Paper-ID: VGI_198912

Eine externe Lösung der Normalgleichungen inklusive Genauigkeits- und Zulässigkeitsberechnung

M. Gsandtner [1], Helmut J. Kager [2]

[1] *Technische Universität Wien Institut für Photogrammetrie und Fernerkundung, Gußhausstraße 27-29/122 A-1040 Wien*
[2] *Technische Universität Wien Institut für Photogrammetrie und Fernerkundung, Gußhausstraße 27-29/122 A-1040 Wien*

Österreichische Zeitschrift für Vermessungswesen und Photogrammetrie **77** (4), S. 180–189

1989

BibTEX:

```
@ARTICLE{Gsandtner_VGI_198912,
Title =  {Eine externe L{\"o}sung der Normalgleichungen inklusive Genauigkeits-
    und Zul{\"a}ssigkeitsberechnung},
Author =  {Gsandtner, M. and Kager, Helmut J.},
Journal =  {{\"O}sterreichische Zeitschrift f{\"u}r Vermessungswesen und
   Photogrammetrie},
Pages =  {180--189},
Number =  {4},
Year =  {1989},
Volume =  {77}
}
```

Eine externe Lösung der Normalgleichungen inklusive Genauigkeits- und Zuverlässigkeitsberechnung

von M. Gsandtner und H. Kager, Wien

Abstract

ORIENT is a general bundle adjustment program. The sparse normal equation solution of ORIENT is presented. The Cholesky method is used. Singularities are detected and removed during solution. The normal equations as well as the observation equations are organized by grid-shaped partitions which contain merely non-zero submatrices. Therefore, paging can be done especially suited for the equation solution algorithm, and this is more efficient than paging by a virtual operating system. Moreover, equations of any number of unknowns may be solved, even on non virtual operating systems, an important fact especially for PC's. Additionally, algorithms are presented for the computation of the accuracy of unknowns and for inner and outer reliability information.

Zusammenfassung

ORIENT ist ein allgemeines Ausgleichsprogramm insbesondere für photogrammetrische Bündelblöcke. Die Methode mit der die Normalgleichungen in ORIENT aufgelöst werden, wird vorgestellt. Sie berücksichtigt, daß die Normalgleichungsmatrix nur schwach besetzt ist, d.h. daß sie viele Nullen enthält. Es wird die Methode von Cholesky angewandt. Singularitäten werden während der Gleichungslösung erkannt und beseitigt. Die Normalgleichung, genauso wie die Fehlergleichungen, sind rasterförmig in sogenannten Partitionen organisiert, die nur die von Null verschiedenen Submatrizen enthalten. Der Algorithmus für das „Hin- und Herblättern" (paging) zwischen Kernspeicher und Hintergrundspeicher ist speziell auf das Problem der Gleichungslösung abgestimmt. Es können nun, auch unter nicht virtuellen Betriebssystemen, Normalgleichungen mit praktisch beliebig vielen Unbekannten gelöst werden. Diese Tatsache ist besonders für PC's von Bedeutung. Außerdem werden die Algorithmen vorgestellt, mit denen die mittleren Fehler der Unbekannten, sowie Werte für die innere und äußere Zuverlässigkeit berechnet werden.

1. Einleitung

Das allgemeine Ausgleichungsprogramm ORIENT ist insbesondere auf photogrammetrische Bündelblöcke, d.h. auf die Ausgleichung von PHOTO-Punkten, ausgelegt. Es erlaubt aber auch die Einbeziehung von Punkten in einem lokalen dreidimensionalen kartesischen Koordinatensystem (MODEL-Punkte) und von Punkten entlang von Kurven und auf Flächen (GESTALT-Punkte)[3]. Die Fehlergleichungen (FGL) und die Normalgleichungen (NGL) einer solchen Ausgleichung werden bei praktischen Aufgabenstellungen sehr groß; sie enthalten aber viele von Null verschiedene Koeffizienten, d.h. sie sind dünn (sparse) besetzt. Der Rechenaufwand für eine photogrammetrische Ausgleichung hängt entscheidend davon ab, wie effizient man diese dünne Besetzung ausnützt. Wenn man große Ausgleichungen durchführt, ist für die Speicherung der FGL und NGL, die selbstverständlich in gepackter Form durchgeführt wird, der Kernspeicher zu klein; es muß der Hintergrundspeicher (Disk) einbezogen werden. Ein solches „Hin- und Herblättern" (pa-

ging) zwischen Kernspeicher und Hintergrundspeicher übernehmen virtuelle Betriebssysteme. Für manchen Rechner, insbesondere für personal computer (PC's) gibt es keine virtuellen Betriebssysteme. Für PS's muß deshalb ein „paging" selbst programmiert werden. Im folgenden wird die in ORIENT verwirklichte Methode erläutert.

2. Die hierarchische Struktur der Gleichungen

Aufstellung und Reduktion der NGL erfolgt überlappend. Die Struktur der NGL und FGL ist sehr ähnlich. Wir beginnen damit, die Struktur der FGL zu beschreiben.

Jeder beobachtete Punkt liefert eine, zwei oder drei FGL. Ein PHOTO-Punkt z.B. liefert zwei FGL, eine für die x-Koordinate und eine für die y-Koordinate jedes Bildpunktes; ein MODEL-Punkt liefert drei FGL, ein GESTALT-Punkt (Fläche bzw. Kurve) liefert eine bzw. zwei FGL. Jede Unbekannte, die in einer FGL vorkommt, erzeugt - im allgemeinen - einen von Null verschiedenen Skalar in der FGL-Matrix. Wenn z.B. das Projektionszentrum eines PHOTOS unbekannt ist, liefert jeder Bildpunkt in jeder der beiden Gleichungen drei von Null verschiedene Skalare, da das Projektionszentrum im Objektkoordinatensystem drei unbekannte Koordinaten hat. Der beobachtete Bildpunkt liefert zusammen mit dem unbekannten Projektionszentrum eine Sub-Matrix in der FGL, die von Null verschiedene Skalare enthält (in obigem Beispiel ist dies eine Matrix mit zwei Zeilen und drei Spalten). Die Unbekannten können zu Punkten zusammengefaßt werden, z.B. zu Objektpunkten oder zu einem „Drehungspunkt", der die drei Drehwinkel enthält, die für eine räumliche Drehmatrix nötig sind (z.B. die Drehwinkel eines PHOTOS als ein Teil der äußeren Orientierung). Die Unbekannten innerhalb eines solchen „Unbekanntenpunktes" stehen hintereinander im Vektor der Unbekannten. So liefert jeder beobachtete Punkt zusammen mit einem Unbekanntenpunkt, der in dieser FGL vorkommt, eine Matrix in der FGL, die von Null verschiedene Skalare enthält. Wir nennen solche Matrizen „Submatrizen" (SM).

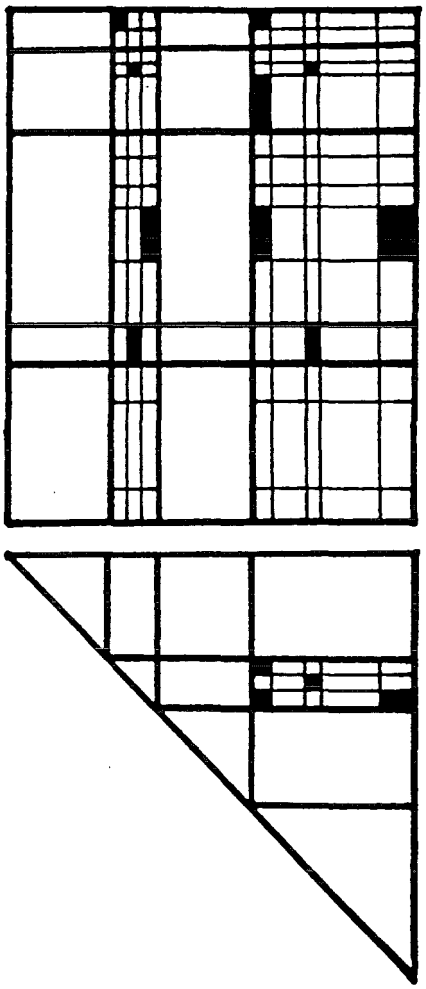
Zu einer SM gehören auch noch einige Verwaltungsinformationen, z.B. zu welcher FGL diese SM gehört. Nur solche SM, die von Null verschiedene Skalare enthalten, werden gespeichert. Alle anderen Skalare der FGL und auch der NGL sind Null und werden nicht gespeichert.

Für jeden beobachteten Punkt werden die Submatrizen für die FGL berechnet. Die Submatrizen werden zeilenweise geliefert und auch zeilenweise gespeichert. Die Submatrizen werden sequentiell aneinander gereiht im Kernspeicher gesammelt. Sobald diese einen bestimmten Pufferbereich überschreiten, wird ein Block auf die Disk geschrieben, womit dieser Teil des Kernspeichers wieder zur Verfügung steht (ein Block ist jener Bereich, der bei einem Zugriff vom Kernspeicher auf die Disk kopiert wird).

Um die Anzahl der Diskzugriffe zu verringern, die während der Gleichungslösung viel Zeit verbrauchen würden, werden die FGL neu organisiert, nachdem sie alle aufgestellt wurden. Genauso wie vorhin Skalare zu Submatrizen zusammengefaßt wurden, werden nun aufeinanderfolgende Zeilen und Spalten von Submatrizen zu größeren Matrizen zusammengefaßt, den sogenannten Partitionen. Partitionen werden also aus Submatrizen aufgebaut.

Solch eine Partition muß einige Bedingungen erfüllen; die Anzahl der Unbekanntenpunkte für jede Partition wird so bestimmt, daß im schlimmsten Fall jede Partition in jeweils einem Block gespeichert werden kann. Der schlimmste Fall ist jener, in dem die sich ergebende Partition aus dem Cholesky Reduktionsalgorithmus voll besetzt ist, d.h., daß keine Nullmatrizen in dieser Partition enthalten sind. Infolge dieses Axioms erhalten wir ein Partitionsschema, das durch ein rechtwinkeliges Raster definiert wird. Die Anzahl der Unbekannten kann von einer zur anderen Partition unterschiedlich sein.

Die folgende Abbildung zeigt ein Beispiel einer Partitionierung. Für die NGL-Partition mit der Zeilennummer zwei und der Spaltennummer vier sind die dazugehörigen Submatrizen eingezeichnet.



Nachdem der Raster für die Partitionierung bestimmt worden ist, werden die FGL umorganisiert. Üblicherweise ist die Anzahl der Submatrizen in einer Partition wesentlich kleiner als im oben erläuterten schlimmsten Fall. Daher können sehr oft nicht nur eine, sondern mehrere Partitionen in einem Block gespeichert werden.

3. Der Reduktionsalgorithmus

Nun folgt die Beschreibung, wie NGL aufgestellt und gelöst werden. Zunächst wird die Methode von Cholesky im skalaren Niveau zusammenfassend erläutert. Es sei $\mathbf{v} = \mathbf{Ax}-\mathbf{l}$ die FGL, $\mathbf{Nx} = \mathbf{A}^T\mathbf{l}$ die NGL. Es werden normalisierte FGL benützt, d.h. jede FGL wird bereits bei der Aufstellung mit der Quadratwurzel aus dem Gewicht der jeweiligen Beobachtung multipliziert. Daher erhalten wir: $\mathbf{Nx} = \mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{l}$ anstelle der üblichen Form: $\mathbf{Nx} = \mathbf{A}^T\mathbf{PAx} = \mathbf{A}^T\mathbf{Pl}$ mit der Gewichtsmatrix $\mathbf{P}$.

Das Element Nij ist das Skalarprodukt der Spalten i mit j der A-Matrix. Wir nennen diesen Vorgang Akkumulation. Auf diese Weise kann die **N**-Matrix berechnet werden. Da **N** symmetrisch ist, wird nur der obere dreieckige Teil von **N** berechnet.

Die **N**-Matrix kann als Produkt einer unteren mit einer oberen Dreiecksmatrix dargestellt werden; sie wird nach der Methode von Cholesky faktorisiert: $\mathbf{N} = R^T \cdot \mathbf{R}$:

$$R_{ii}^2 = R_{ii} \cdot R_{ii} = N_{ii} - \sum_{k=1}^{i-1} R_{ki} \cdot R_{ki} \qquad i = j$$

$$R_{ij} = R_{ii}^{-1} \cdot (N_{ij} - \sum_{k=1}^{i-1} R_{ki} \cdot R_{kj}) \qquad i < j$$

$$Rij = 0 \qquad i > j$$

Man kann diesen Algorithmus interpretieren als Multiplikation mit $R^{T^{-1}}$:

$$\mathbf{N}x = \mathbf{R}^T \mathbf{R}x = \mathbf{A}^T\mathbf{l}$$
$$\mathbf{R}x = R^{T^{-1}} \cdot \mathbf{A}^T\mathbf{l} =: \mathbf{g}$$
$$\mathbf{R}x = \mathbf{g}$$

Wir nennen dies Reduktion.

Die Berechnung wird spaltenweise von links nach rechts durchgeführt. Die Reduktion einer Spalte beginnt bei Zeile eins und endet beim jeweiligen Diagonalelement.

Sobald **N** in das Produkt $R^T \cdot R$ aufgespalten wurde, können die Unbekannten durch sogenanntes Rückwärtseinsetzen berechnet werden, d.h. es wird das dreieckige Gleichungssystem $Rx = g$ gelöst.

ORIENT verwendet diesen Algorithmus mit zwei Besonderheiten:

a) Der Algorithmus wird auf alle Niveaus der hierarchischen Struktur angewendet, beginnend beim Partitionsniveau, hinunter zum Submatrizenniveau und hinunter zum Skalarniveau.

b) Akkumulation und Reduktion erfolgen nicht nacheinander, sondern überlappend.

ad a) Die obigen Formeln für die Akkumulation und die Reduktion bleiben auch dann gültig, wenn die Skalare als Submatrizen oder als Partitionen interpretiert werden.

$$R_{ii}^T \cdot R_{ii} = N_{ii} - \sum_{k=1}^{i-1} R_{ki}^T \cdot R_{ki} \qquad i = j$$

$$R_{ij} = R_{ii}^{T^{-1}} \cdot (N_{ij} - \sum_{k=1}^{i-1} R_{ki}^T \cdot R_{kj}) \qquad i < j$$

Für die Operationen mit Partitionen und mit Submatrizen wird Matrixalgebra angewendet. Innerhalb jeder zu bearbeitenden Partition wird Cholesky's Algorithmus auf Submatrizenniveau angewendet, und innerhalb jeder zu bearbeitenden Submatrix wird Cholesky's Algorithmus auf Skalarniveau angewendet. Das Produkt zweier von

Nullmatrizen verschiedenen Submatrizen ergibt immer eine von der Nullmatrix verschiedene Submatrix. Das Produkt zweier Nicht-Nullpartitionen - Nullpartitionen sind Partitionen, die nur aus Nullen bestehen, also nicht gespeichert werden - kann eine von der Nullpartition verschiedene Partition ergeben, muß aber nicht. Mit anderen Worten, Partitionen können orthogonal sein.

Die R-Matrix kann Partitionen enthalten, die in der N-Matrix noch nicht vorhanden, also Null waren. Wir nennen solche Partitionen, die erst durch die Reduktion entstehen fill-in-Partitionen.

Auf Partitionsniveau und auf Submatrizenniveau wird der Algorithmus für Sparsematrizen angewendet. Auf Skalarniveau wird der Algorithmus für voll besetzte Matrizen angewendet, da hier alle Zahlen einer Submatrix, also auch eventuell auftretende Nullen, gespeichert werden.

Im Kernspeicher muß Speicherplatz für zwei Eingangspartitionen als Operanden und für eine Ausgangspartition für den Akkumulations- und Reduktionsalgorithmus vorhanden sein. Andererseits wird der zur Verfügung stehende Kernspeicher bestmöglich ausgenutzt. Auf einem Hintergrundspeicher, üblicherweise einer Disk, wird erst dann Information aus dem Kernspeicher ausgelagert, wenn nicht mehr genügend Speicher frei ist und daher ohne Auslagerung Daten überschrieben würden. Die Organisation in Partitionen ist speziell für den oben beschriebenen Algorithmus geeignet und erlaubt einen effizienten Datenaustausch zwischen Disk und Kernspeicher.

ad b) Nach Abschluß der Akkumulation für eine Partition erfolgt sofort deren Reduktion, da alle nötige Information bereits vorhanden ist. Erst nach dieser Reduktion wird die nächste Partition akkumuliert und dann reduziert. Auf diese Weise kann die Struktur von R leicht sequentiell aufgebaut werden. Anderenfalls, wenn die Reduktion erst nach der Akkumulation der vollständigen **N**-Matrix erfolgte, müßten die fill-in-Partitionen in eine bereits bestehende Struktur eingefügt werden, was einige programmiertechnische Komplikationen mit sich brächte.

4. Das Problem der Singularitäten

Dieser Abschnitt beschreibt, wie Singularitäten erkannt und behandelt werden.

Während der Reduktion kann es vorkommen, daß ein Diagonalelement auf Skalarniveau Null wird (sogenannte mathematische oder wahre Singularität), oder fast Null wird (sogenannte numerische Singularität). In so einem Fall kann die entsprechende Unbekannte nicht eindeutig berechnet werden. Man sagt, diese Unbekannte ist singulär.

Zur Vermeidung numerischer Probleme wird eine Singularität in ORIENT wie folgt definiert:

$$\frac{N_{ii} - \sum_{k=1}^{i-1} R_{ki}^2}{N_{ii}} < \varepsilon,$$

ε wird üblicherweise 10^{-6} für 32 bit Real-worte gewählt. Diese Formel ist selbstskalierend, womit Probleme mit der absoluten Größe und den Einheiten der Koeffizienten der Gleichungen vermieden werden.

Ein Grund für eine wahre Singularität könnte beispielsweise ein räumlicher Vorwärtsschnitt sein, bei dem ein Objektpunkt in nur einem Bild beobachtet wurde.

Die Singularitäten werden von ORIENT gemeldet und während der Reduktion beseitigt. Zu diesem Zweck wird eine weitere FGL erzeugt, die die entsprechende

Geodimeter 444

NEU

= hohe Genauigkeit 0,3 mgon und ± (2 mm + 3 ppm)
= RS-232 C-Zweiwegeschnittstelle
= Sekundenauflösung 0,1 mgon
= Reichweite bis 7000 m
= alphanumerische Tastatur
= Standby-Funktion

Bitte senden Sie mir weitere Informationen:

Name: _____

Anschrift: _____

_____

_____

Geodimeter Ges.m.b.H., Prinz-Eugen-Straße 72, 1041 Wien, Tel. (0222) 505 57 54, Telex 1

Geodimeter

Unbekannte mit ihrem momentanen Wert beobachtet (v = x). Diese zusätzliche Gleichung beeinflußt den bis dahin reduzierten Teil der R-Matrix überhaupt nicht. Diese Gleichung erhält ein hohes Gewicht, sodaß die singuläre Unbekannte berechnet werden kann. Diese Unbekannte bleibt nach dem Ausgleich also auf ihrem Näherungswert.

Auf diese Art können die NGL sinnvoll gelöst werden, selbst wenn Singularitäten auftreten.

5. Das Problem der Reihenfolge der Unbekannten

Die Reihenfolge der Unbekannten beeinflußt sehr stark die CPU- und I/O-Zeit, die für die Reduktion verbraucht wird.

Es gibt viele Algorithmen, die die Unbekannten so umordnen, daß möglichst wenig Fill-in entsteht [1]. Aber auch die Umsortierung erfordert recht beachtliche CPU- und I/O-Zeiten, sodaß ein optimaler Kompromiß gefunden werden muß. Derzeit verwendet ORIENT immer dieselbe vordefinierte Reihenfolge. Den größten Teil der Unbekannten bilden meistens die Objektpunkte. Wenn diese am Beginn des Unbekanntenvektors stehen, wird der entsprechende Teil der NGL - bis hin zum Submatrizenniveau - blockdiagonal. Ein solcher Blockdiagonalteil kann sehr rasch reduziert werden. Daher lautet die vordefinierte Reihenfolge: Objektpunkte, dann Projektionszentren und dann alle anderen Transformationsparameter. Die inneren Orientierungen stehen am Ende des Unbekanntenvektors, da diese Unbekannten viele andere miteinander verknüpfen [2].

6. Fehlerrechnung, innere und äußere Zuverlässigkeit

Als Ergebnis einer Ausgleichung werden nicht nur die Unbekannten gewünscht. Es könnten vielmehr Informationen gebraucht werden über die mittleren Fehler der

- Unbekannten (aus der $\mathbf{Q}_{xx}$-Matrix abzuleiten)
  oder
- Verbesserungen (aus der $\mathbf{Q}_{vv}$-Matrix abzuleiten), die nötig sind, um grobe Fehler zu entdecken.

Für beide Fragestellungen brauchen wir nur die Diagonalelemente, oder einige Elemente in der Nähe der Diagonale der $\mathbf{Q}_{xx}$- bzw. $\mathbf{Q}_{vv}$-Matrix. Die $\mathbf{Q}_{xx}$-Matrix ist definiert als invertierte NGL-Matrix:

$$\mathbf{Q}_{xx} = \mathbf{N}^{-1}$$

Die invertierte NGL-Matrix spielt auch eine Rolle für die $\mathbf{Q}_{vv}$-Matrix:

$$\mathbf{Q}_{vv} = \mathbf{E} - \mathbf{A} \cdot \mathbf{N}^{-1} \cdot \mathbf{A}^T$$

Diese einfache Form der $\mathbf{Q}_{vv}$-Matrix, aufgebaut mit der Einheitsmatrix $\mathbf{E}$ anstelle der invertierten Gewichtsmatrix $\mathbf{Q}_{ll} = \mathbf{P}^{-1}$, kann deshalb verwendet werden, weil $\mathbf{P}$ bereits berücksichtigt wurde, indem normalisierte FGL aufgestellt wurden.

Die invertierte NGL-Matrix spielt mathematisch eine große Rolle, aber nicht rechentechnisch: Es ist nicht nötig, die Inversion durchzuführen. Jede beliebige Submatrix der $\mathbf{Q}_{xx}$ bzw. $\mathbf{Q}_{vv}$-Matrix kann alleine mit dem Choleskyfaktor R berechnet werden:

$$\mathbf{Q}_{xx} = \mathbf{R}^{-1} \cdot \mathbf{R}^{T^{-1}} =: \mathbf{B}^T \cdot \mathbf{B}$$

$$\mathbf{Q}_{vv} = \mathbf{E} - \mathbf{A}\mathbf{R}^{-1} \cdot \mathbf{R}^{T^{-1}} \cdot \mathbf{A}^T =: \mathbf{E} - \mathbf{C}^T \cdot \mathbf{C}$$

Da der Reduktionsalgorithmus die Matrix **N** mit $\mathbf{R}^{T^{-1}}$ multipliziert:

$$\mathbf{R}^{T^{-1}} \cdot \mathbf{N} = \mathbf{R}^{T^{-1}} \cdot \mathbf{R}^T \mathbf{R} = \mathbf{R}$$

können wir diesen bei bekanntem **R** anwenden auf:

- $\mathbf{A}^T\mathbf{l}$ zwecks $g = \mathbf{R}^{T^{-1}} \cdot \mathbf{A}^T\mathbf{l}$
  (als Zwischenergebnis auf dem Weg zu x)
- **E** zwecks $B = \mathbf{R}^{T^{-1}}$
- $\mathbf{A}^T$ zwecks $\mathbf{C} = \mathbf{R}^{T^{-1}} \cdot \mathbf{A}^T$

Der Reduktionsalgorithmus liefert partitionierte Matrizen B und **C**. Die Matrizen **E** und $\mathbf{A}^T$ müssen geeignet partitioniert werden, um den Algorithmus anwenden zu können. Jede Teilmenge von Spalten von E und AT, die mit dem Algorithmus behandelt wird, führt zu den entsprechenden reduzierten Spalten. Dann müssen nur noch die diagonalen Submatrizen der Produkte $\mathbf{B}^T \cdot B$ und $\mathbf{C}^T \cdot \mathbf{C}$ berechnet werden, um alle wesentlichen Submatrizen von $\mathbf{Q}_{xx}$ und $\mathbf{Q}_{vv}$ zu erhalten.

Mit dem Choleskyfaktor **R** können somit alle interessierenden Submatrizen der $\mathbf{Q}_{xx}$- und $\mathbf{Q}_{vv}$-Matrix berechnet werden, ohne **N** explizit invertieren zu müssen, was viel Speicher und CPU-Zeit einspart.

Die innere Zuverlässigkeit [4,5] ist definiert als jene Größe, die angibt, wie groß ein Beobachtungsfehler

$$E(|\Delta l|)_{rel} = \frac{\delta_o \cdot \sigma_{l_i}}{\sqrt{r_i}}$$

δ_o =: Nichtzentralitätsparameter, der meistens 4 gesetzt wird

sein muß, sodaß er von einem statistischen Test, der normalisierte Residuen verwendet, mit einer bestimmten Wahrscheinlichkeit (93%) als solcher erkannt wird. Dieser Wert kann aus den Diagonalsubmatrizen der $\mathbf{Q}_{vv}$-Matrix berechnet werden.

Die äußere Zuverlässigkeit [4,5] gibt den maximalen Einfluß jenes Beobachtungsfehlers $E(|\Delta l|)_{rel}$ auf die Unbekannten an, der mit einer bestimmten Wahrscheinlichkeit (7%) gerade nicht als grob falsch entdeckt wurde. Es ist nicht angebracht, hier das Problem detailliert zu schildern. Wir beschränken uns bloß auf die Berechnung der Zuverlässigkeit.

Ein Beobachtungsfehler Δl_i hat den Einfluß Δx_i auf jede der Unbekannten:

$$\Delta X_i = \mathbf{N}^{-1} \mathbf{A}^T \Delta l_i$$

Betrachten wir die Beobachtung i, so erhalten wir für Δl_i zunächst einen Nullvektor - nur seine i-te Komponente erhält den Wert seiner inneren Zuverlässigkeit $E(|\Delta l_i|)_{rel}$, wie sie oben berechnet wurde.

Für alle Beobachtungen zusammen erhalten wir:

$$\Delta x = \mathbf{N}^{-1} \mathbf{A}^T \text{diag } E(|\Delta l|) rel_i) =: \mathbf{N}^{-1} \mathbf{A}^T \Delta L$$

zusammen mit $\mathbf{R}^{T^{-1}} \mathbf{A}^T = \mathbf{C}$ erhalten wir:

$$\Delta \mathbf{X} = \mathbf{R}^{-1} \cdot \mathbf{C} \cdot \Delta \mathbf{L}$$

Die Matrix C, die bei der Berechnung der Q_{vv}-Matrix angefallen ist, wird nun mit den Werten der inneren Zuverlässigkeit skaliert (Δ L ist diagonal), und dann erfolgt mit Hilfe von R für jede der Spalten von C Rückwärtseinsetzen, genauso wie bei der Lösung X der NGL:

Falls nicht genug Speicherplatz für die gesamte C-Matrix vorhanden ist, können die Zeilen der A-Matrix mit den Werten der inneren Zuverlässigkeit skaliert werden. Die weitere Vorgangsweise entspricht der Berechnung von Q_{vv}, wie oben beschrieben.

Benötigt werden die absoluten Maxima jeder Zeile der Matrix Δ X, die alle in einem Vektor gesammelt werden können.

Es ist unnötig zu betonen, daß der Rechenaufwand verglichen mit einer Lösung der NGL, die ausschließlich im Kernspeicher (Kernlösung) abläuft, sehr hoch ist. Aber es ist wichtig zu betonen, daß der beschriebene Algorithmus auch auf partitionierte Matrizen angewendet werden kann, die die Ausnützung der Sparsity (= dünne Besetzung) in einem hohen Maß gewährleistet. Und wieder müssen wir betonen, daß der partitionierte R-Faktor von Cholesky eine zentrale Rolle spielt, der zufolge seiner Dreiecksgestalt nicht explizit invertiert werden muß.

7. Vergleich der Rechenzeiten

Die folgende Tabelle gibt einen Überblick darüber, wieviel CPU-Zeit bei unterschiedlichen Projekten auf zwei Standardrechnern verbraucht wird.

Jedes Projekt ist als typisch für seine Klasse von Projekten anzusehen.

Projekt 1: Verkehrsunfall mit wenigen, aber sehr unterschiedlichen Unbekannten.
Projekt 2: Kammerkalibrierung.
Projekt 3: Terrestrische Präzisionsauswertung mit einer Meßkamera.
Projekt 4: Präzisionsauswertung mit Selbstkalibrierung.
Projekt 5: Aerotriangulation mit einer Meßkamera.

Als Rechner wurden eine Micro Vax II, sowie ein PC unter MSDOS mit einem 80386 Prozessor und einem 80387 Coprozessor verwendet, die jeweils mit einer Taktfrequenz von 20 MHZ arbeiten.

Man erkennt, daß auch am PC, auf dem relativ wenig Kernspeicher zur Verfügung steht, große Gleichungssysteme in angenehm kurzer Zeit gelöst werden können, und daß selbst so riesige Projekte wie die Aerotriangulation (Projekt 5) noch bearbeitet werden können.

PJ	NX	NO	OBJ	IOR	ROT	ADP	R		R + QXX	
							KERN	EXTERN	KERN	EXTERN
1	526	743	160	1	7	22	6	8	17	22
							10	27	40	72
2	1101	4174	353	1	13	0	52	65	102	137
							∞	456	∞	757
3	1194	3985	370	0	28	0	67	85	279	438
							∞	408	∞	1187
4	3388	9973	1106	2	20	4	322	231	683	760
							∞	1138	∞	3023
5	2586	4338	764	0	98	0	327	353	3512	4352
							∞	1022	∞	12321

PJ	Projektnummer aus obigem Text
NX	Anzahl Unbekannte
NO	Anzahl Beobachtungen
OBJ	Anzahl unbekannter Objektpunkte
IOR	Anzahl unbekannter innerer Orientierungen
ROT	Anzahl unbekannter Drehmatrizen
ADP	Anzahl unbekannter Polynomkoeffizienten, die für GESTALTen und Verzeichnung gebraucht werden.
R	CPU-Zeit in Sekunden für die Kern- bzw. externe NGL-Lösung.
R+QXX	CPU-Zeit in Sekunden für die Lösung der NGL inklusive Berechnung der mittleren Fehler.
∞	wegen zuwenig Kernspeicher unlösbar.

Die erste Zeile gibt die CPU-Zeit für die Micro Vax II an, die zweite Zeile für den PC.

Verwendete Abkürzungen:

NGL	Normalgleichungen
FGL	Fehlergleichungen
SM	Submatrix
CPU	Central Processor Unit
I/O	Input/Output

Literatur:

[1] *George, A., Lui, J.:* Computer Solution of large sparse positive definite systems. Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1981.
[2] *Hell, G.:* Terrestrische Bildtriangulation mit Berücksichtigung zusätzlicher Beobachtungen, DGK, Reihe C, Heft Nr. 252, München, 1979.
[3] *Kager, H.:* Das interaktive Programmsystem ORIENT im Einsatz. Presented Paper, 14. Kongress der Int. Gesellschaft für Photogrammetrie, Comm. V, Hamburg, 1980. International Archives of Photogrammetry XXIII, B 5, 1980, pp. 390-401.
[4] *Kraus, K.:* Photogrammetrie. Band 2, Dümmlers Verlag, Bonn, pp. 69-81, 1987.
[5] *Ackermann, F., Förstner, R., Mierlo, J.v.:* Vorträge des Lehrganges Numerische Photogrammetrie (IV) an der Universität Stuttgart, Schriftenreihe Inst. f. Photogrammetrie Stuttgart, Heft 7, 1981.

Diese Arbeit wurde vom Fonds zur Förderung der Wissenschaflichen Forschung unter der Projektnummer S3803 gefördert. Dieses Ausgleichsprogramm ist ein wichtiger Modul zu einem Programmpaket für die geometrische Rektifizierung von Scanneraufnahmen.

Manuskript eingelangt im Juli 1989