster mit n Knoten proportional der Funktion $n \log_2 \sqrt{n}$ (*GEORGE (1973)[31]*). Diese Methode erweist sich somit für große Daten mengen als die am besten geeignete. Wesentlich neue Aspekte ergeben sich bei dieser Numerierungsart durch die Möglichkeit, einzelne Blöcke unabhängig voneinander zu verarbeiten. Jeder der vier an der Diagonale angeordneten Blöcke kann unabhängig von den anderen und somit parallel nebeneinander berechnet werden. Diesen für sehr große Systeme ausnutzbaren Vorteil der parallelen Verarbeitung ist Abschnitt 5.2 gewidmet.

Literatur:

[31] GEORGE Alan (1973): Nested Dissection of a Regular Finite Element Mesh. SIAM J. Numer. Anal., Vol. 10, No. 2, pp. 345-363.

[72] SCHUH Wolf-Dieter (1981): Programmierung rationeller Algorithmen zur Umordnung, Auflösung und Inversion der Normalgleichungen geodätischer Netze. Diplomarbeit, TU Graz.

4.3 Spektrale Eigenschaften von schwach besetzten Matrizen

Wesentliche Teile des Kapitels 2.1 weisen auf die Bedeutung der Eigenwerte und Eigenvektoren bei der numerischen Berechnung von Gleichungssystemen hin. Dieser Abschnitt soll eine bessere Vorstellung des Verhaltens der Eigenwerte und Eigenvektoren bei aus Netzen resultierenden schwach besetzten Matrizen ermöglichen. Ausgehend von ein fachen Strukturen bei eindimensionalen Ketten mit den unterschiedlichsten Randbedingungen wird das Verhalten der Eigenwerte studiert und besonderes Augenmerk auf die Eigenvektoren der kleineren Eigenwerte gelegt. Daß die gewonnenen Erkenntnisse auf mehrdimensionale Ausdehnungen übertragbar sind, zeigt die Behandlung von zweidimensionalen Beispielen.

Warum das Hauptaugenmerk auf die den kleineren Eigenwerten zugeord neten Eigenvektoren gerichtet ist, wird durch die Darstellung der Inversen als mit den reziproken Eigenwerten gewichtete Summe der dyadischen Produkte der Eigenvektoren mit sich selbst unmittelbar einsichtig (siehe Kap. 2.2.1),

$$\boldsymbol{N}^{-1} = \sum_{i=1}^{n} \frac{1}{\lambda_i}\, \boldsymbol{v}_i \boldsymbol{v}_i^T = \frac{1}{\lambda_1}\, \boldsymbol{v}_1 \boldsymbol{v}_1^T + \frac{1}{\lambda_2}\, \boldsymbol{v}_2 \boldsymbol{v}_2^T + \ldots + \frac{1}{\lambda_n}\, \boldsymbol{v}_n \boldsymbol{v}_n^T \tag{4.1}$$

Jene Eigenwerte und Eigenvektoren, die diese Summe hauptsächlich formen, werden im folgenden als **wesentliche** Eigenwerte bzw. Eigenvektoren bezeichnet. Der dem kleinsten (größten) Eigenwert zugeordnete normierte Eigenvektor wird vereinfacht als kleinster (größter) Eigenvektor angesprochen. Die gewählte Bezeichnungsart beruht auf Veröffentlichungen von *Rutherford (1947)[69] bzw. (1952)[70].* Zur Berechnung der Determinanten regelmäßiger eindimensionaler Systeme der Form

$$\boldsymbol{R} = \begin{bmatrix} x{+}b & 1 & & & & \\ 1 & x & 1 & & & \\ & 1 & x & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & x & 1 \\ & & & & 1 & x{+}a \end{bmatrix}_n \tag{4.2}$$

wird abkürzende Schreibweise $\boldsymbol{R}_n(x, a, b)$ eingeführt. Da für die folgenden Betrachtungen sowohl die Systeme mit positiven Nebendiagonalgliedern (Approximationsaufgaben) als auch mit negativen Nebendiagonalgliedern (Differenzengleichungen) von Bedeutung sind, wird die Bezeichnungsart erweitert auf

$$\boldsymbol{R}_n^{+}(x, a, b) = \begin{bmatrix} x{+}b & 1 & & & & \\ 1 & x & 1 & & & \\ & 1 & x & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & x & 1 \\ & & & & 1 & x{+}a \end{bmatrix}_n \tag{4.3}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

beziehungsweise

$$\boldsymbol{R}_n^-(x,a,b) = \begin{bmatrix} x{+}b & -1 & & & & \\ -1 & x & -1 & & & \\ & -1 & x & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & x & -1 \\ & & & & -1 & x{+}a \end{bmatrix}_n . \tag{4.4}$$

Durch die Wahl der Faktoren a und b können die unterschiedlichsten Randbedingungen modelliert werden. Beispiele für den Typ mit positiven Nebendiagonalgliedern sind alle Systeme, die durch kubische Spline Interpolation bei äquidistanten Stützstellen hervorgerufen werden. Der Typ $\boldsymbol{R}_n^+(4,0,0)$ entspricht der Vorgabe der Krümmung im Anfangs- und im Endknoten. Bei Vorgabe der Steigung im Anfangs- und Endknoten erhält man ein System vom Typ $\boldsymbol{R}_n^+(4,-2,-2)$. Die einseitige Festlegung von Krümmung und Steigung im Anfangs- oder Endknoten entspricht dem Typ $\boldsymbol{R}_n^+(4,0,-2)$ oder $\boldsymbol{R}_n^+(4,2,0)$. Weitere Beispiele für das Auftreten von Matrizen dieser Art mit positiven Nebendiagonalgliedern sind Ko varianzmatrizen. Bei regel mäßiger Anordnung der Knoten und sinnvoller Wahl der Halbwertsbreite einer homogenen Kovarianzfunktion entsteht im eindimensionalen Fall ein zum $\boldsymbol{R}_n^+(.,.,.)$-Typ ähnliches System.

Viele netzartige Strukturen erzeugen Differenzengleichungen und sind daher dem Typ mit negativen Nebendiagonalgliedern $\boldsymbol{R}_n^-(.,.,.)$ zuzuordnen. Typische Vertreter sind die aus geodätischen Netzen resultierenden Normalgleichungssysteme. Die einfachsten Systeme bilden die beidseitig eingespannte Kette (beidseitig abgeschlossener Zug, $\boldsymbol{R}_n^-(2,0,0)$), die einseitig eingespannte Kette (fliegender Zug, $\boldsymbol{R}_n^-(2,0,-1)$ oder $\boldsymbol{R}_n^-(2,-1,0)$) und die freie Kette (frei schwimmender Zug, $\boldsymbol{R}_n^-(2,-1,-1)$).

Die Eigenwerte und Eigenvektoren von regelmäßigen Strukturen können sehr oft in geschlossener analytischer Form angegeben werden. In der geodätischen Literatur sind vor allem bei theoretischen Untersuchungen über Fehlerausbreitungen in regelmäßigen Netzen geschlossene Formeln für Eigenwerte und Eigenvektoren zu finden (*MEISSL (1982)[56]*, *SÜNKEL (1985)[88]*, *WIESER (1988)[97]*). Die nachfolgend zusammengestellten Formeln beruhen auf den Veröffentlichungen von *RUTHERFORD (1947)[69]* bzw *(1952)[70]* und sind ergänzt durch unpublizierte Arbeiten von *BORRE (1979)[10]*, und *MEISSL (1980)[54]*. Die Ergänzungen für den Typ $\boldsymbol{R}_n^+(.,.,.)$ sind auf Grund von empirischen Berechnungen vorgenommen.

$$\begin{array}{lll} \boldsymbol{R}_n^+(x,0,0) & \lambda_i = x - 2\cos\frac{i\pi}{n+1} & \boldsymbol{v}_i = \sqrt{\frac{2}{n+1}}\sin\frac{(n+1-i)j\pi}{n+1} \\ \boldsymbol{R}_n^-(x,0,0) & \lambda_i = x - 2\cos\frac{i\pi}{n+1} & \boldsymbol{v}_i = \sqrt{\frac{2}{n+1}}\sin\frac{ij\pi}{n+1} \\ & i = 1,\ldots,n & j = 1,\ldots,n,\ i = 1,\ldots,n \end{array} \tag{4.5}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

$$
\begin{array}{lll}
\boldsymbol{R}_n^+(x,0,-1) & \lambda_i = x - 2\cos\frac{2i\pi}{2n+1} & \boldsymbol{v}_i = \sqrt{\frac{4}{2n+1}}\sin\frac{2(n+1-i)j\pi}{2n+1} \\
\boldsymbol{R}_n^-(x,0,-1) & \lambda_i = x - 2\cos\frac{2i\pi}{2n+1} & \boldsymbol{v}_i = \sqrt{\frac{4}{2n+1}}\sin\frac{(2i-1)j\pi}{2n+1} \\
 & i = 1,\ldots,n & j = 1,\ldots,n,\ i = 1,\ldots,n
\end{array} \tag{4.6}
$$

$$
\begin{array}{lll}
\boldsymbol{R}_n^+(x,-1,-1) & \lambda_i = x - 2\cos\frac{(i-1)\pi}{n} & \begin{cases} \boldsymbol{v}_1 = \frac{1}{\sqrt{n}}(-1)^{(j-1)} \\ \boldsymbol{v}_i \\ \sqrt{\frac{2}{n}}\cos\left[i\text{–}1+2(n\text{–}i\text{+}1)j\right]\frac{\pi}{2n} \end{cases} = \\
\boldsymbol{R}_n^-(x,-1,-1) & \lambda_i = x - 2\cos\frac{(i-1)\pi}{n} & \begin{cases} \boldsymbol{v}_1 = \frac{1}{\sqrt{n}} \\ \boldsymbol{v}_i = \sqrt{\frac{2}{n}}\cos\left[i\text{–}1+2(i\text{–}1)j\right]\frac{\pi}{2n} \end{cases} \\
 & i = 1,\ldots,n & j = 1,\ldots,n,\ i = 2,\ldots,n
\end{array} \tag{4.7}
$$

Bemerkenswert für die numerische Behandlung sind folgende Eigenschaften, die unabhängig von den Randbedingungen auftreten und daher exemplarisch am Modell $\boldsymbol{R}_n(x,0,0)$ behandelt werden.

- Der Faktor der Diagonalelemente x geht linear in die Berechnung der Eigenwerte ein und hat keinen Einfluß auf die Eigenvektoren. Eine additive Veränderung der Diagonalelemente führt auf eine additive Änderung der Eigenwerte. Die Kondition der Matrix und somit die Stabilität der Lösung kann beeinflußt werden, ohne die durch die Eigenvektoren festgelegten Eigenschaften zu verändern.

- Durch Berechnung des Graphen für den typischen Ausdruck der Eigenwerte $x - 2\cos\alpha$,$0 \le \alpha \le \pi$ und deren Reziprokwerte $1/(x - 2\cos\alpha)$ ist es möglich, die Änderung der Eigenwerte mit wachsender Ausdehnung zu studieren (Abb. 4.21). Das Verhalten für alle Ketten, deren Faktor x die Bedingung $|x-2| > \epsilon$ für ein genügend großes ϵ erfüllt, weisen ein stabiles Verhalten auf. Dies wird durch die gleiche Größenordnung aller reziproken Eigenwerte dokumentiert. Für den häufig auftretenden Grenzfall $x = 2$ kann an Hand des Graphen erkannt werden, daß stark unterschiedliche Größenordnungen in den reziproken Eigenwerten auftreten. Bemerkenswert ist dabei, daß nur wenige Eigenwerte signifikanten Einfluß auf die Inverse besitzen (z.B. $n = 7$; der kleinste Eigenwert $\lambda_1 = 0.1522$ unterscheidet sich vom zweitkleinsten $\lambda_2 = 0.5858$ um einen Faktor von 4).

- Auf Grund der Signifikanz der kleineren Eigenwerte sind dessen Eigenvektoren verantwortlich für das Verhalten der Lösung. Nieder frequente Eigenvektoren, die große Wellenlängen besitzen, sorgen für großräumige Unsicherheiten in der Lösung. Weisen die wesentlichen Eigenvektoren eine sehr kurze Wellenlänge auf, so ist mit dem Auf treten von lokalen Unsicherheiten in der Lösung zu rechnen (geringe Nachbarschaftsgenauigkeit). Bemerkenswert ist es nun, daß für den Typ $\boldsymbol{R}^+(.,.,.)$ die wesentlichen Eigenvektoren hochfrequent sind, während beim Typ $\boldsymbol{R}^-(.,.,.)$ langwellige Eigenvektoren den signifi kanten Eigenwerten zugeordnet sind. Es ist somit ein konträres Verhalten bei iterativen Lösungsverfahren zu erwarten.

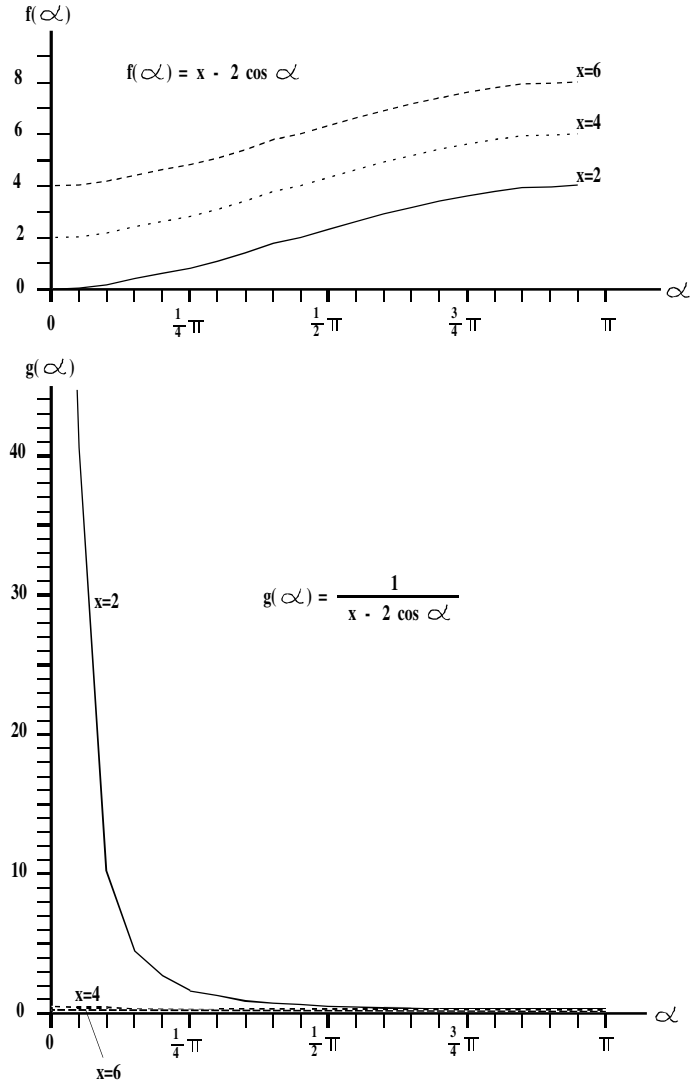


Abbildung 4.21: Graphen der Funktionen zur Berechnung der Eigenwerte

Daß diese Ergebnisse auch bei mehrdimensionalen Netzen gültig sind, kann mit Hilfe von empirischen Berechnungen nachgewiesen werden. Beispiele für geschlossene analytische Formeln sind z.B. bei *MEISSL (1970)[52]* für den frei schwimmenden 2-dimensionalen Typ $\boldsymbol{R}^{-}(.,.,.)$ angegeben.

Unbeantwortet blieb einstweilen die Frage, wie weit lassen sich die gewonnenen Erkenntnisse auf beliebige Anwendungen verallgemeinern?

Für den Fall der Differenzengleichungen kann das Beispiel einer einfach gelagerten, durch mehrere Zwischenglieder verstärkte Kette aufschluß über das Verhalten bei unregelmäßiger Anordnung geben. Abb. 4.22 veranschaulicht die umfangreichen Verstei-

fungen.

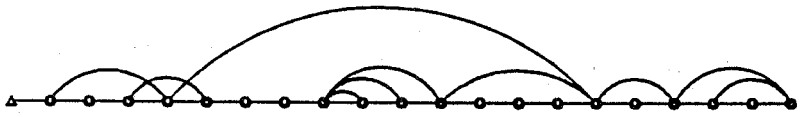


Abbildung 4.22: Stark verstärkter Nivellementzug.

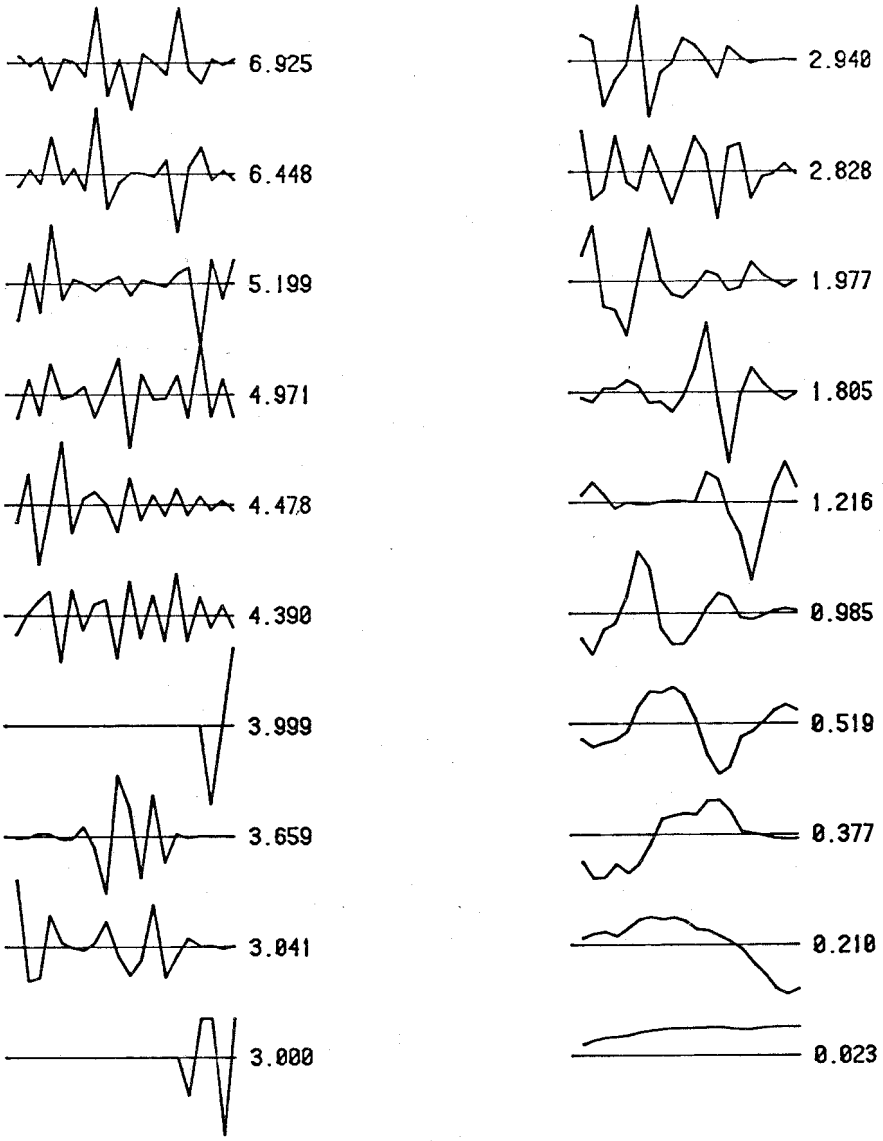


Abbildung 4.23: Eigenwerte und Eigenvektoren des stark verstärkten Nivellementzuges von Abb. 4.22.

Aus Abb. 4.23 sind die Eigenwerte und Eigenvektoren ersichtlich (Beispiel entnommen aus *SCHUH (1984)[73]*, Kap. 5.1.2). Man erkennt, daß die wesentlichen Eigenvektoren

nahezu unbeeinflußt von den Verstärkungen sind.

Bei finiten Netzen (Netze mit lokalen Verknüpfungen) sind immer langwellige Eigenvektoren vorhanden, die im wesentlichen nur durch die Randbedingungen beeinflußbar sind. Einige Studien über das Verhalten der Eigenwerte und Eigenvektoren in unterschiedlichsten geodätischen Netzen bei verschiedenen Randbedingungen sind bei *ORESCHNIK (1985)[60]* zu finden.

Eine der Mechanik entnommene Bestätigung dieses Verhaltens ist bei *DANKERT (1977)[20]* zu finden. Er schreibt auf Seite 48, 2. Absatz:

> Die Koeffizientenmatrizen der sehr großen Gleichungssysteme, die bei Anwendung der Diskretisierungsverfahren (Methode der finiten Elemente, Differenzenverfahren) entstehen, sind in der Regel sogenannte *Steifigkeitsmatrizen*, deren Elemente physikalisch als *Federsteifigkeiten* gedeutet werden können. Wenn an den Knotenpunkten dieses Federsystems geeignete Einzelmassen angebracht wer den, sind die Eigenwerte der Steifigkeitsmatrix proportional zu den Eigenfrequenzen eines solchen Schwingungssystems. Damit bestätigt sich die Aussage, daß (von wenigen Ausnahme abgesehen) sich der kleinste Eigenwert deutlich von den übrigen abhebt.
>
> Anderersetis wird klar, daß die größten Eigenwerte (analog zu den hohen Frequenzen des Schwingungsmodells) relativ dicht liegen,...

Die Verallgemeinerung des Typs $\boldsymbol{R}^{+}(.,.,.)$, der bei Approximationsaufgaben vorherrscht, führt auf die Frage des Zusammenhanges zwischen verwendeten Grundfunktionen und den Eigenwerten beziehungsweise Eigenvektoren. Am besten kann dieser Zusammenhang an von Randerscheinungen weitgehend unbeeinflußten Beispielen demonstriert werden. Dies könnte durch die Wahl einer großen Anzahl von Knoten geschehen. Eine andere Möglichkeit stellt die Berechnung von zyklischen Systemen dar. Durch das Verknüpfen des Anfangs- und Endknotens werden bei Approximationsaufgaben spezielle Typen von Matrizen erzeugt, die man als zirkulierende Matrizen bezeichnet (siehe Abs. 6).

$$\mathbf{C} = \begin{bmatrix}
c_0 & c_1 & c_2 & \dots & \dots & c_h & \dots & \dots & \dots & c_2 & c_1 \\
c_1 & c_0 & c_1 & \dots & \dots & \dots & c_h & \dots & \dots & c_3 & c_2 \\
c_2 & c_1 & c_0 & \dots & & & & c_h & & c_4 & c_3 \\
\dots & \dots & \dots & \dots & \dots & & & & \dots & \dots & \dots \\
\dots & \dots & & \dots & \dots & \dots & & & & c_h & \dots \\
c_h & \dots & & & \dots & \dots & \dots & & & \dots & c_h \\
\dots & c_h & & & & \dots & \dots & \dots & & \dots & \dots \\
\dots & \dots & c_h & & & & \dots & \dots & \dots & \dots & \dots \\
\dots & \dots & & \dots & & & & \dots & \dots & \dots & \dots \\
c_2 & c_3 & c_4 & \dots & c_h & \dots & \dots & \dots & \dots & c_0 & c_1 \\
c_1 & c_2 & c_3 & \dots & \dots & c_h & \dots & \dots & \dots & c_1 & c_0
\end{bmatrix} \tag{4.8}$$

Wie im Abschnitt 6.1.3 ausgeführt ist, können die Eigenvektoren von zirkulierenden

Matrizen direkt angegeben werden,

$$\mathbf{f}_j = \begin{bmatrix} 1 \\ \varepsilon_j \\ \varepsilon_j^2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_j^{n-1} \end{bmatrix} \qquad j = 0, 1, \ \ldots, n-1 \tag{4.9}$$

wobei abweichend von der bisherigen Darstellung die Numerierung von 0 bis $n-1$ üblich ist und daher übernommen wird. Die Eigenwerte ergeben sich somit aus

$$\lambda_k = \sum_{t=0}^{n-1} c_t \; e^{-itk\frac{2\pi}{n}} \tag{4.10}$$

beziehungsweise

$$\lambda_k = \sum_{t=0}^{n-1} c_t \; \cos\left(tk \; \frac{2\pi}{n}\right) \; . \tag{4.11}$$

bei Verwendung symmetrischer Grundfunktionen. Die Glieder der Inversen $c^{(-1)}_{|p-q|}$ können durch

$$c^{(-1)}_{|p-q|} = \frac{1}{n} \sum_{k=0}^{n-1} \lambda_k \; \cos\left(k|p-q| \; \frac{2\pi}{n}\right) \; . \tag{4.12}$$

ermittelt werden. Durch Ausnützung von Additionstheoremen und Symmetrie eigenschaften für Winkelfunktionen können die Berechnungen sehr rasch durchgeführt werden (Diskrete Fast-Fourier-Transformation, siehe Abs. 6.1.3.2). Für das Weiter von Interesse ist einstweilen nur die Zuordnung der Eigenwerte λ_k zu den Frequenzen der Winkelfunktion. In der Regel sind die Eigenwerte mit kleineren Zahlenwerten den in diesen Fällen den hohen Frequenzen (kleine Wellenlängen) zugeordnet, was aus folgenden Erläuterungen einsichtig wird.

Steigert man die Zahl der Realisierungen n gegen unendlich, so kann der Übergang von der Summe zum Integral durchgeführt werden. Die diskreten Realisierungen der Grundfunktion und die diskreten Eigenwerte gehen über in kontinuierliche Funktionen

$$\lambda(k) = \int_{-\infty}^{\infty} c(t) e^{-i2\pi tk} dt = C(T) \tag{4.13}$$

wobei $\lambda(k)$ die fouriertransformierte Funktion (Spektrum) der Funktion $c(t)$ darstellt, welches in einheitlicher Nomenklatur im weiteren durch Großbuchstaben - also $C(T)$ - bezeichnet wird,

$$\lambda(k) = \mathcal{F}\{c(t)\} = C(T). \tag{4.14}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

Da von vielen Funktionen die Fouriertransformierte in geschlossener, analytischer Form bekannt ist, wäre es interessant den Zusammenhang zwischen dem kontinuierlichen Spektrum der Funktion und den diskreten Eigenwerten zu kennen. Da ein analytischer Übergang im Allgemeinen nicht möglich ist, versuchen wir hier exemplarisch bei einer Grundfunktionen mit Hilfe von numerischen Berechnungen einen Übergang zu schaffen, um bestimmte Eigenheiten zu erkennen. Verwendet man als Grundfunktionen kubische Splines, so kann folgende Näherungsformeln zur Berechnung der Eigenwerte empirisch ableitet werden.

$$\lambda_k = \frac{t}{2}\left(\frac{\sin kt\frac{\pi}{2n}}{kt\frac{\pi}{2n}}\right)^4 e^{c_1\left|\frac{k}{n}\right|^{c_2}} \tag{4.15}$$

$n \ldots gerade \qquad k = 0, \pm 1, \ldots, \pm(\frac{n}{2} - 1), +\frac{n}{2}$
$n \ldots ungerade \qquad k = 0, \pm 1, \ldots, \pm\frac{n}{2}$

$c_1 = const. \sim 40.427$
$c_2 = const. \sim 5.385$
$t \ldots$ Träger

Bei der Analyse dieser Formel erkennt man, daß sie aus zwei Teilen zusammengesetzt ist. Der Klammerausdruck bezieht sich auf die verwendete Grundfunktion und stellt das Spektrum der fouriertransformierten Funktion dar (*SÜNKEL (1984)[87]*, Seite 5). Der zweite, durch die Exponentialfunktion gebildete Anteil wird hervorgerufen durch den Übergang vom Kontinuierlichen zum Diskreten. Er stellt für diese Annahme stellt eine glatte, kontinuierlich wachsende Funktion dar und bewirkt somit eine Vergrößerung der Eigenwerte (Abb.: 4.24). Während das Spektrum für niedere Frequenzen k unverändert bleibt, tritt mit wachsendem k eine zunehmende Vergrößerung bis zum Zweifachen auf. Für die Approximation der Eigenwerte, wie sie hier bei einer endlichen Grundfunktion durchgeführt wird, stößt man bei nichtendlichen Grundfunktionen oder bei Toeplitz-Matrizen wegen schwer modellierbarer Randerscheinungen auf Schwierigkeiten.

Interessant für die nachfolgenden Berechnungen ist besonders der erste Anteil, der das Spektrum der Grundfunktion darstellt. Er vermittelt den Zusammenhang zwischen den Eigenschaften der Grundfunktion (spezielles Spektrum bei semidefiniten Funktionen, der Wahl des Trägers bzw. der Halbwertsbreite) und den resultierenden Eigenwerten. Man erkennt folgendes:

- Die beim Einsatz von monoton fallenden Grundfunktionen auftretenden Spektren stellen zumeist, aber nicht notwendiger Weise, abklingende Funktionen dar. Wenn kleine Eigenwerte auftreten, sind sie vor allem bei hohen Frequenzen zu erwarten.
- Vorsicht ist bei manchen Funktionen geboten, deren Spektren kritische Stellen aufweisen. Nullstellen im Spektrum bei positiv semidefinite Funktionen können durch spezielle Wahl der Anzahl der Datenpunkte umgangen werden (z.B. bei kubischen Splines). Analoge Hilfe kann bei Funktionen Verwendung finden, wo Werte des Spektrums gegen unendlich gehen (z.B. bei der Besselfunkion erster Art, nullter Ordnung).

Abbildung 4.24: Darstellung der Komponenten des Formelsystems 4.15.

- Eine Vergrößerung (Verkleinerung) des Trägers bewirkt eine Schrumpfung (Dehnung) des Spektrums.

Der mathematische Zusammenhang für die Schrumpfung und Dehnung des Spektrums bezeichnet man als *Time Scaling* bzw. *Frequency Scaling* (siehe z.B. *BRIGHAM (1974)[13]*, Seite 32-35). Wenn $C(T)$ die Fouriertransformierte von $c(t)$ darstellt, so bildet sich die Fouriertransformierte bei $c(mt)$, wobei m ein positive Konstante (Maßstab) darstellt, durch Substitution mit $mt = t'$ im Fourier Integral,

$$\int_{-\infty}^{\infty} c(mt)e^{-i2\pi tT}dt = \int_{-\infty}^{\infty} c(t')e^{-i2\pi t'\frac{T}{m}}\frac{dt'}{m} = \frac{1}{m}C\left\{\frac{T}{m}\right\} \ . \tag{4.16}$$

Die Umkehrung mit positiver Konstante M

$$\int_{-\infty}^{\infty} C(MT)e^{-i2\pi Tt}dT = \int_{-\infty}^{\infty} C(T')e^{-i2\pi T'\frac{t}{M}}\frac{dT'}{M} = \frac{1}{M}c\left\{\frac{t}{M}\right\} \ . \tag{4.17}$$

errechnet sich analog. Dieser Effekt wird auch bei der Spektralanalyse von diskreten Funtktionen verwendet. Durch eine größere Datendichte - entspricht einer Vergrößerung der Trägers - wird der gesuchte Teil des Spektrums in einem Bereich konzentriert, der nicht durch Diskretisierungseffekte (Alaising) gestört ist (siehe z.B. ??Blaha??).

Das spektrale Verhalten von Matrizen, die durch finite Grundfunktionen gebildet werden, kann folgendermaßen zusammengefaßt werden:

> Bei den meisten Anwendungen bestimmen einige wenige signifikante Eigenwerte das Verhalten des Systems. Bei Netzen, die auf Differenzenverfahren beruhen, sind die den signifikanten Eigenwerten zugeordneten Eigenvektoren durch langwellige Schwingungen geprägt. Bei Approximationsaufgaben treten kritische Situationen meist bei den hochfrequenten Schwingungen auf.

Literatur:

[10] BORRE Kai (1979): Covariance Matrices / Functions for 1-D Levelling Problems. Unpublished manuscript.

[13] BRIGHAM E.Oran (1974): The Fast Fourier Transform. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 32-35.

[20] DANKERT Jürgen (1977): Numerische Methoden der Mechanik. Springer-Verlag, Wien-New York.

[52] MEISSL Peter (1970): Über der Fehlerfortpflanzung in gewissen regelmäßigen flächig ausgebreiteten Nivellementnetzen. Zeitschrift für Vermessungswesen (ZfV), Konrad Wittwer, Stuttgart, Vol. 95, Heft 3, S. 103-109.

[54] MEISSL Peter (1980): Die Eigenwertmethode am Beispiel des endlichen eingehängten Nivellementzuges. Unveröffentlichtes Manusskript.

[56] MEISSL Peter (1982b): Fourier Analysis of Geodetic Networks. Unpublished manuscript.

[60] ORESCHNIK Kurt (1985): Analyse von zweidimensionalen Netzen mit Hilfe von Eigenwerten und Eigenvektoren. Diplomarbeit, TU Graz.

[69] RUTHERFORD D. E. (1947): Some Continuant Determinants Arising in Physics and Chemistry. Proc. Roy. Soc. Edin., A, LXII, pp. 229-236.

[70] RUTHERFORD D. E. (1952): Some Continuant Determinants Arising in Physics and Chemistry - II. Proc. Roy. Soc. Edin., A, LXIII, pp. 232-241.

[87] SÜNKEL Hans (1984): SPLINES: Their Equivalence to Collocation. Ohio State University (OSU), Reports of the Department of Geodetic Science, No. 357.

[88] SÜNKEL Hans (1985): Fourier Analysis of Geodetic Networks. Eigenvalues. GRAFAREND E.W., F. SANSO (Editors): 'Optimization and Design of Geodetic Networks', Springer, Heidelberg, pp. 257-300.

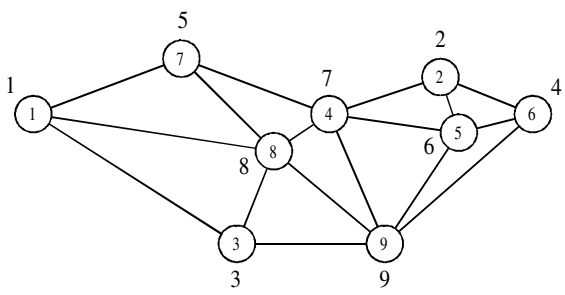
[97] WIESER Manfred (1988): Theoretische Untersuchungen spektraler Methoden zur Analyse regelmässiger Strukturen am Beispiel der Fouriertransformation geodätischer Netze. Mitteilungen der geodätischen Institute der TU Graz, Folge 60.

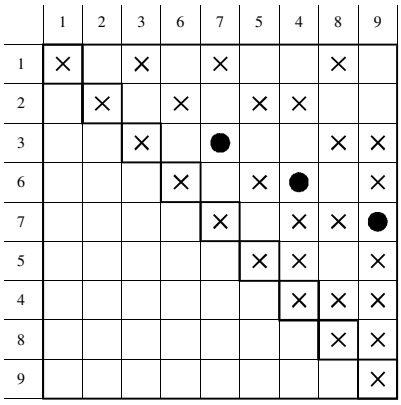
4.4 Umordnungsalgorithmen

Die Umordnung kann auf verschiedene Arten durchgeführt werden, wobei die Anzahl der Füllelemente und damit die Platzersparnis meist umgekehrt proportional dem Rechenaufwand der Umordnung ist. Mit dem Absinken der Anzahl der Rechenoperationen verringern sich auch die Rundungsfehler. Die Umordnung kann nur sinnvoll ausgenützt werden, wenn eine komprimiert und effiziente Abspeicherung erfolgt.

4.4.1 Umordnungsalgorithmen mit dynamischer Speicherung

Zunächst sei an Hand einer sehr einfachen Methode zur Umordnung von symmetrischen Gleichungssystemen (*SCHEK et al. (1977)[71]*) die Problematik der dynamischen Speicherung dargestellt. Die Knoten werden in einer solchen Reihenfolge sortiert, daß jene Knoten, die einen niedrigen Grad aufweisen, zu Beginn stehen und jene Knoten mit einem hohem Grad d.h. mit vielen Kanten an das Ende gereiht werden. Wie man sich leicht überlegen kann, ist dadurch die Wahrscheinlichkeit sehr groß, daß erst in der letzten Phase der Reduktion Füllelemente auftreten. Abbildung 4.25 zeigt die neue Reihenfolge der Knoten und gibt das reduzierte Gleichungssystem mit den Füllelementen wieder.





Knotennamen	1	2	3	4	5	6	7	8	9
Grad	3	3	3	5	4	3	3	5	5
Platznummer	1	2	3	7	6	4	5	8	9

× Nichtnullelement
● Füllelement

Abbildung 4.25: Ergebnis eines einfachen Umordnungsverfahrens, minimaler Grad Algorithmus.

Der Vorteil dieses Verfahrens liegt darin, daß relativ wenige Operationen für die Umordnung des Normalgleichungssystems erforderlich sind. Die Umordnung ist aber nur Suboptimal und hat den großen Nachteil, daß die Nichtnullelemente beziehungsweise die Füllelemente nicht geordnet auftreten. Es ist eine elementweise, dynamische Abspeicherung des Systems notwendig, wofür unter anderem folgende vier Arten in Frage kommen (siehe auch Abb. 4.26):

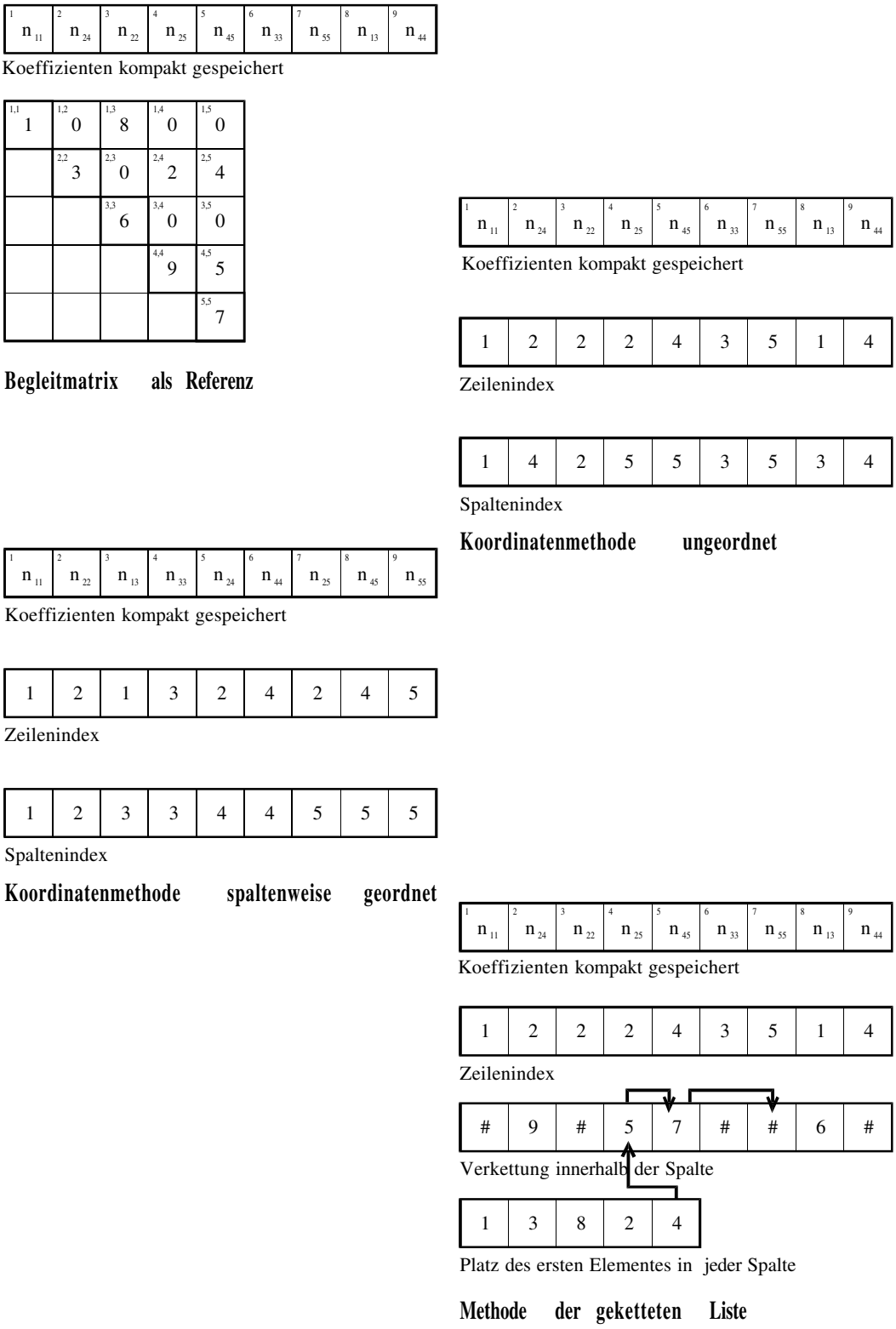


Abbildung 4.26: Dynamische Speichertechniken

- **Begleitmatrix als Referenz:**
  Die von Null verschiedenen Elemente der Matrix werden in beliebiger Reihenfolge bei direktem Zugriff gespeichert. Um den Platz jedes dieser Elemente bestimmen zu können, wird zusätzlich eine Begleitmatrix angelegt, die die Verbindung zwischen Zeile, Spalte und Speicherplatz herstellt.

- **Koordinatenmethode ungeordnet:**
  Bei der Koordinatenmethode wird außer dem Wert eines Nichtnullelementes noch dessen Zeilen- und Spaltennummer gespeichert.

- **Koordinatenmethode geordnet:**
  Oft ist es vorteilhaft, die Elemente zeilen- oder spaltenweise zu ordnen und die Adressen des ersten Elementes einer Zeile oder Spalte mitzuführen.

- **Methode der geketteten Liste:**
  Bei der Methode der einfach geketteten Liste werden außer dem Wert eines Nichtnullelementes noch bei zeilenweiser Speicherung, die Spaltennummer und die Adresse des nächsten Elements der Zeile angegeben. Die Elemente müssen nicht in einer bestimmten Reihen folge gespeichert werden. Die Adresse des ersten Elementes einer jeden Zeile werden in einem zusätzlichen Vektor gespeichert.

Die Methode 1 ermöglicht eine nicht sehr komprimierte jedoch einfache Abspeicherung, ist für einen zeilen- und spaltenweisen Zugriff geeignet aber nicht sehr effizient. Der Zugriff auf bestimmte Elemente ist direkt möglich.

Für sehr schwach besetzte Matrizen stellt die Koordinatenmethode eine einfache und effiziente Methode dar. Sobald die Struktur (Nichtnullelemente und Füllelemente) feststeht, bildet die geordnete Form eine gute Basis für den Reduktionsprozeß.

Die Methode der einfach geketteten Liste ist vor allem bei dynamischen Systemen mit spalten- oder zeilenweisem Zugriff besonders günstig. Werden beide Zugriffsarten benötigt, kann eine zweite Liste mitgeführt werden. Der Zugriff auf einzelne Elemente ist nur index-sequentiell möglich.

Der Reduktionsprozeß wird in zwei Stufen durchgeführt. Durch eine symbolische Berechnung wird zunächst die Struktur festgelegt, wobei sowohl der Platz der Nichtnullelemente als auch der Füllelemente bestimmt wird. Anschließend kann auf der festgelegten Struktur die numerische Berechnung erfolgen.

Die hier im besonderen zu betrachtende symbolische Berechnung kann mit Hilfe der Verknüpfungsmatrix oder der Begleitmatrix sehr rasch durch logische Vergleiche vollzogen werden, wobei die Ordnung der Operationen mit der dritten Potenz der Anzahl der Variablen steigt. Eine effizientere Methode beruht auf folgenden Überlegungen. Wenn ein Knoten eliminiert wird, entsteht ein neuer Graph, bei dem alle Nachbarknoten des eliminierten Knotens nötigenfalls durch Einführung zusätzlicher Kanten zu Nachbarn werden. Jede zusätzliche Kante bewirkt die Einführung eines neuen Füllelementes (Abb. 4.27).

Eine effiziente Methode der Umordnung wird dadurch eröffnet, daß während der symbolischen Reduktion der aktuelle Grad der verbleibenden Knoten ermittelt wird und auf

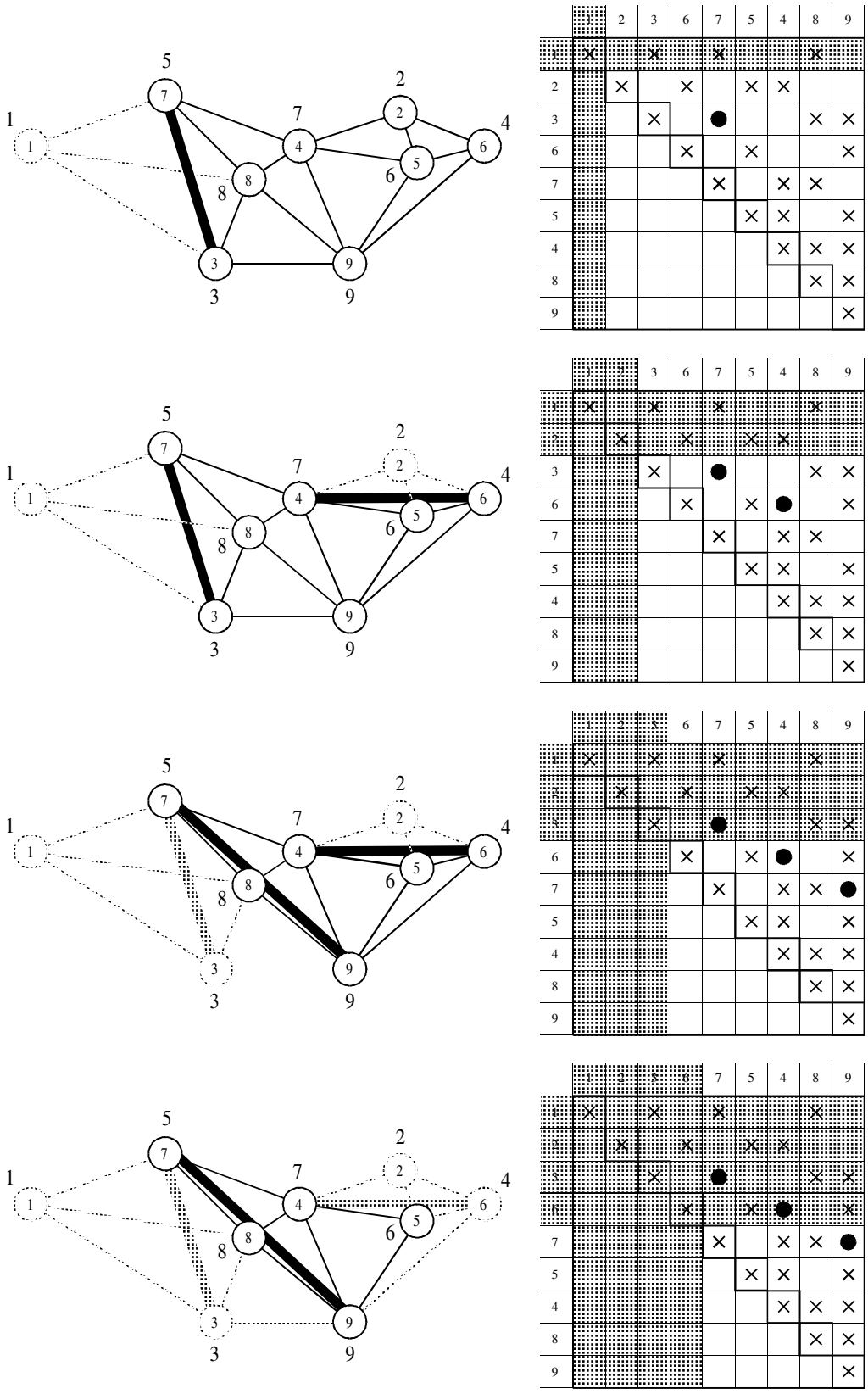


Abbildung 4.27: Graph und Verknüpfungsmatrix während der symbolischen Berechnung.
(Fortsetzung siehe nächste Seite)

Wolf-Dieter Schuh
ITGB, Uni Bonn

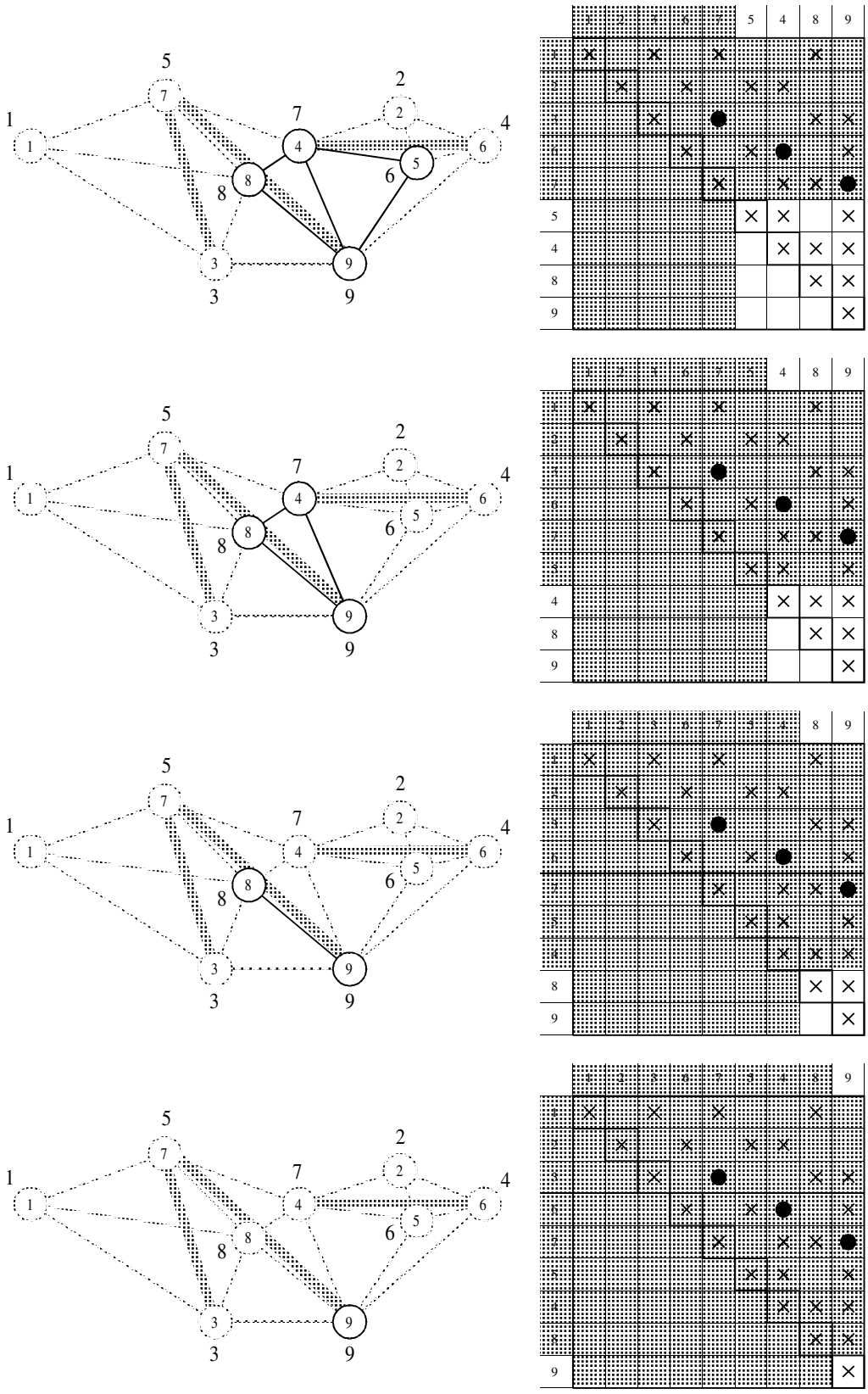


Abbildung 4.27: Graph und Verknüpfungsmatrix während der symbolischen Berechnung.

Wolf-Dieter Schuh
ITGB, Uni Bonn

Grund des laufendend aktualisierten Grades die Neunumerierung durchgeführt wird. Überlegt man die Anwendung bei zugartigen Strukturen (Nivellement- oder Polygonzug) und auf Dreiecken basierende Ketten, so ist der Gewinn unmittelbar einsichtig. Begonnene Züge und Ketten werden im Falle der aktuellen Berechnung des Grades bevorzugt abgearbeitet. Der ausführlichen Beschreibung dieses Verfahrens mit entsprechenden Hinweisen auf speichertechnisch günstige Datenstrukturen und codierten FORTRAN-Programmen ist bei *GEORGE et al. (1981)[34]* das 5. Kapitel (Seite 92-158) gewidmet. Anwendungen an geodätischen Netzen bei *SCHEK et al. (1977)[71]*, Seite 26 zeigen, daß der Mehraufwand der Bestimmung des aktuellen Grades zu beachtlichen Speicherplatzeinsparungen führen kann (über 50gegenüber der Vorwegbestimmung aller Grade).

Um eine elementweise Abspeicherung zu umgehen, verwendet man Umordnungsverfahren, die es gestatten die Form der Matrix bzw. die Orte, an denen Füllelemente entstehen, einzuschränken. Man versucht, die neue Reihenfolge der Knoten so zu ermitteln, daß eine Bandstruktur oder eine Profilstruktur (variable Bandstruktur) aufgebaut wird. Wie aus der Graphentheorie oder aus einer symbolischen Reduktion erkennbar ist, kann außerhalb des Bandes oder Profiles kein neues Füllelement entstehen.

Literatur:

[34] GEORGE Alan and Joseph W-H. LIU (1981): Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

[71] SCHEK Hans-Jörg, Franz STEIDLER und Ulrich SCHAUER (1977): Ausgleichung großer geodätischer Netze mit Verfahren für schwach besetzte Matrizen. Deutsche Geodätische Kommission (DGK), Reihe A, Heft 87.

4.4.2 Bandorientierte Umordnungsalgorithmen

Der Nachteil der Umordnungsmethode nach dem minimalem Grad (Abschnitt 4.4.1) liegt darin, daß Füllelemente an beliebigen Stellen der Matrix auftreten können. Der sehr weit verbreitete Umordnungsalgorithmus von Cuthill-McKee (*CUTHILL et al. (1969)[18]*) beruht darauf, daß benachbarte Knoten möglichst rasch nacheinander bei der Neunumerierung berücksichtigt werden. Die somit erzeugten Strukturen nähern sich Bandmatrizen sehr gut an und können wie solche behandelt werden.

Ein idealer Startknoten ist ein Knoten, der am Rande des Gebietes liegt und möglichst wenige benachbarte Knoten aufweist. Die Festlegung der weiteren Knoten erfolgt stufenweise. Die erste Stufe wird durch die benachbarten Knoten des Startknotens gebildet. Die zweite Stufe bilden dann alle Nachbarpunkte der Punkte der ersten Stufe und so fort. Die Numerierung innerhalb einer Stufe erfolgt auf Grund des zunehmenden Grades, womit Knoten mit vielen Nachbarpunkten eine möglichst hohe Nummer erhalten und die Bandbreite damit möglichst klein gehalten wird.

Algorithmus 4.1 : Umordnungsalgorithmus von Cuthill-McKee

Aufgabenstellung: Suche eine neue Reihenfolge der Knotennummern eines zusammenhängenden Graphen, sodaß sich die Verknüpfungsmatrix einer Bandmatrix mit möglichst kleiner Bandbreite annähert.

Schnittstellen:

Eingabe: `graph` ... Graph mit Knotennummern 1 bis n
`knoten(n)` ... Knotentabelle
`kanten(m)` ... Kantentabelle
`n` ... Anzahl der Knoten
`m` ... Anzahl der Kanten

Ausgabe: `platz(n)` ... neue Platznummer jedes Knotens

Hilfsproceduren:
`STARTKNOTEN(graph,start)` ... siehe Algorithmus 4.2

Lösungsweg: Pseudocode

1. *Initialisiere den Bearbeitungszeiger (akt), den Numerierungszeiger (num) und den Ausgabevektor (platz):*
   `akt = 0 ; num=0 ; platz(1:n)=0 .`
2. *Wähle einen unnumerierten Knoten als Startknoten (siehe Anmerkungen oben, beziehungsweise Algorithmus 4.2):*
   `STARTKNOTEN(graph,start)`
   Erhöhe die Zeiger: `akt=akt+1 ; num=num+1 .`
   Numeriere den Startknoten mit num: `platz(start) = num .`
3. *Numeriere alle Nachbarknoten entsprechend ihres Grades:*
   `Suche alle unnumerierten Nachbarknoten des Knotens akt und numeriere sie fortlaufend (num+1,...,num+u) in der Reihenfolge ihres Grades (u...Anzahl der neunumerierten Nachbarknoten).`
4. *Sind alle Knoten numeriert, dann beende:*
   `IF num=n GOTO Ende:` 'O.K. ENDE'.
5. *Wenn der aktuelle Punkt der letzte numerierte Punkt war, dann wähle einen neuen Startpunkt:*
   `IF akt=num GOTO 2:`.
6. *Erhöhe den Bearbeitungszeiger und bearbeite den nächsten Punkt:*
   `akt=akt+1 ; GOTO 3:`.

Ressourcen: Anzahl der Operationen: $\mathcal{O}(2m)$
Platzbedarf: –

Versucht man eine ungefähre Abschätzung der Anzahl der Operationen, so erkennt man, daß vor allem der dritte Schritt des Lösungsweges analysiert werden muß. Die notwendigen Operationen setzen sich aus zwei Anteilen zusammen. Zum Einen sind zu jedem Knoten einmal die Nachbarknoten aufzusuchen, wofür

$$c_1 * n * m$$

Operationen erforderlich sind (c_1 ... Konstante, n ... Anzahl der Knoten, m ... mittlere Anzahl der Operationen zur Suche der Nachbarknoten). Wurde eine Abspeicherung der Verknüpfungsmatrix gewählt, die einen gezielten Zugriff auf alle Nachbarknoten erlaubt, kann die Größe m proportional zur mittleren Anzahl der Nachbarknoten angesetzt werden. Die zweite Aufgabe stellt die Sortierung aller Knoten einer Stufe dar, wofür

$$c_2 * n * s$$

Operationen notwendig sind (c_2 ... Konstante, s ... mittlere Anzahl der Operationen zur Sortierung der unnumerierten Nachbarknoten). Die Wahl des Sortierverfahrens beeinflußt bei einer großen Anzahl von Verknüpfungen den letzten Faktor ($\mathcal{O}(o^2)$ bis $\mathcal{O}(o \log_2 o)$, wobei o die Anzahl der zu sortierenden Objekte kennzeichnet).

Prinzipiell kann somit festgehalten werden, daß sich die Anzahl der Operationen des Verfahrens proportional zum Produkt der Anzahl der Knoten mit der Anzahl der Verknüpfungen pro Knoten verhält. Dies entspricht der zweifachen Anzahl der ungerichteten Kanten.

Wie die Abb. 4.28 bestätigt, weist die Verknüpfungsmatrix der auf diese Art durchgeführten Neuordnung eine bandähnliche Struktur auf und kann ohne große Verluste als Bandmatrix verarbeitet werden. Neue Füllelemente treten nur innerhalb des Bandes auf, wodurch eine komprimierte Abspeicherung ermöglicht wird.

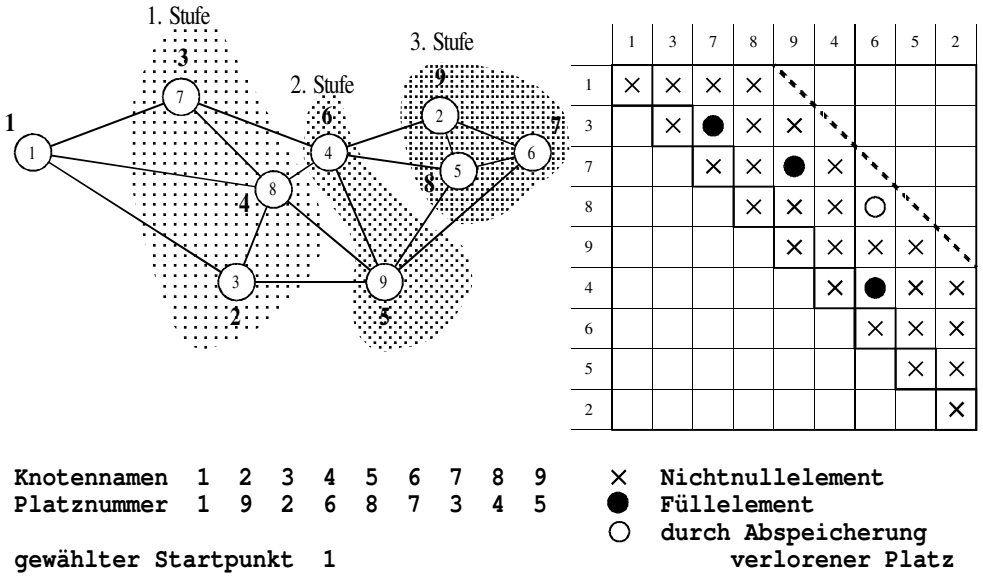
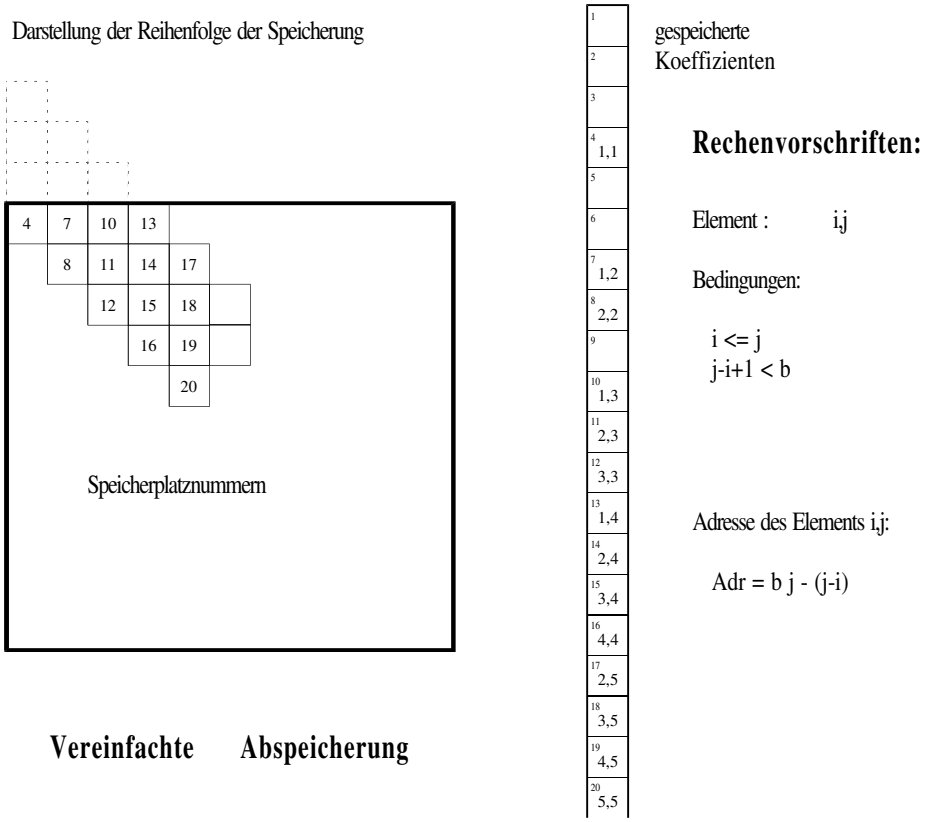


Abbildung 4.28: Ergebnis des Umordnunsverfahrens nach Cuthill-McKee

Entgegen der früher üblichen Speicherung des Bandes in einer Matrix wird hier eine spaltenweise Abspeicherung in einem Vektor propagiert, da somit eine komprimierte interne Darstellung im Rechner ermöglicht wird (siehe einführende Betrachtungen). Abb. 4.29 veranschaulicht die Speicherung und gibt die Rechenformeln zur Adreßrechnung an.



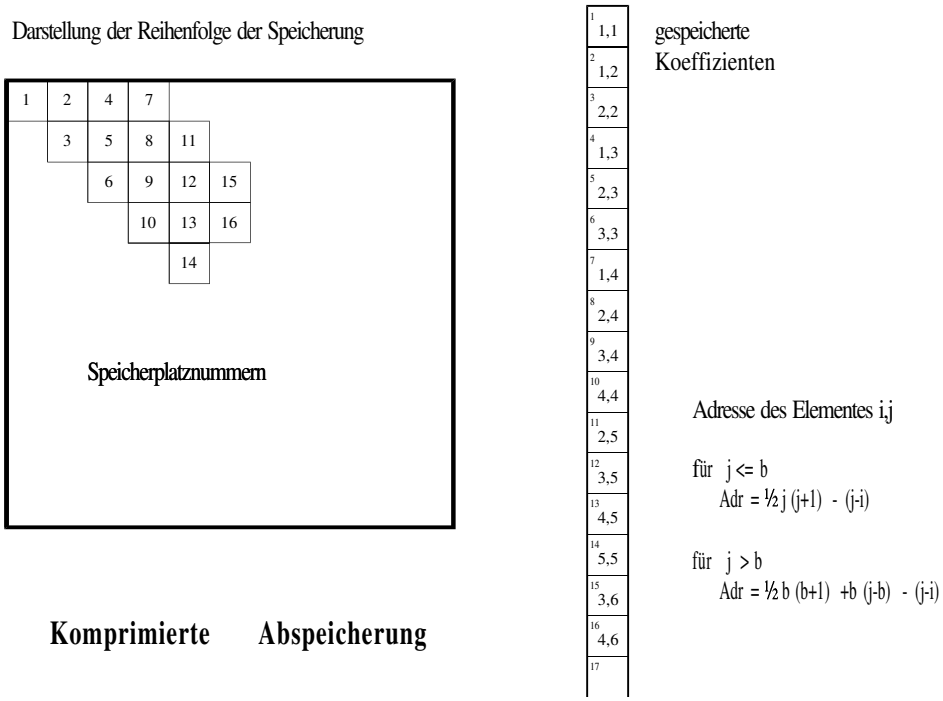


Abbildung 4.29: Speicherung von Bandmatrizen in Vektoren

Eine gute Numerierung durch den Algorithmus von Cuthill-McKee ist erreicht, wenn die Anzahl der Stufen möglichst groß gehalten wird. Damit fällt die Bandbreite meist klein aus, da nur wenige Knoten pro Stufe auftreten. Die Anzahl der Stufen kann durch geeignete Wahl des Startknotens beeinflußt werden.

Als guter Startknoten kann ein Knoten am Rande des Gebietes mit möglichst geringem Grad angesehen werden. Für ovale oder rechteckige Gebiete soll der Knoten am Ende des größeren Durchmessers liegen ('Pseudo-peripherer Knoten'). Sind a priori Informationen wie die Lage der Knoten vorhanden, kann der Startknoten nach den oben festgelegten Kriterien gewählt werden. Eine automatische Festlegung kann in grober Weise durch die Wahl eines Knotens mit maximalem oder minimalem Koordinatenwert durchgeführt werden. Eine exaktere Möglichkeit ist durch die Berechnung der Hauptträgheitsachse mit minimalen Moment gegeben. Jener Knoten, dessen orthogonale Projektion auf die Trägheitsachse vom Schwerpunkt den maximalen Abstand hat, stellt eine gute Wahl als Startpunkt dar (siehe *SCHUH (1981)[72]*, Seite 62-65).

Benützt man nur die Information der Verknüpfungsmatrix, so ist folgende Verfahren in Verwendung. Von der ursprünglichen von *CUTHILL et al. (1969)[18]* vorgeschlagenen Methode für alle Knoten mit einem Grad unter einer bestimmten Schranke eine Neunumerierung testweise durchzuführen ist man abgekommen. Neuere Methoden Versuchen den Startknoten mit Hilfe des maximalen Durchmessers zu ermitteln. *GIBBS et al. (1976)[37]* schlagen dabei folgenden Weg ein. Nach der Wahl eines Knotens werden stufenweise die Nachbarknoten markiert. Alle Knoten der letzten Stufe werden als mögliche Startpunkte getestet und der Knoten mit der maximalen Anzahl von Stufen ermittelt. Mit den Knoten der letzten Stufe kann das Verfahren wiederholt werden bis keine Steigerung der Anzahl der Stufen möglich ist.

Ein modifiziertes Verfahren von *GEORGE et al. (1979)[32]* beschreibt folgenden Weg. Ausgehend von einem Startknoten wird stufenweise markiert. Der Knoten der Letzten Stufe mit minimalem Grad wird als neuer Startknoten gewählt. Das Verfahren wird wiederholt bis keine Steigerung der Anzahl der Stufen möglich ist.

Die Suche nach einem geeigneten Startknoten kann übersichtlich gestaltet werden, wenn man die in Abbschnitt 4.1 eingeführten Bezeichnungen benutzt. Alle Punkte einer Stufe die Nachbarknoten zu schon bearbeiteten Knoten sind, werden als Anrainer bezeichnet. Die Nachbarpunkte dieser Anrainer, die weder bearbeitet noch Anrainer sind, werden in der Gruppe der Zweitanrainer zusammengefaßt.

Algorithmus 4.2 : Ermittle einen Startknoten

Aufgabenstellung: Suche einen geeigneten Startknoten für die Neunumerierung der Knoten. Abhängig von der Wahl des ersten Knotens kann mit diesem Algorithmus von einem zusammenhängenden Graphen oder mehrere nicht zusammenhängenden Graphen ein Startknoten ermittelt werden. Sind mehrere nicht zusammenhängende Graphen vorhanden, so wird jeweils der Startknoten im Teilgraphen ermittelt, wo der erste Knoten gewählt wurde.

Schnittstellen:

Eingabe: `graph` ... Graph mit Knotennummern 1 bis n
`knoten(n)` ... Knotentabelle
`kanten(m)` ... Kantentabelle
`n` ... Anzahl der Knoten
`m` ... Anzahl der Kanten

Ausgabe: `start` ... Nummer des Startknotens

Hilfsproceduren:
`DURCHMESSER(graph,start,stufen,Kandidaten)` ... siehe Algorithmus 4.3

Lösungsweg nach Gibbs-Poole Stockmeyer: Pseudocode

1. Wähle einen (unnumerierten) Startknoten mit niedrigem Grad. Merke den Startknoten.
2. Führe den Hilfsalgorithmus 4.3 durch. DURCHMESSER(graph,start,Kandidaten).
3. Bei Beendigung des Hilfsalgorithmus 4.3 mit 'Startknoten ist ein isolierter Knoten' GOTO Ende: 'o.k. Ende'.
4. Merke den aktuellen Stufenzähler zum gemerkten Startknoten (Kandidaten). Alle Kandidaten der letzten Stufe sind als mögliche Startknoten zwischenzuspeichern.
5. Untersuche alle möglichen Startknoten durch Anwendung des Hilfsalgorithmus 4.3. Wähle jenen Knoten als neuen Startknoten, der die maximale Anzahl an Stufen aufweist.
6. Weist der neue Startknoten eine höhere Anzahl von Stufen auf, dann merke den neuen Startknoten und GOTO 2:.
7. Ende: 'o.k. Ende'.

Lösungsweg nach George-Liu: Pseudocode

1. Wähle einen beliebigen, unnumerierten Startknoten. Merke den Startknoten.
2. Führe den Hilfsalgorithmus 4.3 durch. DURCHMESSER(graph,start,Kandidaten).
3. Bei Beendigung des Hilfsalgorithmus 4.3 mit 'Startknoten ist ein isolierter Knoten' GOTO Ende: 'o.k. Ende'.
4. Wähle den Knoten der letzten Stufe mit minimalem Grad als neuen Startknoten. Merke den Knoten als Versuchsknoten.
5. Führe den Hilfsalgorithmus 4.3 durch. DURCHMESSER(graph,start,Kandidaten).
6. Wenn der aktuelle Stufenzähler des Versuchsknotens kleiner gleich dem gemerkten Stufenzähler des Startknotens ist, dann stellt der gemerkte Startknoten den gesuchten Knoten dar. GOTO Ende: 'o.k. Ende'.

7. `Merke den aktuellen Stufenzähler und den zugehörigen Versuchsknoten als Startknoten.`
8. `GOTO 5:`

Algorithmus 4.3 : Ermittle Durchmesser

Aufgabenstellung: Bestimme zu einem vorgegebenen Startknoten die Anzahl der Stufen und ermittle die Knoten der letzten Stufe. Wenn nur ein Knoten im Graph vorhanden ist, so bezeichne ihn als isolierten Knoten und belasse den Stufenzähler auf dem initialisierten Wert.

Schnittstellen:

Eingabe:	`graph`	... Graph mit Knotennummern 1 bis n
		`knoten(n)` ... Knotentabelle
		`kanten(m)` ... Kantentabelle
		`n` ... Anzahl der Knoten
		`m` ... Anzahl der Kanten
	`start`	... Startknoten
Ausgabe:	`stufen`	... Anzahl der Stufen
	`Kandidaten`	... Knoten der letzten Stufe

Lösungsweg: Pseudocode

1. `Initialisiere die Gruppe der Anrainer und den Stufenzähler.`
2. `Alle Nachbarknoten des Startknotens kommen in die Gruppe der Anrainer.`
3. `Ist die Gruppe der Anrainer leer, dann ist der Startknoten isolierter Knoten ohne zu numerierende Nachbarpunkte. GOTO Ende:` 'STARTKNOTEN IST EINE ISOLIERTER KNOTEN'.
4. `Erhöhe den Stufenzähler und initialisiere die Gruppe der Zweitanrainer.`
5. `Markiere alle Anrainer als zur entsprechenden Stufe gehörig und sammle deren unnumerierten Nachbarknoten, die noch keine Anrainer sind, in der Gruppe der Zweitanrainer.`
6. `Ist die Gruppe der Zweitanrainer nicht leer, dann wandle alle Zweitanrainer zu Anrainern um und GOTO 4:`
7. `Merke den Stufenzähler und alle Anrainer der letzten Stufe.` `Ende:` 'O.K. ENDE'

Da bei diesen Lösungswegen die Sortierung der Knoten einer Stufe wegfällt, kommt nur der erste Teil der Betrachtungen über die Rechenoperationen des Algorithmus von Cuthill-McKee zu tragen. Die Anzahl der Operationen zur Suche des Startpunktes ist somit proportional der Anzahl der Kanten. Zur Wahl der Version ist auf die einfachere Implementierung der Version nach George-Liu hinzuweisen. Die Zwischenspeicherung

aller Knoten der letzten Stufe und die möglicherweise große Anzahl von Knoten der letzten Stufe stellen die Schwachpunkte der Version nach Gibbs-Poole-Stockmeyer dar. Ausführliche vergleichende Tests der beiden Algorithmen und die Untersuchung anderer Ansätze sind bei *GEORGE et al. (1979)[32]* finden. Alle Algorithmen für die Neunumerierung und die Suche nach einem guten Startpunkt können miteineinander kombiniert aber auch ineinander geschachtelt werden, sodaß mit jedem Versuchsknoten als Startknoten eine mögliche Neunumerierung sofort durchgeführt wird. Neben einem Mehraufwand an Speicherplätzen und einem erhöhten Zeitaufwand zur Suche des Startknotens, tritt kein zusätzlicher Aufwand durch das Numerierungsverfahren ein. Bei guten Startlösungen kann dadurch ein Arbeitsgewinn erreicht werden.

Aufgrund der willkürlichen Numerierung von Kandidaten mit gleichem Grad innerhalb einer Stufe liefert der Algorithmus von Cuthill-McKee nicht die optimale Lösung (minimale Bandbreite). Die fast optimale Numerierung bildet aber eine gute Voraussetzung zur Minimierung der Bandbreite durch den Algorithmus von Rosen (*ROSEN (1968)[68]*). Mit Hilfe dieser Rechenvorschrift kann die Bandbreite verringert aber auch optimal gestaltet werden. Die Grundidee ist folgende: Knotenpaare, die die Bandbreite bestimmen, sind so mit anderen Knoten zu tauschen, daß die Bandbreite verkleinert wird oder zumindest gleich bleibt.

Algorithmus 4.4 : Bandminimierung nach Rosen

Aufgabenstellung: Suche eine Reihenfolge der Knotennummern eines Graphen, sodaß sich die Verknüpfungsmatrix einer Bandmatrix mit (möglichst) kleinster Bandbreite annähert.

Schnittstellen:

Eingabe:
- `graph` ... Graph mit Knotennummern 1 bis n
  - `knoten(n)` ... Knotentabelle
  - `kanten(m)` ... Kantentabelle
  - `n` ... Anzahl der Knoten
  - `m` ... Anzahl der Kanten
- `band_ziel` ... ≤ 0 ... bestmögliche Numerierung; > 0 ... suboptimale Numerierung mit einer Bandbreite kleiner dieser Schranke

Ausgabe:
- `platz(n)` ... neue Platznummern jedes Knotens
- `platz_rev(n)` ... invertierter Indexvektor `platz(n)`

Anmerkung: kleine Buchstaben deuten auf die ursprüngliche Numerierung; große Buchstaben weisen auf das neunumerierte System

- `platz(i)` ... Platznummer des ursprünglichen Knotens `i` im neunumerierten System → `I`
- `platz_rev(I)` ... Platznummer des neunumerierten Knotens `I` im ursprünglichen System → `i`

Felder:
- `tausch(n)`

Lösungsweg: Pseudocode

Wolf-Dieter Schuh
ITGB, Uni Bonn

1. *Initialisierung der Platznummern, der aktuellen Bandbreite und der Wiederholungszeiger:*
   `DO i=1 TO N ; platz(i)=i ; platz_rev(i)=i ; END DO i ;`
   `band_akt=n+1 ; tausch(1:n)=.FALSE. .`

2. *Festlegung des bandbestimmenden Elementes:*
   `Durchsuche alle Knoten i, i=1,...,n und suche jene Kante (i,j)`
   `für die gilt: platz(j)-platz(i) ... maximum. Für den`
   `so bestimmten Knoten i, merke gleichzeitig die Kante (i,k) für`
   `die gilt, daß platz(i)-platz(k) ... minimal wird.`

3. *Berechnung der aktuellen Bandbreite:*
   `I=platz(i) ; J=platz(j) ; K=platz(k); band_neu=J-I+1 .`
   Abbruch wenn die Zielbandweite erreicht ist:
   `IF band_neu` $\leq$ `band_ziel GOTO` Ende: 'O.K. ENDE ZIELBANDWEITE UNTERSCHRITTEN' .

4. *Überprüfe ob eine Verkleinerung der Bandbreite eingetreten ist und initialisiere bei verkleinerter Bandbreite die Wiederholungszeiger:*
   `IF band_neu=band_akt GOTO` 5:
   `band_akt=band_neu ; tausch(1:n)=.FALSE.`

5. *Versuche das kleinere Element i mit Gewinn zu tauschen: Durchsuche die Knoten H:*
   `DO H=I+1 TO J+K-I ;`
   `h = platz_rev(H) ;`
   `Wenn von allen Kanten des Knotens h keine Kante (h,o),`
   `existiert, wo platz(o)`$\geq$`J so ist eine Austauschzeile gefunden.`
   Merke für den Austausch i und I und gehe zum Austausch:
   `p=i; P=I; GOTO` 10: .
   `END DO H .`

6. *Versuche das größere Element j mit Gewinn zu tauschen:*
   `Suche im Knoten j jene Kante (j,`ℓ`) für die gilt:`
   `platz(`ℓ`)-platz(j) ... maximum.`
   `L=platz(`ℓ`) DO H=H=L+I-J TO J-1; h = platz_rev(H) ;`
   `Wenn von allen Kanten des Knotens h keine Kante (h,o),`
   `existiert, wo platz(o)`$\leq$`I so ist eine Austauschzeile gefunden.`
   Merke für den Austausch i und I und gehe zum Austausch:
   `p=i; P=I; GOTO` 10: .
   `END DO H .`

7. *Versuche das kleinere Element i ohne Gewinn zu tauschen:*
   Durchsuche die Knoten H:
   `DO H=I+1 TO J+K-I+1 ;`
   `h = platz_rev(H) ;`
   `Wenn von allen Kanten des Knotens h keine Kante (h,o),`
   `existiert, wo platz(o)>J so ist eine Austauschzeile gefunden.`
   Merke für den Austausch i und I und gehe zum Austausch:
   `p=i; P=I; GOTO` 10: .
   `END DO H .`

Wolf-Dieter Schuh
ITGB, Uni Bonn

8. *Versuche das größere Element j ohne Gewinn zu tauschen:*
   `DO H=H=L+I-J-1 TO J-1; h = platz_rev(H) ;`
   `Wenn von allen Kanten des Knotens h keine Kante (h,o),`
   `existiert, wo platz(o)<I so ist eine Austauschzeile gefunden.`
   Merke für den Austausch i und I und gehe zum Austausch:
   `p=i; P=I; GOTO` `10:` .
   `END DO H .`
9. *Beende da kein Knoten zum Tausch gefunden werden konnte:*
   `Ende:` 'O.K. ENDE – BANDBREITE KANN NICHT WEITER VERRINGERT WERDEN' .
10. *Tausche die Knoten h und p sofern beide Knoten bei dieser Bandbreite noch nicht getauscht wurden:*
    `IF tausch(h)=.TRUE. AND tausch(p)=.TRUE. GOTO` `9:`
    Führe Austausch durch:
    `Hilfe=platz(h), platz(h)=platz(p), platz(p)=Hilfe;`
    `Hilfe=platz_rev(H), platz_rev(H)=platz_rev(P),`
    `platz_rev(P)=Hilfe.`
    Markiere die getauschten Knoten und versuche weiteren Tausch:
    `tausch(h)=.TRUE. , tausch(p)=.TRUE. ; GOTO` `2:`

Gegenüber der Originalveröffentlichung dieser Rechenvorschrift sind folgende Modifikationen empfehlenswert und im Algorithmus 4.4 eingearbeitet.

- Durch den Übergang auf eine knotenweise geordnete Knoten-Kantenstruktur kann die Bandbreite zu jedem Knoten ermittelt werden, ohne daß alle Kanten durchsucht werden.
- Durch Zugriff auf die Platznummer mit Hilfe der indirekten Adressierungen `platz` und `platz_rev` kann die Umarbeitung der Knoten-Kantentabelle vermieden werden.
- Indem die mögliche Anzahl der Zeilen (Spalten), die für den Tausch zur Verfügung stehen, eingeschränkt werden, kann eine Steigerung der Verarbeitungsgeschwindigkeit erreicht werden. Der Einschränkung liegt die aktuelle Bandbreite der bandbestimmenden Knoten zu Grunde (siehe Abb. 4.30).

Dieser Algorithmus kann direkt auf die Ausgangsdaten angewandt werden, aber auch durch Modifikation des Initialsierungsschrittes (Algorithmus 4.4, Schritt 1) auf vornumerierte Systeme. Wie empirische Studien (*CUTHILL et al. (1969)[18]*) mit dem Originalverfahren zeigen, vergrößert sich der Rechenaufwand bei Verwendung einer beliebigen Numerierung. Für unregelmäßige Strukturen wird eine Vorbehandlung durch einen bandannähernden Algorithmus (z.B. Cuthill-McKee) empfohlen. Ferner werden in dieser Arbeit einige regelmäßige Strukturen aufgezeigt, wo der Algorithmus von Rosen nicht die minimale Bandbreite erzeugt.

Die Grenzen aller bandorientierter Verfahren zeigen sich unmittelbar bei Graphen deren Knoten stark unterschiedlichen Grad aufweisen. Hat ein Knoten viele Nachbarpunkte

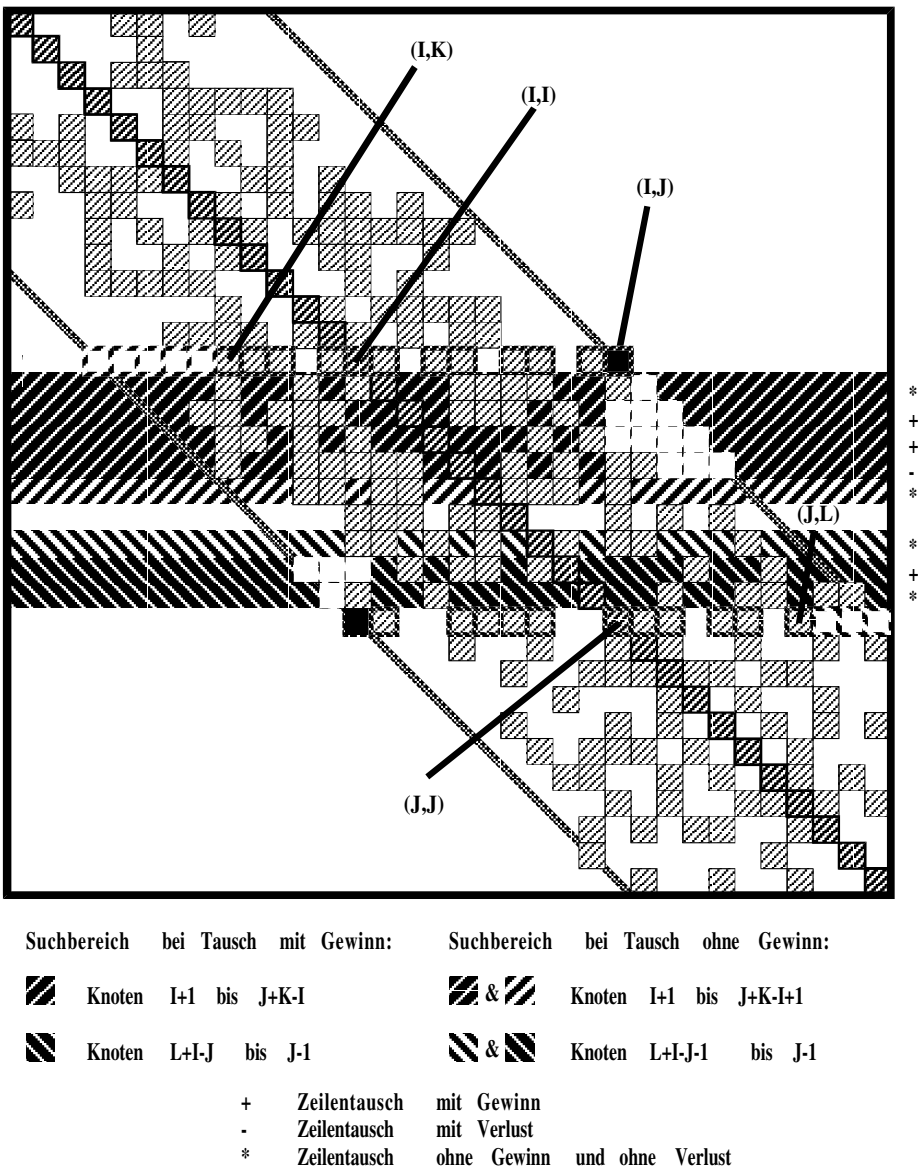


Abbildung 4.30: Einschränkung von möglichen Austauschknoten.

so beeinflußt er die Bandbreite, wobei der Knoten eines Graphen mit maximalem Grad eine untere Schranke für die minimale Bandbreite darstellt. Die minimale Bandbreite muß größer sein als der halbe Wert des maximalen Grades. Betrachtet man beispielsweise ein Streckennetz mit unbekanntem Maßstabsfaktor, so ist bei der Darstellung des Netzes als Graph ein Knoten mit Kanten zu allen Nachbarpunkten einzuführen. Bei diesem Graph versagen alle bandorientierten Verfahren. Sonderbehandlungen von einzelnen, große Bandbreiten erzeugenden Knoten können dafür Abhilfe schaffen (siehe z.B. *CRESPI et. al. (1989)[17]*).

Literatur:

[17] CRESPI Mattia, Gianfranco FORLANI, Luigi MUSSIO (1989): Optimization of the Reordering Algorithm for Least Squares Problems. Tutorial on

"Mathematical Aspects of Data Analysis", ISPRS, Intercommission Working Group III/VI, Pisa, pp. 185-202.

[18] CUTHILL E. and J. McKEE (1969): Reducing the Bandwidth of Sparse Symmetric Matrices. Proc. ACM National Conference pp. 157-172. Association for Computing Machinery. New York.

[32] GEORGE Alan, Joseph W-H. LIU (1979): An Implementation of a Pseudoperipheral Node Finder. ACM Transactions on Mathematical Software. Vol. 5, No. 3., pp. 284-295.

[37] GIBBS N.E., W.G. POOLE and P.K. STOCKMEYER (1976): An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. SIAM Numerical Analysis Vol. 13, No. 2, pp. 236-250.

[68] ROSEN Richard (1968): Matrix Bandwidth Minimization. Proc. ACM National Conference, pp. 585-595. Brandon System Press, Princeton, New York.

[72] SCHUH Wolf-Dieter (1981): Programmierung rationeller Algorithmen zur Umordnung, Auflösung und Inversion der Normalgleichungen geodätischer Netze. Diplomarbeit, TU Graz.

4.4.3 Hüllenorientierte Umordnungsalgorithmen

Durch die komprimierte Speicherung von schwachbesetzten Matrizen in einer Bandform wird eine Einsparung an Speicherplatz und Adreßinformation erzielt. Durch die sehr einfache Adreßrechnung wird ein rascher Zugriff gewährleistet. Diese positiven Eigenschaften werden bei einer hüllenorientierten (profilorientierten) Speicherung übernommen. Durch die variable Bandbreite wird eine größere Vielseitigkeit erlangt, die bei allen unregelmäßigen Netzen Vorteile bringt.

Zunächst zur Speicherung: Eine einfachen und raschen Zugriff gewährende Speicherform ist durch die spaltenweise Ablage der oberen Dreieckmatrix in einem Vektor gegeben. Die Adressierung wird durch einen Indexvektor IND ermöglicht, wo jeweils die Adresse des Diagonalgliedes im Vektor gemerkt ist. Abb. 4.31 veranschaulicht diese Art der Speicherung.

Ein mögliches Verfahren zur Umnumerierung stellt die umgekehrte Numerierung nach Cuthill-McKee dar (*'Reversed Cuthill-McKee'*). Aus Abb. 4.28 erkennt man die Motivation für die Umkehrung der Numerierung. Da der Algorithmus von Cuthill-McKee die Numerierung stufenweise vollzieht, werden Knoten mit hohem Grad meist vor dem Großteil ihrer Nachbarknoten numeriert, wodurch typische waagrechte Balken in der neugeordneten Verknüpfungsmatrix entstehen. Durch Umkehrung der Numerierung werden diese waagrechten Balken zu senkrechten Balken, die somit vorteilhaft mit hüllenorientierter Speicherung verarbeitbar sind. Die Umkehrung der Numerierung hat zwar keinen Einfluß auf die Bandbreite, bewirkt jedoch sehr häufig eine Reduzierung des Speicherplatzes für das Profil. Wie J. Liu und A.H. Sherman (1976) (siehe *GIBBS et al. (1976)[36]*, Seite 327) zeigten kann die Anforderung an Speicherplatz durch die Umkehrung nie steigen. *GIBBS et al. (1976)[37]* modifizierten den Algorithmus von Cuthill-McKee und erstellten einen Algorithmus der sowohl das Band wie auch das Profil möglichst klein hält. Der Nachteil dieser beiden zuvor erwähnter Verfahren bei unregelmäßiger Struktur liegt in der Bandminimierung (siehe dazu letzter Absatz von Kap. 4.4.2).

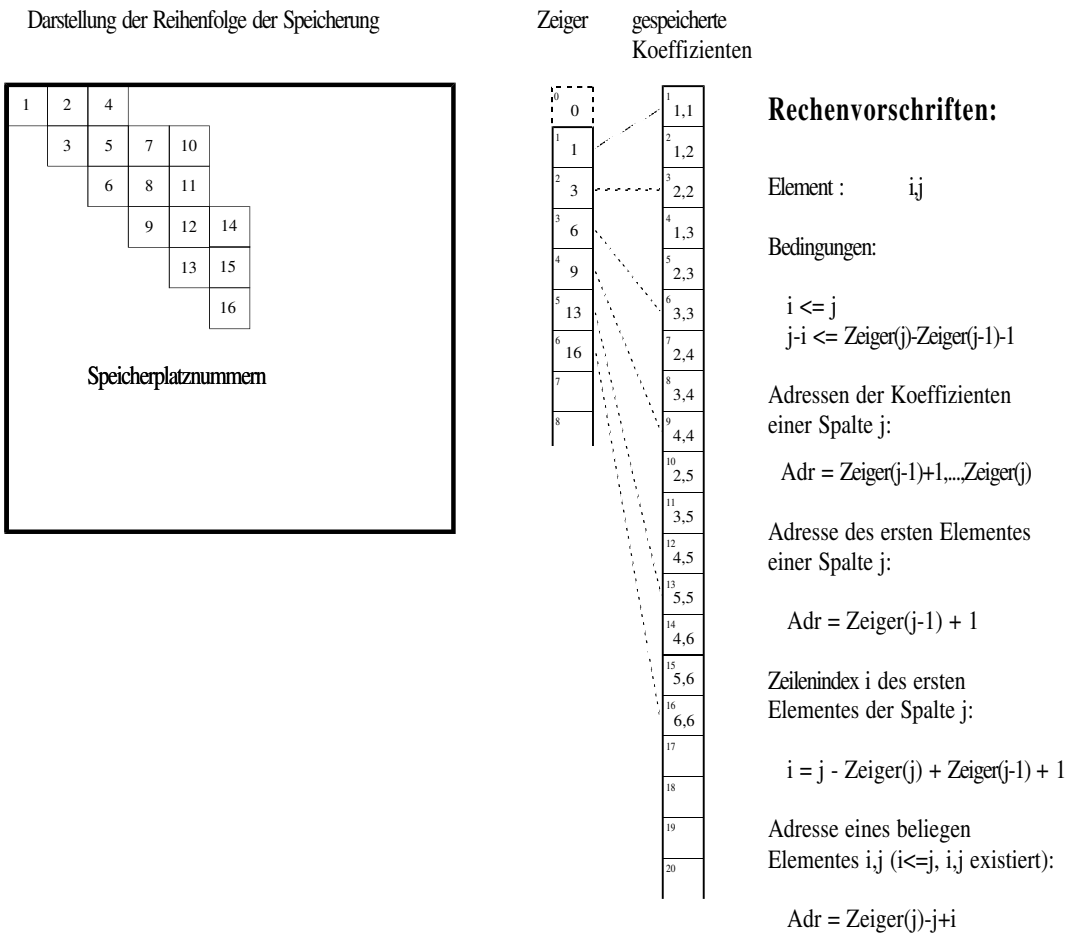


Abbildung 4.31: Hüllenorienterte Speicherung.

Ein anderer Weg zur Erstellung einer Neunumerierung bei hüllenorientierter Speicherung wurde von *KING (1970)[46]* initiiert. Er propagierte eine Numerierungsvorschrift, die den nächsten zu numerierenden Punkt so auswählt, daß möglichst wenige bisher unberührten Knoten zu Nachbarknoten werden. Die auf dieser Idee aufbauenden Algorithmen unterscheiden sich im wesentlichen in zwei Punkten. Erstens in der Auswahl der Starpunkte: Während *KING (1970)[46]* eine versuchsweise Numerierung mit mehreren beliebig gewählten Startpunkten in Betracht zog, verbesserte *GIBBS (1976)[35]* die Effizienz durch die Wahl eines Knotens der am maximalen Durchmesser liegt (siehe dazu Kap. 4.4.2). Das zweite Unterscheidungsmerkmal stellt die Menge der Punkte dar, aus denen der neu zu numerierende Knoten auszuwählen ist. Die ursprüngliche Idee von *KING (1970)[46]* war es, den neuen Knoten nur aus den Nachbarknoten der schon numerierten Knoten zu ermitteln. R. LEVY wählt ein Jahr später den neu zu numerierenden Knoten aus allen möglichen unnumerierten Knoten aus, wodurch eine bessere Numerierung um den Preis eines erhöhten Arbeitsaufwands erreicht wurde. Aus vergleichenden Arbeiten von *CUTHILL (1972)[19]* und *GIBBS (1976)[37]* geht hervor, daß dieser erhöhte Rechenaufwand nicht zu rechtfertigen ist. Einen sehr effizienten Kompromiß stellt der 'Bankers'-Algorithmus (*SNAY (1976)[80]*) dar, wo als

Kandidat für den neu zu numerierenden Knoten, sowohl die unmittelbaren Nachbarknoten der numerierten Knoten (wartende Knoten), als auch die Nachbarknoten dieser Knoten herangezogen werden können. Der Name des Algorithmus wurde geprägt durch die Möglichkeit der anschaulichen Darstellung der Funktionsweise durch folgendes Warteschlangenproblem:

> Eine Gemeinde will ein großes Waldgrundstück teilen und auf alle Familien des Dorfes abgeben. Sie wünscht jedoch, daß alle Familien alle bestehenden Schulden untereinander begleichen. Die Gemeinde beauftragt einen Notar in der nächsten Stadt, der die einzelnen Grundstücke nach dem Bezahlen der Schulden vergeben soll. Der Notar lädt eine Familie ein. Am nächsten Tag müssen alle Familien anwesend sein, die bei der eingeladenen Familie Schulden haben, oder bei denen diese Familie verschuldet ist, um diese Verpflichtungen zu begleichen. Am Abend wählt der Notar eine Familie aus, die nicht unbedingt schon anwesend sein muß, die aber eine Schuldbeziehung zu mindestens einer anwesenden Familie haben muß. Diese Familie soll so ausgewählt werden, daß möglichst wenige Familien neu anreisen müssen. Auf diese Art wird die Zahl der angereisten Familien möglichst gering gehalten. Die einmal angereisten und damit bis zur eigenen Bearbeitung anwesenden Familien bezeichnet man als die Gruppe der wartenden Familien. Alle Familien, die für die Wahl als nächste Familie in Frage kommen, werden Kandidaten genannt. Das sind alle Familien, die eine Schuldbeziehung zu mindestens einer der früher angereisten noch nicht bearbeiteten Familien hat.

Die Beziehung zwischen dem aufgezeigten Warteschlangenproblem und der Problemstellung eine Neunumerierung zur profilorientierten Speicherung zu finden ist leicht herzustellen. Die Familien haben dieselbe Bedeutung wie die Knoten eines Graphen. Die Schuldbeziehungen sind mit den ungerichteten Kanten zu vergleichen. Die Reihenfolge, in der der Notar die Grundstücke vergibt, entspricht der neunumerierung der Knoten. Die einzelne Familie muß erst dann anreisen und in der Stadt warten, wenn die erste Familie, die mit dieser Familie in einer Schuldbeziehung steht, ein Grundstück erhält. Diese Wartezeit, die in der Summe aller Familie beinahe minimiert wird, entspricht der Spaltenhöhen oder Profilen der Verknüpfungsmatrix. Als Spaltenhöhen oder Profile bezeichnet man die Anzahl der Glieder zwischen dem ersten Nichtnullelement in der Spalte und dem Diagonalglied. Wie man erkennt, kann mit diesem Ansatz auch ein Bandalgorithmus leicht veranschaulicht werden. Beim Bandalgorithmus wird nicht die Summe der Wartezeiten aller Familien zu einem Minimum, wie beim Profilalgorithmus, sondern die längste Wartezeit einer einzelnen Familie wird möglichst minimal gehalten.

Der Profilalgorithmus wie er von *SNAY (1976)[80]* beschrieben wird, hat folgenden Ablauf.

Algorithmus 4.5 : Umordnungsalgorithmus nach Snay

> **Aufgabenstellung:** Suche eine neue Reihenfolge der Knotennummern eines Graphen, sodaß die Verknüpfungsmatrix durch eine hüllenorientierte Speicherung angenähert werden kann.

Schnittstellen:

Eingabe: `graph` ... Graph mit Knotennummern 1 bis n
`knoten(n)` ... Knotentabelle
`kanten(m)` ... Kantentabelle
`n` ... Anzahl der Knoten
`m` ... Anzahl der Kanten

Ausgabe: `platz(n)` ... neue Platznummern jedes Knotens

Hilfsproceduren:
`STARTKNOTEN(graph,start)` ... siehe Algorithmus 4.2

Lösungsweg: Pseudocode

1. *Initialisierung der Platznummern und des Numerierungszeigers (num):*
   `platz(1:n) = 0 ; num = 0 .`
2. *Wähle einen unnumerierten Knoten als Startknoten:*
   Führe Algorithmus 4.2 aus:
   `STARTKNOTEN(graph,start)`
   Erhöhe die Zeiger: `num = num+1 .`
   Numeriere den Startknoten mit num: `platz(start) = num .`
   `Initialisiere die Gruppe der wartenden Knoten und die Gruppe der Kandidaten.`
3. *Aktualisiere der Gruppe der wartenden Knoten:*
   `Jeder unnumerierte Knoten, der mit dem zuletzt numerierten Knoten in Verbindung steht, kommt in die Gruppe der wartenden Knoten.`
4. *Aktualisiere die Gruppe der Kandidaten:*
   `Alle Punkte, die mit dem zuletzt numerierten Knoten oder mit dessen Nachbarknoten eine direkte Verbindung haben, werden in die Gruppe der Kandidaten aufgenommen, sofern sie nicht`
   - `schon numeriert sind;`
   - `nicht schon in der Kandidatenlist sind.`
5. *Überprüfe ob noch Kandidaten vorhanden sind:*
   `IF Kandidatenliste leer GOTO` 8:
6. *Wahl des neuzunumerierenden Knotens:*
   `Wähle einen Knoten aus der Kandidatenliste für den gilt:`

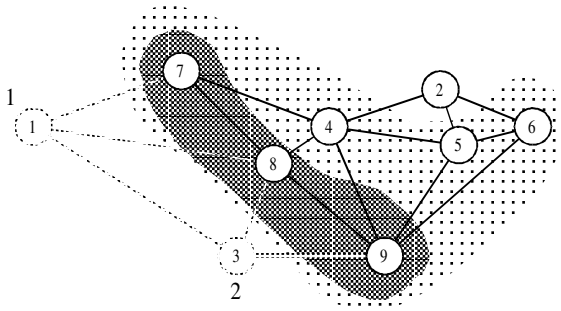
   `mv-nv = Minimum`

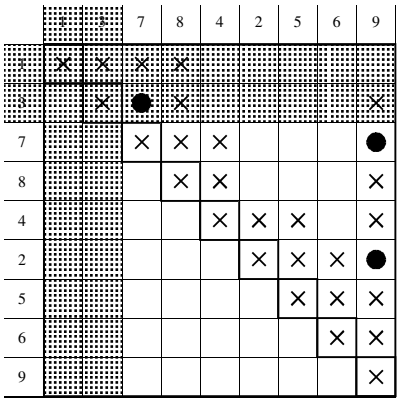
   `mv...Anzahl der nicht numerierten Knoten, die mit diesem Knoten in Verbindung sind und noch nicht zu den wartenden Knoten gezählt werden.`
   `nv...= 1 wenn der Knoten ein wartender Knoten ist.`
   `...= 0 wenn dieser Knoten noch kein wartender Knoten ist.`
7. *Führe die Neunumerierung durch und bereinige der Gruppen:*
   Erhöhe den Numerierungszeiger: `num=num+1`
   `Numeriere den gefundenen Knoten mit num.`

```
Lösche den Knoten aus der Gruppe der Kandidaten und wenn
notwendig auch aus der Gruppe der wartenden Knoten. GOTO 3:
```

8. *Überprüfe ob alle Knoten numeriert sind:*
   `IF num=n GOTO Ende:` 'O.K. ENDE'
9. *Suche neuen Startknoten, da mehrere unabhängige Teilgraphen vor liegen:*
   `GOTO 2:`

Kernpunkt des vorhergehenden Algorithmus 4.5 stellt die Berechnung des aktuellen Grades der Knoten dar. Der aktuelle Grad gibt an, wieviele Knoten bei der Numerierung dieses Knotens als wartende Knoten hinzu- oder wegkommen. Abb. 4.32 ermöglicht das Studium eines Zwischenschrittes. Bei Numerierung des Knotens 4 als





Neunumerierte Knoten 1, 3:

Gruppe der wartenden Knoten (7,8,9)

Gruppe der Kandidaten (4,5,6,7,8,9)

× **Nichtnullelement**
● **Füllelement**

Wahl des neuzunumerierenden Knotens:

Kandidaten	**4**	**5**	**6**	**7**	**8**	**9**
mv	**2**	**3**	**2**	**1**	**1**	**3**
nv	**0**	**0**	**0**	**1**	**1**	**1**
mv-nv	**2**	**3**	**2**	**0**	**0**	**2**

==> Knoten 7 oder 8 soll numeriert werden.

Abbildung 4.32: Umordnungsalgorithmus nach Snay, Zwischenergebnis.

nächster Knoten müssen demnach zwei weitere Knoten in die Gruppe der wartenden Knoten aufgenommen werden (Knoten 2 und 5). Da der Knoten 4 selbst nicht der Gruppe der wartenden Knoten angehört, verläßt kein Knoten die Gruppe der wartenden Knoten. Zur Numerierung des Knoten 7 wird kommt der Knoten 4 in die Gruppe der wartenden Knoten und der Knoten 7 verläßt diese Gruppe (aktueller Grad=0). Da die Numerierung der Knoten nach dieser Vorschrift nicht eindeutig möglich ist, wird keine streng minimierte Profilmatrix erlangt. Der Rechen- beziehungsweise Verwaltungsaufwand für die Umnumerierung ist höher wie beim Verfahren von Cuthill-McKee. Um diesen Aufwand zu verringern, kann der Algorithmus modifiziert werden, indem man die Berechnung $mv - nv$ fortlaufend aktualisiert und nicht immer neu durchführt.

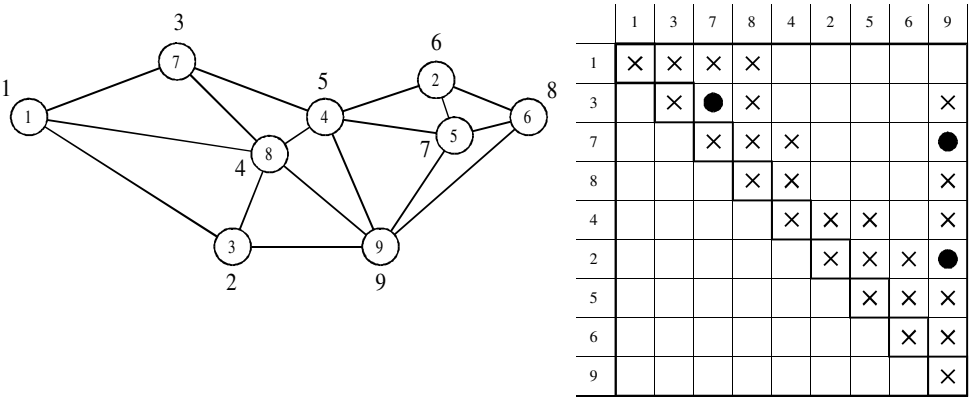


Abbildung 4.33: Umordnungsalgorithmus nach Snay, Endergebnis.

Da die Numerierung der Knoten nach dieser Vorschrift nicht eindeutig möglich ist, wird keine streng minimierte Profilmatrix erlangt. Der Rechen- beziehungsweise Verwaltungsaufwand für die Umnumerierung ist höher wie beim Verfahren von Cuthill-McKee. Um diesen Aufwand zu verringern, kann der Algorithmus modifiziert werden, indem man die Berechnung des aktuellen Grades $mv - nv$ fortlaufend aktualisiert und nicht immer neu durchführt.

Algorithmus 4.6 : Modifizierter Umordnungsalgorithmus nach Snay

Aufgabenstellung: Suche eine neue Reihenfolge der Knotennummern eines Graphen, sodaß die Verknüpfungsmatrix durch eine hüllenorientierte Speicherung angenähert werden kann.

Schnittstellen:

Eingabe: `graph` ... Graph mit Knotennummern 1 bis n
`knoten(n)` ... Knotentabelle
`kanten(m)` ... Kantentabelle
`n` ... Anzahl der Knoten
`m` ... Anzahl der Kanten

Ausgabe: `platz(n)` ... neue Platznummern jedes Knotens

Hilfsfelder: `akt_grad(n)`

Hilfsproceduren:
`STARTKNOTEN(graph,start)` ... siehe Algorithmus 4.2

Lösungsweg: Pseudocode

1. *Initialisierung der Platznummern und des Numerierungszeigers (num):*
   `platz(1:n) = 0 ; num = 0 .`

2. Stelle den Grad (=Anzahl der Nachbarverbindungen) jedes Knotens fest und trage die Zahl (aktueller Grad) in eine Liste ein.

3. *Wähle einen unnumerierten Knoten als Startknoten:*
   Führe Algorithmus 4.2 aus:
   `STARTKNOTEN(graph,start)`
   Erhöhe die Zeiger: `num = num+1 .`
   Numeriere den Startknoten mit num: `platz(start) = num .`
   Der frühere Zustand dieses Punktes ist gleichzusetzen mit dem eines Kandidaten. Initialisiere die Gruppe der wartenden Knoten und die Gruppe der Kandidaten.

4. *Schleife über alle Nachbarknoten:*
   Suche der Reihe nach alle unnumerierten Nachbarknoten des zuletzt numerierten Knotens.
   Wenn kein Knoten mehr gefunden wird dann GOTO 10:

5. *Aktualisiere den Grad der Nachbarknoten auf Grund des früheren Zustandes des neunumerierten Knotens:*
   Wenn der frühere Zustand des neunumerierten Knotens der eines Kandidaten war, dann subtrahiere vom unnumerierten Nachbarknoten eine Einheit vom aktuellen Grad.

6. *Überprüfe den Zustand des Nachbarknotens:*
   Wenn der Nachbarknoten schon ein wartender Knoten ist, dann GOTO 4:

7. *Ändere den Zustand des Nachbarknotens: und aktualisiere den Grad und den Zustand der Nachbarknoten diese Knotens:* Der Nachbarknoten wird ab nun zu den wartenden Knoten gezählt und der aktuelle Grad um eine Einheit verringert.

8. *Aktualisiere die Zustände und aktuellen Grade der Nachbarknoten des neuen wartenden Knotens:*
   Jedem unnumerierten Nachbarknoten des neuen wartenden Knotens wird eine Einheit vom aktuellen Grad abgezogen. Außerdem wird der Knoten zum Kandidaten, wenn er nicht schon ein wartender Knoten oder ein Kandidat war.

9. *Ende der Schleife über alle unnumerierten Nachbarpunkte des neunumerierten Punktes:*
   GOTO 4:

10. *Überprüfung auf neuzunumerierende Knoten:*
    Ist die Gruppe der Kandidaten leer, dann GOTO 13:

11. *Wahl des neuzunumerierenden Knotens:*
    Suche jenen Knoten aus der Gruppe der Kandidaten der den geringsten aktuellen Grad aufweist.

12. *Führe die Neunumerierung durch und bereinige der Gruppen:*
    Erhöhe den Numerierungszeiger: `num=num+1`

```
Numeriere den gefundenen Knoten mit num.
Lösche den Knoten aus der Gruppe der Kandidaten und wenn
notwendig auch aus der Gruppe der wartenden Knoten.
GOTO [4:]
```

13. *Überprüfe ob alle Knoten numeriert sind:*
`IF num=n GOTO [Ende:]` 'O.K. ENDE'

14. *Suche neuen Startknoten, da mehrere unabhängige Teilgraphen vor liegen:*
`GOTO [3:]`

Abb. 4.34 veranschaulicht die Tabellen, wobei der Grad und der Zustand der Knoten laufend aktualisiert werden. In der Spalte des Zustandes können vier Situationen auftreten:

– Knoten noch unberührt,

W Knoten ist Mitglied der Gruppe der wartenden Knoten und der Gruppe der Kandidaten,

K Knoten ist Mitglied der Gruppe der Kandidaten,

Z Platznummer des Knotens (Z=1,...,n) .

Da immer wieder alle Kandidaten nach dem geringsten Grad durchsucht werden, empfiehlt es sich, das rasche Auffinden dieser Knoten zu ermöglichen. Durch eine gekettete Liste über alle wartende Knoten kann dies bewerkstelligt werden (Abb. 4.35). Zur Anzahl der Rechenoperationen kann folgendes gesagt werden. Jeder Knoten steht maximal zweimal im Mittelpunkt der Berechnungen. Einmal wenn der Knoten zum wartenden Knoten erhoben wird, müssen alle Nachbarknoten besucht werden. Ein zweites Mal ist dieser Aufwand erforderlich, wenn der Knoten numeriert wird. Da auch nichtwartende Knoten zur Numerierung herangezogen werden, entfällt in solchen Fällen der erste Anteil. Als Obergrenze der mögliche Operationen ist somit maximal zweimal die Anzahl der Kanten festzusetzen. Das Aufsuchen des nächsten zu numerierenden Knoten erfordert das Durchsuchen aller Kandidaten.

Existieren im Graphen Knoten, die nicht in die Neunumerierung miteinbezogen werden sollen (Unbekannte mit vorgegebenen Werten, z.B. Festpunkte), so muß nicht notwendigerweise ein Untergraph ohne diese Knoten erstellt werden um den Algorithmus 4.6 anzuwenden. Betrachtet man die nichtzunumerierden Knoten als schon numerierte Knoten. Wenn ein Knoten numeriert ist, verlieren die Nachbarknoten eine Einheit im Grad, da die nichtzunumerierenden Knoten keine wartenden Knoten werden können. Um also auf sehr einfach Art zu der gewünschten Ausgangslage zu gelangen, muß eine symbolische Neunumerierung der nichtzunumerierenden Knoten durchgeführt werden, wobei von allen Nachbarknoten dieser Knoten eine Einheit im Grad abgezogen wird, diese Knoten jedoch nicht zu wartenden Knoten oder Kandidaten werden.

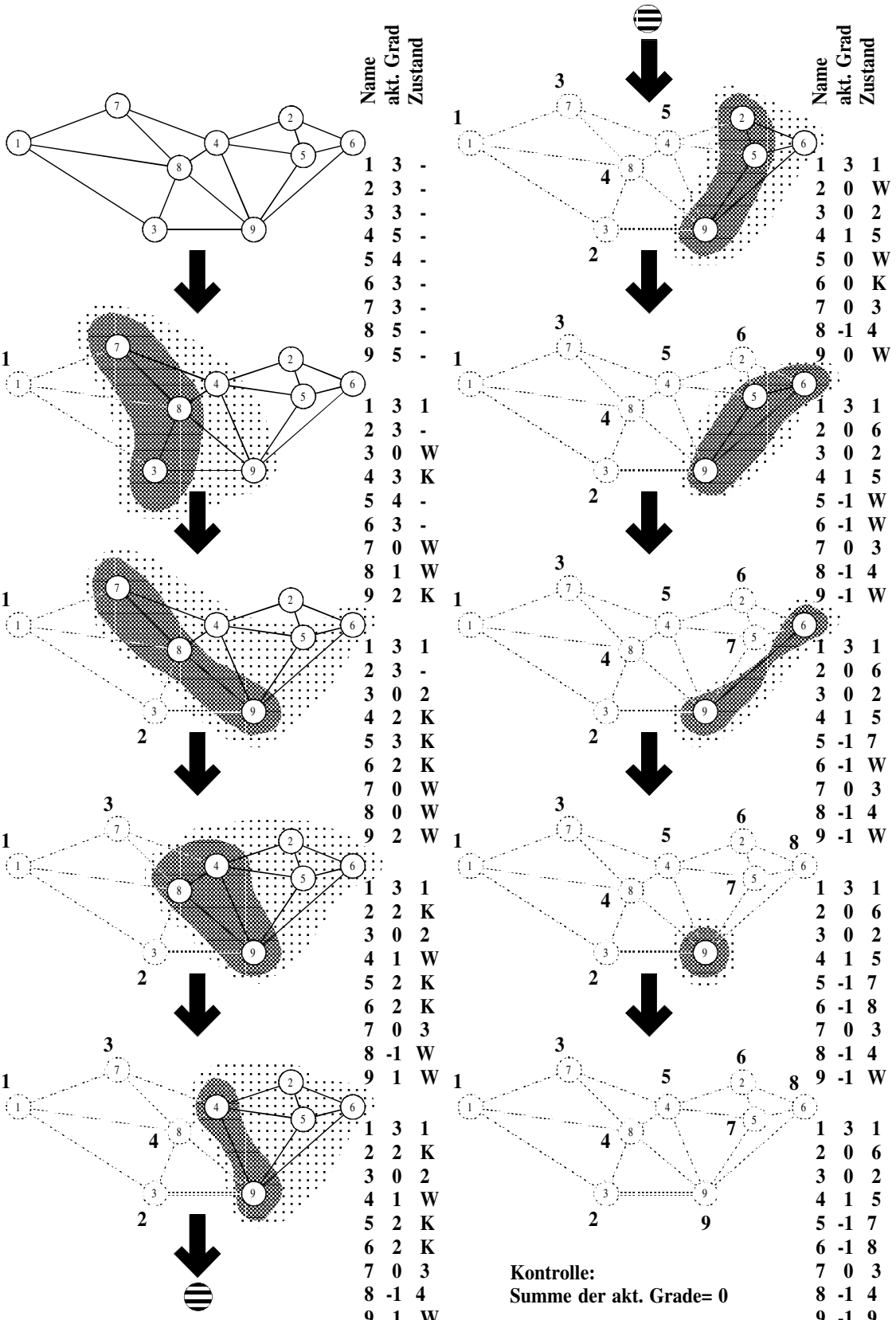


Abbildung 4.34: Aktualisierte Tabellen während des modifizierten Umordnungsverfahren nach Snay (Algorithmus 4.6).

Knotentabelle

Name	akt. Grad	Zustand	Zeiger
1	3	1	1
2	3	0	4
3	0	-1	7
4	3	-2	10
5	4	0	15
.	.	.	.
.	.	.	.
.	.	.	.

-2 = Kandidat
-1 = wartender Knoten

Kanten-tabelle

1	9
2	8
3	7
4	4
5	5
6	6
7	1
8	8
9	9
10	7
11	8
12	9
13	5
14	2
15	

Abbildung 4.35: Empfohlene Datenstruktur der Tabellen für Algorithmus 4.6 (Zustand nach Numerierung des Knotens 3; siehe Abb. 4.34)

4.4.4 Hüllenorientierte Umordnungsalgorithmen für heterogene Knoten

Alle bisher behandelten Verfahren beziehen sich auf **homogene** Knoten, das bedeutet, daß alle Knoten gleiche Wertigkeit aufweisen. Bei geodätischen Anwendungen sind dafür typische Beispiele (Höhen- und Schwerenetze, 3D-Netze). Die Anwendung dieser Algorithmen auf Netze mit ungleichwertigen Knoten (Lagenetze mit Orientierungen und Maßstäben, 3D-Netze mit zusätzlichen Nivellementbeobachtungen, Berücksichtigung von Randbedingungen bei finiten Elementen) kann auf verschiedene Art erfolgen. Zum Einen durch Vernachlässigung der Verschiedenheit der Knoten und damit einer näherungsweisen aber raschen Berechnung. Soll die Umordnung streng erfolgen, so sind die mehrwertigen Knoten in mehrere gleichwertige Knoten aufzuspalten und durch Einfügen aller mögliche Kanten zwischen den aufgespaltenen Knoten zu verknüpfen. Die wegführenden Kanten müssen ebenfalls aufgespalten werden.

Die Abbildungen 4.36 bis 4.38 zeigen die Situation in einem geodätischen Lagenetz auf. Zur Bestimmung der unbekannten Lagekoordinaten der Punkte *PP1*, *PP2*, *EP1* und *EP2* werden zwischen diesen Punkten und den lagemäßig bekannten Punkte *TP1*,

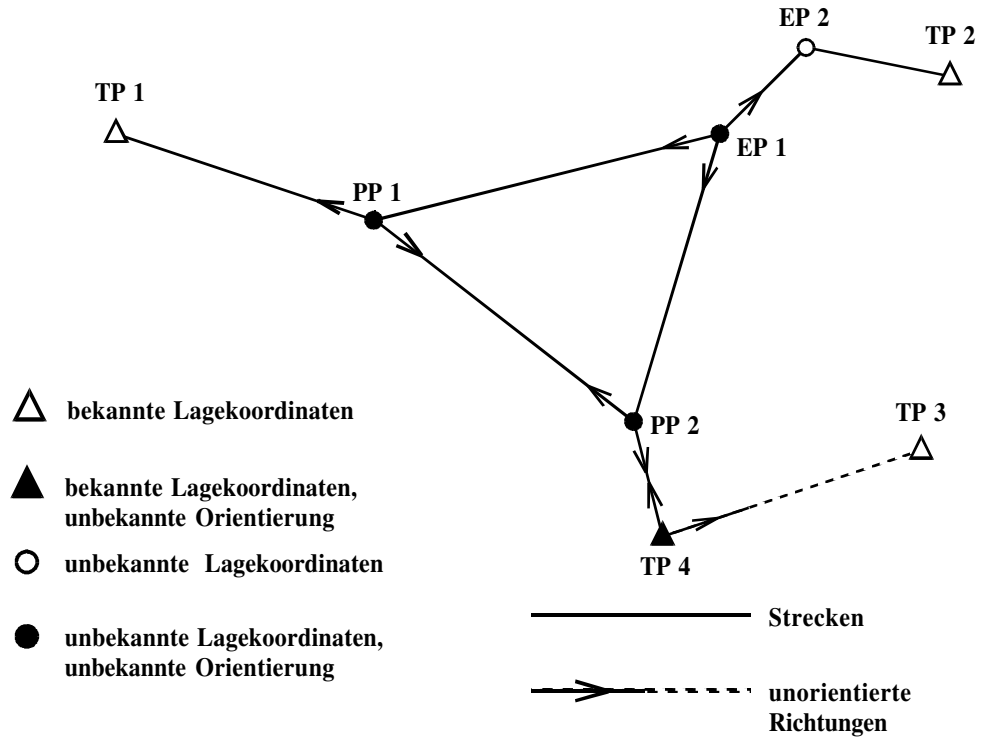


Abbildung 4.36: Kleines geodätisches Lagenetz

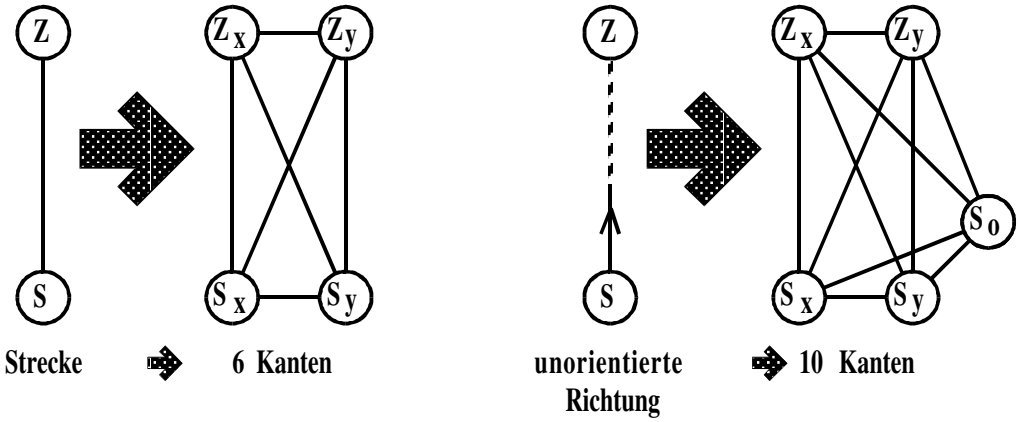


Abbildung 4.37: Aufspaltung einer Strecke bzw. unorientierten Richtung.

TP2, *TP3* und *TP4* Strecken und unorientierte Richtungen gemessen. Die mehrwertigen Knoten des kleinen Lagenetzes (Abb. 4.36) werden aufgespalten, wobei zwei die in Abb. 4.37 dargestellten typischen Situationen auftreten. Eine einfache Streckenverbindung erzeugt bei der Aufspaltung sechs Kanten. Zur Darstellung der Verbindungen einer unorientierten Richtung werden zehn Kanten benötigt. Die Verknüpfungen des kleinen Lagenetzes nach der Aufspaltung aller mehrwertigen Knoten ist in Abb. 4.38 veranschaulicht, woraus sofort die stark erhöhte Anzahl der Kanten ersichtlich wird.

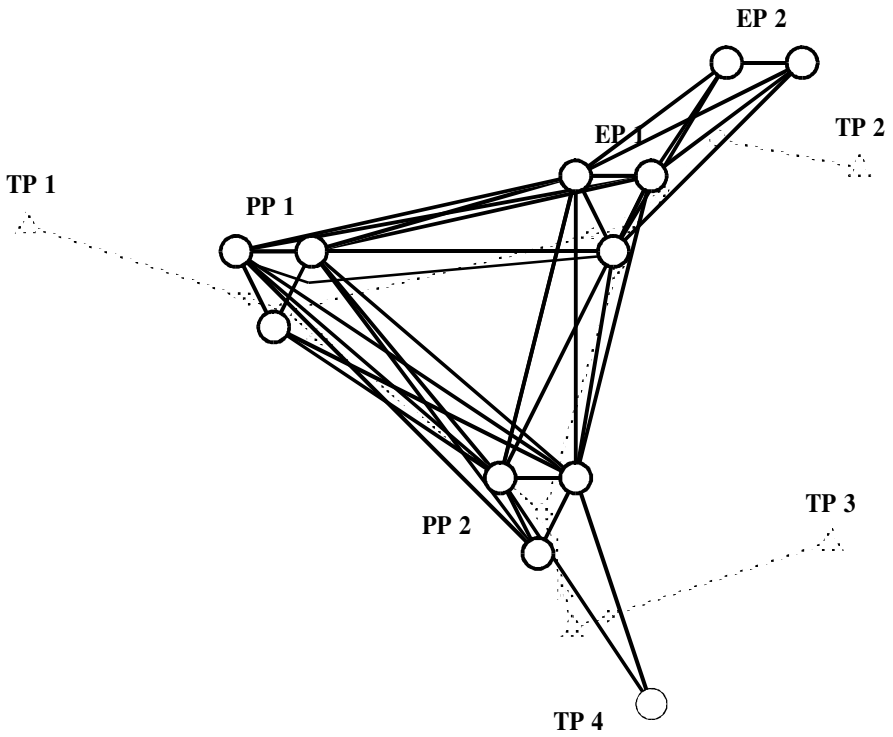


Abbildung 4.38: Verknüpfungen im kleinen Lagenetz;
Darstellung mit einwertigen Knoten.

Da sich die Anzahl der notwendigen Operationen zur Umordnung proportional der Anzahl der Kanten verhält, sind lange Rechenzeiten zu erwarten. Wenn beachtet wird, daß sich bestimmte Knoten während der Umordnung gleich verhalten und auch der Einfluß zu anderen Knoten synchron vor sich geht, können Vereinfachung durchgeführt werden.

Zunächst verallgemeinern wir die Situation auf viele Knoten mit sehr unterschiedlicher Wertigkeit. Die Einführung des Begriffs **Cluster** für eine stark verknüpfte Knotengruppe mit bestimmten Eigenschaften ist dabei dienlich. Ein Cluster ist eine Gruppe von Knoten, die dadurch gekennzeichnet ist, daß zwischen den Knoten im Cluster alle möglichen Kanten existieren (Innenkanten) und daß alle Knoten im Cluster die gleichen Nachbarknoten aufweisen (Außenkanten).

Eine Vereinfachung erreicht man nun, indem alle Knoten eines Clusters ($c \ldots$ Anzahl der Knoten im Cluster) zu einem mehrwertigen (c-wertigen) Knoten (z.B. Lagekoordinaten eines Punktes (2-wertig)) zusammengefaßt werden. Die Kanten von Knoten außerhalb des Clusters zu allen Knoten im Cluster (Außenkanten) werden ebenfalls zu mehrwertigen Kanten zusammengezogen. Eine Kante kann nun zwei unterschiedliche Wertigkeiten aufweisen, wobei die Wertigkeit des Zielknotens ausschlagegebend ist. Betrachtet man eine Kante zwischen einem Knoten K_1 (2-wertig) und einem Knoten K_2 (4-wertig), so weist sie die Wertigkeit 2 auf, wenn sie von K_1 nach K_2 betrachtet wird. In Richtung von K_2 nach K_1 beträgt die Wertigkeit 4. Da alle Kanten innerhalb des Clusters (Innenkanten) eliminiert werden, müssen sie bei der Berechnung zusätzlich

beachtet werden. Da dabei $c-1$ Kanten unberücksichtigt sind muß dieser Wert zum Grad eines Knotens hinzugezählt werden. Der Grad eines Knotens ergibt sich also aus der Summe der Wertigkeiten der Nachbarknoten plus der eigenen Wertigkeit minus eine Einheit.

Unter Beachtung der angeführten Zusammenhänge ist es möglich den Grad eines mehrwertigen Knotens immer aktuelle zu halten, indem man, wenn mehrwertige (c-wertig) Knoten zu wartenden Knoten erklärt werden, von den Nachbarknoten c Einheiten in Grad abzieht, beziehungsweise den Grad des neuen wartenden Knotens um $c-1$ Einheiten verringert. Kommt ein mehrwertiger Knoten zur Numerierung, so werden alle Komponenten mit den nächsten freien Platznummern markiert.

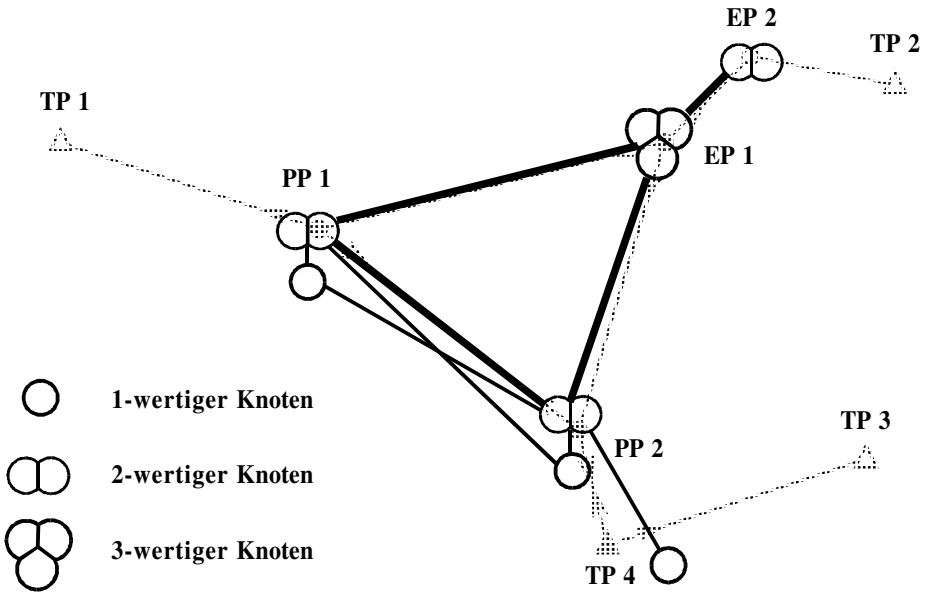


Abbildung 4.39: Verknüpfungen im kleinen Lagenetz; Darstellung mit mehrwertigen Knoten.

Die Anzahl der Kanten und Knoten wird durch diese Zusammenfassung stark reduziert, da pro benachbartem Cluster nur mehr eine Kanten existiert. Abb. 4.39 zeigt die Situation für das kleine Lagenetz auf.

Wenn man die gewonnen Erkenntnisse in den modifizierten hüllenorientierten Algorithmus 4.6 einbaut, so entsteht die nachfolgende Rechenvorschrift.

Algorithmus 4.7 : Hüllenorientierter Umordnungsalgorithmus für heterogene Knoten

Aufgabenstellung: Suche eine neue Reihenfolge der Knotennummern eines Graphen mit Knoten unterschiedlicher Wertigkeiten, sodaß die Verknüpfungsmatrix durch eine hüllenorientierte Speicherung angenähert werden kann.

Schnittstellen:

Eingabe: `graph` ... Graph mit Knotennummern 1 bis n
- `knoten(n)` ... Knotentabelle
- `wertigkeiten(n)` ... Wertigkeiten der Knoten
- `kanten(m)` ... Kantentabelle
- `n` ... Anzahl der Knoten
- `m` ... Anzahl der Kanten

Ausgabe: `platz(n)` ... neue Platznummern jedes Knotens

Hilfsfelder: `akt_grad(n)`

Hilfsproceduren:
`STARTKNOTEN(graph,start)` ... siehe Algorithmus 4.2

Lösungsweg: Pseudocode

1. *Initialisierung der Platznummern und des Numerierungszeigers (num):*
   `platz(1:n) = 0 ; num = 0 .`
2. `Stelle den Grad Summe der Wertigkeiten der Nachbarknoten plus Wertigkeit des Knoten vermindert um Eins) jedes Knotens fest und trage die Zahl (aktueller Grad) in eine Liste ein.`
   Merke die Summe alle Wertigkeiten: `sumwert = SUM(wert(1:n)).`
3. *Wähle einen unnumerierten Knoten als Startknoten:*
   Führe Algorithmus 4.2 aus:
   `STARTKNOTEN(graph,start)`
   Merke den Knoten und dessen Wertigkeit: `wert = wertigkeit(start)`
   Numeriere die Variablen des Startknotens mit num+1,...,num+wert:
   `DO i = 1 TO wert; platz(start+i-1) = num+i; END DO i`
   Erhöhe die Zeiger: `num=num+wert. Der frühere Zustand dieses Knotens ist gleichzusetzen mit dem eines Kandidaten. Initialisiere die Gruppe der wartenden Knoten und die Gruppe der Kandidaten.`
4. *Schleife über alle Nachbarknoten des neunumerierten Knotens:*
   `Suche der Reihe nach alle unnumerierten Nachbarknoten des zuletzt numerierten Knotens.`
   `Wenn kein Knoten mehr gefunden wird dann GOTO 10:`
5. *Aktualisiere den Grad der Nachbarknoten auf Grund des früheren Zustandes des neunumerierten Knotens:*
   `Wenn der frühere Zustand des neunumerierten Knotens der`

```
eines Kandidaten war, dann subtrahiere vom unnumerierten Nach-
barknoten die Wertigkeit des neunumerierten Knotens vom
aktuellen Grad.
```

6. *Überprüfe den Zustand des Nachbarknotens:*
   `Wenn der Nachbarknoten schon ein wartender Knoten ist, dann GOTO [4:]`
7. *Ändere den Zustand des Nachbarknotens und aktualisiere den Grad und den Zustand der Nachbarknoten diese Knotens:*
   `Der Nachbarknoten wird ab nun zu den wartenden Knoten gezählt und der aktuelle Grad wird um den Betrag der eigene Wertigkeit verringert.`
8. *Aktualisiere die Zustände und aktuellen Grade der Nachbar- knoten des neuen wartenden Knotens:* `Jedem unnumerierten Nachbarknoten des neuen wartenden Knotens wird die Wertigkeit des neuen wartenden Knotens vom aktuellen Grad abgezogen. Außerdem wird der Knoten zum Kandidaten, wenn er nicht schon ein wartender Knoten oder ein Kandidat war.`
9. *Ende der Schleife über alle unnumerierten Nachbarpunkte des neunumerierten Punktes:*
   `GOTO [4:]`
10. *Überprüfung auf neuzunumerierende Knoten:*
    `Ist die Gruppe der Kandidaten leer, dann GOTO [13:]`
11. *Wahl des neuzunumerierenden Knotens:*
    `Suche jenen Knoten aus der Gruppe der Kandidaten der den geringsten aktuellen Grad aufweist (min_grad).`
12. *Führe die Neunumerierung durch und bereinige der Gruppen:*
    Merke den Knoten und dessen Wertigkeit:
    `wert = wertigkeit(min_grad)`
    Numeriere die Variablen des Startknotens mit num+1,...,num+wert:
    `DO i = 1 TO wert; platz(min_grad+i-1) = num+i; END DO i`
    Erhöhe die Zeiger: `num=num+wert.`
    `Lösche den Knoten aus der Gruppe der Kandidaten und wenn notwendig auch aus der Gruppe der wartenden Knoten.`
    `GOTO [4:]`
13. *Überprüfe ob alle Knoten numeriert sind:*
    `IF num=n GOTO [Ende:]` 'O.K. ENDE'
14. *Suche neuen Startknoten, da mehrere unabhängige Teilgraphen vor liegen:*
    `GOTO [3:]`

Die Datenorganisation, die Behandlung von Festpunkten und die Suche nach dem Startknoten kann analog zu früher durchgeführt werden (siehe Kap. 4.4.3). Die Abb. 4.40 dient zum leichteren Verständnis des aufgezeigten Verfahrens und stellt die Durchführung am Beispiel des kleinen Lagenetzes (Abb. 4.36) dar. Das Ergebnis der Umnumerierung (siehe Abb. 4.41) erweist sich als optimal, da bei der Lösung kein Füllelement auftritt.

Name (Art)	Wertigkeit	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand	akt. Grad Zustand
TP 4 (Ori)	1	2 -	2 1	2 1	2 1	2 1	2 1	2 1	2 1	2 1
PP 1 (Koo,	2	8 -	6 K	3 W	2 W	2 4,5	2 4,5	2 4,5	2 4,5	2 4,5
Ori)	1	4 -	2 K	2 2	2 2	2 2	2 2	2 2	2 2	2 2
PP 2 (Koo,	2	9 -	6 W	3 W	2 W	-1 W	-16,7	-1 6,7	-1 6,7	
Ori)	1	4 -	2 K	0 K	0 3	0 3	0 3	0 3	0 3	
EP 1 (Koo,Ori)	3	8 -	6 K	4 K	4 K	1 W	1 W	1 8,9,10	1 8,9,10	
EP 2 (Koo)	2	4 -	4 -	4 -	4 -	1 K	1 K	-1 W	-1 11,12	

Kontrolle: Summe der akt. Grad - Summe der Wertigkeiten + Anzahl der Knoten = 0

Abbildung 4.40: Anwendung des Algorithmus 4.7 auf das kleine Lagenetz von Abb. 4.36.

	TP 4 (Ori)	PP 1 (Ori)	PP 2 (Ori)	PP 1 (x-Koo)	PP 1 (y-Koo)	PP 2 (x-Koo)	PP 2 (y-Koo)	EP 1 (x-Koo)	EP 1 (y-Koo)	EP 1 (Ori)	EP 2 (x-Koo)	EP 2 (y-Koo)
TP 4 (Ori)	X					X	X					
PP 1 (Ori)		X		X	X	X	X					
PP 2 (Ori)			X	X	X	X	X					
PP 1 (x-Koo)				X	X	X	X	X	X	X		
PP 1 (y-Koo)					X	X	X	X	X	X		
PP 2 (x-Koo)						X	X	X	X	X		
PP 2 (y-Koo)							X	X	X	X		
EP 1 (x-Koo)								X	X	X	X	X
EP 1 (y-Koo)									X	X	X	X
EP 1 (Ori)										X	X	X
EP 2 (x-Koo)											X	X
EP 2 (y-Koo)												X

optimale Numerierung (kein Füllelement)

Abbildung 4.41: Umnumerierte Verknüpfungsmatrix bei Anwendung des Algorithmus 4.7 auf das kleine Lagenetz von Abb. 4.36.

Literatur:

[19] CUTHILL E. (1972): Several Strategies for Reducing the Bandwidth of Matrices. ROSE D.J. and R.A. WILLOUGHBY (1972): (Editors) 'Sparse Matrices and their Applications', Academic Press, New York-London.

[35] GIBBS N.E. (1976): Algorithm 509: A Hybrid Profile Reduction Algorithm. ACM Transactions on Mathematical Software Vol. 2, No. 4, pp. 378-387.

[36] GIBBS N.E., W.G. POOLE, P.K. STOCKMEYER (1976): A Comparison of Several Bandwidth and Profile Reduction Algorithms. ACM Transactions on Mathematical Software Vol. 2, No. 4, pp. 322-330.

[37] GIBBS N.E., W.G. POOLE and P.K. STOCKMEYER (1976): An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. SIAM Numerical Analysis Vol. 13, No. 2, pp. 236-250.

[46] KING I.P. (1970): An Automatic Reordering Scheme for Simultaneous Equations Derived from Network Problems. International Journal for Numerical Methods in Engineering, Vol. 2, pp. 523-533.

[80] SNAY Richard A. (1976): Reducing the Profile of Sparse Symmetric Matrices. NOAA Technical Memorandum NOS NGS-4.

4.5 Direkte Methoden bei schwach besetzten Systemen

In den einführenden Darstellungen zu diesem Kapitel 4 wird die Entstehung von Füllelementen beim Eliminationsprozeß auf zwei Arten demonstriert. Die angegebenen Eliminationsregel 4.1 wird mit Hilfe der Graphendarstellung erarbeitet. Die Geschehnisse während des Eliminationsprozeßes werden stellvertretend an Hand eines Austausschrittes demonstriert. In analoger Weise kann das Entstehen von Füllelementen bei verschiedenen Eliminationsverfahren studiert werden. Wegen der hauptsächlichen Benützung des Verfahrens von Cholesky bei symmetrischen, positiv definiten Systemen wird hier nur auf dieses Verfahren näher eingegangen.

Entscheidend sind dabei die zur Reduktion dienenden Formeln (2.76)

$$\begin{aligned} r_{ij} &= \left(n_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj} \right) / r_{ii} \qquad && i = 1, \ldots, n \\ & && j = i+1, \ldots, n \\ r_{ii} &= \sqrt{n_{ii} - \sum_{k=1}^{i-1} r_{ki}^2} && i = 1, \ldots, n \end{aligned} \tag{4.18}$$

die im Abschnitt 2.4.1 erarbeitet und auch graphisch dargestellt sind (siehe Abb. 2.3 und Abb. 2.3 An Hand einer symbolischen Auflösung, das bedeutet, es interessiert nur, ob das errechnete Element Null ist, Null bleibt oder ungleich Null wird, kann die Situation nach der Auflösung ermittelt werden. Analysiert man die erste Formel von (4.18), so können zwei Bedingungen festgelegt werden, wovon zumindest eine erfüllt sein muß, wenn das reduzierte Glied r_{ij} ungleich Null ist.

- $n_{ij} \neq 0$
- $\sum_{k=1}^{i-1} r_{ki} r_{kj} \neq 0$

Die erste Bedingung besagt, daß überall, wo vor der Reduktion ein Element ungleich Null vorhanden ist, auch nach der Reduktion ein Element steht. Die zweite Bedingung bezieht sich auf das Inprodukt der reduzierten Spalte i mit der reduzierten Spalte j. Ein Verschwinden kann dabei zwei Ursachen aufweisen. Einerseits kann eine Auslöschung auftreten, wenn die reduzierten Spalten i und j zwar voll oder teilweise besetzt sind, sich aber eine Null errechnet, die reduzierten Spalten also orthogonal aufeinander stehen. Diese *'errechneten Nullen'* können durch Vorinformationen bei der Modellbildung eventuell beachtet werden. Die andere Ursache, der wir das Hauptaugenmerk zuwenden, für das Auftreten einer Null beim Inprodukt liegt darin, daß kein Paar von Elementen $r_{ki} r_{kj}$ vorhanden ist, wo nicht ein Element gleich Null ist.

Treten in der Spalte oberhalb des zu reduzierenden Gliedes nur Nullen auf und ist das zu reduzierende Glied selbst null, wie dies bei band- oder hüllenorientierter Speicherung außerhalb des besetzten Bereichs der Fall ist, so können keine Füllelemente entstehen.

4.5.1 Auflösung bei hüllenorientierter Speicherung

Stellvertretend für die Auflösung von umgeordneten kompakt gespeicherten, schwach besetzten Systemen wird hier die hüllenorientierte Speicherung behandelt. Die im Abschnitt 4.4.3 vorgeschlagene spaltenweise Ablage des besetzten Teils der oberen Dreiecksmatrix wird angewendet (Abb. 4.31). Zur Adressierung werden die Speicheradressen der Diagonalelemente in einem Index mitgeführt. Die Rechenvorschriften für die wichtigsten Operationen sind neben der Abb. 4.31 angegeben. Bei allen Anwendungen ist darauf zu achten, daß der spaltenweise Zugriff sehr einfach und rasch realisierbar ist. Eine wichtige Funktion bildet dabei die Rückrechnung des Zeilenindex des ersten gespeicherten Elementes einer vorgegeben Spalte. Die Berechnung der Adresse jedes beliebigen Elementes und die Überprüfung auf die Zugehörigkeit zum besetzten (gespeicherten) Gebiet stellen zwei weitere Hauptoperationen dar.

Da oft auf den $(j-1)$-ten Index zugegriffen wird, empfiehlt es sich zur Vereinheitlichung des Ablaufs die Matrix um eine fiktive Nullspalte zu erweitern. Diese leere Spalte wird durch Einführung eines Nullelementes als nullter Index bewerkstelligt. Damit erreicht man, daß die Bearbeitung der ersten Spalte, wo ein Rückgriff auf die Adresse des Diagonalelementes der nullten Spalte erforderlich ist, gleich wie alle anderen Spalten durchzuführen ist.

Legt man die dargestellte Abspeicherungsform zu Grunde, so kann der Algorithmus 2.1 bzw. 2.2 in zwei Punkten modifiziert werden, um eine effiziente Berechnung zu gewährleisten. Zum Einen werden nur die Elemente innerhalb der Hülle reduziert, da alle Nullelemente außerhalb der Hülle durch die Reduktion unbeeinflußt bleiben. Durch die Position des ersten Nichtnullelementes in der zu reduzierenden Spalte j können die zur Reduktion benötigten Spalten eingeschränkt werden. Zur Vereinfachung wird für die weitere Darstellung der Index ℓ_j für die Position (Zeilennummer) des ersten Nichtnullelementes in der Spalte j vergeben. Befindet sich das erste Nichtnullelement der Spalte j in der i-ten Zeile ($\ell_j = i$), so beginnen die Berechnungen zur Reduktion der Spalte j nun nicht bei der Verknüpfung mit der ersten Spalte, sondern erst bei der Spalte ℓ_j. Abbildung 4.42 veranschaulicht diese Einschränkung der Spalten.

Neben der alleinigen Berechnung der Elemente innerhalb der Hülle kann als zweite Modifikation die Inproduktbildung bei der Reduktion jedes Elementes r_{ij} vereinfacht werden. Die Berechnung muß nicht, wie in Formel (4.18) angegeben, beim ersten Element begonnen werden, sondern kann beim ersten Paar, wo beide Elemente $r_{ki}r_{kj}$ ungleich Null sind, gestartet werden (siehe Abb. 4.43). Der größere der beiden Werte ℓ_i und ℓ_j legt somit den Beginn fest.

Beachtet man diese beiden Modifikationen und benützt den Begriff der individuellen Bandbreite b_j (Anzahl der Nichtnullelemente in der Spalte j, $b_j = j - \ell_j + 1 = \texttt{IND}(j) - \texttt{IND}(j-1)$) so kann eine Rechenformel für die Anzahl der Rechenoperationen zur Lösung eines Schwach bestzten Systems erarbeitet werden. Die Hauptkomponente stellt dabei die Anzahl der Operationen für die Reduktion der Elemente r_{ij}, $i \neq j$ dar. Pro Reduktion einer Spalte ergeben sich dabei

$$\frac{1}{2}\,(b_j - 1)\,b_j - d_j \tag{4.19}$$

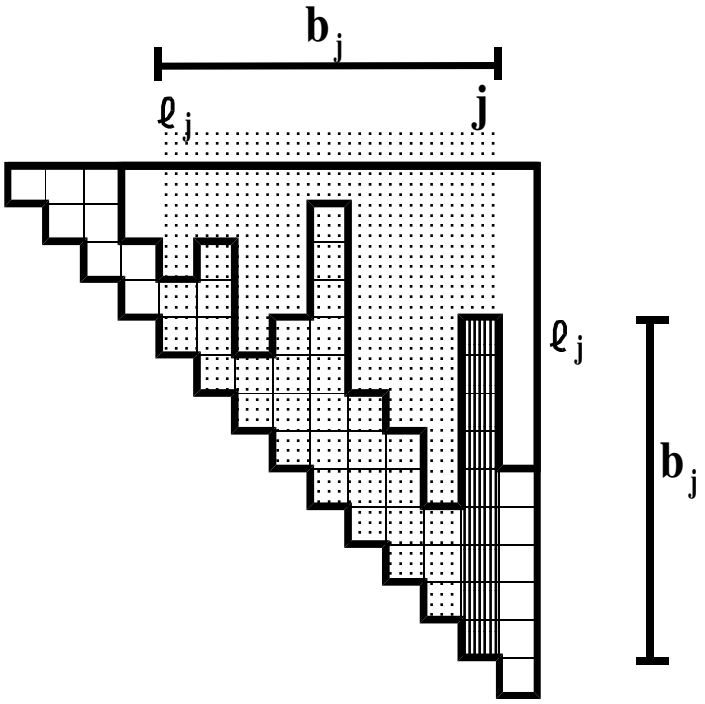


Abbildung 4.42: Eingrenzung der Spalten i, die zur Reduktion der j-ten Spalte erforderlich sind.

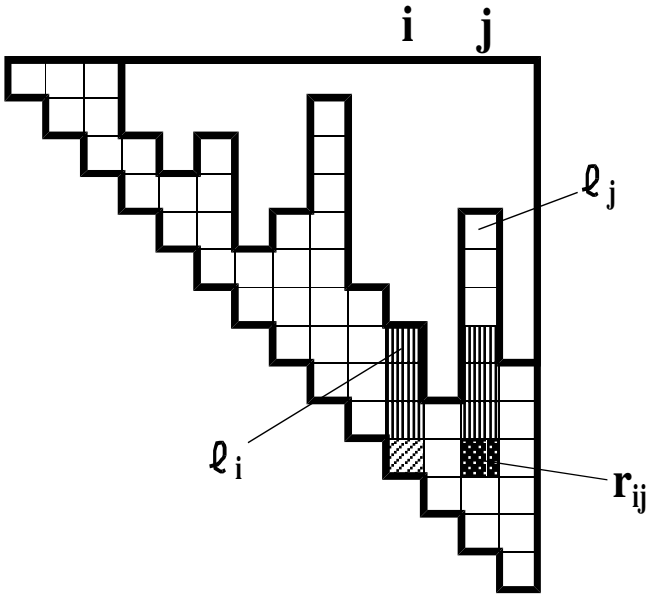


Abbildung 4.43: Eingrenzung der Spaltenhöhen zur Skalarproduktsberechnung bei der Reduktion des Gliedes r_{ij} .

Rechenschritte, wobei die Größe d_j die Bedeutung eines Defizites an Elementen (*'Elementdefizit'*) bei der Berechnung der Spalte j angibt. Abbildung 4.44 soll Klarheit über

die Bedeutung dieser Größe bringen. Die Berechnung ergibt sich aus

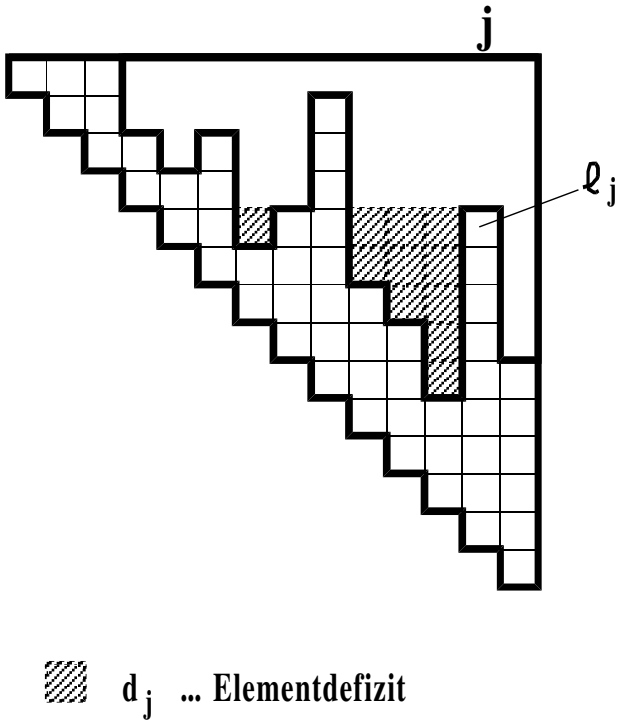


Abbildung 4.44: Veranschaulichung der Defizitelemente d_j der Spalte j. des Gliedes r_{ij} .

$$d_j = \sum_{i=\ell_j}^{j} (\ell_j - \ell_i)\, f_i \quad \begin{cases} f_i = 0 & \ell_i \geq \ell_j \\ f_i = 1 & \ell_i < \ell_j \ . \end{cases} \tag{4.20}$$

Zur Reduktion aller Elemente r_{ij}, $i \neq j$ ergeben sich somit

$$\sum_{j=1}^{n} \left\{ \frac{1}{2} (b_j - 1)\, b_j - d_j \right\} \tag{4.21}$$

Operationen. Die weiteren Komponenten bei der Ermittlung der Anzahl der Operationen zur Lösung bilden die Rechenoperationen zur Reduktion der Diagonalglieder r_{ii},

$$\sum_{j=1}^{n} (b_j - 1) \ , \tag{4.22}$$

zur Reduktion der rechten Seite

$$\frac{1}{2} (n+1)\, n - d_{n+1} \tag{4.23}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

und die dazu idente Anzahl an Operationen für das Rückwärtseinsetzen. Beachtet man, daß das Defizit der rechten Seite durch

$$d_{n+1} = \frac{1}{2} n\,(n+1) - \sum_{i=1}^{n} b_i \tag{4.24}$$

ausgrückt werden kann, so ergibt sich die Anzahl der Operationen für die gesamte Berechnung mit

$$\frac{1}{2}\sum_{j=1}^{n} b_j^2 + \frac{5}{2}\sum_{j=1}^{n} b_j - \sum_{j=1}^{n} d_j - n\ . \tag{4.25}$$

Der nachfolgende Algorithmus 4.8 nimmt Bedacht auf die beiden angeführten Modifikationen und versucht die notwendigen Abfragen und Adressrechnungen minimal zu halten.

Algorithmus 4.8 : Auflösung eines hüllenorientiert gespeicherten, symmetrischen, positiv definiten Gleichungssystems durch das Verfahren von Cholesky.

Aufgabenstellung: Gegeben ist ein symmetrisches, positiv definites Gleichungssystem $\boldsymbol{N} \in \mathbb{R}^{n\times n}$, $n \neq 0$ und der Konstantenvektor $\boldsymbol{b}$. Zur Speicherung der Matrix wird der Vektor `VEK()` verwendet, wo der besetzte Anteil der oberern Dreiecksmatrix spaltenweise gespeichert wird. Der um ein $(n+1)$-tes Glied erweiterte Konstantenvektor wird wie die $(n+1)$-te Spalte behandelt. Zur Adressierung wird ein Indexvektor `IND()` mitgeführt, der den Platz der Diagonalelemente (0 bis n+1) festlegt. Gesucht ist die Lösung des Systems, wenn das Gleichungs- system positiv definit ist. Andernfalls ist das Verfahren mit einer entsprechenden Meldung zu unterbrechen.

Schnittstellen:

Eingabe:	`VEK(1:IND(n))`	... Koeffizienten der symmetrischen Matrix
	`VEK(IND(n)+1:IND(n+1))`	... Konstantenvektor
	`IND(1:n+1)`	... Indexvektor
	`n`	... Anzahl der Zeilen und Spalten
Ausgabe:	`VEK(1:IND(n))`	... modifizierte Koeffizienten der symmetrischen Matrix
	`VEK(IND(n)+1:IND(n+1)-1)`	... Lösung des Gleichungssystems

Hilfsproceduren: —

Wolf-Dieter Schuh
ITGB, Uni Bonn

Ausgangssituation:

VEK	IND
n_{11}	0
n_{12}	1
n_{22}	3
n_{13}	6
n_{23}	8
n_{33}	$\vdots$
n_{34}	$Platz(n_{nn})$
n_{44}	$Platz(b_{n+1})$
$\vdots$	
b_1	
b_2	
$\vdots$	
b_n	

Endsituation:

VEK	IND
r_{11}	0
r_{12}	1
r_{22}	3
r_{13}	6
r_{23}	8
r_{33}	$\vdots$
r_{34}	$Platz(n_{nn})$
r_{44}	$Platz(x_{n+1})$
$\vdots$	
x_1	
x_2	
$\vdots$	
x_n	

Lösungsweg: Pseudocode

1. *Initialisiere den Spaltenindex, die Adresse des als zu bearbeitenden Elements und die Adresse des Konstantenvektors (Unbekanntenvektors) für die Choleskyreduktion bzw. das Vorwärtseinsetzen (spaltenweise Reduktion):* `j=0 ; adr_akt=0 ; adr_unb=IND(n) .`
2. *Bearbeite nächsten Spalte; rechne neue Adresse des Spalten- beginns und erhöhe den Spaltenindex:* `adr_j=IND(j) ; j=1+1`
   Initialisiere aufgrund des ersten Nichtnullementes in der Zeile j den Zeilenindex und errechne die Zeile, in der die Spalte j erstmals ungleich null ist: `i=j-IND(j)+adr_j, j_beg=i+1 .`
3. *Bearbeite nächstes Element; rechne neue Adresse des Zeilen- koeffizienten, erhöhe den Zeilenindex, errechne das erste Nichtnullelement von i und erhöhe die Adresse des zu be- arbeitenden Elements:*
   `adr_i=IND(i) ; i=i+1 ; i_beg=i-IND(i)+adr_i+1 ;`
   `adr_akt=adr_akt+1 .`
4. *Ermittle die Startzeile des inneren Produktes (entspricht dem Maximum von i-beg und j-beg) und beginne die Reduktion des Gliedes in der i-ten Zeile und j-ten Spalte:*
   ```
   k_beg=MAX(i_beg,j_beg)
   VEK(adr_akt)=VEK(adr_akt) -
   DOT_PRODUCT(VEK(adr_i+k_beg:adr_i+i-1),VEK(adr_j+k_beg:adr_j+i-1))
   ```
5. *Überprüfe ob zu reduzierende Glied ein Diagonalglied ist oder ist oder nicht und setze entsprechend fort:* `IF i=j GOTO 8:`
6. *Reduziertes Glied stellt kein Diagonalglied dar. Vervoll- ständige die Berechnung durch Berücksichtigung der Division:*
   `VEK(adr_akt) = VEK(adr_akt) / VEK(adr_i+i) .`

Wolf-Dieter Schuh
ITGB, Uni Bonn

7. *Beende entsprechend dem Zeilenindex den Vorgang der Reduktion bzw. des Vorwärtseinsetzens oder setze die Reduktion beim nächsten Glied dieser Spalte fort:*
   `IF i=n GOTO 10:`
   `GOTO 3:`
8. *Reduziertes Glied ist ein Diagonalglied. Überprüfe das Diagonalglied:*
   `IF VEK(adr_akt) < num_Null GOTO Ende:` 'KEINE POSITIV DEFINITE MATRIX'
9. *Vervollständige die Berechnung des Diagonalgliedes durch Be- rücksichtigung der Wurzel:*
   `VEK(adr_akt) = SQRT(VEK(adr_akt))`
   `GOTO 2:`
10. *Starte den Prozeß des Rückwärtseinsetzens, wobei wieder spaltenweise vorgegangen wird. Berechne die i-te Unbekannte:*
    `VEK(adr_unb+i) = VEK(adr_unb+i) / VEK(adr_i+i) .`
11. *Überprüfe ob alle Unbekannten bearbeitet wurden:*
    `IF i=1 GOTO Ende:` 'O.K. ENDE'
12. *Berücksichtige den Anteil der i-ten Unbekannten bei allen Gleichungen von i_beg bis i-1:*
    `VEK(adr_unb+i_beg:adr_unb+i-1) = VEK(adr_unb+i_beg:adr_unb+i-1)`
    `- VEK(adr_i+i_beg:adr_i+i-1) VEK(adr_unb+i) .`
13. *Verringere den Schleifenzähler, rechne die neue Adresse und Platz des ersten Elementes der nächstniederen Unbekannten:*
    `i=i-1 ; adr_i=IND(i-1), i_beg=i-IND(i)+adr_i+1 .`
    `GOTO 10:`

Da diese Aufgabenstellung eine Verallgemeinerung der bandorientierten Speicherung darstellt, können Bandmatrizen mit der gleichen Speicher anordnung bearbeitet werden. Der Index kann in analoger Form entweder gespeichert oder bei Bedarf durch die Speicherfunktion (siehe Abb. 4.29) errechnet werden. Eine Umarbeitung des Algorithmus auf die spezielle Speicherung von Band matrizen, wie sie in Abb. 4.29 mit den notwendigen Rechenvorschiften angeführt ist, wirft somit keine großen Probleme auf. Die Berechnung der notwendigen Operationen kann leicht von der allgemeineren Formulierung übernommen werden. Nimmt man an Stelle der individuellen Bandbreite einen konstanten Wert an, so ergibt sich die Berechnungsformel für die Anzahl der Operationen mit

$$\frac{1}{2}(n-b)\left(b^2+5b-2\right)+\frac{1}{6}b\,(b+1)\,((b+8) \tag{4.26}$$

da für

$$\begin{array}{lcll} \sum_{j=1}^{n} b_j^2 & = & (n-b)\,b^2+\sum_{j=1}^{n} j^2 & =(n-b)\,b^2+\frac{1}{6}b\,(b+1)\,((2b+1) \\ \sum_{j=1}^{n} b_j & = & (n-b)\,b+\sum_{j=1}^{n} j & =(n-b)\,b+\frac{1}{2}b\,(b+1) \\ \sum_{j=1}^{n} d_j & \sim & 0 & = \end{array} \tag{4.27}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

anzugeben ist. In der Literatur wird meist eine Näherungsformel angeführt, die dann verwendet werden kann, wenn die Anzahl der Unbekannten n groß gegenüber der Bandbreite b angenommen wird. Es ergibt sich somit eine obere Schanke für die Anzahl der Rechenoperationen zur Lösung einer Bandmatrix

$$\frac{1}{2}n\left(b^2+5b-2\right) \tag{4.28}$$

(vergleiche auch Näherungsformel bei *SCHWARZ (1988)[78]*, Seite 43, Formel (1.108) unter Beachtung, daß die Bandbreite dort mit $b = \ell_j - j$ definiert ist, hier jedoch mit $b = \ell_j - j + 1$ eingeführt ist).

Literatur:

[78] SCHWARZ Hans Rudolf (1988): Numerische Mathematik. Teubner, Stuttgart, 2. Auflage.

4.5.2 Inversion bei hüllenorientierter Speicherung

Usually the computation of the inverse of a matrix needs a huge effort, but is not necessary for the solving an equation system. However, if least squares adjustment techniques are applied then the inverse of the normal equations matrix holds important statistic information, namely the variances and covariances information of the estimated parameters. From this point of view it is often of interest to determine special parts of the inverse, especially the diagonal or block diagonal part, bearing in mind that the inverse of a sparse matrix is in general a full matrix. This section explains a method to compute a partial inverse, this means, to compute only these elements of the inverse where non-zero elements in the reduced equation system occurs. The base idea of this algorithm was published by *[44]*. This computation can be organized *in place* that means that no additional storage requirements occur.

The algorithm can be derived by blocking the matrices $\boldsymbol{N}$ and $\boldsymbol{R}$, respectively.

$$\begin{aligned}
\boldsymbol{N} &= \begin{bmatrix} \boldsymbol{N}_{11} & \boldsymbol{N}_{12} \\ (\boldsymbol{N}_{12})^T & \boldsymbol{N}_{22} \end{bmatrix} \\
\boldsymbol{R} &= \begin{bmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ & \boldsymbol{R}_{22} \end{bmatrix} \\
\boldsymbol{N}^{-1} &= \begin{bmatrix} \boldsymbol{N}_{11}^{(-1)} & \boldsymbol{N}_{12}^{(-1)} \\ \left(\boldsymbol{N}_{12}^{(-1)}\right)^T & \boldsymbol{N}_{22}^{(-1)} \end{bmatrix} .
\end{aligned} \tag{4.29}$$

The blocks $\boldsymbol{N}_{11}^{(-1)}$, $\boldsymbol{N}_{12}^{(-1)}$ and $\boldsymbol{N}_{22}^{(-1)}$ of the inverse matrix $\boldsymbol{N}^{-1}$ are marked by `'(-1)'` in contrast to the inverse `'-1'` of the block itself. These blocks can be computed from the blocks of $\boldsymbol{N}$ in different ways, e.g. for.

$$\boldsymbol{N}_{22}^{(-1)} = \left(\boldsymbol{N}_{22} - \boldsymbol{N}_{12}^T\boldsymbol{N}_{11}^{-1}\boldsymbol{N}_{12}\right)^{-1} \tag{4.30}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

$$\boldsymbol{N}_{12}^{(-1)} = -\boldsymbol{N}_{11}^{-1}\boldsymbol{N}_{12}\boldsymbol{N}_{22}^{(-1)} \tag{4.31}$$

$$\boldsymbol{N}_{11}^{(-1)} = \boldsymbol{N}_{11}^{-1} - \boldsymbol{N}_{12}^{(-1)}\boldsymbol{N}_{12}^{T}\boldsymbol{N}_{11}^{-1} \ . \tag{4.32}$$

This form is chosen, because the Cholesky's reduction in block matrix form shows a very similar behaviour, especially for the $\boldsymbol{R}_{22}$ term

$$\boldsymbol{R}_{22} = \left(\boldsymbol{N}_{22} - \boldsymbol{N}_{12}^{T}\boldsymbol{N}_{11}^{-1}\boldsymbol{N}_{12}\right)^{C} \tag{4.33}$$

$$\boldsymbol{R}_{12} = \left(\boldsymbol{R}_{11}^{T}\right)^{-1}\boldsymbol{N}_{12} \tag{4.34}$$

$$\boldsymbol{R}_{11} = \boldsymbol{N}_{11}^{C} \ , \tag{4.35}$$

where *'C'* means the Cholesky reduced of a matrix. Therefore, it is easy to express (cf. *[?]*, Ch. 2.4.2) the blocks of the inverse $\boldsymbol{N}_{11}^{(-1)}$, $\boldsymbol{N}_{12}^{(-1)}$ (4.30-4.32) by the Cholesky reduced blocks $\boldsymbol{R}_{11}$, $\boldsymbol{R}_{12}$, $\boldsymbol{R}_{22}$ (4.33-4.35),

$$\boldsymbol{N}_{22}^{(-1)} = \left(\boldsymbol{R}_{22}^{T}\boldsymbol{R}_{22}\right)^{-1} \tag{4.36}$$

$$\boldsymbol{N}_{12}^{(-1)} = -\boldsymbol{R}_{11}^{-1}\boldsymbol{R}_{12}\boldsymbol{N}_{22}^{(-1)} \tag{4.37}$$

$$\boldsymbol{N}_{11}^{(-1)} = \left(\boldsymbol{R}_{11}^{T}\boldsymbol{R}_{11}\right)^{-1} - \boldsymbol{N}_{12}^{(-1)}\boldsymbol{R}_{12}^{T}\boldsymbol{R}_{11}^{-1} \ . \tag{4.38}$$

This set of equations can be used to elaborate an recursive algorithm. Starting from the element $n_{nn}^{(-1)}$ in the lower right of the matrix in a row-by-row backspace approach the whole inverse matrix can be computed.

Eq. (4.39)-(4.41) summarize the scalar form of this algorithm.

$$n_{nn}^{(-1)} = \frac{1}{r_{nn}^{2}} \tag{4.39}$$

$$n_{ij}^{(-1)} = -\frac{1}{r_{ii}}\sum_{k=i+1}^{n} r_{ik}n_{kj}^{(-1)} \ , \qquad \begin{matrix} j=i+1,\ldots,n \\ i=n,\ldots,1 \end{matrix} \tag{4.40}$$

$$n_{ii}^{(-1)} = \frac{1}{r_{ii}^{2}} - \frac{1}{r_{ii}}\sum_{k=i+1}^{n} r_{ik}n_{ik}^{(-1)} \ , \ i=n,\ldots,1 \tag{4.41}$$

Again the main step of this algorithm the computation of the element $n_{ij}^{(-1)}$ (cf. Eq. (4.40)) is figured out graphically. Fig. 4.45 shows that the scalar product of the lower part of the j^{th} column of the inverse with the Cholesky reduced i^{th} row forms the major step. A row-to-row backspace strategy makes sure that all the necessary inverse element in the lower part are computed before.

Wolf-Dieter Schuh
ITGB, Uni Bonn

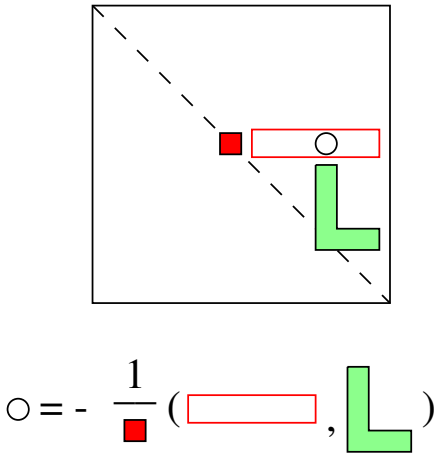


Abbildung 4.45: Graphical representation of eq. (4.40). Computation of the inverse element $n_{ij}^{(-1)}$ from Cholesky reduced matrix.

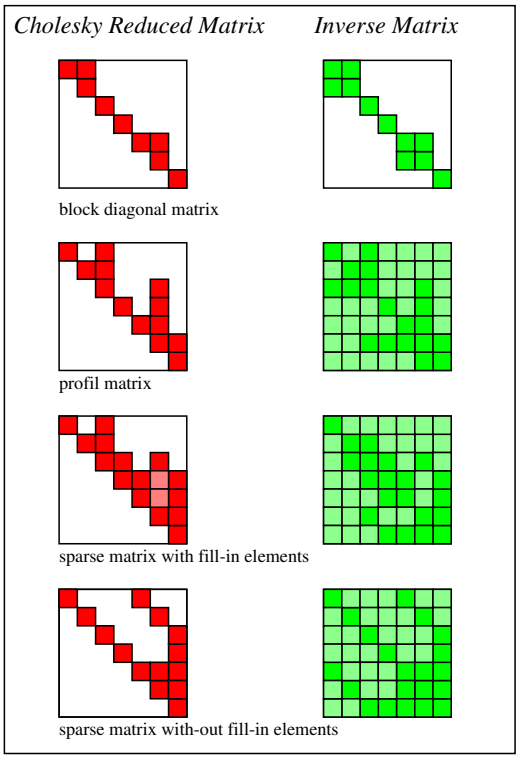


Abbildung 4.46: For basic examples on fill-in elements during Inversion from Cholesky reduced matrix. The darker elements in the Inverses can be computed separately.

It is now a well-known effect, that in general the inversion turns over a sparse system to a dense one. Only a block-diagonal structure is preserved (cf. Fig. 4.46). But if we are interested only in special elements of the inverse, e.g. the main diagonal elements, or all elements within a defined bandwidth then special algorithms exists to compute these elements in an efficient way. A special algorithm for profile structured matrices

was introduced by *[44]*. He shows, that all the inverse elements within the profile can be computed in a strict manner without any information of inverse elements outside of the profile. It's up to the fact that the non-computed elements within each column of the inverse corresponds exactly with the zero-elements within the Cholesky reduced row. Therefore, the scalar product (Eq. (4.40), see also Fig. 4.45) for all elements within the profile is not influenced by inverse element, that are positioned outside of the profile. This argument holds also for our special *'kite'* structure (cf. Fig. **??**). This means we can compute the inverse elements for all non-zero elements, denoted as (*'partial inverse'*), in a very efficient *'in place'* strategy. Fig. 4.47 illustrates the fact that all zero-elements

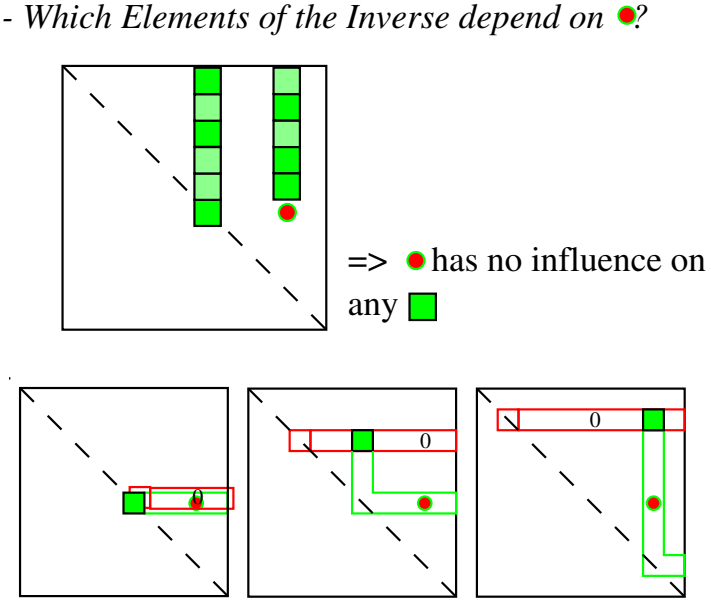


Abbildung 4.47: Computation of the partial inverse from the Cholesky reduced matrix. The inverse element $n_{ij}^{(-1)}$ (dot) influences the inverse values in the i^{th} and j^{th} columns. Since the factorization produces no fill-in elements (cf. **??**) the influence of the inverse element $n_{ij}^{(-1)}$ on all lighter inverse elements runs into zero products.

after Cholesky reduction (due to vanishing inner products, cf. Fig. **??**) do not influence the computation of the partial inverse of all non-zero elements, since the always run into zero products.

Literatur:

[44] HANSON R.H. (1978): A Posteriori Error Propagation. Second International Symposium on Problems Related to the Redefinition of North American Geodetic Networks. Proceedings, pp. 427-445. Teubner, Stuttgart, 2. Auflage.

4.6 Iterative Methoden bei schwach besetzten Systemen

Wolf-Dieter Schuh
ITGB, Uni Bonn

Kapitel 5

Lösung großer Gleichungssysteme

5.1 Prinzipien der Parallelverarbeitung

Wolf-Dieter Schuh
ITGB, Uni Bonn

5.2 Direkte parallele Methoden

Wolf-Dieter Schuh
ITGB, Uni Bonn

5.3 Iterative parallele Methoden

Wolf-Dieter Schuh
ITGB, Uni Bonn

Kapitel 6

Regelmäßige Strukturen

Bei der Verwendung von regelmäßig verteilten Daten stößt man auf Matrizen von ganz spezieller Natur. Bedingt durch gleiche sich wiederholende Koeffizienten in den einzelnen Zeilen, ergibt sich die jeweils nachfolgende Zeile aus einer Verschiebung der Vorgängerzeile um eine Spalte.

$$\mathbf{T} = \begin{bmatrix} t_0 & t_{-1} & t_{-2} & \dots & \dots & \dots & t_{-(n-2)} & t_{-(n-1)} \\ t_1 & t_0 & t_{-1} & \dots & \dots & \dots & t_{-(n-3)} & t_{-(n-2)} \\ t_2 & t_1 & t_0 & & & & t_{-(n-4)} & t_{-(n-3)} \\ \dots & \dots & & \dots & & & \dots & \dots \\ \dots & \dots & & & \dots & & \dots & \dots \\ \dots & \dots & & & & \dots & \dots & \dots \\ t_{n-2} & t_{n-3} & t_{n-4} & \dots & \dots & \dots & t_0 & t_{-1} \\ t_{n-1} & t_{n-2} & t_{n-3} & \dots & \dots & \dots & t_1 & t_0 \end{bmatrix} \tag{6.1}$$

Diese Matrix, benannt nach dem deutschen Mathematiker Otto Toeplitz (1881-1940), bildet die allgemeinste Form einer Gruppe von besonders effizient lösbaren Matrizen. Normalerweise werden die Elemente a_{ij} einer beliebigen Matrix $\mathbf{A}$ durch einen Zeilenindex i und einen Spaltenindex j eindeutig zugeordnet. In dieser speziellen Anordung ist durch die Differenz zwischen den Spaltenindex und dem Zeilenindex $k = j - i$ das Element eindeutig festgelegt, womit ein Index k zur Unterscheidung der Elemente ausreichend ist. Durch die Elemente t_0 bis $t_{-(n-1)}$ bzw. t_{n-1} also durch die Angabe einer Zeile und der korrespondierenden Spalte ist die gesamte Information der Matrix dargestellt. Es sei hier auch noch auf eine andere spezielle Eigenschaft von Toeplitz-Matrizen hingewiesen, die nicht sofort ins Auge sticht. Diese Matrizen ist persymmetrisch , also symmetrisch bezüglich der Diagonal Nordost-Südwest. Das Element a_{ij} ist somit gleich dem Element $a_{n-j+1,n-i+1}$. Analog zur Symmetrie vererbt sich diese Eigenschaft bei bestimmten Rechenoperationen z.B. bei der Inversion und kann somit zur Reduktion der Anzahl der Rechenoperationen beitragen.

Eine spezielle Form stellt die zirkulierende Toeplitz-Matrix, kurz zirkulierende Matrix, dar, wo die Zeilen durch zyklische Verschiebungen gebildet werden. Dies bedeutet, daß

der letzte Koeffizient der ersten Zeile wieder den ersten Koeffizient der zweiten Zeile bildet.

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & \dots & \dots & c_{n-2} & c_{n-1} \\ c_{n-1} & c_0 & c_1 & \dots & \dots & \dots & c_{n-3} & c_{n-2} \\ c_{n-2} & c_{n-1} & c_0 & & & & c_{n-4} & c_{n-3} \\ \dots & \dots & & \dots & & & \dots & \dots \\ \dots & \dots & & & \dots & & \dots & \dots \\ \dots & \dots & & & & \dots & \dots & \dots \\ c_2 & c_3 & c_4 & \dots & \dots & \dots & c_0 & c_1 \\ c_1 & c_2 & c_3 & \dots & \dots & \dots & c_{n-1} & c_0 \end{bmatrix} \tag{6.2}$$

Diese Matrix ist somit durch die Elemente c_0 bis c_{n-1}, also die Information über eine Zeile oder Spalte vollständig definiert. Ein Element der Zeile i und Spalte j kann mit Hilfe der Funktion $k = (j - i + n) \bmod n$ dem Elementen c_k zugeornet werden.

Entgegen der bisher verwendeten Numerierung beginnend von 1 bis n wird hier aus Gründen der einheitlichen Darstellung mit der vorhanden Literatur auf die Numerierung von 0 bis $n-1$ übergegangen. Das Hauptaugenmerk bei geodätischen Anwendungen wird auf symmetrische Toeplitz-Matrizen

$$\mathbf{T} = \begin{bmatrix} t_0 & t_1 & t_2 & \dots & \dots & \dots & t_{n-2} & t_{n-1} \\ t_1 & t_0 & t_1 & \dots & \dots & \dots & t_{n-3} & t_{n-2} \\ t_2 & t_1 & t_0 & & & & t_{n-4} & t_{n-3} \\ \dots & \dots & & \dots & & & \dots & \dots \\ \dots & \dots & & & \dots & & \dots & \dots \\ \dots & \dots & & & & \dots & \dots & \dots \\ t_{n-2} & t_{n-3} & t_{n-4} & \dots & \dots & \dots & t_0 & t_1 \\ t_{n-1} & t_{n-2} & t_{n-3} & \dots & \dots & \dots & t_1 & t_0 \end{bmatrix} \tag{6.3}$$

gelegt. Die Zuordung eines Elementes der Zeile i und Spalte j erfolgt durch die Funktion $k = |j - i|$. Die Elemente einer Zeile oder Spalte t_0 bis t_{n-1} genügen zur eindeutigen Festlegung. Einen Sonderfall bilden die symmetrischen zirkulierenden Toeplitz-Matrizen,

$$\mathbf{C} = \begin{bmatrix} c_0 & c_1 & c_2 & \dots & \dots & c_h & \dots & \dots & \dots & c_2 & c_1 \\ c_1 & c_0 & c_1 & \dots & \dots & \dots & c_h & \dots & \dots & c_3 & c_2 \\ c_2 & c_1 & c_0 & \dots & & & c_h & & & c_4 & c_3 \\ \dots & \dots & \dots & \dots & \dots & & & & \dots & \dots & \dots \\ \dots & \dots & & \dots & \dots & \dots & & & & c_h & \dots \\ c_h & \dots & & & \dots & \dots & \dots & & & \dots & c_h \\ \dots & c_h & & & & \dots & \dots & \dots & & \dots & \dots \\ \dots & \dots & c_h & & & & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & & \dots & & & & \dots & \dots & \dots & \dots \\ c_2 & c_3 & c_4 & \dots & c_h & \dots & \dots & \dots & \dots & c_0 & c_1 \\ c_1 & c_2 & c_3 & \dots & \dots & c_h & \dots & \dots & \dots & c_1 & c_0 \end{bmatrix} \tag{6.4}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

die durch Angabe einer halben Zeile oder Spalte, also die Elemente c_0 bis c_h mit $h = \text{int}(n/2)$, eindeutige definiert sind. Ein Element der Zeile i und Spalte j kann mit $k = |((j - i + h + n) \bmod n) - h|$ dem Element c_k zugeordnet werden.

Die Anwendungen der bisher vorgestellten regelmäßigen Strukturen sind auf eindimensionale Problemstellungen beschränkt. Die Rückführung des gesamten Informationsgehalts der zweidimensionalen Matrix auf maximal eine Zeile und Spalte und somit auf nur einen Index, deutet darauf hin, daß vor allem ortsunabhängige (homogene) Problemstellungen mit solchen Strukturen darstellbar sind. Der Übergang auf zunächst zweidimensionale Problemstellungen kann dadurch erfolgen, daß die skalaren Koeffizienten durch Matrizenblöcke ersetzt werden. Wir sprechen somit von **Block-Toeplitz-Matrizen** oder **Block-zirkulierenden Strukturen**. Je nachdem, ob in der hinzugefügten Dimension wieder regelmäßige Strukuren auftreten, können diese Blöcke im Inneren wieder spezielle Strukturen bilden. Beispiele die hier behandelt werden sind

- Block-Toeplitz-Matrizen mit beliebigen Blöcken,
- Block-Toeplitz-Matrizen mit Toeplitz-Blöcken,
- Block-Toeplitz-Matrizen mit zirkulierenden Blöcken,
- Block-zirkulierende Matrizen mit beliebigen Blöcken,
- Block-zirkulierende Matrizen mit Toeplitz-Blöcken und
- Block-zirkulierende Matrizen mit zirkulierenden Blöcken.

Die Erweiterung auf drei oder höher dimensionale Ausdehnungen fordert eine systematische Anwendung der im zweidimensionalen erarbeiteten Grundsätze. Durch stufenweise Verwendung der bekannten Werkzeuge kann die Komplexität reduziert werden, wobei vor allem zirkulierende Strukturen zur Reduktion der Koplexität genutzt werden können.

Bei höherdimensionalen Problemen kommt uns aber auch oftmals der Aufbau der Aufgabenstellung zur Hilfe. Vielfach können die Ansätze so gewählt werden, daß die Dimensionen trennbar, im Sinne des Kronecker-Produktes sind. Die gesamte Matrix $\mathbf{A}$ kann somit durch das Kronecker-Produkt

$$\mathbf{A} = \mathbf{G} \otimes \mathbf{H} = \begin{bmatrix} g_{11}\mathbf{H} & g_{12}\mathbf{H} & \dots & g_{1n}\mathbf{H} \\ g_{21}\mathbf{H} & g_{22}\mathbf{H} & \dots & g_{2n}\mathbf{H} \\ \dots & & & \dots \\ \dots & & & \dots \\ g_{m1}\mathbf{H} & g_{12}\mathbf{H} & \dots & g_{mn}\mathbf{H} \end{bmatrix} \tag{6.5}$$

dargestellt werden. Die Koeffizienten der beliebigen Matrix $\mathbf{G}$ dienen somit der Matrix $\mathbf{H}$ als Multiplikatoren. Begingt durch diesen speziellen Ansatz kann eine Reihe von Rechenregeln verwendet werden, die unter dem Begriff *Array Algebra* zusammengefaßt sind. Wie später gezeigt wird, erlauben diese Ansätze ein sehr schematisiertes Arbeiten

auch bei höherdimensionalen Problemstellungen. Die Fülle der Parameter kann vor allem durch die effiziente Lösungsmöglichkeit der Inversen sehr leicht bewältigt werden.

Die Verwendung dieser Methoden zwingt uns den Begriff der Regelmäßigkeit bei mehrdimensionalen Problemen neu festzulegen. Bei der Verwendung von Toeplitz- und zirkulierenden Matrizen war die Gleichabständigkeit der Punkte und die Ortsunabhängigkeit der Funktion ein wesentliches Kriterium. Bei Ansätzen der Array Algebra steht hingegen nur das gleiche Verhalten innerhalb einer Dimension im Vordergrund. Eine einmal definiertes Bild in der Dimension x_i muß sich unabhängig von den anderen Dimension x_k, $k = 1, ..., i-1, i+1, ...n$ wiederholen. Ein wesentliches Merkmal dieser Aufgabenstellungen ist somit die Ortsunabhängigkeit der Funktion in x_i-Richtung bezüglich der Dimensionen x_k.

Bedingt durch diese speziellen Formen können für die Auflösungen und die Inversionen der hier dargestellten Matrizen besondere Algorithmen verwendet werden, bei der die Anzahl der Operationen nicht wie üblich mit der dritten Potenz der Dimension wächst.

Anmerkungen und Hinweise:

Im geodätischen Umfeld treten eindimensionalen regelmäßigen Strukturen im Bereich von regelmäßigen linienartigen Netzen auf. Typische Vertreter von linienartigen Netzstrukturen sind Nivellementzüge. Durch Messung von Höhendifferenzen zwischen diskreten Punkten entlang einer Linie wird eine Höhenübertragung zwischen dem Anfangspunkt A und dem Endpunkt E durchgeführt. Während bei analytischen Netzanalysen unendliche oder zyklische Netze im Vordergrund der Untersuchungen stehen, da meist nur unter dieser Annahme geschlossene Formelapparate entwickelbar sind, stehen bei praktischen Untersuchungen endliche Strukturen im Vordergrund. Regelmäßige Züge mit endlicher Ausdehnung führen zu Toeplitz-Matrizen. Zyklische Strukturen (in sich geschlossene Züge) oder periodische Strukturen erzeugen zirkulierende Matrizen. Bedingt durch die lokale Meßanordnung (Messung zwischen benachbarten Punkten) sind gebänderte Strukturen die Regel.

Komplexer wird die Aufgabe, wenn eine mehrdimensionale Lageübertragung (x,y oder x,y,z) mit kombinierten Richtungs-, Strecken- und Winkelmessungen vorgenommen wird. Bei diesen kettenartigen Gebilden wird der Einzelknoten durch ein Knotengebilde (Cluster) ersetzt, womit regelmäßige Blockstrukturen im Vordergrund stehen. Die Blöcke entlang der Diagonalen definieren das Verhalten innerhalb eines Knotengebildes, die Blöcke außerhalb der Diagonale legen die Wechselbeziehung zwischen den sich wiederholenden Knotengebilden fest. Die Blöcke haben daher meist kleine Dimensionen und keine regelmäßige innere Struktur. Bei flächenhaft ausgedehnten Netzen wächst die Dimension dieser Blöcke, wobei aber regelmäßige innere Strukturen auftreten. Flächennivellements sind typische Vertreter dieser Gattung. Eine Ausdehnungsrichtung legt die Abfolge der Blöcke fest, die zweite Ausdehnungsrichtung bestimmt die Struktur innerhalb der Blöcke.

Meist theoretische Untersuchungen über Steifigkeit und Fehlerausbreitungen innerhalb von regelmäßig aufgebauten Netzen sind in folgenden Arbeiten zu finden:

[5] BARTELME Norbert, Peter MEISSL (1979): Application of Toeplitz Forms to the Mathematical Analysis of Geodetic Networks. Bollettino di Geodesia e Scienze Affini - No. 4, pp. 577-587.

[10] BORRE Kai (1979): Covariance Matrices / Functions for 1-D Levelling Problems. Unveröffentlicht.

[11] BORRE Kai (1981): A Priori Estimation of the Strength of Control Networks. Invited Paper, 'FIG XVI. Intern. Kongress', Montreux.

[51] MEISSL Peter (1969): Über zufällige Fehler in regelmässigen gestreckten Ketten. Zeitschrift für Vermessungswesen (ZfV), Konrad Wittwer, Stuttgart, Vol. 94, Heft 1, S. 14-26.

[53] MEISSL Peter (1976): Strength Analysis of Two-Dimensional Angular Anblock Networks. Manuscripta Geodaetica, Vol. 1, 293-333.

[88] SÜNKEL Hans (1985): Fourier Analysis of Geodetic Networks. Eigenvalues. GRAFAREND E.W., F. SANSO (Editors): 'Optimization and Design of Geodetic Networks', Springer, Heidelberg, pp. 257-300.

[97] WIESER Manfred (1988): Theoretische Untersuchungen spektraler Methoden zur Analyse regelmässiger Strukturen am Beispiel der Fouriertransformation geodätischer Netze. Mitteilungen der geodätischen Institute der TU Graz, Folge 60.

Eine zweite große Anwendungsgruppe bilden die Interpolations- und Approximationsaufgaben. Bei der Verwendung von homogenen, also ortsunabhängigen Grundfunktionen (z.B. Splines, finite Elemente, Kovarianzfunktionen) und regelmäßiger Verteilung der Stützpunkte treten bei eindimensionalen örtlich begrenzten Anwendungen Toeplitz-Systeme auf. Periodischen Anwendungen (z.B. periodischer Spline) oder Anwendungen auf geschlossenen Kurven (z.B. Kreis) führen auf zirkulierende Systeme.

Bei flächenhaften Anwendungen Anwendungen wird in jeder Ausbreitungsrichtung eine spezielle Struktur hervorgerufen. Die resultierenden Matrizen sind in Blöcke strukturiert, wobei eine Dimension das Verhalten der Blöcke untereinander festlegt und die zweite Ausbreitungsrichtung das Verhalten innerhalb der Blöcke prägt.

Die Beschränktheit in beiden Dimensionen führt auf Toeplitz-Toeplitz-Systeme, wo sowohl die Blöcke untereinander, als auch jeder Block in sich die Toepltiz-Struktur aufweisen. Die Referenzfläche für diese Anwendungen wäre die Ebene. Anwendungen mit dem Zylinder als Referenzfläche oder einer beschränkten Anzahl von Meridianen auf einer Kugel führen zu Block-Toeplitz-Matrizen mit zirkulierenden Blöcken oder auf Block-zirkulierende Systeme mit Toeplitz-Blöcken. Anwendungen am Torus ergeben zirkulierende Systeme mit zirkulierenden Blöcken. Weisen die Grundfunktionen endliche Träger auf (finite Grundfunktionen) so führt dies zu gebänderten Systemen. Zusätzliche Vereinfachungen in der Berechnung ergeben sich bei separierbaren Grundfunktionen, wo $f(x_1, x_2)$ durch $f_1(x_1) * f_2(x_2)$ darstellbar ist.

Durch die Automatisation der Meßtechnik mehren sich die geodätischen Anwendungen. Abtastung von Information zum Beispiel durch regelmäßige Messungen der Flughöhe des Satelliten über dem Meer (Altimetrie) oder Auswertung der Information von CCD-Kameras führen auf die oben angeführten Strukturen. Auch die systematische Aufbereitung von umfangreichen Datenmaterial, wie Geländehöhen, Schwerewerte erfolgen in regelmäßigen Abständen oder beziehen sich vielfach auf regelmäßige Gitter im erweiterten Sinn (Kronecker-Trennbarkeit).

Wolf-Dieter Schuh
ITGB, Uni Bonn

[6] BATH Markus (1974): Spectral Analysis in Geophysics. Developments in Solid Earth Geophysics 7, Elsevier Scientific Publishing Company, Amsterdam-Oxford-New York.

[9] BLAHA George (1977): Least Squares Prediction and Filtering in Any Dimension Using the Principles of Array Algebra. Bulletin Géodésique, Vol. 51, No. 4, pp. 265-286.

[12] BOTTONI Gian Paolo, Riccardo BARZAGHI (1993): Fast Collocation. Bulletin Géodésique, Vol. 67, No. 2, pp. 119-126.

[16] COLOMBO Oscar L. (1979): Optimal Estimation from Data Regularly Sampled on a Sphere with Applications in Geodesy. Department of Geodetic Science, Report No. 291, Ohio State University, Columbus, Ohio.

[27] EBNER Heinrich (1970): Die theoretische Lagegenauigkeit ausgeglichener Blöcke mit bis zur 10000 unabhängigen Modellen. Bildmessung und Luftbildwesen, Vol. 38, S. 225-232.

[28] EREN Kamil (1980): Spectral Analysis of GEOS-3 Altimeter Data and Frequency Domain Collocation. Department of Geodetic Science, Report No. 297, Ohio State University, Columbus, Ohio.

[29] FRITSCH Dieter (1989): Multivariante Data Analysis. Proceedings of the Tutorial on 'Mathematical Aspects of Data Analysis', ISPRS Intercommission Working Group III/VI, pp. 169-183, Pisa, June 1-2, 1989.

[65] RAUHALA Urho A. (1978): Array Algebra DTM. Proceedings of Digital Terrain Models (DTM) Symposium, May, 9-11, St. Louis, American Society of Photogrammetry, Falls Church, Virginia.

[66] RAUHALA Urho A. (1979): Development of Experimental Array Algebra Algorithms for Filtering and Compaction of AS-11B-X and Seasat Altimetry Data. Report of Geodetic Services, Indialantic, Florida.

[79] SCHWARZ K. P., M. G. SIDERIS, R. FORSBERG (1990): The Use of FFT Techniques in Physical Geodesy. Geophys. J. Int. 100, pp. 485-514.

[95] VERMEER Martin (1992): Terrain Reduction and Gridding Techniques for Geoid Determination. Lecture Notes for NKG-Autumn School for 'Geodesy and Geophysics', Publications of the Finnish Geodetic Institute 115, Helsinki. pp. 173-181.

[96] WOTRUBA Markus F. (1982): Kollokation für strukturierte Datensätze. Diplomarbeit TU Graz.

6.1 Regelmäßige eindimensionale Strukturen

6.1.1 Lösung von Toeplitz-Systemen

Der gesuchte Lösungsvektor $\mathbf{x}$ ist durch das lineare Gleichungssystem

$$\mathbf{T}\mathbf{x} = \mathbf{b} \tag{6.6}$$

festgelegt. $\mathbf{T}$ entspricht einer symmetrischen, positiv definite Toeplitz-Matrix der Dimension n und $\mathbf{b}$ stellt einen beliebigen Konstantenvektor dar. Der hier dargestellte Algorithmus zur Lösung von symmetrischen, positiv definiten Toeplitz-Systemen beruht auf einer Arbeit von *N. Levinson (1947)[[48]*. Eine weitere Darstellung für diese Problemstellung ist z.B. bei *G. Golub et. al. (1983)[38]*, Seite 128-129 zu finden. Bei *W. Press et. al. (1989)[63]* sind auf den Seiten 47-51 sowohl die Lösungsstategie als auch ein FORTRAN Programm für nichtsymmetrische Toeplitz-Matrizen aufgezeigt.

Da die gesuchte Lösung durch ein *Ränderungs*-Verfahren errechnet wird, ist es notwendig, eine Zähler k $(1 \leq k \leq n)$ einzuführen, der die Zugehörigkeit zum Ränderungsschritt durch eine Hochzahl kennzeichnet. Das Gleichungssystem

$$\mathbf{T}^k\mathbf{x}^k = \mathbf{b}^k \tag{6.7}$$

stellt ein Teilproblem von (6.6) der Form

$$\begin{bmatrix} t_0 & t_1 & t_2 & \dots & \dots & \dots & t_{k-2} & t_{k-1} \\ t_1 & t_0 & t_1 & \dots & \dots & \dots & t_{k-3} & t_{k-2} \\ t_2 & t_1 & t_0 & & & & t_{k-4} & t_{k-3} \\ \dots & \dots & & \dots & & & \dots & \dots \\ \dots & \dots & & & \dots & & \dots & \dots \\ \dots & \dots & & & & \dots & \dots & \dots \\ t_{k-2} & t_{k-3} & t_{k-4} & \dots & \dots & \dots & t_0 & t_1 \\ t_{k-1} & t_{k-2} & t_{k-3} & \dots & \dots & \dots & t_1 & t_0 \end{bmatrix} \begin{bmatrix} x_0^k \\ x_1^k \\ x_2^k \\ \dots \\ \dots \\ \dots \\ x_{k-2}^k \\ x_{k-1}^k \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \dots \\ \dots \\ \dots \\ b_{k-2} \\ b_{k-1} \end{bmatrix} \tag{6.8}$$

dar. Die Matrizen $\mathbf{T}^k$ und $\mathbf{b}^k$ ergeben sich als Teilmatrizen $\mathbf{T}_{0:k-1,0:k-1}$ bzw. $\mathbf{b}_{0:k-1}$ aus den Matrizen $\mathbf{T}$ und $\mathbf{b}$. Der Vektor $\mathbf{x}^k$ bildet den Lösungsvektor des k-ten Teilproblems. Dabei ist zu beachten, daß die einzelnen Koeffizienten x_i^k nicht im direkten Zusammenhang mit den Koeffizienten x_i der Lösung $\mathbf{x}$ des Gesamtproblems stehen.

Die Lösungsstrategie besteht nun darin, aus der Lösung $\mathbf{x}^k$ eines beliebigen Teilproblems $\mathbf{T}^k\mathbf{x}^k = \mathbf{b}^k$ auf die Lösung $\mathbf{x}^{k+1}$ zu schließen. Als Hilfsgrößen benötigen wir dabei die erste Zeile $\mathbf{Z}^k_{0,0:k-1}$ der Inversen $\mathbf{Z}^k$ der regulären Teilmatrix $\mathbf{T}^k$.

Beginnend mit $k = 1$ und der Teilmatrix $\mathbf{T}^1$ mit nur einem Element t_0 im ersten Ränderungsschritt wird die Matrix sukzessive um eine Dimension erweitert, sodaß die Matrix $\mathbf{T}^n$ im n-ten Ränderungsschritt die Dimension n aufweist und somit die gesamte Matrix $\mathbf{T}$ darstellt. Da von einem nicht singulären Gleichungssystem ausgegangen wird, existiert bei jedem Schritt k die Inverse $\mathbf{Z}^k$ der Teilmatrix $\mathbf{T}^k$.

Wolf-Dieter Schuh
ITGB, Uni Bonn

Wir starten zunächst bei einem beliebigen Schritt k und nehmen an, ein Vielfaches α^k der ersten Zeile der Inversen $\mathbf{Z}^k$ der Teilmatrix $\mathbf{T}^k$ zu kennen, womit zufolge der Definition der Inversen ($\mathbf{Z}^k\mathbf{T}^k = \mathbf{I}$) folgende Relation gültig ist:

$$\left[\begin{array}{cccc} z^k_{00} & z^k_{01} & \ldots & z^k_{0,k-1} \end{array}\right] \quad \mathbf{T}^k = \left[\begin{array}{cccc} \alpha^k & 0 & \ldots & 0 \end{array}\right] \tag{6.9}$$

Erweitert man nun die Teilmatrix $\mathbf{T}^k$ der zu lösenden Toeplitz-Matrix um eine Zeile und eine Spalte zum System $\mathbf{T}^{k+1}$ und fügt ein Nullelement an den Vektor $\mathbf{Z}^k_{0,0:k-1}$ an, so ergibt sich im Allgemeinen ein Koeffizient $\beta^k \neq 0$ durch die hinzukommende Zeile

$$\left[\begin{array}{ccccc} z^k_{00} & z^k_{01} & \ldots & z^k_{0,k-1} & 0 \end{array}\right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{ccccc} \alpha^k & 0 & \ldots & 0 & \beta^k \end{array}\right] \tag{6.10}$$

als letztes Glied auf der rechten Seite. Die hinzugefügte Spalte übt auf Grund des Nullelements als Multiplikator keinen Einfluß auf die rechte Seite aus. Das ausführlich angeschriebene System (6.11)

$$\begin{array}{ccccccccccccc}
z^k_{00}\,t_0 & + & z^k_{01}\,t_1 & + & z^k_{02}\,t_2 & + & \ldots & + & z^k_{0,k-1}\,t_{k-1} & + & 0\,t_k & = & \alpha^k \\
z^k_{00}\,t_1 & + & z^k_{01}\,t_0 & + & z^k_{02}\,t_1 & + & \ldots & + & z^k_{0,k-1}\,t_{k-2} & + & 0\,t_{k-1} & = & 0 \\
z^k_{00}\,t_2 & + & z^k_{01}\,t_1 & + & z^k_{02}\,t_0 & + & \ldots & + & z^k_{0,k-1}\,t_{k-3} & + & 0\,t_{k-2} & = & 0 \\
\ldots & & \ldots & & \ldots & & & & \ldots & & \ldots & & \\
\ldots & & \ldots & & \ldots & & & & \ldots & & \ldots & & \\
z^k_{00}\,t_{k-1} & + & z^k_{01}\,t_{k-2} & + & z^k_{02}\,t_{k-3} & + & \ldots & + & z^k_{0,k-1}\,t_0 & + & 0\,t_1 & = & 0 \\
z^k_{00}\,t_k & + & z^k_{01}\,t_{k-1} & + & z^k_{02}\,t_{k-2} & + & \ldots & + & z^k_{0,k-1}\,t_1 & + & 0\,t_0 & = & \beta^k
\end{array} \tag{6.11}$$

soll die Situation verdeutlichen und die Berechnung von β^k mit

$$\beta^k = \sum_{i=0}^{k-1} z^k_{0i}\; t_{k-i} \tag{6.12}$$

auszeigen. Formt man das System (6.11) um, indem man die Zeilen und Spalten in umgekehrter Reihenfolge anschreibt, so erlangt man

$$\left[\begin{array}{ccccc} 0 & z^k_{0,k-1} & \ldots & z^k_{01} & z^k_{00} \end{array}\right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{ccccc} \beta^k & 0 & \ldots & 0 & \alpha^k \end{array}\right] \quad . \tag{6.13}$$

Ziel des nächstfolgenden Schrittes ist es, die um Null erweiterte Inversenzeile $\mathbf{Z}^k_{0,0:k-1}$ so zu verändern, daß in (6.10) das Glied β^k zu Null wird. Ist diese Bedingung erfüllt, so bildet die geänderte Zeile die erste Zeile der Inversen des erweiterten Systems $\mathbf{T}^{k+1}$, also $\mathbf{Z}^{k+1}_{0,0:k}$. Formt man die Linearkombination von (6.10) und (6.13) mit dem zunächst unbekannten Faktor γ^k

$$\begin{aligned}
&\left[\begin{array}{ccccc} z^k_{00} & z^k_{01} + \gamma^k z^k_{0,k-1} & \ldots & z^k_{0,k-1} + \gamma^k z^k_{01} & \gamma^k z^k_{00} \end{array}\right] \quad \mathbf{T}^{k+1} = \\
&\qquad = \left[\begin{array}{ccccc} \alpha^k + \gamma^k\beta^k & 0 & \ldots & 0 & \beta^k + \gamma^k\alpha^k \end{array}\right]
\end{aligned} \tag{6.14}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

und achtet auf das Verschwinden des letzten Gliedes der rechten Seite, so ermittelt sich der Faktor γ^k durch

$$\gamma^k = -\frac{\beta^k}{\alpha^k} \quad . \tag{6.15}$$

Führt man die Berechnungen der linearkombinierten Zeile von (6.14) durch,

$$\begin{array}{lcllll} z_{00}^{k+1} & = & z_{00}^{k} & & \\ z_{01}^{k+1} & = & z_{01}^{k} & + & \gamma^k z_{0,k-1}^{k} \\ z_{02}^{k+1} & = & z_{02}^{k} & + & \gamma^k z_{0,k-2}^{k} \\ \ldots & & \ldots & & \ldots \\ \ldots & & \ldots & & \ldots \\ \ldots & & \ldots & & \ldots \\ z_{0,k-1}^{k+1} & = & z_{0,k-1}^{k} & + & \gamma^k z_{01}^{k} \\ z_{0k}^{k+1} & = & & + & \gamma^k z_{00}^{k} \end{array} \tag{6.16}$$

so wird das Vielfache der ersten Zeile der Inversen $\mathbf{Z}^{k+1}$ des erweiterten Systems $\mathbf{T}^{k+1}$ errechnet, womit wieder die Ausgangssituation (6.9), allerdings im $(k+1)$-ten Schritt,

$$\left[\begin{array}{cccc} z_{00}^{k+1} & z_{01}^{k+1} & \ldots & z_{0k}^{k+1} \end{array} \right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{cccc} \alpha^{k+1} & 0 & \ldots & 0 \end{array} \right] \tag{6.17}$$

errechnet ist. Der Übergang der rechten Seite ist durch

$$\alpha^{k+1} = \alpha^k + \gamma^k \beta^k \tag{6.18}$$

gegeben.

Durch Wiederholung dieser Berechnungsschritte beginnend mit der Startlösung

$$z_{00}^{1} = 1 \tag{6.19}$$

und daraus folgend

$$\alpha^1 = t_0, \qquad \beta^1 = t_1 \tag{6.20}$$

kann nach n Ränderungsschritten das Vielfache der ersten Inversenzeile berechnet werden. Die erste Zeile der Inversen dient als Ausgangspunkt zur Berechnung der gesamten Inversen (siehe nachfolgender Abschnitt 6.1.2). Die Lösung eines Toeplitz-Systems könnte nun über die Inverse berechnet werden. Wenn man jedoch bedenkt, daß die Multiplikation einer quadratischen Matrix der Dimension n mit einem Vektor noch zusätzlich n^2 Operationen benötigt, so zeigt sich der Weg der Mitberechnung der Lösung während der Berechnungen der ersten Zeile der Inversen als günstiger.

Wolf-Dieter Schuh
ITGB, Uni Bonn

Unter der Annahme, daß die Lösung $\mathbf{x}^k$ zu einem bestimmten Teilsystem $\mathbf{T}^k$ gehört, gilt

$$\left[\begin{array}{cccc} x_0^k & x_1^k & \ldots & x_{k-1}^k \end{array}\right] \quad \mathbf{T}^k = \left[\begin{array}{cccc} b_0 & b_1 & \ldots & b_{k-1} \end{array}\right] \tag{6.21}$$

Erweitert man das Teilsystem der Toeplitz-Matrix um eine Zeile und Spalte und erweitert den Lösungsvektor durch eine Null, so errechnet sich durch das erweiterte System im allgemeinen nicht die um das Element b_k erweiterte rechte Seite, sondern der um ein Element δ^k erweiterten Vektor

$$\left[\begin{array}{ccccc} x_0^k & x_1^k & \ldots & x_{k-1}^k & 0 \end{array}\right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{ccccc} b_0 & b_1 & \ldots & b_{k-1} & \delta^k \end{array}\right], \tag{6.22}$$

das sich ähnlich zu β^k (siehe Formel (6.11) bzw. (6.12)) durch

$$\delta^k = \sum_{i=0}^{k-1} x_i^k \; t_{k-i} \tag{6.23}$$

bestimmen läßt. Ziel der folgenden Überlegungen ist es, den Lösungsvektor so zu verändern, daß anstelle des Wertes δ^k das k-te Glied des Konstantenvektors, also b_k, errechnet wird. Mit Hilfe der Umkehrung von (6.17)

$$\left[\begin{array}{cccc} z_{0k}^{k+1} & z_{0,k-1}^{k+1} & \ldots & z_{00}^{k+1} \end{array}\right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{cccc} 0 & \ldots & 0 & \alpha^{k+1} \end{array}\right]. \tag{6.24}$$

kann in Linearkombination mit (6.22) mit dem zunächst unbekannten Faktor ε^k folgende Beziehung

$$\begin{aligned} \left[\begin{array}{cccc} x_0^k + \varepsilon^k z_{0k}^{k+1} & x_1^k + \varepsilon^k z_{0,k-1}^{k+1} & \ldots & x_k^k + \varepsilon^k z_{01}^{k+1} \quad \varepsilon^k z_{00}^{k+1} \end{array}\right] \quad \mathbf{T}^{k+1} = \\ = \left[\begin{array}{cccc} b_0 & \ldots & b_{k-1} & \delta^k + \varepsilon^k \alpha^{k+1} \end{array}\right] \end{aligned} \tag{6.25}$$

errechnet werden. Da das letzte Element identisch b_k sein soll, können wir aus der Bedingung

$$b_k = \delta^k + \varepsilon^k \alpha^{k+1} \tag{6.26}$$

den Multiplikationsfaktor ε^k

$$\varepsilon^k = \frac{1}{\alpha^{k+1}} \left(b_k - \delta^k \right) \tag{6.27}$$

bestimmen. Die durch (6.25) errechnete neue Lösung $\mathbf{x}^{k+1}$ des erweiterten Teilsystems $\mathbf{T}^{k+1}$ wird wieder mit

$$\left[\begin{array}{ccccc} x_0^{k+1} & x_1^{k+1} & \ldots & x_{k-1}^{k+1} & x_k^{k+1} \end{array}\right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{ccccc} b_0 & b_1 & \ldots & b_{k-1} & b_k \end{array}\right] \tag{6.28}$$

bezeichnet, wobei $\mathbf{x}^{k+1}$ aus

$$\begin{array}{rcccl} x_0^{k+1} & = & x_0^k & + & \varepsilon^k z_{0k}^{k+1} \\ x_1^{k+1} & = & x_1^k & + & \varepsilon^k z_{0,k-1}^{k+1} \\ x_2^{k+1} & = & x_2^k & + & \varepsilon^k z_{0,k-2}^{k+1} \\ \ldots & & \ldots & & \ldots \\ \ldots & & \ldots & & \ldots \\ \ldots & & \ldots & & \ldots \\ x_{k-1}^{k+1} & = & x_{k-1}^k & + & \varepsilon^k z_{01}^{k+1} \\ x_k^{k+1} & = & & + & \varepsilon^k z_{00}^{k+1} \end{array} \tag{6.29}$$

berechnet wird. Ausgehend von der Startlösung $\mathbf{x}^1$

$$x_0^1 = \frac{b_0}{t_0} \tag{6.30}$$

und damit

$$\delta^1 = x_0^1 t_1 \tag{6.31}$$

kann die Lösung während der Iteration mitberechnet werden.

Für alle hier getroffenen Überlegungen wird vorausgesetzt, daß das lineare Gleichungssystem positiv definit ist. Interessant ist nun die Frage, wie sich die Nichteinhaltung dieser Forderung auf das erstellte Verfahren auswirkt.

Die einzige kritische Situation während des Verfahrens stellt die Division durch α^k bei der Berechnung des Faktors γ^k in Formel (6.15) dar. Wenn α^k zu numerisch Null wird, so kann dies zwei Ursachen haben:

- $\alpha^1 = 0$, da $t_0 = 0$: Somit sind alle Diagonalelemente gleich Null ($t_0 = 0$) und die Summe der Diagonalelemente Spur($\mathbf{T}$) verschwindet. Andererseits entspricht die Spur($\mathbf{T}$) einer Matrix der Summe alle Eigenwerte. Da ein positiv definites System nur positive Eigenwerte besitzt, widerspricht dieser Ansatz der Forderung nach einer positiv definiten Matrix. Ein symmetrisches Toeplitz-System mit Nullen in der Diagonale ist anderseits nicht notwendigerweise singulär, kann jedoch mit dem zuvor dargestellen Verfahren nicht berechnet werden.

- $\beta^k = \pm\alpha^k$, und somit $\alpha^{k+1} = 0$: Während der Berechnung entartet die Gleichung (6.21), indem die gesamte rechte Seite verschwindet.

$$\left[\begin{array}{cccc} z_{00}^{k+1} & z_{01}^{k+1} & \ldots & z_{0k}^{k+1} \end{array}\right] \quad \mathbf{T}^{k+1} = \left[\begin{array}{cccc} 0 & 0 & \ldots & 0 \end{array}\right] \tag{6.32}$$

  Der Vektor $\mathbf{Z}_{0,0:k}^{k+1}$ bildet somit die im Allgemeinen nicht triviale Lösung des homogenen Gleichungssystem und spannt den Orthogonalraum zum Spaltenraum von $\mathbf{T}^{k+1}$ auf. Somit liegt ein Rangdefekt der Matrix $\mathbf{T}^{k+1}$ vor.

Das zuvor dargestellte Verfahren kann somit in folgenden Algorithmus zusammengefaßt werden:

Algorithmus 6.1 : Lösung eines symmetrischen, positiv definiten Toeplitz-Systems (skalar)

Aufgabenstellung: Gegeben ist in einem Vektor **t** gespeichert die erste Zeile einer symmetrischen, positiv definiten Toeplitz-Matrix **T** der Dimension n und ein Konstantenvektor **b**. Gesucht ist die Lösung **x** des Systems

$$\mathbf{Tx} = \mathbf{b} \ ,$$

wenn das Gleichungssystem positiv definit ist. Bei nicht positiv definiter Matrix wird das Verfahren abgebrochen.

Anmerkungen: Der Konstantenvektor **b** wird vom Lösungsvektor **x** überschrieben. Ferner liefert dieser Algorithmus die erste Zeile der Inversen **Z**, die im Vektor **z** gespeichert wird.

Achtung: Die Indizierung beginnt wie gewohnt bei 1 und geht bis n. Dies steht zwar im Gegensatz zur dargestellten Theorie, sorgt jedoch für eine einheitliche Bezeichnung innerhalb der Algorithmen.

Schnittstellen:

Eingabe:	`n`	... Größe des Systems
	`t(n)`	... erste Zeile der Toeplitz-Matrix $\mathbf{T}^n_{0,0:n-1}$
	`b(n)`	... rechte Seite **b** des Systems
Ausgabe:	`b(n)`	... Lösung **x** des Systems
	`z(n)`	... erste Zeile der Inversen $\mathbf{Z}^n_{0,0:n-1}$
Felder:	`t(n), b(n), z(n)`	

Lösungsweg: Pseudocode

1. *Initialisierung der Ränderungsschleife und setzen der Startwerte:*
   `k=1 ; z(1)=1.0 ; alpha=t(1) ; delta=0 .`
2. *Überprüfe ob Ränderungsschritt sinnvoll ist und beende wenn Matrix nicht positiv definit ist:*
   `IF ABS(alpha)<num_Null GOTO Ende:` 'MATRIX NICHT POSITIV DEFINIT'.
3. *Berechnung der neuen Lösung:*
   Ermittle den Multiplikationsfaktor : `epsilon=(b(k)-delta)/alpha`
   `DO i=1 TO k-1 ; b(i)=b(i)+epsilon*z(k-i+1) ; END DO i`
   `b(k)=epsilon*z(1) .`
4. *Überprüfe auf reguläre Ende und erhöhe den Ränderungszähler:*
   `IF k=n GOTO 8:`
   `k=k+1 .`
5. *Berechne die Faktoren β und δ :*
   `beta =` $\sum_{i=1}^{k-1}$ `z(i)*t(k-i+1)`
   `delta =` $\sum_{i=1}^{k-1}$ `b(i)*t(k-i+1).`

Wolf-Dieter Schuh
ITGB, Uni Bonn

6. *Berechne die neue Inversenzeile* $\mathbf{Z}^{k+1}_{0,0:k-1}$ *:*
Ermittle den Multiplikationsfaktor : `gamma=-beta/alpha`
`ih=INT(k/2)` (`INT ...` *größte ganze Zahl*).
```
DO i=2 TO ih
     h=z(i); z(i)=h+gamma*z(k-i+1);z(k-i+1)=z(k-i+1)+gamma*h
END DO i
IF INT(k/2)≠k/2 THEN z((k+1)/2)=z((k+1)/2)+gamma*z(k+1)/2)
z(k)=gamma*z(1) .
```

7. *Berechne* α^{k+1} *und führe nächsten Ränderungsschritt aus:*
```
alpha=alpha+gamma*beta
GOTO 2: .
```

8. *Bearbeite die Inversenzeile und beende:*
```
DO i=1 TO n ; z(i)=z(i)/alpha ; END DO i
```

`Ende:` 'O.K. ENDE '.

Ressourcen: Anzahl der Operationen: $2n^2 + \mathcal{O}(n)$
Platzbedarf: $3n$

Kern der Berechnungen sind die Neuermittlung von $\mathbf{Z}^{k+1}_{0,0:k}$ im Schritt 3 bzw. $\mathbf{x}^{k+1}_{0,0:k}$ im Schritt 5. Ferner sind die Summationen der Faktoren β und δ im Schritt 7 zu beachten. Jede dieser Berechnungen benötigt k Operationen (Additionen und Multiplikationen). Somit sind pro Ränderungsschritt $4k + 3$ Rechenoperationen notwendig, wobei k von 1 bis n läuft. Die Bearbeitung der Inversenzeile im Schritt 8 erfordert weitere n Operationen. In Summe benötigt der Algorithmus 6.1 somit $2n^2 + \mathcal{O}(n)$ Operationen.

Der Ablauf des Algorithmus 6.1 ist weitgehend mit dem abgeleiteten Formelapparat abgestimmt.

Der Algorithmus 6.1 berechnet sowohl die Lösung $\mathbf{x}$ als auch die erste Zeile $\mathbf{Z}_{0,0:n-1}$ der Inversen. Da das Vielfache der ersten Inversenzeile ein notwendiges Hilfsmittel zur Berechnung der Lösung ist, kann diese Berechnung nicht eingespart werden. Wird hingegen nur die erste Zeile der Inversen benötigt, nicht jedoch die Lösung $\mathbf{x}$, so kann der Algorithmus entsprechend modifiziert werden. Im Algorithmus 6.1 entfällt der Schritt 3 vollständig und im Schritt 5 kann die Berechnung von δ unterbleiben. Der so modifizierte Algorithmus benötigt nur mehr $n^2+\mathcal{O}(n)$ Rechnenoperationen bei einem Platzbedarf von $2n$ Speicherplätzen.

Kern des Lösungsalgorithmus 6.1 bilden die Summationsschleifen zur Berechnung der Faktoren β und δ und die beiden Schleifen zur Aktualisierung von $\mathbf{x}$ und $\mathbf{z}$. Durch Einführung eines Hilfsvektors $\mathbf{h}$ und leichte Modifikationen können alle Schleifen durch Vektoroperationen ersetzt werden. Der Hilfvektor $\mathbf{h}$ beinhaltet die Werte der aktuellen ersten Inversenzeile $\mathbf{z}$ in umgekehrter Reihenfolge. Ferner ist die Umkehr des Vektors $\mathbf{b}$ vor der Berechnung erforderlich. Die Speicherung der Unbekannten $\mathbf{x}$ in $\mathbf{b}$ erfolgt ebenfalls in umgekehrter Reihenfolge, sodaß nach der Berechnung wieder für die Richtigstellung gesorgt werden muß. Der Algorithmus 6.2 beinhaltet diese Modifikationen.

Algorithmus 6.2 : Lösung eines symmetrischen, positiv definiten Toeplitz-Systems (vektoriell)

Aufgabenstellung: Gegeben ist in einem Vektor **t** gespeichert die erste Zeile einer symmetrischen, positiv definiten Toeplitz-Matrix **T** der Dimension n und ein Konstantenvektor **b**. Gesucht ist die Lösung **x** des Systems

$$\mathbf{Tx} = \mathbf{b} \ ,$$

wenn das Gleichungssystem positiv definit ist. Bei nicht positiv definiter Matrix wird das Verfahren abgebrochen.

Anmerkungen: Der Konstantenvektor **b** wird vom Lösungsvektor **x** überschrieben. Ferner liefert dieser Algorithmus die erste Zeile der Inversen **Z**, die im Vektor **z** gespeichert wird.

Achtung: Die Indizierung beginnt wie gewohnt bei 1 und geht bis n. Dies steht zwar im Gegensatz zur dargestellten Theorie, sorgt jedoch für eine einheitliche Bezeichnung innerhalb der Algorithmen.

Schnittstellen:

Eingabe:	`n`	... Größe des Systems
	`t(n)`	... erste Zeile der Toeplitz-Matrix $\mathbf{T}^n_{0,0:n-1}$
	`b(n)`	... rechte Seite **b** des Systems
Ausgabe:	`b(n)`	... Lösung **x** des Systems
	`z(n)`	... erste Zeile der Inversen $\mathbf{Z}^n_{0,0:n-1}$
Felder:	`t(n), b(n), z(n), h(n)`	
Routinen:	`REV_ROWS(`*vek*`)`	... Umkehr der Reihenfolge der Elemente
	`DOT_PRODUCT(`vek_1, vek_2`)`	... Skalarprodukt

Lösungsweg: Pseudocode

1. *Initialisierung der Ränderungsschleife und setzen der Startwerte:*
   `k=1 ; z(1)=1.0 ; alpha=t(1) ; delta=0 ; b = REV_ROWS(b) .`
2. *Überprüfe ob Ränderungsschritt sinnvoll ist und beende wenn Matrix nicht positiv definit ist:*
   `IF ABS(alpha)<num_Null GOTO Ende:` 'MATRIX NICHT POSITIV DEFINIT'.
3. *Berechnung der neuen Lösung:*
   Erstelle Hilfsvektor **h** *:* `h = REV_ROWS(z)`
   Berechne Hilfsadresse : `adr = n-k+1` *Ermittle den Multiplikationsfaktor :*
   `epsilon=(b(adr)-delta)/alpha`
   `b(adr) = 0.0 ; b(adr:n) = b(adr:n) + epsilon * z(1:k) .`
4. *Überprüfe auf reguläre Ende und erhöhe den Ränderungszähler:*
   `IF k=n GOTO 8:`
   `k=k+1 .`
5. *Berechne die Faktoren β und δ :*
   `beta = DOT_PRODUCT(t(2:k),h)`
   `delta = DOT_PRODUCT(t(2:k),b(h:n))`

Wolf-Dieter Schuh
ITGB, Uni Bonn

6. *Berechne die neue Inversenzeile* $\mathbf{Z}^{k+1}_{0,0:k-1}$ *:*
Ermittle den Multiplikationsfaktor : `gamma=-beta/alpha`
`z(k) = 0.0 z(2:k) = z(2:k) + gamma * h`

7. *Berechne* α^{k+1} *und führe nächsten Ränderungsschritt aus:*
`alpha=alpha+gamma*beta`
`GOTO` `2:`.

8. *Bearbeite die Inversenzeile und beende:*
`z = 1.0/alpha * z`
`b = REV_ROWS(b)`
`Ende:` 'O.K. ENDE '.

Ressourcen: Anzahl der Operationen: $4n\ \mathcal{VEK}(\frac{n}{2}) + \mathcal{O}(n)$
Platzbedarf: $4n$

Alle inneren Schleifenkonstruktionen sind durch Vektoroperationen ersetzt, wobei alle Vektorlänge von 1 bis n anwachsen.

Literatur:

[38] GOLUB Gene H., Charles F. van LOAN (1983): Matrix Computations. North Oxford Academic, Oxford, pp. 128-129.

[48] LEVINSON Norman (1947): The Wiener RMS (Rott Mean Square) Error Criterion in Filter Design and Prediction. Journal of Mathematics & Physics, No. 25, pp. 261-278.

[63] PRESS William H., Brian P. FLANNERY, Saul A. TEUKOLSKY, William T. VETTERLING (1989): Numerical Recipes. The Art of Scientific Computing (FORTRAN Version). Cambridge University Press, Cambridge-London-New York-New Rochelle-Melbourne-Sydney, pp. 47-52.

6.1.2 Inversion von Toeplitz-Matrizen

W. F. Trench (1964)[90] zeige erstmals einen Weg, der es ermöglicht, eine positiv definite, symmetrische Toeplitz-Matrix mit einer Operationsanzahl, die nur quadratisch mit der Dimension ansteigt, zu invertieren. Eine Darstellung dieses Verfahrens ist zum Beispiel auch bei *G. Golub et. al. (1983)[38]*, Seite 130-132 zu finden.

Der Algorithmus geht von der Inversen der Toeplitz-Matrix $(\mathbf{T}^n)^{-1} = \mathbf{Z}^n$ aus und teilt die Matrix so, daß die erste Zeile und Spalte abgetrennt wird

$$(\mathbf{T}^n)^{-1} = \begin{bmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_1 & & & \\ \dots & & \mathbf{T}^n_{1:n-1,1:n-1} & \\ t_{n-1} & & & \end{bmatrix}^{-1} = \begin{bmatrix} z_{00} & z_{01} & \dots & z_{0,n-1} \\ z_{01} & & & \\ \dots & & \mathbf{Z}^n_{1:n-1,1:n-1} & \\ z_{0,n-1} & & & \end{bmatrix}. \quad (6.33)$$

Für die folgenden Berechnungen wird die verkürzte Schreibweise für (6.33)

$$(\mathbf{T}^n)^{-1} = \begin{bmatrix} t_0 & \mathbf{T}^n_{0,1:n-1} \\ \mathbf{T}^n_{1:n-1,0} & \mathbf{T}^n_{1:n-1,1:n-1} \end{bmatrix}^{-1} = \begin{bmatrix} z_{00} & \mathbf{Z}^n_{0,1:n-1} \\ \mathbf{Z}^n_{1:n-1,0} & \mathbf{Z}^n_{1:n-1,1:n-1} \end{bmatrix} \quad (6.34)$$

gewählt. Speziel sei hier auf den unteren quadratischen Teil $\mathbf{T}^n_{1:n-1,1:n-1}$ hingewiesen, der wieder eine - um eine Zeile und Spalte reduzierte - Toeplitz-Matrix $\mathbf{T}^{n-1}$ darstellt. Mit Ausnahme dieser Hochzahl werden im Folgenden aus Gründen der Übersichtlichkeit auf die Hochzahl n verzichtet. Alle nicht explizit anders definierten Matrizen beziehen sich somit auf die Gesamtmatrizen $\mathbf{T}^n$ und $\mathbf{Z}^n$.

Aus der Inversendefinition $\mathbf{T}^n\mathbf{Z}^n = \mathbf{I}$

$$\begin{bmatrix} t_0 & \mathbf{T}_{0,1:n-1} \\ \mathbf{T}_{1:n-1,0} & \mathbf{T}^{n-1} \end{bmatrix} \begin{bmatrix} z_{00} & \mathbf{Z}_{0,1:n-1} \\ \mathbf{Z}_{1:n-1,0} & \mathbf{Z}_{1:n-1,1:n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{6.35}$$

folgt aus der zweiten Zeile mal der zweiten Spalte die Indentität

$$\mathbf{T}_{1:n-1,0}\ \mathbf{Z}_{0,1:n-1} + \mathbf{T}^{n-1}\ \mathbf{Z}_{1:n-1,1:n-1} = \mathbf{I}\ , \tag{6.36}$$

die unter Voraussetzung der Existenz von $(\mathbf{T}^{n-1})^{-1}$ (positiven Definitheit) zu

$$\mathbf{Z}_{0,1:n-1} = (\mathbf{T}^{n-1})^{-1} - (\mathbf{T}^{n-1})^{-1}\ \mathbf{T}_{1:n-1,0}\ \mathbf{Z}_{0,1:n-1} \tag{6.37}$$

umgeformt werden kann. Aus der zweiten Zeile mal der ersten Spalte von (6.35) folgt ferner

$$\mathbf{T}_{1:n-1,0}\ z_{00} + \mathbf{T}^{n-1}\ \mathbf{Z}_{1:n-1,0} = \mathbf{0}\ . \tag{6.38}$$

Aus der letzten Gleichung kann der Ausdruck

$$(\mathbf{T}^{n-1})^{-1}\ \mathbf{T}_{1:n-1,0} = -\frac{1}{z_{00}}\ \mathbf{Z}_{1:n-1,0}\ . \tag{6.39}$$

gewonnen werden, der in (6.37) eingesetzt

$$\mathbf{Z}_{1:n-1,1:n-1} = (\mathbf{T}^{n-1})^{-1} + \frac{1}{z_{00}}\ \mathbf{Z}_{1:n-1,0}\ \mathbf{Z}_{0,1:n-1} \tag{6.40}$$

ergibt. Kennt man also die erste Zeile z_{00}, $\mathbf{Z}_{0,1:n-1}$ und Spalte $\mathbf{Z}_{1:n-1,0}$ der Inversen und besitzt außerdem Informationen über einzelne Elemente der Teilmatrix $(\mathbf{T}^{n-1})^{-1}$ so können die korrespondierenden Koeffizienten von $\mathbf{Z}_{1:n-1,1:n-1}$ einzeln aus (6.40) errechnet werden.

Durch einfache Umformung von (6.40) in

$$(\mathbf{T}^{n-1})^{-1} = \mathbf{Z}_{1:n-1,1:n-1} - \frac{1}{z_{00}}\ \mathbf{Z}_{1:n-1,0}\ \mathbf{Z}_{0,1:n-1} \tag{6.41}$$

kann aber auch der umgekehrte Weg eröffent werden, wo die Kenntnis von z_{00}, $\mathbf{Z}_{0,1:n-1}$, $\mathbf{Z}_{1:n-1,0}$ und einzelner Elemente von $\mathbf{Z}_{1:n-1,1:n-1}$ die Berechung der Elemente von $(\mathbf{T}^{n-1})^{-1}$ einzeln möglicht.

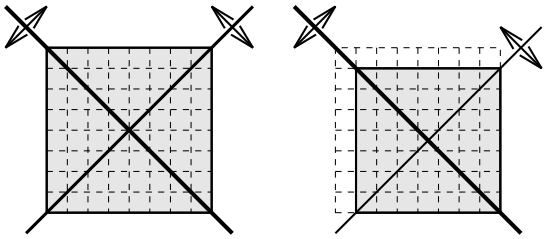


Abbildung 6.1: Symmetrieverhältnisse bei den Matrizen $\mathbf{Z}^n$ und $(\mathbf{T}^{n-1})^{-1}$:
Links: Symmetrie und Persymmetrie von $\mathbf{Z}^n$,
Rechts: Symmetrie und Persymmetrie der Teilmatrix $\mathbf{T}^n_{1:n-1,1:n-1} = \mathbf{T}^{n-1}$ bzw. deren Inversen $(\mathbf{T}^{n-1})^{-1}$.

In der Einführung zu diesem Abschnitt 6 wurde schon auf die Eigenschaft der Persymmetrie bei Toeplitz-Matrizen und deren Inversen hingewiesen. Nützen wir nun diese Symmetrie bezüglich der Diagonale Südwest - Nordost, so ist durch die Kenntnis der ersten Zeile auch die letzte Spalte bekannt. Da wir zusätzlich die Eigenschaft der Symmetrie bezüglich der Hauptdiagonale nutzten können sind zusätzlich auch die erste Spalte und die letzte Zeile festgelegt.

Die Formeln (6.40) bzw. (6.41) stellen einen Zusammenhang zwischen den Matrizen $\mathbf{Z}_{1:n-1,1:n-1}$ und $(\mathbf{T}^{n-1})^{-1}$ her. Die Abbildung 6.1 verauschaulicht die Symmetrieverhältnisse innerhalb dieser beiden Matrizen.

Aufgrund der Kenntnis der ersten Zeile der Inversen $\mathbf{Z}^n$ sind alle Koeffizienten an den Rändern von $\mathbf{Z}^n$ bekannt. Womit durch die Formel (6.41) die letzte Zeile und Spalte von $(\mathbf{T}^{n-1})^{-1}$ berechenbar wird. Da diese wegen der Persymmetrie äquivalent zur ersten Spalte und Zeile von $(\mathbf{T}^{n-1})^{-1}$ sind, kann mit Hilfe von (6.40) die zweite Zeile und Spalte von $\mathbf{Z}^n$ berechnet werden. Anhand der Abb. 6.2 wird der gesamte Rechengang veranschaulicht. Zur Erarbeitung der Rekursionsformel wird auf Abb. 6.3 verwiesen.

Mit Hilfe der Rekursionformel

$$z_{ij} = z_{i-1,j-1} - \frac{1}{z_{00}} \left(z_{0,n-j}\ z_{0,n-i} - z_{0i}\ z_{0j} \right) \tag{6.42}$$

kann die Inverse im oberen Viertel (siehe Abb.6.4) bestimmt werden. Wegen der Symmetrie und Persymmetrie ist somit die gesamte Inverse vollständig festgelegt.

Die Anzahl der Operationen pro berechnetem Element mit Hilfe der Rekursionsformel (6.42) beträgt drei. Da rund ein Viertel der Matrix berechnet werden muß, genau $\frac{1}{4}n(n+2)$ Elemente bei geradem n und $\frac{1}{4}n(n+2)+1$ bei ungeradem n ergeben sich für die reine Inversion ungefähr $\frac{3}{4}n^2 + \mathcal{O}(n)$ Operationen. Basis des Algorithmus 6.3 bildet die Rekursionsformel (6.42). Die restlichen Anweisungen organisieren dem Ablauf und die Adressrechnung.

Algorithmus 6.3 : Inversion einer symmetrischen, positiv definiten Toeplitz-Matrix

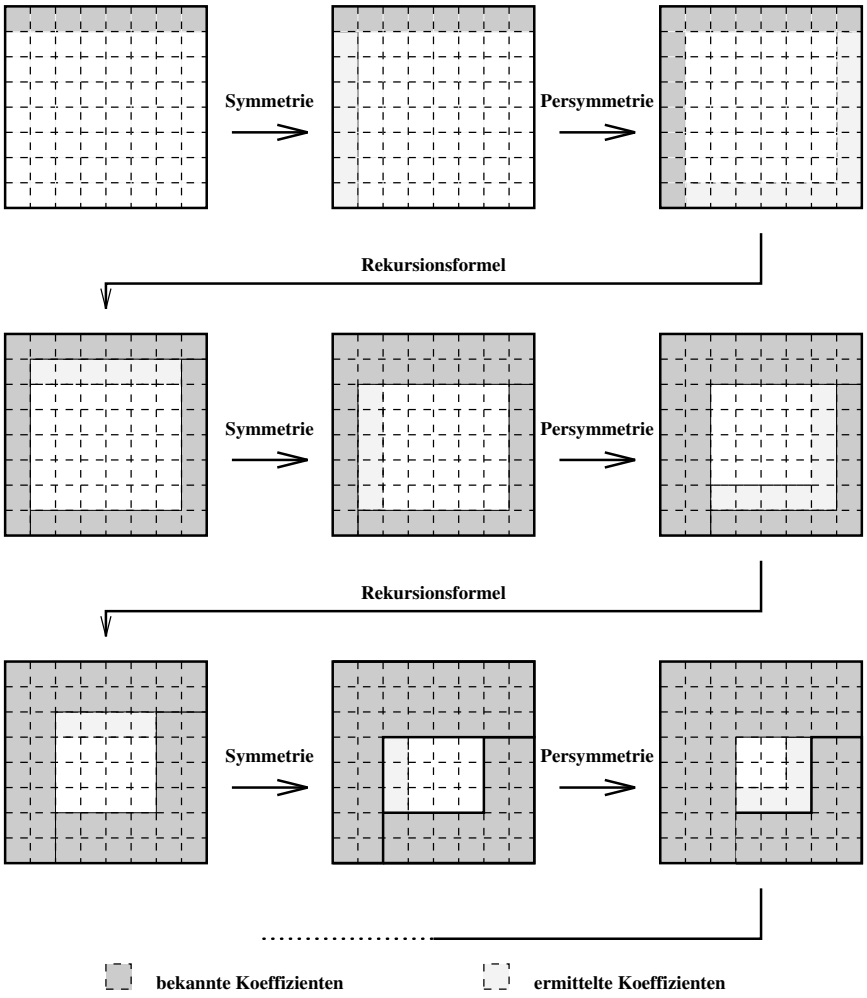


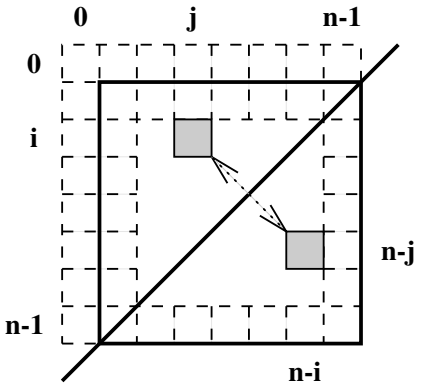
Abbildung 6.2: Ablauf bei der rekursiven Berechnung der Inversenkoeffizienten von Toeplitz-Matrizen.

Aufgabenstellung:
Gegeben ist, die in einem Vektor **z** gespeicherte, erste Zeile der Inversen einer symmetrischen, psoitiv definiten Toeplitz-Matrix der Dimension n. Gesucht ist die Vervollständigung des oberen Viertels der Inversen, sodaß die Inverse auf Grund der Symmetrie und Persymmetrie eindeutig festlegt ist. Ist der bereitgestellte Vektor **z** mit der Dimension `max` zur Aufnahme der Inversen zu klein, so erfolgt der Abbruch.

Ausgangssituation:
Im Vektor **z** sind auf den Positionen 1 bis n die Koeffizienten der ersten Inversenzeile gespeichert.

Achtung: Die Indizierung beginnt wie gewohnt bei 1 und geht bis n. Dies steht zwar im Gegensatz zur dargestellten Theorie, sorgt jedoch für eine einheitliche Bezeichnung innerhalb der Algorithmen.

Endsituation:



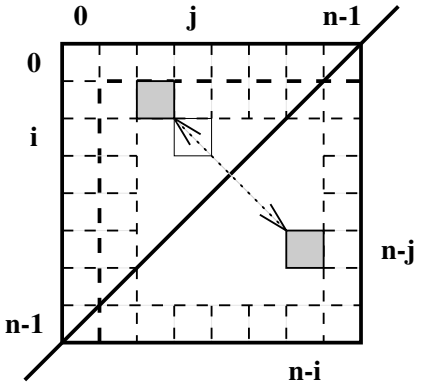
Skalare Darstellung von (6.40)

$$z_{ij} = t_{ij}^{(-1)} + \frac{1}{z_{00}}\, z_{0i}\, z_{0j}$$

Persymmetrie von $(\mathbf{T}^{n-1})^{-1}$

$$t_{ij}^{(-1)} = t_{n-j,n-i}^{(-1)}$$

$$z_{ij} = t_{n-j,n-i}^{(-1)} + \frac{1}{z_{00}}\, z_{0i}\, z_{0j}$$



Skalare Darstellung von (6.41)

$$t_{n-j,n-i}^{(-1)} = z_{n-j,n-i} - \frac{1}{z_{00}}\, z_{0,n-j}\, z_{0,n-i}$$

Persymmetrie von $\mathbf{Z}^n$

$$z_{n-j,n-i} = z_{i-1,j-1}$$

$$t_{n-j,n-i}^{(-1)} = z_{i-1,j-1} - \frac{1}{z_{00}}\, z_{0,n-j}\, z_{0,n-i}$$

Zusammengefaßt:

$$z_{ij} = z_{i-1,j-1} - \frac{1}{z_{00}} \left(z_{0,n-j}\, z_{0,n-i} - z_{0i}\, z_{0j} \right)$$

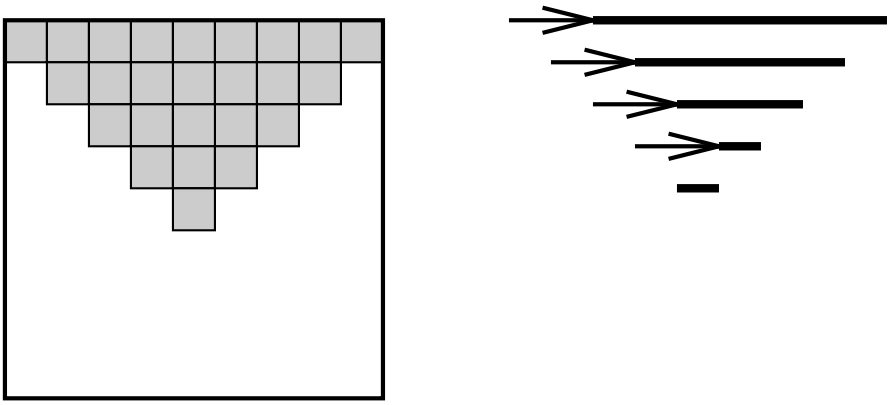
Abbildung 6.3: Erarbeitung der Rekursionsformel zur Inversion von Toeplitz-Matrizen.

Auf dem Vektor **z** ist zeilenweise das obere Viertel der Inversen abgespeichert (siehe Abb. 6.4).

Lösungsweg:Pseudocode

1. *Berechnung der Abbruchsschranke für den Zeilenindex:*
   `ih=INT((n-1)/2)` (`INT ...` *größter ganzer Wert*).
2. *Berechnung der notwendigen Dimensionierung akt:*
   `akt=INT((n*(n+2)+1)/4)`.
3. *Überprüfe ob der dimensionierte Platz des Vektors* **z** *ausreicht:*
   `IF akt>max+1 GOTO Ende:` 'DIMENSION DES VEKTORS **z** ZU KLEIN'.

Wolf-Dieter Schuh
ITGB, Uni Bonn



1	2		n	$n+1$	$n+2$	
z_{00}	z_{01}	...	$z_{0,n-1}$	z_{11}	z_{12}	...

Rechenvorschriften bei zeilenweiser Speicherung:

z_{ij} gespeichert, wenn $\begin{cases} i \le j & : \text{ sonst } z_{ji} \\ i \le \texttt{INT}(\frac{n-1}{2}) & : \text{ sonst } z_{n-j,n-i} \end{cases}$

Adresse von z_{ij} : $i(n-i+1)+(j-i)+1$

maximale Anzahl : $\texttt{INT}\left(\frac{n(n+2)+1}{4}\right)$

Abbildung 6.4: Speicherungswürdiger Inversenteil mit Rechenvorschriften.

4. *Berechnung der weiteren Zeilen:*
   Setze Adresse von z_{ij} : `adr=n`
   Setze Adresse von $z_{i-1,j-1}$: `vor=1`
   Initialisiere Zeilenindex : `i=0 .`
5. *Schleifenbeginn über alle Zeilen:* `i=i+1`
   `IF i>ih GOTO` `8:`.
6. *Schleife über die Spalten:*
```
DO j=i TO n-i-1
     z(adr)=z(vor)-(z(n-j)*z(n-i)-z(i)*z(j))/z(1)
     adr=adr+1 ; vor=vor+1
END DO j.
```
7. *Modifiziere die Adresse des Elements* $z_{i-1,j-1}$ *und bearbeite nächste Zeile:*
   `vor=vor+2`
   `GOTO` `5:`.
8. `Ende:` 'O.K. ENDE' .

Ressourcen:

Anzahl der Operationen : $\frac{3}{4}n^2 + \mathcal{O}(n)$

Platzbedarf : $\texttt{INT}\left(\frac{n(n+2)+1}{4}\right)$

Wolf-Dieter Schuh
ITGB, Uni Bonn

Bei näherer Betrachtung der Rekursionsformel (6.42) erkennt man, daß durch Division der nullten Zeile durch die Quadratwurzel von z_{00} (reell bei positiv definiten Matrizen), die Anzahl der Operationen pro Element von drei auf zwei reduziert werden kann. Für die Inversenglieder der zweiten Zeile $i = 1$ ist jedoch zu beachten, daß die ursprüngliche nullte Zeile im ersten Glied verwendet wird. Die Gesamtanzahl der Operationen mit vorhergehender Division der ersten Zeile und anschließender Multiplikation kann somit auf $\frac{1}{2}n^2 + \mathcal{O}(n)$ verringert werden. Bei Algorithmus 6.4 sind diese Modifikationen durchgeführt.

Algorithmus 6.4 : Inversion einer symmetrischen, positiv definiten Toeplitz-Matrix, zeitoptimierter Algorithmus

Aufgabenstellung:
Gegeben ist, die in einem Vektor **z** gespeicherte, erste Zeile der Inversen einer symmetrischen, positiv definiten Toeplitz-Matrix der Dimension n. Gesucht ist die Vervollständigung des oberen Viertels der Inversen, sodaß die Inverse auf Grund der Symmetrie und Persymmetrie eindeutig festlegt ist. Ist der bereitgestellte Vektor **z** mit der Dimension `max` zur Aufnahme der Inversen zu klein, so erfolgt der Abbruch.

Ausgangssituation:
Im Vektor **z** sind auf den Positionen 1 bis $n-1$ die Koeffizienten der ersten Inversenzeile gespeichert.

Achtung: Die Indizierung beginnt wie gewohnt bei 1 und geht bis n. Dies steht zwar im Gegensatz zur dargestellten Theorie, sorgt jedoch für eine einheitliche Bezeichnung innerhalb der Algorithmen.

Endsituation:
Auf dem Vektor **z** ist zeilenweise das obere Viertel der Inversen gespeichert (siehe Abb. 6.4).

Lösungsweg: Pseudocode

1. *Berechnung der Abbruchsschranke für den Zeilenindex:*
   `ih=INT((n+1)/2)` (`INT` ... *größter ganzer Wert*).
2. *Berechnung der notwendigen Dimensionierung akt:*
   `akt=INT(n*(n+2)+1)/4)`.
3. *Überprüfe ob der dimensionierte Platz des Vektors* **z** *ausreicht:*
   `IF akt>max+1 GOTO` `Ende:` 'DIMENSION DES VEKTORS **z** ZU KLEIN'.
4. *Überprüfe auf positve Definitheit:*
   `IF z(1)≤0 GOTO` `Ende:` 'TOEPLITZ-MATRIX NICHT POSITIV DEFINIT'.
5. *Wenn zumindest zwei Zeilen existieren, dann beginne mit der Berechnung:*
   `IF n=2 GOTO` `Ende:` 'O.K. ENDE'.
6. *Modifiziere die Zeile mit dem Index* 0 *zur schnelleren Berechnung:*
   `wurzel=SQRT(z(1))`
   `DO i=1 TO n ; z(i)=z(i)/wurzel ; END DO i .`

7. *Berechne die Zeile mit dem Index* 1*:*
   Setze Adresse : `adr=n`
   ```
   DO j=1 TO n-2
       z(adr)=z(j-1)*wurzel-z(n-j)*z(n-1)+z(1)*z(j)
       adr=adr+1
   END DO j .
   ```

8. *Berechne die weiteren Zeilen:*
   Setze Adresse von $z_{i-1,j-1}$: `vor=n`
   Initialisiere Zeilenindex : `i=1 .`

9. *Schleifenbeginn über alle Zeilen:*
   ```
   i=i+1
   IF i>ih GOTO 12:.
   ```

10. *Schleife über die Spalten:*
    ```
    DO j=i TO n-i-1
        z(adr)=z(vor)-z(n-j)*z(n-i)+z(i)*z(j)
        adr=adr+1 ; vor=vor+1
    END DO j .
    ```

11. *Modifiziere die Adresse des Elements* $z_{i-1,j-1}$ *und bearbeite nächste Zeile::*
    ```
    vor=vor+2
    GOTO 9:.
    ```

12. *Rückmodifikation der Zeile mit dem Index* 0*:*
    ```
    DO i=0 TO n-1 ; z(i)=z(i) wurzel ; END DO i.
    ```

13. `Ende:` 'O.K. ENDE'.

Ressourcen:

Anzahl der Operationen	: $\frac{1}{4}n^2 + \mathcal{O}(n)$
Platzbedarf	: `INT` $\left(\frac{n(n+2)+1}{4}\right)$

Bei allen Überlegungen sind wir von der Tatsache ausgegangen, daß eine Zeile ($i = 0$) der Inversen bekannt oder berechnet ist. Dies könnte durch Anwendung der Algorithmen 6.1 oder 6.2 auf einen Einheitsvektor $e^T = \left[\begin{array}{cccc} 1 & 0 & \ldots & 0 \end{array}\right]$ geschehen. Der günstigere Weg ist in den Algorithmen 6.1 und 6.2 schon vorgesehen. Da ein Vielfaches der nullten Inversenzeile für die Berechnung der Lösung ermittelt wird, kann durch Division des Vektors `z` durch `alpha` (siehe Schritt 9 bei Algorithmus 6.1 und Schritt 8 bei Algorithmus 6.2) die Inversenzeile gefunden werden. Zur alleinigen Ermittlung der Inversenzeile können die Berechnungen für die Lösung, also alle Rechenschritte für `x`, `delta` und `epsilon` weggelassen werden, wodurch sich die Anzahl der Rechenoperationen auf $n^2 + \mathcal{O}(n)$ vermindert. Summiert man die Anzahl der Operationen für die Berechnung der ersten Inversenzeile und die rekursive Inversenberechnung, so kann die Inverse mit $\frac{3}{2}n^2 + \mathcal{O}(n)$ Operationen berechnet werden.

Wolf-Dieter Schuh
ITGB, Uni Bonn

K. Eren (1980)[28], Seite 76-81 (FORTRAN-Programm Seite 145-146) verwendet den Weg über die rekursive Inversion zur Berechnung einer speziellen Lösung des Gleichungssystems. Nach der Berechnung des entsprechenden Inversengliedes und der Multiplikation mit dem Vektor der rechten Seite wird die Inversenzeile überspeichert durch die nachfolgende Zeile. Der benötigte Platzbedarf beträgt $4n$, die Anzahl der notwendigen Operationen ist mit $\frac{5}{2}n^2 + \mathcal{O}(n)$ gegeben.

Literatur:

[28] EREN Kamil (1980): Spectral Analysis of GEOS-3 Altimeter Data and Frequency Domain Collocation. Department of Geodetic Science, Report No. 297, Ohio State University, Columbus, Ohio, pp. 76-81, FORTRAN-Source pp. 145-146.

[38] GOLUB Gene H., Charles F. van LOAN (1983): Matrix Computations. North Oxford Academic, Oxford, pp. 130-132.

[90] TRENCH William F. (1964): An Algorithm for the Inversion of Finite Toeplitz Matrices. J. Soc. Indust. Appl. Math., Vol. 12, No. 3, pp. 515-522, September 1964.

6.1.3 Zirkulierende Systeme

Bei der Verwendung von zirkulierenden Matrizen ergibt sich auf Grund der periodischen Wiederkehr der Koeffizienten ein großer Vorteil bei den Berechnungen, da die Eigenvektoren $\mathbf{f}_j$ vorweg bekannt sind. Sie sind festgelegt durch die Lösungen des Systems $x^n - 1 = 0$ und werden als n-te Einheitswurzeln bezeichnet,

$$\varepsilon_j = \frac{1}{\sqrt{n}}\, e^{ij\frac{2\pi}{n}} \qquad j = 0, 1, \ \ldots, n-1; \qquad i = \sqrt{-1} \tag{6.43}$$

Die Eigenvektoren $\mathbf{f}$ ergeben sich durch

$$\mathbf{f}_j = \begin{bmatrix} 1 \\ \varepsilon_j \\ \varepsilon_j^2 \\ \cdot \\ \cdot \\ \cdot \\ \varepsilon_j^{n-1} \end{bmatrix} \qquad j = 0, 1, \ \ldots, n-1 \tag{6.44}$$

(siehe *R. Zurmühl et. al. (1984)[100]*, Seite 229). Die zirkulierende Matrix $\mathbf{C}$ kann somit leicht in die spektralen Anteile

$$\mathbf{C} = \mathbf{F}\ \mathbf{\Lambda}\ \mathbf{F}^* \tag{6.45}$$

aufgespalten werden, mit $\mathbf{\Lambda}$ als Diagonalmatrix (Spektralmatrix) und der unitären Matrix $\mathbf{F}$ (Modalmatrix), deren Spalten aus den Eigenvektoren $\mathbf{f}_j$ aufgebaut sind. $\mathbf{F}^*$

stellt die konjugiert transponierte Form der Modalmatrix $\mathbf{F}$ dar, die folgendermaßen gebildet wird

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \varepsilon_1 & \varepsilon_2 & \dots & \varepsilon_{n-1} \\ 1 & \varepsilon_1^2 & \varepsilon_2^2 & \dots & \varepsilon_{n-1}^2 \\ 1 & \varepsilon_1^3 & \varepsilon_2^3 & \dots & \varepsilon_{n-1}^3 \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 1 & \varepsilon_1^{n-1} & \varepsilon_2^{n-1} & \dots & \varepsilon_{n-1}^{n-1} \end{bmatrix} \tag{6.46}$$

Die Koeffizienten von $\mathbf{F}$ sind durch

$$f_{k\ell} = \frac{1}{\sqrt{n}} e^{ik\ell\frac{2\pi}{n}} \tag{6.47}$$

gegeben. Aus Symmetriegründen kann

$$f_{k\ell} = \frac{1}{\sqrt{n}} e^{ik\ell\frac{2\pi}{n}} = f_{\ell k} \tag{6.48}$$

festgehalten werden und der Übergang auf eine neue Bezeichnungsweise

$$\varepsilon^{k\ell} = f_{k\ell} = f_{\ell k} \tag{6.49}$$

wird somit ermöglicht. Die Matrix der Eigenvektoren (6.46) kann folgendermaßen geschrieben werden

$$\mathbf{F} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \varepsilon^1 & \varepsilon^2 & \dots & \varepsilon^{n-1} \\ 1 & \varepsilon^2 & \varepsilon^4 & \dots & \varepsilon^{2(n-1)} \\ 1 & \varepsilon^3 & \varepsilon^6 & \dots & \varepsilon^{3(n-1)} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 1 & \varepsilon^{n-1} & \varepsilon^{2(n-1)} & \dots & \varepsilon^{(n-1)(n-1)} \end{bmatrix} \tag{6.50}$$

6.1.3.1 Analytische Rechenvorschriften

Bedingt durch die besondere Form der Modalmatrix können für die Berechnung der Eigenwerte, der Inversen, der Matrizenmultiplikation und der Lösung eines Gleichungssystems besondere Formeln entwickelt werden.

Zunächst wenden wir uns der Berechnug der Eigenwerte zu. Durch Umformung von (6.45) unter Beachtung der unitären - beziehungsweise im reellen Fall der orthogonalen - Eigenschaft der Modalmatrix

$$\mathbf{F}^* \, \mathbf{F} = \mathbf{I} \tag{6.51}$$

errechnet sich die Spektralmatrix $\mathbf{\Lambda}$ mit

$$\mathbf{\Lambda} = \mathbf{F}^* \ \mathbf{C} \ \mathbf{F} \ . \tag{6.52}$$

Führt man die Matrizenmultiplikation in Indizesschreibweise durch, so erlangt man

$$\lambda_{pq} = \frac{1}{n} \sum_{r=0}^{n-1} e^{-ipr\frac{2\pi}{n}} \sum_{s=0}^{n-1} c_{|r-s|} \ e^{isq\frac{2\pi}{n}} \ . \tag{6.53}$$

Faßt man die Summationen zusammen

$$\lambda_{pq} = \frac{1}{n} \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} c_{|r-s|} \ e^{-ipr\frac{2\pi}{n}} \ e^{isq\frac{2\pi}{n}} \tag{6.54}$$

und formt die Potenzen durch Erweiterung um $e^{irq\frac{2\pi}{n}}$ um

$$\lambda_{pq} = \frac{1}{n} \sum_{r=0}^{n-1} \sum_{s=0}^{n-1} c_{|r-s|} \ e^{-i(r-s)q\frac{2\pi}{n}} \ e^{-ir(p-q)\frac{2\pi}{n}} \tag{6.55}$$

so kann man $r-s$ durch t ersetzen und erlangt

$$\lambda_{pq} = \frac{1}{n} \sum_{t=0}^{n-1} c_t \ e^{-itq\frac{2\pi}{n}} \sum_{r=0}^{n-1} e^{-ir(p-q)\frac{2\pi}{n}} \ . \tag{6.56}$$

Aufgrund der Orthogonalitätsbeziehungen ist die zweite Summe mit

$$\sum_{r=0}^{n-1} e^{-ir(p-q)\frac{2\pi}{n}} \qquad \begin{cases} = n \text{ für } p = q \\ = 0 \text{ für } p \neq q \end{cases} \tag{6.57}$$

bestimmt. Von den Koeffizienten der Matrix λ_{pq} sind nur die Diagonal elemente λ_k ungleich Null und können durch

$$\lambda_k = \sum_{t=0}^{n-1} c_t \ e^{-itk\frac{2\pi}{n}} \tag{6.58}$$

berechnet werden. Formeln dieses Typs treten in der Folge sehr oft in Erscheinung. Die Durchführung der Berechnung, dieser sowohl im Kontinuierlichen als auch im Diskreten auftretenden Form, wird als Fourier-Transformation bezeichnet. Die Spektralmatrix $\mathbf{\Lambda}$ wird dabei als Fourierttransformierte der Matrix oder genauer der ersten Zeile von $\mathbf{C}$ bezeichnet.

Da symmetrische, zirkulierende Matrizen durch Diskretisierung gerader (symmetrisch gegenüber der Nullachse) Grundfunktionen erzeugt werden und nur die reellen Cosinusanteile dieselben Symmetrieeigenschaften wiederspiegeln, müssen alle Eigenwerte reell

sein *O. Brigham (1974)[13]*, Seite 40). Wenn alle Eigenwerte reell sind, kann (6.58) umgeformt werden in

$$\lambda_k = \sum_{t=0}^{n-1} c_t \cos\left(tk \frac{2\pi}{n}\right) . \tag{6.59}$$

Die Inverse der Matrix $\mathbf{C}$ errechnet sich formal durch

$$\mathbf{C}^{-1} = \mathbf{F}\ \mathbf{\Lambda}^{-1}\ \mathbf{F}^* . \tag{6.60}$$

In Indizesschreibweise ergibt sich ein beliebiges Element $c_{pq}^{(-1)}$ Inversen $\mathbf{C}^{-1}$ mit

$$c_{pq}^{(-1)} = \frac{1}{n} \sum_{r=0}^{n-1} e^{ipr\frac{2\pi}{n}}\ \lambda_r^{-1}\ e^{-irq\frac{2\pi}{n}} \tag{6.61}$$

beziehungsweise zusammengefaßt

$$c_{pq}^{(-1)} = \frac{1}{n} \sum_{r=0}^{n-1} \lambda_r^{-1}\ e^{ir(p-q)\frac{2\pi}{n}} . \tag{6.62}$$

Man erkennt, daß das Inversenglied nur von der Differenz $q-p$ ab hängig ist und somit unabhängig von der Zeile wird. Es ergibt sich somit wieder eine reelle zirkulierende Matrix, deren Glieder durch

$$c_{|p-q|}^{(-1)} = \frac{1}{n} \sum_{t=0}^{n-1} \lambda_t^{-1} \cos\left(t_{|p-q|}\frac{2\pi}{n}\right) \tag{6.63}$$

errechnen lassen. Man beachte die Analogie zu den Formeln (6.58) und (6.62) bei der Berechnung der Eigenwerte. Die Formeln (6.58) und (6.62) unterscheiden sich nur durch den Multiplikationsfaktor, über den noch verfügt wird, und das Vorzeichen im Exponent. Während (6.58) die Transformation in den Spektralraum gewährleistet, ermöglicht (6.62) die Rücktransformation in den Ortsbereich. Da hier nur symmetrische Matrizen behandelt werden und somit nur der Realteil von Bedeutung ist, was durch (6.59) und (6.63) (siehe auch *R. Zurmühl et. al. (1984)[100]*, Seite 190) dokumentiert ist, tritt, bei entsprechender Berücksichtigung des Multiplikationsfaktor, kein Unterschied bei den Berechnungen auf.

Literatur:

[13] BRIGHAM E.Oran (1974): The Fast Fourier Transform. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 40ff.

[100] ZURMÜHL Rudolf, Sigurd FALK (1984): Matrizen und ihre technische Anwendungen (5. Auflage); Teil 1: Grundlagen. Springer-Verlag, Berlin-Göttingen-Heidelberg, Seiten 229-230.

6.1.3.2 Diskrete Fourier-Transformation (DFT)

Die Auswertung der schon mehrmals aufgetretene Form

$$x_k = \sum_{t=0}^{n-1} X_t \; e^{itk\frac{2\pi}{n}} \tag{6.64}$$

bezeichnet man als diskrete Fourier-Transformation (DFT). Für die numerische Durchführung dieser Transformation stehen sehr effiziente Verfahren zur Verfügung, denen der nächste Abschnitt gewidmet ist. In zusammenfassender, abgekürzter Schreibweise wird dieser Übergang von den Variablen X_t bzw. $\mathbf{X}$ im Spektralbereich zu den Variablen x_t bzw. $\mathbf{x}$ im Ortsbereich folgendermaßen dargestellt,

$$\mathbf{x} = \mathcal{F}\{\mathbf{X}\} \tag{6.65}$$

und als *Fouriersyntese* (Syntese der spektralen Einzelkomponenten) bezeichnet. Bezüglich der durch (6.50) eingeführten Matrix der Eigenvektoren $\mathbf{F}$ gilt somit der Zusammenhang

$$\mathbf{x} = \mathcal{F}\{\mathbf{X}\} = \sqrt{n}\; \mathbf{F}\; \mathbf{X}\;. \tag{6.66}$$

Betrachtet man die umgekehrte Operation, die wie erwähnt durch den negativen Exponenten, vom konstanten Faktor abgesehen, gegeben ist

$$X_t = \sum_{k=0}^{n-1} x_k \; e^{-itk\frac{2\pi}{n}} \tag{6.67}$$

so ist die Bezeichnung die inverse Operation kennzeichnende Formulierung

$$\mathbf{x} = \mathcal{F}^{-1}\{\mathbf{X}\} \tag{6.68}$$

üblich. Diese Operation dient der Überführung von Werten im Ortbereich in den Spektralbereich. Diese Operation der Aufspaltung und Zuordnung zu einzelnen Frequenzbereichen bezeichnet man als *Fourieranalyse* . Unter Verwendung der Matrix der Eigenvektoren $\mathbf{F}$ kann die Analyse auch durch unitäre (konjugiert transponierte) Form $\mathbf{F}^*$ vollzogen werden,

$$\mathbf{X} = \mathcal{F}^{-1}\{\mathbf{x}\} = \sqrt{n}\; \mathbf{F}^*\; \mathbf{x}\;. \tag{6.69}$$

Da nur reelle Vektoren $\mathbf{x}$ im Ortbereich verwendet werden, kann die inverse Operation rückgeführt werden auf die ursprüngliche Transformation $\mathcal{F}\{\;\}$ und Umwandlung des Ergebnisses in seine konjugiert komplexe Form

$$\mathbf{X} = \overline{\mathcal{F}\{\mathbf{x}\}} = \mathcal{F}^{-1}\{\mathbf{x}\}\;. \tag{6.70}$$

Unter Verwendung der eingeführten Sprachregelung können die in Formeln zur Berechnung der Eigenwerte (6.58) und der Inversen (6.62) eines zirkulierenden Systems sehr vereinfacht formuliert werden. Die Fourier-Transformation angewandt auf die ersten Zeile liefert die Eigenwerte (siehe Abschnitt 6.1.3.4). Die Berechnung der Inversen erfolgt durch zweimalige Anwendung der Fouriertransformation, wobei durch die erste Transformation die Eigenwerte berechnet werden. Die Anwendung der Fourier-Transformation auf die reziproken Eigenwerte liefert die n-fache Zeile der Inversen (siehe Abschnitt 6.1.3.5).

Betrachtet man die Anzahl der notwendigen Rechenoperationen für die zuletzt aufgezeigten Berechnungen, so scheinen sie auf den ersten Blick im Vergleich zu den effizienten Verfahren bei Toeplitz-Matrizen nicht sehr attraktiv. Zur Erlangung der Spektralmatrix ist eine, für die Berechnung der Inversen sind zwei diskrete Fourier-Transformationen durchzuführen. Die sture Auswertung der Formel (6.64) bringt eine quadratisch wachsende Anzahl von Rechenoperationen, wobei jeweils die Auswertung von Winkelfunktionen bewerkstelligt werden muß. Nur durch eine rasche numerische Berechnung der Fourier-Transformation ermöglicht eine effiziente Vorgangsweise über den Spektralraum.

6.1.3.3 Numerische Durchführung der Fourier-Transformation

Für die numerische Berechnung der Fourier-Transformation

$$x_k = \sum_{t=0}^{n-1} X_t \; e^{itk\frac{2\pi}{n}}$$

dargestellt durch

$$\mathbf{x} = \mathcal{F}\{\mathbf{X}\}$$

beziehungsweise

$$\mathbf{x} = \mathcal{F}\{\mathbf{X}\} = \sqrt{n}\; \mathbf{F}\; \mathbf{X}$$

können zwei Varianten gewählt werden (siehe Formeln (6.64), (6.65) und 6.66)). Der konventionelle Weg der Formelauswertung unter Ausnützung von Symmetrien (Abschnitt 6.1.3.3.1) und der moderne Zugang mit Hilfe der rekursiven Zerteilung des Gesamtproblems in kleinere Einzelprobleme (Abschnitt 6.1.3.3.2).

6.1.3.3.1 Diskrete Fourier-Transformation (DFT)

Die Spalten der Modalmatrix $\mathbf{F}$ bilden die Eigenvektoren $\mathbf{f}_j$, $j = 0, \ldots, n-1$. Diese können in den Realteil $\mathbf{v}_j$ und den Imaginärteil $\mathbf{u}_j$ auf gespaltet werden,

$$\mathbf{f}_j = \mathbf{v}_j + i\,\mathbf{u}_j \qquad i = \sqrt{-1}\;, \tag{6.71}$$

wobei die Vektoren $\mathbf{v}_j$ und $\mathbf{u}_j$ folgendermaßen definiert sind,

$$\begin{aligned} \mathbf{v}_j &= \tfrac{1}{\sqrt{n}} \left[\begin{array}{ccccc} 1 & \cos j\frac{2\pi}{n} & \cos 2j\frac{2\pi}{n} & \dots & \cos(n-1)j\frac{2\pi}{n} \end{array} \right]^T \\ \mathbf{u}_j &= \tfrac{1}{\sqrt{n}} \left[\begin{array}{ccccc} 1 & \sin j\frac{2\pi}{n} & \sin 2j\frac{2\pi}{n} & \dots & \sin(n-1)j\frac{2\pi}{n} \end{array} \right]^T . \end{aligned} \tag{6.72}$$

Bei Anwendung auf symmetrische zirkulierende Matrizen kann die komplexe Modalmatrix $\mathbf{F}$ durch die reelle Matrix $\mathbf{V}$ ersetzt werden. Die Spalten der orthogonalen Matrix $\mathbf{V}$ werden aus den reellen Anteilen von $\mathbf{f}$, $j = 0, \dots, n-1.$ gebildet, wobei der skalare Faktor aus Gründen der Normierung geändert werden muß. Der Faktor zur Normierung der Spalten kann mit Hilfe der Formel

$$\sum_{k=1}^{n} \cos^2 k\xi = \frac{n}{2} + \frac{\cos(n+1)\xi \sin n\xi}{2 \sin \xi} \tag{6.73}$$

(siehe z.B. *I. S. Gradshteyn et. al. (1980)[39]*, Formel 1.351.2b) abgeleitet werden. Setzt man für

$$\xi = j\frac{2\pi}{n} , \tag{6.74}$$

und beachtet

$$\cos n\xi = \cos 0\xi \qquad \text{für } \xi = j\frac{2\pi}{n} \tag{6.75}$$

so errechnet sich die quadratische Summe der Winkelfunktion mit

$$\sum_{k=0}^{n-1} \cos^2 kj\frac{2\pi}{n} = \left\{ \begin{array}{l} n \text{ für } \quad j = 0 \\ \frac{n}{2} \text{ für } j = 1, \dots, n-1 \end{array} \right. \tag{6.76}$$

Die normierten Eigenvektoren weisen somit folgende Form für $j = 0$

$$\mathbf{v}_0 = \frac{1}{\sqrt{n}} \left[\begin{array}{ccccc} 1 & 1 & 1 & \dots & 1 \end{array} \right]^T \tag{6.77}$$

beziehungsweise für $j = 1, \dots, n-1$

$$\mathbf{v}_j = \sqrt{\frac{2}{n}} \left[\begin{array}{ccccc} 1 & \cos j\frac{2\pi}{n} & \cos 2j\frac{2\pi}{n} & \dots & \cos(n-1)j\frac{2\pi}{n} \end{array} \right]^T \tag{6.78}$$

auf. Da bei symmetrischen Matrizen nur der gegenüber der Nullachse symmetrische Realteil $\mathbf{v}_j$ in Erscheinung tritt, muß der Wegfall des Imaginärteils $\mathbf{u}_j$ nur bei der Normierung berücksichtigen werden. Zur effizienten Berechnung der reellen Matrix $\mathbf{V}$ nützt man bestmöglich alle Symmetrieerscheingen der Cosinusfunktionen (siehe Abb. **??**).

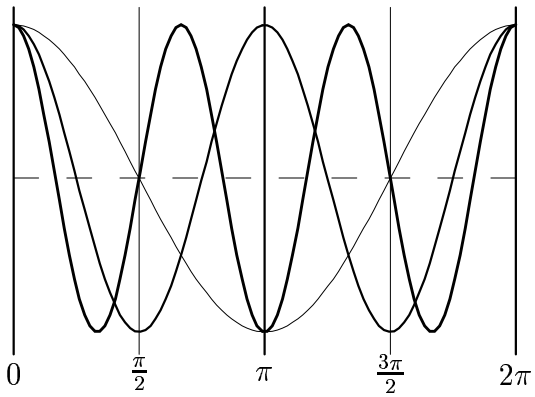


Abbildung 6.5: Symmetrieeigenschaften von $\cos\xi$, $\cos 2\xi$, $\cos 3\xi$.

Abb. 6.5 veranschaulicht die Symmetrieeigenschaften der Funktionen $\cos\xi$, $\cos 2\xi$, $\cos 3\xi$. Man erkennt hervorgerufen durch

$$\cos kj\frac{2\pi}{n} = \cos(n-k)j\frac{2\pi}{n} \tag{6.79}$$

die Symmetrie bezüglich π, die zur Berechnung benützt werden kann. Es sind daher jeweils der 0-te Wert, der mittlere Wert $\frac{n}{2}$ bei gerader Anzahl von n und die Werte $1, \ldots, \frac{n}{2}-1$ zu berechnen. Die zweite Hälfte der Eigenwerte und der Inversenglieder ist symmetrisch bezüglich der Achse π. Weiter Vorteile ergeben sich durch die Symmetrie von $\mathbf{V}$ was sich in der Vertauschbarkeit von k und j äußert (beachte jedoch den Skalierungsfaktor der ersten Spalte). Bei geradzahliger Anzahl von Stützstellen n kann weiters die Symmetrie bzw. Schiefsymmetrie bezüglich $\frac{\pi}{2}$ genützt werden.

Für ein rasches Verarbeitungsprogramm wird zur Berechnung der Funktionen $\cos(k+1)\alpha$ das mehrmaliges Aufschlagen der Winkelfunktion durch Anwendung des Additionstheorem

$$\begin{aligned}\cos(k\alpha+\alpha) &= \cos k\alpha\cos\alpha - \sin k\alpha\sin\alpha\\ \sin(k\alpha+\alpha) &= \sin k\alpha\cos\alpha + \cos k\alpha\sin\alpha\end{aligned} \tag{6.80}$$

vermieden. All diese Recheneinsparungen auf Grund der Symmetrieüberlegungen ändern aber nichts an quadratischer Ordnung $\mathcal{O}(n^2)$ mit der die Anzahl der Operation bei gegebener Dimension n wächst. Lediglich der Multiplikationsfaktor kann auf maximal $\frac{1}{4}$ verringert werden. Diese Art der Berechnung wurde von *C. Runge* im Jahre 1903 eingeführt (siehe *'Algorithmus von Runge', H. R. Schwarz (1988)[78]*, Seite 153-156). Für die Berechnung der Inversen sind somit unter Beachtung aller Symmetrien $\frac{1}{2}n^2 + \mathcal{O}(n)$ Operationen notwendig, was zwar einerseits den halben Aufwand einer Matrix-Vektor-Multiplikation bedeutet, aber andererseits noch nicht das Optimum bei zirkulierenden Matrizen darstellt.

6.1.3.3.2 Diskrete Fast Fourier Transformation (DFFT)

Mit dem Jahre 1965 trat eine entscheidende Neuerung bei der Verwendung der diskreten Fourier-Transformation ein. *J. W. Cooley* und *J. W. Tukey* zeigten erstmals einen extrem schnellen Weg zur Durchführung der diskreten Fourier Transformation, der als *'Fast Fourier Transformation'* (FFT) bezeichnet wird. Diese Entdeckung revolutionierte die gesamte Fouriertechnik. Von der umfangreichen Literatur kann *E. O. Brigham (1974)[13]* empfohlen werden. Kurze Einführungen und Zusammenfassungen sind bei *G. Strang (1986)[85]*, Seite 448-468, *G. Strang (1988)[86]*, Seite 183-193 oder *H. R. Schwarz (1988)[78]*, Seite 156-165 zu entnehmen.

Hier werden nur die grundsätzlichen Ideen zusammengefaßt und auf keine Details eingegangen. Die rechenintensive Haupttätigkeit bei der Ver wendung von Fourier-Techniken bildet die Transformation

$$x_k = \sum_{t=0}^{n-1} X_t \; e^{itk\frac{2\pi}{n}} \qquad k = 0, 1, \ldots, n-1$$

(siehe Formel (6.64)) für die $\frac{1}{4}n^2 + \mathcal{O}(n)$ Operationen benötigt werden. Spaltet man die Transformation in zwei Teile, wobei hier zur Vereinfachung eine gerade Anzahl n vorausgesetzt wird, so erhält man

$$x_k = \sum_{t=0}^{m-1} X_{2t} \; e^{i2tk\frac{2\pi}{n}} + \sum_{t=0}^{m-1} X_{2t+1} \; e^{i(2t+1)k\frac{2\pi}{n}} \qquad k = 0, 1, \ldots, n-1 \tag{6.81}$$

mit

$$m = \frac{n}{2} \; . \tag{6.82}$$

Führt man jetzt bei den Potenzen ebenfalls m ein

$$x_k = \sum_{t=0}^{m-1} X_{2t} \; e^{itk\frac{2\pi}{m}} + e^{ik\frac{2\pi}{n}} \sum_{t=0}^{m-1} X_{2t+1} \; e^{itk\frac{2\pi}{m}} \qquad k = 0, 1, \ldots, n-1 \tag{6.83}$$

und spalten den Vektor $\mathbf{x}_k$ ebenfalls in zwei Teile, so erhält man

$$\begin{aligned} x_k &= \sum_{t=0}^{m-1} X_{2t} \; e^{itk\frac{2\pi}{m}} + e^{ik\frac{2\pi}{n}} \sum_{t=0}^{m-1} X_{2t+1} \; e^{itk\frac{2\pi}{m}} \\ x_{k+m} &= \sum_{t=0}^{m-1} X_{2t} \; e^{it(k+m)\frac{2\pi}{m}} + e^{i(k+m)\frac{2\pi}{n}} \sum_{t=0}^{m-1} X_{2t+1} \; e^{it(k+m)\frac{2\pi}{m}} \end{aligned} \tag{6.84}$$

Da sich die Potenzen nur um Vielfache von 2π unterscheiden, gilt

$$e^{itk\frac{2\pi}{m}} = e^{it(k+m)\frac{2\pi}{m}} \tag{6.85}$$

beziehungsweise bei Unterscheidung von

$$\omega^k = e^{itk\frac{2\pi}{n}} = -e^{it(k+m)\frac{2\pi}{n}} \; . \tag{6.86}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

Somit ergibt sich

$$\begin{aligned} x_k &= \sum_{t=0}^{m-1} X_{2t}\, e^{itk\frac{2\pi}{m}} + \omega^k \sum_{t=0}^{m-1} X_{2t+1}\, e^{itk\frac{2\pi}{m}} \\ x_{k+m} &= \sum_{t=0}^{m-1} X_{2t}\, e^{itk\frac{2\pi}{m}} - \omega^k \sum_{t=0}^{m-1} X_{2t+1}\, e^{itk\frac{2\pi}{m}} \end{aligned} \tag{6.87}$$

Faßt man x' und x'' mit

$$\begin{aligned} x'_k &= \sum_{t=0}^{m-1} X_{2t}\, e^{itk\frac{2\pi}{m}} \\ x''_k &= \sum_{t=0}^{m-1} X_{2t+1}\, e^{itk\frac{2\pi}{m}} \end{aligned} \tag{6.88}$$

zusammen, wobei x'_k die fouriertransformierten geraden Elemente und x''_k die fouriertransformierten ungeraden Elemente sind. Die Fouriertransformation des n Elemente umfassenden Vektors $\mathbf{x}$ spaltet sich somit auf in zwei Fouriertransformationen der halben Dimension m

$$\begin{aligned} x_k &= x'_k + \omega^k\, x''_k \\ x_{k+m} &= x'_k - \omega^k\, x''_k \end{aligned} \qquad k = 0, 1, \ldots, m-1\ . \tag{6.89}$$

Die Ordnung des Rechenaufwandes verringert sich dabei von $\mathcal{O}(n^2)$ auf $\mathcal{O}(\frac{1}{4}n^2)$. Der wesentliche Vorteil dieses Verfahrens ist es, daß die Berechnung im gestrichenen System wieder durch Unterteilung be schleunigt werden kann. Führt man diese Strategie konsequent bis zu den Einzelelementen durch, so erreicht man im optimalen Fall (n ... Potenz von 2) eine Ordnung des Rechenaufwandes von $\frac{1}{2}\log_2 n$. Der Rechenablauf wird mit Hilfe von Block- und *Schmetterling*-Diagrammen veranschaulicht.

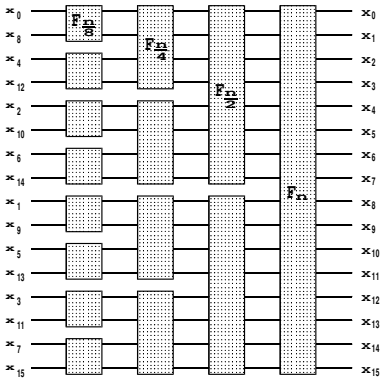


Abbildung 6.6: Blockdiagramm zur Übersichtsdarstellung des Rechenablaufs bei der diskreten-Fast-Fourier-Transformation (DFFT).

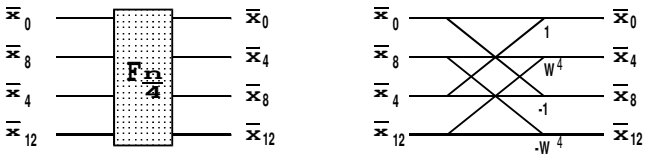


Abbildung 6.7: *Schmetterlingsdiagramm:* Übergang vom Block- auf das Detaildiagramm, Berechnung durch rekursive Anwendung von Formel (6.89).

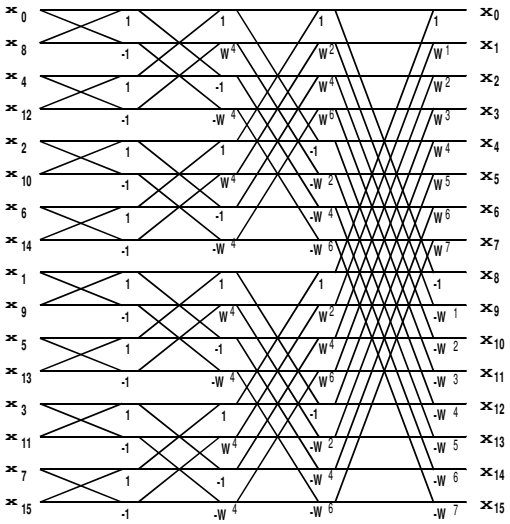


Abbildung 6.8: *Detail-Schmetterlingsdiagramm* zur Darstellung des Rechenablaufs bei der diskreten-Fast-Fourier-Transformation (DFFT).

Moderne Rechenalgorithmen, wie sie in verschiedenen Programmbibliotheken (z.B. IMSL, NAG) vorliegen, sind so aufgebaut, daß beliebige Dimensionen verarbeitet werden können und keine Beschränkung auf Vielfache von 2 notwendig ist. Die Rechengeschwindigkeit von diesen allgemeinen Fast-Fourier-Transformationen ist abhängig der größten enthaltenen Primzahl. Die sehr große Effizienz dieser Methode kann anhand der Beispiele in den nachfolgenden Kapiteln abgelesen werden.

Literatur:

[13] BRIGHAM E.Oran (1974): The Fast Fourier Transform. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 40ff.

[78] SCHWARZ Hans Rudolf (1988): Numerische Mathematik. Teubner, Stuttgart, 2. Auflage, Seite 156-165.

[85] STRANG Gilbert (1986): Introduction to Applied Mathematics. Wellesley-Cambridge-Press, Massachusetts, pp. 448-468.

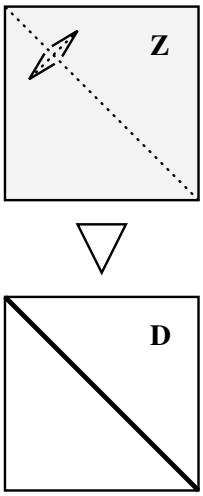
[86] STRANG Gilbert (1988): Linear Algebra and its Applications. Harcourt Brace Jovanovich, 3^{rd} Edition, San Diego, pp. 183-193.

6.1.3.4 Eigenwerte eines zirkulierenden Systems

Legt man die Fourier-Transformation (6.64) zugrunde, so ergeben sich die Eigenwerte durch die konjugiert komplexen Werte der fouriertransformierten ersten Zeile $\mathbf{C}_{0:0,n-1}$ des zirkulierenden Systems $\mathbf{C}$. Wegen der Symmetrie der hier behandelten Systeme wird die Bildung des konjugiert komplexen Anteils gegenstandslos da nur reelle Eigenwerte existieren, sodaß sich die Eigenwerte durch eine Fourier-Transformation

$$\mathbf{\Lambda} = \mathcal{F}\left\{\mathbf{C}_{0:0,n-1}\right\} \tag{6.90}$$

errechen lassen. Für die weiteren Ausführungen wird die Diagonalform der Spektralmatrix $\mathbf{\Lambda}$ stillschweigend teils als Matrix und teils als Vektor behandelt. Diese Ungenauigkeit sollte keine Problems bringen, da die komprimierte Speicherung in Rechenautomaten diese Vernachlässigung ohnehin aufzwingt und der ungenauen Bezeichungsart somit entgegen kommt. In schematischer Darstellung kann die Berechnung folgendermaßen zusammengefaßt (Algorithmus 6.5) und graphisch veranschaulicht (Abb. 6.9) werden.



Z zirkulierende Matrix ▽ Fourier-Transfomation

D Diagonalmatrix

Abbildung 6.9: Ablaufdiagramm zur Berechnung der Eigenwerte eines zirkulierenden, symmetrischen Gleichungssystems.
Eingabe: $\mathbf{C}$ Ausgabe: $\mathbf{\Lambda}$

Algorithmus 6.5 : Eigenwerte einer zirkulierenden, symmetrischen Matrix

Aufgabenstellung:
Gegeben ist in einem Vektor `c` gespeichert die erste Zeile einer zirkulierenden, symmetrischen Matrix $\mathbf{C}$ der Dimension n. Gesucht sind die Eigenwerte $\mathbf{\Lambda}$ der Matrix, die im Vektor `lambda` gespeichert werden.

Ausgangssituation:
Gegebene n-dimensionale Vektoren : `c(n)`
Gesuchte n-dimensionale Vektoren : `lambda(n)`

Endsituation:
Auf dem Vektor `lambda` sind die berechneten Eigenwerte gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{C}_{0,0:n-1}$			FFT(Op_1)	$\mathbf{\Lambda}$	1 FFT(n)	zirk, reell
Summe der Operatoren					1 FFT(n)	

Ressourcen:
Anzahl der Operationen : $\frac{1}{4} n \log_2 n + \mathcal{O}(1)$
Platzbedarf : $4n$

6.1.3.5 Inversion eines zirkulierenden Systems

Die Berechnung der Inversen (analytische Herleitung siehe Abschnitt 6.1.3.1) einer zirkulierenden Matrix kann durch zwei Fourier-Transformationen ausgedrückt werden (siehe Formel 6.62). Nach der Berechnung der Eigenwerte kann die Inverse $\mathbf{C}^{-1}$, oder genauer die erste Zeile der Inversen $\mathbf{C}_{0:0,n-1}$, alle anderen ergeben sich durch zyklische Vertauschung, durch

$$\mathbf{C}_{0:0,n-1} = \mathcal{F}\left\{\frac{1}{n}\mathbf{\Lambda}^{-1}\right\} \tag{6.91}$$

berechnet werden. Die Division durch die Anzahl der Koeffizienten n wird, wie angedeutet, üblicherweise im Spektralraum bewerkstelligt. Algorithmus 6.6 und Abb. 6.10 dokumentieren zusammenfassend den Rechenablauf. Nach der Berechnung der Eigenwerte durch die Anwendung einer Fourier-Transformation auf die erste Zeile der zirkulierenden Matrix wird die erste Zeile der Inversen durch eine weitere Fourier-Transformation, angewandt auf die reziproken Eigenwerte dividiert durch die Anzahl der Koeffizienten, errechnet. Alle weiteren Zeilen können aud Grund der erhalten gebliebenen zirkulierenden Struktur rekonstruiert werden.

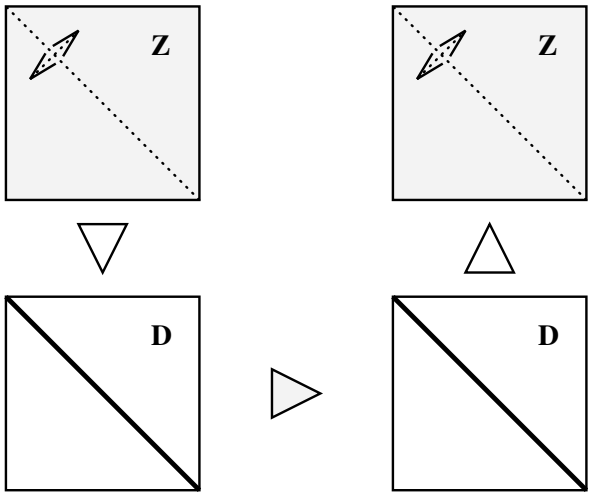


Abbildung 6.10: Ablaufdiagramm zur Berechnung der Inversen einer zirkulierenden, symmetrischen Matrix.

Eingabe: $\mathbf{C}$ Ausgabe: $\mathbf{C}^{-1}$

Algorithmus 6.6 : Inverse einer zirkulierenden, symmetrischen Matrix

Aufgabenstellung:
Gegeben ist in einem Vektor `c` gespeichert die erste Zeile einer zirkulierenden, symmetrischen Matrix $\mathbf{C}$ der Dimension n. Gesucht ist erste Zeile $\mathbf{C}^{-1}_{0,0:n-1}$ der Inversen $\mathbf{C}^{-1}$, die im Vektor `inv_c` gespeichert wird.

Ausgangssituation:
Gegebene n-dimensionale Vektoren : `c(n)`
Gesuchte n-dimensionale Vektoren : `inv_c(n)`

Endsituation:
Auf dem Vektor `inv_c` ist die erste Zeile der Inversen gespeichert. Die weiteren Zeilen sind durch zyklische Vertauschung berechenbar.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{C}_{0,0:n-1}$			$\texttt{FFT}(Op_1)$	$\mathbf{\Lambda}$	1 $\texttt{FFT}(n)$	reell
$\mathbf{\Lambda}$	n		Op_1^{-1}/Op_2	$\frac{1}{n}\mathbf{\Lambda}^{-1}$	n	reell, diag
$\frac{1}{n}\mathbf{\Lambda^{-1}}$			$\texttt{FFT}(Op_1)$	$\mathbf{C}^{-1}_{0,0:n-1}$	1 $\texttt{FFT}(n)$	
Summe der Operatoren					2 $\texttt{FFT}(n) + n$	

Ressourcen:

Anzahl der Operationen : $\frac{2}{4} n \log_2 n + n + \mathcal{O}1$

Platzbedarf : $4n$

6.1.3.6 Diskrete Faltung

Betrachtet man die Multiplikation einer zirkulierenden Matrix mit einem Vektor

$$\mathbf{C}\ \mathbf{x} = \mathbf{y} \tag{6.92}$$

so erkennt man in ausführlicher Form angeschrieben

$$\begin{bmatrix} c_0 & c_1 & ... & c_1 \\ c_1 & c_0 & ... & c_2 \\ c_2 & c_1 & ... & c_3 \\ ... & ... & & ... \\ ... & ... & & ... \\ c_1 & c_2 & ... & c_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ . \\ . \\ . \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} c_0x_0 & + & c_1x_1 & + & ... & + & c_1x_{n-1} \\ c_1x_0 & + & c_0x_1 & + & ... & + & c_2x_{n-1} \\ c_2x_0 & + & c_1x_1 & + & ... & + & c_3x_{n-1} \\ ... & & ... & & & & ... \\ ... & & ... & & & & ... \\ c_1x_0 & + & c_2x_1 & + & ... & + & c_0x_{n-1} \end{bmatrix} \tag{6.93}$$

daß hier die diskrete Form einer Faltung vorliegt. Analog zur Faltung im kontinuierlichen Fall (siehe z.B. *M. Wieser (1989)[97]*, Seite 20) kann die Operation im Spektralraum in eine einfache Multiplikation von Vektoren übergeführt werden.

Spaltet man die zirkulierende Matrix unter Verwendung von (6.45) in ihre spektalen Komponenten auf

$$\mathbf{F}\ \mathbf{\Lambda}\ \mathbf{F}^*\ \mathbf{x} = \mathbf{y} \tag{6.94}$$

und multipliziert das System mit der Matrix $\mathbf{F}^*$ so ergibt sich unter Beachtung der Identität (6.51)

$$\mathbf{\Lambda}\ \mathbf{F}^*\ \mathbf{x} = \mathbf{F}^*\ \mathbf{y} \tag{6.95}$$

Faßt man die Matrix-Vektor Produkte in den Vektoren

$$\mathbf{X} = \mathbf{F}^*\ \mathbf{x} \tag{6.96}$$

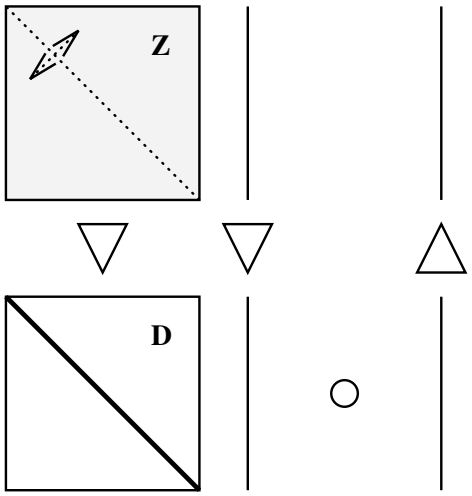


Abbildung 6.11: Ablaufdiagramm zur Berechnung der diskreten Faltung mit einer zirkulierenden, symmetrischen Matrix.
Eingabe: $\mathbf{C}$, $\mathbf{x}$ Ausgabe: $\mathbf{y}$

und

$$\mathbf{Y} = \mathbf{F}^* \, \mathbf{y} \tag{6.97}$$

zusammen, so vereinfacht sich (6.95) zu

$$\mathbf{\Lambda} \, \mathbf{X} = \mathbf{Y} \; . \tag{6.98}$$

Da die Spektralmatrix $\mathbf{\Lambda}$ eine Diagonalmatrix ist, kann der Vektor $\mathbf{Y}$ als Multiplikation der Einzelkomponenten von $\mathbf{\Lambda}$ und $\mathbf{X}$ in n Schritten ermittelt werden. Die Formeln der Berechnung von $\mathbf{X}$

$$X_t = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} x_k \; e^{-itk\frac{2\pi}{n}} \tag{6.99}$$

beziehungsweise die Rückrechnung von $\mathbf{y}$

$$y_k = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} Y_t \; e^{itk\frac{2\pi}{n}} \tag{6.100}$$

weisen wieder die spezielle Form der Fourier-Rücktransformation bzw. der Fourier-Transformation auf. Die diskrete Faltung, also die Multiplikation einer zirkulierenden Matrix mit einem Vektor, kann durchgeführt werden, indem man die Matrix und den Vektor mit Hilfe der Fourier-Transformation in den Spektralraum transferiert. Die Faltung zerfällt somit in eine Multiplikation der Einzelkomponenten der zwei Vektoren. Zum Ergebnis gelangt man durch die Rücktransformation des Vektors mit den Produkten (Algorithmus 6.7 bzw. Abb. 6.11).

Algorithmus 6.7 : Diskrete Faltung mit einer zirkulierenden, symmetrischen Matrix

Aufgabenstellung:
Gegeben ist in einem Vektor `c` gespeichert die erste Zeile einer zirkulierenden, symmetrischen Matrix $\mathbf{C}$ der Dimension n und eine Vektor $\mathbf{x}$. Gesucht ist der Vektor $\mathbf{y}$ der durch $\mathbf{C}\,\mathbf{x}$ errechnet wird.

Ausgangssituation:
Gegebene n-dimensionale Vektoren : `c(n)`, `x(n)`
Gesuchte n-dimensionale Vektoren : `y(n)`

Endsituation:
Auf dem Vektor `y(n)` ist die rechte Seite $\mathbf{y}$ gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{C}_{0,0:n-1}$			$\mathtt{FFT}(Op_1)$	$\mathbf{\Lambda}$	1 $\mathtt{FFT}(n)$	reell
$\mathbf{x}$			$\mathtt{FFT}(Op_1)$	$\sqrt{n}\mathbf{X}$	1 $\mathtt{FFT}(n)$	komplex
$\mathbf{\Lambda}$	$\sqrt{n}\mathbf{X}$	n	$Op_1 * \overline{Op_2}/Op_3$	$\frac{1}{\sqrt{n}}\mathbf{Y}$	2 n	komplex
$\frac{1}{\sqrt{n}}\mathbf{Y}$			$\mathtt{FFT}(Op_1)$	$\mathbf{y}$	1 $\mathtt{FFT}(n)$	reell
Summe der Operatoren					3 $\mathtt{FFT}(n) + 2\ n$	

Ressourcen:
Anzahl der Operationen : $\frac{3}{4} n \log_2 n + 2\ n + \mathcal{O}(1)$
Platzbedarf : $\mathcal{O}(n)$

Literatur:

[97] WIESER Manfred (1988): Theoretische Untersuchungen spektraler Methoden zur Analyse regelmässiger Strukturen am Beispiel der Fouriertransformation geodätischer Netze. Mitteilungen der geodätischen Institute der TU Graz, Folge 60, Seite 20ff.

6.1.3.7 Lösung eines zirkulierenden Systems

Ein ähnlicher Ablauf ist möglich, wenn die Lösung des Gleichungssystems (6.92) gesucht wird. Der Lösungsvektor x läßt sich ermitteln durch

$$\mathbf{x} = \mathbf{C}^{-1}\ y\ . \tag{6.101}$$

Verwendet man anstelle der Inversen die spektale Darstellung (6.60), so erlangt man

$$\mathbf{x} = \mathbf{F}\ \mathbf{\Lambda}^{-1}\ \mathbf{F}^*\ \mathbf{y}\ . \tag{6.102}$$

Formt man das System durch Multiplikation von links mit $\mathbf{F}^*$ unter Beachtung der Identität $\mathbf{F}^*\mathbf{F} = \mathbf{I}$ (6.51) um

$$\mathbf{F}^*\ \mathbf{x} = \mathbf{\Lambda}^{-1}\ \mathbf{F}^*\ \mathbf{y}\ . \tag{6.103}$$

und führt wieder die zusammenfassenden Bezeichnungen X und Y

$$\begin{aligned} \mathbf{X} &= \mathbf{F}^*\ \mathbf{x} \\ \mathbf{Y} &= \mathbf{F}^*\ \mathbf{y} \end{aligned} \tag{6.104}$$

ein, so erlangt man mit

$$\mathbf{X} = \mathbf{\Lambda}^{-1}\ \mathbf{Y} \tag{6.105}$$

ein System, wo nur mehr die Inverse einer Diagonalmatrix mit einem Vektor zu multiplizieren ist, was in n Schritten durchführbar ist. Die Berechnung von

$$Y_t = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} y_k\ e^{-itk\frac{2\pi}{n}} \tag{6.106}$$

beziehungsweise die Rückrechnung von x beziehungsweise die Rückrechnung von $\mathbf{y}$

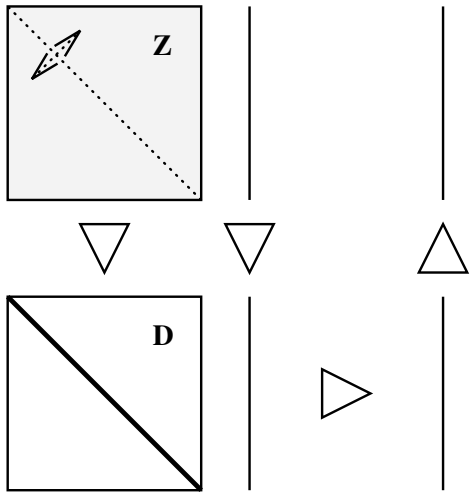
$$x_k = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} X_t\ e^{itk\frac{2\pi}{n}} \tag{6.107}$$

können durch Fourier-Transformationen bewerkstelligt werden (Algorithmus 6.8 bzw. Abb. 6.12).

Algorithmus 6.8 : Lösung eines zirkulierenden, symmetrischen Systems

Aufgabenstellung:
Gegeben ist ein Gleichungssystem durch die in einem Vektor `c` gespeichert erste Zeile einer zirkulierenden, symmetrischen Matrix $\mathbf{C}$ der Dimension n und dem Konstantenvektor $\mathbf{y}$. Gesucht ist die Lösung $\mathbf{x}$ des Systems.

Ausgangssituation:
Gegebene n-dimensionale Vektoren : `c(n)`, `y(n)`
Gesuchte n-dimensionale Vektoren : `x(n)`



Z zirkulierende Matrix $\triangledown\,\triangle$ Fourier-Transfomationen

D Diagonalmatrix $\triangleright$ Gleichungslösung

Abbildung 6.12: Ablaufdiagramm zur Berechnung der Lösung eines zirkulierenden, symmetrischen Gleichungssystems.

Eingabe: $\mathbf{C}$, $\mathbf{y}$ Ausgabe: $\mathbf{x}$

Endsituation:
Auf dem Vektor `x(n)` ist die Lösung $\mathbf{x}$ des Gleichungssystems gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{C}_{0,0:n-1}$			$\texttt{FFT}(Op_1)$	$\mathbf{\Lambda}$	1 $\texttt{FFT}(n)$	reell
$\mathbf{y}$			$\texttt{FFT}(Op_1)$	$\sqrt{n}\mathbf{Y}$	1 $\texttt{FFT}(n)$	komplex
$\mathbf{\Lambda}$	$\sqrt{n}\mathbf{Y}$	n	$Op_1^{-1} * \overline{Op_2}/Op_3$	$\frac{1}{\sqrt{n}}\mathbf{X}$	2 n	komplex
$\frac{1}{\sqrt{n}}\mathbf{X}$			$\texttt{FFT}(Op_1)$	$\mathbf{y}$	1 $\texttt{FFT}(n)$	reell
Summe der Operatoren					3 $\texttt{FFT}(n) + 2\ n$	

Ressourcen:
Anzahl der Operationen : $\frac{3}{4} n \log_2 n + 2\ n + \mathcal{O}(1)$
Platzbedarf : $\mathcal{O}(n)$

Wolf-Dieter Schuh
ITGB, Uni Bonn

6.1.4 Einbringung von beliebigen zusätzlichen Bedingungen bei zirkulierenden Systemen

Vielfach verhindern die strengen Vorschriften einer zirkulierenden Struktur deren Einsatz bei Modellbildungen. Eine regelmäßige Anordnung liegt zwar sehr oft über weite Teile des Modells vor, aber unregelmäßige Randerscheinungen vereiteln den Einsatz dieser effizienten Lösungsmethoden. Bei der Möglichkeit, eine strenge regelmäßige Lösung als Näherung einzubringen und durch iterative Verfahren die unregelmäßigen Randbedingungen zu berücksichtigen, ist Folgendes zu beachten. Wird durch die eingebrachten Bedingungen das Verhalten der Eigenvektoren stark verändert, so ist bei der iterativen Lösung mit einer schlechten Konvergenz zu rechnen. Dies betrifft vor allem jene Eigenvektoren, die für die Einbindung in feste Randbedingungen zuständig sind und daher die Lagerung widerspiegeln. Die Änderungen des Verhaltens dieser Eigenvektoren sind durch iterative Verfahren nur schwer erfaßbar.

Hier soll aus diesem Grunde die Berücksichtugung von beliebigen Randbedingungen auf strenge Art durchgeführt werden. Der Vorteil einer zugrundeliegenden zirkuliernden Struktur wird speziell ausgenützt, wodurch sehr effiziente Berechnungen ermöglicht werden. Im Folgenden werden zwei Varianten behandelt. Die erste Variante ermöglicht das Hinzufügen von beliebigen Bedingungen zu einer zirkuliernden Matrix. Die zweite Variante widmet sich dem Abspalten von Teilen des zirkulierenden Systems. Das daraus resultierende Toeplitz-System wird für die folgenden Abschnitte von Bedeutung sein.

6.1.4.1 Hinzufügen von Randbedingungen zu einem zirkulierenden System

Ausgebend von einem in Blockform angeschriebenen symmetrischen Gleichungssystem

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} \tag{6.108}$$

können die im Abschnitt 1.3 für den nichtsymmetrischen Fall entwickelten Formeln für die Inverse

$$\begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}\mathbf{A}_{12}^T\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1} \\ -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}\mathbf{A}_{12}^T\mathbf{A}_{11}^{-1} & \left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1} \end{bmatrix} \tag{6.109}$$

und die Lösung

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}^{-1}\left(\mathbf{a}_1 - \mathbf{A}_{12}\mathbf{x}_2\right) \\ \left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\right)^{-1}\left(\mathbf{a}_2 - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{a}_1\right) \end{bmatrix} \tag{6.110}$$

direkt übernommen werden. Für die weiteren Betrachtungen wird die Dimension der quadratischen Matrix $\mathbf{A}_{11}$ mit n_1 und der quadratischen Matrix $\mathbf{A}_{22}$ mit n_2 festgesetzt.

Setzt man nun voraus, daß der Matrizenblock $\mathbf{A}_{11}$ eine zirkuliernde Matrix darstellt, deren spektrale Darstellung durch

$$\mathbf{A}_{11} = \mathbf{F}_{11}\mathbf{\Lambda}_{11}\mathbf{F}^*{}_{11} \tag{6.111}$$

gegeben ist so kann die Inverse $\mathbf{A}_{11}^{-1}$ leicht durch zwei Fouriertransformationen (siehe Abschnitt 6.1.3.5) berechnet werden,

$$\mathbf{A}_{11}^{-1} = \mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}\mathbf{F}^*{}_{11} \ . \tag{6.112}$$

Überlegt man sich die Anzahl der notwendigen Operationen zur Lösung des Gesamtsystems bei Verwendung der oben zusammengefaßten Formeln, so erkennt man, daß die Ordnungen $\mathcal{O}(n_1^2 n_2)$, $\mathcal{O}(n_1 n_2^2)$ und $\mathcal{O}(n_2^3)$, die dominierenden Größen darstellen. Algorithmus 6.9 gibt eine detailierte Darstellung der notwendigen Einzelschritte.

Algorithmus 6.9 : Lösung eines symmetrischen Systems unter Nutzung des zirkulierenden Blocks $\mathbf{A}_{11}$

Aufgabenstellung:
Gegeben ist ein symmetrisches Gleichungssystem in Blockform,

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} ,$$

wobei $\mathbf{A}_{11}$ eine zirkulierende Matrix der Dimension n_1 darstellt. $\mathbf{A}_{22}$ bildet eine symmetrische Matrix der Dimension n_2 und $\mathbf{A}_{12}$ ist beliebig. Gesucht ist die Lösung $\mathbf{x}$ des Systems.

Ausgangssituation:
Gegebene Matrizen : $\mathbf{A}_{11}(n_1, n_1)$, $\mathbf{A}_{12}(n_1, n_2)$, $\mathbf{A}_{22}(n_2, n_2)$
Gegebene Vektoren : $\mathbf{a}_1(n_1)$, $\mathbf{a}_2(n_2)$
Gesuchte Vektoren : $\mathbf{x}_1(n_1)$ $\mathbf{x}_2(n_2)$

Endsituation:
Auf den Vektoren $\mathbf{x}_1$ und $\mathbf{x}_2$ ist die Lösung $\mathbf{x}$ des Gleichungssystems gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{A}_{11}$			$\texttt{INV}(Op_1)$	$\mathbf{A}_{11}^{-1}$	$2\ \texttt{FFT}(n_1)+n_1$	zirk.
$\mathbf{A}_{12}^T$	$\mathbf{A}_{11}^{-1}$		$Op_1{*}Op_2$	$\mathbf{A}_{12}^T\ \mathbf{A}_{11}^{-1}$	$n_1^2 n_2$	
$\mathbf{A}_{22}$	$\mathbf{A}_{12}^T\ \mathbf{A}_{11}^{-1}$	$\mathbf{A}_{12}$	$Op_1 - Op_2{*}Op_3$	$\hat{\mathbf{A}}_{22}$	$\frac{1}{2}n_1 n_2(n_2+1)$	symm.
$\mathbf{a}_2$	$\mathbf{A}_{12}^T\ \mathbf{A}_{11}^{-1}$	$\mathbf{a}_1$	$Op_1 - Op_2{*}Op_3$	$\hat{\mathbf{a}}_2$	$n_1 n_2$	
$\hat{\mathbf{A}}_{22}$	$\hat{\mathbf{a}}_2$		$\texttt{SOL}(Op_1, Op_2)$	$\mathbf{x}_2$	$\frac{1}{6}(n_2^3+9n_2^2+2n_2)$	symm.
$\hat{\mathbf{a}}_1$	$\mathbf{A}_{12}$	$\mathbf{x}_2$	$Op_1 - Op_2{*}Op_3$	$\hat{\mathbf{a}}_1$	$n_1 n_2$	
$\mathbf{A}_{11}^{-1}$	$\hat{\mathbf{a}}_1$		$Op_1{*}Op_2$	$\mathbf{x}_1$	$n_1 n_2$	
Summe der Operatoren				$2\ \texttt{FFT}(n_1)+n_1\ +$ $\frac{1}{6}(n_2^3+9n_2^2+2n_2)\ +$ $n_1(n_1 n_2+\frac{7}{2}n_2+n_1+\frac{1}{2}n_2^2)$		$\texttt{INV}_{zir}(n_1)$ $\texttt{SOL}_{sym}(n_2)$
Zwischenergebnisse				$\hat{\mathbf{A}}_{22} = (\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{A}_{12})$ $\hat{\mathbf{a}}_2 = (\mathbf{a}_2 - \mathbf{A}_{12}^T\mathbf{A}_{11}^{-1}\mathbf{a}_1)$ $\hat{\mathbf{a}}_1 = (\mathbf{a}_1 - \mathbf{A}_{12}\mathbf{x}_2)$		

Ressourcen:

Anzahl der Operationen : $\frac{1}{6}n_2^3 + n_1^2 n_2 + \frac{1}{2}n_1 n_2^2 + \mathcal{O}(n_1^2, n_1 n_2, n_2^2)$

Platzbedarf : $\mathcal{O}(n_1 n_2, n_2^2)$

Eine bessere Nutzung der Vorteile der zirkulierenden Struktur von $\mathbf{A}_{11}$ wird erreicht, wenn nicht sofort auf die vollbesetzte Inverse der Matrix $\mathbf{A}_{11}$ übergegangen wird, sondern mit der spekralen Darstellung $\mathbf{\Lambda}_{11}$ und $\mathbf{F}_{11}$ weitergearbeitet wird. Bedingt durch die Diagonalstruktur von $\mathbf{\Lambda}_{11}$ ergeben sich einige rechentechnische Vorteile. Der Formelapparat für die Lösung des Systems wird durch Einsetzen der spektralen Darstellung von $\mathbf{A}_{11}^{-1}$ gewonnen,

$$\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}\left(\mathbf{F}_{11}^*\mathbf{a}_1 - \mathbf{F}_{11}^*\mathbf{A}_{12}\mathbf{x}_2\right) \\ \left(\mathbf{A}_{22} - \mathbf{A}_{12}^T\mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}\mathbf{F}_{11}^*\mathbf{A}_{12}\right)^{-1}\left(\mathbf{a}_2 - \mathbf{A}_{12}^T\mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}\mathbf{F}_{11}^*\mathbf{a}_1\right) \end{bmatrix} . \quad (6.113)$$

Während der Vektor der fouriertransformierten Koeffizienten $\mathbf{a}_1$ äquivalent ist zu

$$\mathcal{F}^{-1}\{\mathbf{a}_1\} = \sqrt{n_1}\mathbf{F}_{11}^*\mathbf{a}_1 \quad (6.114)$$

können analog dazu auch die Spalten der Matrix $\mathbf{A}_{12}$ transformiert werden, wobei die gleiche zusammenfassende Schreibweise benützt wird,

$$\mathcal{F}^{-1}\{\mathbf{A}_{12}\} = \sqrt{n_1}\mathbf{F}_{11}^*\mathbf{A}_{12} \quad (6.115)$$

beziehungsweise die konjugiert komplexe Form

$$\mathcal{F}\{\mathbf{A}_{12}\} = \sqrt{n_1}\mathbf{A}_{12}^T\mathbf{F}_{11} \ . \tag{6.116}$$

Die detailierte Darstellung des Lösungsweges (Algorithmus 6.10) zeigt vor allem bei $n_2 \ll n_1$ eine wesentliche Effizientssteigerung.

Algorithmus 6.10 : Lösung eines symmetrischen Systems unter Nutzung des zirkulierenden Blocks $\mathbf{A}_{11}$ (optimiert)

Aufgabenstellung:
Gegeben ist ein symmetrisches Gleichungssystem in Blockform,

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} \ ,$$

wobei $\mathbf{A}_{11}$ eine zirkulierende Matrix der Dimension n_1 darstellt. $\mathbf{A}_{22}$ bildet eine symmetrische Matrix der Dimension n_2 und $\mathbf{A}_{12}$ ist beliebig. Gesucht ist die Lösung $\mathbf{x}$ des Systems.

Ausgangssituation:
Gegebene Matrizen : $\mathbf{A}_{11}(n_1, n_1)$, $\mathbf{A}_{12}(n_1, n_2)$, $\mathbf{A}_{22}(n_2, n_2)$
Gegebene Vektoren : $\mathbf{a}_1(n_1)$, $\mathbf{a}_2(n_2)$
Gesuchte Vektoren : $\mathbf{x}_1(n_1)$ $\mathbf{x}_2(n_2)$

Endsituation:
Auf den Vektoren $\mathbf{x}_1$ und $\mathbf{x}_2$ ist die Lösung $\mathbf{x}$ des Gleichungssystems gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{A}_{11}$			$\texttt{FFT}(Op_1)$	$\mathbf{\Lambda}_{11}$	$1\ \texttt{FFT}(n_1)$	reell
$\mathbf{a}_1$			$\texttt{FFT}(Op_1)$	$\mathcal{F}\{\mathbf{a}_1\}$	$1\ \texttt{FFT}(n_1)$	komplex
$\mathbf{A}_{12}$			$\texttt{FFT}(Op_1)$	$\mathcal{F}\{\mathbf{A}_{12}\}$	$n_2\ \texttt{FFT}(n_1)$	komplex
$\mathcal{F}\{\mathbf{A}_{12}\}$	n_1	$\mathbf{\Lambda}_{11}$	$Op_1/(Op_2{*}Op_3)$	$\mathbf{H}_1$	$2n_1n_2+n_1$	komplex
$\mathbf{A}_{22}$	$\mathbf{H}_1$	$\mathcal{F}\{\mathbf{A}_{12}\}$	$Op_1-Op_2*\overline{Op_3}$	$\hat{\mathbf{A}}_{22}$	$n_1n_2(n_2+1)$	symm., reell
$\mathbf{a}_2$	$\mathcal{F}\{\mathbf{A}_{12}\}$	$\mathcal{F}\{\mathbf{a}_1\}$	$Op_1-Op_2{*}\overline{Op_3}$	$\hat{\mathbf{a}}_2$	$2n_1n_2$	reell
$\hat{\mathbf{A}}_{22}$	$\hat{\mathbf{a}}_2$		$\texttt{SOL}(Op_1,Op_2)$	$\mathbf{x}_2$	$\frac{1}{6}(n_2^3+9n_2^2+2n_2)$	symm.
$\mathcal{F}\{\mathbf{a}_1\}$	$\mathcal{F}\{\mathbf{A}_{12}\}$	$\mathbf{x}_2$	$Op_1-\overline{Op_2}{*}Op_3$	$\mathbf{H}_2$	$2n_1n_2$	komplex
n_1	$\mathbf{\Lambda}_{11}$	$\mathbf{H}_2$	$Op_1/(Op_2{*}Op_3)$	$\mathcal{F}\{\mathbf{x}_1\}$	$2n_1$	komplex
$\mathcal{F}\{\mathbf{x}_1\}$			$\texttt{FFT}(Op_1)$	$\mathbf{x}_1$	$1\ \texttt{FFT}(n_1)$	reell
Summe der Operatoren				$(3+n_2)\ \texttt{FFT}(n_1)+$ $\frac{1}{6}(n_2^3+9n_2^2+2n_2)+$ $n_1(n_2^2+7n_2+3)$		$\texttt{SOL}_{sym}(n_2)$
Zwischenergebnisse				$\mathbf{H}_1 = \frac{1}{\sqrt{n_1}}\mathbf{A}_{12}^T\mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}$ $\hat{\mathbf{A}}_{22} = (\mathbf{A}_{22}-\mathbf{A}_{12}^T\mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}\mathbf{F}_{11}^*\mathbf{A}_{12})$ $\hat{\mathbf{a}}_2 = (\mathbf{a}_2-\mathbf{A}_{12}^T\mathbf{F}_{11}\mathbf{\Lambda}_{11}^{-1}\mathbf{F}_{11}^*\mathbf{a}_1)$ $\mathbf{H}_1 = (\mathbf{F}_{11}^*\mathbf{a}_1-\mathbf{F}_{11}^*\mathbf{A}_{12}\mathbf{x}_2)$		

Ressourcen:

Anzahl der Operationen : $\frac{1}{6}n_2^3+n_1n_2^2+n_2\texttt{FFT}(n_1)+\mathcal{O}(n_1n_2)$

Platzbedarf : $\mathcal{O}(n_1n_2,n_2^2)$

6.1.4.2 Elimination eines Blocks aus einem zirkuliernden System

Im Gegensatz zum vorhergehenden Abschnitt, wo eine zirkulierendes System durch beliebige, lineare Gleichungen erweitert wurde, wird hier davon ausgegangen, daß ein symmetrisches, zirkulierendes System

$$\mathbf{C}\xi=\mathbf{a} \tag{6.117}$$

vorliegt und nur die Lösung des Teilproblems von Interesse ist. Dazu wird das zirkulierende System in Blöcke zerlegt, wofür wieder die gewohnte Darstellung

$$\begin{bmatrix}\mathbf{A}_{11} & \mathbf{A}_{12}\\ \mathbf{A}_{12}^T & \mathbf{A}_{22}\end{bmatrix}\begin{bmatrix}\xi_1\\ \xi_2\end{bmatrix}=\begin{bmatrix}\mathbf{a}_1\\ \mathbf{a}_2\end{bmatrix} \tag{6.118}$$

gewählt wird. Die quadratischen Matrizen $\mathbf{A}_{11}$ und $\mathbf{A}_{22}$ mit den Dimensionen n_1 bzw. n_2 bilden somit symmetrische Toeplitzmatrizen. Der aus ξ_1 und ξ_2 zusammengestzte Vektor ξ bildet die Lösung des zirkulierenden Systems. Bezeichnet man die Teilblöcke der Inversen durch

$$\begin{bmatrix} \mathbf{A}_{11}^{(-1)} & \mathbf{A}_{12}^{(-1)} \\ \left(\mathbf{A}_{12}^{(-1)}\right)^T & \mathbf{A}_{22}^{(-1)} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}^{-1} \tag{6.119}$$

(beachte $\mathbf{A}_{11}^{(-1)}$ Teilblock der Gesamtinversen, $\mathbf{A}_{11}^{-1}$ Inverse des Blocks $\mathbf{A}_{11}$) so kann die Lösung des zirkulierenden Systems durch

$$\begin{aligned} \xi_1 &= \mathbf{A}_{11}^{(-1)} \mathbf{a}_1 + \mathbf{A}_{12}^{(-1)} \mathbf{a}_2 \\ \xi_2 &= \left(\mathbf{A}_{12}^{(-1)}\right)^T \mathbf{a}_1 + \mathbf{A}_{22}^{(-1)} \mathbf{a}_2 \end{aligned} \tag{6.120}$$

dargestellt werden. Gesucht ist die Lösung $\mathbf{x}$ des durch

$$\mathbf{A}_{11}\, \mathbf{x} = \mathbf{a}_1 \tag{6.121}$$

definierten Teilproblems, wobei die zirkulierende Struktur des Gesamtsystems für die Berechnungen genützt werden soll. Im Abschnitt 1.3 wurde der Formelapparat für diesen Bedarf schon erarbeitet. Formel **??**, die durch Rückumwandlung einer ungebundenen zu einer gebundenen Variablen ermittelt wurde, kann direkt angewendet werden. Die Inverse $\mathbf{A}_{11}^{-1}$ errechnet sich durch

$$\mathbf{A}_{11}^{-1} = \mathbf{A}_{11}^{(-1)} - \mathbf{A}_{12}^{(-1)} \left(\mathbf{A}_{22}^{(-1)}\right)^{-1} \left(\mathbf{A}_{12}^{(-1)}\right)^T \tag{6.122}$$

und die Lösung $\mathbf{x}$ des Teilsystems kann mit

$$\mathbf{x} = \xi_1 - \mathbf{A}_{12}^{(-1)} \left(\mathbf{A}_{22}^{(-1)}\right)^{-1} \xi_2 \tag{6.123}$$

ermittelt werden. Da bei der Inversion einer zirkulierenden Matrix die zirkulierende Struktur erhalten bleibt, bildet Teilblock der Inversen $\mathbf{A}_{22}^{(-1)}$ eine Toeplitz-Matrix. Eine rechenökonomische Lösung kann durch Ausnutzung sowohl der zirkulierenden Gesamtstruktur als auch der Toeplitzstruktur erreicht werden. Algorithmus 6.11 veranschaulicht den Rechengang.

Algorithmus 6.11 : Lösung eines Toeplitz-Systems als Teilproblem eines zirkulierenden Systems

Aufgabenstellung:
Gegeben ist ein symmetrisches Toeplitz-System $\mathbf{A}_{11}\mathbf{x} = \mathbf{a}_1$. Gesucht ist die Lösung $\mathbf{x}$ des Systems. Die Lösung erfolgt mit Hilfe des auf eine zirkulierende Matrix erweiteren Schemas mit beliebigem $\mathbf{a}_2$

$$\mathbf{C\Xi} = \mathbf{a} \qquad \mathbf{C} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{bmatrix}, \qquad \mathbf{\Xi} = \begin{bmatrix} \xi_1 \\ \xi_2 \end{bmatrix}, \qquad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}$$

.

Ausgangssituation:
Gegebene Matrizen : $\mathbf{A}_{11}(n_1, n_1)$
Gegebene Vektoren : $\mathbf{a}_1(n_1)$
Gesuchte Vektoren : $\mathbf{x}(n_1)$ $\mathbf{x}_2(n_2)$

Endsituation:
Auf dem Vektor $\mathbf{x}$ ist die Lösung des Gleichungssystems $\mathbf{A}_{11}\mathbf{x} = \mathbf{a}_1$ gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{C}$			$\mathtt{FFT}(Op_1)$	$\mathbf{\Lambda}$	1 $\mathtt{FFT}(n)$	reell
$\mathbf{\Lambda}$	n		Op_1^{-1}/Op_2	$\frac{1}{n}\mathbf{\Lambda}^{-1}$	$2n$	reell
$\frac{1}{n}\mathbf{\Lambda}^{-1}$			$\mathtt{FFT}(Op_1)$	$\mathbf{C}^{-1}$ bzw. $\mathbf{A}_{12}^{(-1)}, \mathbf{A}_{22}^{(-1)}$	1 $\mathtt{FFT}(n)$	reell
$\mathbf{a}$			$\mathtt{FFT}(Op_1)$	$\mathcal{F}\{\mathbf{a}\}$	1 $\mathtt{FFT}(n)$	komplex
$\frac{1}{n}\mathbf{\Lambda}^{-1}$	$\mathcal{F}\{\mathbf{a}\}$		$Op_1{*}Op_2$	$\mathcal{F}^{-1}\{\mathbf{\Xi}\}$	$2n$	komplex
$\mathcal{F}^{-1}\{\mathbf{\Xi}\}$			$\mathtt{FFT}(Op_1)$	ξ_1, ξ_2	1 $\mathtt{FFT}(n)$	reell
$\mathbf{A}_{22}^{(-1)}$	ξ_2		$\mathtt{SOL}(Op_1, Op_2)$	$\mathbf{h}$	$2n_2(n_2+3)$	Toeplitz
ξ_1	$\mathbf{A}_{12}^{(-1)}$	$\mathbf{h}$	$Op_1 - Op_2{*}Op_3$	$\mathbf{x}$	$n_1 n_2$	
Summe der Operatoren					4 $\mathtt{FFT}(n) + 4\,n +$ $n_2(n_1 + 2n_2 + 6)$	
Zwischenergebnisse					$\mathbf{h} = \left(\mathbf{A}_{22}^{(-1)}\right)^{-1} \xi_2$	

Ressourcen:
Anzahl der Operationen : $n_1 n_2 + 2n_2^2 + 4\mathtt{FFT}(n) + \mathcal{O}(n_1, n_2)$
Platzbedarf : $\mathcal{O}(n_1, n_2)$

Unter der Annahme, daß der erste Block dominiert ($n_1 \gg n_2$),stellt der gemischte Anteil $n_1 n_2$ den dominierenden Faktor bei der Anzahl der Operationen dar. Dieser durch die Matrix-Vektor-Multiplikation hervorgerufen Anteil kann umgangen werden, wenn folgender Weg eingeschlagen wird. Ersetzt man in (6.123) den ersten Teile der Lösung durch den Ausdruck in (6.120) so erhält man

$$\mathbf{x} = \mathbf{A}_{11}^{(-1)}\mathbf{a}_1 + \mathbf{A}_{12}^{(-1)}\left(\mathbf{a}_2 - \left(\mathbf{A}_{11}^{(-1)}\right)^{-1}\xi_2\right) \tag{6.124}$$

Wolf-Dieter Schuh
ITGB, Uni Bonn

beziehungsweise in vektorieller Form dargestellt

$$\mathbf{x} = \left[\begin{array}{cc} \mathbf{A}_{11}^{(-1)} & \mathbf{A}_{12}^{(-1)} \end{array} \right] \left[\begin{array}{c} \mathbf{a}_1 \\ \mathbf{a}_2 - \left(\mathbf{A}_{11}^{(-1)} \right)^{-1} \xi_2 \end{array} \right] \tag{6.125}$$

Der erste Vektor, gebildet aus den Blöcken $\mathbf{A}_{11}^{(-1)}$ und $\mathbf{A}_{12}^{(-1)}$, beinhaltet die ersten n_1 Zeilen der Inversen der zirkulierenden Matrix $\mathbf{C}$. Somit weisen diese Zeilen wieder eine zirkulierende Struktur auf. Die Auswertung der Formel (6.125) kann als diskrete Faltung interpretiert werden. Im Gegensatz zu Abschnitt 6.1.3.6 liegt hier nicht eine quadratischen, zirkulierenden Matrix vor, sondern nur die ersten n_1 Zeilen. Während die ersten n_1 Elemente des Ergebnissvektors von diesem Faktum unbeeinflußt bleiben, existieren die weiteren n_2 Elemente nicht oder sind bedeutungslos. Der Berechnungsablauf wird ganz auf die zirkulierenden Strukturen abgestimmt. Nach der spektralen Lösung und Inversion des zirkulierenden Systems

$$\mathbf{C}\Xi = \mathbf{a} \tag{6.126}$$

wird der Konstantenvektor $\mathbf{a}$ verändert. Der modifizierte Vektor $\hat{\mathbf{a}}$ errechnet sich durch die Subtraktion der Lösung des Toeplitz-Systems vom zweiten Anteil $\mathbf{a}_2$,

$$\hat{\mathbf{a}} = \left[\begin{array}{c} \mathbf{a}_1 \\ \mathbf{a}_2 - \left(\mathbf{A}_{11}^{(-1)} \right)^{-1} \xi_2 \end{array} \right] \tag{6.127}$$

Die ersten n_1 Elemente der Lösung η des zirkulierenden Systems

$$\mathbf{C}\eta = \hat{\mathbf{a}} \tag{6.128}$$

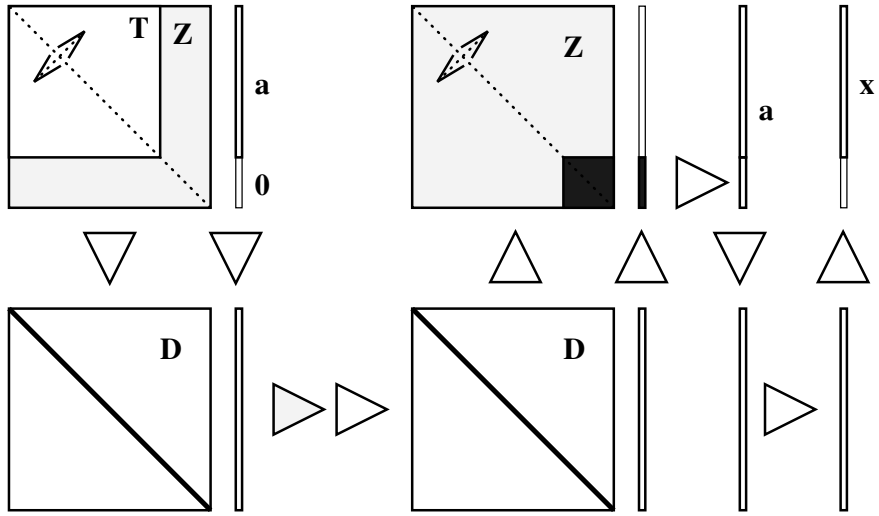
bilden somit die gesuchte Lösung $\mathbf{x}$ des Teilproblems. Algorithmus 6.12 und Abb. **??** veranschaulichen den Ablauf.

Algorithmus 6.12 : Lösung eines Toeplitz-Systems als Teilproblem eines zirkulierenden Systems unter vollständiger Nutzung der regelmäßigen Struktur.

Aufgabenstellung:
Gegeben ist ein symmetrisches Toeplitz-System $\mathbf{A}_{11}\mathbf{x} = \mathbf{a}_1$. Gesucht ist die Lösung $\mathbf{x}$ des Systems. Die Lösung erfolgt mit Hilfe des auf eine zirkulierende Matrix erweiteren Schemas mit beliebigem $\mathbf{a}_2$

$$\mathbf{C}\Xi = \mathbf{a} \qquad \mathbf{C} = \left[\begin{array}{cc} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^T & \mathbf{A}_{22} \end{array} \right] , \qquad \Xi = \left[\begin{array}{c} \xi_1 \\ \xi_2 \end{array} \right] , \qquad \mathbf{a} = \left[\begin{array}{c} a_1 \\ a_2 \end{array} \right]$$

.



Z	zirkulierende Matrix	$\triangledown\,\triangle$	Fourier-Transfomationen
T	Toeplitz-Matrix	$\triangleright$	Inversion
D	Diagonalmatrix	$\triangleright$	Lösung

Abbildung 6.13: Ablaufdiagramm zur Berechnung der Lösung eines Teilsystems eines zirkulierenden Systems unter vollkommener Ausnutzung der regelmäßigen Struktur.

Ausgangssituation:
Gegebene Matrizen : $\mathbf{A}_{11}(n_1, n_1)$
Gegebene Vektoren : $\mathbf{a}_1(n_1)$
Gesuchte Vektoren : $\mathbf{x}(n_1)$ $\mathbf{x}_2(n_2)$

Endsituation:
Auf dem Vektor $\mathbf{x}$ ist die Lösung des Gleichungssystems $\mathbf{A}_{11}\mathbf{x} = \mathbf{a}_1$ gespeichert.

Lösungsweg:

Operatoren			Berechnung	Ergebnis	Anzahl der Operationen	Anmerkung
Op_1	Op_2	Op_3				
$\mathbf{C}$			$\mathtt{FFT}(Op_1)$	$\mathbf{\Lambda}$	1 $\mathtt{FFT}(n)$	reell
$\mathbf{\Lambda}$	n		Op_1^{-1}/Op_2	$\frac{1}{n}\mathbf{\Lambda}^{-1}$	$2n$	reell
$\frac{1}{n}\mathbf{\Lambda}^{-1}$			$\mathtt{FFT}(Op_1)$	$\mathbf{C}^{-1}$ bzw. $\mathbf{A}_{12}^{(-1)}, \mathbf{A}_{22}^{(-1)}$	1 $\mathtt{FFT}(n)$	reell
$\mathbf{a}$			$\mathtt{FFT}(Op_1)$	$\mathcal{F}\{\mathbf{a}\}$	1 $\mathtt{FFT}(n)$	komplex
$\frac{1}{n}\mathbf{\Lambda}^{-1}$	$\mathcal{F}\{\mathbf{a}\}$		$Op_1{*}Op_2$	$\mathcal{F}^{-1}\{\mathbf{\Xi}\}$	$2n$	komplex
$\mathcal{F}^{-1}\{\mathbf{\Xi}\}$			$\mathtt{FFT}(Op_1)$	ξ_1, ξ_2	1 $\mathtt{FFT}(n)$	reell
$\mathbf{A}_{22}^{(-1)}$	ξ_2		$\mathtt{SOL}(Op_1, Op_2)$	$\mathbf{h}$	$2n_2(n_2+3)$	Toeplitz
$\mathbf{a}_2$	$\mathbf{h}$		$Op_1 - Op_2$	$\hat{\mathbf{a}}_2$		reell
$\hat{\mathbf{a}}$			$\mathtt{FFT}(Op_1)$	$\mathcal{F}\{\hat{\mathbf{a}}\}$	1 $\mathtt{FFT}(n)$	komplex
$\frac{1}{n}\mathbf{\Lambda}^{-1}$	$\mathcal{F}\{\hat{\mathbf{a}}\}$		$Op_1{*}Op_2$	$\mathcal{F}^{-1}\{\mathbf{x}\}$	$2n$	komplex
$\mathcal{F}^{-1}\{\mathbf{x}\}$			$\mathtt{FFT}(Op_1)$	$\mathbf{x}$	1 $\mathtt{FFT}(n)$	reell
Summe der Operatoren					6 $\mathtt{FFT}(n) + 6\ n +$ $2n_2(n_2+3)$	
Zwischenergebnisse					$\mathbf{h} = \left(\mathbf{A}_{22}^{(-1)}\right)^{-1} \xi_2$ $\hat{\mathbf{a}} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 - \mathbf{h} \end{bmatrix}$	

Ressourcen:

Anzahl der Operationen : $2n_2^2 + 6\mathtt{FFT}(n) + \mathcal{O}(n_1, n_2)$

Platzbedarf : $\mathcal{O}(n_1, n_2)$

Für das Verschwinden des dominierenden Faktors $n_1 n_2$ der ersten Berechnungsmethode müssen zwei weitere Fourier-Transformationen in Kauf genommen werden. Abhängig von der Teilbarkeit der Größe n und der größten enthaltenen Primzahl kann entweder die erste oder die zweite Methode den günstigeren Weg darstellen. Interessant ist die vollständige Ausnutzung der zirkulierenden Struktur vor allem dann, wenn die Dimension n des zirkuliernden Systems selbst gewählt werden kann. Im nachfolgenden Abschnitt wird eine Anwendung aufgezeigt, wo dieser Vorteil der freien Wahl genutzt werden kann.

Wolf-Dieter Schuh
ITGB, Uni Bonn

6.1.5 Lösung und Inversion von Toeplitz-Systemen mit Bandstruktur

Wolf-Dieter Schuh
ITGB, Uni Bonn

6.2 Regelmäßige zwei- und mehrdimensionale Strukturen

6.2.1 Grundzüge der Array Algebra

Wolf-Dieter Schuh
ITGB, Uni Bonn

Der gesuchte Lösungsvektor $\mathbf{x}$ ist durch das lineare Gleichungssystem

$$\mathbf{T}\mathbf{x} = \mathbf{b} \tag{6.129}$$

6.2.2 Regelmäßige zweidimensionale Strukturen

Literaturverzeichnis

[1] ASHKENAZI V., R.C. WOOD (1973): Merits of the Conjugate Gradients Method for Solving Geodetic Normal Equations. I.A.G. Symposium on "Computational Methods in Geometrical Geodesy", Oxford.

[2] AKAIKE Hirotugu (1973): Block Toeplitz Matrix Inversion. SIAM J. Appl. Math., Vol. 24, N0.2, pp. 234-241.

[3] ANDERSON E., Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. McKENNEY, D. SORENSEN (1990): LAPACK: A Portable Linear Algebra Library for High-Performance Computers. In "Proceedings of Supercomputing '90", IEEE Press, pp. 1-10.

[4] ANDERSON E., J. DONGARRA (1992): Performance of LAPACK: A Portable Library of Numerical Linear Algebra Routines. Technical Report CS-92-156, Department of Computer Science, University of Tennessee, Knoxville (LAPACK Working Note 44).

[5] BARTELME Norbert, Peter MEISSL (1979): Application of Toeplitz Forms to the Mathematical Analysis of Geodetic Networks. Bollettino di Geodesia e Scienze Affini - No. 4, pp. 577-587.

[6] BATH Markus (1974): Spectral Analysis in Geophysics. Developments in Solid Earth Geophysics 7, Elsevier Scientific Publishing Company, Amsterdam-Oxford-New York.

[7] BENOIT (1924):Note sur une methode de resolution des equations normales etc. (procede du commandant Cholesky). Bulletin Geodesique, Vol. 3, pp. 67-77.

[8] BLAHA George (1977): A Few Basic Principles and Techniques of Array Algebra. Bulletin Géodésique, Vol. 51, No. 3, pp. 177-202.

[9] BLAHA George (1977): Least Squares Prediction and Filtering in Any Dimension Using the Principles of Array Algebra. Bulletin Géodésique, Vol. 51, No. 4, pp. 265-286.

[10] BORRE Kai (1979): Covariance Matrices / Functions for 1-D Levelling Problems. Unpublished manuscript.

[11] BORRE Kai (1981): A Priori Estimation of the Strength of Control Networks. Invited Paper, 'FIG XVI. Intern. Kongress', Montreux.

[12] BOTTONI Gian Paolo, Riccardo BARZAGHI (1993): Fast Collocation. Bulletin Géodésique, Vol. 67, No. 2, pp. 119-126.

[13] BRIGHAM E.Oran (1974): The Fast Fourier Transform. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[14] BUNSE Wolfgang, Angelika BUNSE-GERTNER(1985): Numerische lineare Algebra. Teubner Studienbücher, Stuttgart.

[15] BURKARD Rainer E. (1972): Methoden der Ganzzahligen Optimierung. Springer Verlag, Wien-New York.

[16] COLOMBO Oscar L. (1979): Optimal Estimation from Data Regularly Sampled on a Sphere with Applications in Geodesy. Department of Geodetic Science, Report No. 291, Ohio State University, Columbus, Ohio.

[17] CRESPI Mattia, Gianfranco FORLANI, Luigi MUSSIO (1989): Optimization of the Reordering Algorithm for Least Squares Problems. Tutorial on "Mathematical Aspects of Data Analysis", ISPRS, Intercommission Working Group III/VI, Pisa, pp. 185-202.

[18] CUTHILL E. and J. McKEE (1969): Reducing the Bandwidth of Sparse Symmetric Matrices. Proc. ACM National Conference pp. 157-172. Association for Computing Machinery. New York.

[19] CUTHILL E. (1972): Several Strategies for Reducing the Bandwidth of Matrices. ROSE D.J. and R.A. WILLOUGHBY (1972): (Editors) 'Sparse Matrices and their Applications', Academic Press, New York-London.

[20] DANKERT Jürgen (1977): Numerische Methoden der Mechanik. Springer-Verlag, Wien-New York

[21] DANTZIG G. B. (1966): Lineare Programmierung und Erweiterungen. Springer Verlag, Berlin-Heidelberg-New York.

[22] D'AZEVEDO E.F., V.L. EIJKHOUT and C.H. ROMINE (1993): Reducing Communication Costs in the Conjugate Gradient Algorithm on Distributed Memory Multiprocessors. Technical Report CS-93-185, Department of Computer Science, University of Tennessee, Knoxville (LAPACK Working Note 56).

[23] DEMMEL J., J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, D. SORENSEN (1987): Prospectus for the Development of a Linear Algebra Library for High-Performance Computers. Technical Memorandum No. 97, Mathematics and Computer Science Division, Argonne National Laboratory, Illinois (LAPACK Working Note 1).

[24] DONGARRA J., G. DU CROZ, S. HAMMARLING, R. HANSON (1988): An Extended Set of FORTRAN Basic Linear Algebra Subprograms. ACM Trans. Math. Software 14, pp. 1-17.

[25] DONGARRA J., G. DU CROZ, I. DUFF, S. HAMMARLING (1990): A Set of Level 3 Basic Linear Algebra Subprograms. ACM Trans. Math. Software 16, pp. 1-17.

[26] DONGARRA J., R. van DE GEIJN (1991): Two Dimensional Basic Linear Algebra Communication Subprograms. Technical Report CS-91-138, Department of Computer Science, University of Tennessee, Knoxville (LAPACK Working Note 37).

[27] EBNER Heinrich (1970): Die theoretische Lagegenauigkeit ausgeglichener Blöcke mit bis zur 10000 unabhängigen Modellen. Bildmessung und Luftbildwesen, Vol. 38, S. 225-232.

[28] EREN Kamil (1980): Spectral Analysis of GEOS-3 Altimeter Data and Frequency Domain Collocation. Department of Geodetic Science, Report No. 297, Ohio State University, Columbus, Ohio.

[29] FRITSCH Dieter (1989): Multivariante Data Analysis. Proceedings of the Tutorial on 'Mathematical Aspects of Data Analysis', ISPRS Intercommission Working Group III/VI, pp. 169-183, Pisa, June 1-2, 1989.

[30] GALLIVAN K.A., R.J. PLEMMONS, A.H. SAMEH (1990): Parallel Algorithms for Dense Linear Algebra Computations. In GALLIVAN K.A., Michael T. HEATH, Esmond NG, James M. ORTEGA, Barry W. PEYTON, R.J. PLEMMONS, Charles H. ROMINE, A.H. SAMEH, Robert G. VOIGT: "Parallel Algorithms for Matrix Computations", Society for Industrial and Applied Mathematics (SIAM), Philadelphia, pp. 1-82.

[31] GEORGE Alan (1973): Nested Dissection of a Regular Finite Element Mesh. SIAM J. Numer. Anal., Vol. 10, No. 2, pp. 345-363.

[32] GEORGE Alan, Joseph W-H. LIU (1979): An Implementation of a Pseudoperipheral Node Finder. ACM Transactions on Mathematical Software. Vol. 5, No. 3., pp. 284-295.

[33] GEORGE A. (1987): Solution of Sparse Systems of Equations on Multiprocessor Architectures. In TURNER P.R.(ed): 'Numerical Analysis and Parallel Processing', Proceedings, Lancaster 1987. Lecture Notes in Mathematics 1397, p. 32-94, Springer.

[34] GEORGE Alan and Joseph W-H. LIU (1981): Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

[35] GIBBS N.E. (1976): Algorithm 509: A Hybrid Profile Reduction Algorithm. ACM Transactions on Mathematical Software Vol. 2, No. 4, pp. 378-387.

[36] GIBBS N.E., W.G. POOLE, P.K. STOCKMEYER (1976): A Comparison of Several Bandwidth and Profile Reduction Algorithms. ACM Transactions on Mathematical Software Vol. 2, No. 4, pp. 322-330.

[37] GIBBS N.E., W.G. POOLE and P.K. STOCKMEYER (1976): An Algorithm for Reducing the Bandwidth and Profile of a Sparse Matrix. SIAM Numerical Analysis Vol. 13, No. 2, pp. 236-250.

[38] GOLUB Gene H., Charles F. van LOAN (1983): Matrix Computations. North Oxford Academic, Oxford.

[39] GRADSHTEYN I.S., I.M. RYZHIK (1980): Table of Integrals, Series, and Products. Academic Press, 4^th Edition, New York-London-Toronto-Sydney-San Francisco.

[40] GRAY Robert M. (1977): Toeplitz and Circulant Matrices: II Information Systems Laboratory. Technical Report No. 6504-1, Stanford Electronics Laboratories, Stanford University, Stanford, California.

[41] GRÜNDIG L. (1980): Feasibility Study of the Conjugate Gradient Method for Solving Large Sparse Equation Sets. NOAA Technikal Report. NOS 82 NGS 13.

[42] HESTENES M., E. STIEFEL (1952): Methods of Conjugate Gradients for Solving Linear Systems. Jounal of Research of the National Bureau of Standards, Vol. 49, No. 6, Research Paper 2379.

[43] HESTENES M. (1980): Conjugate Direction Methods in Optimization. Applications of Mathematics 12, Springer, Berlin-Heidelberg-New York.

[44] HANSON R.H. (1978): A Posteriori Error Propagation. Second International Symposium on Problems Related to the Redefinition of North American Geodetic Networks. Proceedings, pp. 427-445.

[45] KERSHAW D. (1978): The Incomplete Cholesky-Conjugate Gradient Method for the Iterativ Solution of Systems of Linear Equations. Journal of Computaional Physiks 26, pp. 43-65.

[46] KING I.P. (1970): An Automatic Reordering Scheme for Simultaneous Equations Derived from Network Problems. International Journal for Numerical Methods in Engineering, Vol. 2, pp. 523-533.

[47] LAWSON C.L., R.J. HANSON, D.R. KINCAID, F.T. KROGH (1979): Basic Linear Algebra Subprograms for Fortran Usage. ACM Trans. Math. Software 5, pp. 308-323.

[48] LEVINSON Norman (1947): The Wiener RMS (Rott Mean Square) Error Criterion in Filter Design and Prediction. Journal of Mathematics & Physics, No. 25, pp. 261-278.

[49] MAREL Hans van der (1989): Algorithmic Aspects of Solving Large Systems of Equations. Proceedings of the Tutorial on „Mathematical Aspects of Data Analysis“, ISPRS Intercommission Working Group III/VI, pp. 49-103, 1-2 June 1990, Pisa.

[50] MEISSL Peter (1967): Die verallgemeinerte Inverse einer modifizierten Matrix. Zeitschrift für angewandte Mathematik und Mechanik (ZAMM), 47, S. T66-T67.

Wolf-Dieter Schuh
ITGB, Uni Bonn

[51] MEISSL Peter (1969): Über zufällige Fehler in regelmäßigen gestreckten Ketten. Zeitschrift für Vermessungswesen (ZfV), Konrad Wittwer, Stuttgart, Vol. 94, Heft 1, S. 14-26.

[52] MEISSL Peter (1970): Über der Fehlerfortpflanzung in gewissen regelmäßigen flächig ausgebreiteten Nivellementnetzen. Zeitschrift für Vermessungswesen (ZfV), Konrad Wittwer, Stuttgart, Vol. 95, Heft 3, S. 103-109.

[53] MEISSL Peter (1976): Strength Analysis of Two-Dimensional Angular Anblock Networks. Manuscripta Geodaetica, Vol. 1, 293-333.

[54] MEISSL Peter (1980): Die Eigenwertmethode am Beispiel des endlichen eingehängten Nivellementzuges. Interner Bericht, Mathematische Geodäsie, TU Graz.

[55] MEISSL Peter (1982a): Least Squares Adjustment: A Modern Approach. Mitteilungen der Geodätischen Institute der TU Graz, Folge 43.

[56] MEISSL Peter (1982b): Fourier Analysis of Geodetic Networks. Unpublished manuscript.

[57] NASH J. C. (1979): Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation. Adam Hilger Ltd, Bristol.

[58] OPPENHEIM A., R. SCHAFER (1975): Digital Signal Processing. Prentice-Hall Inc., Englewood Cliffs, New Jersey.

[59] OPPENHEIM A., R. SCHAFER (1992): Zeitdiskrete Signalverarbeitung. Oldenbourg Verlag, München Wien.

[60] ORESCHNIK Kurt (1985): Analyse von zweidimensionalen Netzen mit Hilfe von Eigenwerten und Eigenvektoren. Diplomarbeit, TU Graz.

[61] PABSTMANN Klaus D. (1983): Ein Algorithmus zur Inversion von Block-Toeplitz-Matrizen und dessen Anwendung auf statistische Schätzverfahren. Diplomarbeit TU Graz.

[62] PELZER H. (1980): Geodätische Netze in Landes- und Ingenieurvermessung. Vermessungswesen Band 5, Konrad Wittwer.

[63] PRESS William H., Brian P. FLANNERY, Saul A. TEUKOLSKY, William T. VETTERLING (1989): Numerical Recipes. The Art of Scientific Computing (FORTRAN Version). Cambridge University Press, Cambridge-London-New York-New Rochelle-Melbourne-Sydney.

[64] PRESS W., S. TEUKOLSKY, W. VETTERLING, B. FLANNERY, (1992): Numerical Recipes. The Art of Scientific Computing (FORTRAN Version, 2nd ed.). Cambridge University Press, Cambridge-London-New York-New Rochelle-Melbourne-Sydney.

[65] RAUHALA Urho A. (1978): Array Algebra DTM. Proceedings of Digital Terrain Models (DTM) Symposium, May, 9-11, St. Louis, American Society of Photogrammetry, Falls Church, Virginia.

[66] RAUHALA Urho A. (1979): Development of Experimental Array Algebra Algorithms for Filtering and Compaction of AS-11B-X and Seasat Altimetry Data. Report of Geodetic Services, Indialantic, Florida.

[67] RAUHALA Urho A. (1980): Introduction to Array Algebra. Photogrammetric Engineering and Remote Sensing, Vol. 46, No. 2, pp. 177-192.

[68] ROSEN Richard (1968): Matrix Bandwidth Minimization. Proc. ACM National Conference, pp. 585-595. Brandon System Press, Princeton, New York.

[69] RUTHERFORD D. E. (1947): Some Continuant Determinants Arising in Physics and Chemistry. Proc. Roy. Soc. Edin., A, LXII, pp. 229-236.

[70] RUTHERFORD D. E. (1952): Some Continuant Determinants Arising in Physics and Chemistry - II. Proc. Roy. Soc. Edin., A, LXIII, pp. 232-241.

[71] SCHEK Hans-Jörg, Franz STEIDLER und Ulrich SCHAUER (1977): Ausgleichung grosser geodätischer Netze mit Verfahren für schwach besetzte Matrizen. Deutsche Geodätische Kommission (DGK), Reihe A, Heft 87.

[72] SCHUH Wolf-Dieter (1981): Programmierung rationeller Algorithmen zur Umordnung, Auflösung und Inversion der Normalgleichungen geodätischer Netze. Diplomarbeit, TU Graz.

[73] SCHUH Wolf-Dieter (1984): Analyse und Konvergenzbeschleunigung der Methode der konjugierten Gradienten bei geodätischen Netzen. Mitteilungen der geodätischen Institute der TU Graz, Folge 49.

[74] SCHWARZ Hans Rudolf (1968): Numerik symmetrischer Matrizen. Teubner, Stuttgart.

[75] SCHWARZ Hans Rudolf (1970): Die Methode der konjugierten Gradienten in der Ausgleichsrechnung. Zeitschrift für Vermessungswesen (ZfV), Nr. 4, S. 130-140.

[76] SCHWARZ Hans Rudolf (1980): Methode der finiten Elemente. Teubner Studienbücher für Mathematik, Band 47.

[77] SCHWARZ Hans Rudolf (1981): FORTRAN - Programme zur Methode der finiten Elemente. Teubner Studienbücher für Mathematik.

[78] SCHWARZ Hans Rudolf (1988): Numerische Mathematik. Teubner, Stuttgart, 2.Auflage.

[79] SCHWARZ K. P., M. G. SIDERIS, R. FORSBERG (1990): The Use of FFT Techniques in Physical Geodesy. Geophys. J. Int. 100, pp. 485-514.

[80] SNAY Richard A. (1976): Reducing the Profile of Sparse Symmetric Matrices. NOAA Technical Memorandum NOS NGS-4.

[81] SNAY Richard A. (1978): Applicability of Array Algebra. NOAA Technical Memorandum NOS NGS-11.

[82] STEWART G. W. (1973): Introduction to Matrix Computations. Academic Press, New York-San Fancisco-London.

[83] STIEFEL Eduard (1952): Über einige Methoden der Relaxationsrechnung. Zeitschrift für angewandte Mathematik und Physik (ZAMP), 3, S. 1-33.

[84] STIEFEL Eduard (1976): Einführung in die numerische Mathematik. Teubner Studienbücher Mathematik, 5. Auflage, Stuttgart.

[85] STRANG Gilbert (1986): Introduction to Applied Mathematics. Wellesley-Cambridge-Press, Massachusetts.

[86] STRANG Gilbert (1988): Linear Algebra and its Applications. Harcourt Brace Jovanovich, 3^{rd} Edition, San Diego.

[87] SÜNKEL Hans (1984): SPLINES: Their Equivalence to Collocation. Ohio State University (OSU), Reports of the Department of Geodetic Science, No. 357.

[88] SÜNKEL Hans (1985): Fourier Analysis of Geodetic Networks. Eigenvalues. GRAFAREND E.W., F. SANSO (Editors): 'Optimization and Design of Geodetic Networks', Springer, Heidelberg, pp. 257-300.

[89] SÜNKEL Hans (1993): Skriptum aus Ausgleichsrechnung I. Mathematische Geodäsie und Geoinformatik, TU Graz.

[90] TRENCH William F. (1964): An Algorithm for the Inversion of Finite Toeplitz Matrices. J. Soc. Indust. Appl. Math., Vol. 12, No. 3, pp. 515-522, September 1964.

[91] TRENCH William F. (1974): Inversion of Toeplitz Band Matrices. Mathematics of Computation, Vol. 28, No. 128, pp. 1089-1095.

[92] TÖRNIG Willi, Peter SPELLUCCI (1988): Numerische Mathematik für Ingenieure und Physiker. Band 1: Numersiche Methoden der Algebra (2.Auflage). Springer-Verlag New York-Berlin-Heidelberg.

16. Oktober 2003

Wolf-Dieter Schuh
ITGB, Uni Bonn

[93] TÖRNIG Willi, Peter SPELLUCCI (1990): Numerische Mathematik für Ingenieure und Physiker. Band 2: Numersiche Methoden der Analysis (2.Auflage). Springer-Verlag New York-Berlin-Heidelberg.

[94] VARGA Richard S. (1962): Matrix Iterativ Analysis. Prentice-Hall, Series in Automatic Computation, Englewood Cliffs-New Jersey.

[95] VERMEER Martin (1992): Terrain Reduction and Gridding Techniques for Geoid Determination. Lecture Notes for NKG-Autumn School for 'Geodesy and Geophysics', Publications of the Finnish Geodetic Institute 115, Helsinki. pp. 173-181.

[96] WOTRUBA Markus F. (1982): Kollokation für strukturierte Datensätze. Diplomarbeit TU Graz.

[97] WIESER Manfred (1988): Theoretische Untersuchungen spektraler Methoden zur Analyse regelmässiger Strukturen am Beispiel der Fouriertransformation geodätischer Netze. Mitteilungen der geodätischen Institute der TU Graz, Folge 60.

[98] YOUNG David M. (1971): Iterative Solution of Large Linear Systems. Computer Science and Applied Mathematics, Academic Press, New York-San Franzisco-London.

[99] ZURMÜHL Rudolf (1964): Matrizen und ihre technische Anwendungen (4. Auflage). Springer-Verlag, Berlin-Göttingen-Heidelberg.

[100] ZURMÜHL Rudolf, Sigurd FALK (1984): Matrizen und ihre technische Anwendungen (5. Auflage); Teil 1: Grundlagen. Springer-Verlag, Berlin-Göttingen-Heidelberg.

[101] ZURMÜHL Rudolf, Sigurd FALK (1986): Matrizen und ihre technische Anwendungen (5. Auflage); Teil 2: Numerische Methoden. Springer-Verlag, Berlin-Göttingen-Heidelberg.

Häufig verwendete Literatur

[34] GEORGE Alan and Joseph W-H. LIU (1981): Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey.

[38] GOLUB Gene H., Charles F. van LOAN (1983): Matrix Computations. North Oxford Academic, Oxford.

[55] MEISSL Peter (1982): Least Squares Adjustment: A Modern Approach. Mitteilungen der Geodätischen Institute der TU Graz, Folge 43.

[63] PRESS William H., Brian P. FLANNERY, Saul A. TEUKOLSKY, William T. VETTERLING (1989): Numerical Recipes. The Art of Scientific Computing (FORTRAN Version). Cambridge University Press, Cambridge-London-New York-New Rochelle-Melbourne-Sydney.

[74] SCHWARZ Hans Rudolf (1968): Numerik symmetrischer Matrizen. Teubner, Stuttgart.

[78] SCHWARZ Hans Rudolf (1988): Numerische Mathematik. Teubner, Stuttgart, 2.Auflage.

[86] STRANG Gilbert (1988): Linear Algebra and its Applications. Harcourt Brace Jovanovich, 3^{rd} Edition, San Diego.

[89] SÜNKEL Hans (1993): Skriptum aus Ausgleichsrechnung I. Mathematische Geodäsie und Geoinformatik, TU Graz.

[100] ZURMÜHL Rudolf, Sigurd FALK (1984): Matrizen und ihre technische Anwendungen (5. Auflage); Teil 1: Grundlagen. Springer-Verlag, Berlin-Göttingen-Heidelberg.

[101] ZURMÜHL Rudolf, Sigurd FALK (1986): Matrizen und ihre technische Anwendungen (5. Auflage); Teil 2: Numerische Methoden. Springer-Verlag, Berlin-Göttingen-Heidelberg.

Spezielle Literatur (regelmäßige Strukturen)

[2] AKAIKE Hirotugu (1973): Block Toeplitz Matrix Inversion. SIAM J. Appl. Math., Vol. 24, N0.2, pp. 234-241.

[8] BLAHA George (1977): A Few Basic Principles and Techniques of Array Algebra. Bulletin Géodésique, Vol. 51, No. 3, pp. 177-202.

[40] GRAY Robert M. (1977): Toeplitz and Circulant Matrices: II Information Systems Laboratory. Technical Report No. 6504-1, Stanford Electronics Laboratories, Stanford University, Stanford, California.

[67] RAUHALA Urho A. (1980): Introduction to Array Algebra. Photogrammetric Engineering and Remote Sensing, Vol. 46, No. 2, pp. 177-192.

[81] SNAY Richard A. (1978): Applicability of Array Algebra. NOAA Technical Memorandum NOS NGS-11.

[95] VERMEER Martin (1992): Geoid Determination using Frequency Domain Techniques. Lecture Notes for NKG-Autumn School for 'Geodesy and Geophysics', Publications of the Finnish Geodetic Institute 115, Helsinki. pp. 183-200.

Spezielle Literatur (Verallgemeinerte Inverse)

[66] RAUHALA Urho A. (1979): Intuitive Derivation of Loop Inverses and Array Algebra. Bulletin Géodésique, Vol. 53, pp. 315-342.

Index

Index