Performance Issues for Frontal Schemes on a Cache-Based High Performance Computer

K A Cliffe I S Duff and J A Scott

January 1997
© Council for the Central Laboratory of the Research Councils 1997

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

The Central Laboratory of the Research Councils
Library and Information Services
Rutherford Appleton Laboratory
Chilton
Didcot
Oxfordshire
OX11 0QX
Tel: 01235 445384 Fax: 01235 446403
E-mail library@rl.ac.uk

ISSN 1358-6254

Neither the Council nor the Laboratory accept any responsibility for loss or damage arising from the use of information contained in any of their reports or in any communication about their tests or investigations.
Performance issues for frontal schemes on a cache-based high performance computer

K. A. Cliffe, I. S. Duff, and J. A. Scott

Abstract

We consider the implementation of a frontal code for the solution of large sparse unsymmetric linear systems on a high performance computer where data must be in the cache before arithmetic operations can be performed on it. In particular, we show how we can modify the frontal solution algorithm to enhance the proportion of arithmetic operations performed using Level 3 BLAS thus enabling better reuse of data in the cache. We illustrate the effects of this on Silicon Graphics Power Challenge machines using problems which arise in real engineering and industrial applications.

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

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

1 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 cdsRAL97001.ps.gz.

2 Address: AEA Technology, Harwell, Didcot, Oxon OX11 0RA, England.

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

1 Introduction
2 Frontal solution schemes
   2.1 The use of BLAS in frontal schemes
   2.2 The effect of reordering
   2.3 The use of direct access files
3 Modification for Level 3 BLAS enrichment
4 The reuse of cache
5 The performance of the modified frontal code
6 Concluding remarks
7 Availability of software
1 Introduction

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

\[ AX = B, \]  

where the \( n \times n \) matrix \( A \) is large and sparse, and \( B \) is an \( n \times nrhs \) \((nrhs \geq 1)\) matrix of right-hand sides. The method is a variant of Gaussian elimination and involves the factorization of a permutation of \( A \) which can be written as

\[ A = PLUQ, \]  

where \( P \) and \( Q \) are permutation matrices, and \( L \) and \( U \) are lower and upper triangular matrices, respectively. The code \( MA42 \) developed by Duff and Scott (1996b) for the Harwell Subroutine Library (1996) uses a frontal scheme for solving systems of the form (1.1) with \( A \) unsymmetric. \( MA42 \) includes an option which allows the assembled matrix \( A \) to be input by rows. However, as illustrated by Duff and Scott (1996d), the power of the frontal scheme is more apparent when the matrix \( A \) comprises contributions from the elements of a finite-element discretization. That is, we can express \( A \) as the sum of elemental matrices

\[ A = \sum_{l=1}^{m} A^{(l)}, \]  

where \( A^{(l)} \) is nonzero only in those rows and columns that correspond to variables in the \( l \)-th element. We shall be concerned with this case in the following. Our aim is to study the performance of a frontal solver on a machine where data must be in the cache before being operated upon.

In Section 2, we discuss salient features of the frontal scheme. In particular, we show how the computation in \( MA42 \) is organized to exploit \( \text{GEMM} \), the Level 3 Basic Linear Algebra Subprogram (BLAS) (Dongarra, DuCroz, Duff and Hammarling 1990) that implements dense matrix-matrix multiplication. We show, in Section 3, how we can modify the frontal algorithm to obtain a factorization which requires a larger number of floating-point operations but which is richer in Level 3 BLAS. The main theme of this paper is to see how this trade-off works in practical applications.

We discuss the effect of a cache in Section 4 and indicate the effect of data reuse by looking at the performance of \( \text{GEMM} \) on a Silicon Graphics Power Challenge machine. In Section 5, we illustrate the effects of exploiting Level 3 BLAS in the frontal solver through experiments on Power Challenge machines using practical problems. Numerical experiments on an IBM RS/6000 and on a CRAY J932 are also reported on.

Finally, in Section 6, we present some concluding remarks.
2 Frontal solution schemes

2.1 The use of BLAS in frontal schemes

A key feature of the frontal method is that the system matrix $A$ is never assembled explicitly but the assembly and Gaussian elimination processes are interleaved, with each variable being eliminated as soon as its row and column are fully summed, that is, after the last occurrence in an elemental matrix $A^{(0)}$. This allows all intermediate working to be performed in a full matrix, termed the frontal matrix, whose rows and columns correspond to variables that have not yet been eliminated but have appeared in at least one of the elements that have been assembled.

Using Fortran notation, the innermost loop of a typical frontal method for an elemental problem is of the form

\[
\text{do } j = 1, \text{frnt} \\
p1 = \text{pr}(j) \\
\text{if } (p1 \neq 0.0) \text{ then} \\
\quad \text{do } i = 1, \text{fmt} \\
\quad \quad \text{fa}(i,j) = \text{fa}(i,j) + \text{pc}(i) \times p1 \\
\quad \text{end do} \\
\text{end if} \\
\text{end do}
\]

where $fa$ is the frontal matrix, $pc$ is the pivot column, $pr$ is the pivot row, and $\text{frnt}$ is the order of the frontal matrix. This code represents a rank-one update to the matrix that can be performed using the Level 2 BLAS routine $\text{-GER}$. After the assembly of an element, if there are $k$ fully summed variables which can be eliminated, then $k$ calls to $\text{-GER}$ can be made. However, as we shall illustrate in Section 5, the computation is made more efficient if we avoid updating the frontal matrix until all pivots for the current element have been chosen. If we delay the elimination operations in this way, the Level 3 BLAS routine $\text{-GEMM}$ can be used. We now discuss in a little more detail how this is achieved in the Harwell Subroutine Library (HSL) code MA42.

After the assembly of an element, if the $k$ fully summed variables are permuted to the leading rows and columns, the frontal matrix can be expressed in the form

\[
\begin{pmatrix}
F_{11} & F_{12} \\
F_{21} & F_{22}
\end{pmatrix},
\]

where $F_{11}$ is a square matrix of order $k$. The rows and columns of $F_{11}$, the rows of $F_{12}$, and the columns of $F_{21}$ are fully summed; the variables in $F_{22}$ are not yet fully summed. Pivots may be chosen from anywhere in $F_{11}$. The columns of $F_{11}$ are searched for a pivot and, when chosen, the pivot row and column are permuted to the first row and column of (2.1). Row 1 of the permuted matrix $F_{11}$ is scaled by the pivot and columns 2 to $k$ of the permuted frontal matrix are updated by $k - 1$ calls to the Level 1 BLAS routine $\text{-AXPY}$. Columns 2 to $k$ of the updated matrix $F_{11}$ are then searched for the next pivot. When chosen, the pivot row and column are permuted to row 2 and column 2 of (2.1), row 2 of $F_{11}$ 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, $F_{12}$ is then updated using the Level 3 BLAS routine \_TRSM
\[ F_{12} \leftarrow -F_{11}^{-1}F_{12}, \] (2.2)
and, finally, $F_{22}$ is updated using the Level 3 BLAS routine \_GEMM
\[ F_{22} \leftarrow F_{22} + F_{21}F_{12}. \] (2.3)
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 $F_{12}$ are updated using \_TRSM and then the remaining $k - r$ rows of $F_{12}$, together with $F_{22}$ are updated using \_GEMM. Further details of how this strategy is implemented within the frontal code MA42 are given by Duff and Scott (1996b).

Once all the eliminations have been performed, the upper triangular part of $F_{11}$ (which we denote by $F_U$) and $F_{12}$ are stored for the $UQ$ factor and the lower triangular part of $F_{11}$ (denoted by $F_L$) and $F_{21}$ are stored for the $PL$ factor. The triangular matrices $F_U$ and $F_L$ are held in packed form. To exploit the block structure, MA42 uses direct addressing in the solution phase. At each stage of the forward elimination, all the active components of the partial solution matrix $Y$ (where $(PL)Y = B$) are put into an array $W = (W_1, W_2)^T$, with $W_1$ of dimension $frnt - r$ by $nrhs$ and $W_2$ of dimension $r$ by $nrhs$, where $frnt$ is the current front size, $r$ is the number of pivots chosen and $nrhs$ of the number of right-hand sides which are being solved (the second dimension of $B$). The operations
\[ W_2 \leftarrow -F_L^{-1}W_2 \] (2.4)
followed by
\[ W_1 \leftarrow W_1 + F_{21}W_2 \] (2.5)
are performed before $W$ is unloaded into $Y$. Similarly, during the back substitution, all the active components of the partial solution matrix $Y$ are put into an array $Z_1$ of leading dimension $r$ and the active variables of the solution matrix $X$ are put into an array $Z_2$ of leading dimension $frnt - r$. The operations
\[ Z_1 \leftarrow Z_1 - F_{12}Z_2 \] (2.6)
and then
\[ Z_1 \leftarrow \hat{F}_U^{-1}Z_1 \] (2.7)
are carried out before $Z_1$ is unloaded into $X$ ($\hat{F}_U$ is the triangular matrix $F_U$ with units on the diagonal). Provided $r > 1$, the forward elimination and back substitution are performed using the Level 2 BLAS kernels \_GEMV and \_TPSV if there is only one right-hand side ($nrhs = 1$), and the Level 3 routine \_GEMM and the Level 2 routine \_TPSV if there are multiple right-hand sides (there is no Level 3 BLAS kernel for solving a triangular system of equations with the matrix held in packed form and multiple right-hand sides). We remark that the interior dimension in the call to \_GEMM (or \_GEMV) is $r$ during the forward elimination and $frnt - r$ during the back substitution. At most stages of the solution phase, $frnt - r$ is larger than $r$ and, in general, the Mflop rate for the forward elimination is therefore lower than for the back substitution.
2.2 The effect of reordering

The order of the frontal matrix increases when a variable appears for the first time and decreases when it is eliminated. Consequently the order in which the elements are assembled has a crucial effect on the performance of the frontal solver. Ordering routines have been developed for frontal solvers and use similar logic to bandwidth minimization. The HSL code MC43 offers the user the choice of basing the ordering on the element structure or on the usual sparse matrix pattern (Duff, Reid and Scott, 1989). These two approaches are termed direct and indirect element reordering, respectively. The results presented by Duff et al. (1989) show that there is little difference in the quality of the ordering from the two approaches but, as observed by Duff and Scott (1996a), the former is generally faster if the problem has fewer elements than nodes. In the numerical experiments reported on in Section 5, the direct element reordering algorithm is used.

2.3 The use of direct access files

Another principal feature of the frontal method is that by holding the PL and UQ factors in direct access files, large problems can be solved using a relatively small amount of in-core memory. A lower bound on the in-core memory required can be obtained by performing a symbolic factorization, which is an option offered by the code MA42. This is only a lower bound because numerical pivoting during the factorization may increase the memory requirements. MA42 uses three direct access files, one each for the reals in PL and UQ and one for the row and column indices of the variables in the factors. Corresponding to each of the direct access files is a buffer (or workspace), which is held in-core. During the factorization, each time a block of pivots is chosen and the frontal matrix (2.1) updated, a record is written to each of the buffers. Once a buffer becomes full (or the final eliminations have been performed), it is written to the appropriate direct access file. Use of direct access files is not needed if sufficient in-core storage is available.

In the integer buffer, each record holds lists of the (global) row and column indices of the variables in the front. Each variable enters and leaves the front once only. By storing the row and column indices of all the variables in the front in each record, more integer storage than necessary is used by MA42. In practice, the repetition of the storage of variable indices in MA42 does not require a prohibitively large amount of storage because, as explained earlier, blocks of pivots are used and a record is only written once a block of pivots has been chosen. In our experience, for elemental problems the required integer storage is in the range 15n to 50n and the number of integers stored is less than a quarter the number of reals stored (detailed results are given by Duff and Scott, 1993 and in Section 5 below).

3 Modification for Level 3 BLAS enrichment

We saw, in Section 2.1, that if r pivots are chosen after the assembly of an element into the frontal matrix, the code MA42 uses the Level 3 BLAS routine _GEMM with interior dimension r to update the frontal matrix prior to the next element assembly. If r is small, there may be little advantage gained by using Level 3 BLAS. We can
3 MODIFICATION FOR LEVEL 3 BLAS ENRICHMENT

increase the Level 3 BLAS component by delaying updating the frontal matrix until the number of pivot candidates is at least some prescribed minimum, say \( r_{\text{min}} \). Suppose, at some stage, that the number of fully summed variables is \( k \), then the maximum number of pivots which we can choose is \( k \). If \( k < r_{\text{min}} \) and not all the elements have been assembled, we do not look for pivots but repeat the process of assembling another element into the frontal matrix until either the number of fully summed variables exceeds \( r_{\text{min}} \) or there is insufficient storage allocated for the frontal matrix to accommodate the next element. We then go ahead and choose as many pivots as possible and update the frontal matrix, before assembling the next element.

Delaying the search for pivots until the number of fully summed variables is at least \( r_{\text{min}} \) (\( r_{\text{min}} > 1 \)) will have several effects on the factorization. Firstly, the total number of calls to the Level 3 BLAS routine \_GEMM will decrease but the average interior dimension will increase since, on most of the calls, the interior dimension will be at least \( r_{\text{min}} \) (numerical considerations may prevent all the potential pivots from being chosen). Secondly, when looking for pivots there will generally be a larger number of fully summed variables to test as potential candidates. Once a pivot is chosen, each of the fully summed columns not yet selected as a pivot column is updated using the Level 1 BLAS routine \_AXPY. Therefore, the number of calls to \_AXPY will increase. This increase can be restricted by making greater use of Level 2 BLAS. We now explain how this can be achieved. As discussed in Section 2.1, once MA42 has chosen a pivot, all the remaining fully summed columns are updated using calls to \_AXPY, and then these columns are searched in turn for the next pivot. The process is repeated until no further pivots can be found. An alternative approach is to delay updating the \( i \)-th fully summed column until it is to be searched for a possible pivot. Assuming columns 1 to \( i - 1 \) have already been successfully used as pivot columns, column \( i \) is updated using the Level 2 BLAS routines \_TRSV and \_GEMV. There is a problem with this approach if column \( i \) is updated using the Level 2 BLAS kernels and then is found to be unsuitable for use as a pivot column. In this case, column \( i + 1 \) is updated using \_TRSV and \_GEMV and then searched. If column \( i + 1 \) is chosen as a pivot column, column \( i \) is again updated, but since it has already been updated for the first \( i - 1 \) pivots, \_AXPY is used to perform a single update, and then column \( i \) is restested. Keeping track of which of the fully summed columns have been updated by which of the pivot columns adds to the complexity of this approach. It also requires that the fully summed columns are permuted to lie in a block before the search for pivots begins, whereas MA42 limits the amount of swapping of rows and columns by holding the positions of the fully summed variables and delaying permuting the pivot rows and columns into a block until all the pivots following an assembly have been chosen. Furthermore, since our numerical experiments show that the cost of the calls to the Level 1 BLAS kernels is much less than the total cost of the Level 3 BLAS calls (see Table 5.4), using Level 2 BLAS in place of Level 1 BLAS would not have a dramatic effect on the total factorization time and so we have not used the Level 2 BLAS implementation in our numerical experiments.

Performing additional assemblies before choosing pivots will lead to an increase in the average and maximum front sizes. The number of operations used to perform the matrix factorization will also rise, with many operations being performed on
zeros. The real storage required to hold the matrix factors will increase but, since fewer records will be written to the buffers, the repetition of the storage of the row and column indices will be reduced and the integer storage will consequently decrease.

There will also be effects on the solution phase. In the forward elimination, if \( \text{nrhs} > 1 \) (respectively, \( \text{nrhs} = 1 \)), the interior dimension of the calls to .GEMM (respectively, .GEMV) will increase. The interior dimension for the back substitution is \( \text{frnt} - r \), where \( \text{frnt} \) is the order of the frontal matrix and \( r \) the number of pivots chosen. Our new strategy will lead to an increase in \( \text{frnt} \) and in \( r \) although, in general, the increase in \( \text{frnt} \) will be greater than the increase in \( r \). Therefore, at most stages of the back substitution, the interior dimension will also increase. During the forward elimination and back substitution there will be a smaller number of calls to the Level 2 routine .TPSV, but the order of the matrix in each call will increase. Fewer records will be written to the buffers and, as a result, the time taken by the use of direct addressing during the solution phase will decrease. Since the amount of data which must be copied from the partial solution matrix into the arrays used for direct addressing is related to the number of right-hand sides, the time saved will depend on the number of right-hand sides, \( \text{nrhs} \).

4 The reuse of cache

In this section, we discuss the performance of BLAS kernels on cache-based machines. We present a very simple model for such machines with a multiply-add pipe and derive a formula that gives an upper bound on the performance of the Level 3 BLAS routine .DGEMM in terms of a number of parameters that characterize the machine. This result is compared with the observed performance of a Silicon Graphics Power Challenge XL with 75 MHz R8000 processors.

In our model, we assume that the machine has a clock speed of \( C \) MHz and that, if data is in the cache, \( f \) floating-point multiply-add pairs can be performed in each clock period. We also suppose that the size of the cache line is \( c \) words and that the latency of the cache is \( l \) clocks. We assume that the memory to cache operations cannot be overlapped with the floating-point operations (the cache is a blocking cache), although after the first word of the cache line is accessed computation can be overlapped with the transfer of subsequent words into the cache line.

Now consider using the Level 3 BLAS routine .DGEMM to perform the operation

\[
C \leftarrow \alpha AB + \beta C,
\]

where \( A \) and \( B \) are matrices of order \( m \times r \) and \( r \times m \), respectively. We are interested in the case where \( m \gg r \) and \( m \) is sufficiently large that \( C \) will not fit in the cache.

The number of operations required by (4.1) is \( rm^2 \) floating-point multiply-add pairs plus a further \( m^2 + mr \) floating-point multiplications. The total number of memory to cache operations is \( m^2 + 2mr \). In practice, this is likely to be an underestimate because it may be necessary to load \( A \) and/or \( B \) from memory to cache several times during the operation. Thus the estimate we derive here for the speed of the operation will be greater than that actually observed.
The time (in clocks) taken for the memory to cache operations is
\[(m^2 + 2mr)l/c.\]
The time (in clocks) taken for the floating-point operations is
\[(rm^2 + m^2 + mr)/f.\]
We then estimate the speed of \( \texttt{GEMM} \) (in Mflops) to be
\[C((2r + 1)m^2 + nr)/[(m^2 + 2mr)l/c + ((r + 1)m^2 + nr)/f].\]
That is,
\[fC((2r + 1)m^2 + nr)/[m^2(lf/c + r + 1) + nr(2lf/c + 1)].\]
Using our assumption \( m \gg r \), this simplifies to
\[2fC(r + 1/2)/(lf/c + r + 1).\]
For the Power Challenge workstation with 75 MHz R8000 processors and using double precision arithmetic the parameters have the following values:
\[C = 75,\]
\[f = 2,\]
\[c = 16,\]
\[l \approx 56.\]
This leads to an approximate speed of \( 300(r + 1/2)/(r + 8) \) Mflops for the \( \texttt{Dgemm} \) operation with interior dimension \( r \). In Figure 1 the estimated and observed speeds of \( \texttt{Dgemm} \) (in Mflops) are plotted against the interior dimension \( r \). For these results, \( m = 1000 \) was used.
Using similar analysis, we can estimate the speed of a rank-one update (\( \texttt{Ger} \)) to be \( 300/8 = 37.5 \) Mflops. Note that this is less than the estimated speed of 50 Mflops which is given by our \( \texttt{Dgemm} \) formula with \( r = 1 \).

5 The performance of the modified frontal code
In this section, we illustrate the effects of using the Level 3 BLAS enriched version of the frontal code \texttt{MA42} when solving a range of problems arising from real engineering and industrial applications. We first present results for two examples which arise from groundwater flow calculations undertaken by AEA Technology. Although practical applications can often call for significantly larger models, these problems are typical of the problems which AEA Technology wants to solve using its code \texttt{NAMMU} (Hartley, Jackson and Watson, 1996). \texttt{NAMMU} uses a frontal solver and it is important that the frontal solver is as efficient as possible. The first problem, \texttt{GFLOW2D}, is a two-dimensional coupled groundwater flow salt transport calculation. The problem has 20000 9-noded quadrilateral elements with a total of 80200 degrees of freedom. The second problem, \texttt{GFLOW3D}, is a three-dimensional
5 THE PERFORMANCE OF THE MODIFIED FRONTAL CODE

Figure 1: The estimated (continuous line) and observed speeds (stars) of DGEMM as a function of the interior dimension (rank of update) on an SGI Power Challenge workstation.

groundwater flow problem with pressure interpolated using a mixture of 27-noded triquadratic brick elements and 18-noded prism elements. The problem has 8820 elements with 73943 degrees of freedom. The results quoted in Table 5.1 were obtained using a Silicon Graphics Power Challenge XL with 4 75MHz R8000 processors and a cache size of 4 Mbytes, running IRIX 6.2. All runs were performed on a single processor using double precision arithmetic and the vendor-supplied BLAS.

In all the tables of results in this section in which the number of floating-point operations ("flops") are quoted, we count all operations (+,-,*,/) equally and assume that there are no zeros in the frontal matrices. All CPU timings are given in seconds.

It is clear from the results presented in Table 5.1 for \( r_{\min} = s \) and \( r_{\min} = 1 \) that there are considerable benefits to be gained from the standard MA42 strategy of delaying the elimination of pivots until all possible pivots following an assembly have been chosen. The benefits are greater for the three dimensional problem than for the two dimensional problem. The reason for this is that each of the three dimensional elements has significantly more degrees of freedom. This means that the number of
variables which become fully summed at each stage tends to be larger, resulting in larger pivot blocks and better performance of the BLAS kernel -GEMM when updating the frontal matrix.

It is also clear there are additional benefits to be gained from the Level 3 BLAS enrichment modification. In both two and three dimensions, the operation count and the total factorization time does not appear to be very sensitive to the value of \( r_{\text{min}} \). This suggests that, in practice, it is not necessary to choose the value carefully and it is likely that good performance will be achieved for a wide range of problems with values for \( r_{\text{min}} \) of about 15 and 40 for two and three-dimensional problems, respectively.

We now present, in more detail, results for test problems from a range of other application areas. A brief description of each of the problems is given in Table 5.2. For these problems only the sparsity pattern of the matrix was available and values for the matrix entries were generated using the Harwell Subroutine Library pseudo-random number generator FA04. The experimental results in Tables 5.3 and 5.4 were obtained on a 6 processor Silicon Graphics Power Challenge with the MIPS R10000 chip running at 195 MHz. The runs were performed on a single processor and again double precision arithmetic and the vendor-supplied BLAS were used. In each case, the elements were preordered using MC43 before the frontal solver was used.

In Table 5.3, the size of the largest pivot block used, the maximum front size, the total number of floating-point operations for factorizing the matrix, and the real and integer storage are shown for \( r_{\text{min}} = s \) and for values of \( r_{\text{min}} \) in the range 1 to 40. The real storage is for holding both the PL and the UQ factors (although, in practice, MA42 only requires PL to be stored if the user wishes to solve for subsequent right-hand sides or to solve transpose systems \( A^T X = B \)). It is apparent that modest increases in \( r_{\text{min}} \) have little effect on the size of the largest pivot block and on the
maximum front size, and that the real storage requirement and the operation count grow slowly with $r_{\text{min}}$. However, since large values of $r_{\text{min}}$ reduce the repetition of the storage of the row and column indices, increasing $r_{\text{min}}$ can give substantial savings in the amount of integer storage used. Conversely, if only single pivots are chosen ($r_{\text{min}} = s$), there is much repetition in the integer storage.

Table 5.4 presents the CPU times for the calls to the Level 1 and Level 3 BLAS kernels, and the total time for the matrix factorization, together with the time taken to solve for 1, 2, and 10 right-hand sides. The total factorization time and the solve times include all the overheads for the out-of-core working. We again observe that if no Level 3 BLAS are used ($r_{\text{min}} = s$), the factorization is significantly more expensive than if the frontal matrix is updated at each stage using as many pivots as are available (that is, as in the standard version of MA42, $r_{\text{min}} = 1$). In the latter case, the calls to the Level 1 BLAS kernels account for a small part of the total factorization cost. As $r_{\text{min}}$ is increased, the Level 1 BLAS account for a larger proportion of the factorization time until a point is reached where the savings in the Level 3 BLAS time is more than offset by the increase in the Level 1 BLAS time. The value of $r_{\text{min}}$ at which this occurs is problem-dependent, but our results suggest that, in general, it is advantageous to use a value of about 16. However, if we want to solve for a large number of right-hand sides, it can be beneficial to use an even larger value of $r_{\text{min}}$.

The results in Table 5.4 were all obtained on an SGI Power Challenge machine. We have also performed some experiments on a subset of our test problems on an IBM RS/6000 3BT and on a single processor of a CRAY J932. The results are given in Tables 5.5 and 5.6, respectively. In each case, the vendor-supplied BLAS are used. We see that, on the RS/6000, there are considerable savings to be made by not forcing all pivot blocks to be of size 1, and further modest savings in the factorization and solve times can result from choosing $r_{\text{min}}$ to be greater than 1. The Level 1 BLAS perform well on the CRAY and this is reflected in our results since, on this machine, the difference between the times for factorizing the matrix with $r_{\text{min}} = s$ and $r_{\text{min}} \geq 1$ are less significant. However, because of the significant savings in both the time taken to read the integer data from the direct access file and the time used by the direct addressing in the solution phase, the solve times are substantially reduced by allowing $r_{\text{min}} \geq 1$.
## 5 THE PERFORMANCE OF THE MODIFIED FRONTAL CODE

<table>
<thead>
<tr>
<th>Identifier</th>
<th>( r_{\text{min}} )</th>
<th>Largest pivot block</th>
<th>Maximum front size</th>
<th>Factor flops ((\times 10^4))</th>
<th>Storage (Kwords)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMAGE02</td>
<td>( s )</td>
<td>1</td>
<td>1453</td>
<td>55910</td>
<td>41808 41892</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>32</td>
<td>1453</td>
<td>55910</td>
<td>41808 3496</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>30</td>
<td>1453</td>
<td>55952</td>
<td>41826 3128</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>45</td>
<td>1458</td>
<td>56462</td>
<td>42033 1702</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>54</td>
<td>1474</td>
<td>57082</td>
<td>42275 1074</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>54</td>
<td>1484</td>
<td>57392</td>
<td>42397 912</td>
</tr>
<tr>
<td>AEAC5081</td>
<td>( s )</td>
<td>1</td>
<td>154</td>
<td>202</td>
<td>1431 1456</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>12</td>
<td>154</td>
<td>202</td>
<td>1431 243</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>16</td>
<td>157</td>
<td>205</td>
<td>1441 129</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>26</td>
<td>166</td>
<td>223</td>
<td>1502 86</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>42</td>
<td>182</td>
<td>245</td>
<td>1573 58</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>50</td>
<td>190</td>
<td>264</td>
<td>1630 53</td>
</tr>
<tr>
<td>TRDHEIM</td>
<td>( s )</td>
<td>1</td>
<td>277</td>
<td>961</td>
<td>7551 5232</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>36</td>
<td>277</td>
<td>961</td>
<td>7551 597</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>36</td>
<td>277</td>
<td>961</td>
<td>7551 597</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>42</td>
<td>289</td>
<td>985</td>
<td>7661 550</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>61</td>
<td>308</td>
<td>1073</td>
<td>8039 469</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>68</td>
<td>315</td>
<td>1128</td>
<td>8248 452</td>
</tr>
<tr>
<td>CRPLAT2</td>
<td>( s )</td>
<td>1</td>
<td>538</td>
<td>5065</td>
<td>13012 13089</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>19</td>
<td>539</td>
<td>5065</td>
<td>13012 2133</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>24</td>
<td>545</td>
<td>5141</td>
<td>13116 1101</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>27</td>
<td>550</td>
<td>5221</td>
<td>13225 754</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>44</td>
<td>568</td>
<td>5466</td>
<td>13553 399</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>49</td>
<td>574</td>
<td>5552</td>
<td>13662 346</td>
</tr>
<tr>
<td>OPT1</td>
<td>( s )</td>
<td>40</td>
<td>984</td>
<td>10764</td>
<td>16466 16215</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>40</td>
<td>984</td>
<td>10764</td>
<td>16466 1190</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>39</td>
<td>984</td>
<td>10771</td>
<td>16471 1163</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>45</td>
<td>996</td>
<td>10875</td>
<td>16573 863</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>59</td>
<td>1012</td>
<td>11204</td>
<td>16800 628</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>68</td>
<td>1016</td>
<td>11268</td>
<td>16939 565</td>
</tr>
<tr>
<td>TSYL201</td>
<td>( s )</td>
<td>62</td>
<td>543</td>
<td>10741</td>
<td>20919 20925</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>62</td>
<td>543</td>
<td>10741</td>
<td>20919 1021</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>62</td>
<td>543</td>
<td>10743</td>
<td>20921 1017</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>62</td>
<td>551</td>
<td>10759</td>
<td>20944 985</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>61</td>
<td>572</td>
<td>11202</td>
<td>21369 541</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>73</td>
<td>579</td>
<td>11257</td>
<td>21424 534</td>
</tr>
</tbody>
</table>

Table 5.3: Storage requirements for different pivot block sizes. \( r_{\text{min}} = s \) denotes all pivot blocks are of size 1.
### Table 5.4: Performance for different pivot block sizes.

$r_{\text{min}} = s$ denotes all pivot blocks are of size 1. $\text{nrhs}$ denotes the number of right-hand sides.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>$r_{\text{min}}$</th>
<th>Factor Time (seconds)</th>
<th>Solve Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total</td>
<td>BLAS 3</td>
</tr>
<tr>
<td>RAMAGE02</td>
<td>s 2845.7</td>
<td>0.00</td>
<td>2724.80</td>
</tr>
<tr>
<td></td>
<td>547.8</td>
<td>433.56</td>
<td>5.14</td>
</tr>
<tr>
<td></td>
<td>527.4</td>
<td>411.26</td>
<td>5.42</td>
</tr>
<tr>
<td></td>
<td>447.1</td>
<td>326.42</td>
<td>11.21</td>
</tr>
<tr>
<td></td>
<td>422.5</td>
<td>292.18</td>
<td>19.18</td>
</tr>
<tr>
<td></td>
<td>442.4</td>
<td>300.11</td>
<td>25.66</td>
</tr>
<tr>
<td>AEAC5081</td>
<td>s 3.5</td>
<td>0.00</td>
<td>2.26</td>
</tr>
<tr>
<td></td>
<td>1.6</td>
<td>0.97</td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>1.5</td>
<td>0.85</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>1.6</td>
<td>0.85</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>1.9</td>
<td>0.91</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>0.96</td>
<td>0.38</td>
</tr>
<tr>
<td>TRDHEIM</td>
<td>s 17.3</td>
<td>0.00</td>
<td>10.75</td>
</tr>
<tr>
<td></td>
<td>7.8</td>
<td>3.82</td>
<td>0.49</td>
</tr>
<tr>
<td></td>
<td>7.7</td>
<td>3.84</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td>16.7</td>
<td>3.81</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>32.8</td>
<td>3.73</td>
<td>1.21</td>
</tr>
<tr>
<td></td>
<td>38.9</td>
<td>3.76</td>
<td>1.60</td>
</tr>
<tr>
<td>CRPLAT2</td>
<td>s 235.6</td>
<td>0.00</td>
<td>212.00</td>
</tr>
<tr>
<td></td>
<td>57.0</td>
<td>46.52</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td>43.4</td>
<td>31.64</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>16.3</td>
<td>28.20</td>
<td>1.47</td>
</tr>
<tr>
<td></td>
<td>32.8</td>
<td>24.97</td>
<td>3.12</td>
</tr>
<tr>
<td></td>
<td>38.2</td>
<td>24.26</td>
<td>3.76</td>
</tr>
<tr>
<td>OPT1</td>
<td>s 538.4</td>
<td>0.00</td>
<td>493.86</td>
</tr>
<tr>
<td></td>
<td>92.7</td>
<td>71.24</td>
<td>2.38</td>
</tr>
<tr>
<td></td>
<td>92.0</td>
<td>70.53</td>
<td>2.43</td>
</tr>
<tr>
<td></td>
<td>83.8</td>
<td>61.43</td>
<td>3.67</td>
</tr>
<tr>
<td></td>
<td>32.1</td>
<td>54.70</td>
<td>5.55</td>
</tr>
<tr>
<td></td>
<td>38.2</td>
<td>53.03</td>
<td>7.49</td>
</tr>
<tr>
<td>TSYL201</td>
<td>s 606.1</td>
<td>0.00</td>
<td>555.63</td>
</tr>
<tr>
<td></td>
<td>82.7</td>
<td>58.30</td>
<td>3.08</td>
</tr>
<tr>
<td></td>
<td>83.4</td>
<td>58.63</td>
<td>3.02</td>
</tr>
<tr>
<td></td>
<td>83.3</td>
<td>57.86</td>
<td>3.17</td>
</tr>
<tr>
<td></td>
<td>75.7</td>
<td>50.01</td>
<td>6.41</td>
</tr>
<tr>
<td></td>
<td>75.9</td>
<td>49.77</td>
<td>6.42</td>
</tr>
</tbody>
</table>
### Table 5.5: Performance for different pivot block sizes on an IBM RS/6000.

$r_{\text{min}} = s$ denotes all pivot blocks are of size 1. nrhs denotes the number of right-hand sides.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>$r_{\text{min}}$</th>
<th>Factor Time (seconds)</th>
<th>Solve Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r_{\text{min}}$</td>
<td>nrhs=1</td>
<td>nrhs=2</td>
</tr>
<tr>
<td>AEAC5081</td>
<td>s</td>
<td>9.9</td>
<td>0.51</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3.3</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2.8</td>
<td>0.09</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>3.1</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>3.5</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>4.0</td>
<td>0.14</td>
</tr>
<tr>
<td>CRPLAT2</td>
<td>s</td>
<td>216.0</td>
<td>3.73</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>69.4</td>
<td>1.27</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>60.2</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>58.6</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>62.5</td>
<td>1.01</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>63.7</td>
<td>0.89</td>
</tr>
<tr>
<td>OPT1</td>
<td>s</td>
<td>455.6</td>
<td>4.19</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>115.5</td>
<td>1.39</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>115.3</td>
<td>1.47</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>107.1</td>
<td>1.16</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>110.1</td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>112.3</td>
<td>1.21</td>
</tr>
</tbody>
</table>

### Table 5.6: Performance for different pivot block sizes on CRAY J932.

$r_{\text{min}} = s$ denotes all pivot blocks are of size 1. nrhs denotes the number of right-hand sides.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>$r_{\text{min}}$</th>
<th>Factor Time (seconds)</th>
<th>Solve Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$r_{\text{min}}$</td>
<td>nrhs=1</td>
<td>nrhs=2</td>
</tr>
<tr>
<td>AEAC5081</td>
<td>s</td>
<td>5.3</td>
<td>0.79</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.4</td>
<td>0.21</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>3.9</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>3.9</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>3.9</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>4.0</td>
<td>0.15</td>
</tr>
<tr>
<td>CRPLAT2</td>
<td>s</td>
<td>59.9</td>
<td>4.99</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>54.8</td>
<td>1.23</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>47.6</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>45.0</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>44.1</td>
<td>0.62</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>44.7</td>
<td>0.63</td>
</tr>
<tr>
<td>OPT1</td>
<td>s</td>
<td>109.6</td>
<td>5.40</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>94.1</td>
<td>1.18</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>93.2</td>
<td>1.17</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>84.7</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>80.2</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>81.2</td>
<td>0.79</td>
</tr>
</tbody>
</table>
6 Concluding remarks

We have shown how the frontal method can be implemented to enhance the use of Level 3 BLAS and, using a range of practical problems, we have illustrated that, on cache-based machines, this leads to very good performance in terms of MFlops. The implementation of the frontal method which uses only pivot blocks of size 1 does reasonably well on vector machines but performs poorly on cache-based machines. The plot given in Figure 1 of the speed of DGEMM against the interior dimension indicates that the choice of the minimum pivot block size parameter is not crucial. This is important from a practical point of view since it is possible to get good performance without having to optimize the parameter from run to run.

A disadvantage of frontal schemes is that they usually perform many more operations than are necessary for the numerical factorization and the factors normally have many more entries than those obtained by other techniques. This is illustrated by Duff and Scott (1996a). However, in practice we have found that the convenience of being able to specify memory requirements in advance and being able to hold the factors out-of-core more than compensates for this. As a result, we have made extensive use of MA42 and its predecessor, MA32, for more than 15 years. For problems in three dimensions, other techniques are clearly needed, but for two dimensional problems, ease of use and performance mean the frontal method remains our method of choice.

Clearly, it is important that we implement our algorithms to make effective use of machines which have a hierarchical memory structure. The techniques which we have discussed in this paper for making better reuse of data in the cache are applicable to other direct solvers.

7 Availability of software

MA42 and the element ordering routine MC43 are included in Release 12 of the Harwell Subroutine Library. A complex frontal solver, ME42, as well as a frontal solver for symmetric positive-definite systems, MA62, are also available. These codes are all written in standard Fortran 77; a Fortran 90 version of MA42 is also included in Release 12 of the HSL. Anyone wishing to use the codes should contact the HSL Manager: Dr. S. J. Roberts, Harwell Subroutine Library, AEA Technology, Building 552, Harwell, Oxfordshire, OX11 0RA, England, tel. +44 (0) 1235 434714, fax +44 (0) 1235 434136, or e-mail scott.roberts@aat.co.uk, who will provide details of price and conditions of use.

Acknowledgements

We are grateful to Alison Ramage of the University of Strathclyde for supplying the test problem RAMAGE02, and to Christian Damhaug of DNVR, Norway, for problems CRPLAT2, OPT1, TRDHEIM, and TSYL201. We are also extremely grateful to our colleague Jean-Yves L'Excellent at CERFACS for obtaining the results presented in Tables 5.3 and 5.4.
REFERENCES

References


