

## THE SOLUTION OF LARGE GEODETIC SYSTEMS BY SPARSE MATRIX TECHNIQUES

**Richard L. Branham, Jr.**  
**IANIGLA-CCT, Mendoza**  
[rbranham@lab.cricyt.edu.ar](mailto:rbranham@lab.cricyt.edu.ar)

**Abstract:** Geodetic problems frequently lead to large matrices, tens of thousands of equations of condition and thousands of unknowns. These matrices are generally sparse and possess a characteristic structure: the unknowns, such as the station coordinates, are grouped together in a sparse row in the matrix of the equations of condition,  $A$ . When the equations of condition are formed into normal equations,  $A^T \cdot A$ , these equations remain sparse, although more dense than the equations of condition. A Cholesky decomposition of the normal equations produces a triangular matrix, the Cholesky factor  $S$ , that in general will be dense unless the rows and columns of the normal equations are permuted in such a way as to preserve sparsity. This remains true even should one eschew normal equations and apply orthogonal transformations directly to  $A$  to produce a triangular matrix  $R$  identical, within round-off or truncation error, to  $S$ . The sparsity structure of  $R$  depends on the structure of  $A^T \cdot A$ . In 1880 the German geodesist Helmert developed a permutation method, called nested dissection, that permits one to calculate a sparse Cholesky factor. A test problem with 30,000 equations of condition and 2,000 unknowns using CSparse on a computer with 3 GB of RAM and 2 GHz speed showed that nested dissection runs three times faster than not using sparse matrix techniques. Should the matrix  $A$  be too large to fit into memory,  $A^T \cdot A$  must be formed by sequential accumulation of the rows of  $A$  or by application of orthogonal transformations to the rows to form  $R$ . This becomes time consuming and the principal source of CPU usage. Even here, however, sparse matrix techniques work more efficiently.

**Resumen:** Problemas geodésicos resultan a menudo en matrices grandes, decenas de miles de ecuaciones de condición y miles de incógnitas. Estas matrices generalmente son ralas y poseen una estructura característica: las incógnitas, como las coordenadas de la estación, se agrupan en una fila rala de la matriz de las ecuaciones de condición,  $A$ . Cuando las ecuaciones de condición se acumulan para formar las ecuaciones normales,  $A^T \cdot A$ , las mismas quedan ralas, aunque más densas que las ecuaciones de condición. Una descomposición de Cholesky de las ecuaciones normales calcula una matriz triangular  $S$ , el factor de Cholesky, que en general será denso si no permutamos las filas y columnas de las ecuaciones normales para conservar ralicidad. Esto no cambia si usamos transformaciones ortogonales para reducir la matriz  $A$  a una matriz triangular  $R$ , idéntica dentro de los errores de redondeo o truncamiento al factor  $S$ . La estructura rala de  $R$  depende de la estructura de  $A^T \cdot A$ . En 1880 el geodesta alemán Helmert desarrolló un método de permutación, llamado disección encadenada, que permite calcular un factor ralo de Cholesky. En una prueba usando CSparse con un sistema de 30.000 ecuaciones y 2.000 incógnitas y una computadora con 3 GB de RAM y 2 GHz de velocidad, disección encadenada era tres veces más veloz que técnicas que no usan matrices ralas. Si la matriz  $A$  no cabe en memoria hay que formar  $A^T \cdot A$  por la acumulación secuencial de las filas de  $A$  o por aplicar transformaciones ortogonales a las filas para formar  $R$ . Esto gasta mucho tiempo y representa el principal consumo de ciclos de CPU. Sin embargo, técnicas para matrices ralas quedan más eficientes.

## 1. Introduction

The solution of many problems in science and engineering leads to a lineal system with more equations than unknowns, a superdetermined system:

$$A \cdot x = d. \quad 1)$$

In Eq. (1) the matrix  $A$  is of size  $m \times n$ , where  $m$  is the number of equations and  $n$  the number of unknowns, with  $m \geq n$ . If we wish to solve the system by means of least squares, the solution is given formally by:

$$A^T \cdot A \cdot x = A^T \cdot d, \quad 2)$$

where  $A^T \cdot A$ , called the matrix of the normal equations, is of size  $n \times n$ , symmetric, and positive-definite. Because of these properties, the normal equation can be solved by Cholesky decomposition:

$$\begin{aligned} A^T \cdot A \cdot x &= S^T \cdot S \cdot x = A^T \cdot d; \\ y &= S \cdot x; \\ S^T \cdot y &= A^T \cdot d, \end{aligned} \quad 3)$$

where  $S$  is upper triangular. As an alternative to Cholesky decomposition one can apply orthogonal transformations, such as Givens or Householder, directly to  $A$  to reduce it to upper triangular:

$$Q \cdot A \cdot x = R \cdot x = Q \cdot d, \quad 4)$$

where  $Q$  is orthogonal and  $R$  upper triangular.  $R$  and  $S$  are identical within the limits of round-off or truncation error.

## 2. Solution of sparse systems

Frequently the matrix  $A$  is sparse, a matrix with many zero elements. Unfortunately, even though  $A$  may be sparse,  $A^T \cdot A$  is not necessarily sparse if the null elements are found at random locations within  $A$ . If  $A$  is  $100d$  per cent dense ( $0 \leq d \leq 1$ ) a simple probability argument shows that  $A^T \cdot A$  fills to a density of  $mn(1-d^2)^m$ . Unless  $d \approx 0$  the fill-in can approach 100% even for small  $d$ . Because the Cholesky factor  $S$  is unique and identical to  $R$ , this argument remains germane even if one does not *explicitly* form the normal equations but rather uses orthogonal transformations. As an example a  $200 \times 200$  matrix that is 0.2% dense with non zero elements at random locations fills to 11.8% for the normal equations and 85.3% for the Cholesky factor.

Fortunately, with geodetic data the unknowns tend to occur in groups within a sparse row of  $A$ , for example the station coordinates, with possibly a few unknowns common to all of the rows, such as a star or satellite observed simultaneously from all of the stations. Thus, the matrix  $A$  contains some columns close to 100% dense and other, sparse columns with a far from random sparsity structure. With these conditions  $A^T \cdot A$  will still be sparse, although more dense than  $A$ . For example, a  $150 \times 150$  matrix that is 4.8% dense results in a matrix of normal equations that is 9.5% dense. The Cholesky factor, however, will not necessarily be sparse unless care is taken to preserve sparsity.

Sparsity will be conserved if dense rows are permuted downwards and dense columns permuted to the right. For general matrices, not necessarily symmetric and positive definite, Tewarson (1973, Sec. 2.5) presents an algorithm that performs a one-time permutation of the columns before solution begins and then permutes dense rows downward as the solution progresses.

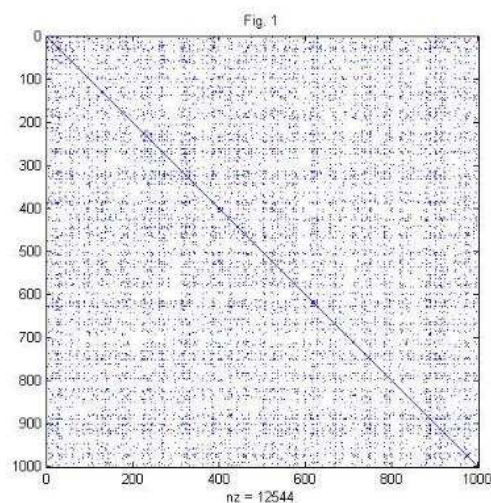
### 3. Nested dissection

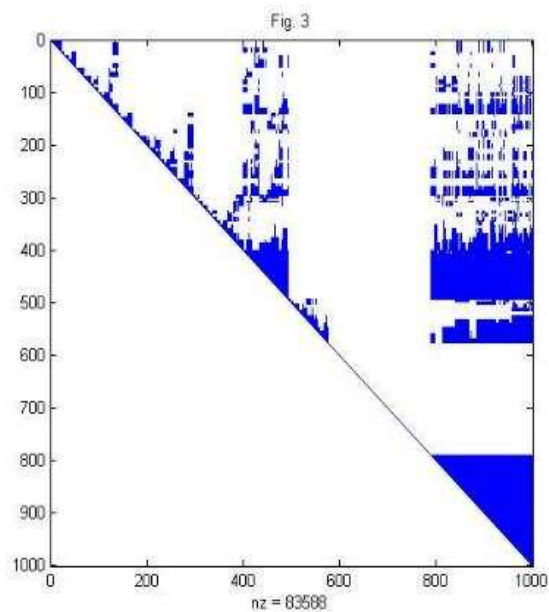
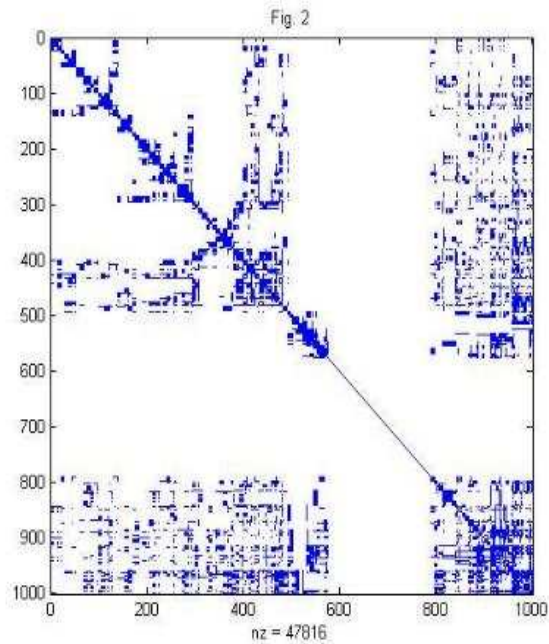
Given the structure of the matrix for the geodetic problem, sparse rows with dense groupings of unknowns, something more sophisticated can be used, nested dissection, which Helmert (1880) developed in 1880. For a discussion of the theory of nested dissection see George and Liu (1981, Ch. 8) and Davis (2006, Sec. 7.6). Davis's book includes routines in C, collectively called CSparse, that can be called from Matlab. Nested dissection finds a group of matrix elements, called a separator, that divides the remaining elements into two blocks of roughly the same size. The separator columns are permuted to the right. Then dissection is applied recursively to the remaining blocks.

Fig. 1 shows a 1.25% dense matrix with elements in far from random positions. Fig. 2 shows the matrix multiplied by its transpose with its rows and columns permuted by the nested dissection ordering; the matrix is 4.78% dense. Finally, Fig. 3 shows the Cholesky factor for the matrix of Fig. 2, 16.70 % dense. Although far more dense than the original matrix, the Cholesky factor can still be considered sparse. Code in C for nested dissection is remarkably compact. Here is Davis's routine from CSparse:

```
function p = cs_nd (A)
n = size (A,1) ;
if (n == 1)
    p = 1 ;
elseif (n < 500)
    p = cs_amd (A) ;           % use cs_amd on small graphs
else
    [s a b] = cs_nsep (A) ;   % find a node separator
    a = a (cs_nd (A (a,a))) ; % order A(a,a)                b = b
    (cs_nd (A (b,b))) ;      % order A(b,b)
    p = [a b s] ;           % obtain final ordering
end
```

There is a check for the trivial situation of a 1x1 matrix. Unless the matrix is of at least size 500x500, *cs\_nd* calls a routine, *cs\_amd*, that implements another sparse matrix routine called "minimum degree." *cs\_nsep* finds a separator, and then *cs\_nd* is applied recursively to the two partitions of the matrix.





## Tests

I applied nested dissection to two test problems satisfying the conditions of the geodetic problem, one of size 30,000x2,000 and the other of size 120,000x6,000. Both matrices were approximately 0.2 % dense. The computer used has 2 GB of RAM and 3.4 GHz of speed. For the first test problem the matrix  $A^T \cdot A$  could be formed in memory. Time to form and solve the normal equations was 0.69 sec. without use of nested dissection and 0.22 sec. with nested dissection, three times faster. If sparse matrix techniques are not used and the matrix  $A$  treated as full, even though it is sparse, time to form and solve the normal

equations jumps to 69.62 sec.

The second test problem could not be solved at all, for lack of memory, without use of sparse matrix techniques. With sparse techniques for the normal equation the problem could be solved in 4205 sec. This much greater use of CPU time is caused by having to read the rows of  $A$  in sequence to form the normal equations because  $A^T \cdot A$  cannot be formed in memory.

Because the inverse of a sparse matrix is in general not sparse, to calculate a covariance matrix needed for the mean errors and correlations would generate catastrophic fill-in of the inverse of the normal equations. To avoid this the covariance matrix can be calculated column by column:

$$\begin{aligned} e_i &= (0 \quad \dots \quad 1 \quad \dots \quad 0); \\ y &= S \cdot c_i; \\ S^T \cdot y &= e_i. \end{aligned} \tag{5}$$

The 1 in the vector  $e_i$  occurs in location  $i$ . The  $c_i$  are the columns of the covariance matrix.

## Conclusions

When the requirements of the problem permit the matrix of Eq. (1) to be represented in sparse format, the solution can be calculated much more efficiently, sometimes by a factor of more than three hundred to one in CPU usage. In extreme cases sparse matrix techniques permit a solution that would exhaust memory with non-sparse techniques. The moral seems clear: if the matrix is sparse, use methods that take advantage of the sparsity.

## References

- Davis, T.A. 2006, *Direct Methods for Sparse Linear Systems*, Philadelphia: SIAM  
George, A., & Liu, J.W. 1981, *Computer Solution of Large Sparse Positive Definite Systems*, Englewood Cliffs, New Jersey: Prentice-Hall  
Helmert, F.R. 1880, *Die Mathematischen und Physikalischen Theorien der höheren Geodäsie, 1 Teil*, Leipzig: Teubner  
Tewarson, R.P. 1973, *Sparse Matrices*, New York: Academic