The LinBox Project for Linear Algebra Computation

A Practical Tutorial

Daniel S. Roche

Symbolic Computation Group
School of Computer Science
University of Waterloo

MOCAA 2008
University of Western Ontario
8 May 2008
Goals for Today

I want to convince you that LinBox is . . .

- **the** tool for exact linear algebra

- easy to install and use (if you want it to be)

- worth getting involved with
Significant Publications

1986  Wiedemann: *Solving sparse linear equations over finite fields*
1991  Kaltofen & Saunders: On Wiedemann’s Algorithm
1993  Coppersmith: Block Lanczos, Block Wiedemann
1995  Montgomery: Implementation of Block Lanczos
1997  Villard: Analysis of Block Wiedemann
2001  Giesbrecht; Dumas, Saunders, & Villard: Smith forms
2001  Chen, Eberly, Kaltofen, Saunders, Turner, Villard: *Efficient Matrix Preconditioners for Black Box Linear Algebra*
2002  Dumas, Gautier, Giesbrecht, Giorgi, Hovinen, Kaltofen, Saunders, Turner, Villard: *Linbox: A Generic Library For Exact Linear Algebra*
2002  Dumas, Gautier, Pernet: *FFLAS: Finite Field Linear Algebra Subroutines*
LinBox Milestones

2000  Initial design meetings
2002  World Scientific paper
2004  BLAS is thoroughly integrated
2005  LinBox 1.0 Released
      JGD gives tutorial at ISSAC in Beijing
      LinBox included in open-source computer algebra CD
2006  Maple interface and web computation server
2007  SAGE integration
2008  Linbox 1.5
What is LinBox?

- C++ Library for Exact Computational Linear Algebra
- Open-source (LGPL), international research project
- Generic, using C++ templates
- Middleware — uses fast, low-level libraries, used by higher-level CA systems
- Originally for sparse & structured blackbox computations
- Now incorporates dense as well
Fathers of LinBox

Dave Saunders

Erich Kaltofen

Gilles Villard

Mark Giesbrecht
Developers

- Jean-Guillaume Dumas
- Bradford Hovinen
- Will Turner
- David Pritchard
- Clement Pernet
- Pascal Giorgi
- Zhendong Wang
- William Stein
- Mike Abshoff
Wiedemann’s Algorithm

The Problem

Given nonsingular $A \in F^{n \times n}$ and $v \in F^n$ find $x \in F^n$ such that $Ax = v$. 
Wiedemann’s Algorithm

The Problem

Given nonsingular $A \in F^{n \times n}$ and $v \in F^n$ find $x \in F^n$ such that $Ax = v$.

The Algorithm

First, project on the right:

The sequence

$$v, \quad Av, \quad A^2v, \quad A^3v, \ldots$$

is linearly recurrent of degree $n$.

Note: $A^i v = A \cdot (A^{i-1} v) = A \cdot (A \cdot (A \cdots A v))$
Wiedemann’s Algorithm

The Problem

Given nonsingular $A \in F^{n\times n}$ and $v \in F^n$ find $x \in F^n$ such that $Ax = v$.

The Algorithm

Then, project on the left with random $u \in F^n$:

The sequence

$$u^Tv, \ u^TAv, \ u^TA^2v, \ u^TA^3v, \ldots$$

has the same linear recurrence with high probability.
Wiedemann’s Algorithm

The Problem
Given nonsingular $A \in F^{n \times n}$ and $v \in F^n$ find $x \in F^n$ such that $Ax = v$.

The Algorithm
Compute $f \in F[x]$, the minimum polynomial of $(u^T A^i v)_{i \geq 0}$.

\[
f(x) = -1 + f_1 x + f_2 x^2 + \cdots + f_n x^n \\
0 = -u^T v + f_1 u^T A v + f_2 u^T A^2 v + \cdots + f_n u^T A^n v
\]
Wiedemann’s Algorithm

The Problem

Given nonsingular $A \in \mathbb{F}^{n \times n}$ and $v \in \mathbb{F}^n$ find $x \in \mathbb{F}^n$ such that $Ax = v$.

The Algorithm

W.h.p. $f$ is the minimum polynomial of $(A^i v)_{i \geq 0}$.

$$f(x) = -1 + f_1 x + f_2 x^2 + \cdots + f_n x^n$$

$$0 = -u^T v + f_1 u^T A v + f_2 u^T A^2 v + \cdots + f_n u^T A^n v$$

$$0 = -v + f_1 A v + f_2 A^2 v + \cdots + f_n A^n v$$
Wiedemann’s Algorithm

The Problem

Given nonsingular $A \in \mathbb{F}^{n \times n}$ and $v \in \mathbb{F}^n$ find $x \in \mathbb{F}^n$ such that $Ax = v$.

The Algorithm

Rearrange and multiply by $A^{-1}$:

\[
\begin{align*}
  f(x) &= -1 + f_1 x + f_2 x^2 + \cdots + f_n x^n \\
  0 &= -u^T v + f_1 u^T Av + f_2 u^T A^2 v + \cdots + f_n u^T A^n v \\
  0 &= -v + f_1 Av + f_2 A^2 v + \cdots + f_n A^n v \\
  x &= A^{-1} v = f_1 v + f_2 A v + \cdots + f_n A^{n-1} v
\end{align*}
\]
Black Box Approach

$\mathbf{v} \in \mathbb{F}^n \rightarrow \mathbf{A}\mathbf{v} \in \mathbb{F}^m$

$\mathbf{A} \in \mathbb{F}^{m \times n}$

“Usual” Goals

- $O(n)$ black box calls
- $O(n^2)$ other work
- $O(n)$ space

Good Black Box Algorithms for:
minimum polynomial, rank, determinant, linear system solving, Smith normal form, ...
Black Box Classes

Common costs for black box vector apply:

- **Dense**
  \[ O(n^2) \]

- **Sparse**
  \[ O(s) \]
  \[ s \text{ entries} \]

- **Toeplitz**
  \[ O(M(n)) \]

**Other blackboxes**: Hankel, Sylvester, Diagonal, Hilbert, . . .
Block Methods

Similar to before, vectors now replaced by rectangular matrices.

\[ B \in \mathbb{F}^{n \times k} \rightarrow AB \in \mathbb{F}^{m \times k} \]
\[ A \in \mathbb{F}^{m \times n} \]

- Many mathematical complications
- Allows use of fast dense methods (i.e. BLAS)
- Usually a tradeoff
LinBox as Middleware
LinBox as Middleware

Packages used by LinBox

- BLAS
- GMP
- NTL
- Givaro
LinBox as Middleware

Higher-level Systems using LinBox
Basic Linear Algebra Subprograms

Extremely efficient low-level routines for floating-point linear algebra:

### BLAS Levels

1. Scalar product, vector-scalar product, vector-vector sum
2. Dot product, matrix-vector product
3. Matrix-matrix product

Use either:

- Proprietary BLAS — for most architectures (e.g. Intel’s MKL)
- Free BLAS (e.g. ATLAS, GotoBLAS)
GNU Multiplie Precision

Provides routines for exact, multiplie-precision integer arithmetic. Classical, Karatsuba, Toom-Cook, and FFT multiplication (with crossovers).

Advantages:
- Many inner loops written in assembly
- Large user community
NTL, Givaro, . . .

- Libraries for number theory (field, ring, polynomial arithmetic).

- Used to work with matrices over finite fields, polynomials, etc.

- Also used for structured matrix computations (e.g. Toeplitz)

- Free, open-source, efficient
Three Choices to Make

1. **Underlying Domain** ("field")
   - Integers, Rationals
   - Prime and Prime Power Fields
   - Polynomial Rings

2. **Matrix Representation**
   - Sparse Black Box
   - Dense Explicit
   - Structured Black Box

3. **Algorithm**
Vertical Structure Approach

Used e.g. by NTL:

<table>
<thead>
<tr>
<th>( \text{zz}_p )</th>
<th>( \text{GF2} )</th>
<th>( \text{ZZ}_pE )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{zz}_pX )</td>
<td>( \text{GF2X} )</td>
<td>( \text{ZZ}_pEX )</td>
</tr>
<tr>
<td>( \text{vec} _\text{zz}_p )</td>
<td>( \text{vec} _\text{GF2} )</td>
<td>( \text{vec} _\text{ZZ}_pE )</td>
</tr>
<tr>
<td>( \text{mat} _\text{zz}_p )</td>
<td>( \text{mat} _\text{GF2} )</td>
<td>( \text{mat} _\text{ZZ}_pE )</td>
</tr>
</tbody>
</table>

\[ \vdots \] \[ \vdots \] \[ \vdots \]

- Only one choice given underlying domain!
Dynamic Type-Checking

- Traditional polymorphism, required in Java, possible in C++

Have to do a table lookup for every variable reference!
Our Approach: C++ Templates

- All type decisions made at compile-time
- Every combination is possible
- Huge efficiency gains over polymorphism (pipelining, inlining)

![Diagram showing algorithms, matrix representation, and domain with nodes labeled GMP Int., Z_p, Z_p/⟨Γ⟩, Polynomial, Sparse, Dense, Toeplitz, Diagonal, Black Box, Direct, Other.]
Drawbacks to C++ Templates

- Programmers can make **bad decisions**
  (e.g. using elimination algorithm with sparse blackbox matrix)

- Binary code bloat
  Every template combination which is used must be compiled.
  Creating a complete, precompiled library is impractical.
## Levels of Use

1. **Online Computation Server**
   - Does not require any installation

2. **Maple, SAGE, examples directory**
   - User sees no C++ code

3. **solutions directory**
   - User chooses field, blackbox, writes in C++

4. **Expert user**
   - Direct programming — complete control
Installation

Installing LinBox is not difficult!

- Installing ATLAS can be tricky
- *All* LinBox dependencies can be found in Debian repo.
- Getting maximal efficiency is inherently difficult
- Middleware is inherently troublesome
## Folder Structure

<table>
<thead>
<tr>
<th>examples</th>
<th>Ready-to-go programs for typical problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>interfaces</td>
<td>Connections to Maple, SAGE, etc.</td>
</tr>
<tr>
<td>linbox</td>
<td>Library code</td>
</tr>
<tr>
<td>/field</td>
<td>Underlying domains</td>
</tr>
<tr>
<td>/blackbox</td>
<td>Matrix representations</td>
</tr>
<tr>
<td>/algorithms</td>
<td>Heart of the library</td>
</tr>
<tr>
<td>/solutions</td>
<td>Code for typical problems</td>
</tr>
<tr>
<td>/randiter</td>
<td>Random element generation</td>
</tr>
<tr>
<td>/util</td>
<td>General utilities (e.g. I/O)</td>
</tr>
<tr>
<td>doc</td>
<td>Documentation generation</td>
</tr>
<tr>
<td>tests</td>
<td>Correctness checks and benchmarks</td>
</tr>
</tbody>
</table>
Input/Output

If using LinBox at “Level” 1 or 2, I/O is handled automatically.

Otherwise, use the MatrixStream class.

- Automatic recognition of many formats
- Outputs (row,column,value) “triples” or a single dense array
- Connects to most matrix representations
- Allows any rep. to be read from any file

Similar ideas for output have been proposed; not yet implemented.
What LinBox is Great At

LinBox is great at computing

\{rank, determinant, linear solution, characteristic polynomial, Smith form\}

for matrices which are

\{dense, sparse, structured\}

over

\{integers, rationals, prime fields, extension fields, polynomials\}. 
What LinBox is Great At

LinBox is great at computing

\{rank, determinant, linear solution, characteristic polynomial, Smith form\}

for matrices which are

\{dense, sparse, structured\}

over

\{integers, rationals, prime fields, extension fields, polynomials\}.
What LinBox is Great At

LinBox is great at computing

\{rank, determinant, linear solution, characteristic polynomial, Smith form\}

for matrices which are

\{dense, sparse, structured\}

over

\{integers, rationals, prime fields, extension fields, polynomials\}. 
What LinBox is *not* Great At

- Inverse computation
- Nullspace computation
- Support for $\text{GF}(2)$
- Parallelism
- Block methods
Software Engineering

LinBox has a rapidly growing user base (!)

We need some software engineering!

- Configure/Install (autohell)
- Removing legacy code
- Restructuring matrix/blackbox distinction
- Interfaces
Software Engineering

LinBox has a rapidly growing user base (!)

We need some software engineering!
- Configure/Install (autohell)
- Removing legacy code
- Restructuring matrix/blackbox distinction
- Interfaces

ANY VOLUNTEERS?