The LinBox library

Clément PERNET & the LinBox group

CAT Workshop,
Aug 29, 2009

Introduction

Exact linear algebra:

- over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}_p, \mathrm{GF}(p^k)$.
- `matrix-multiply, solve, rank, det, echelon, charpoly, Smith-Normal-Form, ...`
- dense, sparse, blackbox matrices

Introduction

Exact linear algebra:

- over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}_p, \mathrm{GF}(p^k)$.
- `matrix-multiply, solve, rank, det, echelon, charpoly, Smith-Normal-Form, ...`
- dense, sparse, blackbox matrices

Growing applicative demand

- CAT: Homology of simplicial complexes
- Number Theory: computing modular forms,
- Crypto: NFS, DLP Groebner bases, ...
- Graph Theory: closure, spectrum, ...
- High precision approximate linear algebra
- ... (*Mathematics is the art of reducing any problem to linear algebra* [W. Stein])

Software solutions for exact computations

Specialized libraries

finite fields: NTL, Givaro, Lidia, ...

integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

Software solutions for exact computations

Specialized libraries

finite fields: NTL, Givaro, Lidia, ...

integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

End-user level softwares

- Maple, Mathematica, MuPad, ... (closed source)
- Sage, Pari, Maxima, ... (open source)

Software solutions for exact computations

Specialized libraries

finite fields: NTL, Givaro, Lidia, ...

integers: GMP, MPIR

polynomials: NTL, Givaro, zn_poly ...

End-user level softwares

- Maple, Mathematica, MuPad, ... (closed source)
- Sage, Pari, Maxima, ... (open source)

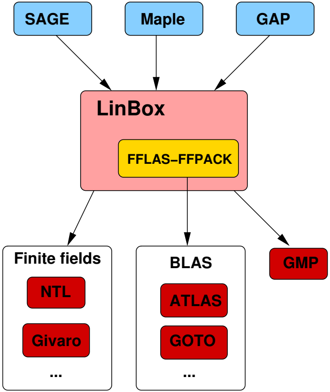
Linear Algebra ?

Outline

Outline

1. Organization and design
2. Algorithmic models
   - Black box matrices
   - Dense matrices
   - Sparse matrices
   - Lifting over the integers
3. Evolution and perspectives

A generic middleware



- uses basic implementations from specialized libraries (GMP, Givaro, NTL, BLAS...)
- Optional libraries used in a Plug & Play manner
- Interfaces to top-level softwares (Maple, Sage, GAP)

The LinBox project, facts

Joint NFS-NSERC-CNRS project.

US: U. of Delaware, North Carolina State U.
Canada: U. of Waterloo, U. of Calgary,
France: Grenoble U., INRIA (Lyon, Grenoble)

The LinBox project, facts

Joint NFS-NSERC-CNRS project.

US: U. of Delaware, North Carolina State U.

Canada: U. of Waterloo, U. of Calgary,

France: Grenoble U., INRIA (Lyon, Grenoble)

A LGPL source library:

- 125 000 lines of C++ code
- about 5 active developers

- Availaible online: `http://linalg.org`
- Google groups: `linbox-devel, linbox-use`
- Distributed in Debian and Sage

Design of LinBox v1

Features:

Solutions

- rank
- det
- minpoly
- charpoly
- solve
- positive definiteness
- Smith normal form

Design of LinBox v1

Features:

Solutions

- rank
- det
- minpoly
- charpoly
- solve
- positive definiteness
- Smith normal form

Domains of computation

- $\mathbb{Z}_p$, $\mathbb{F}_q$
- $\mathbb{Z}$

Matrices

- Dense
- Sparse
- Blackbox

Genericity

- Domain wrt. element representations:

```
template <class Element>
class Modular<Element>;
```

- Matrix wrt. domains:

```
template <class Field>
class DenseMatrix<Field>;
```

- Algorithms wrt. matrices:

```
template <class Matrix>
unsigned long rank (unsigned long & r,
                    const Matrix & A);
```

Interface

Field/Ring Plug & Play interface

- Common interface with Givaro

```
Modular<int> F(11);
int x,y,z;
F.init(x,2);
F.init(y,13);
F.mul(z,x,y);
```

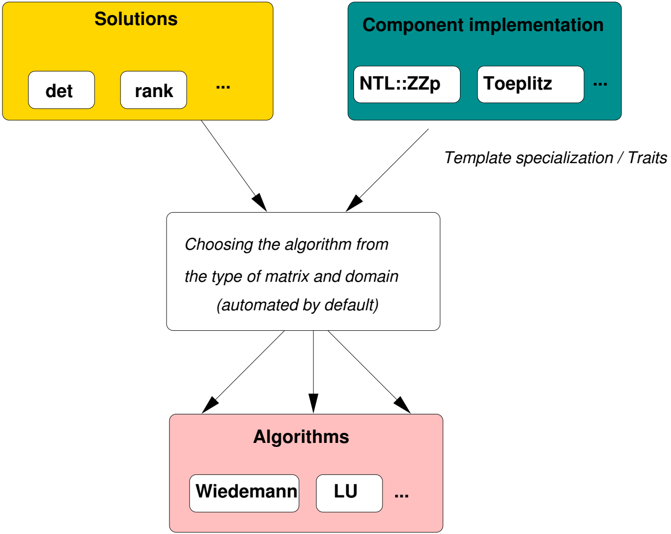
- Wraps NTL, Lidia, Givaro implementations
- Proper floating point based implementations for dense computations

BLAS

Compliant with the standard C-BLAS interface

- GotoBLAS, ATLAS, MKL, GSL, ...

Structure of the library



Several levels of use

- Web servers: `http://www.linalg.org`

Several levels of use

- Web servers: `http://www.linalg.org`
- Executables: `$ charpoly MyMatrix 65521`

Several levels of use

- Web servers: `http://www.linalg.org`
- Executables: `$ charpoly MyMatrix 65521`
- Call to a solution:

```
NTL::ZZp F(65521);
Toeplitz<NTL::ZZp> A(F);
Polynomial<NTL::ZZp> P;
charpoly (P, A);
```

Several levels of use

- Web servers: `http://www.linalg.org`
- Executables: `$ charpoly MyMatrix 65521`
- Call to a solution:

```
NTL::ZZp F(65521);
Toeplitz<NTL::ZZp> A(F);
Polynomial<NTL::ZZp> P;
charpoly (P, A);
```

- Calls to specific algorithms

Outline

1. Organization and design
2. Algorithmic models
   - Black box matrices
   - Dense matrices
   - Sparse matrices
   - Lifting over the integers
3. Evolution and perspectives

Black box linear algebra

Black box linear algebra

- Matrices viewed as linear operators
- algorithms based on matrix-vector apply only $\Rightarrow$cost $E(n)$

Black box linear algebra

- Matrices viewed as linear operators
- algorithms based on matrix-vector apply only $\Rightarrow$cost $E(n)$

Structured matrices: Fast apply (e.g. $E(n) = \mathcal{O}(n \log n)$)

Sparse matrices: Fast apply and no fill-in

Black box linear algebra

- Matrices viewed as linear operators
- algorithms based on matrix-vector apply only $\Rightarrow$cost $E(n)$

Structured matrices: Fast apply (e.g. $E(n) = \mathcal{O}(n \log n)$)

Sparse matrices: Fast apply and no fill-in

$\Rightarrow$

- Iterative methods
- No access to coefficients, trace, no elimination
- Matrix **multiplication** $\Rightarrow$ Black-box **composition**

Black box linear algebra

Minimal polynomial: [Wiedemann 86]

$\Rightarrow$adapts numerical iterative Krylov/Lanczos methods

$\Rightarrow\mathcal{O}\left(dE(n) + n^2\right)$ operations

Black box linear algebra

Minimal polynomial: [Wiedemann 86]

$\Rightarrow$adapts numerical iterative Krylov/Lanczos methods

$\Rightarrow\mathcal{O}\left(dE(n) + n^2\right)$ operations

Rank, Det, Solve: [Kaltofen & Saunders 90, Chen& Al. 02]

$\Rightarrow$reduced to minimal polynomial and preconditioners

$\Rightarrow\mathcal{O}\tilde{}\left(nE(n) + n^2\right)$ operations

where $E(n)$: cost of applying the matrix to a vector

Black box linear algebra

Minimal polynomial: [Wiedemann 86]

$\Rightarrow$adapts numerical iterative Krylov/Lanczos methods

$\Rightarrow\mathcal{O}\left(dE(n)+n^2\right)$ operations

Rank, Det, Solve: [Kaltofen & Saunders 90, Chen& Al. 02]

$\Rightarrow$reduced to minimal polynomial and preconditioners

$\Rightarrow\mathcal{O}\tilde{}\left(nE(n)+n^2\right)$ operations

where $E(n)$: cost of applying the matrix to a vector

Smith Normal Form: [Dumas & Al. 02] cf. J-G. Dumas talk

Dense matrices: FFLAS−FFPACK

Building block: **matrix mutlip. over word-size finite field**
Principle:

- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)

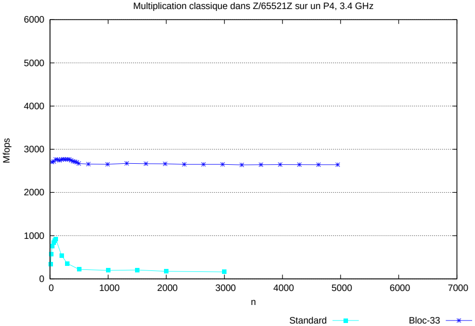
Dense matrices: FFLAS–FFPACK

Building block: **matrix mutlip. over word-size finite field**

Principle:

- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)
- cache tuning

⇒rely on the existing BLAS



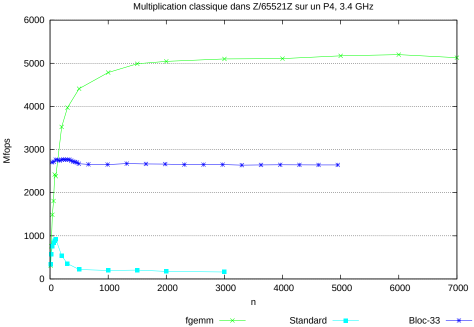
Dense matrices: FFLAS−FFPACK

Building block: **matrix mutlip. over word-size finite field**

Principle:

- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)
- cache tuning

⇒rely on the existing BLAS



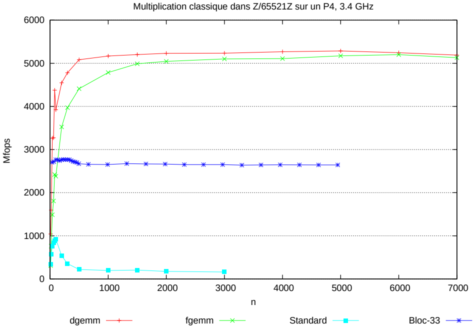
Dense matrices: FFLAS–FFPACK

Building block: **matrix mutlip. over word-size finite field**

Principle:

- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)
- cache tuning

⇒rely on the existing BLAS



Dense matrices: FFLAS−FFPACK

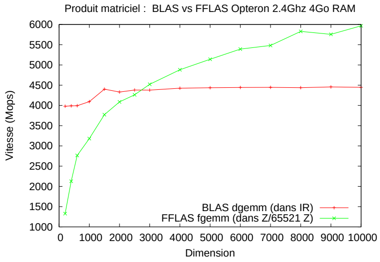
Building block: **matrix mutlip. over word-size finite field**

Principle:

- Delayed modular reduction
- Floating point arithmetic (fused-mac, SSE2, ...)
- cache tuning

⇒rely on the existing BLAS

- Sub-cubic algorithm (Winograd)



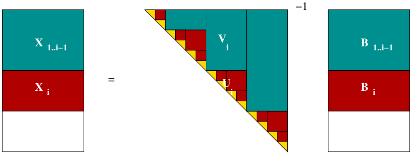
Design of other dense routines

- Reduction to matrix multiplication
- Bounds for delayed modular reductions.

Design of other dense routines

- Reduction to matrix multiplication
- Bounds for delayed modular reductions.

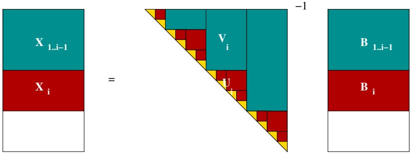
$\Rightarrow$Block algorithm with multiple cascade structures



Design of other dense routines

- Reduction to matrix multiplication
- Bounds for delayed modular reductions.

⇒Block algorithm with multiple cascade structures



	n	*1000*	*2000*	*3000*	*5000*	*10 000*
TRSM	$\frac{ftrsm}{dtrsm}$	1,66	1,33	1,24	1,12	1,01
LQUP	$\frac{lqup}{dgetrf}$	2,00	1,56	1,43	1,18	1,07
INVERSE	$\frac{inverse}{dgetrf+dgetri}$	1.62	1.32	1.15	0.86	0.76

Characteristic Polynomial:

n	500	5000	15 000
`LinBox`	0.91s	4m44s	2h20m
`magma-2.13`	1.27s	15m32s	7h28m

Sparse Matrices

Two approaches:

Blackbox:

- No fill-in,
- $E(n) = \mathcal{O}\left(\#\text{non-zero-elt}\right)$

Sparse elimination:

- local pivoting strategies
- switch to dense elimination when too much fill-in

Lifting over the integers

Multimodular reconstruction

- scalars and vectors
- early termination (with user-specified probability of success)
- or deterministic (e.g. Hadamard's bound)

p-adic lifting

dense matrices: Dixon's lifting with LU decomposition

blackbox/sparse matrices: no inverse nor LU can be computed

- Wiedemann lifter
- block-Wiedemann lifter
- block-Hankel lifter

Outline

Block Krylov projections

- Wiedemann algorithm: scalar projections of A^i for $i = 0..2d$:

$$u^T v, u^T Av, \ldots, u^T A^{2d/k} v \text{ such that } u, v \text{ are } n \times 1$$

- Block Wiedemann: $k \times k$ dense projections of A^i for $i = 1..2d/k$

$$U^T V, U^T AV, \ldots, U^T A^{2d/k} V, \text{ such that } U, V \text{ are } n \times k$$

Block Krylov projections

- Wiedemann algorithm: scalar projections of A^i for $i = 0..2d$:

$$u^T v, u^T Av, \ldots, u^T A^{2d/k} v \text{ such that } u, v \text{ are } n \times 1$$

- Block Wiedemann: $k \times k$ dense projections of A^i for $i = 1..2d/k$

$$U^T V, U^T AV, \ldots, U^T A^{2d/k} V, \text{ such that } U, V \text{ are } n \times k$$

- Building block of the most recent algorithmic advances
- In practice : better balance efficiency between Blackbox and dense methods

Packed matrices over small finite fields

GF(2): M4RI [Albrecht, Bard & Al.]

- Packed representation of elements: `long long` $\equiv GF(2)^{64}$
- Greasing technique: tables, and Gray codes
- SSE2 support and cache friendliness
- sub-cubic matrix arithmetic

Packed matrices over small finite fields

GF(2): M4RI [Albrecht, Bard & Al.]

- Packed representation of elements: `long long` $\equiv \mathrm{GF}(2)^{64}$
- Greasing technique: tables, and Gray codes
- SSE2 support and cache friendliness
- sub-cubic matrix arithmetic

$\mathrm{GF}(3,5,7)$: similar projects [Bradshaw, Boothby]

$\mathrm{GF}(p), p < 2^8$: Kronecker substitution [Dumas 2008]

$$(a_1, a_2, a_3) \to a_1 X^2 + a_2 X + a_2 \to \underbrace{a_1 \alpha^2 + a_2 \alpha + a_2}_{\text{integer on 64 bits}}$$

Packed matrices over small finite fields

GF(2): M4RI [Albrecht, Bard & Al.]

- Packed representation of elements: `long long` $\equiv \mathrm{GF}(2)^{64}$
- Greasing technique: tables, and Gray codes
- SSE2 support and cache friendliness
- sub-cubic matrix arithmetic

$\mathrm{GF}(3,5,7)$: similar projects [Bradshaw, Boothby]

$\mathrm{GF}(p), p < 2^8$: Kronecker substitution [Dumas 2008]

$$(a_1, a_2, a_3) \to a_1 X^2 + a_2 X + a_2 \to \underbrace{a_1 \alpha^2 + a_2 \alpha + a_2}_{\text{integer on 64 bits}}$$

$\Rightarrow$Matrices are no longer containers of field elements

Evolution and perspectives

LinBox 2.0 in the radar: major rewrite of the the library

- Clean up and simplify existing code
- Unify the usage block-Krylov/Wiedemann
- Redesign dense matrices (enabling packing for small finite fields)
- Support for new architecture framework : GPU, GPU/CPU, multi-core, grid computing...
  $\Rightarrow$Workstealing and adaptive scheduling libs: Cilk, Kaapi
- New algorithms...