# Multifrontal multithreaded rank-revealing sparse QR factorization

Timothy Davis[1]

Univ. of Florida, CISE Department
E301 CSE, 32605-6012, Gainesville, FL, USA
davis@cise.ufl.edu

**Abstract.** SuiteSparseQR is a sparse multifrontal QR factorization algorithm. Dense matrix methods within each frontal matrix enable the method to obtain high performance on multicore architectures. Parallelism across different frontal matrices is handled with Intel's Threading Building Blocks library. Rank-detection is performed within each frontal matrix using Heath's method, which does not require column pivoting. The resulting sparse QR factorization obtains a substantial fraction of the theoretical peak performance of a multicore computer.

**Keywords.** sparse matrix algorithms, QR factorization, multifrontal

Sparse QR factorization is one of the key direct methods for solving large sparse linear systems and least-squares problems. Typically, orthogonal transformations such as Givens rotations [1] or Householder reflections [2] are applied to $A$ (or a permuted matrix $AP$), resulting in the factorization $A = QR$ or $AP = QR$. The resulting factors can be used to solve a least-squares problem, to find the basic solution of an under-determined system, or to find a minimum 2-norm solution of an under-determined system.

The earliest sparse direct methods operated on $A$ one row or column at a time (see an overview in [3]). These methods are unable to reach a substantial fraction of the theoretical peak performance of modern computers because of their irregular access of memory. The row-merging method [4] introduced the idea that groups of rows could be handled all at once. This idea was fully realized in the sparse multifrontal QR factorization method, where the factorization of a large sparse matrix is performed in a sequence of small dense frontal matrices. It was first adapted to multifrontal sparse QR by [5,6].

Figure 1 gives an example of the multifrontal sparse QR factorization of a small sparse matrix $A$. The rows of $A$ have been sorted according to the column index of their leftmost nonzero entry. The factor $R$ is shown to the right, with each supernode of $R$ consisting of an adjacent set of rows with identical nonzero pattern. The horizontal lines in $A$ subdivide the rows according to the frontal matrix of $R$ into which they are first assembled. A dot (.) is shown in $A$ for an entry that becomes structurally nonzero in the corresponding frontal matrix. The column elimination tree is shown in the bottom right of the figure. The parent of $i$ is given by the smallest $j > i$ for which $r_{ij} \neq 0$. The supernodes are shown in the column elimination tree as rounded boxes.

Dagstuhl Seminar Proceedings 09061
Combinatorial Scientific Computing
http://drops.dagstuhl.de/opus/volltexte/2009/2078

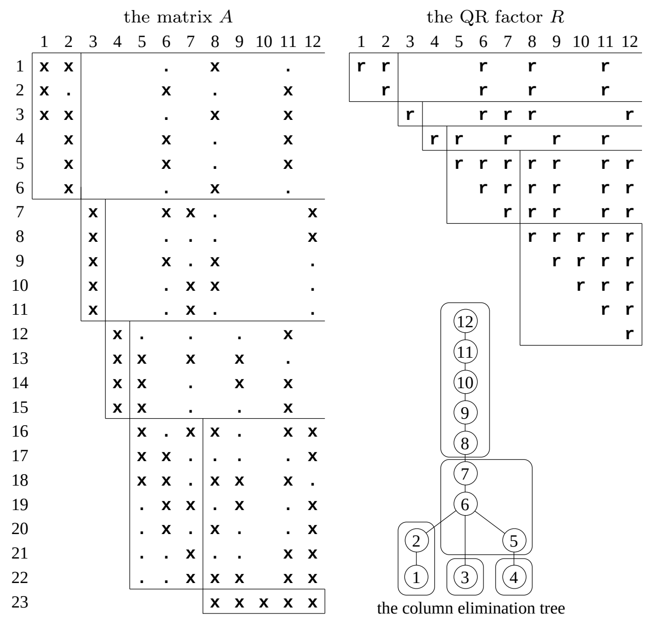


**Fig. 1.** A sparse matrix $A$, its factor $R$, and its column elimination tree

Consider front 4 and its three children in Figure 2. The entries in the $C$ blocks are given subscripts according to the block in which they reside. The pivotal columns are columns 5, 6, and 7. Rows 16 through 22 of $A$ have a leftmost nonzero entry in the pivotal columns of this front.

The contribution blocks of the three children and rows 16 through 22 of $A$ are assembled into the front; this is shown in the bottom left of Figure 2. The Householder QR factorization is shown to its right. A 4-by-4 upper triangular contribution block remains to be assembled by the parent of front 4.

Once the frontal matrix is assembled, the dense Householder QR factorization of the frontal matrix is found, using the DLARF* routines in LAPACK [7]. SuiteSparseQR also handles rank deficient matrices via Heath's method [8].

Two opportunities for parallelism are used. The first arises in the column elimination tree. In the example given in Figure 1, the first three frontal matrices can be factorized in parallel (one front for computing the first two rows of $R$, and the next two which are used to compute rows 3 and 4 of $R$). Using this level of parallelism requires explicit thread-based software in SuiteSparseQR. The second arises within each frontal matrix. The factorization of a frontal matrix relies on LAPACK, which in turn uses the Level-3 BLAS [9].

With a 2 million by 110 thousand sparse matrix, on a 16-core parallel computer, the algorithmic speedup over `x=A\b` is 375×, with an additional parallel speedup of 5.75×, representing a peak performance of 14 GFlops on a 70 GFlop-peak computer (cutting the time from 11 days to 7.3 minutes). On a single core, SPQR obtains a 2.5 GFlop peak, the same as LAPACK's dense QR factorization.

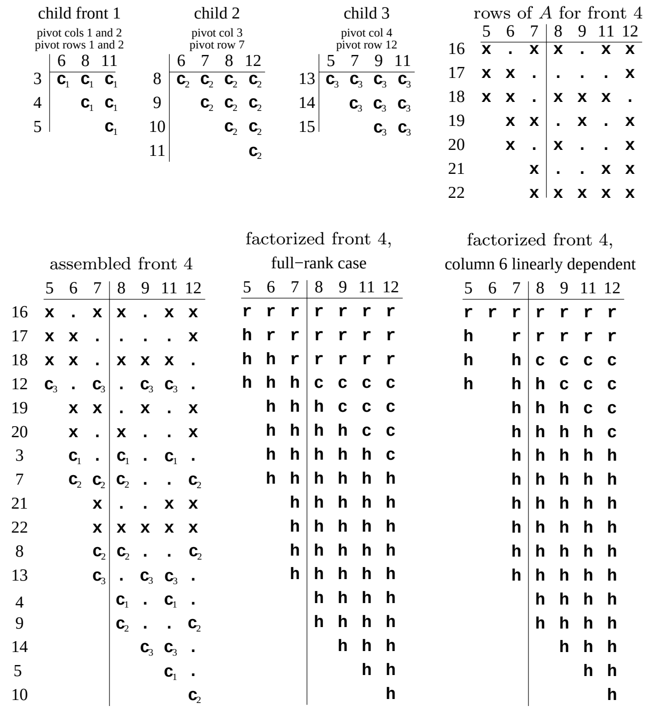


**Fig. 2.** Assembly and factorization of a frontal matrix with three children

## References


1. Givens, W.: Computation of plane unitary rotations transforming a general matrix to triangular form. SIAM J. Appl. Math. **6** (1958) 26–50
2. Householder, A.S.: Unitary triangularization of a nonsymmetric matrix. J. ACM **5** (1958) 339–342
3. Davis, T.A.: Direct Methods for Sparse Linear Systems. SIAM, Philadelphia, PA (2006)
4. Liu, J.W.H.: On general row merging schemes for sparse Givens transformations. SIAM J. Sci. Statist. Comput. **7** (1986) 1190–1211
5. Amestoy, P.R., Duff, I.S., Puglisi, C.: Multifrontal QR factorization in a multiprocessor environment. Numer. Linear Algebra Appl. **3** (1996) 275–300
6. Puglisi, C.: QR factorization of large sparse overdetermined and square matrices using a multifrontal method in a multiprocessor environment. PhD thesis, Institut National Polytechnique de Toulouse (1993) CERFACS report TH/PA/93/33.
7. Anderson, E., Bai, Z., Bischof, C.H., Blackford, S., Demmel, J.W., Dongarra, J.J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.C.: LAPACK Users' Guide. 3rd edn. SIAM, Philadelphia, PA (1999)
8. Heath, M.T.: Some extensions of an algorithm for sparse linear least squares problems. SIAM J. Sci. Statist. Comput. **3** (1982) 223–237
9. Dongarra, J.J., Du Croz, J.J., Duff, I.S., Hammarling, S.: A set of Level 3 Basic Linear Algebra Subprograms. ACM Trans. Math. Softw. **16** (1990) 1–17