Algorithm 8xx: UMFPACK V4.1, an unsymmetric-pattern multifrontal method

Timothy A. Davis*

May 6, 2003

Abstract

An ANSI C code for sparse LU factorization is presented that combines a column pre-ordering strategy with a right-looking unsymmetric-pattern multifrontal numerical factorization. The pre-ordering and symbolic analysis phase computes an upper bound on fill-in, work, and memory usage during the subsequent numerical factorization. User-callable routines are provided for ordering and analyzing a sparse matrix, computing the numerical factorization, solving a system with the LU factors, transposing and permuting a sparse matrix, and converting between sparse matrix representations. The simple user interface shields the user from the details of the complex sparse factorization data structures by returning simple handles to opaque objects. Additional user-callable routines are provided for printing and extracting the contents of these opaque objects. An even simpler way to use the package is through its MATLAB interface. UMFPACK is incorporated as a built-in operator in MATLAB 6.5 as `x = A\b` when `A` is sparse and unsymmetric.

Categories and Subject Descriptors: G.1.3 [Numerical Analysis]: Numerical Linear Algebra – *linear systems (direct methods), sparse and very large systems* G.4 [Mathematics of Computing]: Mathematical Software – *algorithm analysis, efficiency*

General terms: Algorithms, Experimentation, Performance.

Keywords: sparse nonsymmetric matrices, linear equations, multifrontal method, ordering methods.

1 Overview

UMFPACK Version 4.1 is a code written in ANSI C for the direct solution of systems of linear equations, $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is sparse and unsymmetric. The matrix $\mathbf{PAQ}$, $\mathbf{PRAQ}$, or $\mathbf{PR}^{-1}\mathbf{AQ}$ is factorized into the product $\mathbf{LU}$. The column ordering $\mathbf{Q}$ is selected to give a

*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. This work was supported by the National Science Foundation, under grants ASC-9111263, DMS-9223088, and DMS-0203270. Portions of the work were done while on sabbatical at Stanford University and Lawrence Berkeley National Laboratory (with funding from Stanford University and the SciDAC program).

good a priori upper bound on fill-in and then refined during numerical factorization (while preserving the upper bound on fill-in). The row ordering $\mathbf{P}$ is selected during numerical factorization to maintain numerical stability and to preserve sparsity. The diagonal matrix $\mathbf{R}$ scales the rows of the matrix $\mathbf{A}$. The method is a combination of a column pre-ordering and symbolic analysis phase based on COLAMD or AMD and a numerical factorization phase based on a modification of the unsymmetric-pattern multifrontal method (UMFPACK Version 2, or MA38, which does not have a pre-ordering and symbolic analysis phase). The methods used by UMFPACK V4.1 are discussed in a companion paper [1]. The sparse matrix $\mathbf{A}$ factorized by UMFPACK can be real or complex, square or rectangular, and singular or non-singular (or any combination).

UMFPACK analyzes the matrix and automatically selects between three ordering strategies, described below.

- *unsymmetric*: A column pre-ordering is computed by a modified version of COLAMD. The method finds a permutation $\mathbf{Q}$ that limits the fill-in for any subsequent choice of $\mathbf{P}$ via partial pivoting. This modified version of COLAMD also computes the column elimination tree and a depth-first post-ordering of the tree. During factorization, the column pre-ordering can be modified. Columns within a single supercolumn can be reshuffled, to reduce fill-in. Threshold partial pivoting is used with no preference given to the diagonal entry. Within a given pivot column j, an entry a_{ij} can be chosen if $|a_{ij}| \geq 0.1 \max |a_{*j}|$. Among those numerically acceptable entries, the sparsest row i is chosen as the pivot row.

- *symmetric*: The column ordering is computed from AMD applied to the pattern of $\mathbf{A}+\mathbf{A}^\mathsf{T}$, followed by a post-ordering of the supernodal elimination tree. No modification of the column pre-ordering is made during numerical factorization. Threshold partial pivoting is used, with a strong preference given to the diagonal entry. The diagonal entry is chosen if $a_{jj} \geq 0.001 \max |a_{*j}|$. Otherwise, a sparse row is selected, using the same method used by the unsymmetric strategy (with a relative threshold of 0.1).

- *2-by-2*: A row permutation $\mathbf{P}_2$ is found which attempts to reduce the number of small diagonal entries of $\mathbf{P}_2\mathbf{A}$. If a_{ii} is numerically small, the method attempts to swap two rows i and j, such that both a_{ij} and a_{ji} are large. Once these rows are swapped they remain in place. This does not guarantee a zero-free diagonal, but it does tend to preserve the symmetry of the nonzero pattern. Next, the symmetric strategy (see above) is applied to the matrix $\mathbf{P}_2\mathbf{A}$.

2 MATLAB Interface

The MATLAB interface to UMFPACK provides a replacement for MATLAB's LU routine, and the forward slash and backslash matrix operators. It is typically much faster than the built-in routines in MATLAB Version 6.0, uses less memory, and returns sparser LU factors. MATLAB 6.5 includes UMFPACK Version 4.0 as a built-in routine. The interface provided here also allows access to UMFPACK's pre-ordering and symbolic analysis phase.

3 ANSI C Interface

The ANSI C UMFPACK library consists of 31 user-callable routines and one include file. Twenty-seven of the routines come in four versions: real or complex (both double precision), and `int` or `long` integers. Only the `double` / `int` version is described here; the other versions are analogous. UMFPACK requires the BLAS (which perform dense matrix operations) for best performance, but can be used without the BLAS. Five primary UMFPACK routines are required to solve $\mathbf{Ax} = \mathbf{b}$:

- `umfpack_di_symbolic`: Pre-orders the columns of $\mathbf{A}$, finds the supernodal column elimination tree, and post-orders the tree. Returns an opaque `Symbolic` object as a `void *` pointer that can be used by `umfpack_numeric` to factorize $\mathbf{A}$ or any other matrix with the same nonzero pattern as $\mathbf{A}$. Computes upper bounds on the nonzeros in $\mathbf{L}$ and $\mathbf{U}$, the floating-point operations required, and the memory usage of `umfpack_di_numeric`.
- `umfpack_di_numeric`: Numerically factorizes a sparse matrix $\mathbf{PAQ}$, $\mathbf{PRAQ}$, or $\mathbf{PR}^{-1}\mathbf{AQ}$ into the product $\mathbf{LU}$, using the `Symbolic` object. Returns an opaque `Numeric` object as a `void *` pointer.
- `umfpack_di_solve`: Solves a sparse linear system ($\mathbf{Ax} = \mathbf{b}$, $\mathbf{A}^{\mathsf{T}}\mathbf{x} = \mathbf{b}$, or systems involving just $\mathbf{L}$ or $\mathbf{U}$), using the `Numeric` object. Performs iterative refinement with sparse backward error.
- `umfpack_free_symbolic`: Frees the `Symbolic` object.
- `umfpack_free_numeric`: Frees the `Numeric` object.

The matrix $\mathbf{A}$ is represented in compressed column form:

```
int Ap [n+1] ;
int Ai [nz] ;
double Ax [nz] ;
```

The row indices and numerical values of entries in column `j` are stored in `Ai[Ap[j] ... Ap[j+1]-1]` and `Ax[Ap[j] ... Ap[j+1]-1]`, respectively. This simple program illustrates the basic usage of UMFPACK:

```
#include <stdio.h>
#include "umfpack.h"

int    n = 5 ;
int    Ap [ ] = {0, 2, 5, 9, 10, 12} ;
int    Ai [ ] = { 0,  1,  0,   2,  4,  1,  2,  3,   4,  2,  1,  4} ;
double Ax [ ] = {2., 3., 3., -1., 4., 4., -3., 1., 2., 2., 6., 1.} ;
double b [ ] = {8., 45., -3., 3., 19.} ;
double x [5] ;

int main (void)
{
    double *null = (double *) NULL ;
```

```
    int i ;
    void *Symbolic, *Numeric ;
    (void) umfpack_di_symbolic (n, n, Ap, Ai, Ax, &Symbolic, null, null) ;
    (void) umfpack_di_numeric (Ap, Ai, Ax, Symbolic, &Numeric, null, null) ;
    umfpack_di_free_symbolic (&Symbolic) ;
    (void) umfpack_di_solve (UMFPACK_A, Ap, Ai, Ax, x, b, Numeric, null, null) ;
    umfpack_di_free_numeric (&Numeric) ;
    for (i = 0 ; i < n ; i++) printf ("x [%d] = %g\n", i, x [i]) ;
    return (0) ;
}
```

The `Ap`, `Ai`, and `Ax` arrays represent the matrix

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 0 & 0 & 0 \\ 3 & 0 & 4 & 0 & 6 \\ 0 & -1 & -3 & 2 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 4 & 2 & 0 & 1 \end{bmatrix}.$$

and the solution is $\mathbf{x} = [1\,2\,3\,4\,5]^{\mathsf{T}}$. Additional routines are provided for:

- Changing default parameter settings, and for providing a different column pre-ordering.
- Converting a matrix in triplet form to compressed column form, and visa versa. The triplet form is a simpler data structure for the user to manipulate. It consists of three arrays that hold the row index, column index, and numerical value of each entry in matrix.
- Transposing and optionally permuting a compressed column form matrix.
- Getting the contents of the opaque `Symbolic` and `Numeric` objects, saving them to a file, and loading them from a file.
- Printing and verifying control parameters, statistics, sparse matrices, `Symbolic` and `Numeric` objects, permutation vectors, and dense vectors.

In addition to appearing as a Collected Algorithm of the ACM, UMFPACK is available at http://www.cise.ufl.edu/research/sparse. The package includes a user guide that provides full details on how to install the package, how to use the MATLAB interface, and how to use the ANSI C interface. A basic Fortran interface is also provided.

References

[1] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 2003 (under submission). Also TR-03-006 at www.cise.ufl.edu/tech-reports.