Project NSF-CNRS-NSERC LinBox[*]

The LinBox team[†]

July 6, 2006

Home: `http://linalg.org`; documentation: `http://linalg.org/docs.html`; package: `http://linalg.org/download.html`; online computing servers: `http://linalg.org/servers.html`.

1 LinBox

LinBox is a C++ template library of routines for solution of linear algebra problems

Matrices	Functions
integer entries	solve linear system
rational entries	matrix rank
finite field entries	determinant
	minimal polynomial
Sparse	characteristic polynomial
Dense	Smith normal form
Structured	trace

A good collection of finite field and ring implementations is provided, for use with numerous black box matrix storage schemes.

2 Project Goals

Genericity and high performance are the twin goals of LinBox. The genericity is achieved by use of a small set of interfaces. Algorithms are implemented with

[*]École Normale Supérieure de Lyon, Institut National Polytechnique de Grenoble North Carolina State University, Université de Grenoble, University of Calgary, University of Delaware, University of Waterloo.

[†]linbox@yahoogroups.com

C++ template parameters which may be instantiated with any class adhering to the specified interface. High performance is achieved by judicious specializations of the generic algorithms. It is entirely within the spirit of the project to introduce new implementations. Thus a user of the library may invoke a LinBox algorithm, say for determinant or rank of a matrix, but providing a black box class of her own design and perhaps even providing the underlying field (or commutative ring) representation. Conversely, the LinBox field and ring interfaces and the many specific representations can be used for purposes other than linear algebra computation or with algorithms not provided by LinBox.

Three major threads have come together to form the linear algebra library LinBox. The first is the use of modular algorithms when solving integer or rational matrix problems. The second thread and original motive for LinBox is the implementation of black box algorithms for sparse/structured matrices. Finally, it has proven valuable to introduce elimination techniques that exploit the floating point BLAS libraries even when our domains are finite fields. The latter is useful for dense problems and for block iterative methods.

Black box techniques [4] are enabling exact linear algebra computations of a scale well beyond anything previously possible. The development of new and interesting algorithms has proceeded apace for the past two decades. It is time for the dissemination of these algorithms in an easily used software library so that the mathematical community may readily take advantage of their power. LinBox is that library [3].

Exact black box methods are currently successful on sparse matrices with hundreds of thousands of rows and columns and having several million nonzero entries. The main reason large problems can be solved by black box methods is that they require much less memory in general than traditional elimination-based metshods do. This fact is widely used in the numerical computation area. We refer for instance to the templates for linear system solution and eigenvalue problems [1]. This has also led the computer algebra community to a considerable interest in black box methods. Since Wiedemann's seminal paper [5], many developments have been proposed especially to adapt Krylov or Lanczos methods to fast exact algorithms. We refer to [2] and references therein for a review of problems and solutions.

LinBox supplies efficient black box solutions for a variety of problems including linear equations and matrix normal forms with the guiding design principle of re-usability. The most essential and driving design criterion for LinBox is that it is generic with respect to the domain of computation. This is because there are many and various representations of finite fields each of which is advantageous to use for some algorithm under some circumstance. The integral and rational number capabilities depend heavily on modular techniques and hence on the capabilities over finite fields. In this regard, generic software methodology is a powerful tool.

Using examples from demanding applications (Trefethen's one hundred digits challenge, signature of a matrix arising in Lie group representation, Smith form for simplicial homology, determinants for combinatorial identities, characteristic polynomials in the study of graph properties, etc.), we have demonstrated four general levels of use of our library. In order from least involved with the details to most involved, they are:

1. Access using a linbox web server.

   The user at this level must attend to preparation of a matrix in a suitable file format and invoking the service. The server itself provides adequate documentation for this.

2. Access using already compiled functions or through an interface to linbox in a general purposes system such as Maple or GAP.

   The user at this level must see to installation, and then attend to preparation of her matrix in a suitable file format and to the form of the program or procedure invocation. A number of programs are available in the examples directory distributed with LinBox providing for rank, determinant, linear system solution, etc.

3. Use of LinBox as a programmers library for exact linear algebra functions.

   At this level a user must do at least the following:

   (a) Choose a field or ring representation and construct a specific field or ring object R.

   (b) Choose a black box or matrix representation suitable to your data and construct specific matrix A over R.

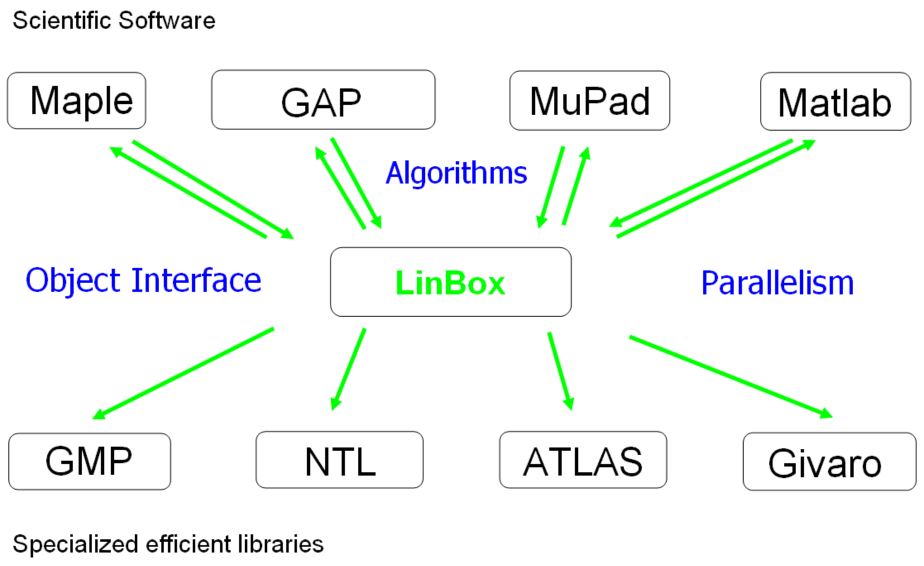
   (c) Call needed algorithm. The solutions directory is designed to support this by providing functions with simple problem oriented interfaces (rank(), det(), solve(), etc.), yet allowing some user control of algorithm details. The programmer may be providing some of the parts such as an application specific black box matrix class.

4. Power development.

   Again, this is use of LinBox as a library, but with hands fully on the details. The programmer at this level apparently needs the best opportunities for high performance and is willing to use the more complicated internal interfaces to get it. Direct calls to functions in the algorithms directory and perhaps to related packages such as fflas, ffpack, or other components are being used.

3 A middleware

LinBox is designed to serve as a middleware product lying between basic software tools and higher level systems as indicated in this picture.



3.1 LinBox (middleware) uses

- GMP for basic large integer arithmetic
- ATLAS (or similar source) for BLAS and Lapack routines. These floating point functions are used judiciously for fast, exact computation.
- NTL for some finite field and ring representations, particularly in the case of GF(q), where q is a prime power or a prime greater than word size. NTL is also used by algorithms that need polynomial operations such as factorization.
- Givaro as another source of field representations and polynomial operations. Importantly, Givaro provides our best representation of small non-prime fields, say $q = p^e < 10^6$.
- Functionality from some other systems has been wrapped also but is currently less widely used.

3.2 LinBox (middleware) is used by

- web servers, providing the main matrix functions (rank, det, solve, minpoly, ...) for integer matrices and matrices mod a prime.
- GAP Homology package, providing Smith form and homology of simplicial complexes.

4 LinBox library organization

The distributed package directory contains subdirectories (documented as 'modules') as follows.

- `linbox`. This contains the library sources (headers). The defined objects are in the namespace `LinBox`.
  - `linbox/field`, field and ring representations.
  - `linbox/blackbox`, matrix blackbox representations providing matrix-vector product..
  - `linbox/matrix`, mutable sparse and dense matrices.
  - `linbox/algorithms`, black box and elimination algorithms. This is the heart of LinBox.
  - `linbox/solutions`, convenience wrappers of algorithms
  - `linbox/util`, basic integers, timer, commentator.
- `examples`, model programs for direct use and/or illustration.
- `doc`, html documentation built using Doxygen.
- `interfaces`, interfaces to other systems.
- `tests`, primarily consisting of correctness checks.

5 Usage example: the determinant

Let us illustrate the use of LinBox with two simple variations on code to compute a determinant.

5.1 Determinant of a sparse matrix over a small finite field, via a Blackbox method

```
#include <linbox/field/modular.h>
#include <linbox/blackbox/sparse.h>
#include <linbox/solutions/det.h>
main(){
// types
   typedef LinBox::Modular<double> Field;
   typedef LinBox::SparseMatrix<Field> Matrix;
// objects
   Field F( 65521 );
   Matrix A( F );
   A.read( std::cin );
   Field::Element d;
// action
   LinBox::det( d, A, LinBox::Method::Blackbox() );
   F.write( std::cout << "the determinant is ", d ) << std::endl;
}
```

5.2 Determinant of an integer dense matrix, via BLAS optimized routines

```
#include <linbox/field/gmp-integers.h>
#include <linbox/blackbox/dense.h>
#include <linbox/solutions/det.h>
using namespace LinBox; using namespace std;
main(){
// objects
   GMP_Integers ZZ; int n = 1000;
   DenseMatrix<GMP_Integers> A( ZZ, n, n );
   for (int i = 0; i < n; ++i)
      for (int j = 0; j < n; ++j)
         A.setEntry( i, j, 1 + i*j );
   GMP_Integers::Element d;
// action
   cout << "the determinant is ";
   ZZ.write( cout, det( d, A ) ) << endl;
}
```

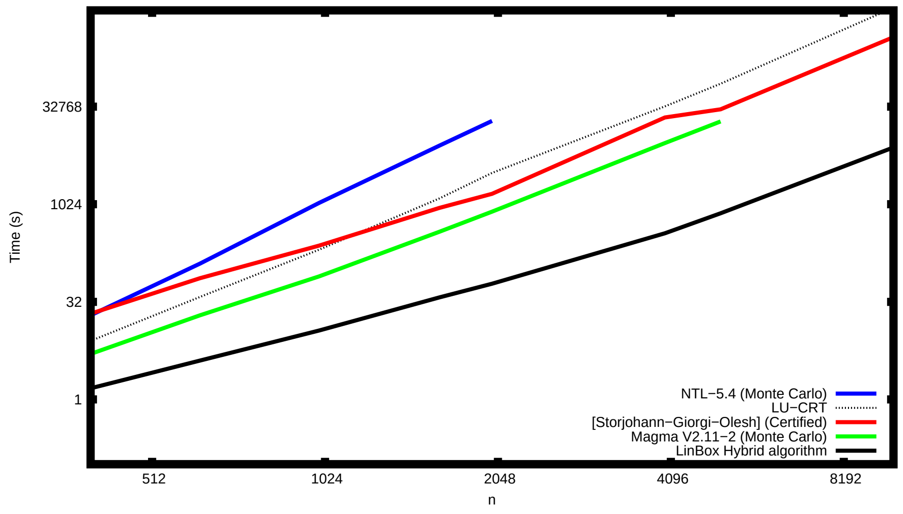


Figure 1: Comparison of determinant with other existing implementation. Tested on random dense matrices of the order 400 to 10000, with entries {-8,-7,. . . ,7,8}

5.3 Performance measurements

We conclude with some timings of determinant computation by LinBox and some other packages. These measurements were made in 2005 on an Itanium 2 running at 1.3Gh. The horizontal (matrix size) and vertical (time) scales are logarithmic.

References

[1] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Ed.* SIAM, 1994.

[2] L. Chen, W. Eberly, E. Kaltofen, B. Saunders, W. Turner, and G. Villard. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and its Applications*, 343-344:119–146, 2002.

[3] J.-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. LinBox: A generic library for exact linear algebra. In A. M. Cohen, X.-S. Gao, and N. Takayama, editors, *Proceedings of the 2002 International Congress of Mathematical Software, Beijing, China*, pages 40–50. World Scientific Pub, Aug. 2002.

[4] E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symb. Comp.*, 9(3):301–320, 1990.

[5] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transf. Inform. Theory*, IT-32:54–62, 1986.