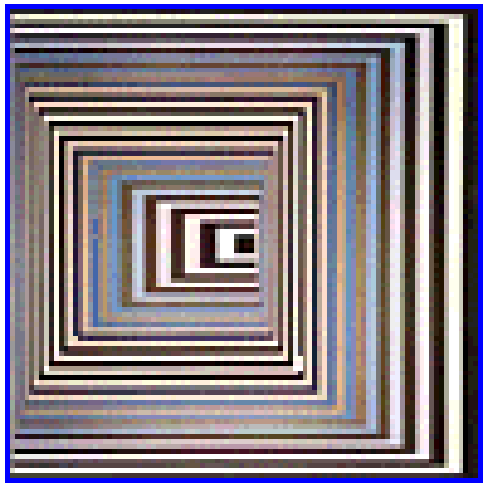
LinBox Lab – University of Delaware

December 5, 2011

D. Saunders, D. Wood, B. Youse

Recent alumni:
(J.P. May, Z. Wan, D. Roche, A. Duran, ...)
(Undergrads: M. Wezowicz, N. Messina, M. Fendt, D. Roche, ...)



linalg.org

LinBox development collaborators at Grenoble, Lyon, Nancy in France; Waterloo and Calgary in Canada, Raleigh (NCSU), Philadelphia(Drexel) in USA; others... High Performance Kernel collaborators: J. Johnson, G. Harrison, L.C. Meng.

Funded by the National Science Foundation

Problems solved by LinBox

- solving systems of linear equations, matrix rank, determinant, minimal polynomial, characteristic polynomial, canonical forms (Frobenius, Smith).

- Exact input: integer and rational matrices

- Exact computation: no numeric approximation. (lots of modular computation and Chinese remaindering).

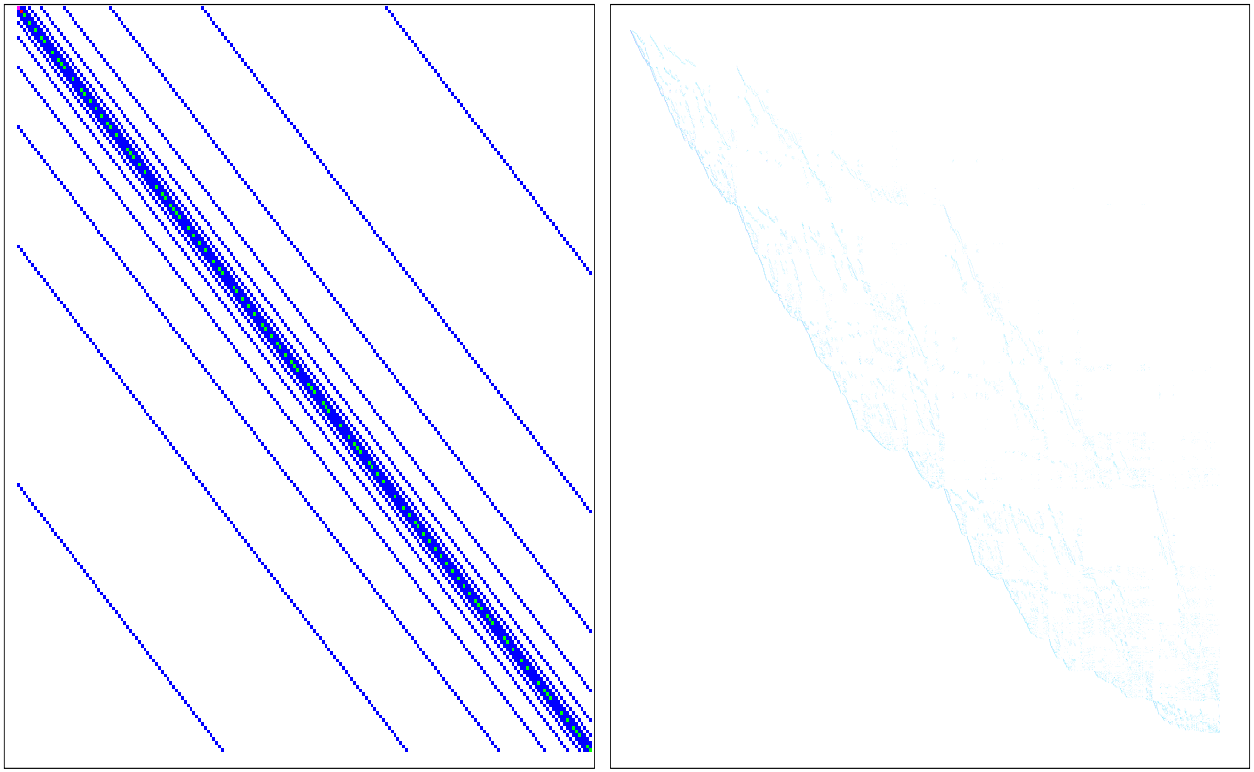
- Sparse matrices

- Parallel computation

- Many applications
  - Homology of simplicial complexes.
  - multivariate polynomial equation system solving.
  - Incidence structures in discrete mathematics.

- Problems may be huge (millions of equations, billions of coefficients)

Picture of Trefethen and TF class matrices



Very sparse matrices, about $2 \log n$ non-zero entries per row in Trefethen matrices.

Central Problems

- Effective representation of sparse matrices (store only nonzeroes, irregular computations).

- Expresssion swell. Let A be a 1000 by 1000 matrix of 10 digit entries and let b a 1000-vector of 10 digit entries. This data requires about 4 megabytes of storage. The solution vector x to $Ax = b$, will have rational number entries of very long numerators and denominators. The output x will require 9 megabytes. This length 1000 vector is more of a memory hog than the 1000 by 1000 matrix!

- Library design: software engineering.

Methods

- Blackbox (BB) methods are excellent for large sparse matrices over finite fields. Wiedemann, Kaltofen-Saunders, Dumas-Saunders-Villard...

- Sparse elimination (such as SuperLU of Demmel, et al) is excellent on matrices which are small, or slow to fill in. Dumas implementation finite fields. Giesbrecht et al, new asymptotically fast algorithm.

- Dense matrix eliminations are fast by using floating point BLAS. Trick is to get exact, not approximate, results from BLAS for our modular computations. Lapack floating point linear system solving can also be used.

Example. Strongly regular graphs. Needed: rank of adjacency matrix modulo 3. Problem is from combinatorialist Prof. Qing Xiang (UD Math Dept).

Dickson strongly regular graph construction: The edges (adjacency matrix entries) are generated by computations over a "semi-field". The adjacency matrix is dense and is n by n, where $n = 9^k$. But (and this is the opening for interesting algorithm development), it is estimated that the rank will be small, only about $2 * 4^k$.

Dickson SRG					
e	dimension	Rank	2007t	2009t	2011t
1	9	4	0.001s	0.0002s	
2	81	20	0.021s	0.0012s	
3	729	85	0.35s	0.022s	
4	6,561	376	33.3s	0.95s	
5	59,049	1654	0.5h	0.017h	
6	531,441	7283	46.7h	1.2h	
7	4,782,969	32064	-	96.4h	
8	43,046,721	128000 ??	-	-	!!

Based on this data, we have conjectured that rank satisfies $R(k) = 4R(k-1) + 2R(k-2) - R(k-3)$.

To add strength to the conjecture, our current goal is to do the $k = 8, n = 3^{16} \approx 43$ million case.
Full storage of the matrix would be 2 petabytes.

Methods: Preconditioning and projection. Just in time matrix representation. Out of core matrix representations. Packed storage for mod 3 values and corresponding arithmetic operators. Parallelism.

This problem is being used to stress test the CITADel project, a UD network upgrade for scientific computation.

Watch for a course on parallel computation in Fall 2012.

Future work for the LinBox team

- Theory: For the run time, best asymptotic lower bounds (problem complexity) $\neq$ best asymptotic upper bounds (algorithm complexity).
  - Design fast algorithms for general case.
  - Design fast algorithms for special matrix classes.
  - Prove *any* non-trivial lower bound.
- Practice: Best practical algorithm is determined by problem size and shape, by hardware properties, by the available tools.
  - Implement and test the best algorithms.
  - Improve the library design for genericity and performance.
  - *Engineer the hybrid algorithms*.
  - Make greater use of parallelism and accelerators (GPU, FPGA).
  - Continue to provide the best performing integer matrix computation package in the world.