RAL-TR-96-102

A comparison of frontal software with other sparse direct solvers

Iain S. Duff and Jennifer A. Scott

ABSTRACT

We compare the performance of sparse frontal codes from the Harwell Subroutine Library (HSL) against other HSL sparse direct solvers and consider the effect of ordering on the frontal solver. We study both the case of assembled and unassembled systems for both symmetric positive-definite and unsymmetric matrices. We use problems arising in real engineering or industrial applications in our tests.

Keywords: sparse matrices, frontal solver, direct methods, finite-elements, BLAS.

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

Current reports available by anonymous ftp from matisa.cc.rl.ac.uk (internet 130.246.8.22)
in the directory "pub/reports". This report is in file dsRAL96102.ps.gz.

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

December 23, 1996.

Contents

1 Introduction

The frontal method (Irons 1970, Hood 1976, Duff 1984, Duff and Scott 1993, Duff and Scott 1996*a*) is a technique for the direct solution of linear systems

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1.1}$$

where $\mathbf{A}$ is a large sparse matrix. This approach has the merit of rather simple logic and relatively little data movement and integer overhead. The floating-point arithmetic can be performed using dense linear algebra kernels so that the computational rate, measured in Mflop/s, is high. However, unless the matrix can be ordered so that the front is never very large, frontal methods require many more floating-point operations for factorization than other approaches. It is thus interesting to see how this trade-off works in practical applications, and this is the main theme of this paper.

Although frontal methods can be used to factorize assembled matrices, the power of the method, as we illustrate in Section 5, is more apparent when the matrix $\mathbf{A}$ comprises contributions from the elements of a finite-element discretization. That is,

$$\mathbf{A} = \sum_{l=1}^{m} \mathbf{A}^{(l)}, \tag{1.2}$$

where $\mathbf{A}^{(l)}$ is nonzero only in those rows and columns that correspond to variables in the lth element.

In Section 2, we discuss salient features of the frontal method and show how the computational kernel consists of _GEMM, the Level 3 Basic Linear Algebra Subprogram (BLAS) that implements **dense** matrix-matrix multiplication (Dongarra, Du Croz, Duff and Hammarling 1990). We briefly discuss, in Section 3, the codes that are available in the Harwell Subroutine Library (HSL 1996) for frontal solution and associated computation. We use test matrices from real problems occurring in engineering or industry in the subsequent experiments in our paper and we discuss their origins and characteristics in Section 4. In Section 5 we consider the effect of reordering the system, the preassembly of element problems, and the disassembly of assembled matrices. We compare our frontal codes with other Harwell Subroutine Library codes in Sections 6 and 7, considering symmetric positive-definite and unsymmetric systems respectively. We present some concluding remarks in Section 9.

2 Frontal methods

It is easiest to describe the frontal method by reference to its application to elemental problems where $\mathbf{A}$ is of the form (1.2). If a_{ij} and $a_{ij}^{(l)}$ denote the (i,j)th entry of $\mathbf{A}$ and $\mathbf{A}^{(l)}$, respectively, the basic assembly operation when forming $\mathbf{A}$ is of the form

$$a_{ij} \Leftarrow a_{ij} + a_{ij}^{(l)}. \tag{2.1}$$

It is evident that the basic operation in Gaussian elimination

$$a_{ij} \Leftarrow a_{ij} - a_{ip}[a_{pp}]^{-1}a_{pj} \tag{2.2}$$

may be performed as soon as all the terms in the triple product (2.2) are *fully summed* (that is, are involved in no more sums of the form (2.1)). The assembly and Gaussian elimination processes can therefore be interleaved and the matrix $\mathbf{A}$ is never assembled

explicitly. This allows all intermediate working to be performed in a dense matrix, termed the *frontal matrix*, whose rows and columns correspond to variables that have not yet been eliminated but occur in at least one of the elements that have been assembled.

For non-element problems, the rows of **A** (equations) are added into the frontal matrix one at a time. A variable is regarded as fully summed whenever the equation in which it last appears is assembled. The frontal matrix will, in this case, be rectangular. A full discussion of the equation input can be found in Duff (1984).

We now describe the method for element input in more detail. After the assembly of an element, if all the fully summed variables are permuted to the first rows and columns of the frontal matrix, we can partition the frontal matrix **F** in the form

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} \\ \mathbf{F}_{21} & \mathbf{F}_{22} \end{pmatrix}, \tag{2.3}$$

where $\mathbf{F}_{11}$ is a square matrix of order k and $\mathbf{F}_{22}$ is of order $k_1 \times k_1$. Note that $k+k_1$ is equal to the current size of the frontal matrix, and $k \ll k_1$, in general. The rows and columns of $\mathbf{F}_{11}$, the rows of $\mathbf{F}_{12}$, and the columns of $\mathbf{F}_{21}$ are fully summed; the variables in $\mathbf{F}_{22}$ are not yet fully summed. Pivots may be chosen from anywhere in $\mathbf{F}_{11}$. For symmetric positive-definite systems, they can be taken from the diagonal in order but in the unsymmetric case, pivots must be chosen to satisfy a threshold criteria. This is discussed in Duff (1984). The pivot row and column are permuted to the first row and column of (2.3), row 1 of $\mathbf{F}_{11}$ is scaled by the pivot and columns 2 to k of the permuted frontal matrix are updated by elimination operations. Columns 2 to k of the updated matrix $\mathbf{F}_{11}$ are then searched for the next pivot. When chosen, the pivot row and column are permuted to row 2 and column 2 of (2.3), row 2 of $\mathbf{F}_{11}$ is scaled by the pivot, and columns 3 to k of the frontal matrix are updated. This process continues until no more pivots can be found. Assuming k pivots have been chosen, $\mathbf{F}_{12}$ is then updated using the Level 3 BLAS routine `_TRSM`

$$\mathbf{F}_{12} \leftarrow -\mathbf{F}_{11}^{-1}\mathbf{F}_{12}, \tag{2.4}$$

and, finally, $\mathbf{F}_{22}$ is updated using the Level 3 BLAS routine `_GEMM`

$$\mathbf{F}_{22} \leftarrow \mathbf{F}_{22} + \mathbf{F}_{21}\mathbf{F}_{12}. \tag{2.5}$$

In practice, for a general matrix **A**, stability restrictions may only allow r pivots to be chosen ($r < k$) and, in this case, the first r rows of $\mathbf{F}_{12}$ are updated using `_TRSM` and then the remaining $k - r$ rows of $\mathbf{F}_{12}$, together with $\mathbf{F}_{22}$, are updated using `_GEMM`.

If the matrix factors are held in direct access files, the frontal method can solve quite large problems with modest amounts of high-speed memory. We remark that, because the size of the frontal matrix increases when a variable appears for the first time and decreases whenever it is eliminated, the order in which the elements (or equations) are assembled is critical. Elements (or equations) should be preordered to reduce the size of the frontal matrices. Various algorithms have been developed for doing this, and we discuss the effect of preordering in Section 5.

For the runs in this paper, we have assumed that the original unassembled elemental problem can be held in main memory. This is to simplify the use of ordering strategies when used with the frontal codes. Although our ordering routines require the integer data to be held in main memory, our frontal codes do not require this. Indeed, if it is possible to generate element matrices on demand, the original matrix need never be computed or stored. This is a common situation for large finite-element problems.

3 Harwell Subroutine Library frontal solvers

The two frontal codes that we use in the following comparisons are the code MA62 for symmetric positive-definite systems and the code MA42 for unsymmetric problems. Both MA42 and MA62 are for real matrices $\mathbf{A}$; a version of MA42 for complex matrices, ME42, is also available. In this section, we describe characteristics of the user interface to our frontal solvers and briefly discuss associated software in the Harwell Subroutine Library (HSL). Further details are given in Duff and Scott (1993 and 1996*b*).

The symmetric positive-definite code, MA62, only permits input by elements, whereas the unsymmetric code, MA42, allows input by either elements or equations. For assembled matrices with a symmetric structure, we can use the HSL routine MC37 to generate a set of element matrices that, if assembled, would yield the same matrix. This allows us to run MA62 on matrices that are supplied in assembled form.

Both MA42 and MA62 use reverse communication to obtain information from the user. The structure of the problem is first provided by the user by calling a subroutine for each element (or equation). The primary reason for these calls is to establish when variables are fully summed and hence are candidates for use as pivots. Thereafter, for the positive-definite solver MA62, a set of calls to another subroutine enables an accurate forecast to be made for the size of the files required to hold the factors and the maximum order of the frontal matrix so that the numerical factorization can be run efficiently and reliably. For the unsymmetric MA42 code, the use of such a prediction routine is optional and will only give lower bounds on the relevant quantities because of the possibility of numerical pivoting in the factorization. In these *symbolic phases*, only the integer indexing information for the elements (or equations) is used. Both codes can use direct access files for the matrix factors, and the user must define these by a simple subroutine call if this option is required.

The numerical factorization is then performed with the user required to call a further subroutine for each element or equation. The information from the earlier symbolic phases is used to control the pivot selection and elimination within the current frontal matrix. Optionally, forward elimination can be performed on a set of elemental right-hand side vectors, in which case a final back-substitution phase yields appropriate solutions. Subsequent right-hand sides can be solved using the matrix factors, in which case the right-hand sides are supplied as dense vectors and a single subroutine call is all that is required. In the unsymmetric case, the same factors can be used by MA42 to solve the equation $\mathbf{A}^T\mathbf{x} = \mathbf{b}$.

Routines for preordering elements for the frontal solvers have been developed using logic similar to that for bandwidth minimization. The HSL code MC43 (Duff, Reid and Scott 1989*b*) offers the choice of basing the ordering on the element structure or on the usual sparse matrix pattern. These two approaches are termed direct and indirect element ordering, respectively. There is little difference in the quality of the ordering from the two approaches but, for the case where we use MC37 to generate elements from the assembled matrix, we use the indirect algorithm because many small elements are generated so that the former option is much slower. For example, MC37 generates 12992 elements for the test problem BCSSTK15 of order 3948 (see Section 4). The times taken by MC43 using the direct and indirect options on a single processor of a CRAY J932 were 10.4 and 2.0 seconds, respectively.

The HSL code MC40 can be used to order symmetrically structured matrices. Since MC40 provides an ordering for the elimination of the variables and the equation input to MA42 needs an ordering for the assembly of the rows, we scan the rows and first order all rows in which the first variable in the elimination ordering appears, then those unordered rows in which the second variable appears, and so on.

Other auxiliary routines for the HSL frontal codes concern their use for elemental problems in a parallel computing environment. In this case, the user partitions the underlying finite-element domain into subdomains and applies `MA42` (element input) to each subdomain. The HSL package `MA52` is then used in completing the solution on the whole domain. This is discussed by Duff and Scott (1994). Ordering schemes have been developed to exploit this different structure (Scott 1996).

It is worth pointing out that there are several criteria on which orderings for frontal or variable-band methods can be based. If the generic entry in the matrix $\mathbf{A}$ of order n is a_{ij}, then the semi-bandwidth is defined as

$$\max_{1\leq i\leq n}\left\{\max_{a_{ij}\neq 0, j=1,..i}(i-j)\right\}$$

and is an appropriate measure for a variable-band code of the type we consider further in Section 6. The profile,

$$\sum_{i=1}^{n}\left\{\max_{a_{ij}\neq 0, j=1,..i}(i-j)\right\},$$

gives a better measure of both the storage and work required. More appropriate measures for frontal schemes are based on the front size, that is the order of the frontal matrix (2.3) after assemblies but before eliminations. If we denote by f_i the front size after assembly of element i, then an important measure, particularly for computing the amount of in-core storage required, is the maximum front size

$$\max_{i=1,m} f_i,$$

where there are m assembly steps. A prediction of the work involved in the frontal algorithm can be obtained from the root-mean squared front size (rms front size) defined by

$$\left(\sum_{i=1}^{m} {f_i}^2/m\right)^{1/2}.$$

We will use these measures in our later comparisons.

4 The problems and environment for numerical testing

In this section, we describe the test problems that we use for the comparisons in this paper and the environment for our numerical testing. All the test problems arise in real engineering and industrial applications. A brief description of each of the unassembled finite-element test problems is given in Table 4.1. The first seven problems are from the Harwell-Boeing Collection (Duff, Grimes and Lewis 1989*a*, Duff, Grimes and Lewis 1992), the RAMAGE01 and RAMAGE02 problems are from Alison Ramage of the University of Strathclyde (Ramage and Wathen 1993), the problem AEAC5081 is from Andrew Cliffe of AEA Technology, and the remaining problems (TRDHEIM, CRPLAT2, OPT1, and TSYL201) were supplied by Christian Damhaug of Det Norske Veritas, Norway. The problem MAN5976 is a condensed version of a matrix from structural engineering. For this problem, we assume that there are three variables at each node, giving a total of 17928 variables. For each of the finite-element test problems, values for the entries of the matrix were generated using the HSL pseudo-random number generator `FA01`. For the symmetric positive-definite test cases, each element was made symmetric and diagonally

Identifier	Number of variables	Number of elements	Description/discipline
CEGB3306	3222	791	2.5D Framework problem
CEGB2919	2859	128	3D cylinder with flange
CEGB3024	2996	551	2D reactor core section
LOCK1074	1038	323	Lockheed gyro problem
LOCK2232	2208	944	Lockheed tower problem
LOCK3491	3416	684	Lockheed cross-cone problem
MAN5976	17928	784	Structural engineering
RAMAGE01	1476	128	3D Navier-Stokes
AEAC5081	5081	800	Double glazing problem
TRDHEIM	22098	813	Mesh of the Trondheim fjord
TSYL201	20685	960	Part of oil production platform
OPT1	15449	977	Part of oil production platform
CRPLAT2	18010	3152	Corrugated plate field
RAMAGE02	16830	1400	3D Navier-Stokes

Table 4.1: The unassembled finite-element test problems

dominant. Unless stated otherwise, the elements were preordered using `MC43` before the frontal solvers were called.

The assembled matrices are shown in Table 4.2. The first fourteen problems are taken from the Harwell-Boeing Collection, the remaining problems WANG3, GARON2, ONETONE2, TWOTONE, and GOODWIN were supplied to us by Tim Davis, University of Florida. For the assembled problems, if numerical values were provided with the matrix, these values are used in our experiments. Otherwise, `FA01` is used to generate numerical values. For element input to the frontal solvers, the symmetrically structured problems were first converted to an equivalent elemental form using `MC37`.

All the HSL codes used in our numerical experiments have control parameters with default values. Unless otherwise stated, we use these defaults in each case, even if different codes sometimes choose a different value for essentially the same parameter.

Identifier	Order	Number of entries	Description/discipline
BCSPWR10	5300	13571	Eastern US Power Network
BCSSTK15	3948	60882	Model of an offshore platform
BCSSTK18	3948	80519	Nuclear power station
BP 1600	1600	4841	Basis matrix from LP problem
GRE 1107	1107	5664	Simulation studies in computer systems
JPWH 991	991	6027	Circuit physics modelling
LNS 3937	3937	25407	Fluid flow modelling
LNSP3937	3937	25407	Fluid flow modelling
NNC1374	1374	8606	Nuclear reactor core modelling
ORSREG 1	2205	14133	Oil reservoir simulation
PORES 3	532	3474	Oil reservoir simulation
SHERMAN3	5005	20033	Oil reservoir simulation
WEST2021	2021	7353	Chemical engineering
PSMIGR 3	3140	543162	Population migration
WANG3	26064	177168	3D semiconductor device simulation
GARON2	13535	390607	2D Navier-Stokes
ONETONE2	36057	227628	Harmonic balance method
TWOTONE	120750	1224224	Harmonic balance method
GOODWIN	7320	324784	Nonlinear fluid mechanics problem

Table 4.2: The assembled test problems. First three problems are symmetric.

The experimental results quoted in this paper were obtained on a single processor of a CRAY J932 using 64-bit floating-point arithmetic, and the vendor-supplied BLAS. In separate runs on the Level 3 BLAS subroutine SGEMM, we found that its peak performance (r_∞) was 195 Mflop/s attained on dense matrices of order greater than 500, that for matrices of order 100 the performance was 186 Mflop/s, and that the vector length for half peak ($n_{1/2}$) was 20. With the exception of the code `VBAN` (see Section 6), all the codes used in our experiments are written in Fortran 77 and were compiled using the CRAY Fortran compiler cf77-7, with compiler option -Zv. The Fortran 90 code `VBAN` was compiled with the CRAY Fortran compiler f90, with default options. All times quoted are CPU times in seconds and include the i/o overhead for the codes that use direct access files. In some of our tables of results, the string "NS" is used to denote that we were unable to run the code. This is usually because the CPU time required for factorization exceeded 30 minutes. We discuss the performance of the BLAS and the effect of using other computers for the experiments in Section 8.

In the tables of results presented in this paper, the "In-core" storage figures are the minimum in-core storage requirements for performing the matrix factorization and solving the linear system $\mathbf{Ax} = \mathbf{b}$. This figure includes both real and integer storage. Since, on the CRAY, both integers and reals are stored in 64-bit words, the value is just the sum of the number of real and the number of integer words needed. We remark that if this minimum in-core storage is used, the performance of the codes in our study will often be considerably degraded since either a large number of data compressions must be performed or a large number of records written to direct access files. In all the tables in which the number of floating-point operations ("ops") are quoted, we count all operations (+,-,*,/) equally. For the frontal codes, the operation counts assume that there are no zeros in the frontal matrices.

We note that the "Solve" times quoted in the tables of results are for a single right-hand side $\mathbf{b}$ and do not include the time required to perform iterative refinement. It should be noted, however, that some of the problems in our test set are so ill-conditioned that iterative refinement is needed for accurate solutions.

5 The effect of ordering and elemental form

As explained in Section 3, the order in which the elements or equations are presented to the frontal solver has a significant effect on its performance. We have already reported on the effect of element ordering on the unsymmetric frontal solver in Duff et al. (1989*b*). For unsymmetric assembled problems, the Harwell Subroutine Library does not contain the equivalent of a profile minimizer but, for problems with a nearly symmetric structure, a good ordering for `MA42` can be obtained by applying the profile reducing code `MC40` to the pattern of $\mathbf{A} + \mathbf{A}^T$. The unsymmetric test problems ORSREG 1, LNS 3937, LNSP3937, SHERMAN3, WANG3, and GARON2 have sparsity patterns which are symmetric or nearly symmetric (LNSP3937 is a permutation of LNS 3937). For these problems, we show the effects of the ordering on `MA42` in Table 5.1. We see that a significant reduction in the profile is achieved for each of the problems except LNSP3937, where there was an excellent initial ordering. This is reflected in the much lower factorization times and operation counts, although we note that the effect of using Level 3 BLAS means that the poorer orderings have a higher Megaflop rate so that the ratio of times, before and after ordering, is not as high as the operation count ratio. The increase in the analyse time between "before" and "after" is the time taken to run `MC40`. The GARON2 problem was not solved with the original ordering since the analyse phase showed that the frontal matrix must be of order 11975×13535 (13535 is the order of the matrix $\mathbf{A}$ in this example).

Identifier	Profile ($*10^3$)		Number of ops ($*10^8$)		Analyse time (seconds)		Factorization time (seconds)	
	Before	After	Before	After	Before	After	Before	After
ORSREG 1	789	157	12.65	0.49	0.15	0.43	12.94	1.22
LNS 3937	5616	204	95.11	0.49	0.27	0.31	96.29	1.70
LNSP3937	236	204	0.76	0.49	0.27	0.32	2.15	1.84
SHERMAN3	917	197	6.01	0.59	0.33	0.71	7.73	1.90
WANG3	22671	12844	804.01	285.41	1.80	5.87	643.84	252.79
GARON2	82220705	2059844	NS	1908.76	1.58	7.09	NS	24.99

Table 5.1: The results of using `MC40` to order matrices with a nearly symmetric pattern for `MA42` equation input. The GARON2 problem was not solved with the original ordering due to excessive storage requirement.

In Table 5.2, we show the results of using the element ordering code `MC43` with our symmetric positive-definite frontal solver, `MA62`. In some cases, a significant reduction in the maximum and root mean squared (rms) front sizes are obtained and this is again reflected in the reduced factorization times and operation counts. We note that, in all cases, the original order is that provided by the application and, in most instances, this was believed, by the originator of the problem, to be a "good" element order. Having generated a new ordering, `MC43` compares the maximum front size of the new ordering with that of the original ordering, and then returns to the user the ordering with the smallest maximum front size. However, it is possible that by doing this `MC43` rejects the ordering with the smallest rms front size (and hence the ordering which would give the smallest operation count and factorization time when used with the frontal solver). In our experiments we therefore made a minor alteration to `MC43` so that the ordering with the smallest rms front size was selected, even if the maximum front size was increased. We see the effect of this on test problems AEAC5081 and TRDHEIM. We note, from the results in Table 5.2, that usually there is a similar reduction in maximum front size and rms front size, and that the number of operations is reduced by a factor of roughly the square of the reduction in the rms front size. The "BLAS effect" is again seen in the lower reduction of factorization time compared with the reduction in operation count.

As observed earlier, the unsymmetric frontal code `MA42` has both an equation and an element input. We can thus compare the efficiency of a frontal code on the original elemental problem and its assembled form. To try and make a valid comparison, we use `MC43` to obtain an ordering for `MA42` in the former case and `MC40` for the ordering in the latter. We can also take a symmetrically structured problem and split it into elements using `MC37`, and then run `MA42` on the elemental and assembled forms, again preordering with `MC43` and `MC40`, respectively. We show the results from these runs in Table 5.3. Not unexpectedly, the factorization times and storage are less when advantage is taken of the element structure in the elemental problems. Indeed the assembled matrix typically requires almost twice the time and storage over using the unassembled elemental formulation. However, perhaps more surprisingly, it is better to "disassemble" assembled problems, although the gains are slightly less in this case. These results suggest that, when using a frontal solver, we should avoid preassembling an elemental problem and that, for symmetrically structured assembled problems, significant savings in both the factorization time and the storage required by the factors can be achieved by generating an equivalent elemental problem, reordering the elements, and using the element input offered by `MA42`.

Identifier	Max front size		rms front size		Number of ops ($*10^8$)		Factorization time (seconds)	
	Before	After	Before	After	Before	After	Before	After
CEGB3306	354	78	123.5	30.2	2.1	0.2	2.5	0.6
CEGB2919	348	291	48.7	43.0	1.2	1.0	1.4	1.3
CEGB3024	152	132	48.0	37.2	0.4	0.2	0.8	0.7
LOCK1074	810	126	333.5	46.1	2.9	0.1	2.4	0.3
LOCK2232	1266	72	484.9	33.0	12.6	0.1	11.1	0.4
LOCK3491	834	217	247.8	62.0	11.6	0.7	10.7	1.3
MAN5976	197	205	66.12	65.65	4.89	4.88	3.0	2.9
RAMAGE01	457	372	102.7	83.7	1.8	1.2	1.7	1.3
AEAC5081	150	161	57.5	44.9	1.1	0.7	1.9	1.6
TRDHEIM	276	348	37.5	35.7	5.4	4.9	7.1	7.1
TSYL201	1200	540	187.8	112.7	154.3	65.3	106.5	45.1
OPT1	2681	983	507.7	156.9	633.8	56.6	430.1	44.3
CRPLAT2	1564	538	494.1	153.7	252.2	26.0	231.8	29.8
RAMAGE02	1717	1452	432.3	375.0	378.3	282.7	279.5	203.2

Table 5.2: The results of using the `MC43` ordering with `MA62`. The root mean-squared front size is denoted by "rms front size".

Identifier	Factorization time (seconds)		Storage (Kwords)	
	elements	assembled	elements	assembled
CEGB3306	1.0	1.8	450	929
CEGB2919	2.3	5.9	1087	2052
CEGB3024	1.2	1.9	593	962
LOCK1074	0.4	0.7	188	260
LOCK2232	0.6	0.9	261	385
LOCK3491	2.2	4.5	1039	1832
BCSPWR10	2.7	2.1	821	1452
BCSSTK15	7.3	12.9	3230	4184
BCSSTK18	57.7	81.5	11397	20770

Table 5.3: The effect of preassembly on `MA42`.

6 A comparison of the frontal code MA62 with other symmetric positive-definite HSL codes

In this section, we examine the performance of the frontal code `MA62` and compare it with the HSL code `MA27` and the code `VBAN`, which is a development version of the HSL code `MA55`.

The code `MA27` uses a multifrontal algorithm (Duff and Reid 1982, Duff and Reid 1983). During the analyse phase, pivots are selected from the diagonal using the minimum degree criterion. During the factorization, this pivot sequence may be modified to maintain numerical stability, and 2×2 diagonal block pivots can also be used. By this means, `MA27` can stably factorize symmetric indefinite problems. However, if the matrix is known to be positive definite, the user can set a parameter in the calling sequence so that a logically simpler path in the code is followed. In all our tests using `MA27`, this option was used.

Our colleague John Reid at the Rutherford Appleton Laboratory is currently developing a variable-band code for the solution of systems of equations whose matrix is

symmetric and positive-definite. It does no interchanges and takes advantage of variation in bandwidth. The code optionally uses a direct access file to store the matrix factor. The intention is that the new HSL code MA55 will replace the HSL code MA36. At present, the development code is written in Fortran 90. A Fortran 77 version of MA55 will be made available in the future. In our comparisons, we have used an early version of MA55 that only uses Level 1 BLAS. It is intended that the MA55 code will use blocking and Level 3 BLAS. We have called this early version VBAN in the tables and in the following text.

We compare the three codes, first on our set of elemental problems described in Table 4.1, then on the symmetric assembled matrices (the first three problems in Table 4.2). In the former case, the matrices are assembled before calling MA27 and VBAN; in the latter case, an elemental problem is first created from the assembled problems using MC37, before calling MA62. In neither case is the cost of this preprocessing included in the times quoted in Table 6.1. Since the efficiency of VBAN depends upon the equations being ordered for a small profile, the assembled matrix is ordered using MC40 prior to calling VBAN, and the time taken to do this is given as the "Analyse" time for VBAN. For MA62, the "Analyse" time is the time needed to order the elements using MC43 together with the time for the symbolic phases discussed in Section 3. For MA27, the "Analyse" time is that taken to select the pivot sequence using the minimum degree criterion and prepare the data structures for subsequent numerical factorization. It is interesting that this more complicated MA27 analyse is usually faster than the "bandwidth reordering" for VBAN. This highlighted for us some deficiencies in the MC40 ordering code which we are now attempting to rectify. Similar deficiencies are also present in the MC43 code but are masked in the case of the elemental matrices (see analyse times for MA62 in these cases) because the ordering works with the connectivity pattern of elements rather than variables. We note that the analyse times for MA62 for the assembled problems are usually closer to the VBAN analyse times.

Neither the MA27 nor the VBAN code use Level 3 BLAS. Although the CRAY J932 is not the best machine to see the importance of this (since Level 1 BLAS perform rather well on this machine), we note that quite often MA62 will require less time for factorization than VBAN although it needs more floating-point operations. In most cases, we see that the minimum degree ordering as expected performs a much better job of reducing the number of entries in the factors than our "band" orderings; sometimes there are nearly four times fewer entries in the factors for MA27 than for the better of the other codes, although the advantage is not usually so marked for the elemental problems. The number of entries in the factors is slightly less for VBAN than for MA62 for the elemental problems, because the blocking in MA62 for Level 3 BLAS creates additional zero entries in the factors. MA62 is generally much worse on the assembled matrices since, for these problems, MA62 seldom is able to choose a large enough pivot block size (order of $\mathbf{F}_{11}$ in (2.3)) to offset the costs of having to store integer information on the factors. Both VBAN and MA62 store their factors in direct access files and so, as expected, usually require much less "In-core" than MA27. However, VBAN sometimes requires a lot of in-core storage (for example, BCSSTK15). This will happen if there is just a single row of high bandwidth towards the end of the reordered matrix. For the simple variable-band scheme used by VBAN, this would require that many previous rows needed to update this be held in memory. The frontal code does not suffer from this problem; the only effect is to add one to the front size for most of the computation. One remedy is to develop better orderings for the variable-band scheme and this is currently being studied.

In nearly all cases, the use of a minimum degree ordering by MA27 gives a substantially lower operation count for the factorization and much less storage for the factors. The RAMAGE02 problem, however, requires considerably more in-core storage and operations for the factorization when using the minimum degree ordering. To check that it was indeed

Identifier	Code	Time (seconds)			Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Solve		In-core	Factors
CEGB3306	MA27	0.4	0.6	0.03	2.1	114	81
	VBAN	0.5	1.1	0.04	11.6	133	187
	MA62	0.2	0.6	0.08	14.6	6	222
CEGB2919	MA27	1.9	3.6	0.04	57.3	590	384
	VBAN	2.6	3.7	0.05	110.6	402	526
	MA62	0.1	1.3	0.08	99.5	85	542
CEGB3024	MA27	0.6	1.0	0.04	7.1	175	146
	VBAN	0.8	1.3	0.04	17.7	106	219
	MA62	0.2	0.7	0.10	23.1	18	295
LOCK1074	MA27	0.3	0.5	0.01	4.9	109	71
	VBAN	0.4	0.5	0.01	6.0	29	77
	MA62	0.1	0.3	0.03	7.8	16	94
LOCK2232	MA27	0.4	0.6	0.02	2.7	133	83
	VBAN	0.5	0.6	0.02	4.6	11	99
	MA62	0.3	0.4	0.05	7.3	5	130
LOCK3491	MA27	0.9	2.0	0.04	20.2	336	254
	VBAN	1.3	3.0	0.05	65.4	322	428
	MA62	0.2	1.3	0.12	67.3	47	518
RAMAGE01	MA27	1.3	4.0	0.03	94.0	549	345
	VBAN	2.6	3.3	0.03	122.7	243	401
	MA62	0.1	1.3	0.06	116.2	139	414
AEAC5081	MA27	1.3	3.1	0.08	44.4	526	430
	VBAN	1.5	3.7	0.07	69.8	95	564
	MA62	0.3	1.6	0.16	69.5	24	626
TRDHEIM	MA27	10.8	17.7	0.27	211.0	2893	2002
	VBAN	15.3	19.6	0.39	459.0	798	2958
	MA62	0.6	7.3	0.55	491.6	121	3601
TSYL201	MA27	13.6	90.0	0.40	4285.0	8922	7069
	VBAN	22.3	96.4	1.18	5262.0	2079	10231
	MA62	0.7	45.1	1.16	5532.6	292	10964
OPT1	MA27	11.9	77.1	0.32	3648.9	7741	5975
	VBAN	20.7	74.2	0.86	4116.5	3315	7215
	MA62	0.8	44.3	1.01	5657.3	966	8936
CRPLAT2	MA27	5.3	40.2	0.30	1623.8	4554	3815
	VBAN	9.4	54.9	0.77	2475.8	2276	6406
	MA62	1.2	29.8	1.25	2597.0	290	7521
RAMAGE02	MA27	20.4	783.0	1.05	44988.9	30569	21297
	VBAN	39.1	338.3	2.16	29922.7	3692	21787
	MA62	1.4	203.2	2.15	28272.8	2108	22646
BCSPWR10	MA27	0.5	0.4	0.06	0.3	56	56
	VBAN	0.3	1.1	0.05	9.5	71	205
	MA62	1.2	1.2	0.22	11.1	4	411
BCSSTK15	MA27	2.7	7.1	0.07	219.4	951	788
	VBAN	1.0	7.0	0.08	230.7	524	904
	MA62	2.9	4.8	0.36	253.2	144	1780
BCSSTK18	MA27	5.1	6.9	0.17	142.6	890	797
	VBAN	2.2	22.4	0.22	1043.2	1987	3097
	MA62	4.9	16.3	1.16	1043.5	236	5709

Table 6.1: A comparison of MA62 and MA27 on symmetric positive-definite systems

the minimum degree ordering that caused the high operation count for the factorization, we ran MA27 using the ordering produced by MC40 for VBAN. This reduced the operation count for MA27 to the same as for VBAN, so we can offer RAMAGE02 to the community as an example showing poor performance of minimum degree when used with a multifrontal scheme. We intend to investigate this problem further. Interestingly the number of operations to factorize this matrix if ordered by the approximate minimum degree ordering (Amestoy, Davis and Duff 1996) is 34762 million, significantly less than that for minimum degree. The fact that the approximate minimum degree ordering can do better on a problem on which the minimum degree ordering does badly is also seen in the results of Rothberg and Hendrickson (1996) and others.

It is apparent, from our results, that the frontal code performs well on unassembled finite-element problems, particularly in the analyse and numerical factorization phases. The benefit of holding the factors out-of-core in order to reduce the maximum amount of storage needed to perform the factorization is evident from the MA62 results for both assembled and unassembled matrices. However, storage for the factors is usually far greater for the frontal method and this is reflected in the much poorer times for subsequent solution, although it should be mentioned that MA62 is more efficient if multiple right-hand sides are being solved at the same time. For example, for problem OPT1, the solve times for MA62 for a single right-hand side and for 10 right-hand sides are 1.01 and 3.98 seconds, respectively (see Duff and Scott (1996*b*) for further results). It is not generally advisable to use a frontal method on an assembled matrix.

7 A comparison of the frontal code MA42 with other HSL codes

In this section, we compare the performance of the frontal code MA42 for unsymmetric problems with some other sparse direct codes from the Harwell Subroutine Library. We consider both assembled and unassembled problems. In the unsymmetric case, we have several alternative HSL codes available, although unlike MA42 none hold the matrix factors out-of-core.

All of the codes that we compare in this section will not accept a potential entry of the reduced matrix, a_{ij} say, as a pivot in Gaussian elimination unless it satisfies an inequality of the form

$$|a_{ij}| \geq u. \max_{1 \leq k \leq n} |a_{kj}|,$$

although the exact test differs from code to code. This type of numerical pivoting is called *threshold partial pivoting* and the parameter u is called the *threshold parameter*.

We first consider codes for problems which are supplied in assembled form. For these problems, we compare the MA42 equation input with the HSL codes MA38 (Davis and Duff 1993), MA41 (Amestoy and Duff 1989), and MA48 (Duff and Reid 1993, Duff and Reid 1996). As discussed earlier, for test problems with a nearly symmetric pattern, the equations are preordered for MA42 by applying MC40 to the pattern of $\mathbf{A} + \mathbf{A}^T$ but for the remaining problems, the original ordering will be used for MA42.

The code MA38 is based on a combined unifrontal/multifrontal algorithm (Davis and Duff 1995) and uses an approximate minimum degree ordering (Amestoy et al. 1996). MA38 does not have separate analyse and factorize phases, but can rapidly factorize a matrix with the same sparsity pattern as one which it has previously factorized. During the factorization, threshold partial pivoting is used, with a default threshold parameter of 0.1. The solve step can be used to solve $\mathbf{Ax} = \mathbf{b}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{b}$, and an option exists for performing iterative refinement.

Identifier	Code	Time (seconds)			Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Solve		In-core	Factors
BP 1600	MA37	3.59	7.34	0.01	104.9	687	349
	MA41	0.46	1.51	0.03	100.3	697	334
	MA41†	0.28	0.11	0.01	0.8	68	35
JPWH 991	MA37	0.33	0.37	0.01	4.4	108	74
	MA41	0.22	0.23	0.01	4.7	138	77
	MA41†	0.24	0.23	0.01	4.7	138	77
PORES 3	MA37	0.06	0.07	0.01	0.1	16	15
	MA41	0.06	0.06	0.01	0.2	33	16
	MA41†	0.07	0.06	0.01	0.2	33	16
GRE 1107	MA37	0.79	1.32	0.01	26.0	284	210
	MA41	0.31	0.59	0.01	21.5	288	288
	MA41†	0.33	0.59	0.01	21.5	271	192
NNC1374	MA37	0.23	10.74	0.02	62.3	481	320
	MA41	0.20	0.52	0.01	9.2	209	154
	MA41†	0.40	0.76	0.01	22.4	159	217
ORSREG 1	MA37	0.78	1.00	0.03	14.8	251	207
	MA41	0.71	0.61	0.03	14.4	315	205
	MA41†	0.76	0.61	0.03	14.4	315	205
LNS 3937	MA37	1.16	5.61	0.05	80.7	674	615
	MA41	0.96	2.20	0.04	73.9	754	575
	MA41†	1.95	3.79	0.04	208.2	1276	1037
LNSP3937	MA37	1.12	5.79	0.04	94.6	728	643
	MA41	1.00	2.04	0.04	63.6	731	546
	MA41†	1.45	3.44	0.04	168.2	1196	937
SHERMAN3	MA37	1.10	1.52	0.05	19.3	320	275
	MA41	1.75	0.96	0.05	18.0	447	276
	MA41†	1.83	0.95	0.05	18.0	447	276
WEST2021	MA37	5.82	40.69	0.04	586.6	1939	1147
	MA41	0.95	5.74	0.04	456.7	1762	987
	MA41†	0.36	0.23	0.03	0.4	105	48
PSMIGR 3	MA37	1084.25	573.08	0.18	8646.8	18121	6995
	MA41	20.09	184.04	0.19	9213.9	20756	6466
	MA41†	21.81	175.69	0.17	8973.5	18579	6389

Table 7.1: A comparison of MA37 and MA41 on unsymmetric assembled problems. (Threshold parameter 0.1). † denotes the matrix is first preordered to have a zero-free diagonal.

Identifier	Code	Time (seconds)				Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Fast Factorize	Solve		In-core	Factors
BP 1600	MA42	0.06	0.44	0.44	0.076	18.70	44	341
	MA41†	0.29	0.12	0.12	0.010	0.72	67	35
	MA48	0.13	0.03	0.01	0.005	0.02	29	12
	MA38		0.22	0.07	0.008	0.03	27	17
JPWH 991	MA42	0.08	0.58	0.58	0.106	21.24	164	372
	MA41	0.23	0.24	0.24	0.012	4.68	138	77
	MA48	0.72	0.24	0.16	0.006	7.87	142	129
	MA38		0.55	0.17	0.012	4.72	118	76
PORES 3	MA42	0.04	0.22	0.22	0.034	5.05	11	106
	MA41	0.06	0.06	0.06	0.007	0.16	33	16
	MA48	0.20	0.09	0.03	0.003	0.98	26	18
	MA38		0.24	0.08	0.008	0.18	34	20
GRE 1107	MA42	0.08	1.28	1.28	0.186	84.58	75	813
	MA41	0.32	0.61	0.61	0.013	20.47	282	192
	MA48	0.81	0.35	0.18	0.007	6.50	144	131
	MA38		0.86	0.22	0.014	6.19	161	101
NNC1374	MA42	0.10	0.45	0.45	0.050	8.18	5	201
	MA41	0.21	0.54	0.54	0.013	9.19	209	154
	MA48	1.01	0.44	0.20	0.010	5.01	155	135
	MA38		1.41	0.34	0.019	5.64	184	129
ORSREG 1	MA42‡	0.41	1.22	1.22	0.301	487.39	21	931
	MA41	0.75	0.66	0.66	0.028	14.44	315	205
	MA48	3.67	1.65	1.07	0.016	60.02	597	564
	MA38		1.86	0.58	0.026	35.91	423	318
LNS 3937	MA42‡	0.77	1.70	1.70	0.426	49.22	8	1217
	MA41	0.99	1.33	1.33	0.043	32.65	575	415
	MA48	13.39	4.97	3.38	0.033	194.75	1287	1231
	MA38		6.53	1.90	0.057	124.68	1097	738
LNSP3937	MA42‡	0.72	1.84	1.84	0.425	49.21	8	1209
	MA41	1.05	1.33	1.33	0.044	32.64	574	414
	MA48	10.02	3.97	2.43	0.033	97.46	981	926
	MA38		5.62	1.62	0.059	89.87	937	621
SHERMAN3	MA42‡	0.69	1.90	1.90	0.505	58.70	16	1190
	MA41	1.76	0.94	0.96	0.051	18.03	447	276
	MA48	5.49	2.04	1.26	0.029	52.29	630	590
	MA38		2.34	0.80	0.046	32.55	520	386
WEST2021	MA42	0.13	1.33	1.33	0.153	70.21	54	774
	MA41†	0.37	0.22	0.22	0.026	0.35	104	47
	MA48	0.34	0.12	0.03	0.011	0.05	44	25
	MA38		0.85	0.27	0.021	0.06	73	43

Table 7.2: A comparison of HSL codes on unsymmetric assembled problems. † denotes the matrix is first preordered to have a zero-free diagonal. ‡ denotes MC40 used to reorder the pattern of $\mathbf{A} + \mathbf{A}^T$.

MA41 is a multifrontal code. Although options exist for using it on shared memory parallel computers, we set the parameters to run it in sequential mode. In this default mode, it is similar to an earlier HSL code, MA37, but MA41 offers more options and exploits high-level BLAS. The analyse phase of MA41 performs an approximate minimum degree ordering on the structure of $\mathbf{A} + \mathbf{A}^T$ and does not consider the numerical values of the entries. It is thus ideally suited to nearly structurally symmetric matrices where diagonal entries will be suitable as pivots. The factorize phase performs numerical pivoting using threshold partial pivoting (choosing off-diagonal entries if necessary) so that any matrix can be stably factorized. The default value for the threshold parameter is 0.01. The more the numerical pivoting perturbs the ordering given by the analyse phase, the more work and storage will likely be needed for the factorization. If the matrix is very unsymmetric (that is for many entries $a_{ij} \neq 0$ but $a_{ji} = 0$), we have found that the effect of the perturbation to the analyse pivot sequence is much reduced if the matrix is permuted to have a zero-free diagonal before the analyse phase. Indeed further gains can sometimes be obtained by permuting entries with large modulus to the diagonal (Duff and Koster 1996). As we shall see from the results in this section, this enables MA41 to perform well on a wide range of matrices. The solve step can be used to solve $\mathbf{Ax} = \mathbf{b}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{b}$, optionally using iterative refinement.

In Table 7.1, the performance of MA41 is compared with that of MA37. For this comparison, both codes use a threshold parameter of 0.1 (the default value for MA37). In this table and the other tables of results presented in this section, a † by MA41 denotes that the matrix has first been preordered to have a zero-free diagonal. It is clear that, for many problems, MA41 offers a significant improvement over MA37, particularly in terms of the analyse and factorize times and the factor storage. For some problems (for example, WEST2021) we see the advantages which can be gained by the initial preordering. We also note the effect of using an approximate minimum degree ordering in MA41. The quality of the ordering does not vary much from the minimum degree ordering used by MA37, although the analyse times are significantly reduced, particularly on example PSMIGR 3. Because of the apparent superiority of MA41, we use MA41 and not MA37 in our comparisons.

The code MA48 is a general sparse code using Gaussian elimination for solving unsymmetric systems whose coefficient matrix need not even be square. The analyse phase first permutes the matrix to block triangular form and then, on each submatrix of the block diagonal, uses a Markowitz criterion for maintaining sparsity and threshold partial pivoting for numerical stability (the default value of the threshold parameter is 0.1). Although numerical factorization is performed during the analyse phase, the factors are not saved and the analyse phase is terminated immediately the reduced matrix is deemed to be sufficiently dense. The point at which this switch is made has a significant effect on performance and the best value is machine dependent. A subsequent factorize phase must then be used to generate the factors. There is a second factorize option to rapidly factorize a matrix with the same sparsity structure as one previously factorized by the routine. The solve step can be used to solve $\mathbf{Ax} = \mathbf{b}$ or $\mathbf{A}^T\mathbf{x} = \mathbf{b}$, and an option exists for performing iterative refinement.

In Tables 7.2 and 7.3, the results of running the HSL codes MA42 (equation input), MA41, MA48, and MA38 on the assembled test problems are presented. No results are given for MA48 for the problem WANG3 and for MA42 for the problem TWOTONE since our CPU limit of 30 minutes was exceeded. For MA41, the default threshold parameter of 0.01 was used for all the tests **except** for problem TWOTONE. For this problem, we used a value of 0.1 since the solution obtained with the default value was not sufficiently accurate. By comparing the MA41 results in Table 7.1 with those in Table 7.2 we see that there is not usually much difference in the performance of MA41 between using a threshold parameter

Identifier	Code	Time (seconds)				Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Fast Factorize	Solve		In-core	Factors
PSMIGR 3	MA42	1.6	153.6	153.6	0.88	18386.	7783	11346
	MA41	19.3	111.9	111.9	0.20	9214.	20756	6466
	MA48	40.9	72.0	69.9	0.09	10512.	13924	12831
	MA38		95.0	72.0	0.10	9565.	22945	6453
WANG3	MA42‡	6.3	252.8	252.8	13.74	28541.	918	75983
	MA41	37.2	125.7	125.7	0.46	10438.	16420	11970
	MA48	NS	NS	NS	NS	NS	NS	NS
	MA38		369.8	522.9	0.72	46956.	55061	33241
GARON2	MA42‡	7.6	25.0	25.0	1.09	1909.	164	9439
	MA41	3.7	12.4	12.4	0.16	342.	3746	2585
	MA48	215.4	92.2	75.2	0.31	4393.	12972	12165
	MA38		59.1	45.8	0.20	5235.	12348	8533
ONETONE2	MA42	2.6	92.3	92.3	5.92	7687.	223	37747
	MA41†	28.7	14.8	14.8	0.46	601.	4538	3000
	MA48	44.9	14.3	8.1	0.24	281.	3126	2627
	MA38		36.5	6.6	0.48	159.	2873	1734
TWOTONE	MA42	NS	NS	NS	NS	NS	NS	NS
	MA41†	456.3	560.7	560.7	1.93	59925.	55418	35142
	MA48	392.3	163.7	143.4	0.94	14094.	23330	21546
	MA38		430.7	77.9	1.97	6988.	19194	11543
GOODWIN	MA42	1.1	7.7	7.7	0.38	617.	47	4107
	MA41	3.0	5.4	5.4	0.08	156.	2327	2327
	MA48	198.5	85.5	68.8	0.17	3772.	10993	10330
	MA38		40.3	22.0	0.14	2169.	7700	5687

Table 7.3: A comparison of HSL codes on unsymmetric assembled problems. † denotes the matrix is first preordered to have a zero-free diagonal. ‡ denotes MC40 used to reorder the pattern of $\mathbf{A} + \mathbf{A}^T$. Cases marked NS were not solved since the CPU limit of 30 minutes was exceeded.

of 0.01 as opposed to 0.1, although occasionally (for example LNS 3937) there can be a significant reduction in factorization time when the threshold parameter is reduced.

In Tables 7.2 and 7.3, the "fast factorize" time is the time to factorize a matrix with the same sparsity structure as one which has already been factorized. For the codes MA42 and MA41, this is the same as the factorize time since these codes do not offer a fast factorize option. In our tests, when doing subsequent factorizations we did not have to change the pivot sequence chosen by the initial matrix factorization. For most of our test problems, the fast factorize offered a significant saving for codes MA48 and MA38. However, for the problem WANG3, the fast factorize offered by MA38 was significantly slower than the original factorization.

For the assembled problems, no one code is clearly better than the others. The choice of code is dependent on the problem being solved. As expected, MA41 generally has the fastest factorize time for problems with a nearly symmetric structure. For these problems, the ordering obtained by applying MC40 to the pattern of $\mathbf{A} + \mathbf{A}^T$ enables MA42 to perform the matrix factorization more rapidly than both MA38 and MA48, but the frontal code has the disadvantage of generally producing many more entries in the factors than the other codes. For problems which are far from symmetric in structure, the factorize time for MA38 is less than the "Analyse + Factorize" time for MA48, except for matrices which are very sparse. Since for MA48 there is an integer associated with each real in the sparse part of the factors, the storage for the MA48 factors is greater than that for MA38, unless the

matrix is very sparse.

For elemental problems, we compare MA42 directly with the multifrontal code MA46 (Damhaug and Reid 1996), which uses element input. We also assemble the elements and then solve the resulting assembled system with the codes MA38, MA41, and MA48. The multifrontal code MA46 requires the matrix to be input in the form of element-node connectivity lists. The analyse phase uses this unassembled form to determine the ordering of the nodes and builds the necessary information for the factorization and solve steps. The minimum degree heuristic is used to reorder the matrix. The factorization step accepts the tentative pivot sequence provided by the analyse step and, if necessary, modifies it to maintain stability. The default threshold partial pivoting parameter used by MA46 is 0.1. Once the numerical factorization is complete, the solve step allows the user to solve for several right-hand sides at once. The performance of the MA46 factorization phase is affected by a control parameter ICNTL(8) that should be set to the size of the cache memory in KBytes. The code uses this parameter to subdivide matrix-matrix updates into blocks. The default value for ICNTL(8) is 64. The CRAY J932 is not a cache-based machine so we also performed some runs with the cache-size parameter set to 0. Our findings are presented in Table 7.4. For the large test problems, we see that using a cache size of zero can reduce the factorize time significantly. For the problems considered, the number of operations and the storage requirements were the same for the two settings of the parameter. In the remainder of this section, the results presented for MA46 on the CRAY are for ICNTL(8) = 0, even though this is different from the default value.

Identifier	Cache Size (KBytes)	Factorize Time (seconds)
TRDHEIM	64	6.0
	0	6.0
TSYL201	64	71.2
	0	55.4
OPT1	64	58.4
	0	44.6
CRPLAT2	64	16.7
	0	16.2
RAMAGE02	64	1706.0
	0	478.5

Table 7.4: A comparison of MA46 on the CRAY J932 with the default cache size and a cache size of 0.

The results of running the HSL codes MA42 (element input), MA46, MA41, MA48, and MA38 on the elemental problems are given in Tables 7.5 and 7.6. It is again clear that for these problems it is advantageous to use a code that accepts input by elements. For all the test problems except RAMAGE02, MA46 had the fastest factorization time. In all cases, MA42 had the fastest analyse time. The analyse times for the two multifrontal codes MA41 and MA46 were similar for all the problems except MAN5976. For this problem, the analyse phase of MA46 was able to exploit the fact that each node had several variables associated with it. For all the problems except RAMAGE02, the factors produced by MA41 and MA46 required similar storage that was usually somewhat less than for the other codes. We remark that, whereas for the assembled problems, MA42 required more storage for the factors than the other codes; for the elemental problems, less storage is required for the MA42 factors than those produced by MA48 and, for a significant number of test problems, the MA42 factors required less storage than the MA38 factors.

Identifier	Code	Time (seconds)			Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Solve		In-core	Factors
CEGB3306	MA42	0.19	0.97	0.07	23.3	6	450
	MA46	0.58	0.41	0.04	4.3	244	170
	MA41	0.60	0.61	0.02	5.0	397	176
	MA48	2.85	1.14	0.02	6.7	450	294
	MA38		1.55	0.02	7.0	445	221
CEGB2919	MA42	0.10	2.26	0.07	183.3	85	1087
	MA46	2.17	1.51	0.04	111.9	938	773
	MA41	2.41	3.22	0.03	96.4	1580	720
	MA48	15.88	8.94	0.04	237.8	2463	1814
	MA38		7.03	0.02	184.2	2279	993
CEGB3024	MA42	0.19	1.21	0.09	39.3	18	593
	MA46	0.70	0.54	0.05	10.4	324	269
	MA41	0.72	0.88	0.03	12.0	503	268
	MA48	5.04	2.33	0.02	19.4	754	539
	MA38		2.40	0.03	33.5	713	416
LOCK1074	MA42	0.14	0.41	0.02	12.8	16	188
	MA46	0.49	0.26	0.01	8.5	179	138
	MA41	0.39	0.48	0.01	8.8	277	136
	MA48	1.98	1.09	0.01	15.3	397	292
	MA38		0.97	0.01	10.9	333	156
LOCK2232	MA42	0.27	0.58	0.04	11.0	5	261
	MA46	0.69	0.40	0.03	5.4	238	173
	MA41	0.61	0.66	0.02	7.6	399	189
	MA48	2.68	1.39	0.01	7.8	487	322
	MA38		1.44	0.01	8.2	450	212
LOCK3491	MA42	0.24	2.18	0.11	120.6	47	1039
	MA46	1.29	0.95	0.05	37.8	604	503
	MA41	1.24	1.73	0.03	38.7	935	482
	MA48	9.63	4.79	0.03	60.2	1359	1031
	MA38		3.84	0.03	75.2	1309	706
MAN5976	MA42	0.36	76.13	1.05	9217.4	378	19108
	MA46	1.83	28.48	0.21	3482.6	10186	9122
	MA41	15.68	49.66	0.25	4284.0	15343	9879
	MA48	403.31	222.60	0.42	8468.2	26801	22717
	MA38		119.74	0.31	10457.5	25772	16448

Table 7.5: A comparison of HSL codes on Harwell-Boeing unsymmetric elemental problems. For MA46, ICNTL(8) = 0.

Identifier	Code	Time (seconds)			Factor ops ($*10^6$)	Storage (Kwords)	
		Analyse	Factorize	Solve		In-core	Factors
RAMAGE01	MA42	0.1	2.4	0.05	221.	140	829
	MA46	1.8	1.9	0.02	187.	1026	690
	MA41	1.6	3.4	0.02	186.	1410	684
	MA48	9.0	6.6	0.02	439.	2227	1819
	MA38		5.8	0.02	249.	1912	846
AEAC5081	MA42	0.3	2.7	0.14	122.	25	1257
	MA46	1.6	1.7	0.09	99.	1076	917
	MA41	1.7	2.7	0.05	71.	1342	794
	MA48	19.7	9.2	0.05	187.	2327	1922
	MA38		7.2	0.04	242.	2397	1378
TRDHEIM	MA42	0.5	11.5	0.48	884.	121	7221
	MA46	11.5	6.0	0.16	356.	4160	3847
	MA41	14.4	16.0	0.17	344.	8113	3734
	MA48	131.8	50.8	0.25	492.	11613	7700
	MA38		36.7	0.15	691.	11246	5273
TSYL201	MA42	0.7	85.8	1.04	10753.	295	21955
	MA46	16.6	55.4	0.33	7254.	15814	13310
	MA41	19.2	73.3	0.31	7160.	20606	13119
	MA48	671.8	354.8	0.54	15115.	37623	32673
	MA38		153.6	0.30	13768.	32295	19961
OPT1	MA42	0.8	87.6	0.92	11098.	966	17891
	MA46	14.8	44.6	0.29	6192.	13140	11057
	MA41	15.6	66.6	0.22	6549.	18961	11079
	MA48	429.1	241.7	0.40	12508.	29556	25665
	MA38		165.2	0.28	17547.	30424	18745
CRPLAT2	MA42	1.2	53.4	1.13	4996.	292	15056
	MA46	7.3	16.2	0.27	1687.	6744	6006
	MA41	7.4	24.7	0.21	1807.	9093	6130
	MA48	253.0	138.3	0.32	5124.	18131	16175
	MA38		65.5	0.24	5082.	16073	10813
RAMAGE02	MA42	1.2	413.1	2.66	55926.	2190	45313
	MA46	26.2	478.5	0.69	78598.	57097	39617
	MA41	27.7	424.8	0.56	62140.	55805	35651
	MA48	NS	NS	NS	NS	NS	NS
	MA38	NS	NS	NS	NS	NS	NS

Table 7.6: A comparison of HSL codes on unsymmetric elemental problems. Cases marked NS were not solved due to excessive computation requirement. For MA46, ICNTL(8) = 0.

8 Performance in other environments

In our experiments on the CRAY J932, we have used the vendor-supplied BLAS which, as we noted in Section 4, have a high Megaflop rate and a fairly low $n_{1/2}$ value. However, we also performed some runs on the CRAY using a standard Fortran implementation of the BLAS and found that, although most factorization times increased, a few decreased and sometimes by a significant amount. For example, the factorization time for `MA62` on the unordered element problem OPT1 reduced from 430 seconds with the system BLAS to 343 seconds with the Fortran BLAS. Similarly, the factorization time for the equation input to `MA42` on ONETONE2 with the original ordering reduced from 92 to 51 seconds. On further investigation, we found that these improvements in performance were caused by a test within the Fortran implementation of the Level 3 BLAS kernel `_GEMM` for zeros in the outer loop whose presence suppresses an execution of the inner loop. The frontal matrices in `MA42` and `MA62` can contain a large number of zero entries if the matrix is not well ordered, and this is particularly true for the equation input to `MA42`. We can monitor the zeros in the front by looking at the number of zeros in the factors. For the problem ONETONE2, the `MA42` factors have $246*10^5$ entries but $210*10^5$ of these are zeros. The zeros in the front account for the reduction in the factorization time when the Fortran BLAS are used. In some experiments where we timed the Fortran implementation of `_GEMM` with matrices having a large number of zero entries, we apparently achieved performances that were significantly better than the peak speed of the CRAY J932.

In order to explore further the effect of different BLAS on the comparative behaviour of the codes, we performed some runs on other machines, namely an IBM RS/6000 550 and a DEC 7000. We show the results in Tables 8.1 and 8.2. For most of the problems tested, `MA42` takes at least twice as long to perform the factorization on the RS/6000 compared with the CRAY, although the analyse times are faster on the RS/6000. By contrast, for the assembled problems, the factorization phase of `MA48` is generally slightly faster on the RS/6000 than on the CRAY. This reflects the fact that the amount of integer processing is higher for `MA48` than for the other codes and that integer processing is poorer on the CRAY, relative to floating-point computations, than on the other machines. We observe from both Table 8.1 and Table 8.2 that the time for the factorization phase of `MA42` relative to that of `MA41` is poorer on the RS/6000 and the DEC than on the CRAY. For instance, for ONETONE2, the factorize time for `MA42` is approximately 6 times that for `MA41` on the CRAY, but `MA42` is almost 11 times slower than `MA41` on the DEC.

The DEC 7000 has a cache size of 4 MBytes. On this machine, the `MA46` runs were performed with the control parameter `ICNTL(8)` set to 4000. For large problems, the factorization times on the DEC increase substantially if the default value of 64 is used. For the problem TSYL201, the factorization time goes up from 96 to 146 seconds but there was no effect on the factorization time for TRDHEIM. Even if `ICNTL(8)` is set to 4000, the performance of `MA46` relative to that of `MA41` is poorer on the DEC than on the CRAY.

The solve times for `MA42` on the DEC are very high. Further investigation shows that the overheads for out-of-core working are considerable and reading the factors from the direct access files accounts for most of the solve time. For example, for TRDHEIM, approximately 6.8 seconds of the solve time of 7.4 seconds is used to read the factors. For TSYL201 the corresponding times are 22.0 and 23.2 seconds, respectively. On the DEC, the use of direct access files can also have a significant effect on the factorization time. If we hold the factors in-core, then the `MA42` factorization time for TRDHEIM reduces from 28.2 seconds to 15.8 seconds.

Identifier	Code	Time (seconds)			
		Analyse	Factorize	Fast Factorize	Solve
BP 1600	MA42	0.01	0.96	0.96	0.06
	MA41†	0.10	0.08	0.08	0.01
	MA48	0.07	0.02	0.01	0.01
	MA38		0.16	0.07	0.01
JPWH 991	MA42	0.02	1.50	1.50	0.10
	MA41	0.10	0.30	0.30	0.01
	MA48	0.44	0.25	0.18	0.01
	MA38		0.57	0.25	0.01
PORES 3	MA42	0.01	0.41	0.41	0.01
	MA41	0.02	0.03	0.03	0.01
	MA48	0.12	0.04	0.01	0.01
	MA38		0.20	0.07	0.01
GRE 1107	MA42	0.04	4.75	4.75	0.22
	MA41	0.13	0.71	0.71	0.03
	MA48	0.69	0.27	0.19	0.01
	MA38		0.98	0.33	0.02
ORSREG 1	MA42‡	0.20	3.24	3.24	0.27
	MA41	0.27	0.60	0.60	0.03
	MA48	4.29	1.77	1.49	0.03
	MA38		2.57	1.17	0.05
LNS 3937	MA42‡	0.38	3.31	3.31	0.35
	MA41	0.42	1.65	1.65	0.04
	MA48	17.70	4.96	4.02	0.06
	MA38		9.50	4.06	0.09
RAMAGE01	MA42	0.05	6.05	6.05	0.09
	MA46	0.46	5.65	5.65	0.05
	MA41	0.70	4.65	4.65	0.06
	MA48	17.59	10.33	9.59	0.07
	MA38		11.98	7.61	0.06

Table 8.1: A comparison of HSL codes on the RS/6000. † denotes the matrix is first preordered to have a zero-free diagonal. ‡ denotes MC40 used to reorder the pattern of $\mathbf{A} + \mathbf{A}^T$.

Identifier	Code	Time (seconds)			
		Analyse	Factorize	Fast Factorize	Solve
PSMIGR 3	MA42	0.4	312.5	312.5	11.76
	MA41	4.5	133.5	133.5	0.75
	MA48	29.6	106.8	104.9	0.71
	MA38		159.4	130.7	0.71
WANG3	MA42‡	4.1	877.7	877.7	84.62
	MA41	7.4	110.1	110.1	1.44
	MA38		644.0	890.9	5.54
ONETONE2	MA42	0.9	165.2	165.2	42.76
	MA41†	5.3	11.2	11.2	0.45
	MA48	27.5	7.4	5.9	0.32
	MA38		10.5	4.6	0.37
GOODWIN	MA42	0.3	16.4	16.4	3.92
	MA41	0.5	3.4	3.4	0.18
	MA48	287.1	80.5	74.6	0.77
	MA38		38.2	32.0	0.68
TRDHEIM	MA42	0.1	28.2	28.2	7.38
	MA46	1.1	7.2	7.2	0.55
	MA41	3.0	8.0	8.4	0.51
	MA48	93.1	25.9	21.8	0.80
	MA38		19.1	15.5	0.66
TSYL201	MA42	0.1	190.3	190.3	23.17
	MA46	1.5	146.1	96.1	1.61
	MA41	3.8	78.2	78.2	1.48
	MA48	1123.3	487.3		5.88
	MA38		205.4	205.6	2.39

Table 8.2: A comparison of HSL codes on the DEC. † denotes the matrix is first preordered to have a zero-free diagonal. ‡ denotes MC40 used to reorder the pattern of $\mathbf{A} + \mathbf{A}^T$. For MA46, ICNTL$(*) = 4000$.

9 Conclusions

We have shown in our runs on the CRAY J932 that frontal codes can be a very powerful approach for the solution of large sparse systems and are particularly efficient for unassembled finite-element problems when a good ordering can be found. We notice that, in this case, although other approaches may result in much less fill-in, the frontal code is often better in terms of the analyse time and the Megaflop rate and, if the factors are held in direct access files, is far superior in terms of main memory. Indeed, in some cases, this reduction in main memory requirement meant that it was feasible to solve a problem with our frontal schemes which could not be solved by other methods. This has indicated to us the desirability of developing out-of-core versions of some of our multifrontal codes. For most assembled problems, the use of approaches other than the frontal method might be better and, if a good ordering is not available, frontal methods can perform badly.

Our experiments, reported in Section 8, suggest that it is important to exploit sparsity within the Level 3 BLAS, that the performance of our out-of-core frontal schemes are significantly affected by the efficiency of the i/o, and that it is important to exploit machine characteristics, such as cache, for efficient implementation.

Finally, it is clear that the performance of some of the codes that we tested is very dependent on the machine and some of the parameters, like the cache size parameter for `MA46` and the switch to full code in `MA48`. We are currently doing further tests on the sensitivity of the codes to such parameters and would warn against making too sweeping a conclusion on the merits of a code without considering the fine-tuning further.

Acknowledgements

We are grateful to Andrew Cliffe, Christian Damhaug, Tim Davis, and Alison Ramage for providing test problems and to Patrick Amestoy and John Reid for their comments on an earlier draft.

References

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., Davis, T. A. and Duff, I. S. (1996), 'An approximate minimum degree ordering algorithm', *SIAM J. Matrix Analysis and Applications* **17**(4), 886–905.

Damhaug, A. C. and Reid, J. K. (1996), MA46, a Fortran code for direct solution of sparse unsymmetric linear systems of equations from finite-element applications, Technical Report RAL-TR-96-10, Rutherford Appleton Laboratory.

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*, January 1997.

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.

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.

Duff, I. S. (1984), 'Design features of a frontal code for solving sparse unsymmetric linear systems out-of-core', *SIAM J. Scientific and Statistical Computing* **5**, 270–280.

Duff, I. S. and Koster, J. (1996), The design and use of algorithms for permuting large entries to the diagonal of sparse matrices, Technical Report To appear, RAL.

Duff, I. S. and Reid, J. K. (1982), MA27 – A set of Fortran subroutines for solving sparse symmetric sets of linear equations, Technical Report AERE R10533, Her Majesty's Stationery Office, London.

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. (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. (1996), '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 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. (1994), The use of multiple fronts in Gaussian elimination, *in* J. G. Lewis, ed., 'Proceedings of the Fifth SIAM Conference on Applied Linear Algebra', SIAM Press, Philadelphia, pp. 567–571.

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

Duff, I. S. and Scott, J. A. (1996*b*), MA62 – a frontal code for sparse positive-definite symmetric systems from finite-element applications, Technical Report To appear, Rutherford Appleton Laboratory.

Duff, I. S., Grimes, R. G. and Lewis, J. G. (1989*a*), '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., Reid, J. K. and Scott, J. A. (1989*b*), 'The use of profile reduction algorithms with a frontal code', *Int J. Numerical Methods in Engineering* **28**, 2555–2568.

Hood, P. (1976), 'Frontal solution program for unsymmetric matrices', *Int J. Numerical Methods in Engineering* **10**, 379–400.

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).

Irons, B. M. (1970), 'A frontal solution program for finite-element analysis', *Int J. Numerical Methods in Engineering* **2**, 5–32.

Ramage, A. and Wathen, A. J. (1993), Iterative solution techniques for the Navier-Stokes equations, Technical Report AM-93-01, School of Mathematics, University of Bristol.

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

Scott, J. A. (1996), 'Element resequencing for use with a multiple front algorithm', *Int J. Numerical Methods in Engineering* **39**, 3999–4020.