Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate.

Article *in* ACM Transactions on Mathematical Software · January 2008

Source: DBLP

4 authors, including:

Tim Davis
Texas A&M University
88 PUBLICATIONS **6,178** CITATIONS
SEE PROFILE

William Hager
University of Florida
99 PUBLICATIONS **3,686** CITATIONS
SEE PROFILE

Siva Rajamanickam
Sandia National Laboratories
50 PUBLICATIONS **490** CITATIONS
SEE PROFILE

Algorithm 8xx: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*

YANQING CHEN†, TIMOTHY A. DAVIS‡, WILLIAM W. HAGER§, and SIVASANKARAN RAJAMANICKAM¶

September 8, 2006

Technical report TR-2006-005, CISE Dept, Univ. of Florida, Gainesville, FL

Abstract

CHOLMOD is a set of routines for factorizing sparse symmetric positive definite matrices of the form A or AA^T, updating/downdating a sparse Cholesky factorization, solving linear systems, updating/downdating the solution to the triangular system $Lx = b$, and many other sparse matrix functions for both symmetric and unsymmetric matrices. Its supernodal Cholesky factorization relies on LAPACK and the Level-3 BLAS, and obtains a substantial fraction of the peak performance of the BLAS. Both real and complex matrices are supported. CHOLMOD is written in ANSI/ISO C, with both C and MATLAB$^{\text{TM}}$interfaces. It appears in MATLAB 7.2 as `x=A\b` when `A` is sparse symmetric positive definite, as well as in several other sparse matrix functions.

1 Overview

Methods for the direct solution of a sparse linear system $Ax = b$ rely on matrix factorizations. When A is symmetric positive definite, sparse Cholesky factorization is typically used. Supernodal and multifrontal methods are among those that can exploit dense matrix kernels (the BLAS [26, 14, 13]) and can thus achieve high performance on modern computers. For more background on direct methods for sparse matrices, see [7, 16, 15]. Supernodal methods are discussed by [3, 12, 24, 29, 32, 33]

*This work was supported by the National Science Foundation, under grants 0203270 and 0620286.

†Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, USA. email: yc1@cise.ufl.edu.

‡Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, USA. email: davis@cise.ufl.edu. http://www.cise.ufl.edu/∼davis.

§Dept. of Mathematics, Univ. of Florida, Gainesville, FL, USA. email: hager@math.ufl.edu. http://www.math.ufl.edu/∼hager.

¶Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, USA. email: srajaman@cise.ufl.edu.

CHOLMOD is a package that provides sparse Cholesky factorization methods and related sparse matrix functions. It includes both a supernodal method and a non-supernodal up-looking method [6] that does not exploit the BLAS. If the original matrix A is replaced by $A \pm WW^T$, where W is n-by-k with $k << n$, the package can update or downdate the factorization while exploiting *dynamic supernodes*. An update or downdate of a conventional supernodal structure is not feasible, since changes in the nonzero pattern of L cause supernodes to merge and split apart. Reorganizing a conventional supernodal structure of L would dominate the run time of the update/downdate. Dynamic supernodes, in contrast, are detected and exploited as the algorithm progresses. For a rank-1 update/downdate, the running time is proportional to the number of entries in L that change; the time for a rank-k update/downdate is proportional to the time for k separate rank-1 update/downdates. [9, 10]. CHOLMOD also exploits dynamic supernodes in the triangular solves, and can update/downdate the corresponding solution to $Lx = b$ after updating/downdating L itself. A detailed discussion of how CHOLMOD uses dynamic supernodes is given in [11].

Section 2 summarizes the features in CHOLMOD. Performance of CHOLMOD's sparse Cholesky factorizations and nested dissection ordering methods are given in Section 3. Section 4 explains how to obtain the code and the packages it relies on.

2 Features

CHOLMOD is composed of a set of modules, each of which defines a set of objects and/or operations on those objects. CHOLMOD's modules (Core, Cholesky, Check, Demo, MATLAB$^{\text{TM}}$, MatrixOps, Modify, Partition, and Supernodal) are described below. In addition, CHOLMOD includes a full user guide, a Makefile, and an exhaustive test suite that exercises 100% of the statements in the code.

2.1 Core Module

The Core Module defines the five basic objects that CHOLMOD supports:

1. `cholmod_sparse`: a sparse matrix in compressed sparse column form, either symmetric or unsymmetric (the latter may be rectangular). In the symmetric case, either the upper or lower triangular part is stored. Most of CHOLMOD's functions operate on `cholmod_sparse` matrices.

2. `cholmod_triplet`: a sparse matrix in *triplet* form (a list of nonzero values with their row and column indices). This is a flexible data structure for the user to generate. Only a few functions are provided (including functions for converting to and from the triplet form).

3. `cholmod_dense`: a dense matrix in column-oriented form (compatible with MATLAB and Fortran).

4. `cholmod_factor`: a sparse Cholesky factorization, either supernodal or non-supernodal, and either LL^T or LDL^T (where D is diagonal).

5. `cholmod_common`: CHOLMOD parameters, workspace, and statistics.

The first four objects can be real or complex (both double precision), or pattern-only (with no numerical values). CHOLMOD supports two forms of complex matrices and factors: C/C++/Fortran-style, and the split-style used in MATLAB. The Core Module provides functions to allocate, reallocate, and free these objects. The Core can also copy all but the `cholmod_common` object, and convert between different object types.

Lower case letters are used for subsets (`f`) or permutation vectors (`p`); `alpha` and `beta` are scalars. Sparse matrix operators (for `cholmod_sparse`) in the Core Module include (in MATLAB notation) `A*A'`, `A(:,f)*A(:,f)'`, `alpha*A+beta*B`, `tril`, `triu`, `diag`, `A'`, `A(p,f)'`, and `A(p,p)'`. The Core Module includes a function to sort row indices within each sparse column vector and a function to extract entries within a band.

The `cholmod_factor` object contains either a symbolic or numeric factorization, in either LL^T or LDL^T form, and either supernodal or non-supernodal. The Core Module includes a function that converts a `cholmod_factor` between these various forms.

2.2 Cholesky Module

The Cholesky Module provides functions for computing or using a sparse Cholesky factorization. These functions compute the elimination tree of `A` or `A*A'` and its postordering, row/column counts of `chol(A)` or `chol(A*A')`, the symbolic factorization of `A` or `A*A'`, an interface to a supernodal Cholesky factorization, non-supernodal LL^T and LDL^T factorizations (up-looking), incremental LL^T and LDL^T factorizations (one row at a time), a *symbolic refactorization* (to remove entries after an update/downdate), and solutions to upper/lower triangular systems (supernodal, dynamic supernodal, and dense, sparse, and/or multiple right-hand sides). The module also provides an interface to the AMD [1] and COLAMD [8] ordering methods. The methods used in this Module are discussed by [28, 18, 17, 16] and [6].

If the Supernodal Module is installed, the sparse Cholesky factorization automatically selects between a non-supernodal up-looking factorization, and a supernodal BLAS-based factorization. The former is much faster than the supernodal method for very sparse matrices (such as tridiagonal matrices). The selection is done during symbolic analysis, when $|L|$ and the floating-point operation count are found.[1] If the operation count divided by $|L|$ is greater than or equal to 40, then the supernodal method is selected. Otherwise, the non-supernodal method is selected. This ratio is a coarse metric of how much memory-reuse a supernodal method will be able to exploit during numeric factorization. If it is high, then the supernodes will tend to be large and the BLAS routines will be efficient. If it is low, there will be little scope for efficient exploitation of cache within the BLAS kernels. The use of this ratio was selected analytically based on a model of cache memory behavior, but the default threshold of 40 was selected based on performance measurements on a large range of sparse matrices arising in real applications. These measurements are given in Section 3.

If the Partition Module is not installed, CHOLMOD uses a minimum degree ordering (AMD or COLAMD). Otherwise, it automatically selects between minimum degree and

[1] $|L|$ denotes the number of nonzeros in L, or `nnz(L)` in MATLAB notation.

nested dissection. The default strategy is to first try AMD. If AMD finds an ordering where the floating point operation divided by $|L|$ is less than 500, or if $|L| < 5|A|$, then the AMD ordering is used and no other ordering is attempted. Otherwise, METIS [25] or CHOLMOD's nested dissection ordering (NESDIS) is tried, and the best ordering (the one with the lowest $|L|$) is used.

CHOLMOD can also be requested to try up to nine different orderings and select the best one found (user-provided, natural, AMD/COLAMD, METIS, NESDIS, and the latter three with varying non-default parameter selections). This is useful in the case where a sequence of matrices with identical nonzero pattern must be factorized, as often occurs in nonlinear solvers, optimization methods, eigensolvers, and many other applications.

Note that MATLAB 7.2 includes neither the Partition Module nor METIS. It thus always uses AMD for its ordering, in `x=A\b` when `A` is sparse and symmetric positive definite.

2.3 Check Module

The Check Module checks the validity of the five CHOLMOD objects, and prints their contents. It can also read a sparse matrix from a file, in triplet or Matrix Market form [4].

2.4 Demo Module

The Demo Module provides sample main programs to illustrate how CHOLMOD can be used in a user's application.

2.5 MATLAB Module

The MATLAB Module is CHOLMOD's interface to MATLAB, providing most of CHOLMOD's functionality to the MATLAB environment. Note that MATLAB 7.2 already includes much of CHOLMOD itself, as built-in functions (namely, the Core, Cholesky, MATLAB, and Supernodal Modules). Built-in functions that rely on CHOLMOD include `x=A\b` and `x=b/A` when `A` is sparse and symmetric positive definite, `chol`, `etree`, and `symbfact`.

CHOLMOD provides additional functions to the MATLAB caller that are not built into MATLAB, including nested dissection orderings (METIS and NESDIS), and the ability to update and downdate a sparse LDL^T factorization. Its sparse-matrix-times-dense-matrix function, `sdmult`, is significantly faster than the corresponding operation in MATLAB 7.2, particularly when the dense matrix has many columns.

2.6 MatrixOps Module

The MatrixOps Module provides functions for the `cholmod_sparse` object, including `[A,B]`, `[A;B]`, `alpha*A*X+beta*Y` and `alpha*A'*X+beta*Y` where `A` is sparse and `X` and `Y` are dense, `A*B`, and `A(i,j)` where `i` and `j` are arbitrary integer vectors. It can also compute norms of sparse or dense matrices. It can perform row and column scaling, and can drop small entries from a sparse matrix.

2.7 Modify Module

The Modify Module provides functions that can add or delete a row of `L` (from an LDL^T factorization) and compute the dynamic supernodal update/downdate of an LDL^T factorization. The solution to $Lx = b$ can also be updated/downdated when L is modified. Details of the methods used are given by [9, 10, 11]. For background on update/downdate methods, see [19, 23, 34, 35].

2.8 Partition Module

The Partition Module provides graph-partitioning based orderings. The graph partitioning methods are currently based on METIS. Thus, this Module can only be used in CHOLMOD if METIS is also available.

The Partition Module includes an interface to three *constrained* minimum degree ordering methods: CAMD, CCOLAMD, and CSYMAMD. These methods are constrained versions of AMD, COLAMD, and SYMAMD, respectively. In a constrained minimum degree ordering method, each node is in one of up to n constraint sets. All nodes in constraint set zero are ordered first, followed by all nodes in set one, and so on. These ordering methods are most useful when combined with nested dissection [30, 31].

The Module provides a direct interface to METIS functions for computing a node separator of an undirected graph (METIS_NodeComputeSeparator), and for computing the nested dissection ordering of an undirected graph (METIS_NodeND). The latter recursively partitions the graph (via node separators) until the subgraphs are small. Small subgraphs are ordered independently with minimum degree (MMD, [27]), not with a constrained minimum degree ordering.

CHOLMOD also includes its own nested dissection ordering, NESDIS. It uses METIS_NodeComputeSeparator to partition the graph recursively, in the same manner as METIS_NodeND. Unlike METIS_NodeND, which orders each subgraph independently, NESDIS uses the separators as ordering constraints for CAMD or CCOLAMD (for symmetric and unsymmetric matrices, respectively). This tends to result in a better ordering at the expense of a modest increase in ordering time. Performance comparisons between NESDIS, METIS_NodeND, and AMD/COLAMD are given in Section 3.

2.9 Supernodal Module

The Supernodal Module provides supernodal symbolic and numeric factorization (LL^T only) of A or AA^T, and a conventional supernodal triangular solver. Details of the method it uses are given in [11].

3 Performance

This section highlights the performance of CHOLMOD's sparse Cholesky factorization methods, and compares its nested dissection ordering with METIS.

mesh size, `n`	120	300
matrix dimension	10,443	66,603
`nnz(A)`	51,743	331,823
`nnz(L)`	1.02×10^6	16.5×10^6
flop count	109×10^6	4434×10^6
MATLAB 7.1 `x=A\b`		
time	0.556 seconds	18.31 seconds
Mflops	196	242
MATLAB 7.2 `x=A\b`		
time	0.124 seconds	2.82 seconds
Mflops	879	1572

Table 1: Performance of the MATLAB sparse benchmark (in `bench`)

3.1 Sparse Cholesky factorization

This next example illustrates the performance of CHOLMOD in MATLAB and also provides an interesting anecdote of how performance results can be misinterpreted.

The introduction of CHOLMOD into MATLAB 7.2 has prompted a recurring spurious "bug report" from MATLAB users. The MATLAB `bench` program benchmarks MATLAB and compares the performance with other computers running the *same* version of MATLAB. The sparse benchmark in `bench` is the sparse Cholesky factorization of a matrix arising from the discretization of a 2D L-shaped domain, using the natural ordering instead of the default fill-reducing ordering. The matrix is `A=delsq(numgrid('L',n))`, where the mesh size is `n`-by-`n`. In MATLAB 7.1, the mesh size was 120-by-120. In MATLAB 7.2, the mesh size was increased to 300-by-300 to reflect the improved performance of `chol`. The results in Table I were obtained on a 3.2GHz Pentium 4 using the Goto BLAS [20].

Some MATLAB users, curious to compare the speed of different versions of MATLAB, compare the sparse `bench` time between MATLAB 7.1 and 7.2 and find that MATLAB 7.2 is "slower" (0.556 seconds in MATLAB 7.1 versus 2.82 seconds in MATLAB 7.2). They report this as a "bug" since MATLAB appears to have slowed down by a factor of about five. They fail to heed the note in `bench` that the problem sizes can differ; MATLAB 7.2 is doing about 40 times the work as the MATLAB 7.1 benchmark. For this matrix, the new sparse Cholesky factorization is about eight times faster than the old one, not five times slower.

For large matrices, sparse backslash when `A` is symmetric positive definite is typically five to ten times faster in MATLAB 7.2 as compared with the left-looking non-supernodal method in MATLAB 7.1. CHOLMOD's up-looking non-supernodal factorization, coupled with the better and faster AMD ordering, provides a speedup even for small and very sparse matrices. For large matrices, a speedup of up to 40 has been observed, due to the supernodal factorization in CHOLMOD and the better ordering. For example, for the ND/ND3K matrix, a 3D mesh, the backslash time reduces from 178.6 seconds in MATLAB 7.1 to 10.7 seconds in MATLAB 7.2 when using the default built-in orderings in `x=A\b` (SYMMMD in MATLAB 7.1 and AMD in MATLAB 7.2). With the Partition Module and CHOLMOD's nested dissection ordering, the time drops to 9.5 seconds (including the ordering time). With METIS, the time drops to 8.5 seconds (0.9 seconds of the difference are from a reduction in ordering time from

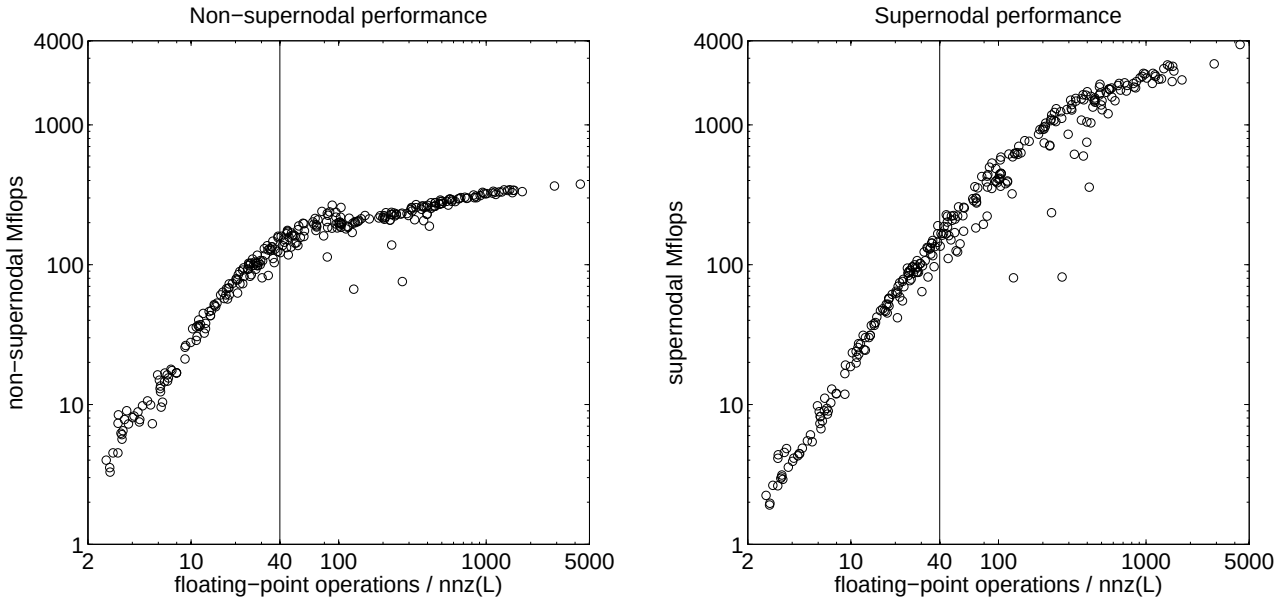

Figure 1: CHOLMOD supernodal and non-supernodal performance

2.1 to 1.2 seconds; the ordering quality is also slightly improved with METIS for this matrix).

Full details of an independent performance comparison between CHOLMOD and ten other solvers are given in [22, 21]. With 87 symmetric positive definite test matrices, CHOLMOD had the fastest total run time (including analysis, factorization, and solution of the triangular systems) for 55 matrices. It typically uses the least amount of memory as well. Considering just the 49 larger systems requiring 5 seconds or more to solve using the fastest method, CHOLMOD was fastest for 32 matrices, and had a run time no higher than 15% more than the fastest run time for all but two matrices.[2] These two matrices are from an interior point linear programming problem and contain many dense rows and columns that cause problems with many solvers. CHOLMOD's run time was 2.2 and 3.8 times the fastest run time for these two matrices, respectively. The bulk of the time was spent in AMD. Modifying AMD's default dense row/column control parameter can greatly reduce the ordering time for these two matrices.

Figure 1 shows the performance of CHOLMOD's non-supernodal up-looking method and its supernodal method, as a function of the ratio of floating-point operations over the number of nonzeros in L. Figure 2 compares the relative performance of these two methods. The test set of 320 matrices consists of all matrices in the UF Sparse Matrix Collection [5] that are either positive definite or binary with a symmetric nonzero pattern. For the binary matrices, numerical values were constructed to ensure the matrices were positive definite; off-diagonal entries were set to -1, and the diagonal entry a_{ii} was set to one plus the number of nonzero entries in column i. Random matrices were excluded. The x-axis of each plot is the floating-point operation count over $|L|$, which is the metric used to automatically select between the two methods. Run times include ordering, analysis, factorization, and solution of the resulting triangular systems. This is all the work done by `x=A\b`, except for the `mldivide`

[2]The GUPTA1 and GUPTA2 matrices, which are also the two worst-case outliers in Figure 1.

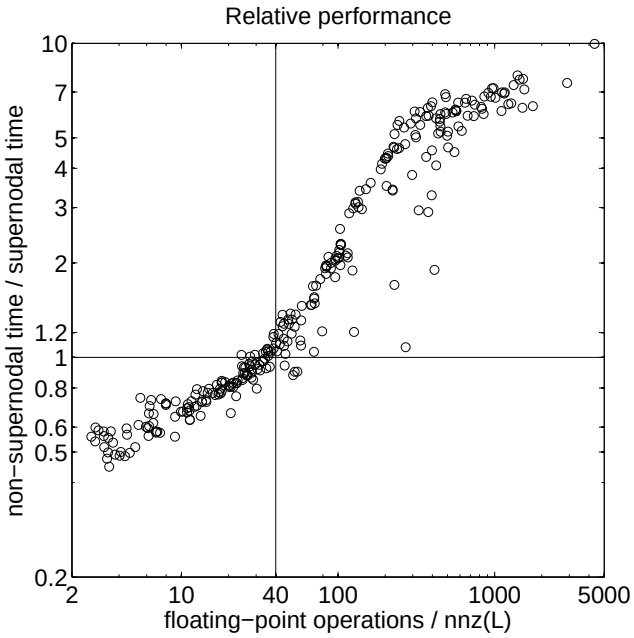

Figure 2: CHOLMOD relative supernodal and non-supernodal performance

meta-algorithm that selects which method to use (LU, Cholesky, QR, banded solver, etc.). Each circle in the figure is a single matrix.

These results indicate that the floating-point count per nonzero in L is a remarkably accurate method for predicting both the absolute and relative performance of these two methods. The relative performance, and more importantly the threshold of 40 flops/$|L|$ used to automatically select between the up-looking and supernodal method, may of course differ on different computers. The value of 40 is the default value of a parameter that the user can modify. Reasonable thresholds on a range of other computers are shown in Table II, based on the same test set. On all platforms listed, the flops/$|L|$ ratio proves to be an accurate predictor of the performance of each method.

On dual-core processors, a high flops/$|L|$ ratio is required to warrant the use of a multi-threaded BLAS, as compared to a single-threaded BLAS (about 300 on the Intel T2500, and 500 on the AMD Opteron).

3.2 Nested dissection ordering

The results in this section demonstrate the improvement in fill-in that can be obtained by combining nested dissection with a constrained minimum degree ordering.

AMD and the CHOLMOD and METIS nested dissection orderings were tested on 649 matrices with symmetric or nearly symmetric nonzero pattern. These are all matrices in the UF Sparse Matrix Collection that are either symmetric or whose nonzero pattern is more than 60% symmetric.

In addition, 203 unsymmetric matrices were selected from same collection. These are either rectangular, or square with a nonzero pattern that is less than 60% symmetric. Diag-

Computer	threshold	up-looking peak GFlops	supernodal peak GFlops	theoretical peak GFlops
Intel Pentium 4				
3.2GHz, 512KB cache, 4GB RAM				
Goto BLAS	40	0.38	3.76	6.4
Intel Pentium 4M				
2GHz, 512KB cache, 1GB RAM				
Intel MKL BLAS	40	0.20	1.77	4.0
Goto BLAS	40	0.20	1.80	4.0
Intel Core Duo T2500 (2-core)				
2GHz, 2MB cache, 2GB RAM				
Intel MKL BLAS (1 thread)	40	0.41	1.34	2.0
Goto BLAS (1 thread)	40	0.41	1.47	4.0
Goto BLAS (2 threads)	120	0.41	2.47	4.0
AMD Opteron 252 (2-core)				
2.6GHz, 1MB cache, 8GB RAM				
ACML BLAS (1 thread)	30	0.38	2.56	5.2
Goto BLAS (1 thread)	30	0.38	3.93	5.2
Goto BLAS (2 threads)	80	0.38	6.65	10.4
Sun UltraSparcII V9+vis (4-core)				
450GHz, 4MB cache, 4GB RAM				
ATLAS BLAS (1 thread)	45	0.06	0.43	0.9

Table 2: Performance of CHOLMOD sparse Cholesky factorization on a range of computers

onal matrices and matrices with fewer than 10,000 nonzeros were excluded from both test sets. CHOLMOD and METIS were compared with COLAMD for these matrices.

For the symmetric case or nearly symmetric case, the graph of `A+A'` was ordered. CHOLMOD's NESDIS function uses CAMD, a constrained version of AMD. For the unsymmetric case, the graph of `A'*A` was ordered if `A` had at least as many rows as columns (for LU factorization, or QR factorization for a least-squares problem). The matrix `A*A'` was ordered if `A` had fewer rows than columns (a linear programming problem). CHOLMOD uses CCOLAMD, a constrained version of COLAMD, in this case.

For the 649 symmetric matrices, AMD finds the best ordering for 311 matrices, METIS for 107, and NESDIS for 236 (the total is higher than 649 because of exact ties). AMD is 4.1 times faster than METIS, and 5.0 times faster than NESDIS (a median result). NESDIS is slower than METIS (by about 25%), but it typically finds a better ordering than METIS. The median fill-in is 2% less with NESDIS than with METIS for all 649 matrices.

Most sparse Cholesky factorization packages that include a nested dissection ordering method also include a minimum degree ordering method. Like CHOLMOD, it is typical to try both and select the method returning the best ordering, since minimum degree is so fast in practice. It is thus useful to compare nested dissection orderings only for matrices for which minimum degree does not find the best ordering. Of these 338 symmetric matrices, METIS finds a significantly better ordering than NESDIS (a reduction in $|L|$ by 5% or more) for 20 matrices, whereas NESDIS finds a significantly better ordering than METIS for 91. They essentially tie (within $\pm 5\%$) for the other 227 matrices.

For the 203 unsymmetric matrices, COLAMD gives the best ordering for 77 matrices,

METIS for 40, and NESDIS for 92. COLAMD is 3.15 and 3.9 times faster than METIS and NESDIS, respectively (a median result). The median fill-in is 3% less with NESDIS than with METIS for all 203 matrices, and 2% for just the 126 matrices for which COLAMD does not find the best ordering. Of these 126 matrices, METIS finds a significantly better ordering than NESDIS for only 8 matrices, NESDIS for 38, and they tie (within ±5%) for 81 matrices.

4 Availability

In addition to appearing as a Collected Algorithm of the ACM, CHOLMOD is available as a built-in function in MATLAB version 7.2 or later, and online at http://www.cise.ufl.edu/research/sparse. It requires the ordering functions AMD, COLAMD, CAMD, and CCOLAMD, the dense matrix libraries LAPACK [2] and the BLAS [26, 14, 13], and optionally uses METIS for its nested dissection orderings. LAPACK and the BLAS are available at http://www.netlib.org. The Goto BLAS is available at http://www.tacc.utexas.edu/resources/software [20]. METIS can be found at http://glarus.dtc.umn/edu [25].

References

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):886–905, 1996.

[2] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, 3rd edition, 1999.

[3] C. C. Ashcraft and R. G. Grimes. SPOOLES: an object-oriented sparse matrix library. In *Proc. 1999 SIAM Conf. Parallel Processing for Scientific Computing*, Mar. 1999.

[4] R. F. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. J. Dongarra. The Matrix Market: A web resource for test matrix collections. In R. F. Boisvert, editor, *Quality of Numerical Software, Assessment and Enhancement*, pages 125–137. Chapman & Hall, London, 1997. (`http://math.nist.gov/MatrixMarket`).

[5] T. A. Davis. University of Florida sparse matrix collection. www.cise.ufl.edu/research/sparse, 1994. NA Digest, vol 92, no. 42.

[6] T. A. Davis. Algorithm 849: A concise sparse Cholesky factorization package. *ACM Trans. Math. Software*, 31(4):587–591, 2005.

[7] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2006.

[8] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. A column approximate minimum degree ordering algorithm. *ACM Trans. Math. Software*, 30(3):353–376, 2004.

[9] T. A. Davis and W. W. Hager. Modifying a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 20(3):606–627, 1999.

[10] T. A. Davis and W. W. Hager. Multiple-rank modifications of a sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 22:997–1013, 2001.

[11] T. A. Davis and W. W. Hager. Dynamic supernodes in sparse Cholesky update/downdate and triangular solves. *ACM Trans. Math. Software*, 2006. (submitted).

[12] F. Dobrian, G. K. Kumfert, and A. Pothen. The design of sparse direct solvers using object oriented techniques. In *Adv. in Software Tools in Sci. Computing*, pages 89–131. Springer-Verlag, 2000.

[13] J. J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling. A set of level-3 basic linear algebra subprograms. *ACM Trans. Math. Software*, 16(1):1–17, 1990.

[14] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson. An extended set of Fortran basic linear algebra subprograms. *ACM Trans. Math. Software*, 14:18–32, 1988.

[15] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. London: Oxford Univ. Press, 1986.

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

[17] J. R. Gilbert, X. S. Li, E. G. Ng, and B. W. Peyton. Computing row and column counts for sparse QR and LU factorization. *BIT*, 41(4):693–710, 2001.

[18] J. R. Gilbert, E. G. Ng, and B. W. Peyton. An efficient algorithm to compute row and column counts for sparse Cholesky factorization. *SIAM J. Matrix Anal. Appl.*, 15(4):1075–1091, 1994.

[19] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28(126):505–535, 1974.

[20] K. Goto and R. van de Geijn. On reducing TLB misses in matrix multiplication. TR-2002-55, Univ. Texas at Austin, Dept. of Computer Sciences, 2002.

[21] N. I. M. Gould, Y. Hu, and J. A. Scott. Complete results from a numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. Technical Report Internal report 2005-1 (revision 1), CCLRC, Rutherford Appleton Laboratory, 2005.

[22] N. I. M. Gould, Y. Hu, and J. A. Scott. A numerical evaluation of sparse direct solvers for the solution of large sparse, symmetric linear systems of equations. *ACM Trans. Math. Software*, 200x. (to appear).

[23] W. W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[24] P. Hénon, P. Ramet, and J. Roman. PaStiX: A high-performance parallel direct solver for sparse symmetric definite systems. *Parallel Computing*, 28(2):301–321, 2002.

[25] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20:359–392, 1998.

[26] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software*, 5:308–323, 1979.

[27] J. W. H. Liu. Modification of the minimum-degree algorithm by multiple elimination. *ACM Trans. Math. Software*, 11(2):141–153, 1985.

[28] J. W. H. Liu. The role of elimination trees in sparse factorization. *SIAM J. Matrix Anal. Appl.*, 11(1):134–172, 1990.

[29] E. G. Ng and B. W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM J. Sci. Comput.*, 14(5):1034–1056, 1993.

[30] F. Pellegrini, J. Roman, and P. R. Amestoy. Hybridizing nested dissection and halo approximate minimum degree for efficient sparse matrix ordering. *Concurrency: Pract. Exp.*, 12:68–84, 2000.

[31] E. Rothberg and S. C. Eisenstat. Node selection strategies for bottom-up sparse matrix orderings. *SIAM J. Matrix Anal. Appl.*, 19(3):682–695, 1998.

[32] E. Rothberg and A. Gupta. Efficient sparse matrix factorization on high-performance workstations - Exploiting the memory hierarchy. *ACM Trans. Math. Software*, 17(3):313–334, 1991.

[33] V. Rotkin and S. Toledo. The design and implementation of a new out-of-core sparse Cholesky factorization method. *ACM Trans. Math. Software*, 30(1):19–46, 2004.

[34] G. W. Stewart. The effects of rounding error on an algorithm for downdating a Cholesky factorization. *J. Inst. Math. Appl.*, 23:203–213, 1979.

[35] G. W. Stewart. *Matrix algorithms, Volume 1: Basic decompositions*. SIAM, Philadelphia, 1998.