Technical Report
RAL-TR-2004-033
(Revised)

Combining direct and iterative methods for the solution of large systems in different application areas

Iain S. Duff

May 11, 2007

Council for the Central Laboratory of the Research Councils

Library and Information Services
CCLRC Rutherford Appleton Laboratory
Chilton Didcot
Oxfordshire OX11 0QX
UK
Tel: +44 (0)1235 445384
Fax: +44(0)1235 446403
Email: library@rl.ac.uk

CCLRC reports are available online at:
http://www.clrc.ac.uk/Activity/ACTIVITY=Publications;SECTION=225;

ISSN 1358-6254

RAL-TR-2004-033
(Revised)

Combining direct and iterative methods for the solution of large systems in different application areas[1]

Iain S. Duff[2]

ABSTRACT

We are concerned with the solution of sets of linear equations where the matrices are of very high order. We first discuss sparse direct methods and consider the size of problems that they can currently solve. We then discuss the limitations of such methods, where current research is going in moving these limitations, and how far we might expect to go with direct solvers in the near future.

This leads us to the conclusion that very large systems, by which we mean three dimensional problems in more than a million degrees of freedom, require the assistance of iterative methods in their solution. However, even the strongest advocates and developers of iterative methods recognize their limitations when solving difficult problems, that is problems that are poorly conditioned and/or very unstructured. It is now universally accepted that sophisticated preconditioners must be used in such instances.

A very standard and sometimes successful class of preconditioners are based on incomplete factorizations or sparse approximate inverses, but we very much want to exploit the powerful software that we have developed for sparse direct methods over a period of more than thirty years. We thus discuss various ways in which a symbiotic relationship can be developed between direct and iterative methods in order to solve problems that would be intractable for one class of methods alone. In these approaches, we will use a direct factorization on a "nearby" problem or on a subproblem.

We then look at examples using this paradigm in four quite different application areas; the first solves a subproblem and the others a nearby problem using a direct method.

Keywords: sparse direct methods, iterative methods, domain decomposition, constrained optimization, nonlinear water waves, electromagnetics.

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

---

[1]Current reports available by anonymous ftp to `ftp.numerical.rl.ac.uk` in directory pub/reports. This report is in file duRAL2004033.ps.gz. The report also available through the URL `www.numerical.rl.ac.uk/reports/reports.html`. An earlier version is available as the CERFACS Technical Report TR/PA/04/128. This revised version has been submitted for publication in the Proceedings of the ISIAM Conference held in Delhi from 4 to 6 December, 2004.

[2]i.s.duff@rl.ac.uk. This work was supported by the EPSRC Grant GR/S42170.

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

May 11, 2007

Contents

1 Introduction

We are concerned with the solution of the set of linear equations

$$Ax = b, \tag{1.1}$$

where the coefficient matrix A is of very high dimension, say in excess of 10^6. In most cases, A will be sparse; that is most of its entries are zero. Although in one of our applications the matrix is dense, even there we use sparse techniques and a sparse direct code. We thus discuss what we mean by sparse matrices in Section 2 and illustrate some of the many applications that give rise to such matrices.

We discuss sparse direct codes in Section 3. We then emphasize that sparse direct codes can be used to solve really quite large problems with test cases in the order of a million degrees of freedom becoming quite common. We summarize the current leading edge of the purely direct approach in Section 4. In this section, we also indicate the limitations and indicate why other techniques are necessary.

An alternative approach to solving systems of linear equations is to use iterative methods. We consider such techniques in Section 5 where we also indicate current ways of accelerating the convergence of such methods.

We then examine approaches that combine direct and iterative methods to solve very large sparse problems. We believe that this is really the only way to solve systems when the number of degrees of freedom exceeds several million. We illustrate this approach by reference to earlier work by this author and others. The main novelty of this paper is that we collect and identify applications where an off-the-shelf sparse direct solver is used so that advantage can be taken of the increasing number of readily available sparse direct codes.

In Section 6 we describe two ways in which direct solvers can be used and in the remaining sections give examples of each of these. We discuss domain decomposition in Section 7, an optimization application in Section 8, an application to nonlinear water waves in Section 9, and the preconditioning of dense systems from electromagnetics applications in Section 10.

2 Sparse matrices

Sparse systems arise in very many application areas. We list just a few such areas in Table 2.1.

This table, reproduced from Duff, Grimes and Lewis (1989), shows the number of matrices from each discipline present in the Harwell-Boeing Sparse Matrix Collection. This was the main source of sparse test problems for many years but there are now several more recent projects that give good access to a wide range of sparse matrices. The Harwell-Boeing collection morphed into the Rutherford-Boeing Sparse Matrix Collection (Duff, Grimes and Lewis 1997*b*) that is available

acoustic scattering	4	demography	3	network flow	1
air traffic control	1	economics	11	numerical analysis	4
astrophysics	2	electric power	18	oceanography	4
biochemical	2	electrical engineering	1	petroleum engineering	19
chemical engineering	16	finite elements	50	reactor modeling	3
chemical kinetics	14	fluid flow	6	statistics	1
circuit physics	1	laser optics	1	structural engineering	95
computer simulation	7	linear programming	16	survey data	11

Table 2.1: **A list of some application areas for sparse matrices**

from CERFACS (`www.cerfacs.fr/algor/Softs/index.html`). The Matrix Market (`http://math.nist.gov/MatrixMarket`) has a well organized collection of the Harwell-Boeing set and several other sparse matrices, and a very comprehensive collection is currently available from Tim Davis at the University of Florida (Davis 2004).

The Grid-TLSE Project, centred in Toulouse, is designing a one-stop shop for sparse direct methods and will inter alia allow users to access a very wide range of sparse problems from several sources (`http://www.enseeiht.fr/lima/tlse`). There are several recent reports on this project (Buvry, Daydé, Pantel and Puglisi 2004, Amestoy, Duff, Giraud, L'Excellent and Puglisi 2004).

The definition of a large sparse matrix is a matter for some debate. Suffice it to say that we regard a matrix as large if it cannot be factorized efficiently using a code for general linear systems from a standard package for dense systems, such as LAPACK (Anderson, Bai, Bischof, Demmel, Dongarra, Du Croz, Greenbaum, Hammarling, McKenney, Ostrouchov and Sorensen 1995). The order of a matrix that is considered large is thus a function of time depending on the development of both dense and sparse codes and advances in computer architecture. We show in Table 2.2 the order of general unstructured matrices which sparse methods have been used to solve as a function of the date. I think this alone serves to illustrate some of the advances in sparse solution methods over the last 35 years.

The matrix is sparse if the presence of zeros within the matrix enables us to exploit this fact and obtain an economical solution.

There are two main classes of techniques for solving (1.1): iterative methods and direct methods. In a direct method, we use a factorized representation and solve the system using these factors in a predetermined amount of memory and time, usually to a high degree of accuracy. In iterative methods, we construct a sequence of approximations to the solution, often the "best" approximation in subspaces of increasing dimension. The work is generally low per iteration but the number of iterations is usually not known *a priori* and may be high, particularly if an accurate solution is required. We discuss direct

date	order
1970	200
1975	1000
1980	10000
1985	100000
1990	250000
1995	1000000
2000	2000000
2005	5000000

Table 2.2: **Order of general sparse matrices solved by direct methods as a function of date**

methods of solution in Section 3 and iterative techniques in Section 5 and discuss the combination of these approaches in Section 6.

The best references for a general overview of work on sparse matrices come from the proceedings of conferences (Willoughby 1969, Reid 1971, Rose and Willoughby 1972, Himmelblau 1973, Bunch and Rose 1976, Barker 1977, Duff and Stewart 1979, Duff 1981, Evans 1985, Duff, Gould, Douglas and Giraud 1997*a*).

3 Direct methods

Direct methods use a factorization of the coefficient matrix to facilitate the solution. The most common factorization for unsymmetric systems is an LU factorization where the matrix A (or rather a permutation of it) is expressed as the product of a lower triangular matrix L and an upper triangular matrix U. Thus

$$PAQ = LU, \tag{3.1}$$

where P and Q are permutation matrices. This factorization can then be used to solve the system (1.1) through the two steps:

$$Ly = Pb, \tag{3.2}$$

and

$$Uz = y, \tag{3.3}$$

whence the solution x is just a permutation of the vector z, viz.

$$x = Qz. \tag{3.4}$$

This use of LU factorization to solve systems of equations is usually termed *Gaussian elimination* and indeed the terms are often used synonymously. Another way of viewing Gaussian elimination is as a multistage algorithm which processes the equations in some order. At each stage, a variable is chosen in the equation and is eliminated from all

subsequent equations by subtracting an appropriate multiple of that equation from all subsequent ones. The coefficient of the chosen variable is called the *pivot* in Gaussian elimination and the multiple of the pivot row or equation is called the *multiplier*. Clearly, there must be some reordering performed (called *pivoting*) if a pivot is zero but equally pivoting will normally be necessary if the pivot is very small (in fact if the multipliers are large) relative to other entries since then original information could be lost (from adding very large numbers to relatively very small numbers in finite-precision arithmetic) and we could solve a problem quite different from that originally intended. A good reference for studying this in more detail is the book by Golub and Van Loan (1996) but a different perspective is given by several other texts (Stewart 1973, Meurant 1999, Watkins 2002).

If the matrix A is symmetric positive definite, it is normal to use the factorization

$$PAP^T = LL^T. \tag{3.5}$$

The factorization (3.5) is called a Cholesky factorization. For more general symmetric matrices, the factorization

$$PAP^T = LDL^T, \tag{3.6}$$

is more appropriate. For a stable decomposition in the indefinite case, the matrix D is block diagonal with blocks of order 1 or 2, and L is unit lower triangular.

In the sparse case, pivoting is also required to preserve sparsity in the factors. For example, if the matrix A is an arrowhead matrix as shown in Figure 3.1, then selecting entry $(1, 1)$ as pivot will give dense triangular factors while choosing pivots from the diagonal in any order with entry $(1, 1)$ chosen last will give no *fill-in* (that is, there will be no entries in positions that were not entries in the original matrix). Of course, such a choice could be bad numerically. The reconciliation of these possibly conflicting goals of maintaining sparsity while preserving numerical stability has been a major topic of research for many years. See, for example, the book by Duff, Erisman and Reid (1986).

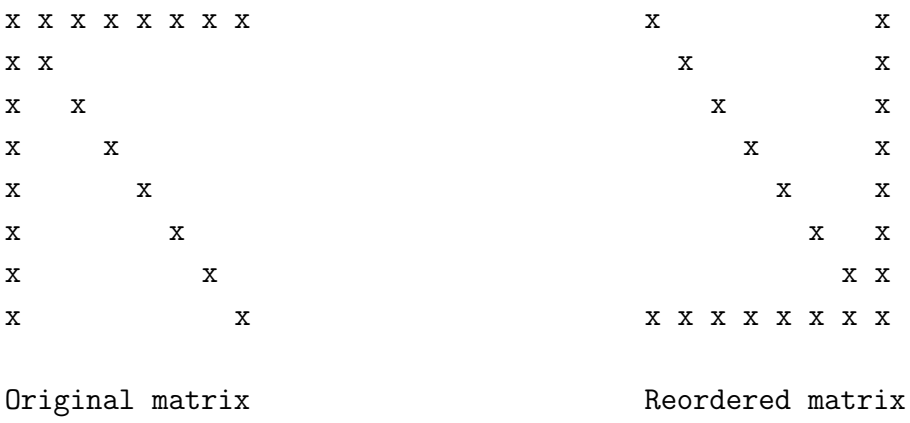

Figure 3.1: **Original and reordered matrix**

Thus, a major concern when the matrix A is sparse is that the factors L and U will

generally be denser than the original A. This is evident if we look at the elementary operations used in Gaussian elimination.

After k steps of elimination on a matrix of order n, the reduced matrix is the lower $n-k$ by $n-k$ matrix modified from the original matrix according to the first k pivots steps. If we denote the entries of the original matrix by $a_{ij}^{(1)}$ and those of the reduced matrix after k stages of Gaussian elimination by $a_{ij}^{(k+1)}$, then fill-in is caused in Gaussian elimination if, in the basic operation

$$a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - a_{ik}^{(k)}[a_{kk}^{(k)}]^{-1}a_{kj}^{(k)}, \tag{3.7}$$

the entry in location (i,j) of the original A was zero. The ordering of the rows and columns of A can be important in preserving sparsity in the factors. Figure 3.1 was an example of a case where ordering the rows and columns to preserve sparsity in Gaussian elimination is extremely effective.

We refer the reader to Duff et al. (1986) for a more detailed presentation on sparse direct methods. The book by George and Liu (1981) has a good discussion for symmetric matrices and there are several other texts that may be of interest (Tewarson 1973, Jennings 1977, Østerby and Zlatev 1983, Pissanetzky 1984, Zlatev 1991).

4 Extent and limitations of sparse direct codes

Modern codes for direct methods for solving (1.1) can be remarkably efficient, the main limitation being the storage of L and U. Although the algorithms for matrix factorization attempt to maintain sparsity in these factors, they are often far denser than A. For example, if a nested dissection ordering is used on the matrix from the discretization of a simple elliptic operator on a square (cubic) grid of side k, the storage for the factors will be $\mathcal{O}(k^2 \log k)$ ($\mathcal{O}(k^4)$). Furthermore, it can be proved that this is an asymptotic lower bound for a direct method using Gaussian elimination.

The power of direct methods for solving sparse linear systems is not always appreciated, particularly by people working in the computational solution of partial differential equations. It is important to emphasize that systems of order more than one million are now solved almost routinely and matrices of this order are now included in benchmark tests. Although the conventional wisdom is that direct methods are more suited for two-dimensional discretizations, many of the larger standard test cases (for example from the `PARASOL` test set, `http://www.parallab.uib.no/parasol/`) are from three-dimensional models.

There has been much recent work on extending the range of problems that can be solved by direct methods. Most of this has been directed at exploiting parallelism and there are several codes available for doing this. It is important to note that nearly all modern sparse direct codes use dense matrix kernels at their inner loop and can thus exploit modern machine architectures very efficiently. A typical rule of thumb is that sparse codes will run at about half the rate of dense matrix-matrix multiply for large scale

realistic factorizations. Other recent work on sparse direct solvers attempts to address the most severe limitation of direct methods by holding the factors out-of-core and even performing some of the factorization operations out-of-core.

However, for the rest of this presentation, we will look at another way of extending the scope of sparse direct methods so that we can solve systems one or two orders of magnitude larger. In addition, we will do this in a way that capitalizes on the increasing body of software for sparse direct methods by using a sparse direct code with very little or no modification. We will call this approach a hybrid direct-iterative approach. Another example of this can be found in many implementations of the multigrid approach where often the coarse grid problem (that could be considered a "nearby" problem) is solved by a direct method within an overall iterative approach.

Before continuing we will first very briefly discuss iterative methods or a subset of them.

5 Iterative methods

In contrast to direct methods, iterative methods do not normally modify the matrix and do not form any representation of the inverse. Furthermore, most iterative methods do not require the matrix to be represented explicitly, sometimes it is sufficient to be able to form the product of the matrix with a vector, although this may restrict the preconditioning available (see Section 5.1).

In an iterative method for solving (1.1), we start with a guess for the solution (often just the zero vector) and then successively refine this guess hopefully getting closer to the solution at each stage. The power of most iterative methods lies in the cheapness with which each iteration is performed. The problem is that very many iterations may be needed. This can be reduced by preconditioning the matrix as indicated in Section 5.1, but then the cost of each iteration is increased. There are very many books that are good references for iterative methods. We think that those by Saad (1996) and van der Vorst (2003) are very accessible but there are many books suitable for the more mathematically inclined (Hackbusch 1993, Axelsson 1996, Greenbaum 1997) or that are more software oriented (Barrett, Berry, Chan, Demmel, Donato, Dongarra, Eijkhout, Pozo, Romine and van der Vorst 1993, Kelley 1995) with the former containing significant code fragments.

Iteration techniques such as successive approximation have been around since the first days of scientific computing and the early iterative techniques for solving sparse linear equations were based on such approaches (Gauss-Seidel, SOR etc). While these methods are still used (for example, as smoothers in multigrid techniques), it is true to say that most modern methods for the iterative solution of sparse equations are based on Krylov sequences.

In a Krylov-sequence based method, we compute an approximate solution to our problem from subspaces of increasing dimension usually with some optimality condition over all vectors in the subspace so that the solution will be obtained when the subspace is

of sufficient dimension. The Krylov sequence is defined as

$$\mathcal{K}^k(A; r_0) = \text{span}\{r_0, Ar_0, A^2r_0, ...A^{k-1}r_0\}, \tag{5.1}$$

where $r_0 = b - Ax^{(0)}$ with $x^{(0)}$ the original "guess" at the solution. From the above, the approximate solution at stage k will be such that

$$x^{(k)} \in x^{(0)} + \mathcal{K}^k(A; r_0)$$

and we will require some optimality condition, for example that

$$\|b - Ax^{(k)}\|_2 \tag{5.2}$$

is minimized. One such method, applicable to general unsymmetric systems is `GMRES` (Saad and Schultz 1986). Krylov methods are discussed in detail in a recent book by van der Vorst (2003). The residual at the i-th iteration of an iterative method can be expressed as

$$r^{(i)} = P_i(A)r^{(0)}, \tag{5.3}$$

where P_i is a polynomial such that $P_i(0) = 1$. If we expand $r^{(0)}$ in terms of the eigenvectors of A we see that we want P_i to be small on the eigenvalues of A so that the spectrum of A is crucial in determining how quickly our method converges. For example, if there are many eigenvalues close to zero or if the spectrum is widely distributed, the degree of polynomial will have to be high in order to be small on all eigenvalues and so the number of iterations required will be large.

A major feature of most Krylov based methods for symmetric systems are that they can be implemented using short term recurrences which means that only a few vectors of length n need be kept and the amount of computation at each stage of the iterative process is modest. However, Faber and Manteuffel (1984) have shown that, for general matrices, one must either lose the cheap recurrences or lose the minimization property. Thus for the solution of general unsymmetric systems, the balance between these, added to the range of quantities that can be minimized and the differing norms that can be used, has led to a veritable alphabet soup of methods (Saad 1996), for example GMRES(k), CGS, Bi-CGSTAB(ℓ), TFQMR, FGMRES, GMRESR, ...

5.1 Preconditioning

The key to developing iterative methods for the solution of realistic problems lies in preconditioning, where by this we mean finding a matrix K such that

1. K is an approximation to A.
2. K is cheap to construct and store.
3. $Kx = b$ is much easier to solve than the system (1.1).

We then solve the preconditioned system

$$K^{-1}Ax = K^{-1}b, \tag{5.4}$$

where we have chosen K so that our iterative method converges more quickly when solving equation (5.4) than equation (1.1). Lest this seem too much of a black art (which to some extent it is), if K were chosen as the product of the factors LU from an LU factorization (admittedly violating point 2 above), then the preconditioned matrix $B = K^{-1}A$ would be the identity matrix and any sensible iterative method would converge in a single iteration. From our earlier discussion, we would like the preconditioned matrix B to have a better distribution of eigenvalues than A.

The preconditioning can also be applied as a right-preconditioning AK^{-1} or as a two-sided preconditioning $K_1^{-1}AK_2^{-1}$, when the matrix K can be expressed as a product K_1K_2. Common preconditioners include using the diagonal of A or a partial or incomplete factorization of A. A more recent class of preconditioners, particularly popular because they are highly parallelizable, are the approximate inverse preconditioners. A recent simple discussion of the merits of different forms of preconditioners and their implementation can be found in the book by Dongarra, Duff, Sorensen and van der Vorst (1998).

However, although some of the techniques that we will now describe can be thought of as preconditioning, we will not view them in this way.

6 The use of sparse direct codes with iterative techniques

The essence of the approaches in this paper is that we will use a direct code with little or no modification in combination with an iterative method to effect the solution. Effectively, we will use our direct code either

1. on a subproblem
   or
2. on a "nearby" problem.

A very simple example of this approach might be when we have a coefficient matrix that is block diagonal except for very few outlying entries. If we then ran the direct method on the block diagonal part, it would both be trivially parallel and very efficient if the blocks on the diagonal were not too large. The iterative solution of the overall problem will thus depend on the outliers and we might assume that the number of iterations would be closely related to the number of outliers. This would correspond to nothing other than a block Jacobi preconditioning of the original system.

In the remainder of this paper, we will consider the solution of large sparse equations in four quite different application areas by four different approaches. The first can be considered as solving a subproblem and the remaining three as solving a nearby system. These are:

1. Domain decomposition in semiconductor modelling,
2. Solution of augmented systems in constrained optimization,
3. Application in nonlinear water waves, and
4. Solution of boundary element problems in electromagnetics

and will be considered in the following four sections respectively.

7 Domain decomposition

My first example comes from the solution of highly nonlinear partial differential equations in semiconductor device modelling. This was the subject of recent work between my colleagues, Luc Giraud and Jean-Christophe Rioual, at CERFACS and Americo Marrocco at INRIA on some large scale industrial problems. The partial differential equations are discretized using mixed finite-element methods and the resulting nonlinear equations are solved by Newton's method. The earlier methods for solving the system that were used by INRIA included a direct method based on a skyline solver that was very memory hungry and required about 24 hours execution time on an large HP machine in serial mode on a problem with 30 thousand elements. They also tried Krylov methods preconditioned by ILU but found they were not robust enough for their application.

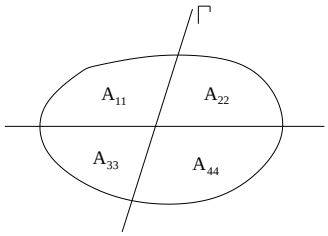

Figure 7.1: Illustration of domain decomposition

The approach used by Giraud, Marrocco and Rioual (2005) to solve this problem is based on domain decomposition where the region over which the problem is solved is split into subregions with artificial boundaries. For example, Figure 7.1 shows the subdivision of a region into four subregions with the artificial boundaries Γ. In this domain decomposition approach, the individual subproblems (four in the case of Figure 7.1) can be discretized separately, leaving a remaining problem on the interface corresponding to the artificial boundary Γ. If the linearization of each subproblem is defined by the matrices $A_{ii},\ i = 1, 4$, then the matrix representation of the subdivided problem in Figure 7.1 is given by

$$\begin{pmatrix} A_{11} & & & & A_{1\Gamma} \\ & A_{22} & & & A_{2\Gamma} \\ & & A_{33} & & A_{3\Gamma} \\ & & & A_{44} & A_{4\Gamma} \\ A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma 3} & A_{\Gamma 4} & A_{\Gamma\Gamma} \end{pmatrix} \tag{7.1}$$

The solution to the system partitioned as in (7.1) can be obtained by first solving for the interface variables corresponding to the border blocks from the equations whose coefficient matrix is the Schur complement matrix

$$A_{\Gamma\Gamma} \; - \; \sum_{i=1,4}^{4} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma}.$$

A solution for the remaining variables is then found by backsubstitution and the solution of the decoupled systems with coefficient matrices $A_{ii},\ i = 1, 4$. One way of implementing this domain decomposition approach is to use a direct method on the A_{ii} subproblems, which is one of the paradigms promoted in this paper, namely using a direct method on a subproblem.

If we expand the local problem and its interface, we obtain the matrix

$$\begin{pmatrix} A_{ii} & A_{i\Gamma} \\ A_{\Gamma i} & A_{\Gamma\Gamma}^{(i)} \end{pmatrix} \tag{7.2}$$

where

- A_{ii} : is the local subproblem,
- $A_{i\Gamma}$: is the boundary of the local problem, and
- $A_{\Gamma\Gamma}^{(i)}$: is the contribution to the stiffness matrix entries from variables on the artificial interface (Γ_i) around the ith subregion.

resulting in a contribution to the Schur complement of

$$S^{(i)} = A_{\Gamma\Gamma}^{(i)} - A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma},$$

called a local Schur (complement).

There are two quite distinct ways of tackling these subproblems that are termed "implicit" and "explicit". In the implicit method, only A_{ii} is factorized (yielding factors L_i and U_i). In this case only the A_{ii} are passed to the direct solver and the local Schur complement $S^{(i)}$ is not computed but can be used to multiply a vector by multiplying the vector by $A_{i\Gamma}$ and then solving for A_{ii} before multiplying the resulting vector by $A_{\Gamma i}$. In the explicit method, the whole of the matrix (7.2) is passed to the direct solver although pivots are only chosen from the A_{ii} block and the $S^{(i)}$ matrix is generated explicitly as a Schur complement of the factorization.

We would thus expect the factorizations for the implicit method to be cheaper but with more work required when using the Schur complements. Giraud et al. (2005) performed some experiments on a regular grid of 400×400 elements divided into 16 subdomains and obtained the results shown in Table 7.1. This means that, for the full solution of the linear system, the explicit method will be faster if more than 5 iterations are required. Since this number of iterations is likely to be exceeded and since preconditioning and scaling for

	Factorization	Matrix-vector
Implicit	10.2	1.60
Explicit	18.4	0.07

Table 7.1: Times in seconds on Origin 2000. Matrix-vector is the cost of computing the product of a vector with the Schur complement.

the interface problem is much simpler if the Schur complement is assembled, Giraud et al. (2005) propose using an explicit method and future results assume this.

For their further experiments, Giraud et al. (2005) use the code MUMPS (Amestoy, Duff, Koster and L'Excellent 2001) as the sparse direct solver. They try several approaches to solve the overall system. These include: running MUMPS on the whole system, using an explicit factorization on the local problems and MUMPS on the interface problem (Schur complement), and using an explicit factorization on the local problems and GMRES, with an additive Schwarz preconditioner (Carvalho, Giraud and Meurant 2001), on the interface problem. On a realistic problem of order 1,214,758 partitioned into 32 subdomains, they obtain the results shown in Table 7.2. We can see that the partitioning induced by

	Newton steps	Simulation time (seconds)
MUMPS on whole system, using AMD	166	3995
MUMPS on whole system, using ND	166	3250
MUMPS on local and on interface	166	2527
MUMPS on local, GMRES on interface	175	1654

Table 7.2: Runs on industrial problem on Origin 2000.

the domain decomposition technique has a benefit even if the whole system is solved by a direct method. Indeed looking at the first three lines of the table we see that the "ordering" induced by the domain decomposition is not only better than an approximate minimum degree ordering but is also better than the nested dissection ordering from MeTiS (METIS_NODEND) that is used by MUMPS. More significantly we see that using a hybrid technique only marginally increases the number of nonlinear iterations but significantly reduces the overall time for the simulation.

The end result, reported by Giraud et al. (2005), is that the the code is far more robust and that the time for the main INRIA production runs was reduced from about 24 hours to 2 minutes, a very substantial reduction by anyone's reckoning.

8 Constrained optimization

The second application that we shall study is the solution of linear systems arising in constrained optimization. Specifically we will look at augmented matrices of the form

$$\begin{pmatrix} H + D & A^T \\ A & 0 \end{pmatrix} \tag{8.1}$$

where H is an approximation of the Hessian, D comes from penalty terms on inequality constraints (note that $d_{ii} = 0$ for unconstrained variables), and A is a matrix of constraints.

Here we will use a direct method on the nearby matrix

$$\begin{pmatrix} B & A^T \\ A & 0 \end{pmatrix} \tag{8.2}$$

and use the factorization of this to precondition the original system.

Duff and Orban have been experimenting with this approach and have used the `MA57` package (Duff 2004) as the direct solver and a conjugate gradient method as the iterative method. Even though the matrix (8.1) is indefinite, the conjugate gradient method is appropriate since the iterations can be constrained to lie in the subspace corresponding to the null space of A (Polyak 1969, Coleman 1994, Gould, Hribar and Nocedal 2001).

They have experimented with a range of values for B and we show some of these in Table 8.1. The problem is `CVXQP3_L` from CUTEr (Gould, Orban and Toint 2003*a*) and the dimensions of the constraint matrix are 7500×10000, with 39984 entries in the (1,1) block and 22497 in the matrix A. We see that the direct method can be used to produce

B	Matrix factorization		Iterations	
	Storage	Time	# its	Time
$H + D$	1,092,370	146.72	1	7.64
I	262,051	0.35	109	25.86
$\mathrm{diag}(H + D)$	262,051	0.35	75	18.81

Table 8.1: Runs for problem `CVXQP3_L` on DEC Alpha workstation.

a very effective preconditioner and that, by adjusting the (1,1) block, the performance of the preconditioner can be improved. The overall time and storage for the solution are significantly less than for a direct method on the original problem. The options for the preconditioner discussed in Table 8.1 are included in the GALAHAD optimization library (Gould, Orban and Toint 2003*b*).

9 Nonlinear water waves

Our next example is taken from a model that is used for the study of the propagation of nonlinear wind-generated waves in harbours and coastlines conducted by Fuhrman and

Bingham (2004). The equations governing this application form a highly coupled set of three 5th-order partial differential equations of the form

$$\begin{bmatrix} \mathcal{A}_{11} - \eta_x \mathcal{B}_{11} & \mathcal{A}_2 - \eta_x \mathcal{B}_{12} & \mathcal{B}_{11} + \eta_x \mathcal{A}_1 \\ \mathcal{A}_2 - \eta_y \mathcal{B}_{11} & \mathcal{A}_{22} - \eta_y \mathcal{B}_{12} & \mathcal{B}_{12} + \eta_y \mathcal{A}_1 \\ \mathcal{A}_{01} + h_x \mathcal{C}_{11} + h_y \mathcal{C}_{21} & \mathcal{A}_{02} + h_x \mathcal{C}_{12} + h_y \mathcal{C}_{22} & \mathcal{B}_0 - h_x \mathcal{C}_{13} - h_y \mathcal{C}_{23} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} U \\ V \\ 0 \end{bmatrix} \tag{9.1}$$

that are discretized in time using a 4-th order, 4-stage Runge-Kutta method and in space using finite differences.

Although the discretized equations can be solved by a direct method (the matrix order can be up to say 100,000 with about 100 entries in each row), it is very expensive since the matrix changes at each time step. However, if we drop the time dependent terms in the local wave elevation η, the matrix of the discretized system is of the form

$$\begin{pmatrix} A_{11} & A_2 & B_{11} \\ A_2 & A_{22} & B_{12} \\ A_{01} + h_x C_{21} & A_{02} + h_x C_{12} + h_y C_{22} & B_0 - h_x C_{13} - h_y C_{23} \end{pmatrix} \tag{9.2}$$

and Fuhrman and Bingham (2004) factorize the matrix (9.2) using the HSL routine `MA41` and use this "linearized" matrix as a preconditioner for GMRES.

For the solution of the Boussinesq-type equations in shallow water on a 33×33 grid (with 3 variables per grid point), they get the results shown in Table 9.1. From this it

Preconditioning	# its per time step	Time
`MA41`	3-12	80.5
ILUT	3-14	61.2
NONE	12-23	78.5

Table 9.1: Shallow water runs. Times in seconds on a Dell P4.

would appear that the preconditioner using `MA41` is not very successful but, if we look at the performance for deep(er) water and more nonlinearity, we see from the results in Table 9.2 that convergence even with an ILUT preconditioner is very slow. Their experiments for even deeper water show that only the `MA41` preconditioning is effective. The times for solving the full problem (7.1) using `MA41` are also shown in this table. From this we see that, although faster than using an ILUT preconditioning, the hybrid approach that we are championing in this paper is very much the best. Such differences become even more pronounced when the depth is increased further or the grid is refined.

10 Electromagnetics

This application differs significantly from the previous ones inasmuch as the coefficient matrices are dense since they are obtained from the boundary element discretization

Preconditioning	# its per time step	Time
MA41	2-9	64.9
ILUT	14-40	309
Direct		269

Table 9.2: Deeper water runs. Times in seconds on a Dell P4.

of problems in electromagnetic scattering. We will solve the resulting set of linear equations using GMRES where we use the fast multipole method to effect the matrix-vector multiplication and obtain a preconditioning matrix through a sparsification of the dense coefficient matrix. There are a wide range of preconditioners described by Carpentieri, Duff and Giraud (2000), one of which (termed SLU) is obtained by factorizing the sparsified matrix using a direct code, thereby following the paradigm of using the direct method on a nearby problem. The direct code used by Carpentieri et al. (2000) is MUMPS (Amestoy et al. 2001).

If we compare the number of iterations required by GMRES(80) on a model problem of a satellite of order 1701, then we have the results shown in Table 10.1 where different sparsifications of the matrix are used so that the resulting preconditioners have the same work to apply per iteration. Most of the names for the preconditioners are self-explanatory; the SPAI code uses a modification of the algorithm of Grote and Huckle (1997) to compute the sparse approximate inverse whereas FROB just computes a sparse approximate inverse of prefixed sparsity.

Preconditioning	# iterations
JACOBI	491
SSOR	301
ILU(0)	474
SPAI	157
FROB	59
SLU	25

Table 10.1: Performance of different preconditioners for the satellite problem.

Alléon, Carpentieri, Duff, Giraud, Martin and Sylvand (2003) have tried to use this approach on more realistic industrial problems and have met with limited success, largely because the number of entries in the factors becomes prohibitive if the SLU preconditioner is good enough to be effective. That is, the nearby problem is too nearby for a sparse factorization to be sensible. However, we show some results on larger problems in Table 10.2 where we see that the SLU approach requires less iterations than FROB, although the fact that FROB is implemented out-of-core means that it is currently our preferred approach for very large problems.

Test case	Order	Entries in millions			GMRES its	
		FROB	SLU	sp(A)	SLU	FROB
AIRCRAFT	94,704	25	49	12.5	366	745
ALMOND	104,793	21	87	10.6	201	233
CETAF	134,775	20	79	9.9	516	617

Table 10.2: Performance of SLU and FROB preconditioners on large test cases.

11 Conclusions

We have shown several examples in very different application areas where we can use a combination of direct and iterative methods to solve really large problems. In all cases an off-the-shelf sparse direct solver has been used enabling it to solve problems of greater complexity than it could have done if run on the original problem.

Acknowledgments

We are very grateful to Luc Giraud for his comments and discussion on Sections 7 and 10, and to David Fuhrman for his comments and discussion on Section 9. We would also like to thank Andy Wathen and Nick Gould for their comments on an early draft of this paper.

References

Alléon, G., Carpentieri, B., Duff, I. S., Giraud, L., Martin, E. and Sylvand, G. (2003), Efficient parallel iterative solvers for the solution of large dense linear systems arising from the boundary element method in electromagnetism, Technical Report TR/PA/03/65, CERFACS, Toulouse, France.

Amestoy, P. R., Duff, I. S., Giraud, L., L'Excellent, J.-Y. and Puglisi, C. (2004), 'GRID-TLSE: A Web Site for Experimenting with Sparse Direct Solvers on a Computational Grid', See `http://www.siam.org/meetings/pp04/`. Conference: SIAM conference on Parallel Processing for Scientific Computing, San Francisco, (USA).

Amestoy, P. R., Duff, I. S., Koster, J. and L'Excellent, J.-Y. (2001), 'A fully asynchronous multifrontal solver using distributed dynamic scheduling', *SIAM J. Matrix Analysis and Applications* **23**(1), 15–41.

Anderson, E., Bai, Z., Bischof, C., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Ostrouchov, S. and Sorensen, D. (1995), *LAPACK Users' Guide, second edition*, SIAM Press.

Axelsson, O. (1996), *Iterative Solution Methods*, Cambridge University Press, Cambridge, England.

Barker, V. A., ed. (1977), *Sparse matrix techniques*, Lecture notes in mathematics, 572, Springer-Verlag.

Barrett, R., Berry, M., Chan, T., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and van der Vorst, H., eds (1993), *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia.

Bunch, J. R. and Rose, D. J., eds (1976), *Sparse matrix computations*, Academic Press.

Buvry, M., Daydé, M., Pantel, M. and Puglisi, C. (2004), 'TLSE project: A grid-based expertise site for sparse matrix solvers', See `http://www.enseeiht.fr/lima/tlse/axgrids04.pdf`. The 2nd European Across Grids Conference, Nicosia (Cyprus).

Carpentieri, B., Duff, I. S. and Giraud, L. (2000), 'Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism', *Numerical Linear Algebra with Applications* **7**(7-8), 667–685.

Carvalho, L. M., Giraud, L. and Meurant, G. (2001), 'Local preconditioners for two-level non-overlapping domain decomposition methods', *Numerical Linear Algebra with Applications* **8**(4), 207–227.

Coleman, T. F. (1994), Linearly constrained optimization and projected preconditioned conjugate gradients, *in* J. Lewis, ed., 'Proceedings of the Fifth SIAM Conference on Applied Linear Algebra', SIAM Press, Philadelphia, PA, pp. 118–122.

Davis, T. A. (2004), 'University of Florida sparse matrix collection, `http://www.cise.ufl.edu/research/sparse/matrices/`'.

Dongarra, J. J., Duff, I. S., Sorensen, D. C. and van der Vorst, H. A. (1998), *Numerical Linear Algebra for High-Performance Computers*, SIAM Press, Philadelphia.

Duff, I. S. (2004), 'MA57 – A code for the solution of sparse symmetric indefinite systems', *ACM Trans. Math. Softw.* **30**(2), 118–144.

Duff, I. S. and Stewart, G. W., eds (1979), *Sparse Matrix Proceedings, 1978*, SIAM, Press, Philadelphia.

Duff, I. S., ed. (1981), *Sparse Matrices and Their Uses: Based on the Proceedings of the IMA Numerical Analysis Group Conference / Organised by the Institute of Mathematics and Its Applications and held at the University of Reading, 9th–11th July, 1980*, Academic Press, London.

Duff, I. S., Erisman, A. M. and Reid, J. K. (1986), *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, England.

Duff, I. S., Gould, N. I. M., Douglas, C. C. and Giraud, L., eds (1997*a*), *Direct methods, linear algebra in optimization, iterative methods. Proceedings from the International Linear Algebra Year Workshops. September 1995 - June 1996.*, Vol. 37 (3), BIT.

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. (1997*b*), The Rutherford-Boeing Sparse Matrix Collection, Technical Report RAL-TR-97-031, Rutherford Appleton Laboratory, Oxfordshire, England. Also Technical Report ISSTECH-97-017 from Boeing Information & Support Services, Seattle and Report TR/PA/97/36 from CERFACS, Toulouse.

Evans, D. J., ed. (1985), *Sparsity and Its Applications*, Cambridge University Press, Cambridge.

Fuhrman, D. R. and Bingham, H. B. (2004), 'Numerical solutions of fully nonlinear and extremely dispersive Boussinesq equations in two horizontal dimensions', *Int. J. Numer. Meth. Fluids* **44**(3), 231–256.

George, A. and Liu, J. W. H. (1981), *Computer Solution of Large Sparse Positive Definite Systems*, Englewood Cliffs, New Jersey: Prentice-Hall.

Giraud, L., Marrocco, A. and Rioual, J.-C. (2005), 'Iterative versus direct parallel substructuring methods in semiconductor device modelling', *Numerical Linear Algebra with Applications* **12**(1), 33–53.

Golub, G. H. and Van Loan, C. F. (1996), *Matrix Computations*, The John Hopkins University Press, Baltimore, Maryland, Third Edition.

Gould, N. I. M., Hribar, M. E. and Nocedal, J. (2001), 'On the solution of equality constrained quadratic programming problems arising in optimization', *SIAM J. Scientific Computing* **23**(4), 1376–1395.

Gould, N. I. M., Orban, D. and Toint, P. L. (2003*a*), 'CUTEr and SifDec: A constrained and unconstrained testing environment, revisited', *ACM Trans. Math. Softw.* **29**(4), 373–394.

Gould, N. I. M., Orban, D. and Toint, P. L. (2003*b*), 'DGALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization', *ACM Trans. Math. Softw.* **29**(4), 353–372.

Greenbaum, A. (1997), *Iterative methods for solving linear systems*, Frontiers in Applied Mathematics, SIAM Press, Philadelphia.

Grote, M. J. and Huckle, T. (1997), 'Parallel preconditioning with sparse approximate inverses', *SIAM J. Scientific Computing* **18**(3), 838–853.

Hackbusch, W. (1993), *Iterative Solution of Large Sparse Systems of Equations*, Applied Mathematical Sciences, Springer–Verlag, Berlin.

Himmelblau, D. M., ed. (1973), *Decomposition of large-scale problems*, North Holland.

Jennings, A. (1977), *Matrix computation for engineers and scientists*, J. Wiley and Sons.

Kelley, C. T. (1995), *Iterative methods for linear and nonlinear equations*, Frontiers in Applied Mathematics, SIAM Press, Philadelphia.

Meurant, G. (1999), *Computer Solution of Large Linear systems*, North-Holland, Elsevier, Amsterdam.

Østerby, O. and Zlatev, Z. (1983), *Direct methods for sparse matrices*, Vol. Lecture notes in computer science 157, Springer-Verlag.

Pissanetzky, S. (1984), *Sparse Matrix Technology*, Academic Press, London.

Polyak, B. T. (1969), 'The conjugate gradient method in extremal problems', *U.S.S.R. Computational Mathematics and Mathematical Physics* **9**, 94–112.

Reid, J. K., ed. (1971), *Large sparse sets of linear equations*, Academic Press.

Rose, D. J. and Willoughby, R. A., eds (1972), *Sparse matrices and their applications*, Plenum Press, New York.

Saad, Y. (1996), *Iterative methods for sparse linear systems*, PWS Publishing, New York, NY.

Saad, Y. and Schultz, M. H. (1986), 'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems.', *SIAM J. Scientific and Statistical Computing* **7**, 856–869.

Stewart, G. W. (1973), *Introduction to Matrix Computations*, Academic Press, New York, NY.

Tewarson, R. P. (1973), *Sparse matrices*, Academic Press.

van der Vorst, H. A. (2003), *Iterative Krylov Methods for Large Linear systems*, Cambridge University Press.

Watkins, D. S. (2002), *Fundamentals of Matrix Computations. Second Edition*, John Wiley & Sons, New York.

Willoughby, R. A., ed. (1969), *Sparse matrix proceedings*, Report RA1(#11707), IBM T. J. Watson Research Centre, P. O. Box 218, Yorktown Heights, NY 10598.

Zlatev, Z. (1991), *Computational Methods for General Sparse Matrices*, Kluwer Academic Publishers.