AN OUT OF CORE SOLUTION OF NORMAL EQUATIONS PROVIDING ALSO ACCURACY AND RELIABILITY DATA

Gsandtner M., Kager H.
Technical University Vienna
Gußhausstraße 27-29, A-1040 Wien
Austria
Commission III WG 1

Abstract:

ORIENT is a general bundle adjustment program /3/. The sparse normal equation (NEQ) 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.

1. The Hierarchical Structure of the Equations

As you will see later, the NEQs are produced and solved intermixed. Moreover, the structure of NEQs and observation equations (OEQs) is very similar. So we will start describing the structure of the OEQs.

Each observed point causes one, two or three OEQs. A photo point e.g. causes two of them, one for image x-coordinate and one for image y-coordinate; a model point causes three OEQs, a GESTALT-point (surface, curve) causes one or two OEQs, respectively. Each unknown that appears in an observation equation causes - in general - a non zero real number in the design matrix. When, e.g., the projection center for a photo is unknown, in each of the two photo equations three non zero real numbers are caused (because of the three object coordinates of the projection center). The observed photo point and the unknown projection center together cause a matrix of non zero real numbers in the design matrix (in our example a matrix of two rows and three columns). In general, the unknowns can be grouped into points, e.g. the object points or the "rotation point" containing the three rotation angles required for the transformation formula (e.g. the rotation angles of a photo as part of outer orientation). The unknowns within an unknown point are placed subsequently in the unknown vector. So each observed point causes, in connection with an unknown point appearing in this equation, a matrix of non zero real numbers in the design matrix. Let's call these matrices "submatrices" (SMs).

Also some management information must be associated to a SM: e.g. to which observation it belongs. Only those non zero SMs are stored. All other elements of the design matrix and of the

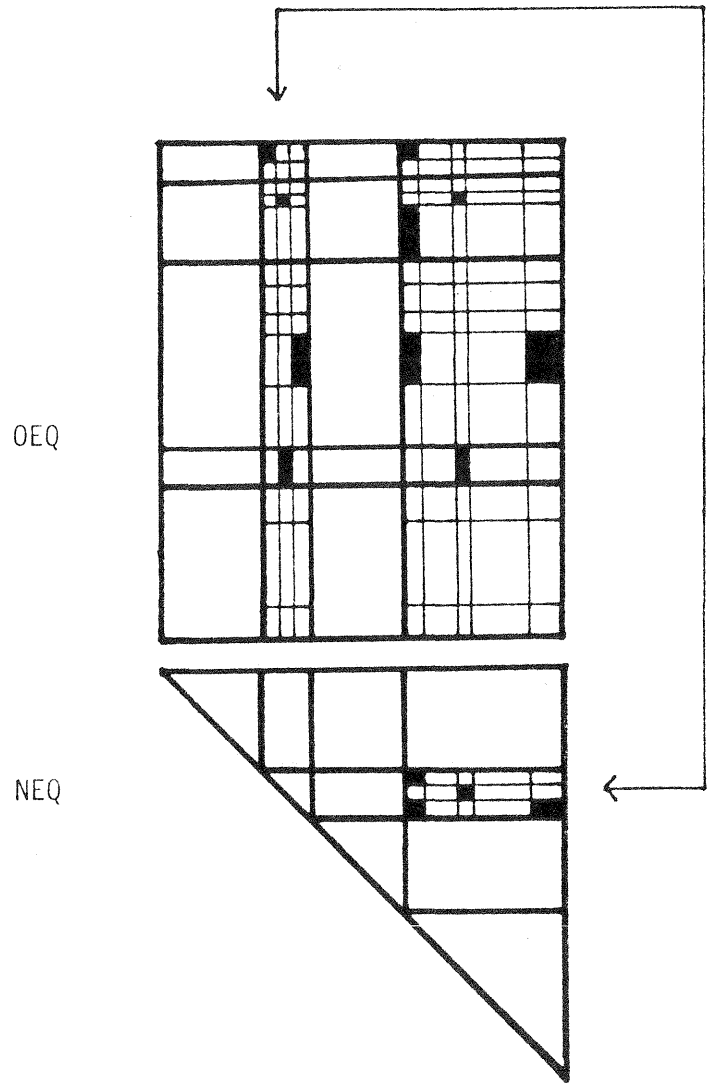
normal equation matrix are zero and are therefore not stored.

For each observed point, the SMs for the OEQs are computed. The SMs are delivered row by row and therefore they are stored rowwise. As soon as the SMs which are gathered sequentially in memory, exceed a certain buffer area, this block will be written to disk (a block is that area which is transferred between disk and memory at one access).

To decrease the count of block transfers which would consume very much time during solution, the OEQs are reorganized after they all have been computed. The system of SMs of equations are gathered into matrices containing some subsequent rows and some subsequent columns. Let's call these matrices (built of SMs) "partitions".

Such a partition must obey some conditions: the number of unknown points for each partition is computed assuming the worst case - that the resulting partition of the NEQs becomes full after factorization (no SM is missing, i.e. no zero-matrices are in that partition). And every partition (in the worst case) must be storeable in only one block. Due to this axiom you get a partitioning scheme, defined by a rectangular grid. The number of unknowns can vary from one partition column to another.

The following figure shows an example of such a partitioning. For one NEQ partition (partition row two, column four), the relevant SMs (which contain the non zeros of the equations) are also shown.



After the partitioning grid has been determined, the OEQs are reorganized. Usually, the actual number of SMs in a partition will be very much smaller than in the worst case. So, very often a set of some partitions instead of only one can be stored in one block.

2. The Factorization Algorithm

Now we will describe how the NEQs are computed and solved. At first, a summary of Cholesky's method operating at the scalar level is presented. Let $\mathbf{v} = \mathbf{Ax-l}$ the OEQs, $\mathbf{Nx} = \mathbf{A}^T\mathbf{l}$ the NEQs (when an OEQ is computed, it will be multiplied by the squareroot of the observation's weight. So we get simply $\mathbf{Nx} = \mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{l}$ instead of the usual form $\mathbf{Nx} = \mathbf{A}^T\mathbf{PAx} = \mathbf{A}^T\mathbf{Pl}$ with the weight matrix P).

The element N_{ij} (for $i \leq j$ since N is symmetric and we need only the upper triangular part of it) is the scalar product of the columns i and j of the **A** matrix. Let's this call accumulation. So the **N** matrix can be computed.

The **N** matrix can be factorized into a product of two upper triangular matrices ($\mathbf{N} = \mathbf{R}^T \cdot \mathbf{R}$) by the Cholesky method:

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

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

$$R_{ij} = 0 \qquad i > j$$

The effect of this algorithm to the normal equation system in matrix notation is:

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

Let's this call decumulation.

Computation is done column by column from left to right. The computation of a column starts at row one and ends at the respective diagonal element.

When **N** has been factorized completely, the unknowns will be computed by backward substitution (i.e. solving the triangular system $\mathbf{Rx} = \mathbf{g}$).

ORIENT uses this algorithm with two refinements:

a) The algorithm is done hierarchically, top down starting on the partition level down to the SM level and down to the scalar level.
b) Accumulation and decumulation are done intermixed.

ad a) The above formulas for accumulation and decumulation remain also correct when you interpret the real numbers as SMs or, more general, as partitions. Merely the operations have to be interpreted in a more general way:

$$\mathbf{R}_{ii}^T \cdot \mathbf{R}_{ii} = \mathbf{N}_{ii} - \sum_{k=1}^{i-1} \mathbf{R}_{ki}^T \cdot \mathbf{R}_{ki} \qquad i = j$$

$$\mathbf{R}_{ij} = \mathbf{R}_{ii}^{T-1} \cdot (\mathbf{N}_{ij} - \sum_{k=1}^{i-1} \mathbf{R}_{ki}^T \cdot \mathbf{R}_{kj}) \qquad i < j$$

For the operations with partitions and with SMs we have to use matrix algebra. Within each partition to be computed, Cholesky's algorithm is used on SM level; and within each SM to be computed Cholesky's algorithm is used on scalar level. The product of two non zero SMs causes always a non zero output SM. The product of two non zero partitions can cause a non zero output partition, but does not need to (non zero partitions may be orthogonal).

On partition level and on SM level an algorithm for sparse matrices is used; on scalar level an algorithm for full matrices is used, because all scalars of a SM are stored.

Storage for two input partitions as operands and one

output partition for accumulation and decumulation must be provided in the computer's memory at the same time. The required input partitions are loaded from external storage (usually from disk) into memory and then the output partition will be computed. As soon as the output partitions fill a block, this block will be written to disk.

ad b) As soon as the accumulation for one partition is completed, we have already all information needed for the decumulation and so it will be performed at this instance. Then the next partition will be accumulated and then decumulated. Doing so, programming becomes easier, because the structure of the R matrix can be computed sequentially. Otherwise, when the accumulation of **N** would be performed completely before decumulation starts, we would get the socalled fill-in problem: fill-in partitions, which are generated during decumulation had to be included into an existing structure.

3. The Singularity Problem

Next, detecting and removing of singularities will be described.

During factorization, it is possible that a diagonal element of scalar level becomes zero (mathematical or true singularity) or nearly zero (numerical singularity). Then, the respective unknown is not computable uniquely. We say it is singular (to avoid numerical problems, the definition of a singularity in ORIENT is:

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

ε is chosen usually 10^{-6} for single precision arithmetic on 32 bit real words; this formula is selfscaling, avoiding problems with the estimation of size and dimension of the equation system's coefficients).

A reason for a true singularity could be (e.g.) a spatial intersection, where a point is observed in only one photo. The singularities are reported, protocolled, and removed during factorization. For this purpose, a new observation equation is generated, which observes the unknown directly with its current value (v = x). This equation has no consequences for the structure of the matrix just being factorized. This equation gets a great dummy weight and so the singular unknown can be computed; its value after adjustment remains mostly unchanged. So the NEQs can be solved even if singularities exist.

4. The Ordering Problem

The ordering of the unknown vector is very important for the amount of CPU- and IO-time doing the following factorization of NEQs by the Cholesky algorithm. There exist many algorithms which reorder the unknowns in such a manner that fill in will

be minimized /1/. But reordering also consumes a lot of CPU- and IO-time, so we have to look for a compromise. We will test some of these algorithms in the future. At this time ORIENT uses always the same predefined order of unknowns. The greatest part of the unknowns is constituted by the object points, in general. When they are placed at the beginning of the unknown vector, the respective part of the NEQs becomes block diagonal (on SM level); so, they can be factorized very quickly. Otherwise many fill-in occurs. So the order is: object points, then projection centers and origins of local coordinate systems (e.g. polar observations), then all other transformation parameters (inner orientation is placed at the end, because this kind of unknowns connects many other unknowns /2/).

5. Error Calculation, Inner-, and Outer Reliability

As an adjustment's result, not only the unknowns are requested; moreover we might need information about

- the accuracy of the unknowns (given in the Q_{xx} matrix)
- the accuracy of the discrepancies (as given by the redundancy numbers contained in the Q_{vv} matrix) needed to do data snooping (searching for gross errors).

For both questions, we generally need merely the diagonal elements or some elements nearby the diagonal of the Q_{xx}- resp. Q_{vv}-matrix. The Q_{xx}-matrix is defined as inverted NEQ-matrix:

$$Q_{xx} = N^{-1}$$

The inverted NEQ-matrix is also of great importance for the Q_{vv}-matrix:

$$Q_{vv} = I - A \cdot N^{-1} \cdot A^T$$

Remark: We may use the simple form of the Q_{vv}-matrix based upon the identity matrix I instead of the inverted weight matrix $Q_{ll} = P^{-1}$ because we have taken P already into account setting up normalized OEQs.

The inverted normal equation matrix is very important mathematically but not computationally: we do <u>not need</u> to compute it explicitly answering the above questions - we can compute any arbitrarily chosen set of diagonal SMs of Q_{xx} respective Q_{vv} based upon the Cholesky factor R:

$$Q_{xx} = R^{-1} \cdot R^{T^{-1}} =: B^T \cdot B$$
$$Q_{vv} = I - AR^{-1} \cdot R^{T^{-1}} A^T =: I - C^T \cdot C$$

Knowing that Cholesky's factorization (= decomposition) algorithm does nothing else than premultiplication of N by $R^{T^{-1}}$

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

giving the Cholesky factor R, we may apply it to

- $A^T l$ to get $g = R^{T^{-1}} \cdot A^T l$ (as intermediate result on the way to x)
- I to get $B = R^{T^{-1}}$
- A^T to get $C = R^{T^{-1}} \cdot A^T$.

The partitioned Cholesky algorithm yields matrices B and C partitioned in the same way. The matrices I and A^T have to be submitted to the algorithm suitably partitioned, also. Whenever subsets (regarding the columns) of I and A^T are submitted, the correspondingly decomposed columns are returned. Then, merely the diagonal SMs of the products $B^T \cdot B$ and $C^T \cdot C$ have to be computed to evaluate the interesting SMs of Q_{xx} and Q_{vv}. Since merely columns (on the SM- resp. partition-level) of B and C are needed, these column by column products are built (accumulated) interleaved with the decomposition process, again (c.f. the accumulation of **N**). This has the advantage that B and C need not to be stored in total - a scratch area for one column of partitions suffices.

Using Cholesky factorization, we can compute all interesting SMs of the Q_{xx}- and Q_{vv}-matrix without inverting **N** explicitly saving a lot of storage and computing time.

The inner reliability /4, 5/ is defined as that magnitude which tells how large an observation error

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

must be to be found by a statistical test using normalized residuals with a certain probability (93%). Its value may be computed directly from the redundancy numbers taken from the diagonal of the Q_{vv}-matrix as obtained above.

The outer reliability /4, 5/ tells the worst case effect of that observation error $E(|\Delta l|)_{rel}$ which was evenly not detected by a certain probability (7%), onto the unknowns. We have here no room to discuss the whole problem in detail, so we restrict ourselves onto the computational aspect, only.

An observation error Δl_i has the effect ΔX_i onto each of the unknowns:

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

for the observation i, we get Δl_i as zero vector at first - merely its i-th component contains the value of its inner reliability $E(|\Delta l_i|)_{rel}$ as computed above.

For all observations together, we get

$$\Delta X = N^{-1} A^T \operatorname{diag}(E(|\Delta l|)_{rel_i}) =: N^{-1} A^T \Delta L$$

using $R^{T^{-1}} A^T = C$ we get

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

If we had saved matrix C computing our Q_{vv}-matrix, we need now to scale its columns by the values of inner reliability (since ΔL is diagonal), and then to do backward substitution using R (as we did solving for the NEQ's solution) for each of C's columns.

If we have no storage capacity for the whole C matrix, we may scale the rows of the A-matrix by the values of inner

reliability and proceed as outlined above discussing the computation of Q_{vv}.

Effectively, we need the rowwise absolute maxima of the matrix ΔX which may be gathered in one vector.

There is no need to stress the fact that the computational effort is huge compared to "simple" NEQ solution; but we must stress the fact that the outlined algorithm is also workable on partitioned matrices saving sparsity to a high degree, and again we must stress the central role of the partitioned Cholesky R-factor which needs not be inverted explicitly due to its triangular shape.

Table of abbreviations:

NEQ	Normal Equation
OEQ	Observation Equation
SM	SubMatrix
CPU	Central Processing Unit
IO	Input/Output

References:

/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. Congress 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.