Calculating the rank of sparse matrices using spnrank

Leslie Foster
Department of Mathematics
San Jose State University
4/18/2009

We briefly discuss the basis for spnrank and its use. In the near future these comments will be replaced by a paper discussing spnrank in more detail.

Note that the ideas used in spnrank are well known – see the references. However to our knowledge these ideas have not been implemented in Matlab code directed toward finding the rank of sparse matrices.

The location http://www.math.sjsu.edu/singular/matrices/software/SJsingular/spnrank.m contains the spnrank code.

Mathematical basis:

The algorithm uses Sylvester's inertial theorem:
For a Hermitian matrix C if B = F * C * F'
for an invertible matrix F, then B and C have the same
number of eigenvalues greater than zero.
We apply this result to the Hermitian matrix
C = B - tol * I where B = [0 A ; A' 0] and use Matlab's
ldl decomposition: [L,D,P]=ldl(C). With this transformation

$$C = P * L * D * L' * P'$$

By Sylvester's theorem C and D have the same number of positive eigenvalues. This is also the number of eigenvalues of B greater than tol which is equal to the number of singular values of A greater than tol.

Mathematically, in exact arithmetic, SPNRANK will correctly determine the rank of A. In computer arithmetic SPNRANK correctly determines the rank of the calculated LDL factorization, which may have computer arithmetic errors. However, in computer arithmetic we have (see p. 218 of Accuracy and Stability of Numerical Algorithms, 2nd ed. by Higham)

$$C + E = P * L * D * L' * P'$$

where P, L and D are the calculated factors and where

$$|E| <= p(m+n) (|C| + P |L||D||L'| P') eps + O(eps^2). \quad \text{(eqn 1)}$$

Here eps is relative machine precision and p(x) is a linear polynomial in x. Since E is of magnitude proportional to eps, in computer arithmetic SPNRANK will correctly determine the rank of A except when tol is very close (in an interval of magnitude proportional to eps) to a singular value of A.

If tol is O(norm(A)*eps) and A is singular then SPNRANK may not determine the correct numerical rank since

tol may be close to the zero singular values of A.
Therefore one should select tol larger than some small
value. For example if A is m by n:

tol >= max(m,n)* eps * norm(A)

appears to work well. Note that Matlab's RANK also can be
inaccurate of its tolerance is chosen too small.

As discussed below, SVALS and SVALS_ERR can be used to
provide an indication that tol should be changed.

The error bounds in SVALS_ERR are based on Theorem 5.5, p. 205,
of Applied Numerical Linear Algebra by James Demmel. Calculation
of the bounds directly uses C and does not use the calculated
LDL factorization. Therefore the accuracy of the calculated
error bounds is not affected by the potential growth in errors
from the factor |L| |D| |L'| in (eqn 1) above. However, as with
any computation in floating point arithmetic, potentially there
are errors in the calculations and, to be precise, it is not
guaranteed that A has a singular value in

[SVALS(i) - SVALS_ERR(i), SVALS(i) + SVALS_ERR(i)]

but it is guaranteed that A+E does where

||E|| <= q(m,n) ||A|| eps (eqn 2)

and q(m,n) is a lower order polynomial in m and n. If one
characterizes the error in A using norms, (eqn 2) is the best one
can expect of a floating point computation involving matrices.

Use of the code:

SPNRANK Numerical rank of dense or sparse matrices.
SPNRANK(A) provides an estimate of the number of linearly
independent rows or columns of a matrix A.
SPNRANK(A,TOL) is the number of singular values of A
that are larger than TOL.
SPNRANK(A) uses the default TOL = max(size(A)) * eps(norm of A), where
the norm of A is estimated using NORMEST_ERR, a minor modification of
Matlab's NORMEST.
Note: SPNRANK returns -1 for NRANK if the algorithm fails.

SPNRANK works for real or complex full matrices in Matlab 7.3 or higher
and for real (but not complex) sparse matrices in Matlab 7.5 or higher.

[NRANK, SVALS] = SPNRANK(A, [], NSVALS), for the default tolerance, or
[NRANK, SVALS] = SPNRANK(A, TOL, NSVALS), for a user supplied TOL,
will return estimates of the NSVALS singular values of A closest to
TOL that are larger than TOL and the NSVALS singular values of A
closest to TOL that are smaller than TOL. If there are not NSVALS
singular values smaller or larger than TOL then as many as exist are
returned. The entries in SVALS will be ordered largest to smallest.
If NSVALS is a vector with two components then SPNRANK returns NSVALS(1)
singular values greater than TOL and NSVALS(2) singular values less than
TOL, if this many exist. If there is one output argument to SPNRANK

the default value of NSVALS is 0 and if there are two or more output
arguments to SPNRANK the default value of NSVALS is 3.

[NRANK, SVALS, INDICES] = SPNRANK(A, . . .) also returns the indices
of the estimated singular values in SVALS. SVALS(i) is an estimation
for singular value number INDICES(i) of A.

[NRANK, SVALS, INDICES, SVALS_ERR] = SPNRANK(A, [], NSVALS) or
[NRANK, SVALS, INDICES, SVALS_ERR] = SPNRANK(A, TOL, NSVALS) will
also return error bounds for the calculated singular values.
For each i, A (or, to be precise, a perturbation of A where the
perturbation is order of magnitude ||A|| eps where eps is relative
machine precision) is guaranteed to have a singular value in the
interval
    [SVALS(i) - SVALS_ERR(i), SVALS(i) + SVALS_ERR(i)].
The bound does not guarantee that the true singular value of A
is the ith singular value but this is usually the case.

Note that if TOL lies in such an interval then the calculated numerical
rank, NRANK, may be incorrect (see STATS.flag_svals_err, below). This
can occur, for example, if TOL is very close to a singular value of A.
To reduce SVALS_ERR one can decrease OPTS.tol_eigs, or increase
OPTS.thresh (see below). Also changing TOL and NSVALS can affect
SVALS_ERR.

SPNRANK(A,TOL,NSVALS,OPTS) specifies options:
OPTS.thresh - for sparse A OPTS.thresh is the threshold in the LDL
   calculation (see LDL) [scalar in [0, 0.5] | {0.01}]. Only
   used in Matlab 7.6 or higher.
OPTS.tol_smax - the tolerance in estimating the norm of A (if
   required)using NORMEST_ERR, a minor modification of Matlab's
   NORMEST [scalar | {1.e-6}]
The remaining fields are used only when NSVALS > 0. In this case
   EIGS_MAXTIME is used to estimate the singular values of A near TOL.
   EIGS_MAXTIME is identical to Matlab's EIGS, except an option to
   limit the run time has been added.
OPTS.tol_eigs: stopping tolerance in EIGS [scalar | {1.0e-6}]
OPTS.maxit: maximum number of iterations in EIGS [integer | {300}]
OPTS.disp: diagnostic information display level in EIGS and SPNRANK
       [{0} | 1 | 2]
OPTS.maxtime: EIGS_MAXTIME is terminated after OPTS.maxtime seconds
  [scalar > 0 | {Inf (no time limit) }]

[NRANK, SVALS, INDICES, SVALS_ERR, STATS] = SPNRANK(A, ...) will return
the structure STATS where
STATS.tol is the tolerance used to calculate the numerical rank
STATS.smax is an estimate of the two norm of A, if it is calculated
STATS.smax_err is a bound on | smax - (singular value of A closest
   to smax)|, if smax is calculated
STATS.time_smax is the elapsed (wall clock) time to calculate smax
STATS.time_ldl is the elapsed time to run LDL

STATS.flag_ldl is 0 if LDL succeeded and 1 if LDL failed
STATS.message_ldl is the error message produced by LDL, if LDL fails
STATS.max_L is the largest magnitude element in L produced by LDL
STATS.time_eigs is the elapsed time to run EIGS_MAXTIME
STATS.flag_eigs is the flag returned by EIGS (if STATS.flag_eigs is 0
then all the eigenvalues converged; otherwise not all converged.)
STATS.time_svals_err is the time to calculate SVALS_ERR
STATS.flag_svals_err = 1 when TOL lies within at least one interval
[SVALS(i) - SVALS_ERR(i), SVALS(i) + SVALS_ERR(i)] and the
calculated NRANK may be incorrect. STATS.flag_svals_err = 0
is consistent with a correctly calculated numerical rank, NRANK.
STATS.flag_svals_err is NaN if any component of SVALS or SVALS_ERR
is not finite (which is possible when flag_eigs is 1).
Examples:

```
load west0479;
A=west0479*west0479;
spnrank(A)
%or
[nrank, svals, indices, svals_err, stats]=spnrank(A);
%or
opts.tol_eigs = 1.e-10;
[nrank, svals, indices, svals_err, stats]=spnrank(A,[],[],opts);
```

References:

Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe and Henk van der Horst, Templates for the Solution of Algebraic Eigenvalue Problems, SIAM, Philadelphia, 2000.

James Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, 1997.

Nicholas J. Higham, Accuracy and Stability of Numerical Algorithms, Second Edition, SIAM, Philadelphia, 2002.

R. B. Lehoucq, Danny C. Sorensen and C. Yang. ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. SIAM, Philadelphia, 1998.

Beresford. N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, NJ, 1980. Reprinted as Classics in Applied Mathematics 20, SIAM, Philadelphia, 1997.

Hong Zhang, Barry Smith, Michael Sternberg and Peter Zapol, SIPs: Shift-and-Invert Parallel Spectral Transformations, ACM Transactions on Mathematical Software, 33:1-19, 2007.