Sparse Matrix Research and Software Development at the University of Florida

Tim Davis
with P. Amestoy, Y. Chen, I. Duff, J. Gilbert, W. Hager, S. Larimore, E. Ng, S. Rajamanickam.

with support from NSF, DOE (Sandia)

July 11, 2008

Some background

- PhD, Univ. of Illinois (1989)
- CERFACS postdoc (1989-1990, with Iain Duff)
- Professor, Univ. of Florida (1991 to date)
- sparse matrix software development:
  - UMFPACK: sparse multifrontal LU
  - AMD, COLAMD: ordering methods
  - CHOLMOD: sparse Cholesky, update/downdate
  - KLU: sparse LU for circuit simulation
  - CSparse: a concise sparse matrix package
  - UF Sparse Matrix Collection
  - codes in use in: MATLAB, Mathematica, NASTRAN, Cadence, IBM, Octave, R, ARPACK, Trilinos, Xyce, Debian / Ubuntu Linux, Computational EM Works, deal.II, SciPi, dozens of others

A personal viewpoint

- other examples of robust academic codes: LINPACK, EISPACK, LAPACK, BLAS, FFTW, ARPACK, FDLIB, MINOS, PARDISO, ...
- and codes that once were: MATLAB, CPLEX, ...
- Why should an academic write robust, commercial-quality code?
  - impact
  - software is publishable (ACM TOMS, books, ...)
  - royalty income
  - consulting income

Cultural gap

Goal	Academia	Industry/Govt
new theory, algo, data structures	*	
publishable papers	*	
meets a theoretical need	*	
advances state-of-the-art (theory)	*	
advances state-of-the-art (practice)	*	*
competitive performance	*	*
meets a practical need	*	*
software prototypes	goal	means
usable in production code	specific	general
software robustness		*
software generality		*
well-thought out API		*

Software development philosophies

- comments $>$ code
- well-designed API
- robust error handling
- scaffolding code: use it, test it, post it.
- asserts: executable comments
- Valgrind (or Purify)
- it ain't code until it's documented
- it ain't code unless it's maintained
- 100% statement coverage
- MATLAB as prototype, test engine
  (and end target, depending on the project)

Payoff: reliable, usable code

code	date	# lines main code	# lines test code	bugs affecting MATLAB	total # bugs	bugs/ 1k lines
UMFPACK	1994	41,777	13,237	2	6	0.14
AMD	1996	4,395	685	0	0	0
COLAMD	1998	4,311	1,486	0	1	0.15
LDL	2003	1,157	539	-	0	0
CHOLMOD	2005	47,487	13,542	0	8	0.17
CAMD	2005	4,333	590	-	1	0.23
CCOLAMD	2006	6,481	373	-	0	0
CSparse	2006	2,229	415	0	1	0.45
KLU	2007	10,223	2,059	-	0	0
SF/SSMULT	2007	5,466	693	0	0	0
total		127,859	33,619	2	17	0.13

Payoff: reliable, usable code

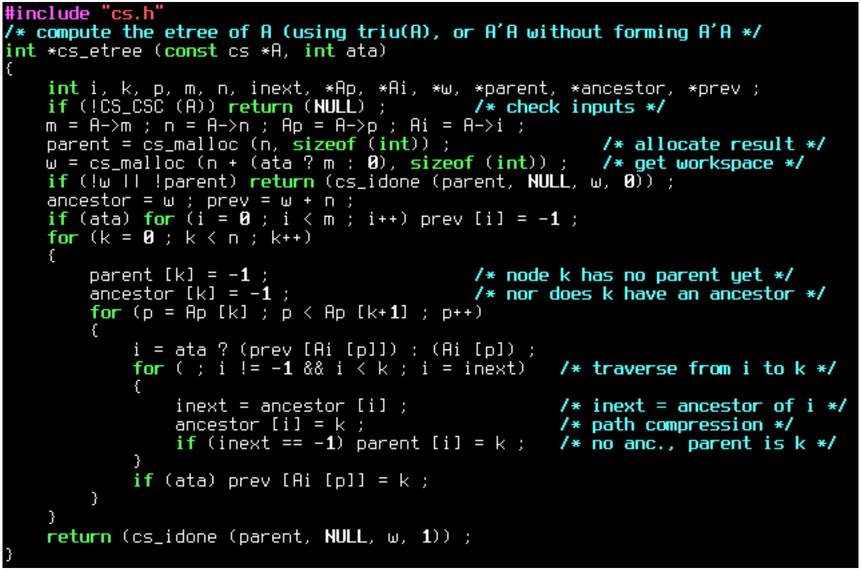
- typical high-quality production code (CMM5): 1 bug/Kloc
- NASA Software Assurance Technology Center:
  - obtained 0.1 bug/Kloc
  - “unachievable for real-world codes”
- UF sparse matrix codes
  - 128K lines total, 17 bugs (0.13 bug/Kloc)
  - 105K lines code in MATLAB, 2 bugs (0.02 bug/Kloc)
  - modest # of code-related publications:
    15 journal articles, 1 book

Licensing

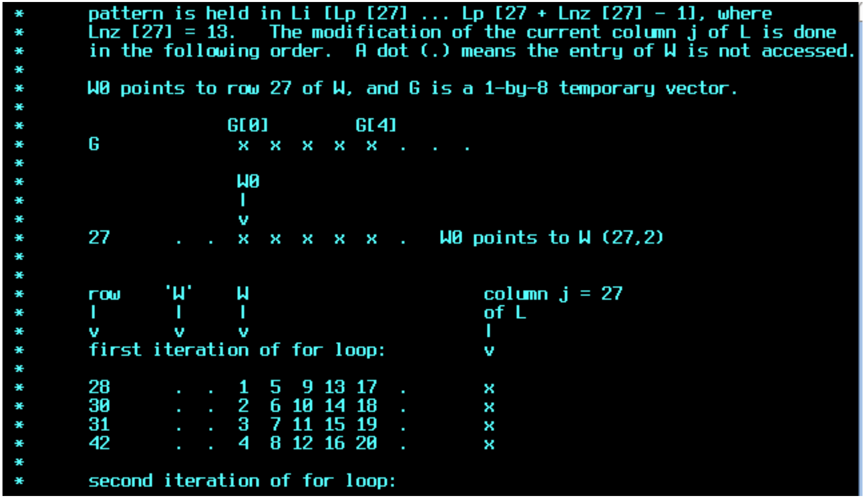
- open-source flavors
  - BSD-like
  - GNU LGPL (“non-viral GNU”)
  - GNU GPL (“viral GNU”)
- dual-source
  - provide both open-source and commercial licenses
  - works well in conjunction with consulting
- closed-source

Software as art

- academic payoff aside: do what you love ...
- software as art
  - beauty of logic
  - beauty of form



```c
#include "cs.h"
/* compute the etree of A (using triu(A), or A'A without forming A'A */
int *cs_etree (const cs *A, int ata)
{
    int i, k, p, m, n, inext, *Ap, *Ai, *w, *parent, *ancestor, *prev ;
    if (!CS_CSC (A)) return (NULL) ;           /* check inputs */
    m = A->m ; n = A->n ; Ap = A->p ; Ai = A->i ;
    parent = cs_malloc (n, sizeof (int)) ;              /* allocate result */
    w = cs_malloc (n + (ata ? m : 0), sizeof (int)) ;   /* get workspace */
    if (!w || !parent) return (cs_idone (parent, NULL, w, 0)) ;
    ancestor = w ; prev = w + n ;
    if (ata) for (i = 0 ; i < m ; i++) prev [i] = -1 ;
    for (k = 0 ; k < n ; k++)
    {
        parent [k] = -1 ;                      /* node k has no parent yet */
        ancestor [k] = -1 ;                    /* nor does k have an ancestor */
        for (p = Ap [k] ; p < Ap [k+1] ; p++)
        {
            i = ata ? (prev [Ai [p]]) : (Ai [p]) ;
            for ( ; i != -1 && i < k ; i = inext)   /* traverse from i to k */
            {
                inext = ancestor [i] ;              /* inext = ancestor of i */
                ancestor [i] = k ;                  /* path compression */
                if (inext == -1) parent [i] = k ;   /* no anc., parent is k */
            }
            if (ata) prev [Ai [p]] = k ;
        }
    }
    return (cs_idone (parent, NULL, w, 1)) ;
}
```



```
*       pattern is held in Li [Lp [27] ... Lp [27 + Lnz [27] - 1], where
*       Lnz [27] = 13.    The modification of the current column j of L is done
*       in the following order.  A dot (.) means the entry of W is not accessed.
*
*       W0 points to row 27 of W, and G is a 1-by-8 temporary vector.
*
*                    G[0]        G[4]
*       G             x   x   x   x   x   .   .   .
*
*                     W0
*                     |
*                     v
*       27        .   .   x   x   x   x   x   .   W0 points to W (27,2)
*
*
*       row     'W'     W                           column j = 27
*       |        |      |                           of L
*       v        v      v                           |
*       first iteration of for loop:                v
*
*       28        .   .   1   5   9  13  17   .     x
*       30        .   .   2   6  10  14  18   .     x
*       31        .   .   3   7  11  15  19   .     x
*       42        .   .   4   8  12  16  20   .     x
*
*       second iteration of for loop:
```

Software as poetry

Software as poetry

Sea Fever, by Masefield (1902)

I must go down to the seas again, to the lonely sea and the sky,
And all I ask is a tall ship and a star to steer her by,
And the wheel's kick and the wind's song
    and the white sail's shaking,
And a grey mist on the sea's face and a grey dawn breaking.

Software as poetry

Sea Fever, by Masefield (1902)

I must go down to the seas again, to the lonely sea and the sky,
And all I ask is a tall ship and a star to steer her by,
And the wheel's kick and the wind's song
    and the white sail's shaking,
And a grey mist on the sea's face and a grey dawn breaking.

C Fever, by T.D. (2008)

I must go code in the C again, to the lonely C and VI,
And all I ask is a Linux box and a mouse to steer her by,
And the while's break and the if-then
    and the valgrind's shaking,
And a dash O so the C's fast and a switch case breaking.

Software as poetry

Sea Fever

I must go down to the seas again, for the call of the running tide
Is a wild call and a clear call that may not be denied;
And all I ask is a windy day with the white clouds flying,
And the flung spray and the blown spume, and the sea-gulls crying.

Software as poetry

Sea Fever

I must go down to the seas again, for the call of the running tide
Is a wild call and a clear call that may not be denied;
And all I ask is a windy day with the white clouds flying,
And the flung spray and the blown spume, and the sea-gulls crying.

C Fever

I must go code in the C again, for the call of recursive code
Is a wild call and a clear call that runs and won't be slowed;
And all I ask is a tall latte with the white foam frothing,
And no flung err and no blown stack, and the C goes flying.

Software as poetry

Sea Fever

I must go down to the seas again, to the vagrant gypsy life,
To the gull's way and the whale's way
where the wind's like a whetted knife;
And all I ask is a merry yarn from a laughing fellow-rover,
And quiet sleep and a sweet dream when the long trick's over.

Software as poetry

Sea Fever

I must go down to the seas again, to the vagrant gypsy life,
To the gull's way and the whale's way
        where the wind's like a whetted knife;
And all I ask is a merry yarn from a laughing fellow-rover,
And quiet sleep and a sweet dream when the long trick's over.

C Fever

I must go code in the C again, to the lonely code frontier,
Where no goto's in the while's way,
        where the logic is sharp and clear;
And all I ask is a #define from a fellow C-programmer,
And quiet sleep when a clean doc's in the LaTeX grammar.

Summary

- Robust software development very compatiable with an academic career
- Academic / Industrial cultural gap can be bridged
- Good code is a useful research result
- Dual-source (multi-licensing):
  - open-source (GNU), with commercial licenses available
  - allows for wide use
  - allows for licensing and consulting income