

CIS 6930/4930: Sparse matrix algo.: Project 1

Timothy A. Davis
Computer and Information Science and Engineering Department,
University of Florida. email: davis@cise.ufl.edu.
<http://www.cise.ufl.edu/~davis>.

Fall 2007

1 Policies

Assigned Friday, Aug 31. Due Friday, Sept 7.

Turn in any new or modified files, via email, but zip them up or tar them into a single archive, first. Be sure to include a description where the files go. I prefer that you have a directory CSpase that mimics the structure of the whole CSpase. Include only the files that you have modified or added. Modify `cs_make` as necessary to not compile your mexFunctions. Use CSpase Version 2.2.0, or CXSpase Version 2.2.0 if you use a 64-bit computer.

Note that for each project, it is **vital** that you start with a fresh copy of CSpase (unless instructed otherwise). Otherwise, your code may fail when I try to compile it (if it relies on code from an earlier project, for example).

You may not look at any other student's code.

2 Problem (A)

Do Problem 2.1 in the book.

3 Problem (B)

Write a function `cs_taxpy` which computes $\mathbf{z}=\mathbf{A}\mathbf{x}+\mathbf{y}$ where \mathbf{x} , \mathbf{y} , and \mathbf{z} are dense vectors, and \mathbf{A} is a sparse matrix in triplet form. Your code should pass the following test with errors around $1e-16$ or so:

```

problems = [449 262] ;
nprobs = length (problems) ;

for k = 1:nprobs

    Problem = UFget (problems (k))
    A = Problem.A ;
    [Ai Aj Ax] = find (A) ;
    n = size (A,1) ;
    y = (1:n)' ;
    x = (n:-1:1)' ;
    z1 = A*x + y ;
    z2 = cs_gaxpy (A, x, y) ;
    err1 = norm (z1 - z2) / norm (z1)

    try

        Ai = int32 (Ai) ;
        Aj = int32 (Aj) ;
        z3 = cs_taxpy (Ai, Aj, Ax, x, y) ;
        err2 = norm (z1 - z3) / norm (z1)

        nz = length (Ax) ;
        p = randperm (nz) ;

        Ai = Ai (p) ;
        Aj = Aj (p) ;
        Ax = Ax (p) ;
        z4 = cs_taxpy (Ai, Aj, Ax, x, y) ;
        err3 = norm (z1 - z4) / norm (z1)

    catch
        lasterr
        fprintf ('something failed ...\n') ;
    end
end
end

```

Note that A_j and A_i are int32 vectors. Use statements in your new `cs_taxy` mexFunction such as:

```
int *Aj, *Ai ;
double *Ax ;
Ai = (int *) mxGetData (pargin [0]) ;
Aj = (int *) mxGetData (pargin [1]) ;
Ax = (double *) mxGetData (pargin [2]) ;
```

To access them in your `cs_taxy` mexFunction. Note that you should use `int64` in MATLAB and `mxSignedInteger` in C if you have a 64-bit MATLAB and CXSparse.

Next, use a large matrix, and compare the run time for computing z_1 , z_2 , z_3 , and z_4 . Matrix id 939 (ND/ND24k) is a good test. Try matrix 916 (vanHeukelum/cage15) if you have enough memory. Which code is fastest? Use `tic` and `toc`. Be sure to run a mexFunction at least once before timing it, since the first time it runs it has to be loaded into memory, which takes additional time that should not be counted.

Finally, modify `cs_gaxy` to improve its performance, as much as you can. Now which one is fastest?