The FFLAS-FFPACK and LinBox libraries
OpenDreamKit Software presentation

Clément Pernet & the LinBox group

September 2, 2015
Exact linear algebra

Matrices can be

- **Dense**: store all coefficients
- **Sparse**: store the non-zero coefficients only
- **Black-box**: no access to the storage, only apply to a vector
## Exact linear algebra

### Matrices can be

- **Dense**: store all coefficients
- **Sparse**: store the non-zero coefficients only
- **Black-box**: no access to the storage, only apply to a vector

### Coefficient domains:

- **Word size**:
  - integers with a priori bounds
  - $\mathbb{Z}/p\mathbb{Z}$ for $p$ of $\approx$ 32 bits
- **Multi-precision**: $\mathbb{Z}/p\mathbb{Z}$ for $p$ of $\approx$ 100, 200, 1000, 2000, ... bits
- **Arbitrary precision**: $\mathbb{Z}, \mathbb{Q}$
- **Polynomials**: $K[X]$ for $K$ any of the above
Exact linear algebra

Matrices can be

- **Dense**: store all coefficients
- **Sparse**: store the non-zero coefficients only
- **Black-box**: no access to the storage, only *apply* to a vector

Coefficient domains:

- **Word size**: integers with a priori bounds
  - \( \mathbb{Z}/p\mathbb{Z} \) for \( p \) of \( \approx 32 \) bits
- **Multi-precision**: \( \mathbb{Z}/p\mathbb{Z} \) for \( p \) of \( \approx 100, 200, 1000, 2000, \ldots \) bits
- **Arbitrary precision**: \( \mathbb{Z}, \mathbb{Q} \)
- **Polynomials**: \( K[\mathcal{X}] \) for \( K \) any of the above

Several implementations for the same domain: better fits FFT, LinAlg, etc
Exact linear algebra

Matrices can be

- **Dense:** store all coefficients
- **Sparse:** store the non-zero coefficients only
- **Black-box:** no access to the storage, only apply to a vector

Coefficient domains:

- **Word size:**
  - integers with a priori bounds
  - $\mathbb{Z}/p\mathbb{Z}$ for $p$ of $\approx 32$ bits

- **Multi-precision:** $\mathbb{Z}/p\mathbb{Z}$ for $p$ of $\approx 100, 200, 1000, 2000, \ldots$ bits

- **Arbitrary precision:** $\mathbb{Z}, \mathbb{Q}$

- **Polynomials:** $K[X]$ for $K$ any of the above

Several implementations for the same domain: better fits FFT, LinAlg, etc

Requires genericity.
Software stack for exact linear algebra

**Arithmetic**

- **GMP**: multiprecision integers and rationals
- **MPIR**: multiprecision integers and rationals
- **GAP**: finite fields and polynomials
- **NTL**: finite fields and polynomials
- **BLAS**: Basic Linear Algebra Subroutines (floating point)
- **FFLAS-FFPACK**: Basic Exact Linear Algebra over \( \mathbb{Z}/p\mathbb{Z} \)
- **LinBox**: Linear Algebra over \( \mathbb{Z} \), \( \mathbb{Z}/p\mathbb{Z} \) and \( \mathbb{K}[X] \)
Software stack for exact linear algebra

**Arithmetic**

- **GMP**, **MPIR**: multiprecision integers and rationals
- **C. Pernet**, **NTL**: finite fields and polynomials

**BLAS**: Basic Linear Algebra Subroutines (floating point)

**FFLAS-FFPACK**: Basic Exact Linear Algebra over $\mathbb{Z}/p\mathbb{Z}$,

**LinBox**: Linear Algebra over $\mathbb{Z}, \mathbb{Z}/p\mathbb{Z}$ and $K[X]$
Software stack for exact linear algebra

**Arithmetic**

- **GMP**, **MPIR**: multiprecision integers and rationals
- **GAP**, **NTL**: finite fields and polynomials
- **BLAS**: Basic Linear Algebra Subroutines (floating point)
- **FFLAS-FFPACK**: Basic Exact Linear Algebra over \( \mathbb{Z}/p\mathbb{Z} \)
- **LinBox**: Linear Algebra over \( \mathbb{Z}, \mathbb{Z}/p\mathbb{Z} \) and \( K[X] \)
Outline

1. The LinBox library
2. Blackbox linear algebra
3. Dense linear algebra
4. Parallelization
The LinBox project

- International collaboration: Canada, USA, France
- Strongly generic C++ code, focus on efficiency
- Free software (LGPL 2.1+)
- $\approx 200$ K loc
- [http://linalg.org/](http://linalg.org/)
The LinBox project

- International collaboration: Canada, USA, France
- Strongly generic C++ code, focus on efficiency
- Free software (LGPL 2.1+)
- \( \approx 200 \text{ K loc} \)
- http://linalg.org/

**Milestones**

- **1998** First design: Black box and sparse matrices
- **2003** Dense linear algebra using BLAS \( \leadsto \) FFLAS-FFPACK
- **2005** LinBox-1.0
- **2008** Integration in Sage
- **2012-..** Parallelization
- **2014** SIMD & Sparse BLAS in FFLAS-FFPACK (Brice’s talk)
Outline

1. The LinBox library
2. Blackbox linear algebra
3. Dense linear algebra
4. Parallelization
Black box linear algebra

Matrices viewed as linear operators

Algorithms based on matrix-vector apply only

Cost $E(n)$

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

Sparse matrices: Fast apply and no fill-in

Iterative methods

No access to coefficients, trace, no elimination

Matrix multiplication $\Rightarrow$ Black-box composition
Black box linear algebra

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

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

structured matrices: Fast apply (e.g. $E(n) = O(n \log n)$)

sparse matrices: Fast apply and no fill-in
Black box linear algebra

- Matrices viewed as linear operators
- Algorithms based on matrix-vector apply only \( \sim \) cost \( E(n) \)

\[ A \in K^{n \times n} \]
\[ v \in K^n \rightarrow A v \in K^n \]

Structured matrices: Fast apply (e.g. \( E(n) = O(n \log n) \))
Sparse matrices: Fast apply and no fill-in

\( \Rightarrow \)

- Iterative methods
- No access to coefficients, trace, no elimination
- Matrix multiplication \( \Rightarrow \) Black-box composition
Example: blackbox composition

template <class Mat1, class Mat2>
class Compose {
    protected:
        Mat1 _A;
        Mat2 _B;
    public:
        Compose(Mat1& A, Mat2& B) : _A(A), _B(B) {}

        template<class InVec, class OutVec>
        OutVec& apply (const InVec& x) {
            return _A.apply(_B.apply(x));
        }
};
Black box linear algebra

Matrix-Vector Product: building block,
~\rightarrow \text{costs } E(n)

Minimal polynomial: [Wiedemann 86]
~\rightarrow \text{iterative Krylov/Lanczos methods}
~\rightarrow O(nE(n) + n^2)


**Matrix-Vector Product:** building block, 
\[ \sim \text{ costs } E(n) \]

**Minimal polynomial:** [Wiedemann 86] 
\[ \sim \text{ iterative Krylov/Lanczos methods} \]
\[ \sim O(nE(n) + n^2) \]

**Rank, Det, Solve:** [Chen & Al. 02] 
\[ \sim \text{ reduces to MinPoly + preconditioners} \]
\[ \sim O\tilde{(nE(n) + n^2)} \]
Black box linear algebra

Matrix-Vector Product: building block,
\[ \leadsto \text{costs } E(n) \]

Minimal polynomial: [Wiedemann 86]
\[ \leadsto \text{iterative Krylov/Lanczos methods} \]
\[ \leadsto O(nE(n) + n^2) \]

Rank, Det, Solve: [Chen & Al. 02]
\[ \leadsto \text{reduces to MinPoly + preconditioners} \]
\[ \leadsto O(\tilde{n}E(n) + n^2) \]

Characteristic Poly.: [Dumas P. Saunders 09]
\[ \leadsto \text{reduces to MinPoly, Rank, …} \]
Matrix-Vector Product: building block, 
\[ \sim \text{costs } E(n) \]

Minimal polynomial: \[ \text{[Wiedemann 86]} \]
\[ \sim \text{iterative Krylov/Lanczos methods} \]
\[ \sim O(nE(n) + n^2) \]

Rank, Det, Solve: \[ \text{[Chen & Al. 02]} \]
\[ \sim \text{reduces to MinPoly + preconditioners} \]
\[ \sim O^*(nE(n) + n^2) \]

Characteristic Poly.: \[ \text{[Dumas P. Saunders 09]} \]
\[ \sim \text{reduces to MinPoly, Rank, . . .} \]
Outline

1. The LinBox library
2. Blackbox linear algebra
3. Dense linear algebra
4. Parallelization
Reductions: linear algebra’s arithmetic complexity

< 1969: \( O(n^3) \) for everyone (Gauss, Householder, Danilevskii, etc)
Reductions: linear algebra’s arithmetic complexity

< 1969: $O(n^3)$ for everyone (Gauss, Householder, Danilevskiï, etc)

Matrix Product

[Strassen 69]: $O(n^{2.807})$

...:

[Schönhage 81]: $O(n^{2.52})$

...:

[Coppersmith, Winograd 90]: $O(n^{2.375})$

...:

[Le Gall 14]: $O(n^{2.3728639})$

$\leadsto$ MM$(n) = O(n^\omega)$
### Reductions: linear algebra’s arithmetic complexity

< 1969: \(O(n^3)\) for everyone (Gauss, Householder, Danilevskii, etc)

#### Matrix Product

<table>
<thead>
<tr>
<th>Reference</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Strassen 69]</td>
<td>(O(n^{2.807}))</td>
</tr>
<tr>
<td>[Schönhage 81]</td>
<td>(O(n^{2.52}))</td>
</tr>
<tr>
<td>[Coppersmith, Winograd 90]</td>
<td>(O(n^{2.375}))</td>
</tr>
<tr>
<td>[Le Gall 14]</td>
<td>(O(n^{2.3728639}))</td>
</tr>
</tbody>
</table>

\(\Rightarrow\) \(\text{MM}(n) = O(n^\omega)\)

#### Other operations

<table>
<thead>
<tr>
<th>Reference</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Strassen 69]</td>
<td>Inverse in (O(n^\omega))</td>
</tr>
<tr>
<td>[Schönhage 72]</td>
<td>QR in (O(n^\omega))</td>
</tr>
<tr>
<td>[Bunch, Hopcroft 74]</td>
<td>LU in (O(n^\omega))</td>
</tr>
<tr>
<td>[Ibarra &amp; al. 82]</td>
<td>Rank in (O(n^\omega))</td>
</tr>
<tr>
<td>[Keller-Gehrig 85]</td>
<td>CharPoly in (O(n^\omega \log n))</td>
</tr>
</tbody>
</table>
Reductions

- HNF(\mathbb{Z})
- Det(\mathbb{Z})
- LinSys(\mathbb{Z})
- MM(\mathbb{Z})
- SNF(\mathbb{Z})
- Det(\mathbb{Z}_p)
- LU(\mathbb{Z}_p)
- CharPoly(\mathbb{Z}_p)
- MinPoly(\mathbb{Z}_p)
- TRSM(\mathbb{Z}_p)
- MM(\mathbb{Z}_p)

References:
- [P. and Storjohann 07]
- [Ibarra, Moran and Hui 82]
- [Jeannerod, P. and Storjohann 13]
- [Dumas, P. and Sultan 13]
Making theoretical reductions effective
Making theoretical reductions effective

Common mistrust

Fast linear algebra is

\( \times \) never faster

\( \times \) numerically unstable
Making theoretical reductions effective

Common mistrust
Fast linear algebra is
✗ never faster
✗ numerically unstable

Lucky coincidence
✓ building blocks in theory happen to be the most efficient routines in practice
⇝ reduction trees are still relevant
Making theoretical reductions effective

**Common mistrust**
- Fast linear algebra is never faster
- Fast linear algebra is numerically unstable

**Lucky coincidence**
- ✔ building blocks **in theory** happen to be the most efficient routines **in practice**
- ⇝ reduction trees are still relevant

**Roadmap**
1. Tune building blocks (MatMul)
2. Improve existing reductions (LU, Echelon)
   - leading constants
   - memory footprint
3. Produce new reduction schemes (CharPoly)
Matrix Multiplication over $\mathbb{Z}/p\mathbb{Z}$

**Ingredients [Dumas, Gautier and P. 02]**

- Compute over $\mathbb{Z}$ and delay modular reductions

$$k \left( \frac{p-1}{2} \right)^2 < 2^{\text{mantissa}}$$
Matrix Multiplication over $\mathbb{Z}/p\mathbb{Z}$

Ingredients [Dumas, Gautier and P. 02]

- Compute over $\mathbb{Z}$ and delay modular reductions

$$k \left( \frac{p-1}{2} \right)^2 < 2^{\text{mantissa}}$$

- Fastest integer arithmetic: double, float (SIMD and pipeline)

- Cache optimizations

$$\leadsto \text{numerical BLAS}$$
Matrix Multiplication over $\mathbb{Z}/p\mathbb{Z}$

Ingredients [Dumas, Gautier and P. 02]

- Compute over $\mathbb{Z}$ and delay modular reductions

  \[ 9^\ell \left\lfloor \frac{k}{2^\ell} \right\rfloor \left(\frac{p-1}{2}\right)^2 < 2^{\text{mantissa}} \]

- Fastest integer arithmetic: double, float (SIMD and pipeline)
- Cache optimizations

  \[ \sim \text{ numerical BLAS} \]

- Strassen-Winograd $6n^{2.807} + \ldots$
Matrix Multiplication over $\mathbb{Z}/p\mathbb{Z}$

**Ingredients [Dumas, Gautier and P. 02]**

- Compute over $\mathbb{Z}$ and delay modular reductions
  \[
  9^\ell \left\lfloor \frac{k}{2^\ell} \right\rfloor \left( \frac{p-1}{2} \right)^2 < 2^{\text{mantissa}}
  \]
- Fastest integer arithmetic: double, float (SIMD and pipeline)
- Cache optimizations
- Strassen-Winograd $6n^{2.807} + \ldots$

**with memory efficient schedules [Boyer, Dumas, P. and Zhou 09]**

**Tradeoffs:**

- Extra memory
- Overwriting input
- Leading constant
- Fully in-place in $7.2n^{2.807} + \ldots$
Sequential Matrix Multiplication

i5–3320M at 2.6Ghz with AVX 1

$2n^3/time/10^9 \text{ (Gfops equiv.)}$ vs matrix dimension

OpenBLAS sgemm
Sequential Matrix Multiplication

\[ p = 83, \sim 1 \mod / 10000 \text{ mul.} \]
Sequential Matrix Multiplication

$p = 83, \simmod 1 / 10000 \text{ mul.}$

$p = 821, \simmod 1 / 100 \text{ mul.}$
Sequential Matrix Multiplication

$p = 83, \equiv 1 \mod / 10000 \text{ mul.}$

$p = 821, \equiv 1 \mod / 100 \text{ mul.}$

$p = 1898131, \equiv 1 \mod / 10000 \text{ mul.}$

$p = 18981307, \equiv 1 \mod / 100 \text{ mul.}$
### LU decomposition

- **Block recursive algorithm** reduces to **MatMul** \( \sim O(n^\omega) \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
<th>15000</th>
<th>20000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK-dgetrf</td>
<td>0.024s</td>
<td>2.01s</td>
<td>14.88s</td>
<td>48.78s</td>
<td>113.66s</td>
</tr>
<tr>
<td>fflas-ffpack</td>
<td>0.058s</td>
<td>2.46s</td>
<td>16.08s</td>
<td>47.47s</td>
<td>105.96s</td>
</tr>
</tbody>
</table>

Intel Haswell E3-1270 3.0Ghz using OpenBLAS-0.2.9

---

**Dense linear algebra**

**Other routines**
### Other routines

#### LU decomposition

- Block recursive algorithm \(\leadsto\) reduces to MatMul \(\leadsto O(n^\omega)\)

<table>
<thead>
<tr>
<th>(n)</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
<th>15000</th>
<th>20000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK-dgetrf</td>
<td>0.024s</td>
<td>2.01s</td>
<td>14.88s</td>
<td>48.78s</td>
<td>113.66s</td>
</tr>
<tr>
<td>fflas-ffpack</td>
<td>0.058s</td>
<td>2.46s</td>
<td>16.08s</td>
<td>47.47s</td>
<td>105.96s</td>
</tr>
</tbody>
</table>

Intel Haswell E3-1270 3.0Ghz using OpenBLAS-0.2.9

#### Characteristic Polynomial

- A new reduction to matrix multiplication in \(O(n^\omega)\).

<table>
<thead>
<tr>
<th>(n)</th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>magma-v2.19-9</td>
<td>1.38s</td>
<td>24.28s</td>
<td>332.7s</td>
<td>2497s</td>
</tr>
<tr>
<td>fflas-ffpack</td>
<td>0.532s</td>
<td>2.936s</td>
<td>32.71s</td>
<td>219.2s</td>
</tr>
</tbody>
</table>

Intel Ivy-Bridge i5-3320 2.6Ghz using OpenBLAS-0.2.9
Other routines

LU decomposition

- Block recursive algorithm \(\leadsto\) reduces to MatMul \(\leadsto O(n^\omega)\)

<table>
<thead>
<tr>
<th>n</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
<th>15000</th>
<th>20000</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPACK-dgetrf</td>
<td>0.024s</td>
<td>2.01s</td>
<td>14.88s</td>
<td>48.78s</td>
<td>113.66s</td>
</tr>
<tr>
<td>f blas-ffpack</td>
<td>0.058s</td>
<td>2.46s</td>
<td>16.08s</td>
<td>47.47s</td>
<td>105.96s</td>
</tr>
</tbody>
</table>

Intel Haswell E3-1270 3.0Ghz using OpenBLAS-0.2.9

\[ \times 7.63 \quad \times 6.59 \]

Characteristic Polynomial

- A new reduction to matrix multiplication in \(O(n^\omega)\).

<table>
<thead>
<tr>
<th>n</th>
<th>1000</th>
<th>2000</th>
<th>5000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>magma-v2.19-9</td>
<td>1.38s</td>
<td>24.28s</td>
<td>332.7s</td>
<td>2497s</td>
</tr>
<tr>
<td>f blas-ffpack</td>
<td>0.532s</td>
<td>2.936s</td>
<td>32.71s</td>
<td>219.2s</td>
</tr>
</tbody>
</table>

Intel Ivy-Bridge i5-3320 2.6Ghz using OpenBLAS-0.2.9

\[ \times 7.5 \quad \times 6.7 \]
Outline

1. The LinBox library
2. Blackbox linear algebra
3. Dense linear algebra
4. Parallelization
Parallel matrix multiplication

A1
A2
B1 B2
C11 C12
C21 C22
1st recursion cutting
2nd recursion cutting

pfgemm over $\mathbb{Z}/131071\mathbb{Z}$ on a Xeon E5-4620 2.2Ghz 32 cores

C. Pernet
Parallelization

Gaussian elimination

Parallel PLUQ over double on full rank matrices on 32 cores

- FFLAS-FFPACK<double> explicit synch
- FFLAS-FFPACK<double> dataflow synch
- Intel MKL dgetrf
- PLASMA-Quark dgetrf tiled storage (k=212)
- PLASMA-Quark dgetrf (k=212)
Thank You.