Algorithm 8xx: UMFPACK V3.2, an unsymmetric-pattern multifrontal method with a column pre-ordering strategy *

Timothy A. Davis †

January 2, 2002

Technical report TR-02-002. Department of Computer and Information Science and Engineering, University of Florida.

Abstract

An ANSI C code for sparse LU factorization is presented that combines a left-looking 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. A future version of MATLAB will incorporate UMFPACK as its default sparse matrix factorization method.

*This work was supported by the National Science Foundation, under grants DMS-9504974 and DMS-9803599.

†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.

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 3.2 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 $\mathbf{PAQ}$ is factorized into the product $\mathbf{LU}$. The column ordering $\mathbf{Q}$ is selected to give a 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 method is a combination of a column pre-ordering and symbolic analysis phase based on COLAMD [5, 6, 9], and a numerical factorization phase based on a modification of the unsymmetric-pattern multifrontal method (UMFPACK Version 2, or MA38 [3, 4], which does not have a pre-ordering and symbolic analysis phase). The methods used by UMFPACK V3.2 are discussed in a companion paper [2]. The sparse matrix $\mathbf{A}$ factorized by UMFPACK must be real, square, and non-singular.

2 MATLAB Interface

The MATLAB interface to UMFPACK provides a replacement for MATLAB's LU routine, and the forward slash and backslash matrix operators (when $\mathbf{A}$ is sparse, square, real, and non-singular). It is typically much faster than the built-in routines in MATLAB Version 6.0, uses less memory, and returns sparser LU factors. A future version of MATLAB will include UMFPACK as a built-in routine. Access is also provided to UMFPACK's symbolic analysis phase (a version of the COLAMD ordering routine that also computes information about each frontal matrix and returns the supernodal column elimination tree). The COLAMD ordering is suitable for

Table 1: Using UMFPACK's MATLAB interface

Function	Using UMFPACK	MATLAB 6.0 equivalent
Solve $\mathbf{Ax} = \mathbf{b}$.	`x = umfpack (A,'\',b) ;`	`x = A \ b ;`

Solve $\mathbf{Ax} = \mathbf{b}$ using a different column pre-ordering.	`S = spones (A) ;`
	`Q = symamd (S+S') ;`
	`x = umfpack (A,Q,'\',b) ;`
		`spparms ('autommd',0) ;`
		`S = spones (A) ;`
		`Q = symamd (S+S') ;`
		`x = A (:,Q) \ b ;`
		`x (Q) = x ;`
		`spparms ('autommd',1) ;`

Solve $\mathbf{A}^{\mathsf{T}}\mathbf{x}^{\mathsf{T}} = \mathbf{b}^{\mathsf{T}}$.	`x = umfpack (b,'/',A) ;`	`x = b / A ;`

Factorize $\mathbf{A}$, solve $\mathbf{Ax} = \mathbf{b}$.	`[L,U,P,Q] = umfpack (A) ;`
	`x = U \ (L \ (b (P))) ;`
	`x (Q) = x ;`
		`Q = colamd (A) ;`
		`[L,U,P] = lu (A (:,Q)) ;`
		`x = U \ (L \ (P*b)) ;`
		`x (Q) = x ;`

most matrices, but not all, so a different input ordering can be given. Table 1 summarizes the most common uses of the UMFPACK mexFunction, and its approximate equivalent in built-in MATLAB 6.0 routines. UMFPACK returns `P` and `Q` as permutation vectors, so that `A(P,Q)` is equal to `L*U`. In the MATLAB equivalent, `P` is a permutation matrix, so that `P*A(:,Q)` is equal to `L*U`.

3 ANSI C Interface

The ANSI C UMFPACK library consists of 24 user-callable routines and one include file. Twenty-three of the routines come in dual versions, one that uses `int`'s for integers, and another that uses `long`'s instead. The latter is required if you wish to solve matrices requiring more than 4GB of memory, assuming you can make use of the LP64 model of execution (in which `long`'s and pointers are both 64 bits in length). Only the `int` version is described here; the `long` version is analogous. Both versions use double pre-

cision floating-point arithmetic. The BLAS [7] is optional, but provides the best performance. Five UMFPACK routines are required to solve $\mathbf{Ax} = \mathbf{b}$:

- `umfpack_symbolic`: Pre-orders the columns of $\mathbf{A}$ to reduce fill-in based on its sparsity pattern only, 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_numeric`.

- `umfpack_numeric`: Numerically factorizes a sparse matrix into $\mathbf{PAQ} = \mathbf{LU}$, using the `Symbolic` object. Returns an opaque `Numeric` object as a `void *` pointer.

- `umfpack_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. Includes iterative refinement with sparse backward error [1].

- `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 (int argc, char **argv)
{
    double *Control = (double *) NULL, *Info = (double *) NULL ;
    int i ;
    void *Symbolic, *Numeric ;
    umfpack_symbolic (n, Ap, Ai, &Symbolic, Control, Info) ;
    umfpack_numeric (Ap, Ai, Ax, Symbolic, &Numeric, Control, Info) ;
    umfpack_free_symbolic (&Symbolic) ;
    umfpack_solve ("Ax=b", Ap, Ai, Ax, x, b, Numeric, Control, Info) ;
    umfpack_free_numeric (&Numeric) ;
    for (i = 0 ; i < n ; i++) printf ("x [%d] = %g\n", i, x [i]) ;
}
```

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.
- Transposing and optionally permuting a compressed column form matrix [8].
- Getting the contents of the opaque `Symbolic` and `Numeric` objects.
- Printing and verifying the control parameters, return status, statistics, sparse matrices, `Symbolic` and `Numeric` objects, permutation vectors, and dense vectors.

UMFPACK is available at 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.

References

[1] M. Arioli, J. W. Demmel, and I. S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Applic.*, 10:165–190, 1989.

[2] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. Technical Report TR-02-001, Univ. of Florida, CISE Dept., Gainesville, FL, January 2002. (www.cise.ufl.edu/tech-reports. Submitted to *ACM Trans. Math. Softw.)*.

[3] T. A. Davis and I. S. Duff. An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. Matrix Anal. Applic.*, 18(1):140–158, 1997.

[4] T. A. Davis and I. S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Softw.*, 25(1):1–19, 1999.

[5] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. Algorithm 8xx: COLAMD, a column approximate minimum degree ordering algorithm. Technical Report TR-00-006, Univ. of Florida, CISE Dept., Gainesville, FL, October 2000. (www.cise.ufl.edu/tech-reports. Submitted to *ACM Trans. Math. Softw.)*.

[6] T. A. Davis, J. R. Gilbert, S. I. Larimore, and E. G. Ng. A column approximate minimum degree ordering algorithm. Technical Report TR-00-005, Univ. of Florida, CISE Dept., Gainesville, FL, October 2000. (www.cise.ufl.edu/tech-reports. Submitted to *ACM Trans. Math. Softw.)*.

[7] J. J. Dongarra, J. J. Du Croz, I. S. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Softw.*, 16:1–17, 1990.

[8] F. G. Gustavson. Two fast algorithms for sparse matrices: Multiplication and permuted transposition. *ACM Trans. Math. Softw.*, 4(3):250–269, Sept. 1978.

[9] S. I. Larimore. An approximate minimum degree column ordering algorithm. Technical Report TR-98-016, Univ. of Florida, Gainesville, FL, 1998. www.cise.ufl.edu/tech-reports.