Science & Technology
Facilities Council

Technical Report
RAL-TR-2011-024

HSL_MA97: a bit-compatible multifrontal code for sparse symmetric systems

JD Hogg, JA Scott

November 2011

RAL Library
STFC Rutherford Appleton Laboratory
R61
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358- 6254

`HSL_MA97`: a bit-compatible multifrontal code for sparse symmetric systems

Jonathan D. Hogg and Jennifer A. Scott[1]

ABSTRACT

The multifrontal method is widely used for the numerical solution of large sparse symmetric linear systems of equations. In this report, we discuss the design and implementation of a new multifrontal code `HSL_MA97`. Our motivation for the new code is discussed along with the key design features. `HSL_MA97` is for real and complex problems and is designed to be efficient for both positive definite and indefinite systems. `HSL_MA97` can be run in serial or in a shared memory environment using OpenMP. An important feature is that in parallel it offers bit-compatible solutions. Numerical results are presented for a range of problems from practical applications and comparisons are made with existing codes. Future plans for `HSL_MA97` are outlined.

Keywords: sparse symmetric linear systems, indefinite systems, direct solver, multifrontal method, OpenMP, bit-compatible.

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

---

[1] Computational Science and Engineering Department, STFC Rutherford Appleton Laboratory,
Harwell Oxford, Oxfordshire, OX11 0QX, UK.
Emails: jonathan.hogg@stfc.ac.uk and jennifer.scott@stfc.ac.uk
Work supported by EPSRC grants EP/E053351/1 and EP/I013067/1.
Report available from `http://www.cse.stfc.ac.uk/nag/reports.shtml`.

November 29, 2011

1 Background and motivation

The multifrontal method for the numerical solution of sparse linear systems of equations is a highly successful method that has been in widespread use in many scientific and engineering applications for almost 30 years. The term *multifrontal* was introduced by Duff and Reid [19, 20] since they developed the method as a generalisation of the *frontal* method of Irons [34], although the essential idea of the multifrontal method had first appeared in 1973 in the *generalised element* method of Speelpenning (Speelpenning's unpublished manuscript appeared (unmodified) 5 years later as a technical report [47]). The element merge model of Eisenstat, Schultz and Sherman [23] also uses the notion of generalised elements and shares basic features with the multifrontal method.

The multifrontal method is not restricted to problems arising from finite-element applications. The first multifrontal code `MA27` developed by Duff and Reid [19] in 1982 for the Harwell Subroutine Library (which subsequently became the HSL software library [33] `http://www.hsl.rl.ac.uk/`) was for problems in assembled form. Moreover, very importantly, `MA27` incorporated numerical pivoting so that it could be used for symmetric positive-definite and symmetric indefinite systems [20]. Duff and Reid went on to generalise the multifrontal method further to unsymmetric systems with a (almost) symmetric sparsity pattern [21] and Davis and Duff [12] developed an unsymmetric multifrontal algorithm. In this report, we restrict our attention to the symmetric case.

Despite its age, `MA27` remains an extremely widely-used code. This is partly because, since 2000, it has been freely available worldwide for non-commercial use within the HSL Archive (see `http://www.hsl.rl.ac.uk/archive`). It has also been incorporated into a number of commercial packages (including the nonlinear optimization package KNITRO) and `MA27` is one of the symmetric indefinite linear solvers that the Ipopt software package [49] for large-scale nonlinear optimization offers an interface for. However, as we would expect, over the years as new developments have taken place, a number of other packages that implement multifrontal algorithms for symmetric linear systems have been developed. Within the HSL software library these include the well-known `MA57` package [15] and, more recently, `HSL_MA77` [39, 40]. The former is included within Matlab as the default sparse solver for indefinite systems and Ipopt also offers an `MA57` solver interface. `HSL_MA77` is an out-of-core code (that is, it allows the matrix factor and most of the work arrays used by the code to be held in files on disk) and is thus primarily intended for the solution of very large-scale problems. Other multifrontal codes include the Boeing commercial code BCSEXT-LIB, MUMPS [4, 5], SPOOLES [8], TAUCS [44], and WSMP [26]. Versions of many of these packages are available for parallel computation (for example, MUMPS is designed primarily for distributed-memory machines while there are shared-memory versions of TAUCS and WSMP). A comparison of the serial versions of these packages is given in [25].

With the availability of this range of multifrontal-based linear solvers the obvious question is: why develop another multifrontal code? There are a number of reasons behind our decision to write `HSL_MA97`, the principle one being that we require a code that we can use as the basis for further research and development work. Our plans include porting `HSL_MA97` to run on GPU-accelerated architectures and the design and implementation of new pivoting strategies for indefinite systems. Some of these plans are outlined in more detail in Section 7. Clearly, we could take as our starting point either of the existing HSL codes `MA57` or `HSL_MA77` (we rule out the earlier code `MA27` since `MA57` was itself designed to supersede `MA27`). From our point of view, a disadvantage of `MA57` is that it is written in Fortran 77 (there is a Fortran 95 version, `HSL_MA57`, but this is essentially a Fortran 95 wrapper for `MA57` with some additional functionality offered). Furthermore, the design of `MA57` (in particular, the factorization phase) is less modular than is ideal, with all the factorization being performed by a single long and complicated subroutine `MA57OD`. This makes the code quite difficult to modify and develop further. It has also been reported in numerical experiments that, on a few problems, the performance of `MA57` can be unacceptably slow compared with other multifrontal codes (see Section 4 and the results given in [24, 25]). For these reasons, we have chosen not to use `MA57` as a starting point. As for `HSL_MA77`, we have already observed that it is an out-of-core code. While it does offer an option of working in core (in which case, arrays in main memory replace the use of files on disk), this option was added later and, not being part of the original design, we anticipate that it will not perform as efficiently as a code designed from the start to work in core. In particular, the in-core version of `HSL_MA77` involves additional overheads of copying between the internal data structures that can be avoided by redesigning the implementation and handling of the factors. We also want to design a solver that is able to take advantage of our new implementation of the analysis phase `HSL_MC78` [31] and to include real and complex versions within a single package.

The existing HSL multifrontal solvers were all designed as serial codes. Some parallel performance can be achieved by `MA57` and `HSL_MA77` through the use of multithreaded BLAS. One of the aims for `HSL_MA97` is to obtain improved parallel performance through the use of tree-level and node-level parallelism. The code employs OpenMP and so can be run in parallel on a shared memory machine. A key design criterion was imposed that differentiates `HSL_MA97` from other parallel sparse direct solvers, that of bit-compatibility. That is to say, regardless of the number of threads used by the solver, the factorization and solutions produced are bit-for-bit identical. This property aids users in debugging and increases their confidence in results that are then repeatable. Moreover, it has recently been requested by a number of users of the HSL library.

The rest of this report is organised as follows. Section 2 presents a brief overview of the multifrontal method (further details may be found in, for example, [16] and [38]). In Section 3, we describe the design of `HSL_MA97` and the features offered to the user in the first release. Numerical results are presented in Section 4. These include comparisons with HSL codes and with WSMP and PARDISO (a parallel solver not based on a multifrontal approach) [46]. Finally, in Section 7, we discuss our future plans for `HSL_MA97`.

2 Overview of the multifrontal method

Our interest is in efficiently solving linear systems of the form

$$AX = B$$

where the $n \times n$ matrix A is sparse, symmetric and possibly indefinite. Direct methods such as the frontal and multifrontal methods split the solution process into a number of phases: order, analyse, factorize and solve. The order phase selects a pivot sequence (elimination order) that is designed to minimize the number of entries in the matrix factors. The analyse phase takes the pivot sequence and the sparsity pattern of A and uses it to construct data structures in preparation for the numerical factorization. In some implementations (including the original multifrontal code `MA27` and `MA57`), the order and analyse phases are combined into one. The factorization phase forms the matrix decomposition

$$A = (PL)D(PL)^T,$$

where P is a permutation matrix, L is a unit lower triangular matrix, and D is a block diagonal matrix with blocks of size 1×1 and 2×2. Note that if A is positive definite, D is diagonal with positive diagonal entries and, in this case, $\hat{L}$ is defined to be $\hat{L} = LD^{1/2}$, giving the Cholesky factorization

$$A = P\hat{L}(P\hat{L})^T.$$

Finally, the solve phase uses the matrix factors to perform forward substitution where

$$PLY = B$$

is solved for Y, then (in the indefinite case) the block diagonal system

$$DZ = Y$$

is solved for Z, followed by backward substitution where

$$(PL)^T X = Z$$

is solved for X. It is standard practice for direct solvers to allow several factorizations of matrices with the same sparsity pattern to follow the analyse phase and for more than one call to the solve phase after the factorize phase. Some packages (including `MA57` and `HSL_MA77`) offer an option to solve for more than one right-hand-side at once as this is more efficient (particularly in the out-of-core case).

In the multifrontal method, the factorization of A proceeds using a succession of assembly operations of small dense matrices (the so-called *frontal matrices*), interleaved with partial factorizations of these matrices. Assume that a pivot sequence (that is, an elimination order) has been chosen. For each pivot in turn, the multifrontal method first assembles all the rows that contain the pivot. This involves merging the index lists for these rows (that is, the lists of columns involved) into a new list, setting up a frontal matrix of order the size of the new list, and then adding the rows into this frontal matrix. A row of A that has been added to the frontal matrix is said to be *assembled*; rows that have not yet been assembled are referred to as *unassembled*. A partial factorization of the frontal matrix is performed (that is, the pivot and any other variables that are only involved in the assembled rows are eliminated). The computed columns of the matrix factor L are not needed again until the solve phase and so can be stored while the reduced matrix (the *generated element* or *contribution block*), together with a list of the variables involved, is stored separately. At the next and subsequent stages, not only must unassembled rows of A that contain the pivot be assembled into the frontal matrix but so too must any generated elements that contain the pivot. The basic multifrontal algorithm is summarised in Figure 2.1.

```
do for each pivot in the given pivot sequence
    if the pivot has not yet been eliminated
        assemble unassembled rows of $A$ and generated elements
            that contain the pivot into a frontal matrix;
        perform a partial factorization;
        store the generated element
    end if
end do
```

Figure 2.1: Basic multifrontal factorization

At each stage, the frontal matrix can be expressed in the form

$$F = \begin{pmatrix} F_{11} & F_{21}^T \\ F_{21} & F_{22} \end{pmatrix}, \tag{2.1}$$

where F_{11} and F_{21} are *fully summed*, that is, all the entries in the corresponding part of the overall matrix have been assembled, while F_{22} is not yet fully summed. If F_{11} has order p and q pivots can be chosen stably from F_{11} (if A is positive definite, p pivots can be chosen in order down the diagonal but, in the indefinite case, if $q < p$ are selected, $p - q$ pivots are said to be *delayed*), the partial factorization of F takes the form

$$F = Q \begin{pmatrix} L_1 & 0 \\ L_2 & I \end{pmatrix} \begin{pmatrix} D_1 & 0 \\ 0 & F_S \end{pmatrix} \begin{pmatrix} L_1^T & L_2^T \\ 0 & I \end{pmatrix} Q^T. \tag{2.2}$$

where Q is a permutation matrix of the form

$$Q = \begin{pmatrix} Q_1 & 0 \\ 0 & I \end{pmatrix}$$

with Q_1 having order p. If A is positive definite, L_1 is lower triangular and $D_1 = I$, the identity matrix; if A is indefinite, L_1 is a unit lower triangular matrix of order q, and D_1 is a block diagonal matrix of order q. The matrices Q_1, L_1, and D_1 are not required again until the forward and back substitution phases and so may be stored, while the Schur complement F_S is the generated element.

The assembly operations can be recorded as a tree, termed an *assembly* tree. The partial factorization of the frontal matrix at a node v in the tree can be performed once the partial factorizations at all the nodes belonging to the subtree rooted at v are complete. If the nodes of the tree are ordered using a depth-first search, the generated elements required at each stage are the most recently generated ones of those so far unused. This makes it convenient to use a stack (the so-called *multifrontal stack*) for temporary storage during the factorization. This alters the pivot sequence, but the arithmetic is identical apart from the round-off effects of reordering the assemblies and the knock-on effects of this.

3 The design of `HSL_MA97`

In this section, we briefly discuss the design of `HSL_MA97`, the user interface and the options that it offers. The package covers the following cases:

1. A is real, symmetric and positive-definite or complex Hermitian and positive-definite.
2. A is real or complex, symmetric and indefinite or complex Hermitian and indefinite.

The efficiency of `HSL_MA97` is dependent on the elimination order used. During the past 20 years or so, considerable research has gone into the development of algorithms that generate good pivot sequences. The original HSL multifrontal code `MA27` used the minimum degree ordering [48]. Minimum degree and variants including approximate minimum degree (AMD) [2, 3] and multiple minimum degree [37], perform well on many small and medium-sized problems (typically, those of order less than 50,000). However, nested dissection has been found to work better for very large problems, particularly those from 3D discretizations (see, for example, the results of [24]). Many direct solvers now offer users a choice of orderings, including either their own implementation of nested dissection or an explicit interface to the generalised multilevel nested-dissection routine `METIS_NodeND` from the METIS graph partitioning package [35, 36].

As well as allowing the user to input the pivot order, `HSL_MA97` offers the use of either AMD or nested dissection orderings through an internal call to the `HSL_MC68` package. Additionally, it implements the heuristic used by `MA57` and described in [22] for automatically choosing one of these orderings based on the size and sparsity of the matrix involved (this is the default option).

The main procedures that are called by the user of `HSL_MA97` are as follows:

- `ma97_analyse` accepts the sparsity pattern of the lower triangular part of A in compressed sparse column format. It optionally checks the data for out-of-range entries (they are ignored) and duplicates (they will be summed during the factorization) then, if the user has not supplied a pivot order, computes a pivot order using `HSL_MC68`. Using the HSL package `HSL_MC78` [31], the sparsity pattern of the matrix is next analysed and the data structures for the factorization are prepared.
- `ma97_factor` uses the data structures set up by `ma97_analyse` to compute a sparse factorization using the multifrontal algorithm. The user must set a parameter `matrix_type` to indicate whether A is to be treated as positive-definite or indefinite. If A is positive-definite, no numerical pivoting is used and a Cholesky factorization is produced rather than a symmetric indefinite one. More than one call to `ma97_factor` may follow a call to `ma97_analyse`. This allows more than one matrix with the same sparsity pattern but different numerical values to be factorized without recalling `ma97_analyse`. An option is offered to scale the matrix. In this case, the factorization of the scaled matrix $\overline{A} = SAS$ is computed, where S is a diagonal scaling matrix. Scaling the matrix can significantly improve performance; this is discussed further in, for example, [30]. If the user already has a suitable scaling, this may be passed to the factorize phase. In Release 1 of `HSL_MA97`, the scaling offered is either a symmetrized version of `MC64` [17, 18] or an iterative procedure that attempts to make all the row and column norms of the matrix unity (see [45]).
- `ma97_solve` uses the computed factors generated by `ma97_factor` to solve systems $AX = B$ for one or more right-hand sides B. Multiple calls to `ma97_solve` may follow a call to `ma97_factor`.
- `ma97_finalise` should be called after all other calls are complete for a problem (including after an error return that does not allow the computation to continue). It deallocates the components of the derived data types used by the package.

In addition to the above procedures, `HSL_MA97` includes an number of optional routines as follows:

- `ma97_analyse_coord` may be used in place of `ma97_analyse` if the user has the matrix A held in coordinate format. Coordinate input format is offered since it is used by `MA57` and we were keen to make it easy for a user to switch from `MA57` to `HSL_MA97`.
- `ma97_enquire_posdef` may be called in the positive-definite case to obtain the pivots used.
- `ma97_enquire_indef` may be called in the indefinite case to obtain the pivot sequence used by the factorization and the entries of D^{-1}.
- `ma97_alter` may be called in the indefinite case to alter the entries of D^{-1}. Note that this means a $(PL)D(PL)^T$ factorization of A is no longer available. This facility is useful for computing a modified Cholesky factorization (see [11]) which is used, for example, in optimization in dealing with indefinite Hessian matrices in Newton methods.

Full details of the user interface are provided in the user documentation that accompanies the code.

3.1 Data structures

A traditional supernodal data structure is used to store the factors in `HSL_MA97`. Each supernode is represented by an instance of a derived type with components specifying the number of eliminated variables (q), and number of delays *into* the node. From the information on delays and the analyse phase data, the number of fully summed columns (p) and the number of nonzero rows (m) in the supernode can be determined. The derived type also stores the ordering of the columns after pivoting.

The floating-point values of L are stored in an $m \times p$ array using full storage. In the indefinite case, a $2 \times p$ array containing the values of D^{-1} is stored in memory after those of L. We note that, if a pivot is delayed, memory associated with that column is wasted.

Generated elements are stored (using full storage) on a stack until required. The stack is grown and shrunk dynamically as required.

3.2 The subtree factorization

The basic work unit for `HSL_MA97` is the factorization of a subtree. In the serial case, the entire factorization is treated as a single subtree factorization, while in the parallel case, a number of subtrees are factorized simultaneously.

An assembly tree ordered using a depth-first search has the property that all the nodes in a subtree are numbered consecutively. Thus we can describe a subtree by specifying only its start and end nodes. Such a range can also describe a forest (and indeed this is exploited by the code), but this can, without loss of generality, be treated as a number of independent subtree factorizations.

The task of a subtree factorization is to compute the entries of the factors L and D associated with the nodes within the subtree and to compute the generated element associated with the root of the subtree (unless it is a root of the assembly tree). To do this, contributions from generated elements of any child subtrees must be included.

At each node, the columns of the frontal matrix are divided into those that are fully summed and may be eliminated and those that are not (see (2.1)).The factorization proceeds as follows:

1. Assemble the fully summed columns, including both contributions from child nodes (including any delays) and from original rows of A.
2. Factorize the fully summed columns using the dense factorization kernel (see Section 3.3).
3. Calculate the contribution to the generated element from the fully summed columns.
4. Assemble the contributions from the child nodes into the generated element. Free the stack memory associated with child nodes.

We split computing the generated element into two steps (steps 3 and 4) as this is more efficient. If we were to assemble the contribution from the child nodes first, we would need to start by setting the entries of the generated element to zero. As the calculated contribution to each entry of the generated element is (symbolically) non-zero, by computing these contributions first, this is avoided.

Our out-of-core solver `HSL_MA77` was designed specifically for large-scale problems and, as such, aimed to minimise of the amount of main memory that it required. The penalty for this was additional copying that resulted from maintaining a single dedicated area of memory that it reuses for each frontal matrix. In `HSL_MA97`, we avoid unnecessary copying by storing values directly where they are required.The fully summed columns are stored in the data structure for L. In the indefinite case, this can result in an increased memory footprint for L as delayed columns are left in place and not overwritten. The generated element is stored on a stack.

We need to be able to copy from the generated element for a child node into the generated element of its parent. This cannot be done using a single stack. Instead, we use two stacks: one for generated elements from nodes on even-numbered levels of the tree, and one for generated elements on odd-numbered levels of the tree.

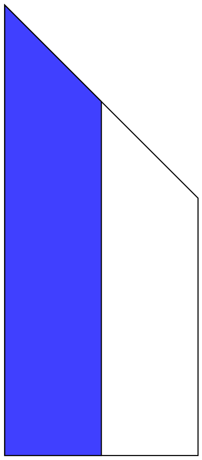
3.3 The dense factorization kernel: indefinite case

The development of the out-of-core solver `HSL_MA77` involved writing a separate package, `HSL_MA64` [43], to perform the partial factorization of the frontal matrices. A modified version was subsequently used within our indefinite DAG-based sparse solver `HSL_MA86` [27]; the dense factorization kernel used within `HSL_MA97` for the indefinite case is based upon this modification. In the first release of `HSL_MA97`, the static and relaxed pivoting options have been removed. To further simplify the kernel, full factor storage is used rather than the packed storage employed by `HSL_MA64` (note that as `HSL_MA64` was developed primarily for use within `HSL_MA77`, one of its original objectives was to limit memory requirements).

We observe that LAPACK cannot be used as it does not offer routines to perform partial factorizations. The LAPACK routine `_sytrf` could be used to factor the diagonal block but (at non-root nodes) this is insufficient for stability as it is not able to take into account the size of the entries in the off-diagonal block. The pivoting strategy used by `HSL_MA64` considers all the entries in the pivot column and is backward stable.

The use of full factor storage enables blocking to be implemented using the following recursive factorization scheme, which is cache agnostic. Given an $m \times p$ block to factorize (that is, p fully summed columns of the frontal matrix), divide the block in half, as shown in Figure 3.1. The factorization routine is called recursively on the left half, and the right half is updated using the generated factors. Any delays are swapped to the end (of the right half); the factorization routine is called on the right half, including

Figure 3.1: The recursive dense factorization



these delays. As the calls to the factorization routine are recursive, multiple levels of division in both the left and right halves occur in practice. Recursion stops once p is small ($p \leq 16$), or the recursion depth exceeds a maximum number of levels that depends on the node in the assembly tree and the number of fully summed columns in the frontal matrix. Our `HSL_MA64`-derived kernel is used to perform the lowest level factorization. Except at the top level of the recursion, pivot candidates are only examined once, limiting rescanning of the column to $O(\log n)$ times per delay.

For small blocks that fit entirely within the L1 cache, our kernel uses a fully right-looking algorithm that prefers 1×1 pivots to maximise instruction throughput. For larger blocks that do not fit entirely within L1 cache, it uses a mixed left-/right-looking algorithm that prefers 2×2 pivots that minimise communication. For small blocks, calls to the Level 1 and 2 BLAS routines are additionally replaced with in-line loops to avoid the overheads of function calls. In total, we found that these modifications gave a 10% improvement in overall performance of the factorize phase on small problems.

3.4 Update routine `ldsrk`

Both during the recursive factorization and while calculating the contribution to the generated element it is necessary to compute outer product sums of the form

$$E = E + L(LD)^T.$$

As E is symmetric, using a generic `_gemm` call performs twice as many operations and memory accesses as are necessary. To limit this inefficiency while still taking advantage of high-level BLAS, we have developed a subroutine `ldsrk` to perform this update operation.

Our initial implementation used a recursive subroutine to try and be cache agnostic. However, this proved difficult to parallelise effectively while maintaining bit-compatibility of the result. In particular, the load imbalance caused by the few large BLAS operations (which are good for serial performance) could not be overcome without dividing these into smaller operations, which substantially complicated the code and required complex tuning for good performance. Instead, a straightforward blocking of E is used, with blocks in the upper triangular part ignored.

Node-level parallelism is used within the `ldsrk` subroutine. Each block operation is assigned a separate task, with a synchronisation point to ensure all such tasks are complete before the return from `ldsrk`.

It is worth noting that to avoid preserving and carrying around the entries of LD, they are recalculated on the fly as required. Cached versions from previous block operations are used if available. As well as limiting the total memory footprint, this reduces the number of cache misses by only transferring L rather than L and LD from lower levels of memory.

We found that for small blocks (m or p being 1 or 2) using explicit inline code is more efficient than calling the (Goto or Intel) BLAS. We have thus implemented a wrapper for some of the calls to the BLAS routines `_gemm` and `_gemv` that exploits small block sizes.

3.5 The dense factorization kernel: positive definite case

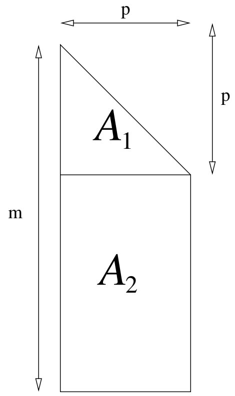
The dense factorization kernel for positive-definite matrices is written in a similar recursive fashion to that for the indefinite case. The main difference is that LAPACK and BLAS routines are called in place of the `HSL_MA64`-derived kernel. Following recursion, we factor a $m \times p$ block, with p small. This block is divided into a $p \times p$ block A_1 on the diagonal and a $(m-p) \times p$ block A_2 below it, as shown in Figure 3.2. The LAPACK routine `_potrf` (`_herk` in the Hermitian case) is called to perform the Cholesky factorization $A_1 = L_1 L_1^T$. Subsequently, the BLAS routine `_trsm` is used to calculate $L_2 = A_2 L_1^{-T}$.

A modified version of the `ldsrk` routine is implemented to perform the update operation

$$E = E + LL^T$$

in the same manner as for the LDL^T factorization.

Figure 3.2: The partitioning of a $m \times p$ block in the positive-definite kernel.



3.6 Tree-level parallelism

In our internal representation of the assembly tree, each root is stored as a child of a single virtual root. We can thus treat the assembly tree as having a single root when writing our algorithms.

Tree-level parallelism is exposed recursively. At each node i, the algorithm proceeds as follows:

(a) If the amount of work (the number of flops) associated with the subtree rooted at i is small ($< 10^5$ flops), the subtree factorization of Section 3.2 is used.

(b) Otherwise, a task is created for each child node (and the subtree rooted at that child). If there is only a small amount of work in consecutive children, they are merged into a single task. Once all child node tasks have been run (in parallel), the subtree factorization code is called to perform the relevant assembly and factorization operations at i.

3.7 Achieving bit-compatibility

To achieve bit-compatibility regardless of the number of threads, care must be taken to utilise the same blocking and arithmetic order.

With tree-level parallelism, bit-compatibility requires that during assembly the contributions from the children are always summed in the same order (as opposed to the order in which they were placed on the stack). This means that memory locality of the most recent child factorization cannot be exploited in all cases.

With node-level parallelism, bit-compatibility requires that the blocking used is independent of the number of available threads. Instead, only position in the assembly tree and size of the node can be used to determine the blocking to be used. This means that, in general, the number of blocks will be sub-optimal for the number of threads running: either threads will be idle, or smaller than required blocks will be used, resulting in BLAS calls of inefficient size.

3.8 Parallel solve

Our initial implementation of the solve phase followed a supernodal scheme: updates were made directly to the right-hand side vectors. This has the advantage of a minimal memory footprint, but the obvious parallel variants (see [32]) do not offer bit-compatibility in the forward substitution as the order of updates is not well defined. However, the backward substitution only involves reads from ancestors in the assembly tree and does admit a bit-compatible implementation.

To enable parallelism in the forward substitution, a multifrontal solution scheme was implemented. Updates are passed up the assembly tree using a stack as in the factorization. This has the additional benefit of giving the same bit-compatible answer if the solve subroutine is called after the factorization as is obtained using the combined factor-solve subroutine (this is not true if the supernodal scheme is used). The multifrontal approach requires more storage than the supernodal scheme, but has more localised access patterns. The upshot is that for small problems that fit into cache the supernodal approach is faster, but for larger problems the multifrontal approach is better. The backward substitution in both the supernodal and multifrontal approaches is the same.

Parallel execution on the assembly tree is achieved by splitting the tree into subtrees representing roughly equal parts of L in terms of the number of entries. If consecutive subtrees have fewer than the target number of operations, they are merged to former larger subtrees. Dependencies of these subtrees are then established based on the assembly tree. Once all dependencies for such a subtree have been satisfied, the associated operations are executed.

At present only tree-level parallelism is exploited. While we could add node-level parallelism, we do not expect significant gains from doing so as the solve phase is memory bound [32].

Full results based on the time to perform 10 sequential solves are available in the Appendix A.3. These show that, except for the largest problems, the serial supernodal solve generally outperforms the serial multifrontal solve. Both solves suffer a parallel slowdown on small problems. However, because of the additional parallelism available in the multifrontal forward substitution, the parallel performance of the multifrontal solve is often better than that of the supernodal solve on problems of medium size. We have therefore decided to offer both schemes as options within `HSL_MA97` with a control parameter (`solve_mf`) to select between them; if a user wishes to make repeated calls to the solve phase, we suggest trying out both versions of the solve. The supernodal solve is the default. There is a further user control (`solve_min`) that specifies a cut-off value of $nz(L)$, below which parallelism is not used.

Additional experiments show that occasionally using Level 3 rather than Level 2 BLAS can improve the solve time for a single right-hand side. Thus we offer a control (`solve_blas3`) to enable this. We finally note that an apparent bug in the version of the Intel MKL we used for our experiments results in a failure to achieve bit-compatible results if Level 2 BLAS are used in the solve, but using Level 3 BLAS avoids this problem.

3.9 Overcoming compiler shortcomings

Our experience while developing and testing `HSL_MA97` has been that a number of the current Fortran compilers suffer from several common shortcomings and we have had to work around these. In this section, we briefly describe what this has involved.

To keep the user interface simple and to help with the readability and maintenance of the software, we wish to exploit dynamic memory allocation. However, this can result in multiple levels of indirection, for example `nodes(node)%val(ip)`. We have found that, for small problems, the consequent dereferencing can lead to significant overheads in the inner loops. While waiting for widespread support of the Fortran 2003 `ASSOCIATE` construct, there are two possible ways to work around this:

- Pass the components of interest to assumed-size array arguments of a subroutine. The penalty for this is subroutine call overheads.
- Use pointers. This adds a small overhead when the pointers are associated and a large overhead when they are passed as arguments to subroutines (due to contiguity checking, which can be significantly reduced through the use of the Fortran 2003 `CONTIGUOUS` attribute where the compiler supports this feature).

We have chosen the second option, with a workaround involving a pointer to a derived type containing an allocatable array and an offset into that array when passing to subroutines.

In our experiments, we found that for problems with many small allocations the time spent performing memory allocation could dominant the total cost of the factorization. By allocating large blocks and sub-partitioning these, this overhead is largely eliminated.

Figure 4.1: Description of the machine `mitchell`.

`mitchell`	
Processor	2 × Intel Xeon E5620
Physical Cores	8
Memory	24GB
Compiler	Intel Fortran 12.0.0 ifort -g -fast -openmp
BLAS	MKL 10.3.0

4 Numerical results

All our experiments were performed on the machine `mitchell`, as detailed in Figure 4.1. All reported times are elapsed times in seconds, measured using the Fortran routine `system_clock`; each time is the average time over 5 runs.

We use the following test sets:

Test Set 1: 40 small indefinite matrices (including some KKT systems). For these problems, the reported factorize times are for 100 factorizations.

Test Set 2: 40 positive-definite matrices.

Test Set 3: 20 general indefinite matrices (non-KKT systems).

Test Set 4: 20 KKT indefinite matrices.

All the problems, details of which are given in Tables 4.1 to 4.4, are taken from the University of Florida Sparse Matrix Collection [13]. The problems in Test Set 4 are indefinite problems with a system matrix of the form

$$A = \begin{pmatrix} A_1 & A_2^T \\ A_2 & A_3 \end{pmatrix},$$

The first 5 problems in Table 4.4 have non-zero A_1 and A_3; those in the centre of the table have $A_3 = 0$; and the final 3 problems have $A_1 = A_3 = 0$.

All solvers used in our experiments were called with default parameters except as detailed below. An LDL^T factorization with pivoting was specified for Test Sets 1, 3 and 4. An LL^T (or LDL^T for `MA57`) factorization without pivoting or scaling was specified for Test Set 2.

`MA57` **v3.7.0** [15] Initially real and integer memory was allocated to be 10% greater than the minimum sizes `info(9)` and `info(10)` returned by the analyse phase (more memory is required for delayed pivots). If this was insufficient, memory was reallocated to twice the estimated minimum sizes supplied by `info(17)` and `info(18)`.

`HSL_MA86` **v1.3.0** [27] The ordering was selected using the same method as that used by `HSL_MA97` (see Section 3). Scaling using `MC64` was enabled for Test Sets 1, 3 and 4. For Test Set 2, the positive-definite solver `HSL_MA87 v2.1.0` [29] was used instead of `HSL_MA86`.

Table 4.1: **Test Set 1**: Small general indefinite test matrices. $nz(A)$ is the number of entries in the lower triangular part of A; $nz(L)$ is the number of entries in L; flops is number of floating point required to calculate factors. These statistics are without node amalgamation (`nemin` $= 1$), using the stated ordering method chosen by `HSL_MA97`.

Problem index	Identifier	Ordering	n (10^3)	$nz(A)$ (10^6)	$nz(L)$ (10^6)	Flops (10^9)	Application/description
1.	Cunningham/m3plates	AMD	11.1	0.007	0.007	0.000007	3 plates in a line
2.	Boeing/bcsstm37	AMD	25.5	0.01	0.02	0.00004	Track Ball
3.	GHS_indef/qpband	AMD	20.0	0.03	0.05	0.0002	Optimization
4.	HB/zenios	AMD	2.87	0.02	0.03	0.0005	Air traffic control
5.	HB/saylr3	AMD	1.00	0.002	0.02	0.0009	Fluid Dynamics
6.	HB/sherman1	AMD	1.00	0.002	0.02	0.0009	Fluid Dynamics
7.	Oberwolfach/filter2D	AMD	1.67	0.006	0.05	0.002	Model reduction
8.	RAL/a2ensndl-00	AMD	35.0	0.09	0.26	0.002	Ipopt iteration matrix
9.	RAL/a2ensndl-49	AMD	35.0	0.09	0.26	0.002	Ipopt iteration matrix
10.	RAL/a2ensndl-62	AMD	35.0	0.09	0.26	0.002	Ipopt iteration matrix
11.	Schenk_IBMNA/c-29	AMD	5.03	0.02	0.13	0.005	Non-linear optimization
12.	Boeing/nasa1824	AMD	1.8	0.02	0.07	0.005	Structural problem
13.	GHS_indef/spmsrtls	AMD	30.0	0.13	0.35	0.005	Matrix square root
14.	IPSO/OPF_3754	AMD	15.4	0.09	0.26	0.005	Power Network
15.	GHS_indef/stokes64	AMD	12.5	0.07	0.76	0.006	Stokes problem
16.	GHS_indef/brainpc2	AMD	27.6	0.10	0.40	0.006	Biological optimization model
17.	TSOPF/TSOPF_FS_b9_c6	AMD	14.5	0.08	0.35	0.009	Optimal power flow
18.	Nemeth/nemeth05	AMD	9.51	0.20	0.30	0.01	Newton-Schultz iteration
19.	Nemeth/nemeth10	AMD	9.51	0.21	0.30	0.01	Newton-Schultz iteration
20.	FIDAP/ex14	AMD	3.25	0.04	0.17	0.01	Fluid Dynamics
21.	Nemeth/nemeth15	AMD	9.51	0.27	0.38	0.02	Newton-Schultz iteration
22.	GHS_indef/ncvxqp9	AMD	16.6	0.03	0.41	0.02	Non-convex QP
23.	Schenk_IBMNA/c-41	AMD	9.77	0.06	0.34	0.02	Non-linear optimization
24.	GHS_indef/tuma2	AMD	13.0	0.03	0.42	0.03	Mine model
25.	Marini/eurqsa	AMD	72.5	0.03	0.35	0.03	Time series
26.	Oberwolfach/rail_20209	AMD	20.2	0.08	0.71	0.03	Model reduction
27.	Newman/hep-th	AMD	8.36	0.02	0.24	0.04	Collaboration graph
28.	Nemeth/nemeth20	AMD	9.51	0.49	0.60	0.04	Newton-Schultz iteration
29.	PARSEC/Si2	METIS	0.77	0.009	0.17	0.04	Real-space pseudopotential
30.	Oberwolfach/t2dah_a	AMD	11.4	0.09	0.72	0.06	Model reduction
31.	IPSO/TSC_OPF_300	AMD	9.77	0.42	0.83	0.08	Power Network
32.	Nemeth/nemeth25	AMD	9.51	0.76	0.87	0.08	Newton-Schultz iteration
33.	Schenk_IBMNA/c-50	AMD	22.4	0.11	0.83	0.10	Non-linear optimization
34.	Newman/cond-mat	AMD	16.7	0.05	0.86	0.33	Collaboration graph
35.	Boeing/crystk01	AMD	4.88	0.16	1.11	0.33	Crystal free vibration
36.	TSOPF/TSOPF_FS_b39_c7	AMD	28.2	0.37	2.71	0.40	Optimal power flow
37.	Boeing/bcsstk37	AMD	25.5	0.58	3.21	0.61	Airplane engine component
38.	GHS_indef/exdata_1	AMD	6.00	1.14	1.16	1.16	Optimization
39.	TSOPF/TSOPF_FS_b162_c1	AMD	10.8	0.31	2.99	1.79	Optimal power flow
40.	Boeing/crystk02	METIS	14.0	0.49	4.59	1.97	Crystal free vibration

`HSL_MA97` **v1.0.0** Scaling using `MC64` was enabled for Test Sets 1, 3 and 4.

PARDISO (Intel MKL 10.3) [46] Except for Test Set 2, matching and scaling were enabled (`iparm(11)=1`, `iparm(13)=1`). We observe that while this version of PARDISO is distributed with the up-to-date version of the MKL, it lacks some of the features and enhancements available in more recent versions distributed on a pay-for basis by the University of Basel. PARDISO does not seem to supply bit-compatible results (though this is an extra feature offered by the University of Basel version). We note that by default the solve phase of PARDISO includes the use of iterative refinement.

WSMP v11.5.20 [26] An option was set so that the solver did not abort on detecting a singular matrix (`iparm(11)=1`). WSMP seems to yield bit-compatible results on all runs with the same number of threads, but not when run on different numbers of threads.

Note that we do not include a comparison with the MUMPS package as it is designed more for distributed memory than for shared memory architectures.

In each case, the right-hand size was computed so that the exact solution was $\hat{x} = e$, the vector of all ones. This allows the determination of forward errors $\|x - e\|_\infty$ and scaled backwards errors $\|Ax -$

Table 4.2: **Test Set 2**: Positive-definite test matrices. * indicates only the sparsity pattern is provided. $nz(A)$ is the number of entries in the lower triangular part of A; $nz(L)$ is the number of entries in L; flops is number of floating point required to calculate factors. These statistics are without node amalgamation (`nemin` $= 1$), using the stated ordering method chosen by `HSL_MA97`.

Problem index	Identifier	Ordering	n (10^3)	$nz(A)$ (10^6)	$nz(L)$ (10^6)	Flops (10^9)	Application/description
41.	Mulvey/finan512	METIS	74.7	0.3	1.9	0.2	Portfolio optimization
42.	MaxPlanck/shallow_water1	METIS	81.9	0.2	2.0	0.3	Weather shallow water equations
43.	UTEP/Dubcova3	METIS	147	1.9	7.5	1.3	PDE problem
44.	Nasa/nasasrb	AMD	54.9	1.4	11.9	4.6	Shuttle rocket booster
45.	CEMW/tmt_sym	METIS	727	2.9	30.0	9.4	Electromagnetics
46.	Schmid/thermal2	METIS	1228	4.9	51.6	14.6	Unstructured thermal FEM
47.	Rothberg/gearbox*	METIS	154	4.6	37.1	20.6	Aircraft flap actuator
48.	INPRO/msdoor	METIS	416	10.3	52.9	17.6	Structural problem: medium door
49.	DNVS/m_t1	AMD	97.6	4.9	31.5	21.3	Tubular joint
50.	McRae/ecology2	AMD	1000	3.0	46.0	21.9	Electrical network theory
51.	Boeing/pwtk	METIS	218	5.9	48.6	22.4	Pressurised wind tunnel
52.	Chen/pkustk13*	METIS	94.9	3.4	30.4	25.9	Machine element
53.	BenElechi/BenElechi1	METIS	246	6.7	53.8	26.8	Unknown
54.	Rothberg/cfd2	METIS	123	1.6	38.3	32.7	CFD pressure matrix
55.	DNVS/thread	METIS	29.7	2.2	24.1	34.9	Threaded connector
56.	DNVS/shipsec8	METIS	115	3.4	35.9	38.1	Ship section
57.	DNVS/shipsec1	METIS	141	4.0	39.4	38.1	Ship section
58.	GHS_psdef/crankseg_2	METIS	63.8	7.1	43.8	46.7	Linear static analysis
59.	DNVS/fcondp2*	METIS	202	5.7	52.0	48.2	Oil production platform
60.	Schenk_AFE/af_shell3	METIS	505	9.0	93.6	52.2	Sheet metal forming
61.	DNVS/troll*	METIS	214	6.1	64.2	55.9	Structural analysis
62.	GHS_psdef/bmwcra_1	METIS	149	5.4	69.8	60.8	Automotive crankshaft
63.	DNVS/halfb*	METIS	225	6.3	65.9	70.4	Half-breadth barge
64.	GHS_psdef/crankseg_1	AMD	52.8	5.3	45.6	71.4	Linear static analysis
65.	Um/2cubes_sphere	METIS	102	0.9	45.0	74.9	Electromagnetics
66.	GHS_psdef/ldoor	METIS	952	23.7	145	78.3	Structural problem: large door
67.	DNVS/ship_003	METIS	122	4.1	60.2	81.0	Ship structure
68.	DNVS/fullb*	METIS	199	6.0	74.5	100	Full-breadth barge
69.	UM/offshore	METIS	256	2.3	84.5	106	Electromagnetics
70.	GHS_psdef/inline_1	METIS	504	18.7	173	144	Inline skater
71.	Chen/pkustk14*	METIS	152	7.5	107	146	Tall building
72.	GHS_psdef/apache2	METIS	715	2.8	135	174	3D structural problem
73.	Koutsovasilis/F1	METIS	344	13.6	174	219	AUDI engine crankshaft
74.	Oberwolfach/boneS10	METIS	915	28.2	278	282	Bone micro-FEM
75.	AMD/G3_circuit	AMD	1586	4.6	193	298	Circuit simulation
76.	ND/nd12k	METIS	36.0	7.1	117	505	3D mesh problem
77.	JGD_Trefethen/ Trefethen_20000	AMD	20.0	0.3	86.8	745	Integer matrix
78.	ND/nd24k	METIS	72.0	14.4	321	2054	3D mesh problem
79.	Oberwolfach/bone010	METIS	987	36.3	1076	3876	Bone micro-FEM
80.	GHS_psdef/audikw_1	METIS	944	39.3	1242	5804	Automotive crankshaft

Table 4.3: **Test Set 3**: General indefinite test matrices. $nz(A)$ is the number of entries in the lower triangular part of A; $nz(L)$ is the number of entries in L; flops is number of floating point required to calculate factors. These statistics are without node amalgamation (`nemin` $= 1$), using the stated ordering method chosen by `HSL_MA97`.

Problem index	Identifier	Ordering	n (10^3)	$nz(A)$ (10^6)	$nz(L)$ (10^6)	Flops (10^9)	Application/description
81.	Oberwolfach/t2dal	AMD	4.25	0.02	0.12	0.006	Model reduction
82.	GHS_indef/dixmaanl	AMD	60.0	0.18	0.34	0.002	Optimization problem
83.	Oberwolfach/rail_79841	AMD	79.8	0.32	1.84	0.18	Semi-discretized heat transfer, steel profile cooling
84.	GHS_indef/dawson5	AMD	51.5	0.53	4.62	1.22	Part of actuator system on airplane
85.	Boeing/bcsstk39	AMD	46.8	1.07	6.69	1.62	Rocket booster
86.	GHS_indef/helm2d03	METIS	392	1.57	19.1	4.75	Helmholtz eq. on a unit square
87.	GHS_indef/copter2	METIS	55.5	0.41	95.0	5.35	CFD helicopter rotar blade
88.	Boeing/crystk03	METIS	24.7	0.89	9.50	5.60	Crystal vibration
89.	Oberwolfach/filter3D	METIS	106	1.41	18.7	7.51	3D heat transfer PDE
90.	Boeing/pct20stif	AMD	52.3	1.38	11.4	9.09	Engine block
91.	Koutsovasilis/F2	AMD	71.5	2.68	20.4	11.2	Piston rod
92.	Cunningham/qa8fk	METIS	66.1	0.86	23.3	21.2	3D acoustic FE stiffness matrix
93.	Oberwolfach/gas_sensor	METIS	66.9	0.89	23.8	21.2	Thermal model single gas sensor device
94.	McRae/ecology1	AMD	1000	3.00	47.2	22.4	Electrical network theory
95.	Oberwolfach/t3dh	METIS	79.2	2.22	47.2	68.9	Mcropyros thruster
96.	Lin/Lin	METIS	256	1.01	107	276	Eigenvalue problem
97.	GHS_indef/sparsine	METIS	227	0.80	200	1368	Structural optimization
98.	PARSEC/Ge99H100	METIS	113	4.28	649	6999	Density functional theory
99.	PARSEC/Ga10As10H30	METIS	113	3.11	668	7189	Density functional theory
100.	PARSEC/Ga19As19H42	METIS	133	4.51	799	9009	Density functional theory

Table 4.4: **Test Set 4:** KKT indefinite test matrices. The first 5 problems have nonzero (1,1) and (2,2) blocks; those in the centre part of the table have a zero (2,2) block; the final 3 problems have a (1,1) and (2,2) zero block. $nz(A)$ is the number of entries in the lower triangular part of A; $nz(L)$ is the number of entries in L; flops is number of floating point required to calculate factors. These statistics are without node amalgamation (`nemin` $= 1$), using the stated ordering method chosen by `HSL_MA97`.

Problem index	Identifier	Ordering	n (10^3)	$nz(A)$ (10^6)	$nz(L)$ (10^6)	Flops (10^9)	Application/description
101.	GHS_indef/boyd1	AMD	93.3	0.65	0.64	0.005	Convex QP problem
102.	GHS_indef/bmw3_2	METIS	227	5.76	46.6	29.1	Linear static analysis of a car body
103.	GHS_indef/c-72	AMD	84.0	0.40	3.11	1.75	Nonlinear optimization
104.	GHS_indef/ncvxqp7	METIS	87.5	0.31	19.1	26.3	Nonconvex QP problem
105.	Andrianov/mip1	AMD	66.5	5.21	44.1	145	Optimization
106.	GHS_indef/blockqp1	AMD	60.0	0.34	0.38	0.0004	QP with block structure
107.	GHS_indef/boyd2	AMD	466	0.89	1.31	0.0004	Convex QP problem
108.	GHS_indef/a5esindl	AMD	60.0	0.15	0.22	0.0008	Linear complementarity problem
109.	GHS_indef/a2nnsnsl	AMD	80.0	0.20	0.36	0.002	Linear complementarity problem
110.	GHS_indef/a0nsdsil	AMD	80.0	0.20	0.39	0.002	Linear complementarity problem
111.	TSOPF/TSOPF_FS_b39_c30	METIS	120	1.58	4.78	0.31	Transient optimal power flow
112.	GHS_indef/cont-201	AMD	80.6	0.24	3.59	0.75	Convex QP problem
113.	GHS_indef/darcy003	AMD	390	1.17	5.71	0.76	Mixed FE discretization of Darcy's equation
114.	GHS_indef/cont-300	AMD	181	0.54	9.82	3.28	Convex QP problem
115.	GHS_indef/turon_m	METIS	190	0.91	11.9	4.14	Model of underground mine
116.	GHS_indef/d_pretok	METIS	183	0.89	12.8	4.97	Model of underground mine
117.	TSOPF/TSOPF_FS_b300_c3	METIS	84.4	6.58	22.8	9.67	Optimal power flow
118.	GHS_indef/dtoc	METIS	25.0	0.03	0.10	0.0005	Discrete-time optimal control
119.	GHS_indef/aug2d	AMD	29.0	0.04	0.25	0.01	2D PDE
120.	GHS_indef/aug3d	AMD	24.3	0.03	0.55	0.14	3D PDE

$b\|_\infty/\left(\|A\|_\infty\|x\|_\infty + \|b\|_\infty\right)$. Note that exceptionally large forward errors indicate the scaled backwards error is a poor measure of solution quality as $\|x\|_\infty$ dominates the calculation. Problems attaining either a forward error greater than 1×10^{14} after a single solution, or a scaled backwards error of greater than 1×10^{-14} after 5 steps of iterative refinement, were deemed to have achieved an incorrect solution; they are flagged in the tables of results that are included in the appendices and are omitted from the graphs of results. In the case of PARDISO, the 5 steps of iterative refinement were performed externally, but without disabling the internal iterative refinement of PARDISO.

4.1 Node amalgamation tuning

The analyse phase of `HSL_MA97` uses the package `HSL_MC78`. This offers the option of node amalgamation. A child node is merged with its parent if either both parent and child have fewer than `nemin` variables that are eliminated or merging parent and child generates no additional nonzeros in L. The choice of the control parameter `nemin` determines the level of node amalgamation, with a value in the range 8 to 32 typically recommended as providing a good balance between sparsity and efficiency in the factorize and solve phases (see, for example, [29, 41]). `HSL_MA97` was run with `nemin` values of 4, 8 and 16 across the full test set in serial and parallel. On the basis of the factorize results reported in Appendix A, `nemin`= 8 or `nemin`= 16 should be chosen as the default. Based on the solve times, we have selected `nemin`= 8 as the default within `HSL_MA97`. Figures 4.2 and 4.3 illustrate the effects of varying `nemin` on the factorize times and on the number of entries in the factors and on solve phase times.

4.2 Parallel speedup

Experiments showed a slowdown on small problems because of parallel overheads and additional communication. We have thus included a control parameter (`factor_min`) that determines the minimum number of flops returned by the analyse phase that are required before parallel computation is attempted. A default setting of 2×10^7 operations was selected, which corresponds to running problems 1 to 24 in Test Set 1 in serial. Clearly, the best value will depend on the machine used and so we advise users with small or medium problems to experiment with running in serial and parallel for their applications. Figure 4.4 shows the speedup of the factorize phase of `HSL_MA97`.

4.3 Sensitivity to choice of u

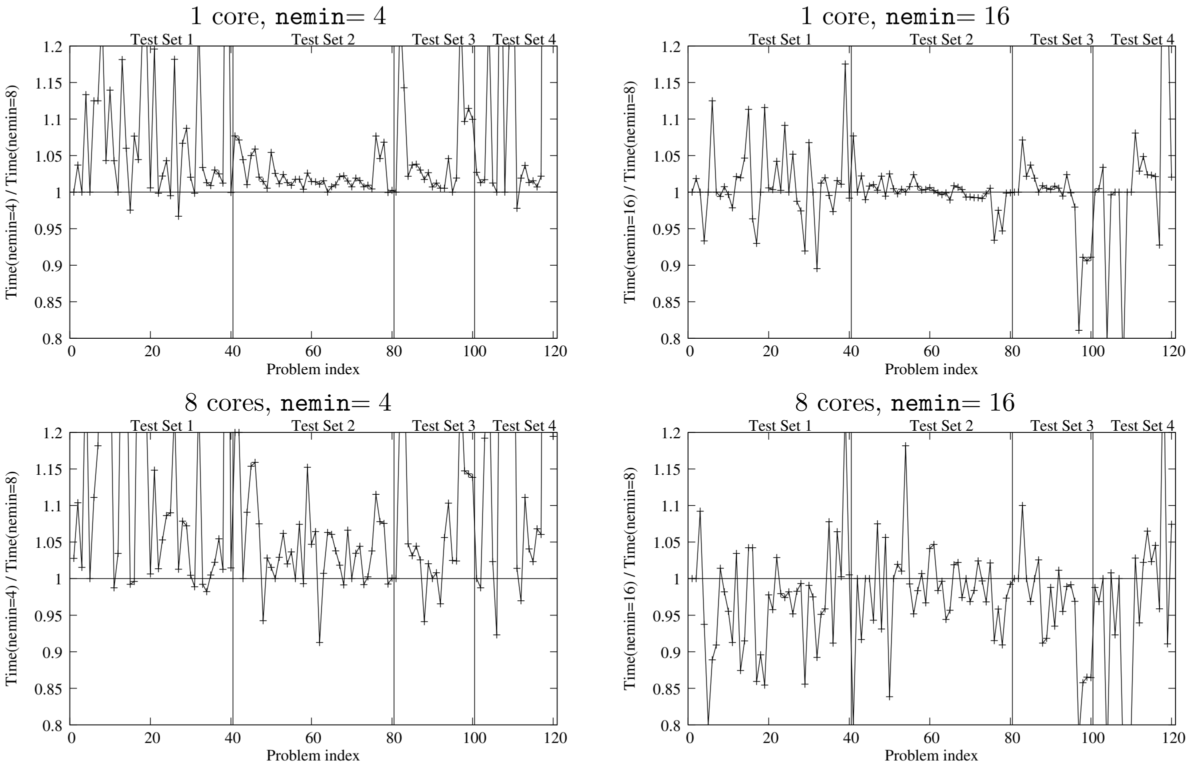
The pivoting strategy used with `HSL_MA97` is discussed by Reid and Scott [43]. It uses a threshold parameter u. Experiments were conducted on indefinite problems to compare the performance of `HSL_MA97` for different values of u. Reducing u relaxes the pivoting rules and can significantly reduce the amount of work done during the selection of pivots and decreases both the number of delayed pivots and the number of entries in L. The values $u = 0.01$ and $u = 0.001$ were compared (see Appendix A.4). It was found that for most problems for which $u = 0.01$ resulted in a significant number of delayed pivots, using $u = 0.001$ was faster. However, the forward errors were sometimes as much as two orders of magnitude worse, although accuracy was recovered using iterative refinement. Since we are aiming at robustness and are aware from experience that many users will leave the control paramaters unchanged, $u = 0.01$ is the default within `HSL_MA97` (it is also the default for `MA57`, `HSL_MA77` and `HSL_MA86`).

4.4 Evaluation of `HSL_MA97` as a replacement for `MA57`

One of our aims for `HSL_MA97` was for it to supersede `MA57`. In this section, we look at how successful we have been in achieving this objective.

Figure 4.5 compares the performance of the analyse phase of `MA57` with that of `HSL_MA97`. Recall that both select the ordering using the same heuristic [22], although the implementations of AMD used by `MA57` and `HSL_MA97` are not the same. The former uses the package `MC47` [1] while the latter uses the

Figure 4.2: Factorize times for `nemin`= 4 and 16 compared to those for the default value `nemin`= 8 on 1 and 8 cores. Points below the line indicate better performance than the default.



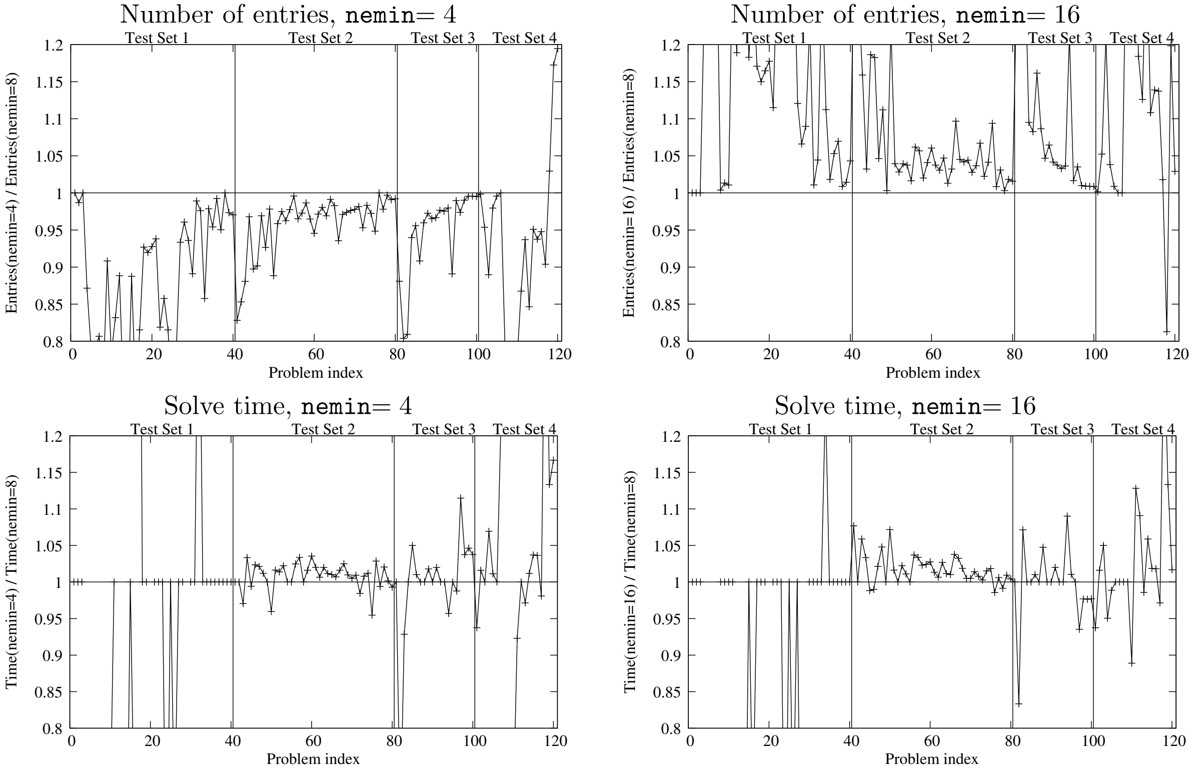
1 core, `nemin`= 4

1 core, `nemin`= 16

8 cores, `nemin`= 4

8 cores, `nemin`= 16

Figure 4.3: The number of entries in the factors (top) and solve times (bottom) for `nemin`= $4, 16$ compared to the default `nemin`= 8. Points below the line indicate better performance than the default.



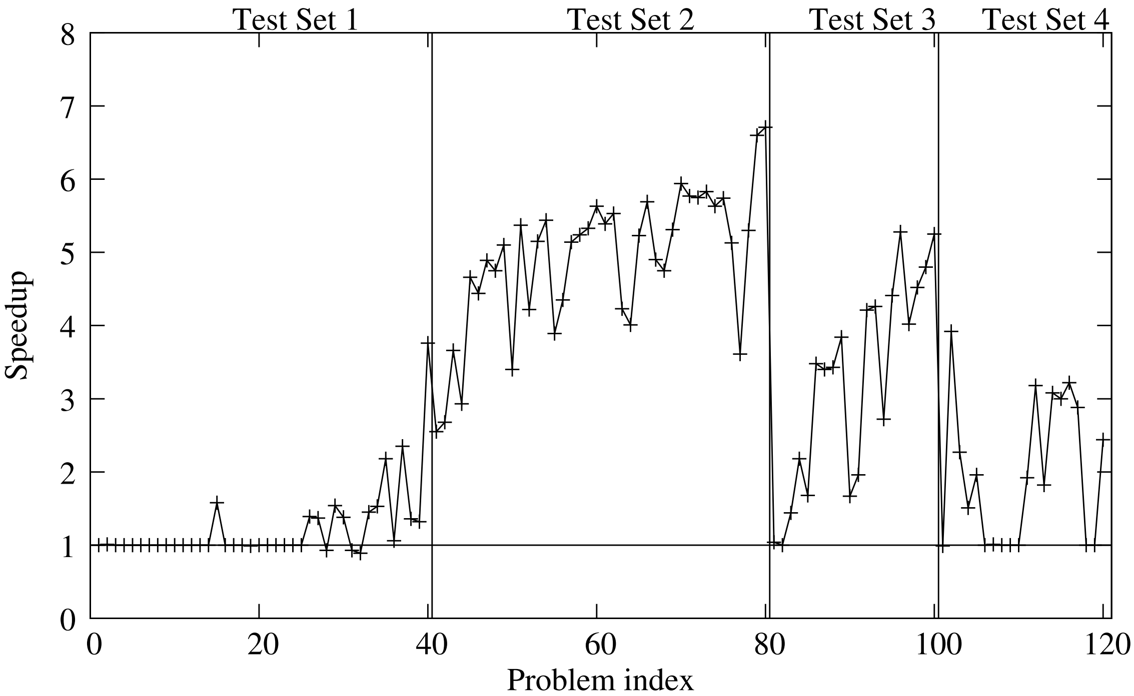
Number of entries, `nemin`= 4

Number of entries, `nemin`= 16

Solve time, `nemin`= 4

Solve time, `nemin`= 16

Figure 4.4: Speedup of `HSL_MA97` when run on 8 cores.



ordering package `HSL_MC68` which, in turn, uses the recent AMD implementation of Dollar and Scott [14]. The general trend is that when an AMD ordering is used (the smaller problems and those that are highly sparse), the `MA57` analyse time outperforms that of `HSL_MA97`. This is probably due to tighter integration of the AMD ordering within `MA57`, allowing reuse of data structures when performing the ordering, while `HSL_MA97` incurs additional overheads of converting between holding only the lower triangular part of A and holding both the upper and lower triangular parts when calling `HSL_MC68`. When calling METIS, similar overheads are present for both solvers and the superior performance of the `HSL_MC78` symbolic factorization used by `HSL_MA97` is then demonstrated.

In Figure 4.6, the serial factorize times of `MA57` and `HSL_MA97` are compared while, in Figure 4.7, the parallel factorize times are compared. `MA57` is not written as a parallel code but parallel performance can be achieved by using multithreaded BLAS. Using multithreaded BLAS, `MA57` will achieve bit-compatibility if the BLAS are bit-compatible (this is the case for the version of the MKL we use). For Test Set 1, the time is generally dominated by the time taken to scale the matrix using `MC64` and, for these problems, there is little to choose between the two codes. For Test Set 2 (positive-definite problems), `HSL_MA97` outperforms `MA57`. This is because the former uses a dedicated Cholesky factorization kernel whereas the latter was designed primarily for indefinite systems. The performance on the larger indefinite problems belonging to Test Sets 3 and 4 demonstrates that, in serial, `HSL_MA97` generally outperforms `MA57` by a small margin. For the larger problems run in parallel, the margins are greater and the parallel scheme used by `HSL_MA97` runs at least twice as fast as `MA57` using the multithreaded BLAS.

Figure 4.8 compares the solve performance. Results for Test Set 1 are ommited as the graph would only demonstrate measurement errors for such small numbers. It is clear that `MA57` is consistently faster in serial. Since the size of the factors computed by the two codes is comparable, we believe this is caused by differences in the data structures used for holding the factors. Given a trapezoidal section of the factors corresponding to a supernode, `MA57` stores the square part in packed storage with the rectangular block below it in contiguous full storage. It may then use the BLAS routines `_tpsv` and `_gemm` to perform the solve. `HSL_MA97` stores the entire trapezoidal section in full storage and uses `_trsv` and `_gemm`. This results in the data for the two calls being interleaved, hindering the effectiveness of hardware prefetching for these

Figure 4.5: Comparison of analyse performance by `MA57` and `HSL_MA97`. Points below the line indicate better performance by `MA57`.

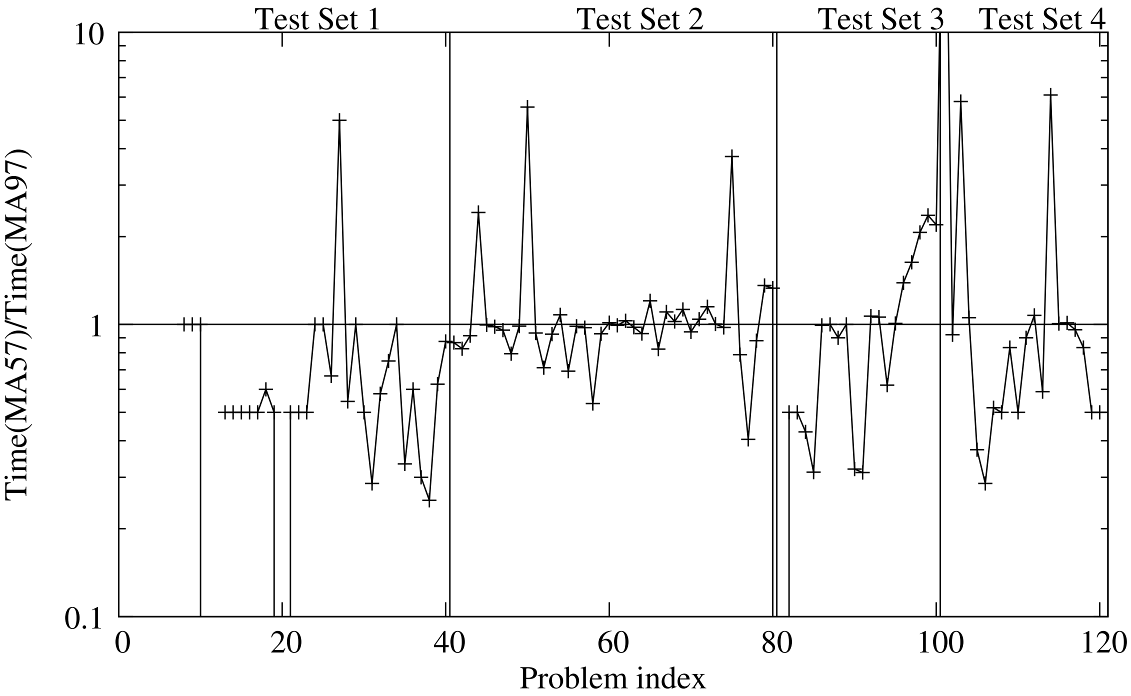


Figure 4.6: Comparison of serial factorize performance of `MA57` and `HSL_MA97`. Points below the line indicate better performance by `MA57`.

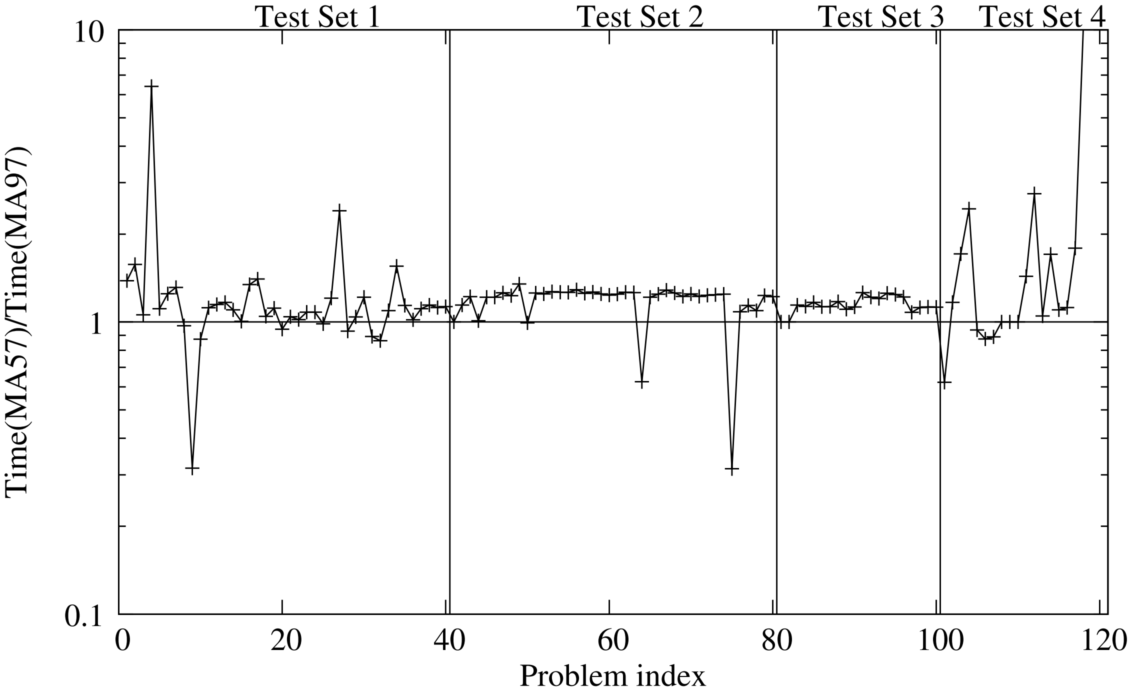


Figure 4.7: Comparison of parallel factorize performance of `MA57` and `HSL_MA97`. Points below the line indicate better performance by `MA57`.

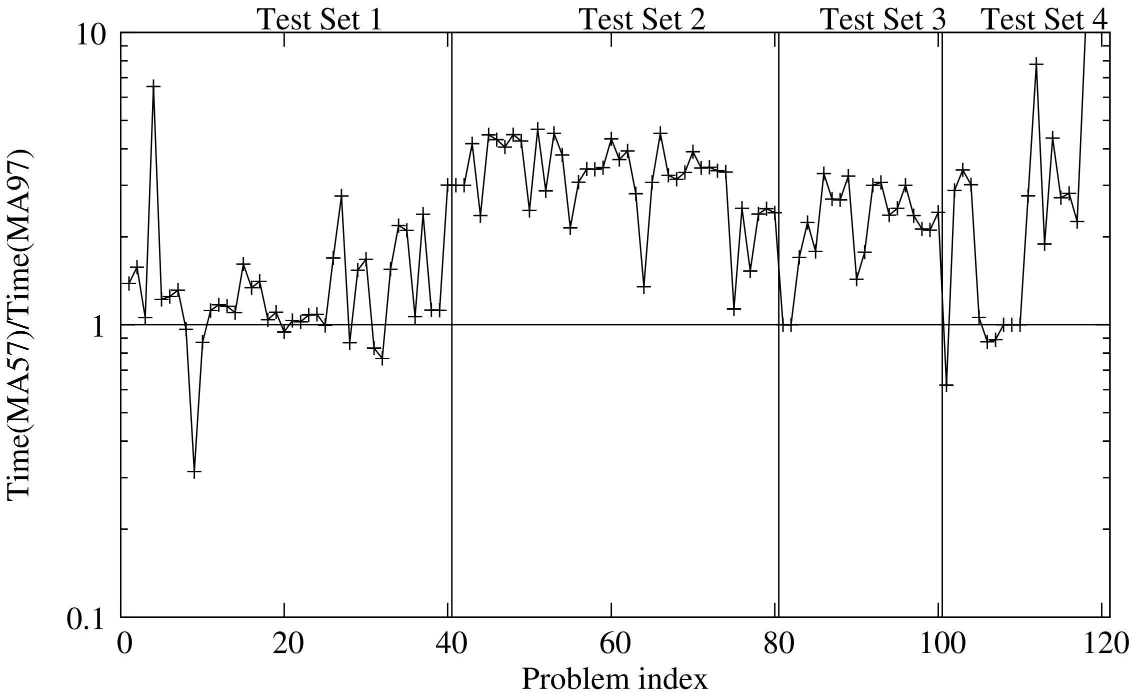
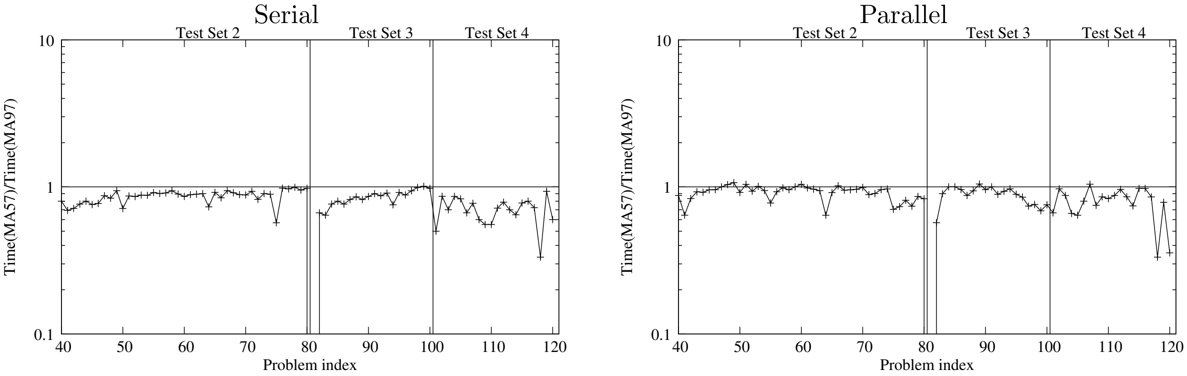


Figure 4.8: Comparison of the solve performance of `MA57` and `HSL_MA97` in serial and parallel. Points below the line indicate better performance by `MA57`.



operations. When both codes are run in parallel, the difference between the solves times is smaller.

Finally, Figures 4.9 and 4.10 compare the times for `MA57` and `HSL_MA97` to complete analyse-factorize-solve for a single right-hand side. Test Set 1 is omitted from these graphs as the timing for such small problems is unreliable.

5 Comparison of HSL_MA86, HSL_MA87 and HSL_MA97

Figures 5.1 and 5.2 compare the factorize and analyse-factor-solve parallel performance of `HSL_MA86`/`HSL_MA87` (indefinite/positive-definite) and `HSL_MA97`; tables of serial and parallel times are given in Appendices A.5 to A.7. Note that `HSL_MA86/7` are supernodal rather than multifrontal solvers that were designed from the ground up to fully exploit all available parallelism. Thus they are generally faster but the penalty for this is that they do not offer bit-compatible results.

We note that the performance gap between `HSL_MA87` and `HSL_MA97` on Test Set 2 is larger than that between `HSL_MA86` and `HSL_MA97` on the indefinite problems. This is probably due to the exploitation of a row-wise data layout by `HSL_MA87`; while a similar layout is employed by `HSL_MA86`, it is handicapped by the data access patterns required for pivoting.

6 Comparison with other sparse solvers

We have also compared the performance of `HSL_MA97` with the shared memory codes PARDISO and WSMP. It is worth noting that while WSMP is multifrontal, PARDISO is not. Additionally, both these codes differ in their pivoting techniques and the threshold pivoting parameter u has different meanings for each of these solvers, so comparisons of this parameter between `HSL_MA97` and PARDISO and WSMP cannot be made. Another important difference is that PARDISO and WSMP are designed to perform an `MC64`-equivalent matching and scaling during the analyse phase, whereas the HSL codes do not use numerical values in the analyse phase and scale on each call to the factorization. As a result, we only present overall analyse-factorize-solve results in serial and parallel (excluding times for external iterative refinement). Results are presented in Figures 6.1 and 6.2 for Test Sets 2 to 4. Note that the times for a single call for problems in Test Set 1 are too small to be reliably measured and repeating the factorize phase 100 times as in our other tests on Test Set 1 would unfairly disadvantage `HSL_MA97` because of the scaling overhead. We observe that, for some problems, results are omitted for one or more of the solvers (as explained earlier, we omit a result if either the forward error is greater than 1×10^{14} after a single solution, or the scaled backwards error is greater than 1×10^{-14} after 5 steps of iterative refinement).

The results suggest that `HSL_MA97` is competitive with both PARDISO and WSMP in parallel, and on indefinite problems in serial it is possibly performing the best overall (although on the positive-definite problems, WSMP is consistently the fastest solver, see Appendix A.6.3). However, the performance of `HSL_MA97` on the largest problems in Test Set 3 is disappointing and, at least as far as PARDISO is concerned, is probably due to the overheads of attempting to achieve a more accurate result through the use of a more stable, but more expensive, pivoting technique.

The forward and backward errors, without external iterative refinement, are presented in Figures 6.3 and 6.4. For each solver, the number of problems that failed to achieve an accurate solution after iterative refinement is reported in the following table. It is clear that the weaker pivoting strategy used by PARDISO can have potentially serious consequences.

Solver	Number failed
`HSL_MA97`	0
PARDISO	20
WSMP	2

Figure 4.9: Comparison of analyse-factorize-solve performance of `MA57` and `HSL_MA97` in serial. Points below the line indicate better performance by `MA57`.

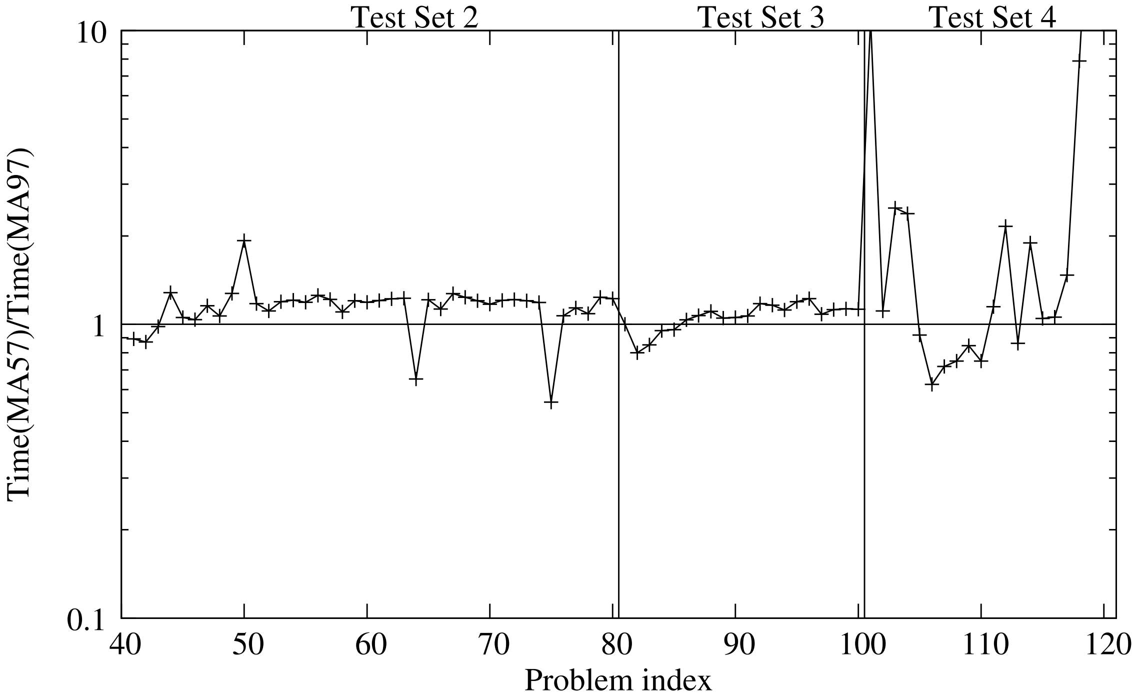


Figure 4.10: Comparison of analyse-factorize-solve performance of `MA57` and `HSL_MA97` in parallel. Points below the line indicate better performance by `MA57`.

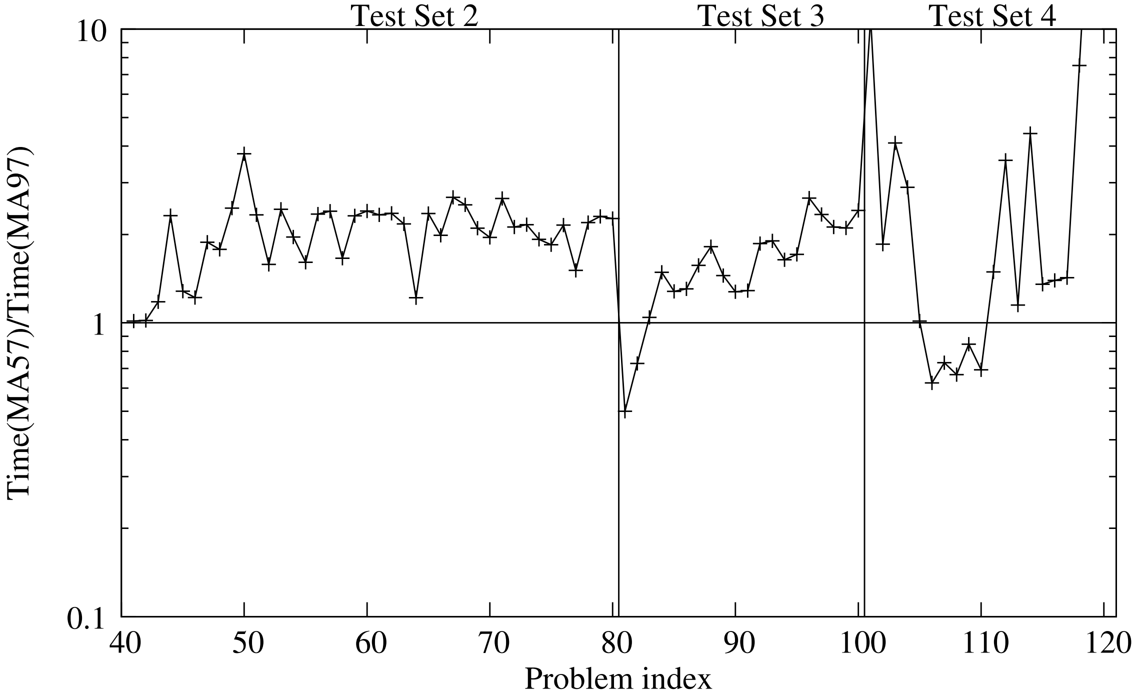


Figure 5.1: Comparison of factorize performance of `HSL_MA86/7` and `HSL_MA97` in parallel. Points below the line indicate better performance by `HSL_MA86/7`.

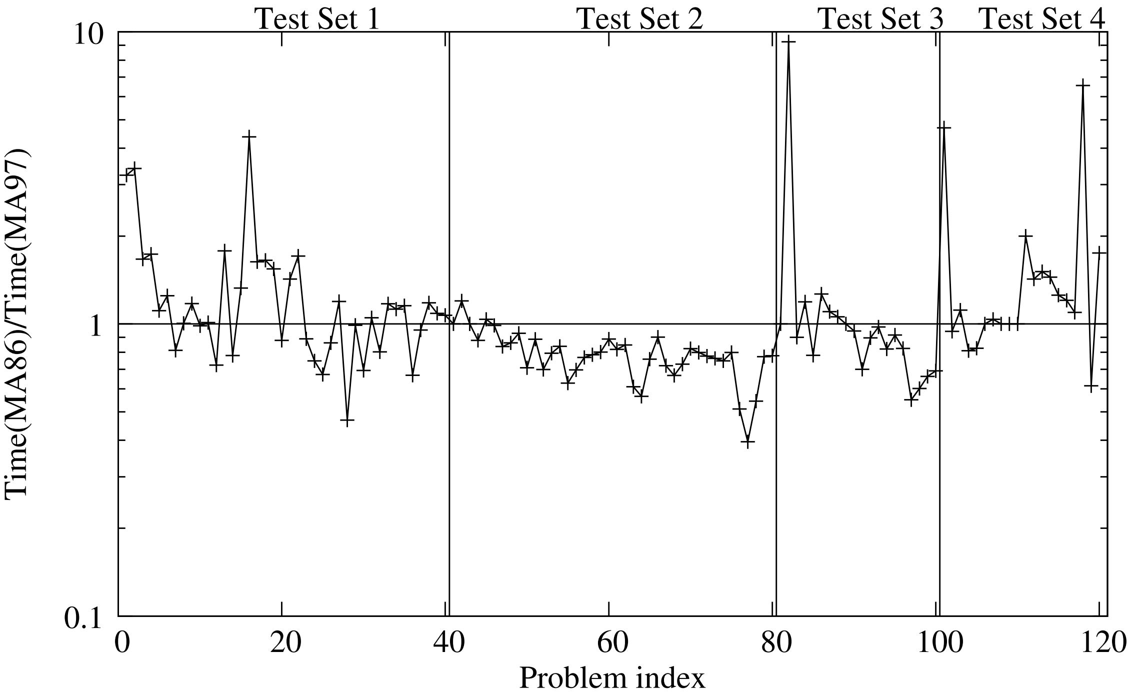


Figure 5.2: Comparison of analyse-factorize-solve performance of `HSL_MA86/7` and `HSL_MA97` in parallel. Points below the line indicate better performance by `HSL_MA86/7`.

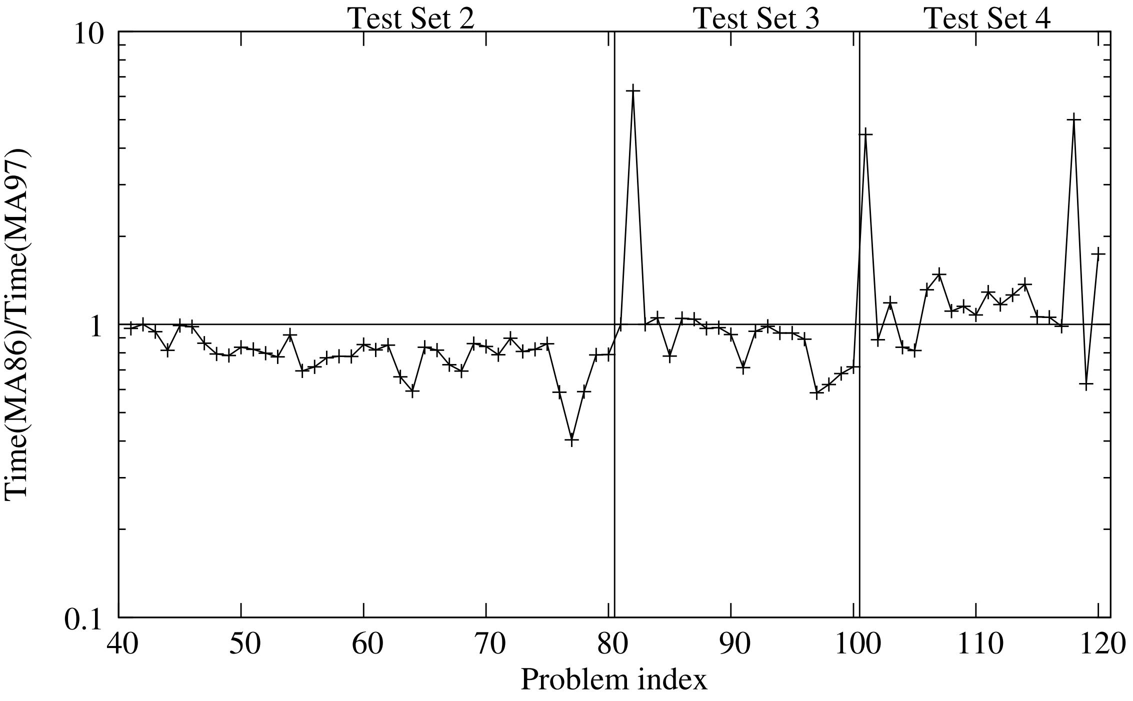


Figure 6.1: Comparison of analyse-factorize-solve performance of WSMP, PARDISO and `HSL_MA97` in serial. Points below the line indicate better performance than `HSL_MA97`.

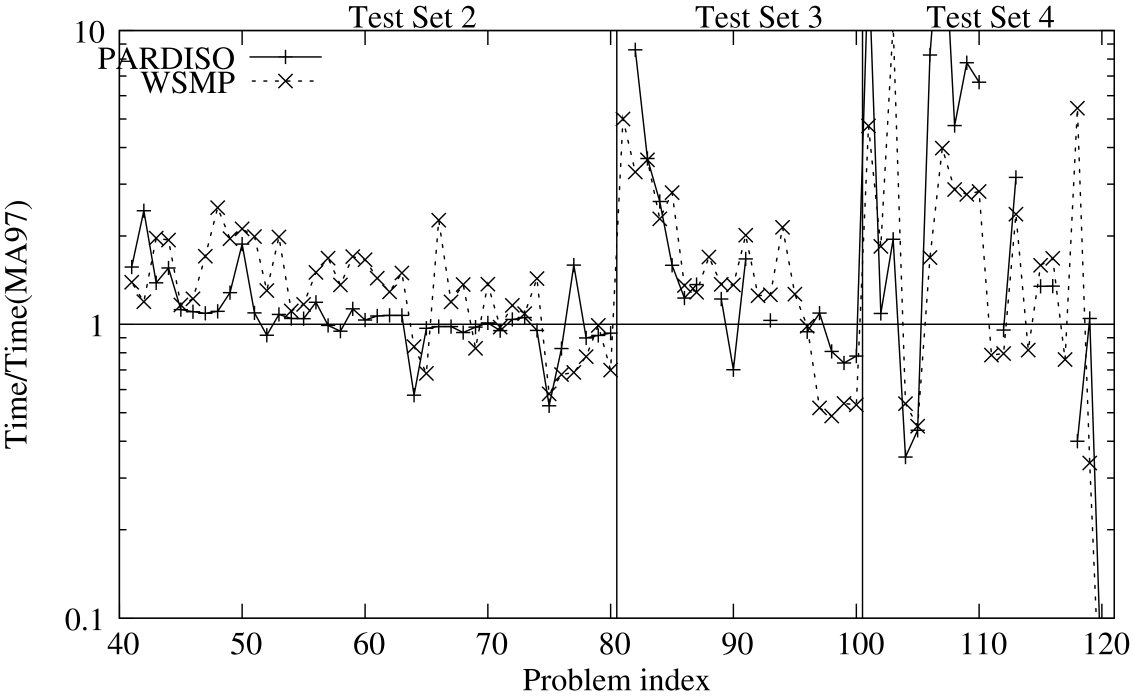


Figure 6.2: Comparison of analyse-factorize-solve performance of WSMP, PARDISO and `HSL_MA97` in parallel. Points below the line indicate better performance than `HSL_MA97`.

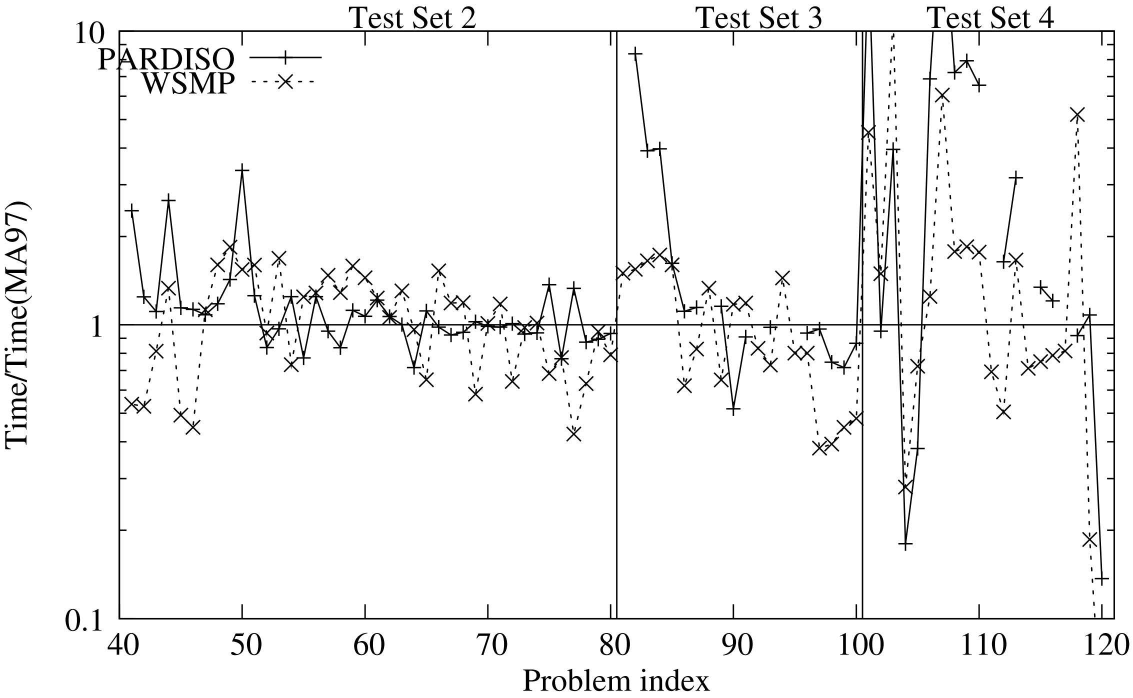


Figure 6.3: Comparison of forward errors by WSMP, PARDISO and `HSL_MA97` in serial.

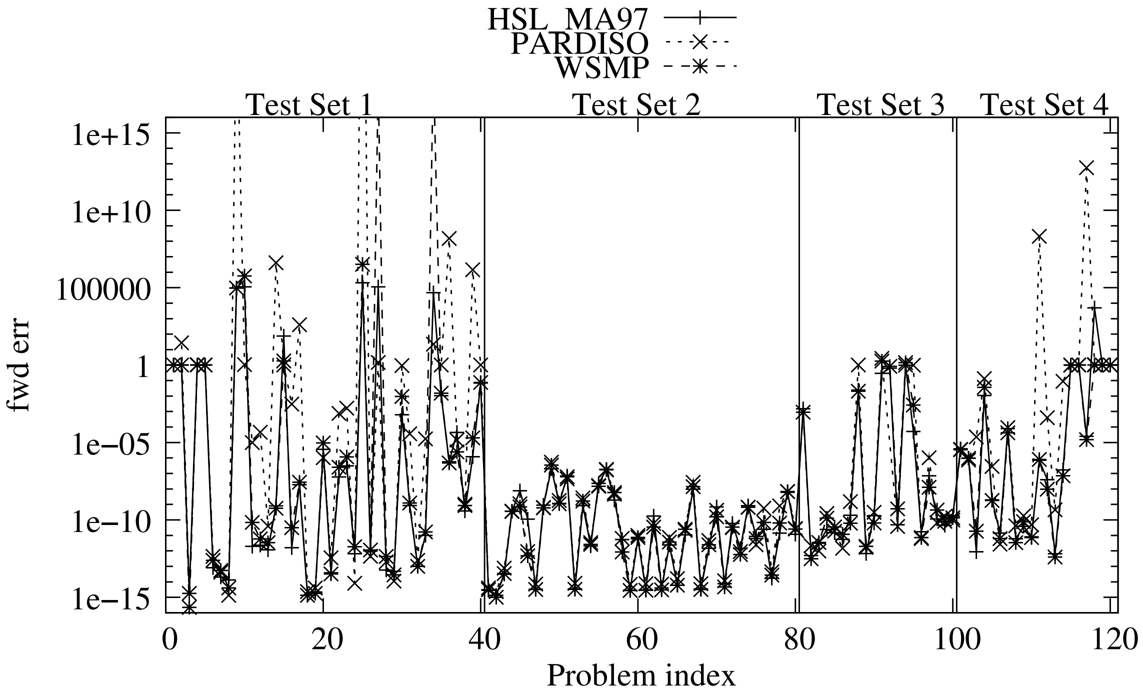
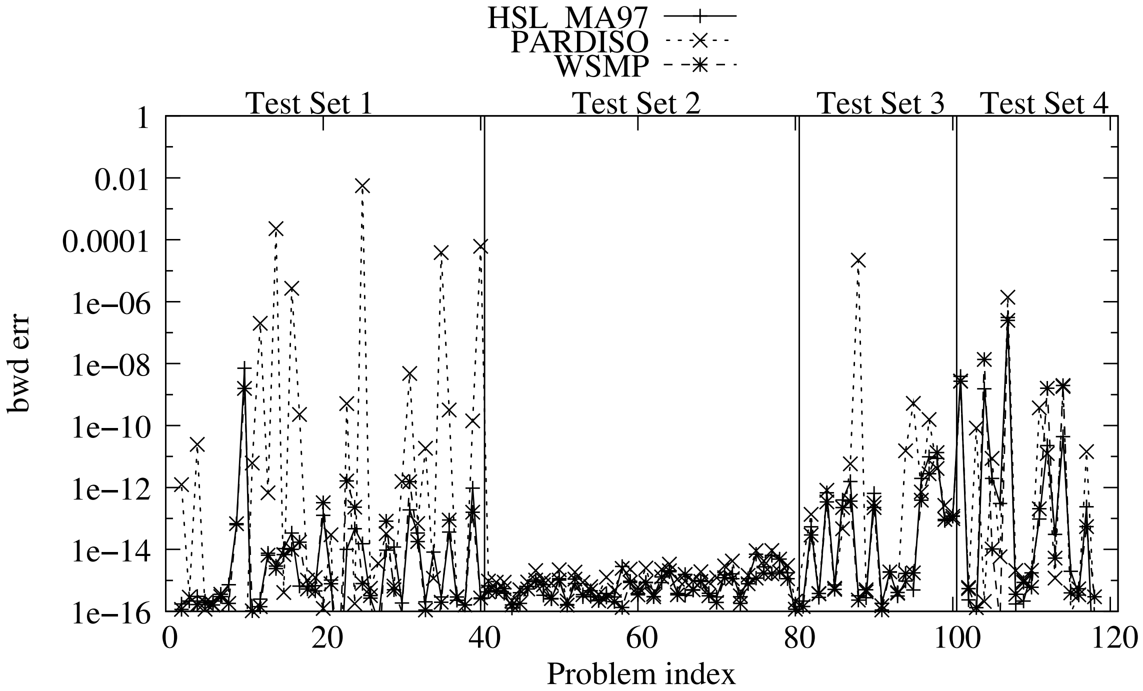


Figure 6.4: Comparison of scaled backward errors by WSMP, PARDISO and `HSL_MA97` in serial.



7 Summary and future directions

In this report, we have discussed the design and development of a new multifrontal solver. This solver takes advantage of our recent work on an efficient analyse phase [31] and dense linear algebra kernels [6, 42]. Numerical experiments have shown that in parallel `HSL_MA97` often outperforms the multifrontal code `MA57` and its performance compares favourably with other sparse solvers. Furthermore, `HSL_MA97` offers bit-compatible solutions.

The new multi- and many-core era has led to a need to redesign algorithms and software to be efficient on these architectures. In addition to parallelism, memory access (communication) is starting to replace computation as the driving force behind algorithm design. We are particularly interested in the important problem of developing direct solvers that address these challenges. The solution of such systems requires the careful selection of pivots on the grounds of numerical stability. This presents a two-fold challenge on multicore machines as both efficient pivot selection and the consequences of maintaining stability in non-definite cases differ significantly from more traditional architectures.

Recent experience of developing Cholesky solvers on multicore machines [10, 28, 29] has led us to identify the need for fine-grained parallelism and careful management of cache locality through well designed data structures as essential to obtaining good performance. The limited memory bandwidth and cache shared between multiple cores means that close attention needs to be paid to this if efficiency and good speedups are to be achieved.

The threshold pivot selection techniques within the dense linear algebra kernel that lies at the heart of `HSL_MA97` require a full update and scan of each candidate column before accepting or rejecting it as a pivot. This limits fine-grained parallelism as a block column cannot be divided into multiple row blocks. Moreover, column scanning can result in extremely undesirable cache access patterns. To circumvent this, we propose developing new block-orientated pivoting techniques that can optionally backtrack and utilise more stringent pivoting if unacceptably large growth is encountered.

The effects of delaying pivot candidates are additional fill in the factors and an increase in the flop count. This affects memory requirements (possibly limiting the size of the system that can be solved) and increases the factorize and solve times. As we have recently demonstrated [32], the solve phase presents a potential bottleneck on multicore architectures as it is memory bound. Thus we want to avoid delayed pivots and keep the factors as sparse as possible. A popular approach is the use of static pivoting techniques. However, not only do these mean that the inertia of the system (a property required by many optimization techniques) may not be computed accurately, a number of refinement steps are normally needed to recovery full accuracy. This involves additional solves with the computed factors which, as we have just noted, we want to avoid on multicore machines.

Thus our interest is in developing new ways of limiting the number of delayed pivots without compromising stability or inertia. These will include novel initial ordering strategies that take better account of zero diagonal entries in the system matrix. Furthermore, inspired by the approach used in the multilevel ILU algorithm, we propose investigating partial rescaling of the matrix during the factorization. This may slow the factorization but may result in significantly sparser factors (without loss of stability) and hence a faster solve phase.

We will implement our ideas within `HSL_MA97`. We will write an Ipopt interface for `HSL_MA97`. Furthermore, we plan to develop a novel task-based interior point code for multicore machines and `HSL_MA97` will form an important component within this project. Our future plans also include developing a GPU-accelerated version of `HSL_MA97`.

Acknowledgements

We are very grateful to our colleague John Reid for numerous discussions relating to sparse matrix computation and multifrontal methods in particular and for his work on developing the dense kernel `HSL_MA64`. Many thanks also to Iain Duff for commenting on a draft on this report and answering our queries about `MA57`. As always, we are grateful to Tim Davis and Yifan Hu for maintaining and making available the invaluable University of Florida Sparse Matrix Collection. Our thanks also to Anshul Gupta for access to his package WSMP.

References

[1] P. Amestoy, H. Dollar, J. K. Reid, and J. A. Scott, *An approximate minimum degree algorithm for matrices with dense rows*, Technical Report RAL-TR-2007-020, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2007.

[2] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[3] ——, *Algorithm 837: AMD, an approximate minimum degree ordering algorithm*, ACM Trans. Math. Softw., 30 (2004), pp. 381–388.

[4] P. R. Amestoy, I. S. Duff, and J. Y. L'Excellent, *Multifrontal parallel distributed symmetric and unsymmetric solvers*, Comput. Methods in Appl. Mech. Eng, 184 (2000), pp. 501–520.

[5] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, *A fully asynchronous multifrontal solver using distributed dynamic scheduling*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 15–41.

[6] B. Andersen, J. Gunnels, F. Gustavson, J. Reid, and J. Wasniewski, *A fully portable high performance minimal storage hybrid format cholesky algorithm*, ACM Trans. Math. Softw., 31 (2005), pp. 201–207.

[7] M. Arioli and I. S. Duff, *FGMRES to obtain backward stability in mixed precision*, Technical Report RAL-TR-2008-006, Rutherford Appleton Laboratory, 2008.

[8] C. Ashcraft and R. Grimes, *SPOOLES: An object-oriented sparse matrix library*, in Proceedings of the Ninth SIAM Conference on Parallel Processing, Philadelphia, 1999, SIAM. Software available at `http://www.netlib.org/linalg/spooles`.

[9] C. Ashcraft, R. G. Grimes, and J. G. Lewis, *Accurate symmetric indefinite linear equation solvers*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 513–561.

[10] A. Buttari, J. Dongarra, J. Kurzak, J. Langou, P. Luszczek, and S. Tomov, *The impact of multicore on math software*, in Proceedings of Workshop on State-of-the-art in Scientific and Parallel Computing (PARA06), 2006.

[11] S. H. Cheng and N. J. Higham, *A modified Cholesky algorithm based on a symmetric indefinite factorization*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 1097–1110.

[12] T. Davis and I. Duff, *An unsymmetric-pattern multifrontal method for sparse LU factorization*, SIAM J. Matrix Anal. Appl., 16 (1997), pp. 140–158.

[13] T. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Softw., 38 (2011).

[14] H. Dollar and J. Scott, *A note on fast approximate minimum degree orderings for matrices with some dense rows*, Numerical Linear Algebra and Applications, 17 (2010), pp. 43–55.

[15] I. S. Duff, *MA57– a new code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Softw., 30 (2004), pp. 118–154.

[16] I. S. Duff, A. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, 1986.

[17] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.

[18] I. S. Duff and S. Pralet, *Strategies for scaling and pivoting for sparse symmetric indefinite problems*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 313–340.

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

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

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

[22] I. S. Duff and J. A. Scott, *Towards an automatic ordering for a symmetric sparse direct solver*, Technical Report RAL-TR-2006-001, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2005.

[23] S. C. Eisenstat, M. H. Schultz, and A. H. Sherman, *Applications of an element model for Gaussian elimination*, in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, 1976, pp. 85–96.

[24] N. I. M. Gould and J. A. Scott, *A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations*, ACM Trans. Math. Softw., 30 (2004), pp. 300–325.

[25] N. I. M. Gould, J. A. Scott, and Y. Hu, *A numerical evaluation of sparse direct solvers for the solution of large, sparse, symmetric linear systems of equations*, ACM Trans. Math. Softw., 33 (2007). Article 10, 32 pages.

[26] A. Gupta, M. Joshi, and V. Kumar, *WSMP: A high-performance serial and parallel sparse linear solver*, Technical Report RC 22038 (98932), IBM T.J. Watson Research Center, 2001. `http://www.cs.umn.edu/~agupta/doc/wssmp-paper.ps`.

[27] J. Hogg and J. Scott, *An indefinite sparse direct solver for large problems on multicore machines*, Technical Report RAL-TR-2010-011, Rutherford Appleton Laboratory, 2010.

[28] J. D. Hogg, *High Performance Cholesky and Symmetric Indefinite Factorizations with Applications*, PhD thesis, University of Edinburgh, 2010.

[29] J. D. Hogg, J. K. Reid, and J. A. Scott, *Design of a multicore sparse Cholesky factorization using DAGs*, SIAM Journal on Scientific Computing, 32 (2010), pp. 3627–3649.

[30] J. D. Hogg and J. A. Scott, *The effects of scalings on the performance of a sparse symmetric indefinite solver*, Technical Report RAL-TR-2008-007, Rutherford Appleton Laboratory, 2008.

[31] J. D. Hogg and J. A. Scott, *A modern analyse phase for sparse tree-based direct methods*, Technical Report RAL-TR-2010-031, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2010. To appear in revised form in Numer. Linear Alg. Applic.

[32] ——, *A note on the solve phase of a multicore solver*, Technical Report RAL-TR-2010-007, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2010.

[33] HSL, *A collection of Fortran codes for large-scale scientific computation*, 2011. `http://www.hsl.rl.ac.uk/`.

[34] B. M. Irons, *A frontal solution program for finite-element analysis*, Intl. J. Numer. Methods Engrng., 2 (1970), pp. 5–32.

[35] G. Karypis and V. Kumar, *METIS - family of multilevel partitioning algorithms*, 1998. `http://glaros.dtc.umn.edu/gkhome/views/metis`.

[36] ——, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Scientific Computing, 20 (1999), pp. 359–392.

[37] J. Liu, *Modification of the Minimum-Degree algorithm by multiple elimination*, ACM Trans. Math. Softw., 11 (1985), pp. 141–153.

[38] J. W. H. Liu, *The multifrontal method for sparse matrix solution: theory and practice*, SIAM Review, 34 (1992), pp. 82–109.

[39] J. K. Reid and J. A. Scott, *An efficient out-of-core sparse symmetric indefinite solver*, Technical Report RAL-TR-2008-024, Rutherford Appleton Laboratory, 2008.

[40] ——, *Algorithm 891: a Fortran virtual memory system*, ACM Trans. Math. Softw., 36 (2009).

[41] ——, *An out-of-core sparse Cholesky solver*, ACM Trans. Math. Softw., 36 (2009). Article 9, 33 pages.

[42] ——, *Partial factorization of a dense symmetric indefinite matrix*, Technical Report RAL-TR-2009-015, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2009. To appear in revised form in ACM Trans. Math. Softw.

[43] ——, *Partial factorization of a dense symmetric indefinite matrix*, ACM Trans. Math. Softw., 38 (2011).

[44] V. Rotkin and S. Toledo, *The design and implementation of a new out-of-core sparse Cholesky factorization method*, ACM Trans. Math. Softw., 30 (2004), pp. 19–46.

[45] D. Ruiz, *A scaling algorithm to equilibrate both rows and columns norms in matrices*, Technical Report RAL-TR-2001-034, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2001.

[46] O. Schenk and K. Gärtner, *Solving unsymmetric sparse systems of linear equations with PARDISO*, Journal of Future Generation Computer Systems, 20 (2004), pp. 475–487.

[47] B. Speelpenning, *The generalized element method*, Report UIUCDCS-R-78-946, Department of Computer Science, University of Illinois, 1978.

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

[49] A. Wachter and L. T. Biegler, *On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming*, Mathematical Programming, 106(1) (2006). 25–57.

A Full tabulated results

Contents

Notes

Throughout the following sections, numbers in brackets indicate a note applies, as detailed below:

Note (1): Significant pivoting is required for stability (more than $0.01*n$ pivots were delayed by `HSL_MA97` with $u = 0.01$).

Note (2): PARDISO does not support the factorization of singular diagonal systems.

Note (3): In our tests, WSMP randomly encountered segfaults on these problems. Results are derived from runs where segfaults did not occur.

Note (4): WSMP required its dense row detection to be enabled to solve these problems.

Note (5): PARDISO returned a -4 error during solve (zero pivot, numerical factorization or iterative refinement problem).

Note (6): Forward error greater than 1×10^8 so that the scaled backwards error is likely not to be a reliable indicator of accuracy.

Note (7): The forward error of the solution given by this solver was greater than 1×10^{14}, or after 5 steps of iterative refinement the backwards error was 1×10^{-14} or greater.

A.1 `nemin` tuning: factorize results

A.1.1 Test Set 1: Small problems

Times for 100 factorizations, in seconds. Fastest time in bold.

Problem	Serial			Parallel		
`nemin=`	4	8	16	4	8	16
Cunningham/m3plates	**0.26**	**0.26**	**0.26**	0.37	**0.36**	**0.36**
Boeing/bcsstm37	0.56	**0.54**	0.55	0.85	**0.77**	**0.77**
GHS_indef/qpband	**0.51**	**0.51**	**0.51**	0.66	**0.65**	0.71
HB/zenios	0.17	0.15	**0.14**	0.21	0.16	**0.15**
HB/saylr3	**0.09**	**0.09**	**0.09**	0.10	0.10	**0.08**
HB/sherman1	0.09	**0.08**	0.09	0.10	0.09	**0.08**
Oberwolfach/filter2D	0.18	**0.16**	**0.16**	0.13	0.11	**0.10**
RAL/a2ensndl-00	2.15	1.72	**1.71**	2.10	**1.41**	1.43
RAL/a2ensndl-49 (1)	8.45	**8.10**	8.16	8.74	5.53	**5.43**
RAL/a2ensndl-62	3.35	2.95	**2.93**	3.33	2.46	**2.35**
Schenk_IBMNA/c-29 (1)	0.97	0.93	**0.91**	0.79	0.80	**0.73**
Boeing/nasa1824	**0.47**	**0.47**	0.48	0.30	**0.29**	0.30
GHS_indef/spmsrtls	2.41	**2.04**	2.08	4.54	3.18	**2.78**
IPSO/OPF_3754 (1)	1.59	**1.50**	1.57	1.81	1.29	**1.18**
GHS_indef/stokes64 (1)	**3.96**	4.06	4.52	**2.58**	2.60	2.71
GHS_indef/brainpc2 (1)	2.95	2.74	**2.64**	5.45	3.91	**3.36**
TSOPF/TSOPF_FS_b9_c6 (1)	3.28	3.14	**2.92**	4.01	3.91	**3.36**
Nemeth/nemeth05	2.41	**1.98**	**1.98**	3.13	2.59	**2.32**
Nemeth/nemeth10	2.34	**1.90**	2.12	3.13	2.54	**2.17**
FIDAP/ex14 (1)	3.46	**3.44**	3.46	3.16	3.14	**3.07**
Nemeth/nemeth15	3.42	**2.86**	2.87	3.79	3.30	**3.16**
GHS_indef/ncvxqp9	**6.39**	6.40	6.67	5.99	**5.91**	6.08
Schenk_IBMNA/c-41	4.16	**4.07**	4.08	3.58	3.40	**3.33**
GHS_indef/tuma2 (1)	1.94	**1.86**	2.03	1.26	1.16	**1.13**
Marini/eurqsa	**2.00**	2.01	2.01	1.21	1.11	**1.09**
Oberwolfach/rail_20209	2.73	**2.31**	2.43	2.05	1.66	**1.58**
Newman/hep-th (1)	**6.15**	6.36	6.28	4.70	4.64	**4.56**
Nemeth/nemeth20	17.86	16.74	**16.31**	19.33	17.92	**17.80**
PARSEC/Si2	1.62	1.49	**1.37**	1.04	0.97	**0.83**
Oberwolfach/t2dah_a	3.01	**2.95**	3.15	2.16	2.15	**2.13**
IPSO/TSC_OPF_300 (1)	**6.70**	6.71	6.71	7.14	7.22	**7.04**
Nemeth/nemeth25 (1)	13.69	11.09	**9.93**	15.57	12.44	**11.10**
Schenk_IBMNA/c-50 (1)	5.82	**5.63**	5.70	3.86	3.89	**3.70**
Newman/cond-mat (1)	40.81	**40.29**	41.08	25.88	26.35	**25.26**
Boeing/crystk01	9.03	8.95	**8.91**	4.13	**4.11**	4.43
TSOPF/TSOPF_FS_b39_c7 (1)	45.49	44.15	**42.98**	42.20	41.47	**37.81**
Boeing/bcsstk37 (1)	22.10	**21.56**	21.90	9.68	**9.18**	9.77
GHS_indef/exdata_1	27.06	**26.73**	27.02	19.88	**19.63**	19.68
TSOPF/TSOPF_FS_b162_c1 (1)	210.91	**133.41**	156.81	191.85	**101.36**	125.20
Boeing/crystk02	43.66	43.68	**43.32**	11.78	**11.61**	11.67

A.1.2 Test Set 2: Positive-definite problems

Times for 1 factorization, in seconds. Fastest time in bold.

Problem	Serial			Parallel		
`nemin=`	4	8	16	4	8	16
Mulvey/finan512	0.14	**0.13**	0.14	0.06	0.05	**0.04**
MaxPlanck/shallow_water1	0.15	**0.14**	**0.14**	0.06	**0.05**	**0.05**
UTEP/Dubcova3	0.47	**0.45**	0.46	0.12	0.12	**0.11**
Nasa/nasasrb	0.98	0.97	**0.96**	0.36	**0.33**	**0.33**
CEMW/tmt_sym	2.52	**2.40**	2.42	0.60	**0.52**	**0.52**
Schmid/thermal2	4.13	**3.90**	3.94	1.02	0.88	**0.83**
Rothberg/gearbox	3.97	**3.89**	3.90	0.86	**0.80**	0.86
INPRO/msdoor	4.18	**4.12**	4.21	0.82	0.87	**0.81**
DNVS/m_t1	3.66	3.64	**3.62**	0.73	**0.71**	0.75
McRae/ecology2	4.65	**4.41**	4.52	1.32	1.30	**1.09**
Boeing/pwtk	4.40	**4.29**	4.31	**0.80**	**0.80**	**0.80**
Chen/pkustk13	4.40	4.35	**4.34**	1.06	**1.03**	1.05
BenElechi/BenElechi1	5.11	**4.99**	5.01	1.03	**0.97**	0.98
Rothberg/cfd2	5.44	5.37	**5.35**	1.42	1.37	**1.36**
DNVS/thread	5.36	**5.31**	5.35	1.42	1.37	**1.36**
DNVS/shipsec8	6.40	**6.29**	6.44	1.45	1.45	**1.38**
DNVS/shipsec1	6.33	**6.22**	6.27	1.30	1.21	**1.19**
GHS_psdef/crankseg_2	6.33	**6.22**	6.27	1.30	1.21	**1.19**
DNVS/fcondp2	8.27	**8.06**	8.09	1.74	1.51	**1.46**
Schenk_AFE/af_shell3	9.73	**9.59**	9.65	1.78	**1.70**	1.77
DNVS/troll	9.38	**9.25**	9.27	1.82	**1.71**	1.79
GHS_psdef/bmwcra_1	10.21	10.10	**10.09**	**1.67**	1.83	1.80
DNVS/halfb	11.75	11.57	**11.53**	1.82	**1.71**	1.79
GHS_psdef/crankseg_1	10.79	10.79	**10.78**	2.86	2.69	**2.54**
Um/2cubes_sphere	12.19	12.10	**11.97**	2.45	2.31	**2.21**
GHS_psdef/ldoor	15.15	**15.00**	15.13	2.73	**2.63**	2.68
DNVS/ship_003	13.56	**13.28**	13.37	2.76	**2.71**	2.77
DNVS/fullb	16.92	**16.55**	16.61	3.45	3.48	**3.39**
Um/offshore	17.87	17.60	**17.48**	3.54	**3.32**	**3.32**
GHS_psdef/inline_1	24.69	24.51	**24.35**	4.13	4.13	**4.00**
Chen/pkustk14	25.39	24.91	**24.72**	4.47	4.32	**4.25**
GHS_psdef/apache2	29.02	28.57	**28.35**	5.19	**4.97**	5.09
Koutsovasilis/F1	35.93	35.67	**35.36**	**6.07**	6.12	6.10
Oberwolfach/boneS10	46.48	46.04	**45.94**	8.20	8.18	**7.92**
AMD/G3_circuit	45.61	**45.41**	45.65	8.22	**7.92**	8.09
ND/nd12k	108.78	101.02	**94.37**	21.96	19.69	**18.02**
JGD_Trefethen/Trefethen_20000	127.80	122.18	**119.14**	36.50	33.86	**32.45**
ND/nd24k	414.76	388.27	**367.69**	78.74	73.21	**66.57**
Oberwolfach/bone010	552.45	552.74	**552.07**	83.08	83.70	**81.46**
GHS_psdef/audikw_1	837.41	835.43	**834.76**	124.62	124.53	**123.55**

A.1.3 Test Set 3: General indefinite problems

Times for 1 factorization, in seconds. Fastest time in bold.

Problem	Serial			Parallel		
`nemin=`	4	8	16	4	8	16
Oberwolfach/t2dal	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**
GHS_indef/dixmaanl	0.05	**0.04**	**0.04**	**0.14**	0.18	0.23
Oberwolfach/rail_79841	0.16	**0.14**	0.15	0.13	**0.10**	0.11
GHS_indef/dawson5	0.47	**0.46**	0.47	0.22	**0.21**	**0.21**
Boeing/bcsstk39	0.56	**0.54**	0.56	0.33	0.32	**0.31**
GHS_indef/helm2d03	1.63	**1.57**	1.60	0.47	**0.45**	**0.45**
GHS_indef/copter2	1.36	**1.32**	**1.32**	0.40	**0.39**	0.40
Boeing/crystk03	1.17	**1.15**	1.16	0.32	0.34	**0.31**
Oberwolfach/filter3D	1.93	**1.88**	1.89	0.50	0.49	**0.45**
Boeing/pct20stif	2.81	**2.79**	2.80	1.67	1.67	**1.65**
Koutsovasilis/F2	2.45	**2.42**	2.44	1.24	1.23	**1.15**
Cunningham/qa8fk	3.69	**3.67**	3.69	**0.84**	0.87	0.88
Oberwolfach/gas_sensor	3.82	3.80	**3.78**	0.94	0.89	**0.85**
McRae/ecology1	5.24	**5.01**	5.13	2.03	1.84	**1.82**
Oberwolfach/t3dh	10.63	10.63	**10.62**	2.47	2.41	**2.39**
Lin/Lin	46.89	46.00	**45.06**	8.92	8.71	**8.44**
GHS_indef/sparsine	431.98	342.98	**278.13**	108.69	85.28	**66.79**
PARSEC/Ge99H100	1457.43	1328.82	**1210.42**	337.36	294.02	**252.14**
PARSEC/Ga10As10H30	1443.77	1295.37	**1173.54**	308.41	269.74	**233.39**
PARSEC/Ga19As19H42	1853.43	1685.49	**1535.43**	365.66	321.20	**277.69**

A.1.4 Test Set 4: KKT indefinite problems

Times for 1 factorization, in seconds. Fastest time in bold.

Problem	Serial			Parallel		
`nemin=`	4	8	16	4	8	16
GHS_indef/boyd1	0.76	**0.74**	**0.74**	0.82	0.82	**0.81**
GHS_indef/bmw3_2 (1)	6.29	**6.21**	6.24	1.57	1.59	**1.54**
GHS_indef/c-72 (1)	0.60	**0.59**	0.61	0.31	**0.26**	**0.26**
GHS_indef/ncvxqp7 (1)	39.81	31.95	**25.50**	27.91	21.14	**15.84**
Andrianov/mip1	29.89	29.52	**29.41**	15.40	**15.05**	15.17
GHS_indef/blockqp1	**0.08**	**0.08**	**0.08**	**0.12**	0.13	**0.12**
GHS_indef/boyd2	0.38	**0.27**	**0.27**	0.40	**0.25**	**0.25**
GHS_indef/a5esindl	0.04	0.04	**0.03**	0.06	0.04	**0.03**
GHS_indef/a2nnsnsl	0.06	**0.05**	**0.05**	0.09	0.05	**0.04**
GHS_indef/a0nsdsil	0.07	**0.05**	**0.05**	0.08	0.05	**0.04**
TSOPF/TSOPF_FS_b39_c30 (1)	**1.33**	1.36	1.47	0.72	**0.71**	0.73
GHS_indef/cont-201 (1)	1.06	**1.04**	1.07	0.32	0.33	**0.31**
GHS_indef/darcy003 (1)	0.85	**0.82**	0.86	0.50	**0.45**	**0.46**
GHS_indef/cont-300 (1)	3.83	**3.78**	3.87	1.28	**1.23**	1.31
GHS_indef/turon_m (1)	1.31	**1.29**	1.32	0.44	**0.43**	0.44
GHS_indef/d_pretok (1)	1.42	**1.41**	1.44	0.47	**0.44**	0.46
TSOPF/TSOPF_FS_b300_c3 (1)	10.74	10.51	**9.75**	3.86	3.64	**3.49**
GHS_indef/dtoc (1)	0.42	**0.18**	0.27	0.19	**0.09**	0.12
GHS_indef/aug2d (1)	1.13	**0.83**	1.06	0.83	0.56	**0.51**
GHS_indef/aug3d (1)	21.85	**17.07**	17.42	8.35	**6.99**	7.51

A.2 `nemin` tuning: solve results

A.2.1 Test Set 1: Small problems

Number of entries in factors and time in seconds for 10 sequential solves in serial. Fastest time in bold.

Problem	Entries			Times		
`nemin=`	4	8	16	4	8	16
Cunningham/m3plates	6.64e+03	6.64e+03	6.64e+03	**0.01**	**0.01**	**0.01**
Boeing/bcsstm37	1.50e+04	1.52e+04	1.52e+04	**0.01**	**0.01**	**0.01**
GHS_indef/qpband	5.00e+04	5.00e+04	5.00e+04	**0.01**	**0.01**	**0.01**
HB/zenios	1.90e+04	2.18e+04	2.73e+04	**0.00**	**0.00**	**0.00**
HB/saylr3	1.28e+04	1.62e+04	2.30e+04	**0.00**	**0.00**	**0.00**
HB/sherman1	1.28e+04	1.62e+04	2.30e+04	**0.00**	**0.00**	**0.00**
Oberwolfach/filter2D	2.71e+04	3.36e+04	4.61e+04	**0.00**	**0.00**	**0.00**
RAL/a2ensndl-00	1.93e+05	2.59e+05	2.60e+05	**0.02**	0.03	0.03
RAL/a2ensndl-49 (1)	6.83e+05	7.52e+05	7.62e+05	**0.03**	0.04	0.04
RAL/a2ensndl-62	2.20e+05	2.81e+05	2.84e+05	**0.02**	0.03	0.03
Schenk_IBMNA/c-29 (1)	7.96e+04	9.57e+04	1.27e+05	**0.01**	**0.01**	**0.01**
Boeing/nasa1824	7.47e+04	8.41e+04	1.00e+05	**0.00**	**0.00**	**0.00**
GHS_indef/spmsrtls	1.73e+05	2.40e+05	3.53e+05	**0.02**	0.03	**0.02**
IPSO/OPF_3754 (1)	1.33e+05	1.91e+05	2.63e+05	**0.01**	0.02	**0.01**
GHS_indef/stokes64 (1)	5.92e+05	6.67e+05	7.89e+05	**0.02**	**0.02**	**0.02**
GHS_indef/brainpc2 (1)	2.78e+05	3.83e+05	4.94e+05	**0.02**	0.03	**0.02**
TSOPF/TSOPF_FS_b9_c6 (1)	3.53e+05	4.33e+05	5.07e+05	0.02	**0.01**	**0.01**
Nemeth/nemeth05	2.41e+05	2.60e+05	2.99e+05	**0.01**	**0.01**	**0.01**
Nemeth/nemeth10	2.40e+05	2.61e+05	3.04e+05	**0.01**	**0.01**	**0.01**
FIDAP/ex14 (1)	1.41e+05	1.52e+05	1.79e+05	**0.00**	**0.00**	**0.00**
Nemeth/nemeth15	3.18e+05	3.39e+05	3.78e+05	**0.01**	**0.01**	**0.01**
GHS_indef/ncvxqp9 (1)	3.12e+05	3.81e+05	5.06e+05	**0.02**	**0.02**	0.02
Schenk_IBMNA/c-41	2.47e+05	2.88e+05	3.59e+05	**0.01**	**0.01**	0.01
GHS_indef/tuma2 (1)	2.47e+05	3.03e+05	4.23e+05	**0.01**	0.02	**0.01**
Marini/eurqsa	1.98e+05	2.85e+05	3.61e+05	**0.01**	**0.01**	**0.01**
Oberwolfach/rail_20209	4.06e+05	5.18e+05	7.07e+05	**0.02**	0.03	**0.02**
Newman/hep-th (1)	4.49e+05	4.81e+05	5.39e+05	**0.01**	**0.01**	**0.01**
Nemeth/nemeth20	5.38e+05	5.60e+05	5.97e+05	0.02	0.02	**0.01**
PARSEC/Si2	1.46e+05	1.56e+05	1.70e+05	**0.00**	**0.00**	**0.00**
Oberwolfach/t2dah_a	5.23e+05	5.87e+05	7.23e+05	**0.02**	**0.02**	**0.02**
IPSO/TSC_OPF_300 (1)	8.23e+05	8.32e+05	8.41e+05	**0.02**	**0.02**	**0.02**
Nemeth/nemeth25 (1)	8.12e+05	8.32e+05	8.69e+05	0.03	**0.02**	**0.02**
Schenk_IBMNA/c-50 (1)	5.72e+05	6.67e+05	8.59e+05	**0.03**	**0.03**	**0.03**
Newman/cond-mat (1)	1.83e+06	1.87e+06	2.08e+06	**0.04**	**0.04**	0.05
Boeing/crystk01	1.04e+06	1.09e+06	1.11e+06	**0.02**	**0.02**	**0.02**
TSOPF/TSOPF_FS_b39_c7 (1)	3.93e+06	3.96e+06	4.17e+06	**0.11**	**0.11**	**0.11**
Boeing/bcsstk37 (1)	2.87e+06	3.02e+06	3.23e+06	**0.09**	**0.09**	**0.09**
GHS_indef/exdata_1	1.15e+06	1.15e+06	1.16e+06	**0.02**	**0.02**	**0.02**
TSOPF/TSOPF_FS_b162_c1 (1)	4.73e+06	4.86e+06	4.93e+06	**0.09**	**0.09**	**0.09**
Boeing/crystk02	4.27e+06	4.40e+06	4.59e+06	**0.10**	**0.10**	**0.10**

A.2.2 Test Set 2: Positive-definite problems

Number of entries in factors and time in seconds for 10 sequential solves in serial. Fastest time in bold.

Problem	Entries			Times		
`nemin=`	4	8	16	4	8	16
Mulvey/finan512	2.36e+06	2.85e+06	3.84e+06	**0.13**	**0.13**	0.14
MaxPlanck/shallow_water1	2.38e+06	2.79e+06	3.53e+06	**0.14**	**0.14**	**0.14**
UTEP/Dubcova3	7.75e+06	8.80e+06	1.02e+07	**0.33**	0.34	0.36
Nasa/nasasrb	1.20e+07	1.24e+07	1.28e+07	0.31	**0.30**	0.31
CEMW/tmt_sym	3.32e+07	3.70e+07	4.39e+07	1.69	1.70	**1.68**
Schmid/thermal2	5.68e+07	6.30e+07	7.45e+07	3.05	2.98	**2.95**
Rothberg/gearbox	3.76e+07	3.88e+07	4.06e+07	0.96	**0.94**	0.96
INPRO/msdoor	5.29e+07	5.71e+07	6.35e+07	1.69	**1.67**	1.75
DNVS/m_t1	3.16e+07	3.23e+07	3.24e+07	**0.71**	**0.71**	**0.71**
McRae/ecology2	4.93e+07	5.55e+07	6.80e+07	**2.14**	2.23	2.39
Boeing/pwtk	4.87e+07	5.08e+07	5.28e+07	1.24	**1.22**	1.24
Chen/pkustk13	3.10e+07	3.18e+07	3.27e+07	0.73	**0.72**	**0.72**
BenElechi/BenElechi1	5.38e+07	5.59e+07	5.81e+07	1.37	**1.34**	1.37
Rothberg/cfd2	3.91e+07	4.00e+07	4.15e+07	**0.90**	**0.90**	0.91
DNVS/thread	2.43e+07	2.44e+07	2.48e+07	**0.47**	**0.47**	**0.47**
DNVS/shipsec8	3.59e+07	3.72e+07	3.95e+07	0.83	**0.81**	0.84
DNVS/shipsec1	3.94e+07	4.05e+07	4.28e+07	0.92	**0.89**	0.92
GHS_psdef/crankseg_2	4.40e+07	4.46e+07	4.55e+07	**0.87**	**0.87**	0.89
DNVS/fcondp2	5.20e+07	5.39e+07	5.61e+07	1.26	**1.24**	1.27
Schenk_AFE/af_shell3	9.36e+07	9.90e+07	1.05e+08	2.63	**2.54**	2.61
DNVS/troll	6.46e+07	6.65e+07	6.90e+07	1.52	**1.49**	1.51
GHS_psdef/bmwcra_1	7.04e+07	7.18e+07	7.40e+07	1.51	**1.50**	1.51
DNVS/halfb	6.59e+07	6.80e+07	7.12e+07	1.53	**1.50**	1.54
GHS_psdef/crankseg_1	4.57e+07	4.61e+07	4.67e+07	0.87	**0.86**	0.87
Um/2cubes_sphere	4.57e+07	4.65e+07	4.80e+07	0.99	**0.98**	0.99
GHS_psdef/ldoor	1.45e+08	1.55e+08	1.70e+08	4.27	**4.24**	4.40
DNVS/ship_003	6.02e+07	6.20e+07	6.48e+07	1.25	**1.23**	1.27
DNVS/fullb	7.45e+07	7.65e+07	7.97e+07	1.64	**1.60**	1.63
Um/offshore	8.63e+07	8.84e+07	9.23e+07	2.06	**2.04**	2.05
GHS_psdef/inline_1	1.76e+08	1.80e+08	1.85e+08	4.07	**4.05**	4.07
Chen/pkustk14	1.07e+08	1.09e+08	1.13e+08	2.14	**2.12**	2.15
GHS_psdef/apache2	1.42e+08	1.49e+08	1.59e+08	**3.78**	3.84	3.87
Koutsovasilis/F1	1.76e+08	1.79e+08	1.83e+08	3.79	**3.76**	3.77
Oberwolfach/boneS10	2.82e+08	2.90e+08	3.02e+08	6.68	**6.60**	6.70
AMD/G3_circuit	2.02e+08	2.13e+08	2.33e+08	**5.69**	5.96	6.07
ND/nd12k	1.17e+08	1.17e+08	1.18e+08	2.15	2.09	**2.06**
JGD_Trefethen/Trefethen_20000	8.88e+07	9.08e+07	9.36e+07	**1.62**	1.63	1.64
ND/nd24k	3.21e+08	3.22e+08	3.23e+08	5.95	5.83	**5.78**
Oberwolfach/bone010	1.08e+09	1.09e+09	1.11e+09	20.30	**20.27**	20.46
GHS_psdef/audikw_1	1.25e+09	1.26e+09	1.28e+09	**24.22**	24.39	24.49

A.2.3 Test Set 3: General indefinite problems

Number of entries in factors and time in seconds for 10 sequential solves in serial. Fastest time in bold.

Problem	Entries			Times		
`nemin=`	4	8	16	4	8	16
Oberwolfach/t2dal	1.33e+05	1.51e+05	1.92e+05	**0.01**	**0.01**	**0.01**
GHS_indef/dixmaanl	4.34e+05	5.40e+05	7.95e+05	**0.04**	0.06	0.05
Oberwolfach/rail_79841	2.12e+06	2.62e+06	3.46e+06	**0.13**	0.14	0.15
GHS_indef/dawson5	4.84e+06	5.15e+06	5.64e+06	**0.17**	**0.17**	**0.17**
Boeing/bcsstk39	6.71e+06	7.02e+06	7.60e+06	0.21	**0.20**	**0.20**
GHS_indef/helm2d03	2.08e+07	2.29e+07	2.66e+07	0.99	**0.98**	0.99
GHS_indef/copter2	9.98e+06	1.04e+07	1.13e+07	**0.28**	**0.28**	**0.28**
Boeing/crystk03	9.57e+06	9.84e+06	1.03e+07	**0.21**	**0.21**	0.22
Oberwolfach/filter3D	1.94e+07	2.01e+07	2.14e+07	0.57	**0.56**	**0.56**
Boeing/pct20stif	1.16e+07	1.20e+07	1.25e+07	**0.29**	**0.29**	**0.29**
Koutsovasilis/F2	2.08e+07	2.13e+07	2.21e+07	0.51	**0.50**	0.51
Cunningham/qa8fk	2.37e+07	2.43e+07	2.51e+07	**0.54**	**0.54**	**0.54**
Oberwolfach/gas_sensor	2.42e+07	2.47e+07	2.56e+07	**0.55**	**0.55**	**0.55**
McRae/ecology1	5.06e+07	5.68e+07	6.89e+07	**2.23**	2.33	2.54
Oberwolfach/t3dh	4.76e+07	4.81e+07	4.89e+07	**0.96**	**0.96**	0.97
Lin/Lin	1.11e+08	1.14e+08	1.18e+08	**2.42**	2.45	2.45
GHS_indef/sparsine	2.00e+08	2.02e+08	2.04e+08	4.66	4.18	**3.91**
PARSEC/Ge99H100	6.51e+08	6.54e+08	6.60e+08	13.04	12.57	**12.28**
PARSEC/Ga10As10H30	6.71e+08	6.74e+08	6.80e+08	13.33	12.74	**12.44**
PARSEC/Ga19As19H42	8.02e+08	8.06e+08	8.13e+08	16.30	15.71	**15.35**

A.2.4 Test Set 4: KKT indefinite problems

Number of entries in factors and time in seconds for 10 sequential solves in serial. Fastest time in bold.

Problem	Entries			Times		
`nemin=`	4	8	16	4	8	16
GHS_indef/boyd1	6.52e+05	6.53e+05	6.54e+05	**0.15**	0.16	**0.15**
GHS_indef/bmw3_2 (1)	4.73e+07	4.96e+07	5.22e+07	1.26	**1.24**	1.26
GHS_indef/c-72 (1)	4.11e+06	4.62e+06	5.59e+06	**0.20**	**0.20**	0.21
GHS_indef/ncvxqp7 (1)	3.84e+07	3.92e+07	4.07e+07	1.08	1.01	**0.96**
Andrianov/mip1	4.52e+07	4.54e+07	4.58e+07	0.89	0.88	**0.87**
GHS_indef/blockqp1	7.80e+05	7.80e+05	7.80e+05	**0.06**	**0.06**	**0.06**
GHS_indef/boyd2	2.07e+06	2.59e+06	2.59e+06	0.38	**0.31**	**0.31**
GHS_indef/a5esindl	3.70e+05	4.70e+05	6.30e+05	**0.04**	0.05	0.05
GHS_indef/a2nnsnsl	6.46e+05	8.41e+05	1.29e+06	**0.07**	0.09	0.09
GHS_indef/a0nsdsil	6.80e+05	8.80e+05	1.32e+06	**0.06**	0.09	0.08
TSOPF/TSOPF_FS_b39_c30 (1)	9.89e+06	1.14e+07	1.35e+07	**0.36**	0.39	0.44
GHS_indef/cont-201 (1)	1.04e+07	1.11e+07	1.25e+07	**0.33**	**0.33**	0.36
GHS_indef/darcy003 (1)	7.77e+06	9.18e+06	1.24e+07	**0.68**	0.70	0.69
GHS_indef/cont-300 (1)	2.90e+07	3.05e+07	3.38e+07	0.86	**0.85**	0.90
GHS_indef/turon_m (1)	1.35e+07	1.44e+07	1.64e+07	0.56	**0.54**	0.55
GHS_indef/d_pretok (1)	1.45e+07	1.53e+07	1.74e+07	0.57	**0.55**	0.56
TSOPF/TSOPF_FS_b300_c3 (1)	4.04e+07	4.47e+07	4.55e+07	1.03	1.05	**1.02**
GHS_indef/dtoc (1)	1.05e+07	1.02e+07	8.29e+06	0.09	**0.06**	0.08
GHS_indef/aug2d (1)	7.27e+06	6.20e+06	7.43e+06	0.17	**0.15**	0.17
GHS_indef/aug3d (1)	4.11e+07	3.44e+07	3.54e+07	0.70	**0.60**	0.61

A.3 HSL_MA97 Solve comparisons

A.3.1 Test Set 1: Small problems

Time in seconds for 10 solves. Fastest times in bold.

Problem	Supernodal		Multifrontal	
	1	8	1	8
Cunningham/m3plates	**0.01**	**0.01**	**0.01**	**0.01**
Boeing/bcsstm37	**0.01**	**0.01**	0.03	0.02
GHS_indef/qpband	**0.01**	**0.01**	0.02	**0.01**
HB/zenios	**0.00**	**0.00**	0.01	0.01
HB/saylr3	**0.00**	**0.00**	**0.00**	**0.00**
HB/sherman1	**0.00**	**0.00**	**0.00**	0.01
Oberwolfach/filter2D	**0.00**	**0.00**	**0.00**	0.01
RAL/a2ensndl-00	**0.03**	**0.03**	0.05	**0.03**
RAL/a2ensndl-49	**0.04**	**0.04**	0.06	**0.04**
RAL/a2ensndl-62	**0.03**	**0.03**	0.05	**0.03**
Schenk_IBMNA/c-29 (1)	**0.01**	**0.01**	**0.01**	0.02
Boeing/nasa1824	**0.00**	**0.00**	**0.00**	0.01
GHS_indef/spmsrtls	**0.03**	**0.03**	0.04	0.07
IPSO/OPF_3754 (1)	**0.02**	**0.02**	**0.02**	0.05
GHS_indef/stokes64 (1)	0.02	**0.01**	0.02	0.02
GHS_indef/brainpc2(1)	**0.03**	0.04	0.05	0.11
TSOPF/TSOPF_FS_b9_c6(1)	**0.01**	0.03	0.02	0.05
Nemeth/nemeth05	**0.01**	0.02	0.02	0.05
Nemeth/nemeth10	**0.01**	0.02	0.02	0.04
FIDAP/ex14 (1)	**0.00**	0.01	0.01	0.01
Nemeth/nemeth15	**0.01**	0.03	0.02	0.04
GHS_indef/ncvxqp9 (1)	**0.02**	**0.02**	0.04	0.04
Schenk_IBMNA/c-41	**0.01**	0.02	0.03	0.04
GHS_indef/tuma2 (1)	**0.02**	**0.02**	**0.02**	0.03
Marini/eurqsa	**0.01**	0.02	**0.01**	0.03
Oberwolfach/rail_20209	0.03	**0.02**	0.03	0.03
Newman/hep-th (1)	**0.01**	**0.01**	0.02	0.02
Nemeth/nemeth20	**0.02**	**0.02**	**0.02**	0.05
PARSEC/Si2	**0.00**	**0.00**	**0.00**	0.01
Oberwolfach/t2dah_a	0.02	**0.01**	0.02	0.02
IPSO/TSC_OPF_300 (1)	**0.02**	**0.02**	**0.02**	0.04
Nemeth/nemeth25 (1)	**0.02**	**0.02**	0.03	0.06
Schenk_IBMNA/c-50 (1)	0.03	**0.02**	0.05	0.03
Newman/cond-mat (1)	**0.04**	0.05	0.06	0.06
Boeing/crystk01	0.02	0.02	0.02	**0.01**
TSOPF/TSOPF_FS_b39_c7 (1)	0.11	**0.10**	0.13	0.12
Boeing/bcsstk37 (1)	0.09	0.07	0.10	**0.05**
GHS_indef/exdata_1	**0.02**	**0.02**	**0.02**	0.03
TSOPF/TSOPF_FS_b162_c1 (1)	**0.09**	0.12	0.11	0.14
Boeing/crystk02	0.10	0.08	0.10	**0.05**

A.3.2 Test Set 2: Positive-definite problems

Time in seconds for 10 solves. Fastest times in bold.

Problem	Supernodal		Multifrontal	
	1	8	1	8
Mulvey/finan512	**0.13**	0.14	0.16	0.17
MaxPlanck/shallow_water1	0.14	0.12	0.19	**0.10**
UTEP/Dubcova3	0.34	0.27	0.39	**0.15**
Nasa/nasasrb	0.30	0.25	0.33	**0.17**
CEMW/tmt_sym	1.70	1.33	1.97	**0.71**
Schmid/thermal2	2.98	2.37	3.46	**1.16**
Rothberg/gearbox	0.94	0.76	1.00	**0.42**
INPRO/msdoor	1.67	1.32	1.80	**0.69**
DNVS/m_t1	0.71	0.58	0.73	**0.33**
McRae/ecology2	2.23	1.69	2.48	**0.88**
Boeing/pwtk	1.22	0.97	1.28	**0.55**
Chen/pkustk13	0.72	0.62	0.75	**0.36**
BenElechi/BenElechi1	1.34	1.10	1.42	**0.64**
Rothberg/cfd2	0.90	0.73	0.94	**0.42**
DNVS/thread	0.47	0.45	0.48	**0.35**
DNVS/shipsec8	0.81	0.70	0.85	**0.48**
DNVS/shipsec1	0.89	0.73	0.94	**0.42**
GHS_psdef/crankseg_2	0.87	0.71	0.89	**0.42**
DNVS/fcondp2	1.24	1.00	1.31	**0.62**
Schenk_AFE/af_shell3	2.54	1.97	2.69	**0.94**
DNVS/troll	1.49	1.22	1.56	**0.62**
GHS_psdef/bmwcra_1	1.50	1.20	1.56	**0.62**
DNVS/halfb	1.50	1.24	1.58	**0.73**
GHS_psdef/crankseg_1	0.86	0.81	0.88	**0.63**
Um/2cubes_sphere	0.98	0.83	1.03	**0.49**
GHS_psdef/ldoor	4.24	3.36	4.53	**1.67**
DNVS/ship_003	1.23	1.02	1.28	**0.61**
DNVS/fullb	1.60	1.32	1.67	**0.80**
Um/offshore	2.04	1.64	2.15	**0.87**
GHS_psdef/inline_1	4.05	3.20	4.22	**1.67**
Chen/pkustk14	2.12	1.77	2.20	**1.10**
GHS_psdef/apache2	3.84	3.07	4.35	**1.70**
Koutsovasilis/F1	3.76	2.98	3.86	**1.63**
Oberwolfach/boneS10	6.60	5.36	6.94	**2.93**
AMD/G3_circuit	5.96	4.60	6.58	**2.63**
ND/nd12k	2.09	1.96	2.17	**1.51**
JGD_Trefethen/Trefethen_20000	1.63	1.91	**1.65**	1.98
ND/nd24k	5.83	5.37	6.02	**3.89**
Oberwolfach/bone010	20.30	16.45	20.92	**9.59**
GHS_psdef/audikw_1	24.39	19.10	25.55	**11.40**

A.3.3 Test Set 3: General indefinite problems

Time in seconds for 10 solves. Fastest times in bold.

Problem	Supernodal		Multifrontal	
	1	8	1	8
Oberwolfach/t2dal	**0.01**	0.02	**0.01**	0.02
GHS_indef/dixmaanl	**0.06**	0.07	0.12	0.33
Oberwolfach/rail_79841	0.14	0.10	0.17	**0.09**
GHS_indef/dawson5	0.17	0.12	0.19	**0.07**
Boeing/bcsstk39	0.20	0.16	0.21	**0.11**
GHS_indef/helm2d03	0.98	0.76	1.14	**0.41**
GHS_indef/copter2	0.28	0.24	0.32	**0.16**
Boeing/crystk03	0.21	0.17	0.22	**0.11**
Oberwolfach/filter3D	0.56	0.43	0.60	**0.23**
Boeing/pct20stif	0.29	0.25	0.30	**0.17**
Koutsovasilis/F2	0.50	0.42	0.53	**0.28**
Cunningham/qa8fk	0.54	0.46	0.56	**0.27**
Oberwolfach/gas_sensor	0.55	0.45	0.57	**0.28**
McRae/ecology1	2.33	1.76	2.60	**0.95**
Oberwolfach/t3dh	0.96	0.82	0993	**0.54**
Lin/Lin	2.45	2.03	2.64	**1.31**
GHS_indef/sparsine	4.18	4.09	4.25	**3.33**
PARSEC/Ge99H100	12.57	11.32	13.00	**8.29**
PARSEC/Ga10As10H30	12.74	12.11	13.19	**9.10**
PARSEC/Ga19As19H42	15.71	13.87	16.23	**10.12**

A.3.4 Test Set 4: KKT problems

Time in seconds for 10 solves. Fastest times in bold.

Problem	Supernodal		Multifrontal	
	1	8	1	8
GHS_indef/boyd1	0.16	**0.12**	0.56	0.34
GHS_indef/bmw3_2 (1)	1.24	1.03	1.32	**0.66**
GHS_indef/c-72 (1)	0.20	0.16	0.29	**0.14**
GHS_indef/ncvxqp7 (1)	1.01	**1.15**	1.13	1.26
Andrianov/mip1	0.88	0.95	0.91	**0.83**
GHS_indef/blockqp1	0.06	**0.05**	0.14	0.08
GHS_indef/boyd2	0.31	**0.23**	0.72	0.43
GHS_indef/a5esindl	0.05	**0.04**	0.09	0.10
GHS_indef/a2nnsnsl	0.09	**0.07**	0.14	0.12
GHS_indef/a0nsdsil	0.09	**0.06**	0.13	0.09
TSOPF/TSOPF_FS_b39_c30 (1)	0.39	0.32	0.48	**0.22**
GHS_indef/cont-201 (1)	0.33	0.26	0.39	**0.16**
GHS_indef/darcy003 (1)	0.70	0.56	0.92	**0.32**
GHS_indef/cont-300 (1)	0.85	0.70	0.97	**0.45**
GHS_indef/turon_m (1)	0.54	0.42	0.64	**0.24**
GHS_indef/d_pretok (1)	0.55	0.43	0.65	**0.25**
TSOPF/TSOPF_FS_b300_c3 (1)	1.05	0.83	1.17	**0.47**
GHS_indef/dtoc (1)	0.06	0.06	0.08	**0.05**
GHS_indef/aug2d (1)	0.15	**0.14**	0.18	**0.14**
GHS_indef/aug3d (1)	**0.60**	0.73	0.63	0.70

A.4 `HSL_MA97` $u = 0.01$ vs $u = 0.001$ comparison

A.4.1 Test Set 1: Small problems

Number of entries in factors, forward error, and factorization times in serial and parallel (seconds).

Problem	Entries		Forward error		Serial		8 cores	
`u =`	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001
Cunningham/m3plates	6.64e+03	6.64e+03	1.00e+00	1.00e+00	0.26	0.26	0.26	0.26
Boeing/bcsstm37	1.52e+04	1.52e+04	1.00e+00	1.00e+00	0.54	0.54	0.54	0.54
GHS_indef/qpband	5.00e+04	5.00e+04	2.22e-16	2.22e-16	0.51	0.51	0.51	0.51
HB/zenios	2.18e+04	2.17e+04	1.00e+00	1.00e+00	0.15	0.15	0.15	0.15
HB/saylr3	1.62e+04	1.62e+04	1.00e+00	1.00e+00	0.09	0.09	0.09	0.09
HB/sherman1	1.62e+04	1.62e+04	1.91e-13	1.91e-13	0.08	0.08	0.08	0.08
Oberwolfach/filter2D	3.36e+04	3.36e+04	3.31e-14	3.31e-14	0.16	0.16	0.16	0.16
jhogg/a2ensndl-ipopt-ma57-0	2.59e+05	2.59e+05	1.71e-14	1.71e-14	1.72	1.71	1.72	1.72
jhogg/a2ensndl-ipopt-ma57-49	7.52e+05	7.45e+05	9.54e+04	3.31e+05	8.10	7.86	8.10	7.86
jhogg/a2ensndl-ipopt-ma57-62	2.85e+05	2.59e+05	9.54e+04	3.31e+05	2.94	2.83	2.94	2.83
Schenk_IBMNA/c-29 (1)	9.56e+04	9.47e+04	4.66e-12	9.37e-12	0.93	0.92	0.93	0.92
Boeing/nasa1824	8.41e+04	8.40e+04	3.09e-12	3.10e-12	0.47	0.47	0.47	0.47
GHS_indef/spmsrtls	2.40e+05	2.40e+05	2.49e-12	2.12e-12	2.04	2.03	2.04	2.02
IPSO/OPF_3754 (1)	1.91e+05	1.88e+05	7.51e-10	9.46e-09	1.50	1.46	1.50	1.45
GHS_indef/stokes64 (1)	6.67e+05	6.67e+05	7.99e-01	7.99e-01	4.06	4.06	2.56	2.63
GHS_indef/brainpc2 (1)	3.71e+05	2.92e+05	7.57e-12	2.27e-12	2.74	1.90	2.74	1.89
TSOPF/TSOPF_FS_b9_c6 (1)	4.06e+05	3.89e+05	1.89e-08	4.87e-08	3.14	2.96	3.13	2.95
Nemeth/nemeth05	2.60e+05	2.60e+05	1.22e-15	1.22e-15	1.98	1.99	1.98	1.99
Nemeth/nemeth10	2.61e+05	2.61e+05	1.55e-15	1.55e-15	1.90	1.91	1.91	1.92
FIDAP/ex14 (1)	1.52e+05	1.43e+05	7.04e-06	2.89e-05	3.44	3.39	3.45	3.39
Nemeth/nemeth15	3.39e+05	3.39e+05	3.40e-14	3.40e-14	2.86	2.87	2.87	2.88
GHS_indef/ncvxqp9 (1)	3.80e+05	3.78e+05	1.33e-07	2.35e-07	6.40	6.37	6.40	6.37
Schenk_IBMNA/c-41	2.88e+05	2.82e+05	9.14e-07	2.17e-06	4.07	4.00	4.06	4.00
GHS_indef/tuma2 (1)	3.03e+05	3.00e+05	7.24e-13	2.73e-12	1.86	1.83	1.86	1.83
Marini/eurqsa	2.81e+05	2.80e+05	8.18e+04	2.36e+04	2.01	2.00	2.01	2.00
Oberwolfach/rail_20209	5.18e+05	5.18e+05	1.74e-12	1.74e-12	2.31	2.31	1.66	1.62
Newman/hep-th (1)	5.02e+05	5.01e+05	8.59e+05	6.72e+05	6.36	5.90	4.64	4.07
Nemeth/nemeth20	5.60e+05	5.60e+05	6.97e-14	1.41e-13	16.74	16.74	17.92	17.74
PARSEC/Si2	1.56e+05	1.56e+05	2.16e-13	2.16e-13	1.49	1.49	0.97	0.93
Oberwolfach/t2dah_a	5.87e+05	5.87e+05	6.73e-03	6.73e-03	2.95	2.96	2.15	2.13
IPSO/TSC_OPF_300 (1)	8.32e+05	8.25e+05	3.11e-08	1.56e-06	6.71	6.65	7.22	7.07
Nemeth/nemeth25 (1)	8.31e+05	8.31e+05	1.19e-13	1.23e-13	11.09	11.10	12.44	12.55
Schenk_IBMNA/c-50 (1)	6.67e+05	6.51e+05	8.65e-12	4.31e-11	5.63	5.47	3.89	3.64
Newman/cond-mat (1)	1.91e+06	1.91e+06	1.52e+06	1.64e+06	40.29	38.61	26.35	23.02
Boeing/crystk01	1.09e+06	1.09e+06	1.17e-02	1.17e-02	8.95	8.95	4.11	4.11
TSOPF/TSOPF_FS_b39_c7 (1)	3.96e+06	3.57e+06	9.25e-07	1.89e-06	44.15	34.05	41.47	32.51
Boeing/bcsstk37	3.02e+06	3.00e+06	1.97e-05	1.81e-05	21.56	21.39	9.18	9.30
GHS_indef/exdata_1	1.15e+06	1.15e+06	4.02e-10	3.97e-10	26.73	26.71	19.63	19.49
TSOPF/TSOPF_FS_b162_c1 (1)	4.90e+06	4.51e+06	1.19e-06	3.38e-05	133.41	114.35	101.36	86.04
Boeing/crystk02	4.40e+06	4.40e+06	6.23e-02	6.23e-02	43.68	43.62	11.61	12.03

A.4.2 Test Set 3: General indefinite problems, 8 cores

Number of entries in factors, forward error, and factorization times in serial and parallel (seconds).

Problem	Entries		Forward error		Serial		8 cores	
`u` =	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001
Oberwolfach/t2dal	1.51e+05	1.51e+05	3.59e-03	3.59e-03	0.01	0.01	0.01	0.01
GHS_indef/dixmaanl	5.40e+05	5.40e+05	7.16e-13	1.45e-12	0.04	0.04	0.04	0.04
Oberwolfach/rail_79841	2.62e+06	2.62e+06	5.90e-12	5.90e-12	0.14	0.14	0.10	0.11
GHS_indef/dawson5	5.15e+06	5.15e+06	6.82e-11	3.36e-10	0.46	0.46	0.21	0.21
Boeing/bcsstk39	7.02e+06	7.02e+06	2.17e-11	2.17e-11	0.54	0.55	0.32	0.31
GHS_indef/helm2d03	2.29e+07	2.29e+07	1.47e-11	1.51e-11	1.57	1.57	0.45	0.47
GHS_indef/copter2	1.04e+07	1.04e+07	1.41e-10	1.99e-10	1.32	1.32	0.39	0.39
Boeing/crystk03	9.84e+06	9.84e+06	2.80e-02	2.80e-02	1.15	1.15	0.34	0.33
Oberwolfach/filter3D	2.01e+07	2.01e+07	7.51e-13	7.51e-13	1.88	1.88	0.49	0.48
Boeing/pct20stif	1.20e+07	1.20e+07	3.73e-10	4.28e-10	2.79	2.78	1.67	1.68
Koutsovasilis/F2	2.13e+07	2.13e+07	5.80e-01	5.80e-01	2.42	2.43	1.23	1.21
Cunningham/qa8fk	2.43e+07	2.43e+07	6.49e-01	6.49e-01	3.67	3.67	0.87	0.85
Oberwolfach/gas_sensor	2.47e+07	2.47e+07	6.36e-11	6.36e-11	3.80	3.81	0.89	0.92
McRae/ecology1	5.68e+07	5.68e+07	9.93e-01	9.93e-01	5.01	5.01	1.84	1.79
Oberwolfach/t3dh	4.81e+07	4.81e+07	6.24e-05	6.24e-05	10.63	10.63	2.41	2.46
Lin/Lin	1.14e+08	1.14e+08	4.34e-11	4.34e-11	46.00	46.02	8.71	8.75
GHS_indef/sparsine	2.02e+08	2.02e+08	1.71e-08	3.15e-08	342.98	342.87	85.28	85.63
PARSEC/Ge99H100	6.54e+08	6.54e+08	1.60e-10	4.35e-10	1328.82	1328.18	294.02	298.59
PARSEC/Ga10As10H30	6.74e+08	6.74e+08	6.65e-11	6.08e-11	1295.37	1295.44	269.74	274.27
PARSEC/Ga19As19H42	8.06e+08	8.06e+08	7.94e-11	1.82e-10	1685.49	1685.07	321.20	315.78

A.4.3 Test Set 4: KKT problems, Serial

Number of entries in factors, forward error, and factorization times in serial and parallel (seconds).

Problem	Entries		Forward error		Serial		8 cores	
`u` =	0.01	0.001	0.01	0.001	0.01	0.001	0.01	0.001
GHS_indef/boyd1 (1)	6.53e+05	6.53e+05	3.85e-06	3.85e-06	0.74	0.74	0.74	0.74
GHS_indef/bmw3_2 (1)	4.96e+07	4.91e+07	1.17e-06	7.21e-07	6.21	6.10	1.59	1.57
GHS_indef/c-72 (1)	4.62e+06	4.60e+06	1.24e-12	1.56e-11	0.59	0.59	0.26	0.26
GHS_indef/ncvxqp7 (1)	3.92e+07	3.91e+07	2.55e-02	7.81e-02	31.95	32.16	21.14	21.30
Andrianov/mip1	4.54e+07	4.53e+07	8.90e-09	3.62e-08	29.52	29.49	15.05	15.20
GHS_indef/blockqp1	7.80e+05	7.80e+05	1.73e-11	1.73e-11	0.08	0.08	0.08	0.08
GHS_indef/boyd2	2.59e+06	2.59e+06	7.87e-05	7.87e-05	0.27	0.27	0.27	0.27
GHS_indef/a5esindl	4.70e+05	4.70e+05	2.98e-11	2.97e-11	0.04	0.04	0.04	0.04
GHS_indef/a2nnsnsl	8.41e+05	8.41e+05	8.84e-11	9.57e-11	0.05	0.05	0.05	0.05
GHS_indef/a0nsdsil	8.80e+05	8.80e+05	4.11e-11	4.17e-11	0.05	0.05	0.05	0.05
TSOPF/TSOPF_FS_b39_c30 (1)	1.14e+07	8.94e+06	1.13e-06	8.53e-07	1.36	0.81	0.71	0.45
GHS_indef/cont-201 (1)	1.11e+07	1.10e+07	2.85e-08	1.02e-07	1.04	1.03	0.33	0.30
GHS_indef/darcy003 (1)	9.18e+06	9.18e+06	1.67e-13	1.67e-13	0.82	0.82	0.45	0.48
GHS_indef/cont-300 (1)	3.04e+07	3.04e+07	1.11e-07	2.97e-07	3.78	3.75	1.23	1.19
GHS_indef/turon_m (1)	1.44e+07	1.44e+07	9.57e-01	9.99e-01	1.29	1.29	0.43	0.42
GHS_indef/d_pretok (1)	1.53e+07	1.52e+07	9.69e-01	9.98e-01	1.41	1.40	0.44	0.43
TSOPF/TSOPF_FS_b300_c3 (1)	4.46e+07	4.09e+07	9.30e-06	7.90e-04	10.51	9.13	3.64	3.19
GHS_indef/dtoc (1)	2.06e+06	1.07e+06	9.17e+05	1.24e+06	0.18	0.09	0.18	0.09
GHS_indef/aug2d (1)	6.20e+06	6.20e+06	1.00e+00	1.00e+00	0.83	0.83	0.83	0.83
GHS_indef/aug3d (1)	3.44e+07	3.44e+07	1.00e+00	1.00e+00	17.07	17.06	6.99	7.02

A.5 Analyse phase comparison

A.5.1 Test Set 1: Small problems

Times for 1 analyse, including ordering, in seconds. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Cunningham/m3plates	0.22	0.01	**0.00**	**0.00**	0.15
Boeing/bcsstm37	**0.00**	0.02	0.01	(7) 1.63	0.02
GHS_indef/qpband	**0.00**	0.01	0.01	0.22	0.07
HB/zenios	**0.00**	**0.00**	**0.00**	(7) 0.78	0.03
HB/saylr3	**0.00**	**0.00**	**0.00**	0.78	0.01
HB/sherman1	**0.00**	**0.00**	**0.00**	0.38	0.01
Oberwolfach/filter2D	**0.00**	**0.00**	**0.00**	0.76	0.01
RAL/a2ensndl-00	**0.02**	0.03	**0.02**	0.67	(3) 0.13
RAL/a2ensndl-49 (1)	**0.02**	0.03	**0.02**	(7) 0.46	0.13
RAL/a2ensndl-62	**0.02**	0.03	**0.02**	0.63	(3) 0.13
Schenk_IBMNA/c-29 (1)	**0.00**	0.01	0.01	0.38	0.09
Boeing/nasa1824	**0.00**	**0.00**	**0.00**	0.39	0.04
GHS_indef/spmsrtls	**0.01**	0.02	0.02	0.66	0.17
IPSO/OPF_3754 (1)	**0.01**	0.02	0.02	(7) 0.73	0.10
GHS_indef/stokes64 (1)	**0.01**	0.02	**0.01**	0.52	0.22
GHS_indef/brainpc2 (1)	**0.01**	0.08	0.02	0.78	0.25
TSOPF/TSOPF_FS_b9_c6 (1)	**0.01**	0.02	0.02	(7) 0.59	0.22
Nemeth/nemeth05	**0.02**	0.04	0.05	0.76	0.22
Nemeth/nemeth10	**0.02**	0.03	0.04	0.83	0.21
FIDAP/ex14 (1)	**0.00**	0.01	0.01	0.58	0.10
Nemeth/nemeth15	**0.03**	0.05	0.06	0.40	0.29
GHS_indef/ncvxqp9 (1)	**0.01**	0.02	0.02	0.30	0.21
Schenk_IBMNA/c-41	**0.01**	0.02	0.02	0.81	0.38
GHS_indef/tuma2 (1)	**0.01**	**0.01**	**0.01**	0.24	0.10
Marini/eurqsa	**0.01**	**0.01**	**0.01**	(7) 0.50	0.13
Oberwolfach/rail_20209	**0.02**	0.03	0.03	0.81	0.18
Newman/hep-th (1)	0.05	**0.01**	**0.01**	0.71	(7) 0.12
Nemeth/nemeth20	**0.06**	0.08	0.11	0.97	0.64
PARSEC/Si2	**0.01**	**0.01**	**0.01**	0.78	0.02
Oberwolfach/t2dah_a	**0.01**	0.02	0.02	(7) 0.89	0.16
IPSO/TSC_OPF_300 (1)	**0.02**	0.04	0.07	0.53	0.57
Nemeth/nemeth25 (1)	**0.11**	0.14	0.19	0.74	0.64
Schenk_IBMNA/c-50 (1)	**0.03**	0.04	**0.03**	0.52	0.89
Newman/cond-mat (1)	0.11	0.11	0.11	0.21	(4,7) 0.20
Boeing/crystk01	**0.01**	0.02	0.03	(7) 0.27	0.21
TSOPF/TSOPF_FS_b39_c7 (1)	**0.06**	0.08	0.10	(7) 0.71	0.43
Boeing/bcsstk37 (1)	**0.03**	0.07	0.10	0.54	0.78
GHS_indef/exdata_1	**0.05**	0.12	0.20	0.72	2.86
TSOPF/TSOPF_FS_b162_c1 (1)	**0.05**	0.07	0.08	(7) 0.63	0.34
Boeing/crystk02	**0.14**	**0.14**	0.16	(7) 0.78	0.71

A.5.2 Test Set 2: Positive-definite problems

Times for 1 analyse, including ordering, in seconds. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Mulvey/finan512	**0.52**	0.59	0.60	1.01	0.88
MaxPlanck/shallow_water1	**0.38**	0.46	0.46	1.34	0.57
UTEP/Dubcova3	**1.19**	1.22	1.30	1.99	2.93
Nasa/nasasrb	0.58	**0.19**	0.24	1.11	1.82
CEMW/tmt_sym	5.59	**5.56**	5.62	6.01	6.52
Schmid/thermal2	**10.86**	10.88	11.04	11.78	14.27
Rothberg/gearbox	1.74	**1.61**	1.82	2.58	6.45
INPRO/msdoor	1.82	**1.75**	2.28	3.12	12.39
DNVS/m_t1	0.78	**0.53**	0.79	1.98	6.03
McRae/ecology2	6.71	**1.15**	1.21	6.93	8.43
Boeing/pwtk	1.16	**0.98**	1.24	1.83	7.50
Chen/pkustk13	**1.06**	1.31	1.49	1.41	4.77
BenElechi/BenElechi1	1.14	**0.95**	**1.23**	2.06	8.39
Rothberg/cfd2	2.15	**1.95**	2.00	2.42	3.89
DNVS/thread	**0.55**	0.64	0.78	1.41	2.90
DNVS/shipsec8	0.69	**0.54**	0.70	1.80	4.73
DNVS/shipsec1	0.79	**0.62**	0.80	1.15	5.30
GHS_psdef/crankseg_2	**1.19**	1.72	2.20	2.01	8.08
DNVS/fcondp2	1.06	**0.87**	1.13	2.27	7.52
Schenk_AFE/af_shell3	2.18	**1.81**	2.15	3.01	11.78
DNVS/troll	1.58	**1.32**	**1.59**	2.40	8.28
GHS_psdef/bmwcra_1	2.10	**1.78**	2.05	2.82	8.30
DNVS/halfb	1.33	**1.05**	1.35	2.26	8.37
GHS_psdef/crankseg_1	0.94	**0.68**	1.02	1.73	6.10
Um/2cubes_sphere	1.59	1.31	**1.32**	1.96	2.83
GHS_psdef/ldoor	4.50	**4.27**	5.45	6.21	30.90
DNVS/ship_003	0.95	**0.67**	**0.86**	1.36	6.32
DNVS/fullb	1.33	**1.01**	1.30	2.15	8.34
Um/offshore	4.45	**3.90**	3.97	4.65	6.90
GHS_psdef/inline_1	7.21	**6.60**	7.66	8.75	27.17
Chen/pkustk14	1.99	**1.50**	1.91	3.14	10.89
GHS_psdef/apache2	7.41	**6.45**	**6.45**	7.42	11.45
Koutsovasilis/F1	6.12	**5.31**	6.12	7.23	21.54
Oberwolfach/boneS10	11.35	**10.31**	11.64	12.88	42.91
AMD/G3_circuit	12.12	3.24	**3.23**	12.43	15.89
ND/nd12k	3.86	4.39	4.86	**4.34**	13.34
JGD_Trefethen/Trefethen_20000	**0.20**	0.48	0.47	1.47	(4) 1.12
ND/nd24k	**9.15**	9.52	10.37	9.43	32.37
Oberwolfach/bone010	19.83	**13.04**	14.62	18.57	70.90
GHS_psdef/audikw_1	22.99	15.39	**17.33**	21.93	76.02

A.5.3 Test Set 3: General indefinite problems

Times for 1 analyse, including ordering, in seconds. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Oberwolfach/t2dal	**0.00**	0.01	0.01	(5) 0.41	0.04
GHS_indef/dixmaanl	**0.03**	0.29	0.06	0.81	0.29
Oberwolfach/rail_79841	**0.06**	0.12	0.12	0.84	0.80
GHS_indef/dawson5	**0.06**	0.14	0.14	1.29	1.11
Boeing/bcsstk39	**0.05**	0.13	0.16	0.41	1.47
GHS_indef/helm2d03	**2.75**	2.81	2.77	3.36	4.09
GHS_indef/copter2	**0.74**	0.76	**0.74**	1.34	1.40
Boeing/crystk03	**0.27**	**0.27**	0.30	(7) 1.19	1.37
Oberwolfach/filter3D	1.89	**1.85**	1.89	2.27	3.20
Boeing/pct20stif	**0.08**	0.20	0.25	0.85	3.02
Koutsovasilis/F2	**0.19**	0.46	0.61	2.17	4.16
Cunningham/qa8fk	1.24	**1.17**	**1.17**	(5) 1.99	2.30
Oberwolfach/gas_sensor	1.23	**1.17**	**1.17**	1.44	2.27
McRae/ecology1	**0.75**	1.29	1.21	(7) 7.36	8.83
Oberwolfach/t3dh	2.62	2.50	**2.60**	(7) 2.98	4.70
Lin/Lin	2.88	2.47	**2.08**	2.84	4.90
GHS_indef/sparsine	3.42	4.12	**2.12**	2.96	6.73
PARSEC/Ge99H100	8.25	9.09	**3.99**	7.13	35.52
PARSEC/Ga10As10H30	8.07	7.41	**3.41**	6.38	30.65
PARSEC/Ga19As19H42	9.96	11.39	**4.54**	8.73	39.84

A.5.4 Test Set 4: KKT indefinite problems

Times for 1 analyse, including ordering, in seconds. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
GHS_indef/boyd1	9.05	0.21	**0.11**	15.34	3.98
GHS_indef/bmw3_2 (1)	1.53	**1.41**	1.66	2.07	8.20
GHS_indef/c-72 (1)	0.87	0.18	**0.15**	0.99	6.94
GHS_indef/ncvxqp7 (1)	1.34	1.63	**1.27**	1.48	3.06
Andrianov/mip1	**0.38**	0.65	1.02	4.05	12.34
GHS_indef/blockqp1	**0.02**	0.07	0.07	1.22	0.22
GHS_indef/boyd2	**0.14**	0.32	0.27	16.73	1.83
GHS_indef/a5esindl	**0.02**	0.05	0.04	0.35	0.19
GHS_indef/a2nnsnsl	**0.05**	0.07	0.06	0.95	0.28
GHS_indef/a0nsdsil	**0.03**	0.07	0.06	0.75	0.28
TSOPF/TSOPF_FS_b39_c30 (1)	**1.35**	1.44	1.50	(7) 1.62	1.74
GHS_indef/cont-201 (1)	0.58	0.55	0.54	1.10	**0.51**
GHS_indef/darcy003 (1)	**0.34**	0.59	0.56	3.64	2.53
GHS_indef/cont-300 (1)	1.16	0.21	**0.19**	(7) 1.96	1.03
GHS_indef/turon_m (1)	**1.65**	1.67	**1.65**	2.47	2.39
GHS_indef/d_pretok (1)	1.61	1.62	**1.59**	2.43	2.44
TSOPF/TSOPF_FS_b300_c3 (1)	6.16	5.96	6.43	(7) **4.55**	6.52
GHS_indef/dtoc (1)	**0.05**	0.06	0.06	0.07	1.34
GHS_indef/aug2d (1)	**0.01**	0.02	0.02	0.85	0.24
GHS_indef/aug3d (1)	**0.01**	0.03	0.02	0.79	0.44

A.6 Factorise phase comparison

A.6.1 Test Set 1: Small problems, Serial

Times for 100 factors, in seconds. Fastest (successful) time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Cunningham/m3plates	0.36	0.89	**0.26**	(1) -	>3000
Boeing/bcsstm37	0.85	2.01	**0.54**	(7) 0.27	2.14
GHS_indef/qpband	0.54	1.14	**0.51**	0.54	0.97
HB/zenios	0.96	0.35	**0.15**	(7) 0.18	2.74
HB/saylr3	0.10	0.18	0.09	0.10	**0.08**
HB/sherman1	0.10	0.17	**0.08**	0.10	**0.08**
Oberwolfach/filter2D	0.21	0.27	0.16	0.22	**0.14**
RAL/a2ensndl-00	1.67	3.30	1.72	**0.94**	(3) 1.42
RAL/a2ensndl-49	2.56	11.16	8.10	(7) 0.95	**1.66**
RAL/a2ensndl-62	2.57	4.60	2.94	**0.96**	(3) 1.62
Schenk_IBMNA/c-29 (1)	1.04	1.47	0.93	**0.42**	0.53
Boeing/nasa1824	0.54	0.61	0.47	0.56	**0.39**
GHS_indef/spmsrtls	2.38	3.22	2.04	2.00	**1.26**
IPSO/OPF_3754 (1)	1.65	2.23	1.50	(7) 1.05	**0.57**
GHS_indef/stokes64 (1)	4.08	5.51	4.06	**3.12**	5.33
GHS_indef/brainpc2 (1)	3.68	11.34	2.74	2.02	**1.19**
TSOPF/TSOPF_FS_b9_c6 (1)	4.41	7.26	3.14	(7) 1.49	**1.31**
Nemeth/nemeth05	2.07	2.93	1.98	4.39	**1.87**
Nemeth/nemeth10	2.12	2.77	1.90	3.77	**1.76**
FIDAP/ex14 (1)	3.25	3.54	3.44	0.70	**0.66**
Nemeth/nemeth15	2.98	3.85	2.86	5.44	**1.23**
GHS_indef/ncvxqp9 (1)	6.52	13.02	6.40	**1.48**	1.96
Schenk_IBMNA/c-41 (1)	4.40	5.25	4.07	**1.49**	1.89
GHS_indef/tuma2 (1)	2.01	2.70	1.86	**1.37**	1.83
Marini/eurqsa	**1.98**	2.80	2.01	(7) 1.59	4.02
Oberwolfach/rail_20209	2.79	4.07	**2.31**	2.50	2.55
Newman/hep-th (1)	15.28	6.42	6.36	**2.02**	(7) 21.52
Nemeth/nemeth20	15.58	9.01	16.74	9.07	**2.67**
PARSEC/Si2	1.55	1.75	1.49	1.69	**1.15**
Oberwolfach/t2dah_a	3.59	3.82	2.95	(7) 3.00	**2.40**
IPSO/TSC_OPF_300 (1)	**5.99**	7.21	6.71	8.79	6.45
Nemeth/nemeth25 (1)	**9.57**	13.26	11.09	15.87	12.83
Schenk_IBMNA/c-50 (1)	6.16	8.94	5.63	**3.99**	6.19
Newman/cond-mat (1)	62.58	36.67	40.29	**11.87**	(4,7) 86.15
Boeing/crystk01	10.20	9.62	8.95	(7) 9.05	**7.01**
TSOPF/TSOPF_FS_b39_c7 (1)	44.94	47.68	44.15	(7) 8.02	**9.44**
Boeing/bcsstk37 (1)	23.93	24.10	**21.56**	24.47	21.83
GHS_indef/exdata_1	30.57	31.40	26.73	**21.57**	56.91
TSOPF/TSOPF_FS_b162_c1 (1)	149.43	131.35	133.41	(7) 9.71	**14.20**
Boeing/crystk02	49.30	47.62	**43.68**	(7) 49.70	45.34

A.6.2 Test Set 1: Small problems, 8 cores

Times for 100 factors, in seconds. Fastest (successful) time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Cunningham/m3plates	0.36	0.84	**0.26**	(2) -	>3000
Boeing/bcsstm37	0.85	1.84	**0.54**	(7) 1.29	55.36
GHS_indef/qpband	0.54	0.85	**0.51**	1.47	1.63
HB/zenios	0.98	0.26	**0.15**	(7) 0.16	1.76
HB/saylr3	0.11	0.10	0.09	0.10	**0.06**
HB/sherman1	0.10	0.10	0.08	0.10	**0.06**
Oberwolfach/filter2D	0.21	0.13	0.16	0.17	**0.08**
RAL/a2ensndl-00	1.66	1.73	1.72	3.08	(3) **1.13**
RAL/a2ensndl-49 (1)	2.55	9.52	8.10	(7) 3.23	**1.07**
RAL/a2ensndl-62	2.56	2.90	2.94	3.17	(3) **1.12**
Schenk_IBMNA/c-29 (1)	1.04	0.94	0.93	0.48	**0.17**
Boeing/nasa1824	0.55	0.34	0.47	**0.22**	**0.22**
GHS_indef/spmsrtls	2.36	3.63	2.04	1.64	**0.36**
IPSO/OPF_3754 (1)	1.65	1.17	1.50	(7) 1.08	**0.29**
GHS_indef/stokes64 (1)	4.13	3.40	2.56	**1.16**	1.50
GHS_indef/brainpc2 (1)	3.67	11.97	2.74	2.56	**0.43**
TSOPF/TSOPF_FS_b9_c6 (1)	4.41	5.11	1.50	(7) 1.27	**0.37**
Nemeth/nemeth05	2.06	3.27	1.98	1.38	**0.59**
Nemeth/nemeth10	2.11	2.95	1.91	1.12	**0.58**
FIDAP/ex14 (1)	3.26	3.03	3.45	0.26	**0.23**
Nemeth/nemeth15	2.97	4.09	2.87	1.21	**0.87**
GHS_indef/ncvxqp9 (1)	6.55	10.93	6.40	1.62	**0.65**
Schenk_IBMNA/c-41	4.39	3.61	4.06	1.09	**0.62**
GHS_indef/tuma2 (1)	2.02	1.39	1.86	1.31	**0.65**
Marini/eurqsa	2.00	**1.35**	2.01	(7) 1.12	1.70
Oberwolfach/rail_20209	2.81	1.43	1.66	1.87	**0.70**
Newman/hep-th (1)	12.80	5.54	4.64	**1.17**	(7) 16.20
Nemeth/nemeth20	15.57	8.40	17.92	**1.49**	1.79
PARSEC/Si2	1.49	0.96	0.97	0.69	**0.56**
Oberwolfach/t2dah_a	3.60	1.49	2.15	(7) 0.90	**0.71**
IPSO/TSC_OPF_300 (1)	6.01	7.58	7.22	**1.95**	2.01
Nemeth/nemeth25 (1)	9.54	9.98	12.44	**2.44**	3.20
Schenk_IBMNA/c-50 (1)	6.02	4.56	3.89	2.66	**1.42**
Newman/cond-mat (1)	57.54	29.61	26.35	**3.91**	(4,7) 41.69
Boeing/crystk01	8.65	4.75	4.11	(7) 1.75	**1.92**
TSOPF/TSOPF_FS_b39_c7 (1)	44.19	27.68	41.47	(7) 3.26	**2.45**
Boeing/bcsstk37 (1)	21.93	8.76	9.18	**4.36**	7.07
GHS_indef/exdata_1	22.05	23.20	19.63	**6.93**	10.15
TSOPF/TSOPF_FS_b162_c1 (1)	113.68	110.19	101.36	(7) 2.38	**3.36**
Boeing/crystk02	34.91	12.44	11.61	(7) 7.55	**10.37**

A.6.3 Test Set 2: Positive-definite problems, Serial

Times for 1 factorization, in seconds. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Mulvey/finan512	0.13	0.19	0.13	0.14	**0.12**
MaxPlanck/shallow_water1	0.16	0.18	**0.14**	0.16	0.15
UTEP/Dubcova3	0.55	0.53	**0.45**	**0.45**	0.53
Nasa/nasasrb	0.98	0.92	0.97	0.77	**0.54**
CEMW/tmt_sym	2.92	2.72	**2.40**	3.04	2.72
Schmid/thermal2	4.74	4.47	3.90	4.79	**3.86**
Rothberg/gearbox	4.90	3.69	3.89	3.61	**3.35**
INPRO/msdoor	5.07	4.19	4.12	3.94	**3.76**
DNVS/m_t1	4.91	3.43	3.64	3.67	**2.70**
McRae/ecology2	4.38	5.05	4.41	3.84	**3.50**
Boeing/pwtk	5.39	4.16	4.29	4.16	**3.61**
Chen/pkustk13	5.43	4.01	4.35	3.90	**2.85**
BenElechi/BenElechi1	6.34	4.79	4.99	4.60	**4.06**
Rothberg/cfd2	6.79	5.03	5.07	5.37	**4.28**
DNVS/thread	6.71	4.77	5.31	4.92	**4.23**
DNVS/shipsec8	8.11	6.05	6.29	6.47	**5.79**
DNVS/shipsec1	7.79	5.81	6.22	**5.75**	6.52
GHS_psdef/crankseg_2	9.58	6.88	7.56	7.16	**5.24**
DNVS/fcondp2	10.05	7.58	8.06	**8.04**	8.18
Schenk_AFE/af_shell3	11.86	9.31	9.59	9.04	**7.81**
DNVS/troll	11.53	8.65	9.25	9.08	**7.34**
GHS_psdef/bmwcra_1	12.78	9.40	10.10	10.10	**7.41**
DNVS/halfb	14.59	10.58	11.55	11.57	**10.99**
GHS_psdef/crankseg_1	6.74	9.75	10.79	4.99	**3.82**
Um/2cubes_sphere	14.71	10.77	12.10	10.99	**6.28**
GHS_psdef/ldoor	18.68	14.73	15.00	**13.73**	15.78
DNVS/ship_003	17.08	12.07	13.28	12.44	**10.55**
DNVS/fullb	20.79	15.02	16.55	**14.46**	16.07
Um/offshore	21.54	16.01	17.60	16.36	**10.93**
GHS_psdef/inline_1	30.56	24.51	23.71	23.58	**16.99**
Chen/pkustk14	30.47	21.96	24.91	22.21	**15.36**
GHS_psdef/apache2	35.26	**26.16**	28.57	28.86	29.10
Koutsovasilis/F1	44.23	32.33	35.67	36.62	**23.93**
Oberwolfach/boneS10	57.33	42.58	46.04	41.75	**39.86**
AMD/G3_circuit	14.28	43.12	45.41	13.13	**11.90**
ND/nd12k	109.46	73.09	101.02	82.97	**58.00**
JGD_Trefethen/Trefethen_20000	139.33	97.42	122.18	193.51	(4) **82.96**
ND/nd24k	424.67	292.39	388.27	348.35	**277.21**
Oberwolfach/bone010	681.74	492.68	552.74	499.50	**491.05**
GHS_psdef/audikw_1	1020.08	740.60	835.43	771.56	**518.80**

A.6.4 Test Set 2: Positive-definite problems, 8 cores

Times for 1 factorization, in seconds. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Mulvey/finan512	0.15	0.05	0.05	0.07	**0.03**
MaxPlanck/shallow_water1	0.15	0.06	0.05	0.09	**0.04**
UTEP/Dubcova3	0.50	0.12	0.12	**0.10**	0.13
Nasa/nasasrb	0.78	0.29	0.33	**0.14**	0.15
CEMW/tmt_sym	2.32	**0.54**	0.52	0.95	0.59
Schmid/thermal2	3.78	**0.87**	0.88	1.53	0.95
Rothberg/gearbox	3.24	0.67	0.80	**0.53**	0.75
INPRO/msdoor	3.89	0.75	0.87	**0.60**	0.79
DNVS/m_t1	3.02	0.66	0.71	**0.57**	0.64
McRae/ecology2	3.20	0.92	1.30	1.30	**0.88**
Boeing/pwtk	3.73	0.71	0.80	**0.66**	0.84
Chen/pkustk13	2.96	0.72	1.03	**0.73**	0.74
BenElechi/BenElechi1	4.38	0.77	0.97	**0.75**	0.95
Rothberg/cfd2	3.77	**0.83**	0.99	1.01	1.02
DNVS/thread	2.94	0.86	1.37	**0.75**	1.17
DNVS/shipsec8	4.46	1.01	1.45	**0.99**	1.05
DNVS/shipsec1	4.13	0.93	1.21	**0.86**	1.15
GHS_psdef/crankseg_2	4.90	1.13	1.44	**1.07**	1.30
DNVS/fcondp2	5.21	**1.21**	1.51	1.36	1.65
Schenk_AFE/af_shell3	7.35	1.51	1.70	**1.45**	1.68
DNVS/troll	6.28	1.40	1.71	1.58	**1.28**
GHS_psdef/bmwcra_1	7.21	1.55	1.83	1.55	**1.37**
DNVS/halfb	7.69	**1.67**	2.74	2.08	2.46
GHS_psdef/crankseg_1	3.63	1.52	2.69	**0.88**	0.99
Um/2cubes_sphere	7.09	1.75	2.31	1.77	**1.48**
GHS_psdef/ldoor	11.89	2.37	2.63	**2.04**	2.38
DNVS/ship_003	8.79	1.95	2.71	**1.98**	2.16
DNVS/fullb	10.94	**2.32**	3.48	2.48	2.86
Um/offshore	11.03	**2.42**	3.32	2.53	2.17
GHS_psdef/inline_1	16.15	3.40	4.13	3.34	**3.26**
Chen/pkustk14	14.85	3.45	4.32	**3.29**	3.46
GHS_psdef/apache2	17.18	**3.86**	4.97	4.65	4.32
Koutsovasilis/F1	20.61	4.67	6.12	5.25	**4.62**
Oberwolfach/boneS10	27.26	6.12	8.18	**5.98**	7.09
AMD/G3_circuit	8.97	6.33	7.92	3.08	**2.22**
ND/nd12k	49.28	10.07	19.69	14.59	**9.81**
JGD_Trefethen/Trefethen_20000	51.74	**13.39**	33.86	44.45	(4) 13.65
ND/nd24k	175.24	39.89	73.21	63.83	**34.60**
Oberwolfach/bone010	208.97	**64.67**	83.70	71.37	73.81
GHS_psdef/audikw_1	301.05	97.05	124.53	113.28	**90.19**

A.6.5 Test Set 3: General indefinite problems, Serial

Times for 1 factorization, in seconds. Fastest (successful) time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Oberwolfach/t2dal	**0.01**	**0.01**	**0.01**	(5) **0.01**	**0.01**
GHS_indef/dixmaanl	0.04	0.35	0.04	0.04	**0.03**
Oberwolfach/rail_79841	0.16	0.21	0.14	**0.13**	0.15
GHS_indef/dawson5	0.52	0.54	0.46	0.34	**0.31**
Boeing/bcsstk39	0.63	0.59	**0.54**	0.72	0.56
GHS_indef/helm2d03	1.77	2.08	**1.57**	1.95	1.78
GHS_indef/copter2	1.49	1.53	1.32	1.49	**1.26**
Boeing/crystk03	1.35	1.28	1.15	(7) 1.34	**1.11**
Oberwolfach/filter3D	2.08	2.19	**1.88**	2.32	1.97
Boeing/pct20stif	3.14	2.91	2.79	1.26	**1.13**
Koutsovasilis/F2	3.05	2.75	2.42	2.91	**1.98**
Cunningham/qa8fk	4.46	4.06	**3.67**	(5) 3.61	3.75
Oberwolfach/gas_sensor	4.57	4.16	3.80	**3.69**	4.01
McRae/ecology1	6.29	6.23	5.01	(7) 4.92	**4.56**
Oberwolfach/t3dh	13.20	11.48	**10.63**	(7) 11.05	12.05
Lin/Lin	56.02	45.88	46.00	42.52	**42.45**
GHS_indef/sparsine	370.29	283.16	342.98	374.13	**172.51**
PARSEC/Ge99H100	1488.83	1163.66	1328.82	1078.56	**613.70**
PARSEC/Ga10As10H30	1458.15	1168.15	1295.37	959.75	**667.78**
PARSEC/Ga19As19H42	1893.60	1488.89	1685.49	1318.50	**859.24**

A.6.6 Test Set 3: General indefinite problems, 8 cores

Times for 1 factorization, in seconds. Fastest (successful) time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Oberwolfach/t2dal	0.01	0.01	0.01	(5) **0.00**	**0.00**
GHS_indef/dixmaanl	0.04	0.37	0.04	0.06	**0.01**
Oberwolfach/rail_79841	0.17	0.10	0.10	0.08	**0.04**
GHS_indef/dawson5	0.47	0.25	0.21	**0.08**	0.15
Boeing/bcsstk39	0.57	0.25	0.32	**0.11**	0.17
GHS_indef/helm2d03	1.48	0.57	0.45	0.54	**0.45**
GHS_indef/copter2	1.05	0.43	0.39	**0.31**	0.37
Boeing/crystk03	0.91	0.36	0.34	(7) 0.22	**0.27**
Oberwolfach/filter3D	1.58	0.49	0.48	**0.39**	0.44
Boeing/pct20stif	2.39	1.58	1.67	**0.20**	0.33
Koutsovasilis/F2	2.18	0.86	1.23	**0.42**	0.49
Cunningham/qa8fk	2.61	**0.78**	0.87	(5) 0.65	0.88
Oberwolfach/gas_sensor	2.73	0.89	0.93	**0.71**	**0.71**
McRae/ecology1	4.36	1.51	1.84	(7) 1.51	**1.04**
Oberwolfach/t3dh	6.03	2.21	2.41	(7) 1.88	**2.13**
Lin/Lin	26.14	7.18	8.71	**6.87**	7.14
GHS_indef/sparsine	201.52	46.90	85.28	81.89	**29.82**
PARSEC/Ge99H100	624.83	176.76	294.02	215.51	**97.67**
PARSEC/Ga10As10H30	568.62	178.35	269.74	189.44	**102.95**
PARSEC/Ga19As19H42	778.60	221.92	321.20	273.75	**131.94**

A.6.7 Test Set 4: KKT problems, Serial

Times for 1 factorization, in seconds. Fastest (successful) time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
GHS_indef/boyd1	0.46	3.65	0.74	**0.05**	0.11
GHS_indef/bmw3_2 (1)	7.25	6.21	**6.24**	6.49	6.38
GHS_indef/c-72 (1)	1.01	0.75	0.59	**0.46**	0.94
GHS_indef/ncvxqp7 (1)	77.91	31.95	33.38	**10.08**	14.74
Andrianov/mip1	27.73	29.09	29.52	9.24	**1.42**
GHS_indef/blockqp1	0.07	0.12	0.08	**0.04**	**0.04**
GHS_indef/boyd2	0.24	0.46	0.27	**0.16**	0.34
GHS_indef/a5esindl	0.04	0.07	0.04	**0.02**	0.03
GHS_indef/a2nnsnsl	0.05	0.11	0.05	**0.03**	0.05
GHS_indef/a0nsdsil	0.05	0.11	0.05	**0.03**	0.05
TSOPF/TSOPF_FS_b39_c30 (1)	1.95	1.59	1.36	(7) 0.38	**0.48**
GHS_indef/cont-201 (1)	2.86	1.21	1.04	**0.35**	0.73
GHS_indef/darcy003 (1)	0.86	1.19	0.82	**0.61**	0.78
GHS_indef/cont-300 (1)	6.45	4.53	3.78	(7) 1.05	**2.19**
GHS_indef/turon_m (1)	1.42	1.54	**1.29**	1.33	2.26
GHS_indef/d_pretok (1)	1.58	1.66	**1.41**	1.46	2.59
TSOPF/TSOPF_FS_b300_c3 (1)	18.80	10.86	10.51	(7) 5.32	**6.31**
GHS_indef/dtoc (1)	1.91	1.25	0.18	**0.01**	0.02
GHS_indef/aug2d (1)	33.41	0.75	0.83	**0.02**	0.04
GHS_indef/aug3d (1)	187.11	14.01	17.07	**0.09**	0.17

A.6.8 Test Set 4: KKT problems, 8 cores

Times for 1 factorization, in seconds. Fastest (successful) time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
GHS_indef/boyd1	0.46	3.47	0.74	0.08	**0.05**
GHS_indef/bmw3_2 (1)	4.58	1.50	1.59	**1.00**	1.60
GHS_indef/c-72 (1)	0.88	0.29	0.26	**0.17**	0.19
GHS_indef/ncvxqp7 (1)	63.72	17.13	21.14	**2.26**	3.86
Andrianov/mip1	15.96	12.44	15.05	1.86	**0.44**
GHS_indef/blockqp1	0.07	0.08	0.08	0.06	**0.02**
GHS_indef/boyd2	0.24	0.28	0.27	0.45	**0.16**
GHS_indef/a5esindl	0.04	0.04	0.04	0.05	**0.02**
GHS_indef/a2nnsnsl	0.05	0.05	0.05	0.08	**0.02**
GHS_indef/a0nsdsil	0.05	0.05	0.05	0.08	**0.02**
TSOPF/TSOPF_FS_b39_c30 (1)	1.96	1.42	0.71	(7) 0.16	**0.16**
GHS_indef/cont-201 (1)	2.57	0.47	0.33	**0.12**	0.19
GHS_indef/darcy003 (1)	0.85	0.68	0.45	0.39	**0.22**
GHS_indef/cont-300 (1)	5.35	1.78	1.23	(7) 0.29	**0.52**
GHS_indef/turon_m (1)	1.17	0.54	0.43	**0.32**	0.50
GHS_indef/d_pretok (1)	1.24	0.53	0.44	**0.33**	0.54
TSOPF/TSOPF_FS_b300_c3 (1)	8.21	3.96	3.64	(7) 1.03	**1.81**
GHS_indef/dtoc (1)	1.83	1.18	0.18	0.02	**0.01**
GHS_indef/aug2d (1)	31.56	0.51	0.83	0.03	**0.01**
GHS_indef/aug3d (1)	172.30	12.23	6.99	**0.04**	0.05

A.7 Solve phase comparisons

A.7.1 Test Set 1: Small problems

Times for 10 sequential solves 8 threads. For `HSL_MA97` the faster of the Multifrontal or Supernodal solve variants is used, but automatic use of serial solve for small problems is disabled. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Cunningham/m3plates	**0.00**	0.07	0.01	(2) -	1.04
Boeing/bcsstm37	**0.01**	0.16	**0.01**	(7) 0.07	0.12
GHS_indef/qpband	**0.01**	0.06	**0.01**	0.04	0.05
HB/zenios	**0.00**	0.02	**0.00**	(7) 0.02	0.01
HB/saylr3	**0.00**	**0.00**	**0.00**	0.01	**0.00**
HB/sherman1	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Oberwolfach/filter2D	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
RAL/a2ensndl-00	**0.02**	0.13	0.03	0.24	**(3) 0.02**
RAL/a2ensndl-49 (1)	**0.02**	0.14	0.04	(7) 0.24	0.03
RAL/a2ensndl-62	**0.02**	0.13	0.03	0.24	(3) 0.03
Schenk_IBMNA/c-29	**0.00**	0.02	0.01	**0.00**	0.01
Boeing/nasa1824	**0.00**	**0.00**	**0.00**	0.01	**0.00**
GHS_indef/spmsrtls	**0.02**	0.09	0.03	**0.02**	**0.02**
IPSO/OPF_3754	**0.01**	0.02	0.02	(7) 0.04	0.02
GHS_indef/stokes64	**0.01**	0.02	**0.01**	0.04	**0.01**
GHS_indef/brainpc2 (1)	**0.02**	0.11	0.03	0.20	**0.02**
TSOPF/TSOPF_FS_b9_c6 (1)	**0.01**	0.03	**0.01**	(7) 0.12	**0.01**
Nemeth/nemeth05	**0.01**	0.04	**0.01**	**0.01**	**0.01**
Nemeth/nemeth10	**0.01**	0.03	**0.01**	**0.01**	**0.01**
FIDAP/ex14 (1)	**0.00**	**0.00**	**0.00**	0.01	**0.00**
Nemeth/nemeth15	**0.01**	0.04	**0.01**	**0.01**	**0.01**
GHS_indef/ncvxqp9 (1)	**0.01**	0.04	0.02	0.04	**0.01**
Schenk_IBMNA/c-41	**0.01**	0.03	**0.01**	**0.01**	**0.01**
GHS_indef/tuma2 (1)	**0.01**	0.02	0.02	0.03	**0.01**
Marini/eurqsa	**0.01**	**0.01**	**0.01**	(7) 0.03	**0.01**
Oberwolfach/rail_20209	0.02	0.03	0.02	**0.01**	0.02
Newman/hep-th (1)	**0.01**	0.02	**0.01**	0.03	(7) 0.02
Nemeth/nemeth20	**0.01**	0.04	0.02	**0.01**	0.02
PARSEC/Si2	**0.00**	**0.00**	**0.00**	0.01	**0.00**
Oberwolfach/t2dah_a	**0.01**	**0.01**	**0.01**	(7) 0.03	**0.01**
IPSO/TSC_OPF_300 (1)	**0.01**	0.03	0.02	0.05	0.02
Nemeth/nemeth25 (1)	**0.02**	0.05	0.02	**0.02**	0.03
Schenk_IBMNA/c-50 (1)	**0.02**	0.08	**0.02**	**0.02**	0.03
Newman/cond-mat (1)	0.03	0.06	0.04	0.06	(4,7) 0.05
Boeing/crystk01	0.02	**0.01**	**0.01**	(7) 0.03	**0.01**
TSOPF/TSOPF_FS_b39_c7 (1)	0.09	0.08	0.10	(7) 0.32	**0.04**
Boeing/bcsstk37 (1)	0.07	0.05	0.05	**0.04**	0.06
GHS_indef/exdata_1	**0.02**	0.05	**0.02**	**0.02**	0.05
TSOPF/TSOPF_FS_b162_c1 (1)	0.08	0.12	0.09	(7) 0.05	**0.02**
Boeing/crystk02	0.07	**0.05**	**0.05**	(7) 0.16	0.06

A.7.2 Test Set 2: Positive-definite problems

Times for 10 sequential solves 8 threads. For `HSL_MA97` the faster of the Multifrontal or Supernodal solve variants is used, but automatic use of serial solve for small problems is disabled. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Mulvey/finan512	0.09	0.12	0.13	**0.03**	0.08
MaxPlanck/shallow_water1	0.10	0.13	0.10	**0.03**	0.08
UTEP/Dubcova3	0.25	0.25	0.15	**0.11**	0.19
Nasa/nasasrb	0.23	0.19	0.17	**0.12**	0.14
CEMW/tmt_sym	1.27	1.21	0.71	**0.45**	0.88
Schmid/thermal2	2.27	2.06	1.16	**0.72**	1.53
Rothberg/gearbox	0.76	0.47	**0.42**	0.45	0.65
INPRO/msdoor	1.37	0.92	0.69	**0.60**	0.92
DNVS/m_t1	0.62	0.36	**0.33**	0.37	0.40
McRae/ecology2	1.55	1.68	0.88	**0.53**	1.27
Boeing/pwtk	1.01	0.62	0.55	**0.50**	0.62
Chen/pkustk13	0.58	0.37	0.36	0.35	**0.34**
BenElechi/BenElechi1	1.11	0.64	0.65	**0.58**	0.67
Rothberg/cfd2	0.69	0.45	0.42	0.44	**0.39**
DNVS/thread	0.35	**0.22**	0.35	0.36	0.28
DNVS/shipsec8	0.65	0.43	0.48	0.50	**0.42**
DNVS/shipsec1	0.72	0.47	**0.42**	0.45	0.49
GHS_psdef/crankseg_2	0.68	**0.42**	0.43	0.51	0.50
DNVS/fcondp2	1.00	0.63	**0.62**	0.72	0.67
Schenk_AFE/af_shell3	2.05	1.30	0.94	**0.91**	1.20
DNVS/troll	1.20	0.74	**0.62**	0.71	0.71
GHS_psdef/bmwcra_1	1.16	0.69	**0.62**	0.73	0.66
DNVS/halfb	1.17	0.76	**0.73**	0.77	0.83
GHS_psdef/crankseg_1	0.52	0.42	0.63	0.43	**0.38**
Um/2cubes_sphere	0.76	0.47	0.49	0.63	**0.39**
GHS_psdef/ldoor	3.42	2.33	1.67	**1.45**	2.14
DNVS/ship_003	0.97	**0.61**	**0.61**	0.68	0.62
DNVS/fullb	1.26	**0.80**	**0.80**	0.82	0.86
Um/offshore	1.58	0.99	0.87	0.96	**0.75**
GHS_psdef/inline_1	3.18	1.90	1.67	**1.65**	1.85
Chen/pkustk14	1.57	1.00	1.10	1.03	**0.99**
GHS_psdef/apache2	2.77	2.14	1.70	**1.50**	1.65
Koutsovasilis/F1	2.86	1.74	**1.63**	1.96	1.65
Oberwolfach/boneS10	5.20	3.32	**2.93**	3.32	3.43
AMD/G3_circuit	3.24	4.04	2.63	**1.25**	2.00
ND/nd12k	1.44	**0.96**	1.51	1.90	1.06
JGD_Trefethen/Trefethen_20000	1.55	0.80	1.65	3.27	**0.78**
ND/nd24k	3.97	**2.53**	3.89	5.22	2.81
Oberwolfach/bone010	14.18	**9.03**	9.59	12.16	9.11
GHS_psdef/audikw_1	15.87	10.25	11.40	12.59	**9.86**

A.7.3 Test Set 3: General indefinite problems

Times for 10 sequential solves 8 threads. For `HSL_MA97` the faster of the Multifrontal or Supernodal solve variants is used, but automatic use of serial solve for small problems is disabled. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Oberwolfach/t2dal	**0.00**	0.01	0.01	(5) -	0.01
GHS_indef/dixmaanl	**0.04**	0.32	0.06	**0.04**	0.05
Oberwolfach/rail_79841	0.09	0.14	0.09	**0.05**	0.10
GHS_indef/dawson5	0.12	0.08	**0.07**	0.25	0.17
Boeing/bcsstk39	0.16	0.10	0.11	**0.09**	0.12
GHS_indef/helm2d03	0.73	0.72	0.41	**0.33**	0.52
GHS_indef/copter2	0.21	0.17	0.16	**0.13**	0.15
Boeing/crystk03	0.16	**0.11**	**0.11**	(7) 0.33	0.12
Oberwolfach/filter3D	0.45	0.27	0.23	**0.21**	0.29
Boeing/pct20stif	0.24	0.18	0.17	**0.12**	0.16
Koutsovasilis/F2	0.42	0.27	0.28	**0.21**	0.27
Cunningham/qa8fk	0.41	**0.27**	**0.27**	(5) -	0.26
Oberwolfach/gas_sensor	0.42	0.27	0.28	**0.23**	0.26
McRae/ecology1	1.71	1.96	**0.95**	(7) 2.57	1.35
Oberwolfach/t3dh	0.73	0.52	0.54	(7) 1.42	**0.48**
Lin/Lin	1.73	1.49	1.31	**1.01**	1.12
GHS_indef/sparsine	3.03	3.21	3.33	2.54	**1.63**
PARSEC/Ge99H100	8.60	8.61	8.29	6.54	**4.46**
PARSEC/Ga10As10H30	8.33	8.63	9.10	5.97	**4.51**
PARSEC/Ga19As19H42	10.49	9.97	10.12	8.01	**5.48**

A.7.4 Test Set 4: KKT problems

Times for 10 sequential solves 8 threads. For `HSL_MA97` the faster of the Multifrontal or Supernodal solve variants is used, but automatic use of serial solve for small problems is disabled. Fastest time in bold.

Problem	MA57	MA86	MA97	PARDISO	WSMP
GHS_indef/boyd1	**0.08**	1.97	0.12	0.24	0.12
GHS_indef/bmw3_2 (1)	0.99	0.59	0.66	**0.47**	0.68
GHS_indef/c-72 (1)	0.14	0.32	0.14	**0.09**	0.14
GHS_indef/ncvxqp7 (1)	0.76	0.55	1.15	0.82	**0.45**
Andrianov/mip1	0.61	0.78	0.83	0.80	**0.28**
GHS_indef/blockqp1	**0.04**	0.42	0.05	0.47	0.06
GHS_indef/boyd2	0.24	2.07	**0.23**	1.06	0.38
GHS_indef/a5esindl	**0.03**	0.16	0.04	0.13	0.05
GHS_indef/a2nnsnsl	0.06	0.17	0.07	**0.04**	0.08
GHS_indef/a0nsdsil	**0.05**	0.17	0.06	**0.05**	0.08
TSOPF/TSOPF_FS_b39_c30 (1)	0.28	0.27	0.22	(7) 1.58	**0.20**
GHS_indef/cont-201 (1)	0.25	0.19	0.16	0.28	**0.14**
GHS_indef/darcy003 (1)	0.48	0.71	**0.32**	0.89	0.47
GHS_indef/cont-300 (1)	0.52	0.49	0.45	(7) 0.57	**0.33**
GHS_indef/turon_m (1)	0.41	0.38	**0.24**	0.63	0.36
GHS_indef/d_pretok (1)	0.42	0.37	**0.25**	0.63	0.36
TSOPF/TSOPF_FS_b300_c3 (1)	0.71	**0.45**	0.47	(7) 3.52	0.46
GHS_indef/dtoc (1)	**0.02**	0.17	0.05	0.05	**0.02**
GHS_indef/aug2d (1)	0.11	0.12	0.14	0.06	**0.03**
GHS_indef/aug3d (1)	0.26	0.61	0.60	0.07	**0.04**

A.8 Forward error comparisons

A.8.1 Test Set 1: Small problems

All results are determined using a serial run of the codes. Numbers in italics represent forward errors more than four orders of magnitude greater than best solve.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Cunningham/m3plates	1.00e+00	1.00e+00	1.00e+00	(2) -	1.00e+00
Boeing/bcsstm37	1.00e+00	1.00e+00	1.00e+00	(7) 2.76e+01	1.00e+00
GHS_indef/qpband	2.22e-16	4.44e-16	1.78e-15	1.78e-15	2.22e-16
HB/zenios	4.40e+00	1.00e+00	1.00e+00	(7) 1.07e+00	1.00e+00
HB/saylr3	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00
HB/sherman1	2.85e-13	9.73e-14	8.16e-14	4.35e-13	2.42e-13
Oberwolfach/filter2D	4.56e-14	5.60e-14	2.04e-14	4.26e-14	5.51e-14
RAL/a2ensndl-00	1.47e-14	4.22e-15	1.31e-14	1.33e-15	(3) 4.00e-15
RAL/a2ensndl-49 (1)	2.76e+05	5.72e+04	9.54e+04	(7) *3.36e+23*	9.54e+04
RAL/a2ensndl-62	*2.57e+05*	*6.64e+04*	*1.14e+05*	1.08e+00	(3) *5.72e+05*
Schenk_IBMNA/c-29 (1)	6.64e-12	5.54e-12	1.95e-12	*9.87e-06*	6.97e-11
Boeing/nasa1824	1.77e-12	6.32e-12	2.35e-12	*4.58e-05*	5.84e-12
GHS_indef/spmsrtls	3.20e-12	9.46e-13	1.17e-12	3.99e-11	3.44e-12
IPSO/OPF_3754 (1)	7.12e-10	2.18e-09	8.08e-10	(7) *4.04e+06*	5.60e-10
GHS_indef/stokes64 (1)	2.19e+00	7.52e-01	7.57e+01	1.00e+00	1.84e+00
GHS_indef/brainpc2 (1)	9.55e-13	2.86e-12	1.61e-12	*3.07e-03*	3.11e-11
TSOPF/TSOPF_FS_b9_c6 (1)	2.90e-07	1.68e-08	1.90e-08	(7) *4.00e+02*	2.77e-08
Nemeth/nemeth05	1.33e-15	1.33e-15	1.33e-15	2.33e-15	1.33e-15
Nemeth/nemeth10	2.00e-15	1.33e-15	2.22e-15	4.00e-15	1.78e-15
FIDAP/ex14 (1)	5.29e-06	3.31e-06	3.66e-06	1.03e-06	9.49e-06
Nemeth/nemeth15	4.56e-14	3.69e-14	3.87e-14	3.56e-13	3.22e-14
GHS_indef/ncvxqp9 (1)	4.53e-08	1.32e-07	5.99e-08	*7.54e-04*	2.51e-07
Schenk_IBMNA/c-41	9.93e-07	7.79e-07	3.06e-07	*1.64e-03*	1.18e-06
GHS_indef/tuma2 (1)	9.38e-13	4.18e-13	6.63e-13	7.99e-15	1.85e-12
Marini/eurqsa	4.06e+06	2.18e+04	2.09e+05	(7) *1.69e+39*	3.20e+06
Oberwolfach/rail_20209	1.51e-12	8.99e-13	9.75e-13	4.45e-13	1.15e-12
Newman/hep-th (1)	*1.87e+05*	*3.04e+05*	*1.13e+05*	1.46e+00	(7) *1.41e+18*
Nemeth/nemeth20	8.66e-14	6.68e-14	5.78e-14	1.95e-13	4.31e-13
PARSEC/Si2	5.65e-14	2.15e-14	4.86e-14	1.11e-14	2.81e-14
Oberwolfach/t2dah_a	1.32e-03	2.57e-04	6.20e-04	(7) *9.31e-01*	9.30e-03
IPSO/TSC_OPF_300 (1)	1.27e-07	5.11e-08	8.61e-10	*3.63e-05*	1.25e-09
Nemeth/nemeth25 (1)	1.52e-13	8.08e-14	1.04e-13	2.42e-13	1.01e-13
Schenk_IBMNA/c-50 (1)	3.28e-12	2.43e-12	9.88e-12	*1.73e-05*	1.68e-11
Newman/cond-mat (1)	*9.36e+05*	*6.70e+05*	4.80e+04	2.26e+01	(4,7) *2.02e+19*
Boeing/crystk01	1.55e-02	7.10e-03	1.05e-02	(7) 1.00e+00	1.54e-02
TSOPF/TSOPF_FS_b39_c7 (1)	7.72e-07	3.36e-07	5.60e-07	(7) *1.50e+08*	4.84e-07
Boeing/bcsstk37 (1)	3.68e-05	5.15e-05	4.50e-05	1.43e-05	2.45e-06
GHS_indef/exdata_1	5.64e-10	4.37e-10	3.74e-10	1.04e-09	8.58e-10
TSOPF/TSOPF_FS_b162_c1 (1)	2.99e-05	1.22e-06	1.22e-06	(7) *1.42e+06*	2.01e-05
Boeing/crystk02	8.68e-02	5.94e-02	6.99e-02	(7) 1.00e+00	7.52e-02

A.8.2 Test Set 2: Positive-definite problems

All results are determined using a serial run of the codes.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Mulvey/finan512	5.11e-15	2.44e-15	2.89e-15	4.00e-15	3.11e-15
MaxPlanck/shallow_water1	1.78e-15	1.11e-15	1.33e-15	2.33e-15	9.99e-16
UTEP/Dubcova3	6.39e-14	6.51e-14	3.15e-14	7.73e-14	3.42e-14
Nasa/nasasrb	3.87e-10	3.70e-10	3.59e-10	3.50e-10	3.67e-10
CEMW/tmt_sym	2.70e-10	8.68e-10	7.62e-09	1.10e-09	7.02e-10
Schmid/thermal2	1.16e-13	4.82e-13	1.11e-10	8.97e-13	4.71e-13
Rothberg/gearbox	7.88e-15	3.22e-15	3.33e-15	7.11e-15	3.22e-15
INPRO/msdoor	9.15e-10	5.99e-10	9.00e-10	6.37e-10	6.09e-10
DNVS/m_t1	4.57e-07	2.34e-07	1.99e-07	5.30e-07	3.51e-07
McRae/ecology2	2.08e-11	1.55e-09	1.16e-09	1.81e-09	1.09e-09
Boeing/pwtk	3.93e-08	7.37e-08	6.87e-08	4.49e-08	6.17e-08
Chen/pkustk13	7.55e-15	2.78e-15	3.55e-15	8.10e-15	3.33e-15
BenElechi/BenElechi1	9.04e-10	2.36e-09	8.51e-10	2.59e-09	1.64e-09
Rothberg/cfd2	2.87e-12	2.49e-12	2.11e-12	3.41e-12	2.60e-12
DNVS/thread	1.96e-08	1.77e-08	2.27e-08	1.69e-08	1.50e-08
DNVS/shipsec8	1.34e-07	1.69e-07	1.76e-07	1.70e-07	1.85e-07
DNVS/shipsec1	3.63e-09	2.02e-09	1.90e-09	4.49e-09	5.92e-09
GHS_psdef/crankseg_2	8.30e-12	2.38e-12	5.33e-12	6.43e-12	8.62e-13
DNVS/fcondp2	5.88e-15	2.55e-15	2.89e-15	6.88e-15	2.78e-15
Schenk_AFE/af_shell3	8.03e-12	6.53e-12	6.00e-12	1.00e-11	7.48e-12
DNVS/troll	7.33e-15	3.11e-15	3.00e-15	7.66e-15	2.89e-15
GHS_psdef/bmwcra_1	7.81e-11	4.99e-11	1.78e-10	6.01e-11	3.46e-11
DNVS/halfb	6.77e-15	2.89e-15	4.00e-15	7.11e-15	3.00e-15
GHS_psdef/crankseg_1	1.69e-11	5.30e-12	2.43e-12	7.39e-12	3.69e-12
Um/2cubes_sphere	1.44e-14	2.66e-15	6.00e-15	1.60e-14	6.00e-15
GHS_psdef/ldoor	5.35e-11	7.53e-12	1.09e-11	2.11e-11	2.53e-11
DNVS/ship_003	5.03e-09	5.21e-09	9.67e-09	2.66e-08	1.41e-08
DNVS/fullb	7.22e-15	3.11e-15	3.55e-15	7.66e-15	3.22e-15
Um/offshore	4.38e-12	1.83e-12	1.48e-12	4.70e-12	2.50e-12
GHS_psdef/inline_1	1.83e-10	3.52e-10	6.94e-10	4.21e-10	1.61e-10
Chen/pkustk14	1.35e-14	3.66e-15	7.66e-15	1.19e-14	4.66e-15
GHS_psdef/apache2	5.57e-12	3.35e-11	5.88e-11	3.18e-11	3.46e-11
Koutsovasilis/F1	6.36e-13	4.63e-13	1.43e-12	8.90e-13	5.88e-13
Oberwolfach/boneS10	6.12e-10	6.47e-10	8.19e-10	6.59e-10	6.51e-10
AMD/G3_circuit	2.69e-12	4.72e-12	8.50e-12	2.45e-12	6.28e-12
ND/nd12k	3.45e-11	6.54e-11	2.47e-11	5.69e-10	6.91e-11
JGD_Trefethen/Trefethen_20000	3.08e-14	3.89e-15	1.74e-14	4.51e-14	(4) 2.72e-14
ND/nd24k	5.30e-11	1.09e-10	1.41e-11	7.96e-10	6.22e-11
Oberwolfach/bone010	5.56e-09	7.51e-09	6.11e-09	6.78e-09	6.84e-09
GHS_psdef/audikw_1	2.57e-11	4.69e-11	1.18e-11	2.77e-11	2.60e-11

A.8.3 Test Set 3: General indefinite problems

All results are determined using a serial run of the codes. Numbers in italics represent forward errors more than four orders of magnitude greater than best solve.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Oberwolfach/t2dal	3.37e-03	5.20e-04	1.52e-03	(5) -	9.13e-04
GHS_indef/dixmaanl	5.18e-13	7.03e-13	5.02e-13	2.15e-12	3.12e-13
Oberwolfach/rail_79841	6.61e-12	2.07e-12	3.13e-12	9.76e-13	3.53e-12
GHS_indef/dawson5	1.37e-10	7.34e-11	2.44e-11	2.62e-10	1.30e-10
Boeing/bcsstk39	2.05e-11	1.34e-11	1.50e-11	1.89e-11	3.53e-11
GHS_indef/helm2d03	1.60e-11	8.02e-12	5.56e-12	1.48e-12	1.18e-11
GHS_indef/copter2	1.13e-10	1.11e-10	2.03e-10	1.53e-09	6.05e-11
Boeing/crystk03	3.70e-02	4.64e-02	2.46e-02	(7) 1.00e+00	2.05e-02
Oberwolfach/filter3D	1.63e-12	1.11e-12	6.79e-13	1.71e-12	1.98e-12
Boeing/pct20stif	2.61e-10	6.15e-10	1.91e-10	3.12e-10	6.29e-11
Koutsovasilis/F2	3.25e-01	6.70e+00	2.89e-01	2.74e+00	1.83e+00
Cunningham/qa8fk	6.42e-01	7.06e-01	6.45e-01	(5) -	8.37e-01
Oberwolfach/gas_sensor	3.76e-11	4.05e-11	4.90e-11	3.73e-11	5.23e-10
McRae/ecology1	9.87e-01	9.83e-01	9.91e-01	(7) 9.97e-01	1.44e+00
Oberwolfach/t3dh	1.62e-05	4.48e-04	5.15e-05	(7) *9.86e-01*	2.60e-03
Lin/Lin	8.30e-11	8.09e-12	1.73e-11	6.30e-12	7.21e-12
GHS_indef/sparsine	1.20e-07	2.63e-07	7.12e-08	1.03e-06	1.27e-08
PARSEC/Ge99H100	4.34e-10	4.81e-10	1.15e-10	6.44e-11	4.37e-10
PARSEC/Ga10As10H30	3.51e-11	3.37e-10	4.41e-11	1.71e-10	8.12e-11
PARSEC/Ga19As19H42	1.37e-10	4.17e-10	1.79e-10	9.87e-11	1.36e-10

A.8.4 Test Set 4: KKT problems

All results are determined using a serial run of the codes. Numbers in italics represent forward errors more than four orders of magnitude greater than best solve.

Problem	MA57	MA86	MA97	PARDISO	WSMP
GHS_indef/boyd1	1.99e-06	2.95e-06	3.95e-06	3.58e-06	3.42e-06
GHS_indef/bmw3_2 (1)	1.21e-06	1.03e-06	9.81e-07	7.46e-07	9.57e-07
GHS_indef/c-72 (1)	4.35e-12	1.32e-12	8.63e-13	*2.31e-05*	1.86e-11
GHS_indef/ncvxqp7 (1)	8.10e-03	6.75e-02	1.08e-02	1.35e-01	3.66e-02
Andrianov/mip1	1.16e-08	8.60e-09	1.97e-09	2.80e-07	1.79e-09
GHS_indef/blockqp1	1.73e-11	2.75e-11	1.39e-11	2.68e-12	6.23e-12
GHS_indef/boyd2	9.40e-05	6.59e-05	8.03e-05	9.30e-05	4.22e-05
GHS_indef/a5esindl	3.01e-11	6.01e-12	6.10e-12	4.18e-11	3.40e-12
GHS_indef/a2nnsnsl	7.86e-11	3.69e-11	3.85e-11	1.83e-10	7.77e-11
GHS_indef/a0nsdsil	3.15e-11	1.21e-11	8.09e-12	4.93e-11	7.27e-12
TSOPF/TSOPF_FS_b39_c30 (1)	1.38e-06	6.13e-07	7.06e-07	(7) *2.10e+08*	8.09e-07
GHS_indef/cont-201 (1)	7.57e-08	3.11e-08	3.92e-08	*4.04e-04*	9.54e-09
GHS_indef/darcy003 (1)	1.59e-12	5.64e-13	6.46e-13	4.49e-10	4.04e-13
GHS_indef/cont-300 (1)	1.87e-08	1.56e-07	1.71e-07	(7) *8.90e-02*	7.19e-08
GHS_indef/turon_m (1)	9.86e-01	8.09e-01	1.01e+00	1.00e+00	9.99e-01
GHS_indef/d_pretok (1)	1.08e+00	1.10e+00	1.00e+00	1.00e+00	1.00e+00
TSOPF/TSOPF_FS_b300_c3 (1)	1.24e-05	8.81e-05	2.50e-05	(7) *5.60e+12*	1.52e-05
GHS_indef/dtoc (1)	8.76e+04	5.00e+03	5.00e+03	1.08e+00	1.00e+00
GHS_indef/aug2d (1)	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00
GHS_indef/aug3d (1)	1.00e+00	1.00e+00	1.00e+00	1.00e+00	1.00e+00

A.9 Scaled backward error comparisons

A.9.1 Test Set 1: Small problems

All results are determined using a serial run of the codes. Italics are used to highlight backward errors greater than 1×10^{-12}.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Cunningham/m3plates	8.52e-17	8.52e-17	8.52e-17	(2) -	8.52e-17
Boeing/bcsstm37	1.73e-16	1.15e-16	1.73e-16	(7) *1.24e-12*	1.16e-16
GHS_indef/qpband	7.16e-17	7.16e-17	2.15e-16	2.87e-16	6.94e-17
HB/zenios	3.86e-16	9.68e-16	3.03e-16	(7) *2.48e-11*	1.65e-16
HB/saylr3	1.89e-16	2.12e-16	2.14e-16	1.17e-16	1.92e-16
HB/sherman1	1.89e-16	2.12e-16	2.14e-16	1.68e-16	1.61e-16
Oberwolfach/filter2D	5.55e-16	2.12e-16	3.59e-16	3.59e-16	2.99e-16
RAL/a2ensndl-00	8.46e-16	2.89e-16	7.24e-16	1.09e-17	(3) 1.80e-16
RAL/a2ensndl-49 (1)	1.41e-14	7.41e-14	6.30e-14	(6,7) 7.10e-25	6.76e-14
RAL/a2ensndl-62	*1.69e-09*	*7.13e-09*	*7.15e-09*	2.95e-19	(3) *1.59e-09*
Schenk_IBMNA/c-29 (1)	1.04e-16	1.04e-16	8.29e-19	*6.18e-12*	1.04e-16
Boeing/nasa1824	2.13e-16	2.74e-16	2.50e-16	*2.02e-07*	1.42e-16
GHS_indef/spmsrtls	6.89e-15	7.32e-15	7.53e-15	*6.79e-13*	6.56e-15
IPSO/OPF_3754 (1)	1.70e-15	2.04e-14	2.89e-15	(6,7) *2.28e-04*	2.42e-15
GHS_indef/stokes64 (1)	9.27e-15	3.02e-15	1.07e-14	4.09e-16	6.76e-15
GHS_indef/brainpc2 (1)	1.34e-14	6.80e-15	3.40e-14	*2.71e-06*	1.01e-14
TSOPF/TSOPF_FS_b9_c6 (1)	8.18e-14	1.10e-14	6.55e-16	(7) *2.29e-10*	1.72e-14
Nemeth/nemeth05	7.05e-16	6.27e-16	6.27e-16	1.02e-15	5.30e-16
Nemeth/nemeth10	6.21e-16	4.14e-16	6.73e-16	1.24e-15	4.59e-16
FIDAP/ex14 (1)	5.40e-14	9.50e-14	1.27e-13	1.24e-16	3.22e-13
Nemeth/nemeth15	1.50e-15	1.58e-15	1.03e-15	2.97e-14	7.79e-16
GHS_indef/ncvxqp9 (1)	1.20e-16	2.70e-19	6.80e-20	1.02e-22	4.21e-18
Schenk_IBMNA/c-41	5.63e-14	1.49e-14	1.00e-14	*5.11e-10*	*1.63e-12*
GHS_indef/tuma2 (1)	1.62e-13	3.51e-14	4.67e-14	1.78e-16	2.30e-13
Marini/eurqsa	(6) 3.49e-15	4.60e-14	1.54e-14	(6,7) *5.59e-03*	(6) 7.92e-16
Oberwolfach/rail_20209	3.18e-16	3.30e-16	2.67e-16	5.59e-16	3.21e-16
Newman/hep-th (1)	4.50e-17	8.83e-18	4.99e-17	3.56e-15	(6,7) 9.11e-19
Nemeth/nemeth20	1.41e-14	1.74e-14	9.55e-15	3.26e-14	8.22e-14
PARSEC/Si2	7.93e-15	7.62e-16	1.19e-14	6.51e-16	4.99e-16
Oberwolfach/t2dah_a	1.36e-16	1.75e-16	1.86e-16	(7) *1.55e-12*	9.80e-17
IPSO/TSC_OPF_300 (1)	*5.94e-12*	*2.59e-12*	1.88e-13	*4.85e-09*	*1.53e-12*
Nemeth/nemeth25 (1)	4.55e-14	2.65e-14	3.41e-14	6.89e-14	1.78e-14
Schenk_IBMNA/c-50 (1)	1.94e-16	1.86e-16	2.03e-16	*1.85e-11*	1.11e-16
Newman/cond-mat (1)	4.38e-17	1.25e-17	8.20e-15	1.16e-15	(6,7) 4.26e-18
Boeing/crystk01	2.11e-16	2.73e-16	2.93e-16	(7) *3.85e-05*	1.96e-16
TSOPF/TSOPF_FS_b39_c7 (1)	8.07e-14	2.55e-14	3.64e-14	(6,7) *3.19e-10*	8.90e-14
Boeing/bcsstk37 (1)	1.62e-16	2.80e-16	2.28e-16	2.94e-16	2.91e-16
GHS_indef/exdata_1	1.61e-16	1.61e-16	1.61e-16	8.06e-17	1.61e-16
TSOPF/TSOPF_FS_b162_c1 (1)	*3.82e-12*	3.50e-13	9.49e-13	(6,7) *1.41e-10*	1.58e-13
Boeing/crystk02	2.48e-16	3.33e-16	2.55e-16	(7) *6.11e-05*	2.70e-16

A.9.2 Test Set 2: Positive-definite problems

All results are determined using a serial run of the codes. Italics are used to highlight backward errors greater than 1×10^{-12}.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Mulvey/finan512	8.59e-16	5.25e-16	6.68e-16	9.55e-16	4.26e-16
MaxPlanck/shallow_water1	8.04e-16	4.69e-16	4.69e-16	8.71e-16	4.35e-16
UTEP/Dubcova3	8.17e-16	3.68e-16	3.81e-16	8.72e-16	5.43e-16
Nasa/nasasrb	1.56e-16	1.17e-16	1.27e-16	2.23e-16	1.62e-16
CEMW/tmt_sym	3.54e-16	2.82e-16	4.00e-16	4.96e-16	1.80e-16
Schmid/thermal2	4.94e-16	5.85e-16	6.40e-16	8.87e-16	5.30e-16
Rothberg/gearbox	1.95e-15	7.67e-16	9.07e-16	2.09e-15	1.02e-15
INPRO/msdoor	7.33e-16	3.30e-16	3.30e-16	9.89e-16	7.29e-16
DNVS/m_t1	7.30e-16	3.52e-16	2.52e-16	8.05e-16	2.61e-16
McRae/ecology2	1.08e-15	1.69e-15	1.14e-15	2.21e-15	1.03e-15
Boeing/pwtk	2.93e-16	1.60e-16	1.65e-16	2.15e-16	1.58e-16
Chen/pkustk13	1.52e-15	8.61e-16	7.95e-16	1.86e-15	1.10e-15
BenElechi/BenElechi1	8.09e-16	3.77e-16	2.94e-16	8.58e-16	3.46e-16
Rothberg/cfd2	7.46e-16	4.30e-16	3.06e-16	6.26e-16	4.61e-16
DNVS/thread	2.30e-16	2.84e-16	2.31e-16	3.55e-16	2.20e-16
DNVS/shipsec8	5.65e-16	3.39e-16	3.77e-16	1.28e-15	2.88e-16
DNVS/shipsec1	3.87e-16	2.58e-16	2.90e-16	4.84e-16	2.07e-16
GHS_psdef/crankseg_2	3.34e-15	1.37e-15	2.81e-15	2.43e-15	1.34e-16
DNVS/fcondp2	2.24e-15	9.41e-16	9.41e-16	2.29e-15	8.59e-16
Schenk_AFE/af_shell3	4.50e-16	4.22e-16	3.72e-16	6.09e-16	3.44e-16
DNVS/troll	2.02e-15	1.01e-15	8.66e-16	2.52e-15	8.45e-16
GHS_psdef/bmwcra_1	3.74e-16	9.45e-16	2.99e-16	3.58e-16	2.95e-16
DNVS/halfb	2.45e-15	1.23e-15	8.49e-16	2.26e-15	1.29e-15
GHS_psdef/crankseg_1	3.95e-15	3.04e-15	1.90e-15	3.34e-15	2.05e-15
Um/2cubes_sphere	7.60e-16	3.26e-16	3.80e-16	1.09e-15	3.42e-16
GHS_psdef/ldoor	1.17e-15	3.27e-16	3.27e-16	9.35e-16	1.56e-15
DNVS/ship_003	4.00e-16	9.22e-16	4.85e-16	9.70e-16	4.96e-16
DNVS/fullb	2.32e-15	9.48e-16	1.00e-15	1.90e-15	9.24e-16
Um/offshore	1.20e-15	4.32e-16	3.84e-16	8.88e-16	3.18e-16
GHS_psdef/inline_1	9.24e-16	1.95e-16	2.92e-16	1.22e-15	1.95e-16
Chen/pkustk14	2.95e-15	1.43e-15	1.52e-15	2.95e-15	1.22e-15
GHS_psdef/apache2	2.42e-15	1.90e-15	1.62e-15	4.21e-15	1.13e-15
Koutsovasilis/F1	2.23e-16	2.53e-16	2.95e-16	3.16e-16	1.81e-16
Oberwolfach/boneS10	9.84e-16	1.61e-15	1.13e-15	1.46e-15	7.81e-16
AMD/G3_circuit	1.48e-14	1.03e-15	7.05e-15	9.10e-15	1.19e-15
ND/nd12k	3.30e-15	2.44e-15	2.74e-15	4.20e-15	1.72e-15
JGD_Trefethen/Trefethen_20000	5.44e-15	7.12e-16	6.41e-15	9.19e-15	(4) 1.62e-15
ND/nd24k	6.76e-15	4.01e-15	4.86e-15	4.85e-15	1.84e-15
Oberwolfach/bone010	1.59e-15	5.04e-15	1.74e-15	3.04e-15	1.13e-15
GHS_psdef/audikw_1	8.37e-17	1.34e-16	7.81e-17	2.12e-16	1.17e-16

A.9.3 Test Set 3: General indefinite problems

All results are determined using a serial run of the codes. Italics are used to highlight backward errors greater than 1×10^{-12}.

Problem	MA57	MA86	MA97	PARDISO	WSMP
Oberwolfach/t2dal	2.07e-16	9.68e-17	2.15e-16	(5) -	1.43e-16
GHS_indef/dixmaanl	2.40e-14	1.87e-14	5.36e-14	1.36e-13	2.92e-14
Oberwolfach/rail_79841	2.83e-16	3.30e-16	2.83e-16	3.66e-16	3.77e-16
GHS_indef/dawson5	*1.31e-12*	9.22e-13	6.72e-13	8.18e-13	3.42e-13
Boeing/bcsstk39	1.84e-15	3.19e-16	7.18e-16	5.59e-16	5.10e-16
GHS_indef/helm2d03	5.10e-13	1.70e-13	3.94e-13	4.62e-14	2.24e-13
GHS_indef/copter2	*4.99e-12*	*2.71e-12*	*1.55e-12*	*5.82e-12*	3.63e-13
Boeing/crystk03	3.23e-16	4.31e-16	3.20e-16	(7) *2.19e-05*	2.32e-16
Oberwolfach/filter3D	3.98e-16	3.48e-16	2.76e-16	4.30e-16	4.98e-16
Boeing/pct20stif	*1.20e-12*	8.08e-13	6.44e-13	3.15e-13	2.24e-13
Koutsovasilis/F2	1.90e-16	1.12e-16	1.82e-16	1.51e-16	1.11e-16
Cunningham/qa8fk	2.17e-15	2.29e-15	1.86e-15	(5) -	1.87e-15
Oberwolfach/gas_sensor	2.30e-16	4.36e-16	3.13e-16	3.86e-16	4.08e-16
McRae/ecology1	1.16e-15	1.43e-15	9.47e-16	(7) *1.53e-11*	1.67e-15
Oberwolfach/t3dh	3.57e-16	3.01e-16	4.92e-16	(7) *5.23e-10*	1.75e-15
Lin/Lin	*1.17e-11*	5.67e-13	*1.97e-12*	7.12e-13	3.88e-13
GHS_indef/sparsine	*1.70e-11*	*2.59e-11*	*9.71e-12*	*1.57e-10*	*2.78e-12*
PARSEC/Ge99H100	3.54e-11	3.97e-11	*8.65e-12*	*4.12e-12*	1.34e-11
PARSEC/Ga10As10H30	8.62e-14	7.20e-13	8.55e-14	2.65e-13	9.33e-14
PARSEC/Ga19As19H42	1.24e-13	9.76e-13	1.20e-13	9.80e-14	1.17e-13

A.9.4 Test Set 4: KKT problems

All results are determined using a serial run of the codes. Italics are used to highlight backward errors greater than 1×10^{-12}.

Problem	MA57	MA86	MA97	PARDISO	WSMP
GHS_indef/boyd1	*1.03e-09*	*1.81e-09*	*3.90e-09*	*2.73e-09*	*2.69e-09*
GHS_indef/bmw3_2 (1)	3.84e-16	2.96e-16	2.36e-16	5.91e-16	5.46e-16
GHS_indef/c-72 (1)	1.60e-16	1.15e-16	6.57e-17	*8.28e-11*	1.31e-16
GHS_indef/ncvxqp7 (1)	*7.98e-10*	*1.03e-09*	*1.53e-09*	2.08e-16	*1.37e-08*
Andrianov/mip1	*6.52e-12*	*8.73e-12*	*1.97e-12*	*8.71e-12*	1.03e-14
GHS_indef/blockqp1	3.01e-13	3.01e-13	3.03e-13	5.82e-15	4.55e-17
GHS_indef/boyd2	*5.52e-07*	*3.96e-07*	*3.09e-07*	*1.40e-06*	*2.50e-07*
GHS_indef/a5esindl	2.49e-16	1.27e-16	1.72e-16	2.07e-15	3.55e-16
GHS_indef/a2nnsnsl	1.02e-15	2.98e-15	2.15e-16	1.00e-15	8.60e-16
GHS_indef/a0nsdsil	1.31e-15	2.84e-15	1.76e-15	2.10e-15	5.98e-16
TSOPF/TSOPF_FS_b39_c30 (1)	1.35e-13	6.75e-14	9.65e-14	(6,7) *3.72e-10*	2.05e-13
GHS_indef/cont-201 (1)	*5.95e-11*	*5.54e-11*	*2.26e-11*	*1.25e-11*	*1.61e-09*
GHS_indef/darcy003 (1)	3.22e-14	2.32e-14	3.05e-14	1.18e-15	5.21e-15
GHS_indef/cont-300 (1)	*9.62e-11*	*6.29e-11*	*4.38e-11*	(7) *1.82e-09*	*1.97e-09*
GHS_indef/turon_m (1)	1.33e-15	1.31e-15	1.97e-15	9.24e-17	3.97e-16
GHS_indef/d_pretok (1)	5.28e-16	3.22e-16	3.38e-16	3.43e-16	5.31e-16
TSOPF/TSOPF_FS_b300_c3 (1)	1.44e-13	1.01e-13	2.37e-13	(6,7) *1.44e-11*	5.45e-14
GHS_indef/dtoc (1)	7.73e-19	6.66e-20	2.00e-19	7.40e-17	2.96e-16
GHS_indef/aug2d (1)	0.00e+00	0.00e+00	0.00e+00	8.95e-22	1.86e-214
GHS_indef/aug3d (1)	0.00e+00	0.00e+00	0.00e+00	3.69e-22	4.06e-215