Sparse numerical linear algebra: direct methods and preconditioning[1]

Iain S. Duff[2]

ABSTRACT

Most of the current techniques for the direct solution of linear equations are based on supernodal or multifrontal approaches. An important feature of these methods is that arithmetic is performed on dense submatrices and Level 2 and Level 3 BLAS (matrix-vector and matrix-matrix kernels) can be used. Both sparse LU and QR factorizations can be implemented within this framework.

Partitioning and ordering techniques have seen major activity in recent years. We discuss bisection and multisection techniques, extensions to orderings to block triangular form, and recent improvements and modifications to standard orderings such as minimum degree.

We also study advances in the solution of indefinite systems and sparse least-squares problems.

The desire to exploit parallelism has been responsible for many of the developments in direct methods for sparse matrices over the last ten years. We examine this aspect in some detail, illustrating how current techniques have been developed or extended to accommodate parallel computation.

Preconditioning can be viewed as a way of extending direct methods or of accelerating iterative ones. We will view it in the former way in this talk and leave the other perspective to the talk on iterative methods.

Finally, we will briefly comment on recent attempts to develop tools and platforms towards a sparse problem solving environment.

Keywords: sparse matrices, direct methods, preconditioning.

AMS(MOS) subject classifications: 65F05, 65F50.

---

[1] This is a preprint of a paper based on the talk given at the State of the Art in Numerical Analysis Meeting in York in April 1996. Also available as Report TR-PA-96-22, CERFACS, 42 Ave G Coriolis, 31057 Toulouse Cedex, France.

[2] I.Duff@rl.ac.uk. Current reports available by anonymous ftp from seamus.cc.rl.ac.uk (internet 130.246.8.32) in the directory "pub/reports". This report is in file duRAL96047.ps.gz.

Department for Computation and Information
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

October 2, 1997.

Contents

1 Introduction

In common with many of my co-authors in this volume, my starting point was to read the review on this subject given at the last State of the Art meeting in Birmingham. The article "Sparse Matrices" was authored by John Reid, and it is perhaps significant that it covered both iterative and direct methods, the subject of two talks at this meeting. Indeed, although I have been working in this field for about twenty five years, I find it amazing how many advances have occurred in the last ten, necessitating the dual lectures of this meeting. Although this paper is primarily on direct methods, I also include a section on preconditioning techniques for use with iterative methods. The viewpoint of this discussion will be quite different from that presented in the chapter on Iterative Methods since I will look at the construction of preconditioners as being an alternative to direct methods when, for one reason or another (usually storage), direct methods are infeasible. Moreover, in common with proponents of iterative methods, I feel strongly that the only way of solving really challenging linear algebra problems is by combining direct and iterative methods through either conventional or novel preconditioning.

Although I intend this paper to be a general State of the Art survey, I wish to emphasize what I consider to be the most important developments of the last ten years. Accordingly, I have structured the text to highlight: implementations that use higher level BLAS kernels, advances in ordering strategies particularly for symmetric problems, the exploitation of parallelism, and the use of direct techniques as preconditioners for iterative methods.

Undoubtedly, the kernel that gets closest to peak performance on modern computers is a **dense** matrix-matrix multiply. A standard version of this kernel is provided by subroutine _GEMM in the Level 3 Basic Linear Algebra Subprograms or BLAS (Dongarra, Du Croz, Duff and Hammarling 1990) and is supported by many vendors of high performance computers. Most sparse direct codes use this and allied kernels to achieve high performance, and we discuss how they are able to do this in Section 2.

Ten years ago, one might have been forgiven for thinking that the problem of ordering a symmetric system for subsequent Gaussian elimination was resolved in favour of minimum degree, one of the earliest proposed orderings. However, quite recently there has been significant work on developing other classes of orderings, related to dissection methods, which are showing great promise of overhauling this perceived wisdom.

Additionally, there has been further understanding of the minimum degree ordering and attempts to make it more efficient and to adapt it for orderings more suited to parallel computation. We discuss orderings for symmetric systems in Section 3.

In 1986, there was little software for unsymmetric sparse problems. Early examples included NSPIV by Sherman (1978), Y12M by Zlatev, Waśniewski and Schaumburg (1981), MA28, MA32, and MA37 from the Harwell Subroutine Library (HSL) and a surrogate MA28 in the NAG Library (F01BRF/F01BSF/F04AXF). The development and production of sparse unsymmetric solvers has now become a veritable growth industry. The multifrontal and supernodal techniques, so powerful in the symmetric case, have been extended to unsymmetric systems. In other approaches, various decompositional and preordering techniques are used to facilitate later factorization, particularly on parallel computers. We consider these issues in Section 4. We discuss the case of sparse symmetric indefinite systems in Section 5 and the solution of sparse least-squares problems in Section 6.

A major impetus for the development of algorithms and software in all linear algebra has been the availability of parallel architectures of various shapes and forms. Although the ICL DAP was originally produced in the 1970s, it was a bit-oriented SIMD architecture, and really the first commercially available parallel machines were the Denelcor HEP in 1984 and the Alliant FX/8 in 1985. Thus any practical implementations of techniques for exploiting parallelism belong strongly to the last decade. Although we have considered issues related to parallelism in previous sections, we study this in more depth in Section 7.

In Section 8, we discuss the use of direct methods to obtain preconditioners for iterative methods. As we mentioned earlier, it is this route that we believe to be the most important for the solution of really large problems from, say, three dimensional partial differential equations.

We briefly review tools for sparse matrix manipulation and computation in Section 9. These include consideration of some tentative first steps towards a problem solving environment for sparse problems.

It is not the intention of this talk or this meeting to perform any crystal-ball gazing, but we indulge in a little of this in our concluding section.

Before continuing with the main paper, it is useful to define a few terms that will be used extensively in the following sections. The solution of a sparse system is usually divided into several phases:

1. Analysis of the sparsity structure to determine a pivot ordering.

2. Symbolic factorization to generate a structure for the factors.

3. Numerical factorization.

4. Solution of set(s) of equations.

In some cases, particularly when it is important to consider numerical values when choosing pivots, the first three phases are combined into an analyse-factorize phase. Additionally, there may be some partitioning scheme prior to all these phases, which are then executed on submatrices from the partition. The relative speeds of the four phases are very dependent on the details of the algorithm and implementation, the problem being solved, and the machine being used. However, one reason for separating the first two phases from the third is that it is usually much faster to perform an analysis and symbolic factorization without reference to numerical values. This does mean that there must be some way of incorporating subsequent numerical pivoting in the numerical factorization if general problems are to be solved.

Most of the algorithms we consider are based on matrix factorization methods like Gaussian elimination, and part-way through the matrix factorization, when we have calculated some of the factors, the remaining matrix (which need not be held explicitly) is composed of original matrix entries and fill-in from the earlier stages. We refer to this matrix as the *reduced* matrix.

Finally, the relationship between graphs and sparse matrices is ubiquitous in sparse matrix work. We do not make heavy use of graphs in the ensuing discussion, but it is important to define the main graphs associated with sparse matrices. With a symmetric sparse matrix of order n, we associate a graph with n vertices, and an edge (i, j) between vertices i and j, if and only if entry $a_{ij} \neq 0$. A clique is a complete subgraph, that is all vertices of the subgraph are pairwise connected by edges in the graph. For an unsymmetric matrix, we associate a directed graph where the edges are now directed. A bipartite graph is sometimes associated with an unsymmetric (even non-square) matrix. This has two sets of disjoint vertices identified with rows and columns of the matrix respectively. An edge (i, j) exists from the row vertices to the column vertices if and only if entry $a_{ij} \neq 0$. The elimination tree is defined by the Cholesky factors of a symmetric matrix and has an edge (i, j) if the first nonzero entry below

the diagonal in column j of the lower triangular Cholesky factor is in row i. Elimination trees were used by Duff (1981) and Schreiber (1982) and are surveyed in depth by Liu (1990).

As a postscript to this section, I should stress that, although error analysis is not discussed, I am concerned throughout that the solution methods will give good answers, often in the sense of providing the solution to a nearby problem. It is worth mentioning that this notion of backward error can be developed to include only perturbations that preserve the sparsity structure (Arioli, Demmel and Duff 1989).

Finally, lest I tread on too many toes, let me say that I have not been entirely consistent in quoting the original source of an idea but sometimes favour an overview or more recent report that itself includes further references and original attributions. I did consult widely to avoid the worst abuses of this (see Acknowledgement Section). Of course, in such a rapidly developing field, many ideas are simultaneously "discovered". Although you might think that there are too many references in this paper, I can assure you that I have been quite selective!

2 High performance sparse factorization

It is a truism that really large problems need large computers to solve them, and it is equally true that at the heart of most large-scale computations lies the solution of a large sparse set of linear equations. Currently, high performance computers come in various guises although nearly all are either vector processors or RISC processors, sometimes as many as a few hundred of them but more commonly only about two or four! In this section, we concentrate more on the uniprocessor performance and leave the discussion of parallelism until Section 7.

On some of these machines, the performance of a general code can be much less than the peak performance, usually because of delays in getting data to the arithmetic processors. The kernel that gets closest to the peak is the matrix-matrix multiply routine, _GEMM in the Level 3 Basic Linear Algebra Subprograms (Dongarra et al. 1990), which on most machines will achieve over 90% of peak on matrices of order only a few hundred. Recently Kågström, Ling and Van Loan (1993) and Daydé, Duff and Petitet (1994) have expressed all the Level 3 BLAS kernels as calls to _GEMM (and trivial calls to _TRSM) and so all can execute at high speed. Many years ago, before the Level 3 BLAS were established, Duff (1981) extolled the virtues of using

dense matrix kernels within sparse direct codes and gave some examples of techniques that used such kernels. It is now true to say that nearly all sparse direct codes use dense matrix kernels in their inner loops and, by doing so, achieve high performance over a wide range of architectures and problems. Indeed, it is on machines with caches or a hierarchical memory structure that the Level 3 BLAS realize their full potential. Rothberg and Gupta (1991) and Ng and Peyton (1993*a*) illustrate the importance of using Level 3 BLAS in sparse factorizations for efficiency on cache-based machines.

An easy way to use such kernels is to recognize that, as the elimination progresses, the reduced matrix in sparse Gaussian elimination becomes denser and at some point it is more efficient to switch to dense code. The point at which to do this depends on the matrix structure and machine, but recent experience indicates that it can be beneficial to switch when the density is as low as 10%. Switching to dense code was discussed by Dembart and Erisman (1973) and was incorporated in an experimental version of the Harwell Subroutine Library (HSL 1996) code MA28 (Duff 1977). However, a switch to dense matrix processing was only included in the analyse-factorize phase. A switch to dense matrix processing in the other phases was introduced in the HSL code MA48 (Duff and Reid 1993, Duff and Reid 1996*a*). However, although such a switch is clearly beneficial, it only addresses the computations towards the end of the factorization. There are two main approaches that can reap the benefit of higher level BLAS working throughout the entire factorization. These are the frontal methods and the supernodal approach.

Since the basic frontal method and many of its advantages have been known for some time and are described in John Reid's 1986 State of the Art paper, we will not discuss them in detail here but rather concentrate on new developments in these methods over the last decade. The important aspect of these methods is that arithmetic is performed within a dense *frontal* matrix, and pivots can be chosen from anywhere within a submatrix of this frontal matrix. In a frontal scheme, when all possible pivots have been chosen and the elimination operations performed, further data from the problem is assembled or summed into the frontal matrix and more pivots are chosen. This scheme has the merits of fairly simple storage management but can suffer significant fill-in for general problems and has restricted possibilities for exploiting parallelism. These two concerns are addressed by the multifrontal methods (Speelpenning 1978, Duff and Reid 1983, Liu 1992), which were again discussed by Reid (1987). Three principal problems with multifrontal methods, as practised in the 1980s, were that

- there could be significant overheads for data movement,
- they assumed structural symmetry, and
- the analyse phase assumed that any diagonal entry could be chosen as pivot.

We now consider how these concerns have been addressed. Other recent work on developing parallel versions of multifrontal methods is considered in Section 7. In the first case, data movement can be reduced by continuing with one frontal matrix rather than stacking it and starting another. Taken to its extreme, this strategy would give a (uni-)frontal scheme. Davis and Duff (1995) discuss the effect of this technique and have incorporated it in the HSL code MA38. Although one of the early multifrontal codes MA37 (Duff and Reid 1984) solved sets of unsymmetric equations, the analyse phase was performed on the pattern of the matrix with entries a_{ij} if either a_{ij} or a_{ji} were an entry in the original matrix. This is clearly fine if the matrix is symmetric in structure but surprisingly can perform quite well for unsymmetric matrices, although it can be inefficient when the matrix is markedly unsymmetric. For unsymmetric systems, a preordering to place nonzeros on the diagonal is often very helpful and is an option in the HSL code MA41, which is designed for shared memory computers and makes much more use of higher level BLAS than its precursor MA37. Davis and Duff (1993) have extended the multifrontal scheme to general unsymmetric matrices, using rectangular fronts and directed acyclic graphs (Gilbert and Liu 1993). This extension works well on most highly unsymmetric systems but can be poorer than MA41 when the matrix is not highly unsymmetric. We discuss both approaches further in Section 4. The third problem is more difficult and is present for any technique that performs an analyse phase separate from the numerical factorization. Clearly, one way to resolve this is to combine the analyse and factorize phases but then many of the benefits of the fast analyse phase are lost. A certain amount of numerical or additional structural information can be supplied to the analyse phase, and we consider an example of this when we study the use of structured 2×2 pivots in Section 5.

The other main approach to using higher level BLAS in sparse direct solvers is a generalization of a sparse column factorization. These can either be left-looking (or fan-in) algorithms, where updates are performed on each column in turn by all the previous columns that contribute to it, then the pivot is chosen in that column and the multipliers calculated; or a

right-looking (or fan-out) algorithm where, as soon as the pivot is selected and multipliers calculated, that column is immediately used to update all future columns that it modifies. Higher level BLAS can be used if columns with a common sparsity pattern are considered together as a single block or supernode and algorithms are termed column-supernode, supernode-column, and supernode-supernode depending on whether target, source, or both are supernodes.

Several authors have experimented with these different algorithms (right-looking, left-looking, and multifrontal) and different blockings. Ng and Peyton (1993*a*) favour the left-looking approach and Amestoy and Duff (1989) show the benefits of Level 3 BLAS within a multifrontal code on vector processors. Rothberg and Gupta (1991) find that on cache-based machines it is the blocking that affects the efficiency (by a factor of 2 to 3) and the algorithm that is used has a much less significant effect. Demmel, Eisenstat, Gilbert, Li and Liu (1995) have extended the supernodal concept to unsymmetric systems although, for irregular problems, they cannot use regular supernodes for the target columns and so they resort to Level 2.5 BLAS. By doing this, the source supernode can be held in cache and applied to the target columns or blocks of columns of the "irregular" supernode, thus getting a high degree of reuse of data and a performance similar to the Level 3 BLAS.

The multifrontal and supernodal methods form the basis for some of the approaches that exploit parallelism, and we consider this aspect further in Section 7.

Although we have stressed the importance of using higher level BLAS to obtain high performance, we should be aware that, if the matrix is very sparse and the factors are also, there will normally be no benefit in using BLAS kernels. It is, however, quite likely that there will be much parallelism in both factorization and solution. The extreme case of this is a diagonal matrix or a permutation thereof.

3 Orderings for symmetric problems

Although predated by some ten years by the paper of Markowitz (1957) on unsymmetric orderings, scheme S2 in the paper by Tinney and Walker (1967) established the main ordering for symmetric problems that has remained almost unchallenged until this present day. Scheme S2 is commonly termed the minimum degree ordering because, at each stage, the pivot chosen

corresponds to a node of minimum degree in the undirected graph associated with the reduced matrix. In matrix terms, this corresponds to choosing the entry from the diagonal that has the least number of entries in its row within the reduced matrix. This ordering algorithm has proved remarkably resistant to competitors and, although based only on a local criterion, does an excellent job of keeping subsequent work and fill-in low over a wide range of problems. The evolution of the minimum degree ordering is studied by George and Liu (1989).

George (1973) proposed a different class of orderings based on a non-local strategy of dissection. In his *nested dissection* approach, a set of nodes is selected to partition the graph, and this set is placed at the end of the pivotal sequence. The subgraphs corresponding to the partitions are themselves similarly partitioned and this process is nested with pivots being identified in reverse order. Minimum degree, nested dissection and several other symmetric orderings were included in the SPARSPAK package (George and Liu 1979, George, Liu and Ng 1980). Many experiments were performed using the orderings in SPARSPAK and elsewhere, and the empirical experience at the beginning of the 90s indicated that minimum degree was the best ordering method for general or unstructured problems.

A major problem with the minimum degree ordering is that it is not susceptible to analysis, in the sense of computing the complexity of the resulting factorization on a regular grid problem. Part of the problem in analysing minimum degree is caused by tie-breaking; that is, choosing which node of minimum degree to use as pivot when, as is usually the case, there are many to choose from. Tie-breaking strategies have become a study in their own right (for example, Cavers 1989) and significantly complicate the algorithm while still not guaranteeing a better ordering nor allowing a theoretical analysis. Additionally, various studies have shown that tie-breaking can have a profound effect on the amount of fill-in (Duff, Erisman and Reid 1976, Berman and Schnitger 1990). In contrast, George (1973) showed that the nested dissection ordering algorithm could be analysed for regular grid problems and, furthermore, Hoffman, Martin and Rose (1973) proved that the amount of fill-in and work for such an ordering was of the lowest order (in terms of the grid size) that could be obtained by a direct method. This led to the hope that a dissection ordering could be found that was both theoretically and practically superior to minimum degree. Many attempts were made to do this but, although the analysis was extended to planar graphs (Lipton, Rose and Tarjan 1979), it was difficult to do for general matrices and practical implementations were superior to minimum

degree for only fairly restricted classes of problems, usually arising from regular grids. Analysis has shown that nested dissection is close to optimal on graphs of bounded degree, although the proofs are not constructive (Gilbert 1988, Bodlaender, Gilbert, Hafsteinsson and Kloks 1995).

We now consider recent advances in ordering strategies: first to the minimum degree orderings and then to methods based on dissection.

Although the minimum degree ordering is simple enough to describe, it is not quite so simple to implement efficiently. There are three main issues:

- selection of pivot,
- update of reduced matrix after selection of pivot, and
- update of degree counts.

For the first, it is easy to keep a list of nodes in order of increasing degree and to choose the node at the head of that list each time. There are two ways in which this task can be made significantly more efficient. The first is to observe that, once a node is selected, all nodes that were in a complete subgraph (or clique) containing that node, have degree one less and so can immediately be eliminated without any extra fill-in, and subsequently all nodes in the clique can be eliminated. This is usually termed *mass node elimination* and was included in some early minimum degree codes. Since there is no fill-in within the clique, a better measure of the "damage" done to the matrix by a potential pivot can be obtained not from its degree but rather from its *external degree*, which corresponds to the number of edges to nodes outside its clique. It is quite natural, in a finite-element application, to perform the minimum degree ordering on a graph where nodes in the same clique are treated as a single node, and this was done by the minimum degree ordering algorithm in the HSL code MA47 (Duff and Reid 1995). Ashcraft (1995) has performed an exhaustive study of this and obtains speed-ups of over 5 for some standard test matrices. The second improvement to pivot selection stems from the observation that, if two nodes of the same degree are not adjacent in the graph, they can be eliminated simultaneously. This clearly has implications for parallelism and for the degree and graph update phases and is termed *multiple elimination* (Liu 1985). We will revisit this multiple selection of pivots in Section 7 when we consider parallel computing.

The resolution of the second issue, graph update, was the main reason why minimum degree codes improved by several orders of magnitude over the decade 1976-1986. The principal saving was made by using the clique

structure of the reduced matrix and updating this rather than individual edges of the graph. This was discussed in the review by Reid (1987).

We thus come to the final issue, that of updating the degree counts. There are two main approaches here. One is to have a threshold and compute the new degrees only if they could fall below this threshold. The threshold must, of course, be changed dynamically, at which point some recalculation of degrees is necessary. The second is to replace the minimum degree count by an approximate degree count that is easier to compute. There have been several attempts at this, for example Gilbert, Moler and Schreiber (1992), but most give worse orderings than full minimum degree. Recently, however, Amestoy, Davis and Duff (1995) have designed an approximate minimum degree ordering (AMD) where the bound is equal to the degree in many cases. They have found that their AMD ordering is almost indistinguishable from the minimum degree ordering in quality but is very much faster to compute. An interesting twist to this is given by the work of Rothberg (1996*b*) who shows surprising promise with a similarly conceived implementation of an approximate minimum fill-in algorithm.

One problem with the minimum degree ordering is that it tends to give elimination trees that are not well balanced and so not ideal for using as a computational graph for driving a parallel algorithm. Liu (1989) has developed a technique for massaging the elimination tree so that it is more suitable for parallel computation but the effect of this is fairly limited for general matrices. Duff, Gould, Lescrenier and Reid (1990) propose modifications to the minimum degree criterion to directly enhance parallelism but have only compared their algorithms using fairly crude models of parallelism. The use of dissection techniques would appear to offer the promise of much better balanced trees, although the inferior performance of the early dissection codes needs to be addressed for them to be viable. We now discuss recent advances in dissection techniques.

It is only within the last year or so that the supremacy of minimum degree has been challenged (by dissection orderings, as may have been expected). The beauty of dissection orderings is that they take a global view of the problem; their difficulty until recently has been the problem of extending them to unstructured problems. Recently, there have been several tools and approaches that make this extension more realistic. The essence of a dissection technique is a bisection algorithm that divides the graph of the matrix into two partitions. If node separators are used, a third set will correspond to the node separators. Such a bisection is then repeated in a nested fashion to obtain an ordering for the matrix. Perhaps

the bisection technique that has achieved the most fame has been spectral bisection. In this approach, use is made of the Laplacian matrix that is defined as a symmetric matrix whose diagonal entries are the degrees of the nodes and whose off-diagonals are -1 if and only if the corresponding entry in the matrix is nonzero. This matrix is singular because its row sums are all zero, but if the matrix is irreducible, it is positive semidefinite with only one zero eigenvalue. We can use this matrix to define a bisection by constructing a vector x that has components x_i equal to $+$ or -1, according to which partition node i lies in. Then the quantity $x^T Ax$ is 4 times the number of edges between the two halves of the bisection. We can thus obtain an "optimal" bisection by minimizing $x^T Ax$ subject to $\sum_i x_i = 0$ with $x_i = \pm 1$. Since the first constraint corresponds to finding a vector orthogonal to the vector of all ones, which is the eigenvector for the zero eigenvalue, in the corresponding continuous problem it is the eigenvector corresponding to the smallest nonzero eigenvalue (called the Fiedler vector) that is of interest. Normally some variant of the Lanczos algorithm is used to compute this (Pothen, Simon and Liou 1990, Pothen, Simon, Wang and Barnard 1992, Barnard, Pothen and Simon 1995, Barnard and Simon 1993). The graph is then bisected according to the components of this eigenvector. If a balanced bisection is desired, commonly all components greater than the median are put in one partition and those lower than the median in the other.

In saying this, of course, one must establish a criterion for "optimality". There are clearly two sometimes conflicting goals: balancing the bisection and minimizing the number of edges joining each set (or if a node separator is used, minimizing the number of nodes in the separator). Rothberg (1996*a*) has experimented with various criteria and has found that minimizing the quantity $|S|/(|C| \times |D|)$, where $|S|$ is the number of nodes in the separator set, and $|C|$ and $|D|$ are the number of nodes in each partition, was the best of the criteria he examined in terms of floating-point operations for a Cholesky factorization with a nested dissection ordering. Another measure that has been used by Ashcraft and Liu (1995) is the quantity $|S| \left(1 + \alpha \frac{\max(|C|,|D|)}{\min(|C|,|D|)}\right)$, although it is somewhat sensitive to the choice of the parameter α.

The spectral method requires much computing time, does not always yield optimal bisections, and naturally produces edge separators, requiring some postprocessing to obtain a node separator set. For these reasons, this technique is not now so strongly favoured, and there has been much current research on alternatives that we now describe.

Gilbert, Miller and Teng (1995) have developed a geometric partitioning scheme, and Chan, Gilbert and Teng (1994) have proposed a hybrid of geometric and spectral methods. Ashcraft and Liu (Ashcraft and Liu 1994*b*, Ashcraft and Liu 1994*a*, Ashcraft and Liu 1995, Ashcraft and Liu 1996) have explored a different approach to obtaining separators for graph bisection. They base their method on a domain decomposition approach, defining a multisection by the nodes on the boundaries of the domains and using these to dissect or bisect the graph. The resulting vertex separator can then be refined using variants of methods like that of Fiduccia and Mattheyses (1982). The other main approach to graph bisection is to perform graph reductions, compute a partition cheaply on the resulting coarse graph, and from this construct a partition of the original graph, using some kind of iterative improvement on the projection of this coarse partition on the finer graph (for example, Kernighan and Lin 1970, Fiduccia and Mattheyses 1982). This approach is nested and is termed a multilevel scheme (Bui and Jones 1993). Multilevel schemes have been used by Hendrickson and Leland (1993), Karypis and Kumar (1995*a*), and Hendrickson and Rothberg (1996), inter alios.

In most of these approaches, the dissection technique is only used for the top levels and the resulting subgraphs are ordered by a minimum degree scheme. This hybrid technique was used many years ago by George, Poole and Voigt (1978) and is included in many current implementations (for example, Ashcraft and Liu 1996 and Hendrickson and Rothberg 1996). As can be seen by the dates on the references, these new schemes are all very recent, but current empirical evidence would suggest that they are at least competitive with minimum degree on some large problems from structural analysis. They also perform far better than a minimum degree ordering on some matrices from financial modelling where Berger, Mulvey, Rothberg and Vanderbei (1995) have found practical problems that exhibit similar behaviour to the pathological examples of Rose (1973) that give an arbitrarily poor performance for minimum degree. In several studies on problems from structural analysis, Rothberg (1996*a*) and Ashcraft and Liu (1996) have shown that dissection techniques can outperform minimum degree by on average about 15% in terms of floating-point operations for Cholesky factorization using the resulting ordering, although Ashcraft and Liu (1996) report that the cost of these orderings is several times that of minimum degree.

Of course, dissection techniques are important for purposes other than generating an ordering for a Cholesky factorization. They can be used to

partition an underlying grid for domain decomposition and are equally useful for the parallel implementation of many iterative methods. Two of the major software efforts for developing graph partitioning based on some of the above techniques are CHACO (Hendrickson and Leland 1994) and METIS (Karypis and Kumar 1995*b*).

4 Solution of sets of sparse unsymmetric equations

The same revolution that was just discussed for symmetric orderings has not happened in the case of unsymmetric systems, where usually a Markowitz ordering (Markowitz 1957) or a column ordering based on minimum degree on the normal equations is used together with a threshold criterion for numerical pivoting. The main recent activities for unsymmetric systems have been the development of methods based on partitioning and the production of efficient codes using higher level BLAS operations.

A decade ago, the only widespread use of partitioning was to preorder the matrix to block triangular form prior to performing the analyse-factorize phase on the blocks on the diagonal of that form. Although there were applications where such matrices arose naturally (for example, chemical engineering), this technique did not apply to most systems, and it was common that the largest irreducible block was close to the order of the original matrix. Arioli and Duff (1990) tried to extend block triangularization ideas by using tearing techniques that yield a bordered block triangular form. They found that usually there were too many columns in the border and the amount of arithmetic was much greater than using a standard sparse code on the whole system. Geschiere and Wijshoff (1995) have pursued the use of tearing further and developed a package called MCSPARSE. Gallivan, Hansen, Ostromsky and Zlatev (1995) propose a partitioning similar to a block triangular form for exploitation of parallelism and gain some more advantage through allowing the blocks on the diagonal to be rectangular. We consider these techniques further in Section 7. A major problem with this approach is that the partitioning does not guarantee that the diagonal blocks are well conditioned, or even nonsingular. Thus some *a posteriori* measures must be taken to improve the stability of the factorization. This issue of stability was discussed by Erisman, Grimes, Lewis and Poole (1985) and Arioli, Duff, Gould and Reid (1990) and, more recently, methods for maintaining stability have been developed by Hansen, Ostromsky and Zlatev (1994) and van Duin, Hansen, Ostromsky, Wijshoff

and Zlatev (1995).

A related approach, which I find very appealing, is to expand the matrix, perhaps artificially, in order to obtain a system that is larger and sparser. Normally, there is a choice of pivots for this system that would reduce it to the original system. The logic is that we might be able to do better by using the extra degree of freedom on the expanded system. A simple example of this matrix stretching technique (Grcar 1990, Vanderbei 1991) is the augmented system discussed in Section 5.

A major tool in the efficient implementation of codes for unsymmetric systems has been the observation of Gilbert and Peierls (1988) that partial pivoting can be performed in time proportional to the number of arithmetic operations, so avoiding any potentially costly sorting operations. A nice refinement of their technique was provided by Eisenstat and Liu (1992), who suggested ways of pruning a search tree to reduce work in the symbolic phase. Variants of this technique are used in nearly all sparse partial pivoting codes, for example Duff and Reid (1993) and Demmel et al. (1995).

Frontal, multifrontal, and supernodal approaches for the solution of unsymmetric problems have all seen significant recent developments and were discussed in Section 2 with respect to their use of higher level BLAS. The HSL frontal code MA42, which solves unsymmetric systems, was redesigned to use standard Level 2 and Level 3 BLAS and can accommodate entry by both equations and elements (Duff and Scott 1993, Duff and Scott 1996). If the matrix is close to symmetric in structure in the sense that a_{ij} is usually nonzero when a_{ji} is, then methods adapted from symmetric multifrontal approaches can be used, with the analyse phase performed on the pattern of $A + A^T$ (Duff and Reid 1984, Amestoy and Duff 1989). This approach can work well even if the matrix is unsymmetric, although for highly unsymmetric cases, as we discussed in Section 2, the approach of Davis and Duff (1993) that uses unsymmetric fronts and a directed acyclic graph may be preferable. The main problem with generalizing supernodal techniques to unsymmetric systems is that some regularity in the pattern of the factors is lost and it is not possible to make efficient use of the Level 3 BLAS. Demmel et al. (1995) have overcome this difficulty by using Level 2.5 BLAS, as discussed in Section 2. Liu (1992) has surveyed the use of multifrontal techniques, and recent surveys of frontal and multifrontal methods that include more discussion of the unsymmetric case can be found in Duff (1996) and Dongarra, Duff, Sorensen and van der Vorst (1991). A further strength of the methods considered in this paragraph is that they can be easily developed to exploit parallelism, as we discuss further in Section 7.

5 Solution of indefinite symmetric systems

During the last ten years, the solution of indefinite symmetric systems has been pursued with some vigour. A major impetus for this interest has come from the solution of sparse least-squares problems as a subproblem in the solution of interior-point problems. See Andersen, Gondzio, Mészáros and Xu (1996) for an extensive list of references in this area. The problem with indefinite systems is that numerical pivoting is required and it may not be possible to form a Cholesky factorization. The standard approach is to generalize the 2×2 pivoting technique of Bunch and Parlett (1971) and form the factorization LDL^T where L is lower triangular and D is block diagonal with blocks of order 1 or 2. Although the HSL code MA27 (Duff and Reid 1983), which implements such a strategy, has been widely used for some time to solve indefinite problems, it has recently become apparent that it has severe limitations if the problem is significantly indefinite. This is because it uses an ordering based on structure alone on the assumption that any diagonal entry is numerically suitable for a pivot in the subsequent numerical factorization. An extreme example of this is the solution of the ubiquitous augmented system

$$\begin{pmatrix} H & A \\ A^T & 0 \end{pmatrix} \quad \begin{pmatrix} x \\ y \end{pmatrix} \quad = \quad \begin{pmatrix} b \\ c \end{pmatrix}, \tag{5.1}$$

where the (2,2) block of zeros can mean that the initial analyse phase gives a pivot sequence where many prospective pivots are zero. Additionally, as shown by Duff, Gould, Reid, Scott and Turner (1991), judicious selection of structured pivots of the form $\begin{pmatrix} \times & \times \\ \times & 0 \end{pmatrix}$ or $\begin{pmatrix} 0 & \times \\ \times & 0 \end{pmatrix}$ can preserve much of the structure of the original matrix.

Another common application that gives rise to sparse indefinite systems, usually of more general structure than in (5.1), is the shift and invert strategy for obtaining interior eigenvalues of large sparse matrices.

Duff and Reid (1996*b*) have developed a code, called MA47 in the Harwell Subroutine Library, based on the work of Duff et al. (1991), but have found that, although it sometimes performs far better than MA27, it can sometimes be significantly worse. This is in part because of the added complications in the code and in part because even the extension to encourage the use of structured pivots can fare badly when numerical considerations cause initial pivot choices to be unsuitable. As in the case

of unsymmetric systems, it would appear that to obtain a robust code it is necessary to combine the analyse and factorization phases so that the numerical values of entries are considered when the initial pivot selection is made. This is done by Fourer and Mehrotra (1993), who show some convincing results in their use of structured pivots when solving problems from interior-point methods. Ashcraft, Grimes and Lewis (1995) have studied and analysed various aspects of the extension of threshold pivoting to the case of 2×2 pivots and show that the particular implementation can have quite different numerical properties. They extend their investigation to larger pivot blocks in a attempt to gain performance through the use of higher level BLAS, but the jury is still out on the benefits of using larger blocks.

6 Sparse least-squares

It is possibly fair to say that one of the major growth areas in sparse matrix computations over the last decade has been the solution of sparse least-squares problems. At least part of the reason for this has been the popularity and success of interior-point methods in optimization (see the paper in this volume by Shanno and Simantiraki 1996), where the central and most costly part of the calculation is the solution of a heavily weighted linear least-squares problem. Three main techniques are used to solve these least-squares problems: normal equations, augmented systems, and QR factorization. For a recent excellent review of this area, the book by Björck (1996) is strongly recommended. We discuss each approach in turn.

The attraction of normal equations is that there are several efficient and readily available codes for the solution of sparse symmetric positive definite equations. The problem with this approach is that, because of the scaling, the systems can be very ill-conditioned. However, experience has shown (Wright 1992, Saunders 1994) that the stability of these methods is better than one would expect or deserve, a phenomenon that is discussed in more detail by Shanno and Simantiraki (1996). A further problem with using the normal equations is that a single dense row in A would result in the normal equations being completely dense. This problem was recognized some time ago (George and Heath 1980, George, Heath and Ng 1983) and is avoided by partitioning the system and treating the dense rows by an updating scheme. The selection of which rows to include in the dense part is still an open question (for example, Sun 1995). There is difficulty in selecting

rows so that the remaining problem remains sparse and is not singular or seriously ill-conditioned. Saunders (1994) gives a handy list of alternatives to using Cholesky factorization, while Rothberg and Hendrickson (1996) have explored the performance of various sparse ordering techniques on normal equations matrices from interior-point methods.

We have already discussed the solution of augmented systems in the previous section. For the solution of least-squares problems, the matrix H is diagonal and the vector c on the right-hand side of (5.1) is zero. The beauty here is that, by suitable choice of 2×2 pivots, small entries on the diagonal of the (1,1) block in (5.1) do not cause stability problems. Arioli, Duff and de Rijk (1989) have used the analysis of Arioli et al. (1989) and have conducted several experiments to support the viability of using 2×2 pivots. Björck (1992) provides a detailed error analysis for this approach. Further computational experience in using a scaling on the (1,1) block is presented in Duff (1994).

At first glance, QR methods do not seem very attractive because of the fill-in to the factor Q. Additionally, dense rows in A will cause the factor R to be full. The storage of the denser Q can be avoided, at the cost of possible instability, by using the semi-normal equations (SNE)

$$R^T R = A^T b.$$

Moreover, analysis by Björck (1987) has shown that, in most cases, numerically satisfactory results can be obtained by using the corrected semi-normal equations (CSNE), where one step of iterative refinement is used. Sparse QR factorization uses the observation exploited by George and Heath (1980) that the factor R is the same as the Cholesky factor of the normal equations matrix. Although this is true in exact arithmetic, the difficulty in recognizing numerical cancellation means that the computed structure of the Cholesky factor can overestimate the structure of R. However, in many cases, this structure accurately predicts that of the factor R (Coleman, Edenbrandt and Gilbert 1986), particularly if the original matrix is first permuted to block triangular form (Pothen and Fan 1990). A column ordering of the rectangular matrix A is obtained using a symmetric ordering on the structure of the normal equations, although the ordering can be obtained from A without requiring the formation of $A^T A$ (Gilbert et al. 1992). The column ordering can then be used to construct a computational tree to drive the numerical factorization, which is performed using a QR factorization (for example, Raghavan 1995*a*, Amestoy, Duff and Puglisi

1996, Matstoms 1995). This can be implemented using a supernodal or multifrontal approach that can borrow extensively from the techniques used for sparse LU factorization. These methods are amenable to parallelization (see Section 7). It is also possible to develop rank revealing factorizations by delaying pivoting on columns that are deemed linearly independent. This can be incorporated economically within a multifrontal factorization scheme (Bischof, Lewis and Pierce 1990, Pierce and Lewis 1995). For cases that are particularly badly scaled, the factor Q can be stored and used in a conventional QR solution scheme (Amestoy et al. 1996). Lu and Barlow (1994) show that, for regular problems, the storage of the factor Q for a multifrontal scheme need be no more than that for the factor R, and Gilbert, Ng and Peyton (1993) consider structure prediction techniques for the factor Q.

7 Parallel computing

In spite of the demise of many vendors of parallel machines, there are still several different parallel architectures and the methods for exploiting them differ accordingly. In this review, we will distinguish only the two main classes of machine: shared memory multiprocessors (SMP), including virtual shared memory machines, and distributed memory computers that require some form of message passing to communicate data between processors. The extreme case of the latter, termed network computing, is currently very fashionable but we will consider it only as a form of distributed computing.

There are three levels at which parallelism can be exploited in the solution of sparse linear systems by direct methods. As we discussed in Section 2, many codes use higher level BLAS to perform most or all of the floating-point operations. Many vendors of shared memory computers offer parallel versions of the BLAS and so at this level parallelization is trivial. Given an appropriate distribution of the matrix, parallel versions of the higher level BLAS for distributed memory machines can be constructed, possibly using tools like the BLACS (Basic Linear Algebra Communications Routines) (Whaley 1994).

At the coarsest level, techniques that we introduced in Section 4 for partitioning the matrix are often designed for parallel computing and are especially appropriate for distributed memory computers. Indeed these methods are often only competitive when parallelism is considered. Pothen and Fan (1990) have developed a partitioning, based on the Dulmage and

Mendelsohn canonical decomposition of a bipartite graph, that generalizes the block triangular form to rectangular systems. This has been used by Amestoy et al. (1996) in the QR factorization of sparse rectangular matrices. Zlatev, Waśniewski and Schaumburg (1993) and Zlatev, Waśniewski, Hansen and Ostromsky (1995) have developed a package, PARASPAR, for parallel solution that uses a preordering to partition the original problem. The MCSPARSE package (Geschiere and Wijshoff 1995, Gallivan, Marsolf and Wijshoff 1996) similarly uses a coarse matrix decomposition to obtain an ordering to bordered block triangular form.

At an intermediate level, we can use the sparsity of the matrix to advantage. This could be simply using the ability to choose several pivots simultaneously. Two matrix entries a_{ij} and a_{rs} can be used as pivots simultaneously if a_{is} and a_{rj} are zero. These pivots are termed *compatible*. This observation (Calahan 1973) has been the basis for several algorithms and parallel codes for general matrices. The central theme is to select a number of compatible pivots that would give a diagonal block if ordered to the top left of the matrix. The update from all these pivots is then performed in parallel. The procedure is then repeated on the reduced matrix. The algorithms differ in how the pivots are selected (clearly one must compromise the Markowitz criterion to get a large compatible pivot set) and in how the update is performed. Alaghband (1995) uses compatibility tables to assist in the pivot search. She uses a two-stage implementation where first pivots are chosen in parallel from the diagonal and then off-diagonal pivots are chosen sequentially to stabilize the ordering. She sets thresholds for both sparsity and stability when choosing pivots. Her original experiments were performed on a Denelcor HEP. Davis and Yew (1990) perform their pivot selection in parallel, which results in the nondeterministic nature of their algorithm because the compatible set will be determined by the order in which potential compatible pivots are found. Their algorithm, D2, was designed for shared-memory machines and was tested extensively on an Alliant FX/8. The Y12M algorithm by Zlatev et al. (1995) extends the notion of compatible pivots by permitting the pivot block to be upper triangular rather than diagonal, which allows them to obtain a larger number of pivots, although the update is more complicated. For distributed memory architectures, van der Stappen, Bisseling and van de Vorst (1993) distribute the matrix over the processors in a grid fashion, perform a parallel search for compatible pivots, choosing entries of low Markowitz cost that satisfy a pivot threshold, and perform a parallel rank-m update of the reduced matrix, where m is the number of compatible pivots chosen. Their code was

originally written in OCCAM and run on a network of 400 transputers, but they have since developed a version using PVM (Koster and Bisseling 1994).

A common structure for both visualizing and implementing parallelism is the elimination tree, or derivatives of it. The main property that we exploit in this tree is that computations corresponding to nodes that are not ancestors or descendants of each other are independent (see, for example, Duff 1986, Liu 1987). The tree can thus be used to schedule parallel tasks. For shared memory machines, this can be accomplished through a shared pool of work with fairly simple synchronizations that can be controlled using locks protecting critical sections of the code (Duff 1987, Amestoy 1991). In this way, nearly all the approaches that we have discussed in earlier sections that use frontal, multifrontal, or supernodal techniques to effect LU or QR factorization can be implemented to exploit parallelism. A major issue for an efficient implementation on shared memory machines concerns the management of data, which must be organized so that book-keeping operations such as garbage collection do not cause too much interference with the parallel processing. Johnson and Davis (1992) examine aspects of a parallel buddy memory system, while Amestoy and Duff (1993) discuss and compare several approaches and recommend a hybrid scheme with different memory management in different subregions of storage.

The original experiments of Gilbert and Schreiber (1992) on the massively parallel SIMD Connection machine were a little disappointing although they did indicate the possibility of using massive parallelism in sparse factorization. The experiments did show that a grid-distributed multifrontal implementation substantially outperformed a "Router Cholesky" algorithm based on a fan-in approach. Conroy, Kratzer and Lucas (1994) used a mapping strategy that can trade work against data movement to design a multifrontal algorithm for the TMC CM-5 that is competitive with codes on vector supercomputers. They tested their code on a MasPar MP-2. Manne and Hafsteinsson (1995) have implemented a supernodal fan-out algorithm on the MasPar MP-2 and use a graph colouring algorithm to map the matrix to processors.

The earliest work on parallelizing sparse codes for distributed memory machines was based on column oriented Cholesky factorizations, either the fan-in or the fan-out algorithm. The original codes just used a column-column formulation of the algorithm as in the fan-in algorithm of George, Heath, Liu and Ng (1986) but it was soon apparent that better efficiency could be obtained as in the supernode-column fan-in approach of Ng and Peyton (1993*b*). Some of this early work on parallel algorithms

for distributed memory computers is reviewed by Heath, Ng and Peyton (1991). For distributed memory machines, processors can be assigned work corresponding to subtrees, but this requires quite balanced trees. Geist and Ng (1989) use a breadth-first search strategy to assign work to processors using a heuristic bin packing algorithm to achieve reasonable load balancing. However, the results of runs on L-shape domains on an INTEL iPSC/2 comparing their strategy with wrap mapping and with a nested-dissection ordering and subtree-to-subcube mapping are rather flat, showing only a slight advantage for their heuristic. Pothen and Sun (1993) have adapted the heuristic of Geist and Ng (1989) to a multifrontal scheme and have compared this with a generalized version of a subtree-to-subcube mapping and have found their algorithm to be twice as fast on an INTEL iPSC/2. All approaches to sparse Cholesky factorization have been used to develop parallel factorization routines on hypercubes: George, Heath, Liu and Ng (1989) use a fan-out algorithm, Ashcraft, Eisenstat and Liu (1990) a fan-in algorithm, and Sun (1992*a*) uses a multifrontal approach. Ashcraft, Eisenstat, Liu and Sherman (1990) have compared all three approaches but none of the methods shows very high performance. Sun (1992*b*) describes a package of subroutines implementing his parallel multifrontal algorithm (Sun 1992*a*). George and Ng (1988) show that, even in a distributed environment, it is very beneficial if some memory is available as shared memory to hold information such as mapping vectors. The first work to exploit parallelism at all phases of the sparse solution process was by Zmijewski (1987), later developed by Gilbert and Zmijewski (1987) and Zmijewski and Gilbert (1988). More recently, the work of Heath and Raghavan (1994) also exploits parallelism in all phases, although they require that the matrix be held in Cartesian form; that is, in a two or three dimensional coordinate system. While this is quite natural in the context of discretized PDEs, it is not a convenient interface in general.

Schreiber (1993) presents a clear discussion showing that a one-dimensional mapping of columns or block columns to processors is inherently unscalable and that a two-dimensional mapping is needed to obtain a scalable algorithm. Rothberg (1996*c*) compares a block fan-out algorithm using two-dimensional blocking with a panel multifrontal method using one-dimensional blocking and favours the former, obtaining a performance of over 1.7 Gflop/s on 128 nodes of an Intel Paragon. He points out that the benefit of using higher level BLAS kernels, coupled with the then recent increases in local memory and communication speed of parallel processors, had at last made the solution of large sparse systems feasible on such

architectures. The 2-D block fan-out algorithm is further investigated by Rothberg and Gupta (1994), and Rothberg and Schreiber (1994) propose some block mapping heuristics to improve the performance to over 3 Gflop/s for a 3-D grid problem on a 196-node Intel Paragon. A similar type of 2-dimensional mapping is used by Gupta, Karypis and Kumar (1994) in their implementation of a multifrontal method, where much of the high performance is obtained through balancing the tree near its root and using a careful and regular mapping of the dense matrices near the root to enable a high level of parallelism to be maintained when the trivial parallelism from subtree assignment is exhausted. This work has provided possibly the best performance for distributed memory implementation of a direct sparse Cholesky code. Although the headline figure of nearly 20 Gflop/s on the CRAY T3D was obtained on a fairly artificial and essentially dense problem, large sparse problems from structural analysis were factorized at between 8 and 15 Gflop/s on the same machine. Karypis, Gupta and Kumar (1994) have used this parallel multifrontal algorithm in the solution of interior-point problems, and similarly Bisseling, Doup and Loyens (1993) and Lustig and Rothberg (1996) have used parallel Cholesky factorizations to speed up this computation (see also Shanno and Simantiraki 1996)

Partly because of the success of fast and parallel methods for performing the numerical factorization, other phases of the solution are now becoming more critical on parallel computers. The package of Heath and Raghavan (1994) executes all phases in parallel, and there has been much recent work in finding parallel methods for performing the reordering. This has been another reason for the growth in dissection approaches (for example, see Karypis and Kumar 1995*c*, and Raghavan 1995*c*). It is possible to parallelize the triangular solve by using a tree, similar or identical to that for the numerical factorization. Anderson and Saad (1989) generate a tree given the sparsity structure of the triangular factor, while Amestoy et al. (1996) use the same elimination tree as for the earlier multifrontal factorization. However, in order to avoid the intrinsically sequential nature of a sparse triangular solve, Alvarado, Yu and Betancourt (1990) have proposed holding L^{-1} or rather a partitioned form of this to avoid some of the fill-in that would be associated with forming the inverse explicitly. Various schemes for this partitioning have been proposed to balance the parallelism (limited by number of partitions) with the fill-in (for example, Alvarado, Pothen and Schreiber 1993, Alvarado and Schreiber 1993, Peyton, Pothen and Yuan 1992). Very recently, Raghavan (1995*b*) has proposed the selective inversion of submatrices produced by a multifrontal factorization algorithm.

8 Preconditioning

One of the main problems with sparse LU factorization is that often the number of entries in the factors is substantially greater than in the original matrix so that, even if the original matrix can be stored, the factors cannot. If we assume that we can store the original matrix, then one possibility is to start a sparse LU factorization but drop some fill-in entries so that the partial factors can still be stored. Algorithms for doing this are called incomplete LU (or ILU) factorizations and they differ depending on the criteria for deciding which entries to drop. At one extreme, we could hold all the factors, while at the other we could store no fill-ins. This partial or incomplete factorization is then used to precondition the matrix for iterative solution, normally using a fairly standard Krylov-sequence based iterative technique like conjugate gradients in the symmetric case or GMRES or BiCG when the matrix is unsymmetric.

The main criteria for deciding which entries to include in an incomplete factorization are location and numerical value. The commonest location-based criterion is to allow a set number of levels of fill-in, where original entries have level zero, original zeros have level ∞ and a fill-in in position (i, j) has level $Level_{ij}$ determined by

$$\min_{1 \le k \le \min(i,j)} \{Level_{ik} + Level_{kj} + 1\}.$$

In the case of simple discretizations of partial differential equations, this gives a simple pattern for incomplete factorizations with different levels of fill-in. For example, if the matrix is from a five-point discretization of the Laplacian in two-dimensions, level 1 fill-in will give the original pattern plus a diagonal inside the outermost band. The other main criterion for deciding which entries to omit is to drop entries less than a predetermined numerical value. For the regular problems just mentioned, it is interesting that the level fill-in and drop strategies give a somewhat similar incomplete factorization, because the numerical value of successive fill-in levels decreases markedly, reflecting the characteristic decay in the entries of the inverse matrix (see, for example, Meurant 1992). For general problems, however, the two strategies can be significantly different. Since it is usually not known *a priori* how many entries will be above a selected threshold, the dropping strategy is normally combined with restricting the number of fill-ins allowed to any one column (Saad 1994*a*). When using a threshold criterion, it is possible to change it dynamically during the factorization to attempt to achieve a target density

of the factors (Munksgaard 1980). Tismenetsky (1991) has developed a more robust but generally denser incomplete factorization by only excluding entries outside a specified pattern (in which case the diagonal is modified) and those fill-ins that would be caused by the product of two small entries. Although the notation is not yet fully standardized, the nomenclature commonly adopted for incomplete factorizations is ILU(k), when k levels of fill-in are allowed and ILUT(α, f), for the threshold criterion when entries of modulus less than α are dropped and the maximum number of fill-ins allowed in any column is f. There are many variations on these strategies and the criteria are sometimes combined. In some cases, constraining the row sums of the incomplete factorization to match those of the matrix can help (Gustafsson 1979). Additionally, in the symmetric positive-definite case, steps are often taken to ensure that the resulting preconditioner is also positive definite by modifying the matrix being factorized (Manteuffel 1980) or adjusting diagonal entries of the factorization (Jennings and Malik 1977, Munksgaard 1980, Ajiz and Jennings 1984).

The use of incomplete factorizations as preconditioners for symmetric systems has a long pedigree (Meijerink and van der Vorst 1977) and good results have been obtained for a wide range of problems. An incomplete Cholesky factorization where one level of fill-in is allowed (ICCG(1)) has proven to provide a good balance between reducing the number of iterations and the cost of computing and using the preconditioning. Although it may be thought that a preordering that would result in low fill-in for a complete factorization (for example, minimum degree) might be advantageous for an incomplete factorization, Duff and Meurant (1989) and Eijkhout (1991) show that it is not true in general and that sometimes the number of iterations of ICCG(0) can double if a minimum degree ordering is used, although this effect is not apparent for ILUT preconditioners. The situation with symmetric systems is quite well analysed and understood. Much recent work for symmetric systems has been to develop preconditioners that can be computed and used on parallel computers. Most of this work has, however, been applicable to highly structured problems from discretizations of elliptic partial differential equations in two and three dimensions, for example van der Vorst (1989). Heroux, Vu and Yang (1991) and Jones and Plassmann (1994) have experimented with unstructured matrices, with reasonable speed-ups being achieved in the latter paper.

The situation for unsymmetric systems is, however, much less clear. Although there have been many experiments on using incomplete factorizations and there have been studies of the effect of orderings on the

number of iterations (D'Azevedo, Forsyth and Tang 1992, Dutto 1993) that show similar behaviour to the symmetric case, there is very little theory governing the behaviour for general systems and indeed the performance of ILU preconditioners is very unpredictable. Allowing high levels of fill-in can help but again there is no guarantee. In fact, a major problem is that the ILU factors can become very much more ill-conditioned than the original system and so the preconditioned system can perform much worse than the original matrix with respect to convergence of the iterative method. This phenomenon is analysed by Elman (1986) for the case of an ILU(0) preconditioner for systems from convection-diffusion problems.

The QR or LQ factorizations can also be used in the unsymmetric case to derive an incomplete factorization (Jennings and Ajiz 1984, Saad 1988*a*). Here the orthogonal factor need not be kept and the resulting incomplete triangular factor can be used to precondition the normal equations. Preconditioners based on these factorizations are generally more expensive to compute and use than ILU preconditioners but they are usually more robust. Saad (1988*b*) has examined the use of incomplete LQ factorizations as preconditioners for nonsymmetric and indefinite systems, and Benzi and Tůma (1996) have confirmed that preconditionings based on incomplete orthogonalization methods can succeed where ILU preconditioners fail. One way of computing an incomplete orthogonal factorization is to use incomplete modified Gram-Schmidt (IMGS). The IMGS approach has been explored by Wang, Gallivan and Bramley (1996) and is described in detail in the thesis by Wang (1993).

Of course, the LU and LQ factorizations are ways of representing the inverse of a sparse matrix in a way that can be economically used to solve linear systems. The main reason why explicit inverses are not used is that, for irreducible matrices, the inverse will always be dense (because we neglect numerical cancellation, see Duff, Erisman, Gear and Reid 1988). However, this need not be a problem if we follow the flavour of ILU factorizations and compute and use a sparse approximation to the inverse. Perhaps the most interesting technique for this is to solve the problem

$$\min_M ||I - AM||_F, \tag{8.1}$$

where M has some fully or partially prescribed sparsity structure. One advantage of this is that this problem can be split into n independent least-squares problems for each of the n columns of M. Each of these least-squares problems only involves a few variables (corresponding to the number

of entries in the column of M) and, because they are independent, they can be solved in parallel. A further benefit of such techniques is that it is possible to successively increase the density of the approximation to reduce the value of (8.1) (Cosgrove, Diaz and Griewank 1992) and so, in principle, ensure convergence of the preconditioned iterative method. Cosgrove et al. (1992), Huckle and Grote (1994), and Gould and Scott (1995) use a (dense) QR factorization to solve the small least-squares problems while Chow and Saad (1994) use GMRES. Gould and Scott (1995) show that this technique gives almost as good a preconditioner as ILU but is much more expensive to compute both in terms of time and storage, at least if computed sequentially. One problem with these approaches is that, although the residual for the approximation of a column of M can be controlled (albeit perhaps at the cost of a rather dense column in M), the nonsingularity of the matrix M is not guaranteed. Partly to avoid this, Kolotilina and Yeremin (1993) have proposed approximating the triangular factors of the inverse and, to this end, Benzi and Tůma (1995) generate sparse approximations to an A-biconjugate set of vectors using drop tolerances. In a scalar or vector environment, it is also much cheaper to generate the factors by this means than to solve the least-squares problems for columns of the approximate inverse.

One of the main reasons for the interest in sparse approximate inverse preconditioners is the difficulty of parallelizing ILU preconditioners, not only in their construction but also in their use, which requires a sparse triangular solution. However, although almost every paper on approximate inverse preconditioners states that the authors are working on a parallel implementation, there are relatively few such studies available. Grote and Simon (1993) perform limited studies when the matrix is highly structured. Gustafsson and Lindskog (1995), using an idea of van der Vorst (1982), have implemented a fully parallel preconditioner based on truncated Neumann expansions to approximate the inverse SSOR factors of the matrix. Their experiments on a CM-200 show a worthwhile improvement over a simple diagonal scaling.

Note that, because the inverse of the inverse of a sparse matrix is sparse (a fact not detected because we do not allow numerical cancellation), there are classes of dense matrices for which a sparse approximate inverse might be a very appropriate preconditioner.

There is a midway house between the LU factors and the inverse and that is to use the explicit inverses of L and U. This has been proposed by Alvarado and Schreiber (1993) because such a factorization is easier to use in parallel than an LU factorization. An incomplete form of this factorization

for use as a preconditioner has been proposed by Alvarado and Dağ (1994).

Of course, it is possible to represent the inverse by a polynomial in the matrix and use this polynomial as a preconditioner. One approach, by Dubois, Greenbaum and Rodrigue (1979), is to use the low order terms of a Neumann expansion of $(I - B)^{-1}$, where $A = I - B$ and the spectral radius of B is less than 1. They use a matrix splitting $A = M - N$ and a truncated power series for $M^{-1}N$ when the condition on B is not satisfied. More general polynomial preconditioners have also been proposed (see, for example, Johnson, Micchelli and Paul 1983, Saad 1985, and Ashby 1991). For efficiency, low degree polynomials are normally used. A polynomial preconditioner is simple to use and can be parallelized, but the results are not generally very encouraging and have been particularly disappointing for unsymmetric problems.

In the case of unassembled element problems, it is important to develop preconditioners that do not require assembly of the matrix. This sometimes involves the factorization of the element submatrices, and hence they use a direct method in computing the preconditioner, albeit usually on a small dense matrix (Gustafsson and Lindskog 1986). Daydé, L'Excellent and Gould (1996) show that performing some subassemblies before computing the preconditioning can help, but the evidence suggests that any savings in iterations for the iterative method is about balanced by the extra work in using the preconditioner.

This concludes our discussion of incomplete factorizations or other incomplete representations of the inverse. Another whole class of preconditioners that use direct methods are those where the direct method is used to solve a subproblem of the original problem. This is often used in a domain decomposition setting, where problems on subdomains are solved by the direct method but the interaction between the subproblems is handled by an iterative technique. A related example of this is the work on block projection methods like Block Cimmino (Arioli, Duff, Noailles and Ruiz 1992) or Block Kacmarz (Bramley and Sameh 1992). Block preconditioning for symmetric systems is discussed by Concus, Golub and Meurant (1985), and Concus and Meurant (1986) use incomplete factorizations within the diagonal blocks. Attempts have been made to preorder matrices to put large entries into the diagonal blocks so that the inverse of the matrix would be well approximated by the block diagonal matrix whose block entries are the inverses of the diagonal blocks (Choi and Szyld 1996). We do not expand on these techniques here but leave further consideration to the chapter on iterative methods.

Multigrid techniques also often combine aspects of both iterative and direct methods. These methods were originally developed for solving partial differential equations but developments such as algebraic multigrid extend their applicability to more general systems, although the jury is still out on how wide this range is. The basic idea is to use corrections on a sequence of coarser grids to update the required solution on a fine grid. In our context, it is common to use a direct method for the solution on the coarsest grid with one or two iterations of usually a simple iterative method on the other grids. Hackbusch (1985) and Wesseling (1992) are worth reading for a background to multigrid methods, which are further considered in the chapter on iterative methods (Golub and van der Vorst 1996).

9 Towards a sparse problem solving environment

Often the solution of a set of sparse linear equations is the most costly part of the computation but, to the user of sparse codes, there might be as much cost in organizing and managing the data for the application in preparation for and after the solution step. A major problem is that a sparse data structure can be represented in many different ways. Most are very problem specific and it may not be trivial to organize the data for a call to the linear equation solution routine. This might be even more complicated should parallelism be exploited. There are some tools being developed to assist in the manipulation and management of sparse matrices.

Gilbert et al. (1992) have introduced a sparse matrix structure and some sparse algorithms into MATLAB. Their aim has been ease of use and functionality rather than efficiency, although increasingly researchers are making codes available to MATLAB users through M-files (for example, Matstoms 1994). Saad (1994*b*) has developed, over many years, a tool kit called SPARSKIT for sparse matrix computations, Gupta and Rothberg (1994) have proposed an environment for handling sparse matrices on distributed memory machines, and Alvarado (1989) has designed an integrated package as a teaching and development tool called SMMS (Sparse Matrix Manipulation System). We have been keen to stress that the important kernels are those for dense linear algebra. However, in the case of iterative methods and the use of preconditioning matrices, a sparse version of the BLAS is appropriate (Duff, Marrone, Radicati and Vittoli 1995). The provision of a standard set of sparse matrix test problems, the Harwell-Boeing Collection (Duff, Grimes and Lewis 1989, Duff, Grimes and

Lewis 1992), has proven to be very useful for the design and comparison of algorithms, and there is currently an effort underway to update this collection and create a more friendly interface through the World Wide Web (presently at URL `http://math.nist.gov/MatrixMarket`).

We do not feel that a software review is appropriate here and indeed a thorough one would require as much space and effort as this present survey. Suffice it to say that there are a number of sparse direct solvers available through `netlib`, at URL `http://www.netlib.org/`, and possibly the largest collection of Library quality sparse direct codes is included in the Harwell Subroutine Library and in a subset of that Library, the Harwell Sparse Matrix Library (HSML), marketed by NAG. Further information on HSL and HSML can be obtained from the Web page `http://www.rl.ac.uk/departments/ccd/numerical/hsl/hsl.html`.

10 Concluding remarks

Far be it from me to try to steal the thunder of the author of the equivalent talk ten years from now. However, after nine sections of reflections, it might be entertaining to suggest where the excitement may lie in the years to come.

First, I believe that the iterative and direct talks will be combined at the next meeting as they were ten years ago. I believe this because it is already becoming clear that the huge (order greater than 500,000) problems of tomorrow can only be solved by combining direct and iterative techniques, which is why I spent a full section on preconditioning methods. It is not that smaller problems (order 10,000 to 100,000) do not need to be solved but we essentially already have the tools to do this efficiently on serial computers, and this will fairly soon be routine on parallel computers also. Looking into my crystal ball, I think that soon the symmetric ordering problem will be resolved in favour of a class of hybrid methods (that is, methods involving both dissection techniques and minimum degree) parameterized to accommodate any sparse structure; the unsymmetric multifrontal and supernodal approaches will be available on distributed memory machines and will be so efficient that partitioning methods will only be used on huge systems prior to constructing a preconditioner. I think that row projection methods will be developed further and robust direct solvers will be used on the projected problems. Rather than developing a LAPACK approach to providing software for direct solution of sparse equations, a MATLAB-like environment will handle everything from problem formulation to post-

analysis of the solution.

One thing I am sure of ... the sparse specialist will still have a job in ten years time ... at least I sincerely hope so!!

Acknowledgements

I asked a large number of people if they would look at a draft of this paper, particularly to check if I had the best reference to their work. I received many replies and am grateful to Patrick Amestoy, Cleve Ashcraft, Michele Benzi, Åke Björck, Randall Bramley, John Gilbert, Jacko Koster, John Reid, Edward Rothberg, Michael Saunders, Jennifer Scott, Henk van der Vorst, and Zahari Zlatev for exceeding their brief by commenting more generally on various aspects of the draft.

References

Ajiz, M. A. and Jennings, A. (1984), 'A robust incomplete Choleski-conjugate gradient algorithm', *Int J. Numerical Methods in Engineering* **20**, 949–966.

Alaghband, G. (1995), 'Parallel sparse matrix solution and performance', *Parallel Computing* **21**(9), 1407–1430.

Alvarado, F. L. (1989), 'Manipulation and visualisation of sparse matrices', *ORSA J. Computing* **2**, 186–207.

Alvarado, F. L. and Dağ, H. (1994), Incomplete partitioned inverse preconditioners, Technical Report (to appear), Department of Electrical and Computer Engineering, University of Wisconsin at Madison. Submitted to Parallel Computing.

Alvarado, F. L. and Schreiber, R. (1993), 'Optimal parallel solution of sparse triangular systems', *SIAM J. Scientific Computing* **14**, 446–460.

Alvarado, F. L., Pothen, A. and Schreiber, R. (1993), Highly parallel sparse triangular solution, *in* A. George, J. R. Gilbert and J. W. H. Liu, eds, 'Graph Theory and Sparse Matrix Computation', Springer-Verlag.

Alvarado, F. L., Yu, D. C. and Betancourt, R. (1990), 'Partitioned sparse A^{-1} methods', *IEEE Trans. Power Systems* **3**, 452–459.

Amestoy, P. R. (1991), Factorization of large sparse matrices based on a multifrontal approach in a multiprocessor environment, INPT PhD Thesis TH/PA/91/2, CERFACS, Toulouse, France.

Amestoy, P. R. and Duff, I. S. (1989), 'Vectorization of a multiprocessor multifrontal code', *Int. J. of Supercomputer Applics.* **3**, 41–59.

Amestoy, P. R. and Duff, I. S. (1993), 'Memory management issues in sparse multifrontal methods on multiprocessors', *Int. J. Supercomputer Applics* **7**, 64–82.

Amestoy, P. R., Davis, T. A. and Duff, I. S. (1995), An approximate minimum degree ordering algorithm, Technical Report TR/PA/95/09, CERFACS, Toulouse, France. To appear in *SIAM J. Matrix Analysis and Applications*.

Amestoy, P. R., Duff, I. S. and Puglisi, C. (1996), 'Multifrontal QR factorization in a multiprocessor environment', *Numerical Linear Algebra with Applications* **3**(4), 275–300.

Andersen, E. D., Gondzio, J., Mészáros, C. and Xu, X. (1996), Implementation of interior point methods for large scale linear programming, Technical Report 1996.3, Logilab, University of Geneva, Switzerland.

Anderson, E. C. and Saad, Y. (1989), 'Solving sparse triangular systems on parallel computers', *Int J. High Speed Computing* **1**, 73–95.

Arioli, M. and Duff, I. S. (1990), Experiments in tearing large sparse systems, *in* M. G. Cox and S. Hammarling, eds, 'Reliable Numerical Computation', Oxford University Press, Oxford, pp. 207–226.

Arioli, M., Demmel, J. W. and Duff, I. S. (1989), 'Solving sparse linear systems with sparse backward error', *SIAM J. Matrix Analysis and Applications* **10**, 165–190.

Arioli, M., Duff, I. S. and de Rijk, P. P. M. (1989), 'On the augmented systems approach to sparse least-squares problems', *Numer. Math.* **55**, 667–684.

Arioli, M., Duff, I. S., Gould, N. I. M. and Reid, J. K. (1990), 'Use of the P^4 and P^5 algorithms for in-core factorization of sparse matrices', *SIAM J. Sci. Stat. Comput* **11**, 913–927.

Arioli, M., Duff, I. S., Noailles, J. and Ruiz, D. (1992), 'A block projection method for sparse equations', *SIAM J. Scientific and Statistical Computing* **13**, 47–70.

Ashby, S. F. (1991), 'Minimax polynomial preconditioning for Hermitian linear systems', *SIAM J. Matrix Analysis and Applications* **12**(4), 766–789.

Ashcraft, C. (1995), 'Compressed graphs and the minimum degree algorithm', *SIAM J. Scientific Computing* **16**, 1404–1411.

Ashcraft, C. and Liu, J. W. H. (1994*a*), Generalized nested dissection: some recent progress, *in* J. G. Lewis, ed., 'Proceedings of the Fifth SIAM Conference on Applied Linear Algebra', SIAM Press, Philadelphia, pp. 130–139.

Ashcraft, C. and Liu, J. W. H. (1994*b*), A partition improvement algorithm for generalized nested dissection, Technical Report BCSTECH-94-020, Boeing Computer Services, Seattle.

Ashcraft, C. and Liu, J. W. H. (1995), Using domain decomposition to find graph bisectors, Technical Report ISSTECH-95-024, Boeing Information and Support Services, Seattle. Also Report CS-95-08, Department of Computer Science, York University, Ontario, Canada.

Ashcraft, C. and Liu, J. W. H. (1996), Robust ordering of sparse matrices using multisection, Technical Report ISSTECH-96-002, Boeing Information and Support Services, Seattle. Also Report CS-96-01, Department of Computer Science, York University, Ontario, Canada.

Ashcraft, C., Eisenstat, S. C. and Liu, J. W. H. (1990), 'A fan-in algorithm for distributed sparse numerical factorization', *SIAM J. Scientific and Statistical Computing* **11**, 593–599.

Ashcraft, C., Eisenstat, S. C., Liu, J. W. H. and Sherman, A. H. (1990), A comparison on three column-based distributed sparse factorization schemes, Technical Report CS-90-09, Department of Computer Science, York University, York, Ontario, Canada.

Ashcraft, C., Grimes, R. G. and Lewis, J. G. (1995), Accurate symmetric indefinite linear equation solvers, Technical Report ISSTECH-95-029, Boeing Computer Services, Seattle.

Barnard, S. T. and Simon, H. (1993), A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems, *in* R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold and D. A. Reed, eds, 'Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing', SIAM Press, pp. 711–718.

Barnard, S. T., Pothen, A. and Simon, H. (1995), 'A spectral algorithm for envelope reduction of sparse matrices', *Numerical Linear Algebra with Applications* **2**(4), 317–334.

Benzi, M. and Tůma, M. (1995), A sparse approximate inverse preconditioner for nonsymmetric linear systems, Technical Report No. 653, Institute of Computer Science, Academy of Sciences of the Czech Republic. To appear in *SIAM J. Scientific Computing*.

Benzi, M. and Tůma, M. (1996), A comparison of some preconditioning techniques for general sparse matrices, *in* S. D. Margenov and P. S. Vassilevski, eds, 'Iterative Methods in Linear Algebra, II', Volume 3 in the IMACS Series in Computational and Applied Mathematics, IMACS, pp. 191–203.

Berger, A., Mulvey, J., Rothberg, E. and Vanderbei, R. (1995), Solving multistage stochastic programs using tree dissection, Technical Report SOR-97-07, Programs in Statistics and Operations Research, Princeton University, Princeton, New Jersey.

Berman, P. and Schnitger, G. (1990), 'On the performance of the minimum degree ordering for Gaussian elimination', *SIAM J. Matrix Analysis and Applications* **11**(1), 83–88.

Bischof, C. H., Lewis, J. G. and Pierce, D. J. (1990), 'Incremental condition estimation for sparse matrices', *SIAM J. Matrix Analysis and Applications* **11**, 644–659.

Bisseling, R. H., Doup, T. M. and Loyens, L. D. J. C. (1993), 'A parallel interior point algorithm for linear programming on a network of transputers', *Annals of Operations Research* **43**, 51–86.

Björck, Å. (1987), 'Stability analysis of the method of semi-normal equations for least squares problems', *Linear Algebra and its Applications* **88/89**, 31–48.

Björck, Å. (1992), Pivoting and stability in the augmented system method, *in* D. F. Griffiths and G. A. Watson, eds, 'Numerical Analysis 1991, Proceedings of the 14th Dundee Conference, June 1991', Pitman Research Notes in Mathematics Series. **260**, Longman Scientific & Technical, Harlow, England, pp. 1–16.

Björck, Å. (1996), *Numerical Methods for Least Squares Problems*, SIAM Press, Philadelphia.

Bodlaender, H. L., Gilbert, J. R., Hafsteinsson, H. and Kloks, T. (1995), 'Approximating treewidth, pathwidth, frontsize, and shortest elimination tree', *Journal of Algorithms* **18**, 238–255.

Bramley, R. and Sameh, A. (1992), 'Row projection methods for large nonsymmetric linear systems', *SIAM J. Scientific and Statistical Computing* **13**, 168–193.

Bui, T. and Jones, C. (1993), A heuristic for reducing fill-in in sparse matrix factorization, *in* R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold and D. A. Reed, eds, 'Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing', SIAM, pp. 445–452.

Bunch, J. R. and Parlett, B. N. (1971), 'Direct methods for solving symmetric indefinite systems of linear equations', *SIAM J. Numerical Analysis* **8**, 639–655.

Calahan, D. A. (1973), Parallel solution of sparse simultaneous linear equations, *in* 'Proceedings 11th Annual Allerton Conference on Circuits and System Theory, University of Illinois', pp. 729–735.

Cavers, I. A. (1989), Using deficiency measure for tie-breaking the minimum degree algorithm, Technical Report 89-2, University of British Columbia, Canada.

Chan, T., Gilbert, J. and Teng, S.-H. (1994), Geometric spectral partitioning, Technical Report CSL-94-15, Palo Alto Research Center, Xerox Corporation, California.

Choi, H. and Szyld, D. B. (1996), Threshold ordering for preconditioning nonsymmetric problems with highly varying coefficients, Technical Report 96-51, Department of Mathematics, Temple University, Philadelphia.

Chow, E. and Saad, Y. (1994), Approximate inverse preconditioners for general sparse matrices, Technical Report UMSI 94/101, University of Minnesota Supercomputer Institute.

Coleman, T. F., Edenbrandt, A. and Gilbert, J. R. (1986), 'Predicting fill for sparse orthogonal factorization', *J. ACM* **33**, 517–532.

Concus, P. and Meurant, G. (1986), 'On computing INV block preconditionings for the conjugate gradient method', *BIT* **26**, 493–504.

Concus, P., Golub, G. H. and Meurant, G. (1985), 'Block preconditioning for the conjugate gradient method', *SIAM J. Scientific and Statistical Computing* **6**, 220–252.

Conroy, J. M., Kratzer, S. G. and Lucas, R. F. (1994), Data-parallel sparse matrix factorization, *in* J. G. Lewis, ed., 'Proceedings 5th SIAM Conference on Linear Algebra', SIAM Press, Philadelphia, pp. 377–381.

Cosgrove, J. D. F., Diaz, J. C. and Griewank, A. (1992), 'Approximate inverse preconditionings for sparse linear systems', *Int J. Computer Mathematics* **44**, 91–110.

Davis, T. A. and Duff, I. S. (1993), An unsymmetric-pattern multifrontal method for sparse LU factorization, Technical Report RAL 93-036, Rutherford Appleton Laboratory. To appear in *SIAM J. Matrix Analysis and Applications*.

Davis, T. A. and Duff, I. S. (1995), A combined unifrontal/multifrontal method for unsymmetric sparse matrices, Technical Report TR-95-020, Computer and Information Science Department, University of Florida.

Davis, T. A. and Yew, P. C. (1990), 'A nondeterministic parallel algorithm for general unsymmetric sparse LU factorization', *SIAM J. Matrix Analysis and Applications* **11**, 383–402.

Daydé, M. J., Duff, I. S. and Petitet, A. (1994), 'A parallel block implementation of Level 3 BLAS kernels for MIMD vector processors', *ACM Trans. Math. Softw.* **20**, 178–193.

Daydé, M. J., L'Excellent, J.-Y. and Gould, N. I. M. (1996), On the preprocessing of sparse unassembled linear systems for efficient solution using element-by-element preconditioners, Technical Report RT/APO/96/2, Département Informatique, ENSEEIHT-IRIT, Toulouse.

D'Azevedo, E. F., Forsyth, P. A. and Tang, W. P. (1992), 'Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems', *SIAM J. Matrix Analysis and Applications* **13**, 944–961.

Dembart, B. and Erisman, A. M. (1973), 'Hybrid sparse matrix methods', *IEEE Trans. Circuit Theory* **CT-20**, 641–649.

Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S. and Liu, J. W. H. (1995), A supernodal approach to sparse partial pivoting, Technical Report UCB//CSD-95-883, Computer Science Division, U. C. Berkeley, Berkeley, California.

Dongarra, J. J., Du Croz, J., Duff, I. S. and Hammarling, S. (1990), 'A set of Level 3 Basic Linear Algebra Subprograms.', *ACM Trans. Math. Softw.* **16**, 1–17.

Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, H. A. (1991), *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM Press, Philadelphia. Second edition in preparation.

Dubois, D. F., Greenbaum, A. and Rodrigue, G. H. (1979), 'Approximating the inverse of a matrix for use on iterative algorithms on vector processors', *Computing* **22**, 257–268.

Duff, I. S. (1977), MA28 – A set of Fortran subroutines for sparse unsymmetric linear equations, Technical Report AERE R8730, Her Majesty's Stationery Office, London.

Duff, I. S. (1981), Full matrix techniques in sparse Gaussian elimination, *in* G. A. Watson, ed., 'Numerical Analysis Proceedings, Dundee 1981', Lecture Notes in Mathematics 912, Springer-Verlag, Berlin, pp. 71–84.

Duff, I. S. (1986), The use of vector and parallel computers in the solution of large sparse linear equations, *in* P. Deuflhard and B. Engquist, eds, 'Large scale scientific computing. Progress in Scientific Computing Volume 7', Birkhäuser, Boston, pp. 331–348.

Duff, I. S. (1987), The influence of vector and parallel computers in the solution of large sparse linear equations, *in* M. J. D. Powell and A. Iserles, eds, 'The State of the Art in Numerical Analysis', Oxford University Press, Oxford, pp. 359–407.

Duff, I. S. (1994), The solution of augmented systems, *in* D. F. Griffiths and G. A. Watson, eds, 'Numerical Analysis 1993, Proceedings of the 15th Dundee Conference, June-July 1993', Pitman Research Notes in Mathematics Series. **303**, Longman Scientific & Technical, Harlow, England, pp. 40–55.

Duff, I. S. (1996), 'A review of frontal methods for solving linear systems', *Computer Physics Communications* **96**, xxx–xxx.

Duff, I. S. and Meurant, G. A. (1989), 'The effect of ordering on preconditioned conjugate gradients', *BIT* **29**, 635–657.

Duff, I. S. and Reid, J. K. (1983), 'The multifrontal solution of indefinite sparse symmetric linear systems', *ACM Trans. Math. Softw.* **9**, 302–325.

Duff, I. S. and Reid, J. K. (1984), 'The multifrontal solution of unsymmetric sets of linear systems', *SIAM J. Scientific and Statistical Computing* **5**, 633–641.

Duff, I. S. and Reid, J. K. (1993), MA48, a Fortran code for direct solution of sparse unsymmetric linear systems of equations, Technical Report RAL 93-072, Rutherford Appleton Laboratory.

Duff, I. S. and Reid, J. K. (1995), MA47, a Fortran code for direct solution of indefinite sparse symmetric linear systems, Technical Report RAL 95-001, Rutherford Appleton Laboratory.

Duff, I. S. and Reid, J. K. (1996*a*), 'The design of MA48, a code for the direct solution of sparse unsymmetric linear systems of equations', *ACM Trans. Math. Softw.* **22**(2), 187–226.

Duff, I. S. and Reid, J. K. (1996*b*), 'Exploiting zeros on the diagonal in the direct solution of indefinite sparse symmetric linear systems', *ACM Trans. Math. Softw.* **22**(2), 227–257.

Duff, I. S. and Scott, J. A. (1993), MA42 – a new frontal code for solving sparse unsymmetric systems, Technical Report RAL 93-064, Rutherford Appleton Laboratory.

Duff, I. S. and Scott, J. A. (1996), 'The design of a new frontal code for solving sparse unsymmetric systems', *ACM Trans. Math. Softw.* **22**(1), 30–45.

Duff, I. S., Erisman, A. M. and Reid, J. K. (1976), 'On George's nested dissection method', *SIAM J. Numerical Analysis* **13**, 686–695.

Duff, I. S., Erisman, A. M., Gear, C. W. and Reid, J. K. (1988), 'Sparsity structure and Gaussian elimination', *SIGNUM Newsletter* **23**(2), 2–8.

Duff, I. S., Gould, N. I. M., Lescrenier, M. and Reid, J. K. (1990), The multifrontal method in a parallel environment, *in* M. G. Cox and S. Hammarling, eds, 'Reliable Numerical Computation', Oxford University Press, Oxford, pp. 93–111.

Duff, I. S., Gould, N. I. M., Reid, J. K., Scott, J. A. and Turner, K. (1991), 'Factorization of sparse symmetric indefinite matrices', *IMA J. Numerical Analysis* **11**, 181–204.

Duff, I. S., Grimes, R. G. and Lewis, J. G. (1989), 'Sparse matrix test problems', *ACM Trans. Math. Softw.* **15**(1), 1–14.

Duff, I. S., Grimes, R. G. and Lewis, J. G. (1992), Users' guide for the Harwell-Boeing sparse matrix collection (Release I), Technical Report RAL 92-086, Rutherford Appleton Laboratory.

Duff, I. S., Marrone, M., Radicati, G. and Vittoli, C. (1995), A set of Level 3 Basic Linear Algebra Subprograms for sparse matrices, Technical Report TR-RAL-95-049, RAL.

Dutto, L. C. (1993), 'The effect of ordering on preconditioned GMRES algorithm, for solving the Navier-Stokes equations', *Int J. Numerical Methods in Engineering* **36**(3), 457–497.

Eijkhout, V. (1991), 'Analysis of parallel incomplete point factorizations', *Linear Algebra and its Applications* **154-156**, 723–740.

Eisenstat, S. C. and Liu, J. W. H. (1992), 'Exploiting structural symmetry in unsymmetric sparse symbolic factorization', *SIAM J. Matrix Analysis and Applications* **13**, 202–211.

Elman, H. C. (1986), 'A stability analysis of incomplete LU factorizations', *Mathematics of Computation* **47**, 191–217.

Erisman, A. M., Grimes, R. G., Lewis, J. G. and Poole, W. G. J. (1985), 'A structurally stable modification of Hellerman-Rarick's P^4 algorithm for reordering unsymmetric sparse matrices', *SIAM J. Numerical Analysis* **22**, 369–385.

Fiduccia, C. M. and Mattheyses, R. M. (1982), A linear-time heuristic for improving network partitions, *in* 'Proceedings 19th ACM IEEE Design Automation Conference', IEEE Press, New York, NY, pp. 175–181.

Fourer, R. and Mehrotra, S. (1993), 'Solving symmetric indefinite systems in an interior-point method for linear programming', *Mathematical Programming* **62**, 15–39.

Gallivan, K., Hansen, P. C., Ostromsky, T. and Zlatev, Z. (1995), 'A locally optimized reordering algorithm and its application to a parallel sparse linear system solver', *Computing* **54**, 39–67.

Gallivan, K., Marsolf, B. A. and Wijshoff, H. A. G. (1996), 'Solving large nonsymmetric sparse linear systems using MCSPARSE', *Parallel Computing* **22**, xxx–xxx.

Geist, A. and Ng, E. (1989), 'Task scheduling for parallel sparse Cholesky factorization', *Int J. Parallel Programming* **18**, 291–314.

George, A. (1973), 'Nested dissection of a regular finite element mesh', *SIAM J. Numerical Analysis* **10**, 345–363.

George, A. and Heath, M. T. (1980), 'Solution of sparse linear least squares problems using Givens rotations', *Linear Algebra and its Applications* **34**, 69–83.

George, A. and Liu, J. W. H. (1979), 'The design of a user interface for a sparse matrix package', *ACM Trans. Math. Softw.* **5**(2), 139–162.

George, A. and Liu, J. W. H. (1989), 'The evolution of the minimum degree ordering algorithm', *SIAM Review* **31**(1), 1–19.

George, A. and Ng, E. (1988), 'Shared versus local memory in parallel sparse matrix computations', *SIGNUM Newsletter* **23**(2), 9–13.

George, A., Heath, M. T. and Ng, E. (1983), 'A comparison of some methods for solving sparse linear least-squares problems', *SIAM J. Scientific and Statistical Computing* **4**, 177–187.

George, A., Heath, M. T., Liu, J. W. H. and Ng, E. (1986), 'Solution of sparse positive-definite systems on a shared memory multiprocessor', *Int J. Parallel Programming* **15**, 309–325.

George, A., Heath, M. T., Liu, J. W. H. and Ng, E. (1989), 'Solution of sparse positive definite systems on a hypercube', *J. Comput. Appl. Math.* **27**, 129–156.

George, A., Liu, J. W. H. and Ng, E. G. (1980), User's guide for SPARSPAK: Waterloo sparse linear equations package, Technical Report CS-78-30 (Revised), University of Waterloo, Canada.

George, A., Poole, J. W. and Voigt, R. (1978), 'Incomplete nested dissection for solving n by n grid problems', *SIAM J. Numerical Analysis* **15**, 663–673.

Geschiere, J. P. and Wijshoff, H. A. G. (1995), 'Exploiting large grain parallelism in a sparse direct linear system solver', *Parallel Computing* **21**(8), 1339–1364.

Gilbert, J. R. (1988), 'Some nested dissection order is nearly optimal', *Information Processing Letters* **26**, 325–328.

Gilbert, J. R. and Liu, J. W. H. (1993), 'Elimination structures for unsymmetric sparse LU factors', *SIAM J. Matrix Analysis and Applications* **14**, 334–354.

Gilbert, J. R. and Peierls, T. (1988), 'Sparse partial pivoting in time proportional to arithmetic operations', *SIAM J. Scientific and Statistical Computing* **9**, 862–874.

Gilbert, J. R. and Schreiber, R. (1992), 'Highly parallel sparse Cholesky factorization', *SIAM J. Scientific and Statistical Computing* **13**, 1151–1172.

Gilbert, J. R. and Zmijewski, E. (1987), 'A parallel graph partitioning algorithm for a message-passing multiprocessor', *Int J. Parallel Programming* **16**, 427–449.

Gilbert, J. R., Miller, G. L. and Teng, S.-H. (1995), Geometric mesh partitioning: Implementation and experiments, *in* 'Proceedings of the 9th International Parallel Processing Symposium', IEEE, pp. 418–427.

Gilbert, J. R., Moler, C. and Schreiber, R. (1992), 'Sparse matrices in MATLAB: Design and implementation', *SIAM J. Matrix Analysis and Applications* **13**(1), 333–356.

Gilbert, J. R., Ng, E. G. and Peyton, B. W. (1993), Separators and structure prediction in sparse orthogonal factorization, Technical Report CSL-93-15, Palo Alto Research Center, Xerox Corporation, California. To appear in *Linear Algebra and Its Applications*.

Golub, G. H. and van der Vorst, H. A. (1996), Closer to the solution: Iterative linear solvers, *in* I. S. Duff and G. A. Watson, eds, 'State of the Art in Numerical Analysis - 1996', Oxford University Press, Oxford.

Gould, N. I. M. and Scott, J. A. (1995), On approximate-inverse preconditioners, Technical Report RAL-TR-95-026, Rutherford Appleton Laboratory. To appear in *SIAM J. Scientific Computing*.

Grcar, J. F. (1990), Matrix stretching for linear equations, Technical Report SAND90-8723, Sandia National Laboratories, Albuquerque.

Grote, M. and Simon, H. (1993), Parallel preconditioning and approximate inverses on the connection machine, *in* R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold and D. A. Reed, eds, 'Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing', SIAM Press, Philadelphia, PA, pp. 519–523.

Gupta, A., Karypis, G. and Kumar, V. (1994), Highly scalable parallel algorithms for sparse matrix factorization, Technical Report TR-94-63, Department of Computer Science, University of Minnesota.

Gupta, S. and Rothberg, E. (1994), DME: A distributed matrix environment, *in* IEEE, ed., 'Proceedings of SHPCC '94, Scalable High-Performance Computing Conference. May 23-25, 1994, Knoxville, Tennessee', IEEE Computer Society Press, Los Alamitos, California, pp. 629–636.

Gustafsson, I. (1979), Stability and rate of convergence of modified incomplete Cholesky factorization methods, PhD thesis, Chalmers University, Göteborg, Sweden.

Gustafsson, I. and Lindskog, G. (1986), 'A preconditioning technique based on element matrix factorizations', *Comput. Methods Appl. Mech. Eng.* **55**, 201–220.

Gustafsson, I. and Lindskog, G. (1995), 'Completely parallelizable preconditioning methods', *Numerical Linear Algebra with Applications* **2**, 447–465.

Hackbusch, W. (1985), *Multigrid Methods and Applications*, Vol. 4 of *Computational Mathematics*, Springer–Verlag, Berlin.

Hansen, P. C., Ostromsky, T. and Zlatev, Z. (1994), Two enhancements in a partitioned sparse solver, *in* J. J. Dongarra and J. Waśniewski, eds, 'Parallel Scientific Computing. Proceedings of the PARA94 Conference, Copenhagen 1994', Lecture Notes in Mathematics 879, Springer, Berlin, pp. 296–303.

Heath, M. T. and Raghavan, P. (1994), Performance of a fully parallel sparse solver, *in* IEEE, ed., 'Proceedings of SHPCC '94, Scalable High-Performance Computing Conference. May 23-25, 1994, Knoxville, Tennessee', IEEE Computer Society Press, Los Alamitos, California, pp. 334–341.

Heath, M. T., Ng, E. G. Y. and Peyton, B. W. (1991), 'Parallel algorithms for sparse linear systems', *SIAM Review* **33**, 420–460.

Hendrickson, B. and Leland, R. (1993), A multilevel algorithm for partitioning graphs, Technical Report SAND93-1301, Sandia National Laboratories, Albuquerque.

Hendrickson, B. and Leland, R. (1994), The CHACO User's Guide. Version 2.0., Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque.

Hendrickson, B. and Rothberg, E. (1996), Improving the runtime and quality of nested dissection ordering, Technical Report SAND96-0868J, Sandia National Laboratories, Albuquerque.

Heroux, M., Vu, P. and Yang, C. (1991), 'A parallel preconditioned conjugate gradient package for solving sparse linear systems on a CRAY Y-MP', *Applied Numerical Math.* **8**, 93–115.

Hoffman, A. J., Martin, M. S. and Rose, D. J. (1973), 'Complexity bounds for regular finite difference and finite element grids', *SIAM J. Numerical Analysis* **10**, 364–369.

HSL (1996), *Harwell Subroutine Library. A Catalogue of Subroutines (Release 12)*, AEA Technology, Harwell Laboratory, Oxfordshire, England. For information concerning HSL contact: Dr Scott Roberts, AEA Technology, 552 Harwell, Didcot, Oxon OX11 0RA, England (tel: +44-1235-434714, fax: +44-1235-434136, email: Scott.Roberts@aeat.co.uk).

Huckle, T. and Grote, M. (1994), A new approach to parallel preconditioning with sparse approximate inverses, Technical Report SCCM-94-03, School of Engineering, Stanford University, California. To appear in *SIAM J. Scientific Computing*.

Jennings, A. and Ajiz, M. A. (1984), 'Incomplete methods for solving $A'Ax = b$', *SIAM J. Scientific and Statistical Computing* **5**, 978–987.

Jennings, A. and Malik, G. M. (1977), 'Partial elimination', *J. Institute of Mathematics and its Applications* **20**, 307–316.

Johnson, O. G., Micchelli, C. A. and Paul, G. (1983), 'Polynomial preconditioning for conjugate gradient calculations', *SIAM J. Numerical Analysis* **20**, 362–375.

Johnson, T. and Davis, T. A. (1992), 'Parallel buddy memory management', *Parallel Processing Letters* **2**(4), 391–398.

Jones, M. T. and Plassmann, P. E. (1994), The efficient parallel iterative solution of large sparse linear systems, *in* A. George, J. R. Gilbert and J. W. H. Liu, eds, 'Graph Theory and Sparse Matrix Computations', IMA Vol **56**, Springer-Verlag, pp. 229–245.

Kågström, B., Ling, P. and Van Loan, C. (1993), Portable high performance GEMM-based Level-3 BLAS, *in* R. F. Sincovec, D. E. Keyes, M. R. Leuze, L. R. Petzold and D. A. Reed, eds, 'Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing', SIAM, Philadelphia, PA, pp. 339–345.

Karypis, G. and Kumar, V. (1995*a*), A fast and high quality multilevel scheme for partitioning irregular graphs, Technical Report TR-95-035, Department of Computer Science, University of Minnesota.

Karypis, G. and Kumar, V. (1995*b*), METIS: unstructured graph partitioning and sparse matrix ordering system, Technical report, Department of Computer Science, University of Minnesota.

Karypis, G. and Kumar, V. (1995*c*), Parallel multilevel graph partitioning, Technical Report TR-95-036, Department of Computer Science, University of Minnesota.

Karypis, G., Gupta, A. and Kumar, V. (1994), A parallel formulation of interior point algorithms, Technical Report TR-94-020, Department of Computer Science, University of Minnesota.

Kernighan, B. and Lin, S. (1970), 'An efficient heuristic procedure for partitioning graphs', *Bell System Technical J.* **49**(2), 291–307.

Kolotilina, L. Y. and Yeremin, A. Y. (1993), 'Factorized sparse approximate inverse preconditioning I: Theory', *SIAM J. Matrix Analysis and Applications* **14**, 45–58.

Koster, J. and Bisseling, R. H. (1994), 'Parallel sparse LU decomposition on a distributed-memory multiprocessor'. Submitted to *SIAM J. Scientific Computing*.

Lipton, R. J., Rose, D. J. and Tarjan, R. E. (1979), 'Generalized nested dissection', *SIAM J. Numerical Analysis* **16**, 346–358.

Liu, J. W. H. (1985), 'Modification of the minimum degree algorithm by multiple elimination', *ACM Trans. Math. Softw.* **11**(2), 141–153.

Liu, J. W. H. (1987), 'On the storage requirement in the out-of-core multifrontal method for sparse factorization', *ACM Trans. Math. Softw.* **12**, 249–264.

Liu, J. W. H. (1989), 'Reordering sparse matrices for parallel elimination', *Parallel Computing* **11**, 73–91.

Liu, J. W. H. (1990), 'The role of elimination trees in sparse factorization', *SIAM J. Matrix Analysis and Applications* **11**, 134–172.

Liu, J. W. H. (1992), 'The multifrontal method for sparse matrix solution: Theory and Practice', *SIAM Review* **34**, 82–109.

Lu, S.-M. and Barlow, J. L. (1994), Multifrontal computation with the orthogonal factors of sparse matrices, Technical Report CSE-94-050, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA.

Lustig, I. J. and Rothberg, E. (1996), 'Gigaflops in linear programming', *Operations Research Letters* **18**(4), 157–165.

Manne, F. and Hafsteinsson, H. (1995), 'Efficient sparse Cholesky factorization on a massively parallel SIMD computer', *SIAM J. Scientific Computing* **16**(4), 934–950.

Manteuffel, T. A. (1980), 'An incomplete factorization technique for positive-definite linear systems', *Mathematics of Computation* **34**, 473–498.

Markowitz, H. M. (1957), 'The elimination form of the inverse and its application to linear programming', *Management Science* **3**, 255–269.

Matstoms, P. (1994), 'Sparse QR factorization in MATLAB', *ACM Trans. Math. Softw.* **20**, 136–159.

Matstoms, P. (1995), 'Parallel sparse QR factorization on shared memory architectures', *Parallel Computing* **21**, 473–486.

Meijerink, J. A. and van der Vorst, H. A. (1977), 'An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix', *Mathematics of Computation* **31**(137), 148–162.

Meurant, G. (1992), 'A review on the inverse of symmetric tridiagonal and block tridiagonal matrices', *SIAM J. Matrix Analysis and Applications* **13**, 707–728.

Munksgaard, N. (1980), 'Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients', *ACM Trans. Math. Softw.* **6**, 206–219.

Ng, E. G. and Peyton, B. W. (1993*a*), 'Block sparse Cholesky algorithms on advanced uniprocessor computers', *SIAM J. Scientific Computing* **14**, 1034–1056.

Ng, E. G. and Peyton, B. W. (1993*b*), 'A supernodal Cholesky factorization algorithm for shared-memory multiprocessors', *SIAM J. Scientific Computing* **14**, 761–769.

Peyton, B. W., Pothen, A. and Yuan, X. (1992), Partitioning a chordal graph into transitive subgraphs for parallel sparse triangular solution, Technical Report ORNL/TM-12270, Engineering Physics and Mathematics Division, Oak Ridge National Laboratory, Tennessee.

Pierce, D. J. and Lewis, J. G. (1995), Sparse multifrontal rank revealing QR factorization, Technical Report MEA-TR-193-Revised, Boeing Information and Support Services, Seattle, WA.

Pothen, A. and Fan, C. (1990), 'Computing the block triangular form of a sparse matrix', *ACM Trans. Math. Softw.* **16**(4), 303–324.

Pothen, A. and Sun, C. (1993), 'A mapping algorithm for parallel sparse Cholesky factorization', *SIAM J. Scientific Computing* **14**(5), 1253–1257. Timely Communication.

Pothen, A., Simon, H. D. and Liou, K. P. (1990), 'Partitioning sparse matrices with eigenvectors of graphs', *SIAM J. Matrix Analysis and Applications* **11**(3), 430–452.

Pothen, A., Simon, H. D., Wang, L. and Barnard, S. (1992), Towards a fast implementation of spectral nested dissection, *in* 'Proceedings of Supercomputing '92', ACM Press, New York, NY, pp. 42–51.

Raghavan, P. (1995*a*), 'Distributed sparse Gaussian elimination and orthogonal factorization', *SIAM J. Scientific Computing* **16**(6), 1462–1477.

Raghavan, P. (1995*b*), Efficient parallel sparse triangular solution with selective inversion, Technical Report CS-95-314, Department of Computer Science, University of Tennessee, Knoxville, Tennessee.

Raghavan, P. (1995*c*), Parallel ordering using edge contraction, Technical Report CS-95-293, Department of Computer Science, University of Tennessee, Knoxville, Tennessee. Submitted to *Parallel Computing*.

Reid, J. K. (1987), Sparse matrices, *in* A. Iserles and M. J. D. Powell, eds, 'The State of the Art in Numerical Analysis', Oxford University Press, Oxford, pp. 59–85.

Rose, D. J. (1973), A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations, *in* R. C. Read, ed., 'Graph Theory and Computing', New York: Academic Press, pp. 183–217.

Rothberg, E. (1996*a*), Exploring the tradeoff between imbalance and separator size in nested dissection ordering, Technical Report Unnumbered, Silicon Graphics Inc.

Rothberg, E. (1996*b*), Ordering sparse matrices using approximate minimum local fill, Technical Report Unnumbered, Silicon Graphics Inc.

Rothberg, E. (1996*c*), 'Performance of panel and block approaches to sparse Cholesky factorization on the iPSC/860 and Paragon multicomputers', *SIAM J. Scientific Computing* **17**(3), 699–713.

Rothberg, E. and Gupta, A. (1991), An evaluation of left-looking, right-looking and multifrontal approaches to sparse Cholesky factorization on hierarchical-memory machines, Technical Report STAN-CS-91-1377, Department of Computer Science, Stanford University.

Rothberg, E. and Gupta, A. (1994), 'An efficient block-oriented approach to parallel sparse Cholesky factorization', *SIAM J. Scientific Computing* **15**(6), 1413–1439.

Rothberg, E. and Hendrickson, B. (1996), Sparse matrix ordering methods for interior point linear programming, Technical Report 96-0475J, SANDIA National Laboratory.

Rothberg, E. and Schreiber, R. (1994), Improved load distribution in parallel sparse Cholesky factorization, Technical Report 94-13, Research Institute for Advanced Computer Science.

Saad, Y. (1985), 'Practical use of polynomial preconditionings for the conjugate gradient method', *SIAM J. Scientific and Statistical Computing* **6**, 865–881.

Saad, Y. (1988*a*), 'Preconditioning techniques for nonsymmetric and indefinite linear systems', *J. Comput. Appl. Math.* **24**, 89–105.

Saad, Y. (1988*b*), Preconditioning techniques for nonsymmetric and indefinite linear systems, Technical Report 792, CSRD, University of Illinois, Urbana, Illinois, USA.

Saad, Y. (1994*a*), 'ILUT: a dual threshold incomplete LU factorization', *Numerical Linear Algebra with Applications* **1**(4), 387–402.

Saad, Y. (1994*b*), SPARSKIT: a basic tool kit for sparse matrix computations. Version 2, Technical report, Computer Science Department, University of Minnesota.

Saunders, M. A. (1994), 'Major Cholesky would feel proud', *ORSA J. Computing* **6**(1), 23–27.

Schreiber, R. (1982), 'A new implementation of sparse Gaussian elimination', *ACM Trans. Math. Softw.* **8**, 256–276.

Schreiber, R. (1993), Scalability of sparse direct solvers, *in* A. George, J. R. Gilbert and J. W. H. Liu, eds, 'Graph Theory and Sparse Matrix Computation', The IMA Volumes in Mathematics and its Applications, Volume 56, Springer-Verlag, New York, pp. 191–209.

Shanno, D. F. and Simantiraki, E. M. (1996), Interior point methods for linear and nonlinear programming, *in* I. S. Duff and G. A. Watson, eds, 'State of the Art in Numerical Analysis - 1996', Oxford University Press, Oxford.

Sherman, A. H. (1978), 'Algorithm 533. NSPIV, A Fortran subroutine for sparse Gaussian elimination with partial pivoting', *ACM Trans. Math. Softw.* **4**, 391–398.

Speelpenning, B. (1978), The generalized element method, Technical Report Technical Report UIUCDCS-R-78-946, Dept. of Computer Science, Univ. of Illinois, Urbana, IL.

Sun, C. (1992*a*), Efficient parallel solutions of large sparse SPD systems on distributed-memory multiprocessors, Technical Report CTC92TR102, Advanced Computing Research Institute, Cornell University, Ithaca, NY.

Sun, C. (1992*b*), A package for solving sparse symmetric positive definite systems on distributed-memory multiprocessors, Technical Report CTC92TR114, Advanced Computing Research Institute, Cornell University, Ithaca, NY.

Sun, C. (1995), Dealing with dense rows in the solution of sparse linear least squares problems, Technical Report CTC95TR227, Advanced Computing Research Institute, Cornell University, Ithaca, NY.

Tinney, W. F. and Walker, J. W. (1967), 'Direct solutions of sparse network equations by optimally ordered triangular factorization', *Proc. of the IEEE* **55**, 1801–1809.

Tismenetsky, M. (1991), 'A new preconditioning technique for solving large sparse linear systems', *Linear Algebra and its Applications* **154-156**, 331–353.

van der Stappen, A. F., Bisseling, R. H. and van de Vorst, J. G. G. (1993), 'Parallel sparse LU decomposition on a mesh network of transputers', *SIAM J. Matrix Analysis and Applications* **14**, 853–879.

van der Vorst, H. A. (1982), 'A vectorizable variant of some ICCG methods', *SIAM J. Scientific and Statistical Computing* **3**, 350–356.

van der Vorst, H. A. (1989), 'High performance preconditioning', *SIAM J. Scientific and Statistical Computing* **10**, 1174–1185.

van Duin, A. C. N., Hansen, P. C., Ostromsky, T., Wijshoff, H. A. G. and Zlatev, Z. (1995), 'Improving the numerical stability and the performance of a parallel sparse solver', *Computers Math. Applic.* **30**, 81–96.

Vanderbei, R. J. (1991), 'Splitting dense columns in sparse linear systems', *Linear Algebra and its Applications* **152**, 107–117.

Wang, X. (1993), Incomplete factorization preconditioning for least squares problems, PhD thesis, Department of Mathematics, University of Illinois, Urbana, Illinois, USA.

Wang, X., Gallivan, K. A. and Bramley, R. (1996), 'CIMGS: an incomplete orthogonal factorization preconditioner', *SIAM J. Scientific Computing* **17**, xxx–xxx.

Wesseling, P. (1992), *An Introduction to Multigrid Methods*, John Wiley & Sons, Chichester.

Whaley, R. C. (1994), Lapack working note 73 : Basic Linear Algebra Communication Subprograms: analysis and implementation across multiple parallel architectures, Technical Report CS-94-234, Computer Science Department, University of Tennessee, Knoxville, Tennessee.

Wright, M. H. (1992), Interior methods for constrained optimization, *in* A. Iserles, ed., 'Acta Numerica 1992', Cambridge University Press, pp. 341–407.

Zlatev, Z., Waśniewski, J. and Schaumburg, K. (1981), *Y12M - Solution of Large and Sparse Systems of Linear Algebraic Equations*, Vol. 121 of *Lecture Notes in Computer Science*, Springer-Verlag, New York.

Zlatev, Z., Waśniewski, J. and Schaumburg, K. (1993), Introduction to PARASPAR. solution of large and sparse systems of linear algebraic equations, specialised for parallel computers with shared memory, Technical Report 93-02, Tech Univ Denmark, Lyngby.

Zlatev, Z., Waśniewski, J., Hansen, P. C. and Ostromsky, T. (1995), PARASPAR: a package for the solution of large linear algebraic equations on parallel computers with shared memory, Technical Report 95-10, Tech Univ Denmark, Lyngby.

Zmijewski, E. (1987), Sparse Cholesky Factorization on a Multiprocessor, PhD thesis, Cornell University.

Zmijewski, E. and Gilbert, J. R. (1988), 'A parallel algorithm for sparse symbolic Cholesky factorization on a multiprocessor', *Parallel Computing* **7**, 199–210.