Bulletin Géodésique (1992) 66:296–305

Bulletin Géodésique

© Springer-Verlag 1992

A comparative study of algorithms for reducing the fill-in during Cholesky factorization

P. J. de Jonge *

Delft Geodetic Computing Centre (LGR), Faculty of Geodesy, Delft University of Technology, Thijsseweg 11, 2629 JA Delft, The Netherlands

Received May 21; Accepted May 22, 1992

Abstract

In this paper several ordering algorithms for the unknowns in geodetic least squares systems are compared. The comparison is restricted to the case of the well known Cholesky factorization of the normal matrix A into a lower triangular factor L.

The algorithms which were investigated are: minimum degree, minimum deficiency, nested dissection, reverse Cuthill-McKee, King's-, Snay's-, and Levy's-banker's and Gibbs-King.

Also some strategies are presented to reduce the time needed to compute the ordering using a priori information about the way the unknowns are connected to each other.

The algorithms are applied to normal matrices of the least squares adjustment of 2D geodetic terrestrial networks, photogrammetric bundle-block adjustments, and a photogrammetric adjustment using independent models.

The results show that ordering the unknowns yields a considerable decrease of the cpu time for computing the Cholesky factor, and that in general the minimum degree and Snay's banker's ordering perform best. Furtheron they show that a priori information about the connection structure of the unknowns speeds up the computation of the ordering substantially.

1 Introduction

One of the most time consuming steps in a least squares adjustment is the solution of the system of linear normal equations

$$Ax = b \tag{1}$$

where A is a square and symmetric positive definite n by n matrix. This is especially true if the system is very large which is quite common in the field of geodesy. Fortunately, often many entries of A are zero, and usually the fraction of nonzero entries (the 'fill') decreases with the size of the matrix.

If we distinguish between zero and nonzero entries ('nonzeros' and 'zeros') in a matrix, we have a sparse matrix. There are two important advantages when using sparse matrices. Firstly, if we operate only upon the nonzeros, we save time; secondly, we only have to store the nonzeros. We need however, special data structures and special algorithms which have an overhead in storage and computing time. Furthermore, we have to design our algorithms in a way that they use and preserve sparsity.

The algorithm used at the Delft Geodetic Computing Centre for the solution of the system in Eq. (1) is based upon Cholesky factorization, resulting in the decomposition of the normal matrix A into the product of a lower triangular matrix L and an upper triangular matrix L^T:

$$A = LL^T \tag{2}$$

We can solve for x by forward ($Lt = b$) and backward ($L^T x = t$) substitution.

If A is sparse then the Cholesky factor L generally is also sparse. L has at least the same nonzeros as the lower triangle of A (neglecting numerical cancellations), but usually some new nonzeros arise in L (the 'fill-in'). The amount of fill-in and its location depends on the order in which the pivots of the Cholesky factor are chosen, and hence on the order of the unknowns. The solution of the permuted system

$$(PAP^T)(Px) = Pb \tag{3}$$

with the orthogonal permutation matrix P and solution $x = P^T(Px)$, is identical (except for small round off) to the solution of the system of Eq. (1). Since there is no need to pivot i.e. exchange rows or columns for stability reasons, we are free to choose the order of the unknowns, or the permutation matrix P, in such a way as to minimize fill-in and hence computing time and storage requirements.

* Supported by the Netherlands Organization for Scientific Research (NWO)

2 Sparse matrices

The representation of a sparse matrix

When using sparse matrices we try to store and operate upon the nonzeros only; therefore we have to specify a storage scheme or data structure in order to store a sparse matrix in computer memory, and design algorithms which use this data structure in the most profitable sense. In this paper we distinguish between two data structures:

1. Nonzero storage; only the nonzeros are stored. The set of elements a_{ij} which are stored is called $Nz(A)$. If we only store the lower triangular part of A, which we usually do if A is symmetric, the set of elements is defined by

$$Nz(A) = \{a_{ij} | a_{ij} \neq 0, j \leq i\} \tag{4}$$

2. Envelope or variable band storage (for symmetric matrices only); all elements within the envelope around the diagonal, zeros and nonzeros alike, are stored. The set of elements a_{ij} which are stored is called $Env(A)$:

$$Env(A) = \{a_{ij} | f_i \leq j \leq i\} \tag{5}$$

where f_i is the first nonzero element in row i. Only half of the matrix is stored since the matrix is symmetric. The number of elements which are stored, $|Env(A)|$, is called the 'profile' and is defined by

$$|Env(A)| = n + \sum_{i=1}^{n}(i - f_i) = n + \sum_{i=1}^{n}\beta_i \tag{6}$$

where β_i is called the i-th or local bandwidth.

The envelope storage scheme involves less overhead and is faster in accessing individual elements than the nonzero storage scheme. However, also zeros within the envelope are stored and hence will be used in computations.

For the description and explanation of sparse matrix algorithms we use the concepts of graphs. A graph $G = (V, E)$ consists of a set of nodes or vertices V, and a set of edges E. We may associate the symmetric n by n matrix A with a labelled undirected graph $G^A = (V^A, E^A)$. The set of nodes corresponds to the diagonal entries of the matrix, the set of edges to the off-diagonal entries. Labelled means that each node has a unique number corresponding to a row, column or unknown; undirected means that we do not distinguish between the edge from node v to w and the edge from w to v, in other words the matrix is symmetric.

Two nodes v and w are adjacent in a graph if there is an edge between them, so the nonzero structure of A corresponds to a so-called 'adjacency structure' of G^A. This adjacency structure is usually stored by a nonzero storage scheme. The nodes which are adjacent to a node

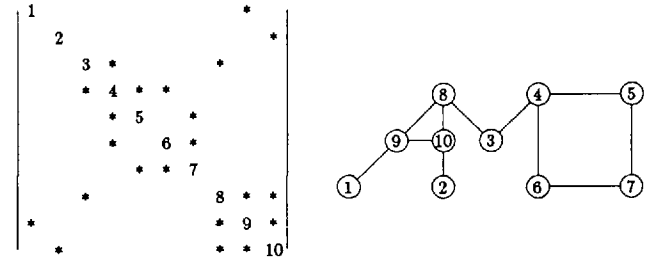

Figure 1: Symmetric matrix and its associated graph.

w are said to be in the adjacent set of w, denoted by $Adj(w)$. The number of nodes in $Adj(w)$ is called the degree of w denoted by $Deg(w)$. Simultaneously we may define the nodes adjacent to a set of nodes W by $Adj(W)$.

Cholesky factorization

One of the methods to perform Cholesky factorization is the outer product method. Since this method is well suited to explain a number of phenomena occurring during factorization, it will be described here:

1. $A_1 = A$
2. For $i = 1, n-1$
   2. (a) partition A_i into $\begin{bmatrix} d_i & \\ c_i & \bar{A}_{i+1} \end{bmatrix}$
      (b) $c_i \leftarrow c_i/\sqrt{d_i}$
      (c) $L(i:n, i) \leftarrow \begin{bmatrix} \sqrt{d_i} \\ c_i \end{bmatrix}$
      (d) $A_{i+1} \leftarrow \bar{A}_{i+1} - c_i c_i^T$
3. $L(n, n) \leftarrow \sqrt{A_n}$

The input is a n by n symmetric and positive definite matrix A with only the elements of the lower triangle stored.

At each step i we have the transformation of $\bar{A}_{i+1}$ into A_{i+1}. If we assume that exact numerical cancellation does not occur, then $(A_{i+1})_{jk}$ is nonzero if and only if $(\bar{A}_{i+1})_{jk}$ is nonzero, or the outer product element $(c_i c_i^T)_{jk}$ is nonzero. The outer product element $(c_i c_i^T)_{jk}$ is nonzero if and only if $(c_i)_j$ and $(c_i)_k$ are nonzero.

If we now introduce the associated graphs G_i and G_{i+1} of A_i and A_{i+1}, then one step of Cholesky factorization can be described as a graph transformation by the following recipe due to [Parter, 1961]

> to obtain G_{i+1} from G_i, delete node v_i and all edges incident to it, and add all possible edges between nodes which are adjacent to v_i in G_i

where node v_i is associated with the pivot d_i, and the set of nodes adjacent to v_i is associated with the nonzero elements in c_i.

The graphs $G_i, i = 1, 2 \ldots, n$ are called elimination graphs, and the whole process of Cholesky factorization can be seen as the formation of the sequence of elimination graphs:

$$G_1 \rightarrow G_2 \rightarrow \cdots \rightarrow G_n \tag{7}$$

The order in which nodes are eliminated influences the number of new formed edges. The storage and cpu time needed for the factorization depends for factorization based on a nonzero storage scheme on the amount and position of the nonzeros in the factor, and for factorization based on an envelope storage scheme on the size and form of the envelope. Table 1 gives the operation count for the two types of storage schemes as a function of the 'i-th frontwidth' h_i. For nonzero factorization h_i is equal to $|Nz(c_i)|$. For envelope factorization it is equal to the number of elements of c_i, zeros and nonzeros alike, which are within the envelope (the 'active' rows).

storage (exclusive administration)	$n + \sum_{i=1}^{n-1} h_i$
multiplicative operation count	$\frac{1}{2}\sum_{i=1}^{n-1} h_i(h_i + 3)$
additive operation count	$\frac{1}{2}\sum_{i=1}^{n-1} h_i(h_i + 1)$

Table 1: Operation count and storage for Cholesky factorization.

Ordering strategies

The ordering algorithms in this paper can be divided into two groups, viz. those that minimize the number of nonzeros in the Cholesky factor, and those that minimize the size of the envelope of the normal matrix.

The former group can be subdivided into the 'minimum fill strategy' where A is permuted such that at each step of the factorization L suffers minimum fill-in (section 3), and the 'dissection strategy' where the unknowns are partitioned in k sets and ordered in such a way that the fill-in is confined to certain regions as small as possible (section 4).

The objective of the 'minimum envelope strategy' is to permute A such that its envelope is as small as possible; the envelope of L equals the envelope of A, and fill-in will only occur within this envelope (section 5).

3 Minimum fill strategy

The minimum degree algorithm

In graph terminology Parter's recipe means that after elimination of v_i, its adjacent set becomes a 'clique', i.e. every node in this set is adjacent to every other node in it. In matrix terms it means a full sub-matrix which may be, depending on the ordering, scattered throughout the matrix.

If we want to minimize, at least locally, the fill we should eliminate at each step of the elimination process a node of minimum degree in the current elimination graph. In matrix terms this corresponds to the column/row with the least nonzeros in it. This is the basic idea of the minimum degree algorithm:

1. Compute the degree of all nodes in G_1.
2. For $i = 1, n-1$
   (a) Choose from G_i a node of minimum degree v_i and label it i.
   (b) Form G_{i+1} by eliminating v_i.
   (c) Update the degree of the nodes which were adjacent to v_i in G_i.
3. The last node v_n gets label n.

The difficulty with the minimum degree algorithm is that the graph changes during the elimination, both in size as in structure. Therefore we have to insert and delete edges from the adjacency structure of the elimination graph. There are several algorithms which can handle this. We have tested the following two:

1. the quotient minimum degree algorithm of [George and Liu, 1980] where elimination is modelled implicitly using 'quotient graphs'.
2. the linked lists minimum degree algorithm where elimination is modelled explicitly by formation of the elimination graphs, using a circular linked list data structure for the successive elimination graphs [de Jonge, 1991].

The latter algorithm has been developed at the Delft Geodetic Computer Centre, and is a derivative of the algorithm of George and Liu in the sense that we have 'borrowed' some concepts from it. The most important feature is the notion that during elimination some nodes become 'indistinguishable' with respect to elimination. It means for the adjacent nodes v and w that $Adj(v) - w = Adj(w) - v$, which implies that we can combine them into one so-called 'super node'. This can speed up the computation of the ordering considerably. As we shall see in section 6, in geodesy we often can point out indistinguishable nodes before the computation of the ordering starts.

The minimum deficiency algorithm

The minimum degree strategy does not guarantee that at each step the number of new edges is minimized (the fill-in). If this is our purpose we have to apply the minimum deficiency algorithm. Then we have to examine at each step for each non-labelled node how many new edges will be formed if we eliminate it, which is computationally far more complex. Although this algorithm at each step locally minimizes the fill-in, the resulting global fill-in is for a large class of graphs of the same order of the fill-in of a graph ordered by a 'good' minimum degree algorithm.

4 Dissection strategy

The rooted level structure of a graph and pseudo-peripheral nodes

The nested dissection algorithm described in [George and Liu, 1981], the (reverse) Cuthill-McKee algorithm and the Gibbs algorithms are all based on a rooted level structure of the graph. Such a rooted level structure rooted at node x (the 'root') of $G = (V, E)$ is defined as the disjoint partitioning

$$\mathcal{L}(v) = \{L_0(v), L_1(v), L_2(v), \ldots, L_{e(v)}(v)\} \qquad (8)$$

where $e(v)$ is called the 'eccentricity' of node v, $L_0(v) = \{v\}$, and $L_i(v), i = 1, \ldots, e(v)$ consists of the nodes adjacent to L_{i-1} which are not already in one of the previous levels. Thus nodes in $L_i(v), i = 1, \ldots, e(v) - 1$ are only adjacent to nodes of levels $i - 1, i, i + 1$.

The length of $\mathcal{L}(v)$, defined as the number of levels minus 1, is equal to the eccentricity $e(v)$ of its root v. The width of $\mathcal{L}(v)$ is defined as the maximum number of nodes in any level.

For a good performance of the level structure based ordering algorithms, a level structure containing many small levels is needed (i.e. a level structure with a small width and a large length).

Such a level structure is usually obtained if we take for the root a node of high or even maximum eccentricity (a peripheral node). Algorithms to find a peripheral node are very time consuming. Usually an algorithm proposed by [Gibbs et al. 1976], or the adjusted one by [George and Liu, 1979] to find a so-called 'pseudo-peripheral' node is used. The latter is as follows:

Choose an arbitrary node v in V, and generate its rooted level structure $\mathcal{L}(v)$. The eccentricity of any node in the last level of $\mathcal{L}(v)$, is at least equal to the length of $\mathcal{L}(v)$ (= eccentricity of v), but it may be larger. Therefore generate $\mathcal{L}(w)$ with w a node in this last level of minimum degree (since such a node is even more likely to have a large eccentricity) and compare its eccentricity with that of v. If it is equal, a pseudo-peripheral node has been found, otherwise repeat the whole procedure with a node of minimum degree from the last level of $\mathcal{L}(w)$.

The nested dissection algorithm

The thought behind the method of nested dissection is the following: the associated graph is systematically partitioned using 'separators'. A separator is as set of nodes which by its removal splits the graph in two or more components. When a separator is found, its nodes are labelled and removed from the graph, thus leaving the graph partitioned into two or more components. Good separators are those which are as small as possible and lead to a partitioning with components comparable in size.

Subsequently, separators are found for each component and the procedure is continued until no separators can be found. If at a certain stage no separator can be found for a component, the whole component is labelled. After this dissection, the labelling is reversed.

Essential for the nested dissection labelling is that the separator is labelled after the components it divides. The fill-in is confined to the areas which are formed by the intersection of the separator with the components divided by it.

In [George and Liu, 1981] a nested dissection scheme is given based on a level structure rooted at a pseudo-peripheral node w. Each level is a separator of the graph although not a minimal one. A small separator which splits a graph in comparably sized components can be obtained by taking only those nodes of the 'middle' $L_i(w)$, which are adjacent to some node in $L_{i+1}(w)$. It can be shown that if the separators are chosen in this way, the internal labelling of their nodes does not influence the fill-in.

5 Minimum envelope strategy

The (reverse) Cuthill-McKee algorithm

The Cuthill-McKee algorithm [Cuthill and McKee, 1969] was the first non-iterative envelope reducing algorithm. The principle is the following: Let v be a labelled node, and w an unlabeled adjacent node. To minimize the local bandwidth it is necessary to label node w as soon as possible after v. By this strategy both the bandwidth as well as the envelope are reduced.

Soon it was noticed that reversing the labelling results in an equally sized bandwidth, and a smaller or equally sized envelope.

The Cuthill-McKee algorithm is based on a rooted level structure of the graph; the nodes are labelled level by level. In each level internally the nodes adjacent to

the lowest labelled node in the previous level are labelled first in order of increasing degree and so on until the highest labelled node in the preceding level. If we reverse the labelling we get the reverse Cuthill-McKee labelling:

1. Determine a starting node v_1 (preferably a pseudo-peripheral one) and label it.
2. For $i = 1, \ldots, n$ find all unlabeled nodes in $Adj(v_i)$ and label them in increasing order of degree.
3. Reverse the labelling.

The banker's algorithms

The banker's algorithm, published in [Snay, 1976], and the algorithms published in [King, 1970] and [Levy, 1971] are essentially the same. The algorithm got his name due to an analogy with a banker's problem presented in the paper of Snay.

Recall that the number of active rows in column i is called the i-th frontwidth. The number of elements within the envelope equals the sum of the i-th frontwidths. If we want to minimize (at least locally) the size of the envelope we should minimize this frontwidth by allowing a minimum growth of it. The price paid for this, is that the bandwidth may become very large.

The algorithms of Levy, King and Snay all use this strategy. They only differ with respect to the way in which nodes are considered a candidate for labelling.

More formally than above, we can describe the algorithm as follows, where V as before is the set of all nodes, L the set of labelled nodes, C the set of candidated nodes, and H the set of 'hopeful' nodes. Nodes are called hopeful if they are adjacent to already labelled nodes, again according to this banker's analogy.

1. Determine a starting node (we take a pseudo-peripheral node found by the scheme of George and Liu) v_1, and label it; $L = v_1$
2. For $i = 2, n$
   (a) $H = Adj(L)$
   (b) $C = H$ (King)
   $C = H \cup (Adj(H) - L)$ (Snay)
   $C = V - L$ (Levy)
   (c) Label next node $v_i \in C$ which minimizes: $|W| - d$, where
   $W = Adj(v_i) - H$ and $d = \begin{cases} 0 \text{ if } v_i \in H \\ 1 \text{ if } v_i \notin H \end{cases}$
   (d) $L = L \cup v_i$

$(|H| + |W| - d)$ equals the i-th frontwidth, where i is the current label, and thus represents the change of the profile. When there is more than one node that fulfills the above mentioned minimum, some kind of tie breaking strategy may be provided. In [King, 1970], [de Jonge, 1987] some strategies are discussed, but it is our current opinion that unless one has to deal very often with the same class of regular graphs, one should not try to find a strategy, and break ties arbitrarily. There is not a tie break strategy that gives good results for all classes of graphs, and besides, it takes time to compute.

The Gibbs-algorithms

The Gibbs-Poole-Stockmeyer algorithm [Gibbs et al. 1976] and the Gibbs-King algorithm [Gibbs, 1976] are both based on the variant of the rooted level structure described in section 4. It is a combination of a level structure rooted at a pseudo-peripheral node, and one rooted at a pseudo-peripheral node in the last level of it. It aims at minimizing the level width still further, while the level length stays the same; it is however more complex.

Again the nodes are labelled level by level. In the Gibbs-Poole-Stockmeyer algorithm which primarily aims at reducing the bandwidth, the labelling within the levels is accomplished using the reverse Cuthill-McKee strategy. In the Gibbs-King algorithm, which aims primarily at reducing the envelope, King's banker's strategy is used. We have only tested the Gibbs-King algorithm.

6 Performance of the algorithms

Introduction

The ordering algorithms have been tested on several geodetic problems, viz. the adjustment of two 2-D terrestrial networks, and four 3-D photogrammetric networks. All test runs were made on a DEC-VAX 751 in a single- or semi-single-user interactive mode. The routines were compiled by the VAX Fortran v5.2 compiler. The cpu times are in seconds and we estimate them to be repeatable within approximately 5 %; two digits after the decimal point are given however, to make comparisons between the several orderings possible. The following algorithms and implementations were tested:

0.	NATural or original ordering	-
1.	Quotient Minimum Degree	Sparspak
2.	Linked lists Minimum Degree	LGR
3.	Linked lists Minimum DeFiciency	LGR
4.	Nested Dissection	Sparspak
5.	Reverse Cuthill-McKee	Sparspak
6.	KING's banker's	LGR
7.	SNAY's banker's	LGR
8.	LEVY's banker's	LGR
9.	Gibbs-King	ToMS

Sparspak is a sparse matrix package developed at the University of Waterloo, Canada. We have only tested the version described in [George and Liu, 1981] adapted to our data structures. The new licensed version is much faster, (see e.g. [George and Liu, 1989] for developments of the minimum degree algorithm), but does not run on our computer. The implementations marked by LGR are developed by the Delft Geodetic Computing Centre. ToMS is a library with the collected algorithms from the 'ACM Transactions on Mathematical Software'. The material in this library is copyrighted by the ACM (Association of Computing Machinery), which "grants general permission to distribute provided the copies are not made for direct commercial advantage". It is available from the 'NETLIB' network (see [Dongarra et al. 1987]), where one can also obtain the Sparspak routines.

For Cholesky factorization we used two schemes, both have as input the nonzero stored lower triangular normal matrix, the output however consists of the nonzero stored factor, resp. the envelope stored factor. Also the computation of the 'sparse inverse' has been used as a benchmark for the orderings. A sparse inverse scheme only computes those elements of the inverse which are nonzero in the Cholesky factor, see e.g. [van der Marel, 1988].

Test matrices

For the surveying networks we tested a cadastral network in the northern part of the Netherlands (ZK85), and a network in Oman from Petroleum Development Oman (OMAN). It are free networks, so the rank deficiency is four, thus four nodes (the S-basis) are - with their incident edges - excluded from the graph.

Unknown parameters are the cartesian x- and y-coordinates of the points; nuisance parameters are an orientation for every point at which direction measurements have been made, and one or more scale factors. The x- and y-coordinate, together with the orientation parameter, form a clique. The x- and y-coordinate are from the start of the elimination indistinguishable.

A set of coordinates is connected to an other set of coordinates by a distance or direction measurement. An orientation parameter is connected to the set of coordinates from which the direction measurement has been made, and to the set of coordinates of the points aimed at. The scale factor is connected to every x- and y-coordinate of a point from which distances have been measured; its initial degree is thus very high.

The original or 'natural' ordering is: coordinates (x- and y-coordinate of one point are together), orientation parameters, scale factor(s).

For the photogrammetric networks we have tested normal matrices, originating from two types of adjustments:

number of:	ZK85	OMAN
points	180	2008
orientation parameters	148	2352
scale factors	1	2
unknowns	509	6370

Table 2: Some characteristics for the terrestrial networks

Firstly, some matrices which originate from a bundle-block adjustment (BU [number of strips] * [number of photos in a strip]). The unknown parameters are cartesian x- , y- and z-coordinates of the terrain points. Nuisance parameters are cartesian x-, y- and z-coordinates of the principal point, and the rotation parameters κ, ω, and ϕ of the camera. The three coordinates of the terrain points form a clique, and are indistinguishable with respect to elimination from the start. The same holds for the coordinates of the principal point together with the rotation parameters of the camera.

The original or 'natural' ordering for the bundle-block adjustment is: 1. the terrain points per row perpendicular to the direction of flight; 2. the photos in increasing order per strip. The internal order of the x-, y- and z-coordinate within a point is arbitrary, as is the internal order of the x-, y- and z-coordinate, and ω, κ, and ϕ within a photo.

Secondly, a matrix which originates from the adjustment with independent models (INMO). Unknown parameters are again x-, y- and z-coordinates of terrain points, and there are 7 nuisance parameters per model.

For the photogrammetric networks we also considered only free networks; so rank deficiency is 7.

number of:	INMO	BU 3*7	BU 3*21	BU 7*21
points	196	49	147	315
models/photos	41	21	63	147
unknowns	881	273	819	1827

Table 3: Some characteristics for the photogrammetric networks

Special ordering techniques for geodetic networks

Since the initial degree of the scale factor in a terrestrial network is very high compared to the degree of all other nodes, we expected it to be ordered last when applying minimum degree. Inspection of several ordering results confirmed this; the scale factor is part of the last ordered set of indistinguishable nodes.

As the scale factor is connected to a majority of the nodes in the initial graph, a substantial part of the time needed to update the successive elimination graphs can

be attributed to this unknown. If we exclude it from the graph and order it manually as last, a considerable decrease of cpu time was found, as expected; in the cases we have investigated, also a decrease of fill-in has been observed.

A decrease in cpu time and a decrease of elements in the envelope was also observed when this was applied to the banker's algorithms. The decrease in cpu time is again explained by the fact that the scale factor has such a high degree; it is involved in almost every updating cycle of the algorithm.

For the reverse Cuthill-McKee and Gibbs-King algorithm besides the decrease of cpu time a large decrease of elements in the envelope was observed. This is explained by the fact that these algorithms are based on a rooted level structure. For good results this structure should have many small levels (a level structure of small width). If the scale factor is included in the graph, the scale factor will inevitably appear in one of the first levels due to its high degree. The next level will contain all coordinates of points which are involved in a distance measurement and which have not already been placed in a level: thus causing a wide level structure. This is prevented by putting the scale factor at the end of the ordering, i.e. leaving it out the level structure.

Generally all global instrumental parameters should be handled this way: for instance in a bundle-block adjustment the camera constant (in our examples this parameter was not modelled).

Other adaptations of the algorithms to the specific structure of geodetic networks is the notion of indistinguishability before the ordering is started. Hereby the size of the graph, i.e. the number of nodes and the number of edges is reduced by introducing super nodes. Besides, since the order $\mathcal{O}$ of the computation time is a function of the number of nodes and the number of edges, it is reduced too. Until now this has only be accomplished for the LMD and LMDF algorithms, but it can be applied to all orderings described in this paper.

Results for the terrestrial networks

The timing results for the two surveying networks are given in table 5 and 6. Table 4 contains the legend for tables 5,6,7,8,9 and 10. For the average sized networks (see table 5) one of the minimum degree algorithms, followed by a nonzero factorization, is the best choice, with the linked lists minimum degree the fastest of the two. The nonzero factorization is superior, even despite the additional overhead due to its more complicated data structure.

If one wants to use an envelope factorization scheme (which is easier to program than the true nonzero factorization scheme), reverse Cuthill-McKee, Snay's banker's or Gibbs-King should be chosen. We prefer to use Snay's banker's since this ordering gives also good results if there are one or more nodes with a high degree.

name	name of the ordering algorithm
'NAME' 1	scale unknown(s) forced to be last
'NAME' 2	a priori formation of super nodes
'NAME' 3	combination of 1 and 2
$\|Nz(.)\|$	number of nonzero elements of a triangle stored matrix
$\|Env(.)\|$	number of elements within the envelope of a matrix
order	cpu time for ordering
fact	cpu time for Cholesky factorization
spinv	cpu time for sparse inverse
total	'order' + 'fact' + 'spinv' + cpu time for permutation of the normal matrix
A_f_l	partial normal matrix; only rows/columns between f and l are included

Table 4: Legend for tables 5-10.

For large networks (see table 6) only minimum degree schemes can be used: on the VAX-751 it was not possible to compute the Cholesky factor using any of the other orderings.

Ordering the scale factor(s) manually as last reduces ordering time considerably, and sometimes also improves the results. If one uses reverse Cuthill-McKee or Gibbs-King one is obliged to do this since otherwise very disappointing results can be expected.

Especially for large networks it is recommended to combine the x- and y-coordinate into super nodes beforehand.

Results for the photogrammetric networks

For an adjustment with independent models (see table 7) the linked lists minimum degree algorithm is the best choice. Both quotient minimum degree and Gibbs-King are surprisingly slow for this type of matrix. Somewhat surprising are the results of Snay's banker's followed by a nonzero factorization: it gives far better results than the envelope factorization.

Again it is recommended to combine indistinguishable nodes into super nodes before the ordering starts (LMD 2).

For a bundle-block adjustment (see table 8-10) the best ordering method depends on the size and the structure of the network, and the number of iterations needed to get convergence, i.e. the number of times one need to compute the Cholesky factor. In table 11 one can find which ordering gives the lowest total cpu time for one iteration as well for three iterations.

So depending on the size and structure of the block and the number of iterations, one should use reverse

name	$\|Nz(L)\|$	order	fact	spinv	total
NAT	39799	0.00	65.44	152.25	217.69
QMD	5753	13.73	2.22	4.37	20.92
QMD 1	5557	4.69	2.08	4.08	11.45
LMD	5639	2.71	2.12	4.16	9.63
LMD 1	5565	1.89	2.02	4.15	8.67
LMD 3	5574	0.96	2.07	4.21	7.84
LMDF 3	5382	6.04	1.90	4.06	12.64
ND	7425	1.39	3.24	6.44	11.59
ND 1	7612	1.20	3.36	6.41	11.59
name	$\|Env(L)\|$	order	fact	spinv	total
NAT	78203	0.00	127.65	330.31	457.96
RCM	91549	0.80	-	-	-
RCM 1	11663	0.43	3.16	6.29	10.47
KING	12594	0.70	3.74	7.66	12.67
KING 1	12581	0.60	3.64	7.49	12.27
SNAY	15935	1.13	5.36	11.81	18.81
SNAY 1	11389	0.68	3.02	6.14	10.43
LEVY	15028	1.19	5.04	10.77	17.58
LEVY 1	11557	1.03	3.12	6.27	11.01
GK	77571	3.53	-	-	-
GK 1	10460	1.08	2.54	5.01	9.15

Table 5: ZK85, $|Nz(A\_5\_509)|$: 3341 (2.61 %).

name	$\|Nz(L)\|$	order	fact	spinv	total
NAT	6609427	0.00	-	-	-
QMD	69343	800.37	-	-	-
QMD 1	68515	84.01	40.58	245.95	379.21
LMD	68963	174.41	-	-	-
LMD 1	68831	107.82	40.39	245.85	402.56
LMD 3	68689	52.65	40.05	247.32	348.51
ND 1	126924	19.24	-	-	-
name	$\|Env(L)\|$	order	fact	spinv	total
NAT	9519599	0.00	-	-	-
RCM 1	634947	5.34	-	-	-
KING 1	390271	8.66	-	-	-
SNAY 1	333195	11.71	-	-	-
LEVY 1	593758	82.57	-	-	-
GK 1	568777	17.22	-	-	-

Table 6: OMAN, $|Nz(A\_5\_6370)|$: 35956 (.18 %).

Cuthill-McKee, Snay's banker's or Gibbs-King followed by an envelope factorization, or the natural ordering followed by a nonzero factorization.

The minimum degree algorithms perform very badly on larger blocks. Generally the number of nonzeros in the Cholesky factor exceeds the number of nonzeros in it when natural ordered, which indicates the quality of this a priori ordering.

As one can see in table 8, the minimum deficiency algorithm may cause a larger amount of fill-in than the minimum degree algorithm.

name	$\|Nz(L)\|$	order	fact	spinv	total
NAT	39680	0.00	67.87	124.69	192.56
QMD	23461	83.91	17.84	29.74	134.14
LMD	23265	8.76	17.72	29.33	58.52
LMD 2	23069	0.55	17.38	27.58	48.10
LMDF 2	22979	5.32	17.48	27.30	52.63
ND	40608	4.34	47.23	92.74	146.83
SNAY	27881	3.71	25.71	39.94	72.04
name	$\|Env(L)\|$	order	fact	spinv	total
NAT	170629	0.00	-	-	-
RCM	123158	2.00	188.03	487.15	679.76
KING	61235	2.93	43.03	104.61	152.87
SNAY	54949	3.87	33.46	81.76	121.68
LEVY	58981	4.99	37.54	93.79	138.66
GK	86295	80.46	85.46	221.84	390.26

Table 7: INMO, $|Nz(A\_8\_881)|$: 12705 (3.32 %).

name	$\|Nz(L)\|$	order	fact	spinv	total
NAT	7770	0.00	5.85	8.41	14.26
QMD	6162	7.39	3.43	5.27	16.72
LMD	6171	1.53	3.43	5.50	11.10
LMD 2	6162	0.17	3.42	5.60	9.82
LMDF 2	6168	0.90	3.37	5.40	10.30
ND	8553	0.74	6.49	10.82	18.67
name	$\|Env(L)\|$	order	fact	spinv	total
NAT	20280	0.00	16.43	43.71	60.14
RCM	10674	0.43	4.29	10.39	15.70
KIN	8445	0.60	2.74	6.44	10.34
SNAY	7410	0.73	2.09	4.79	8.23
LEVY	7779	0.75	2.33	5.35	9.03
GK	10551	1.58	3.96	9.54	15.68

Table 8: BU 3*7, $|Nz(A\_8\_273)|$: 3702 (10.42 %).

name	$\|Nz(L)\|$	order	fact	spinv	total
NAT	26124	0.00	21.73	31.20	52.93
name	$\|Env(L)\|$	order	fact	spinv	total
NAT	165012	0.00	501.64	-	-
RCM	34509	1.42	14.19	32.71	50.28
SNAY	46059	2.93	26.60	64.64	96.19

Table 9: BU 3*21, $|Nz(A\_8\_819)|$: 11976 (3.63 %).

7 Conclusions

- Ordering of unknowns is important if one wants to reduce storage and time needed for Cholesky factorization and computation of the sparse inverse.
- The minimum degree algorithm is generally the best choice if the factorization is based on a nonzero storage scheme.
- Snay's banker's algorithm is generally the best choice if the factorization is based on an envelope storage scheme.
- Excluding nodes with high degree (instrumental parameters) from the graph, and combining indistinguishable nodes (coordinates) into super nodes (points) reduces the time needed for ordering the unknowns considerably.

name	$\lvert Nz(L)\rvert$	order	fact	spinv	total
NAT	101256	0.00	166.75	242.74	409.49
name	$\lvert Env(L)\rvert$	**order**	**fact**	**spinv**	**total**
NAT	849696	0.00	-	-	-
RCM	171261	3.53	152.95	393.43	555.58
SNAY	124827	7.30	82.24	202.49	297.72

Table 10: BU 7*21, $\lvert Nz(A\_8\_1827)\rvert$: 27924 (1.69 %).

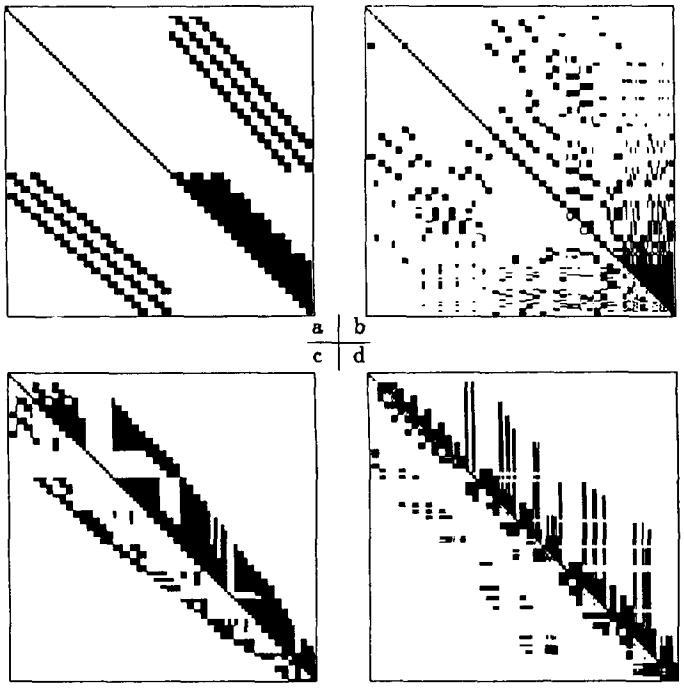

Figure 2: BU 3*7, (a) natural ordering, (b) minimum degree (LMD), (c) reverse Cuthill-McKee, (d) Snay's banker's. In the lower triangle the normal matrix, in the upper triangle the resulting Cholesky factor is depicted.

number of photos in a strip	number of strips 1	2	3	4	5	6	7
2	S	-	-	-	-	-	-
3	S	S	-	-	-	-	-
4	S	S	S	-	-	-	-
5	S	S	S	S	-	-	-
6	S	S	S	S	S	-	-
7	S	S	S	S	S	S	-
8	S	S	S	S	S	S	S
9	S	R	S	S	S	S	S
10	S	R	S	S	S	S	S
11	S	R	S	S	S	S	S
12	S	R	S	S	S	S	S
13	S	R	N (G)	S	S	S	S
14	S	R	N (G)	S	S	S	S
15	S	R	N (G)	S	S	S	S
16	S	R	N (G)	N (S)	S	S	S
17	S	R	R (G)	N (G)	S	S	S
18	S	R	R (G)	N (G)	S	S	S
19	S	R	R (G)	N (G)	N (S)	S	S
20	S	R	R (G)	N (G)	N (S)	S	S
21	S	R	R (G)	N (G)	N (G)	S	S
22	S	R	R (G)	N (G)	N (G)	S	S
23	S	R	R (G)	N (G)	N (G)	N (S)	S
24	S	R	R (G)	N (G)	N (G)	N (S)	S

Table 11: Fastest method for the bundle-block adjustment (ordering + permutation + factorization + sparse inverse). Between brackets the result if three factorizations are computed (only displayed if it differs from one iteration). N=natural ordering, S=Snay's banker's, R=reverse Cuthill-McKee, G=Gibbs-King.

References

Cuthill, E., McKee, J. (1969) Reducing the bandwidth of sparse symmetric matrices. In *Proceedings 24th National Conference ACM.* ACM publication no. P-69, Brandon Systems Press, NJ.

Dongarra, J.J., Grosse, E. (1987) Distribution of mathematical software via electronic mail. In *Comm. ACM* 30, pp. 403-407.

George, A., Liu, J.W.H. (1979) An implementation of a pseudo-peripheral node finder. In *ACM Trans. on Math. Softw.*, Vol. 5, pp. 286-295.

George, A., Liu, J.W.H. (1980) A fast implementation of the minimum degree algorithm using quotient graphs. In *ACM Trans. on Math. Softw.*, Vol. 6, No. 3, pp. 337-358.

George, A., Liu, J.W. (1981) *Computer solution of large sparse positive definite systems*, Prentice-Hall, Englewood Cliffs, NJ.

George, A., Liu, J.W. (1989) The evolution of the minimum degree ordering algorithm. In *SIAM Review*, Vol. 31, No. 1, pp. 1-19.

Gibbs, N.E. (1976) A hybrid profile reduction algorithm. In *ACM Trans. on Math. Softw.*, Vol. 2, No. 4, pp. 378-387.

Gibbs, N.E., Poole, W.G., Stockmeyer, P.K. (1976) An algorithm for reducing the bandwidth and profile of a sparse matrix. In *SIAM J. Numer. Anal.*, Vol. 13, No. 2, pp. 236-250.

Jonge de, P.J. (1987) On the ordering of the unknowns during the Hipparcos reduction on circles. Graduate thesis Department of Geodesy, Delft University of Technology.

Jonge de, P.J. (1991) An analysis of ordering schemes for the unknowns during the solving of geodetic least squares systems, Reports of the Faculty of Geodetic Engineering/Dept. of Mathematical and Physical Geodesy, Delft University of Technology, 91.4.

King, I.P. (1970) An automatic reordering scheme for simultaneous equations derived from network systems. In *Int. J. Numer. Meth. Eng.*, Vol. 2, pp. 497-509.

Levy, R. (1971) Restructuring of the structural stiffness matrix to improve computational efficiency. In *Jet Propulsion Laboratory Technical Review.* 1, pp. 61-70.

Marel, van der, H. (1988) On the *"great circle reduction" in the data analysis for the astrometric satellite Hipparcos.* Netherlands Geodetic Commission, Publications on Geodesy New Series, Vol. 8(2).

Parter, S. (1961) The use of linear graphs in Gauss elimination. In *SIAM Review*, Vol. 3, No. 2, pp. 119-130.

Pissanetzky, S. (1984) *Sparse matrix technology.* Academic Press, London.

Snay, R.A. (1976) Reducing the profile of sparse symmetric matrices. In *Bulletin Geodesique*, Vol. 50, pp. 341-352.