High Performance and Reliable Algebraic Computing
Soutenance d’habilitation à diriger des recherches

Clément Pernet

Université Joseph Fourier (Grenoble 1)

November 25, 2014

Rapporteurs : Daniel Augot, Mark Giesbrecht, Laura Grigori,
Examinateurs : Jean-Guillaume Dumas, Jean-Charles Faugère, Erich L. Kaltofen, Brigitte Plateau.
Computer Algebra

Computing exactly over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}$, $\text{GF}(q)$, $K[X]$.

- Symbolic manipulations.
- Applications where all digits matter:

  - breaking Discrete Log Pb. in quasi-polynomial time [Barbulescu & al. 14],
  - building modular form databases to test the BSD conjecture [Stein 12],
  - formal verification of Hales’ proof of Kepler conjecture [Hales 05].
Introduction

Computer Algebra

Computing exactly over $\mathbb{Z}, \mathbb{Q}, \mathbb{Z}/p\mathbb{Z}, \mathbb{GF}(q), \mathbb{K}[X]$.

- Symbolic manipulations.
- Applications where all digits matter:
  - breaking Discrete Log Pb. in quasi-polynomial time [Barbulescu & al. 14],
  - building modular form databases to test the BSD conjecture [Stein 12],
  - formal verification of Hales’ proof of Kepler conjecture [Hales 05].

Efficiency mostly rely on linear algebra over $\mathbb{Z}$ and $\mathbb{Z}/p\mathbb{Z}$. 
Introduction

Coding theory

Protecting information against alteration:
- deep space communication,
- data storage,
- fault tolerance of large scale computations.
Introduction

Coding theory

Protecting information against alteration:
- deep space communication,
- data storage,
- fault tolerance of large scale computations.

Numerical linear algebra

Computing fast with approximations:
- delivering flops to most scientific computations for over 60 years,
- LinPack: benchmark for the top 500 supercomputers,
- impacts nowadays computer architectures.
Interactions

- Giorgi, Jeannerod and Villard 03
- Berlekamp 68, Massey 69
- Parity check, RS codes
- Krylov methods
- Wiedemann 86: sparse linear system solving over $F_q$
- Chowdhury & al. 14: fast list decoding of Reed-Solomon codes
- Huang and Abraham 84: Algorithm Based Fault Tolerance (ABFT)

C. Pernet (Habilitation defense)
Interactions

[Interactions]

[Berlekamp 68, Massey 69]

Krylov methods

[Wiedemann 86]: sparse linear system solving over $\mathbb{F}_q$

[Chowdhury & al. 14]: fast list decoding of Reed-Solomon codes

[Huang and Abraham 84]: Algorithm Based Fault Tolerance (ABFT)
Interactions

[Giorgi, Jeannerod and Villard 03]

[Berlekamp 68, Massey 69]

[Krylov methods]

[Wiedemann 86]: sparse linear system solving over $\mathbb{F}_q$

[Chowdhury & al. 14]: fast list decoding of Reed-Solomon codes
Interactions

Parity check, RS codes

[Giorgi, Jeannerod and Villard 03]

[Berlekamp 68, Massey 69]

Krylov methods

[Wiedemann 86]: sparse linear system solving over $\mathbb{F}_q$

[Chowdhury & al. 14]: fast list decoding of Reed-Solomon codes

[Huang and Abraham 84]: Algorithm Based Fault Tolerance (ABFT)
Interactions

Contributions:

- design of high performance linear algebra kernels,
Interactions

Contributions:

- design of high performance linear algebra kernels,
- fault tolerant computer algebra.
Outline

1. Design of High Performance Exact Linear Algebra Kernels
   - Matrix multiplication
   - Gaussian elimination
   - Rank profiles
   - Characteristic polynomial

2. Coding Theory for Fault Tolerant Computer Algebra
   - Approximation problems
   - Dense polynomial evaluation codes
   - Rational function codes
   - Sparse evaluation codes
Outline

1. Design of High Performance Exact Linear Algebra Kernels
   - Matrix multiplication
   - Gaussian elimination
   - Rank profiles
   - Characteristic polynomial

2. Coding Theory for Fault Tolerant Computer Algebra
   - Approximation problems
   - Dense polynomial evaluation codes
   - Rational function codes
   - Sparse evaluation codes
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, Danilevskii, 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]: \(O(n^{2.3728639})\)

\(\therefore \text{MM}(n) = O(n^\omega)\)
Reductions: linear algebra’s arithmetic complexity

< 1969: $O(n^3)$ for everyone (Gauss, Householder, Danilevskii, 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]: $O(n^{2.3728639})$

$\Rightarrow$ MM$(n) = O(n^\omega)$

Other operations

[Strassen 69]: Inverse in $O(n^\omega)$

[Schönhage 72]: QR in $O(n^\omega)$

[Bunch, Hopcroft 74]: LU in $O(n^\omega)$

[Ibarra & al. 82]: Rank in $O(n^\omega)$

[Keller-Gehrig 85]: CharPoly in $O(n^\omega \log n)$
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($\mathbb{Z}_p$)
- MatVecProd($\mathbb{Z}_p$)
- MM($\mathbb{Z}_p$)

References:
- Abbott, Bronstein and Mulders 99
- Ibarra, Moran and Hui 82
- Jeannerod, P. and Storjohann 13
- Dumas, P. and Sultan 13
- P. and Storjohann 07
- P. and Stein 10
- Storjohann 05
- Storjohann 07
- Storjohann 05
- Dumas, P. and Saunders 09
- Chen & al. 02
- Wiedemann 86
- Kaltofen Saunders 91
Design of High Performance Exact Linear Algebra Kernels

Reductions

- HNF($\mathbb{Z}$)
- SNF($\mathbb{Z}$)
- Det($\mathbb{Z}$)
  - [P. and Stein 10]
  - [Abbott, Bronstein and Mulders 99]
  - [Storjohann 05]
- LinSys($\mathbb{Z}$)
  - [Storjohann 05]
- MM($\mathbb{Z}$)
- SNF($\mathbb{Z}$)
- Det($\mathbb{Z}$)
- LinSys($\mathbb{Z}$)
  - [Storjohann 05]
- MM($\mathbb{Z}$)
- LinSys($\mathbb{Z}$)
  - [Storjohann 05]
- TRSM($\mathbb{Z}$)
  - [P. and Storjohann 07]
- LU($\mathbb{Z}$)
  - [Ibarra, Moran and Hui 82]
  - [Jeannerod, P. and Storjohann 13]
  - [Dumas, P. and Sultan 13]
- MM($\mathbb{Z}_p$)
  - [P. and Storjohann 07]
- Det($\mathbb{Z}_p$)
  - [P. and Storjohann 05]
- MinPoly($\mathbb{Z}_p$)
  - [Dumas, P. and Saunders 09]
- CharPoly($\mathbb{Z}_p$)
  - [Dumas, P. Saunders 09]
  - [Dumas, P. and Saunders 09]
- MinPoly($\mathbb{Z}_p$)
  - [Chen & al. 02]
  - [Wiedemann 86]
  - [Kaltofen Saunders 91]
- LinSys($\mathbb{Z}_p$)
- MatVecProd($\mathbb{Z}_p$)
  - [Wiedemann 86]
Reductions

- \( \text{HNF}(\mathbb{Z}) \)
- \( \text{Det}(\mathbb{Z}) \)
- \( \text{LinSys}(\mathbb{Z}) \)
- \( \text{SNF}(\mathbb{Z}) \)
- \( \text{MM}(\mathbb{Z}) \)

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

- \( \text{CharPoly}(\mathbb{Z}) \)
- \( \text{MinPoly}(\mathbb{Z}) \)
- \( \text{MinPoly}(\mathbb{Z}_p) \)
- \( \text{Rank}(\mathbb{Z}_p) \)
- \( \text{MatVecProd}(\mathbb{Z}_p) \)

References:
- Abbott, Bronstein and Mulders 99
- Storjohann 05
- P. and Stein 10
- Storjohann 07
- Storjohann 05
- Dumas, P. and Saunders 09
- Chen & al. 02
- Wiedemann 86
- Kaltofen Saunders 91
- Jeannerod, P. and Storjohann 13
- Dumas, P. and Sultan 13
Making theoretical reductions effective
Making theoretical reductions effective

Common mistrust

Fast linear algebra is

✗ never faster
✗ 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
- 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, Rank Profiles)
Design of parallel exact linear algebra

ANR HPAC project:

1. efficient kernels for exact linear algebra on SMP
2. DSL, runtime as a plugin and composition
3. attacking large scale challenges from cryptography
Design of parallel exact linear algebra

ANR HPAC project: Ziad Sultan PhD. Thesis

1. efficient kernels for exact linear algebra on SMP
2. DSL, runtime as a plugin and composition
3. attacking large scale challenges from cryptography
Design of parallel exact linear algebra

ANR HPAC project: Ziad Sultan PhD. Thesis

1. efficient kernels for exact linear algebra on SMP
2. DSL, runtime as a plugin and composition
3. attacking large scale challenges from cryptography
Design of parallel exact linear algebra

ANR HPAC project: Ziad Sultan PhD. Thesis

1. efficient kernels for exact linear algebra on SMP
2. DSL, runtime as a plugin and composition
3. attacking large scale challenges from cryptography

Parallel numerical linear algebra

- cost invariant wrt. splitting
  - $O(n^3)$
  - $\leadsto$ fine grain
  - $\leadsto$ block iterative algorithms
- regular task load
- Numerical stability constraints
Design of parallel exact linear algebra

**ANR HPAC project:**

1. Efficient kernels for exact linear algebra on SMP
2. DSL, runtime as a plugin and composition
3. Attacking large scale challenges from cryptography

**Parallel numerical linear algebra**

- Cost invariant wrt. splitting
  - $O(n^3)$
  - $\leadsto$ fine grain
  - $\leadsto$ block iterative algorithms
- Regular task load
- Numerical stability constraints

**Exact linear algebra specificities**

- Cost affected by the splitting
  - $O(n^\omega)$ for $w < 3$
  - Modular reductions
  - $\leadsto$ coarse grain
  - $\leadsto$ recursive algorithms
- Rank deficiencies
  - $\leadsto$ unbalanced task loads

---

[Ziad Sultan PhD. Thesis]

---

[Broquedis, Danjean and Gautier 12]: libkomp based on XKaapi
Design of parallel exact linear algebra

**ANR HPAC project:**

1. Efficient kernels for exact linear algebra on SMP
2. DSL, runtime as a plugin and composition
3. Attacking large scale challenges from cryptography

---

**Parallel numerical linear algebra**

- Cost invariant wrt. splitting
  - $O(n^3)$
  - $\leadsto$ fine grain
  - $\leadsto$ block iterative algorithms
- Regular task load
- Numerical stability constraints

**Exact linear algebra specificities**

- Cost affected by the splitting
  - $O(n^\omega)$ for $w < 3$
  - Modular reductions
  - $\leadsto$ coarse grain
  - $\leadsto$ recursive algorithms
- Rank deficiencies
  - $\leadsto$ unbalanced task loads

---

[Broquedis, Danjean and Gautier 12]: libkomp based on XKaapi
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

  \( \rightsquigarrow \) 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

- Strassen-Winograd $6n^{2.807} + \ldots$

- Tradeoffs: Extra memory, Overwriting input, Leading constant
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 $6n^{2.807} + \ldots$

**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

$2n^3/time/10^9$ (Gflops equiv.)

matrix dimension

OpenBLAS sgemm

C. Pernet (Habilitation defense)  High Perf. and Reliable Algebraic Computing  November 25, 2014  13 / 39
Sequential Matrix Multiplication

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

\[ p = 83, \mod \equiv 1 \mod 10000 \text{ mul.} \]

\[ p = 821, \mod \equiv 1 \mod 100 \text{ mul.} \]
Sequential Matrix Multiplication

\[ i5-3320M \text{ at } 2.6\text{Ghz with AVX 1} \]

\[ 2n^3/\text{time}/10^9 \text{ (Gfops equiv.)} \]

- \( p = 83, \sim 1 \mod / 10000 \text{ mul.} \)
- \( p = 821, \sim 1 \mod / 100 \text{ mul.} \)
- \( p = 1898131, \sim 1 \mod / 10000 \text{ mul.} \)
- \( p = 18981307, \sim 1 \mod / 100 \text{ mul.} \)
Parallel matrix multiplication

Dumas, Gautier, P. and Sultan 14

![Matrix multiplication diagram](image)

**Graph:**
- **x-axis:** matrix dimension
- **y-axis:** Gflops
- **Graph title:** pfgemm over Z/131071Z on a Xeon E5-4620 2.2Ghz 32 cores
- **Legend:**
  - MKL dgemm
  - PLASMA-QUARK dgemm
Parallel matrix multiplication

Dumas, Gautier, P. and Sultan 14
Parallel matrix multiplication

Dumas, Gautier, P. and Sultan 14

![Matrix multiplication diagram]

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

- pfgemm double
- pfgemm mod 131071
- MKL dgemm
- PLASMA-QUARK dgemm

Gflops vs. matrix dimension graph
Gaussian elimination

- Slab iterative
  - LAPACK
- Slab recursive
  - FFLAS-FFPACK
- Tile iterative
  - PLASMA
- Tile recursive
  - FFLAS-FFPACK

Preferences:
- Recursive algorithms
- Better data locality
Gaussian elimination

- Prefer recursive algorithms

Slab recursive
FFLAS–FFPACK

Tile recursive
FFLAS–FFPACK
Gaussian elimination

- Prefer recursive algorithms
- Better data locality
Full rank Gaussian elimination

Dumas, Gautier, P. and Sultan 14
Comparing numerical efficiency (no modulo)
Full rank Gaussian elimination

Dumas, Gautier, P. and Sultan 14
Comparing numerical efficiency (no modulo)
Full rank Gaussian elimination

Dumas, Gautier, P. and Sultan 14
Comparing numerical efficiency (no modulo)
Full rank Gaussian elimination

Dumas, Gautier, P. and Sultan 14

Over the finite field $\mathbb{Z}/131071\mathbb{Z}$

Parallel PLUQ over $\mathbb{Z}/131071\mathbb{Z}$ with full rank matrices on 32 cores

Gfops vs. matrix dimension

Tiled Rec explicit sync
Tiled Rec dataflow sync
Full rank Gaussian elimination

Dumas, Gautier, P. and Sultan 14
Over the finite field \( \mathbb{Z}/131071\mathbb{Z} \)

Parallel PLUQ over \( \mathbb{Z}/131071\mathbb{Z} \) with full rank matrices on 32 cores

- Tiled Rec explicit sync
- Tiled Rec dataflow sync
- Tiled Iter dataflow sync
- Tiled Iter explicit sync
Rank profiles

**Definition (Row Rank Profile: RowRP)**

Given $A \in K^{m \times n}, r = \text{rank}(A)$.

*informally:* first $r$ linearly independent rows

*formally:* lexico-minimal sub-sequence of $(1, \ldots, m)$ of $r$ indices of linearly independant rows.

**Example**

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

Rank = 3

RowRP = \{1, 2, 4\}

ColRP = \{1, 2, 3\}
Rank profiles

**Definition (Row Rank Profile: RowRP)**

Given $A \in \mathbb{K}^{m \times n}$, $r = \text{rank}(A)$.

- **informally:** *first* $r$ linearly independent rows
- **formally:** lexico-minimal sub-sequence of $(1, \ldots, m)$ of $r$ indices of linearly independant rows.

**Example**

$$
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
$$

Rank = 3
RowRP = \{1, 2, 4\}
Rank profiles

**Definition (Row Rank Profile: RowRP)**

Given $A \in K^{m \times n}$, $r = \text{rank}(A)$.

*informally:* first $r$ linearly independent rows

*formally:* lexico-minimal sub-sequence of $(1, \ldots, m)$ of $r$ indices of linearly indpendant rows.

**Example**

$$
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\end{bmatrix}
$$

Rank = 3

RowRP = \{1, 2, 4\}

ColRP = \{1, 2, 3\}
Rank profiles

**Definition (Row Rank Profile: RowRP)**

Given $A \in K^{m \times n}$, $r = \text{rank}(A)$.

- **informally**: first $r$ linearly independent rows
- **formally**: lexi-co-minimal sub-sequence of $(1, \ldots, m)$ of $r$ indices of linearly independant rows.

**Example**

```
[1 1 0 0]
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
```

Rank = 3
RowRP = \{1,2,4\}
ColRP = \{1,2,3\} → Generic ColRP.
Rank profiles

**Definition (Row Rank Profile: RowRP)**

Given $A \in K^{m \times n}$, $r = \text{rank}(A)$.

- **informally**: *first* $r$ linearly independent rows
- **formally**: lexico-minimal sub-sequence of $(1, \ldots, m)$ of $r$ indices of linearly independant rows.

**Example**

\[
\begin{bmatrix}
1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

Rank = 3

RowRP = \{1,2,4\}

ColRP = \{1,2,3\} → Generic ColRP.

- Major invariant of a matrix (echelon form)
- Gröbner basis computations (Macaulay matrix) [Faugère 99, 02]
- Krylov methods
Computing rank profiles

Via Gaussian elimination revealing echelon forms:

[Ibarra, Moran and Hui 82]

[Keller-Gehrig 85]

[Storjohann 00]

[Jeannerod, P. and Storjohann 13]
Computing rank profiles

Via Gaussian elimination revealing echelon forms:

[Ibarra, Moran and Hui 82]

[Keller-Gehrig 85]

[Storjohann 00]

[Jeannerod, P. and Storjohann 13]

Lessons learned (or what we thought was necessary):

- treat rows in order
- exhaust all columns before considering the next row
- slab block splitting required (recursive or iterative)
  \( \rightsquigarrow \) similar to partial pivoting
Pivoting strategies revealing rank profiles

Pivot Search

Pivot’s \((i, j)\) position minimizes some pre-order:

- **Row order**: any non-zero on the first non-zero row

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pivoting strategies revealing rank profiles

Pivot Search

Pivot's \((i,j)\) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pivoting strategies revealing rank profiles

Pivot Search

Pivot’s \((i, j)\) position minimizes some pre-order:

- **Row/Col order**: any non-zero on the first non-zero row/col
- **Lex order**: first non-zero on the first non-zero row

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rotation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

C. Pernet (Habilitation defense) High Perf. and Reliable Algebraic Computing November 25, 2014 19 / 39
Pivoting strategies revealing rank profiles

Pivot Search

Pivot’s \((i, j)\) position minimizes some pre-order:

- **Row/Col order:** any non-zero on the first non-zero row/col
- **Lex/RevLex order:** first non-zero on the first non-zero row/col

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pivoting strategies revealing rank profiles

Pivot Search
Pivot's \((i, j)\) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col

Lex/RevLex order: first non-zero on the first non-zero row/col

Product order: first non-zero in the \((i, j)\) leading sub-matrix

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pivoting strategies revealing rank profiles

Pivot Search

Pivot's \((i,j)\) position minimizes some pre-order:

**Row/Col order:** any non-zero on the first non-zero row/col

**Lex/RevLex order:** first non-zero on the first non-zero row/col

**Product order:** first non-zero in the \((i,j)\) leading sub-matrix

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>CoIRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td>✓</td>
<td>X</td>
</tr>
</tbody>
</table>

C. Pernet (Habilitation defense)
## Pivoting strategies revealing rank profiles

### Pivot Search

Pivot's \((i, j)\) position minimizes some pre-order:

- **Row/Col order:** any non-zero on the first non-zero row/col
- **Lex/RevLex order:** first non-zero on the first non-zero row/col
- **Product order:** first non-zero in the \((i, j)\) leading sub-matrix

### Permutation

- **Transpositions**
- **Cyclic Rotations**

### Table Summary

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Transposition</td>
<td>Rotation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pivoting strategies revealing rank profiles

Pivot Search

Pivot's \((i, j)\) position minimizes some pre-order:

Row/Col order: any non-zero on the first non-zero row/col

Lex/RevLex order: first non-zero on the first non-zero row/col

Product order: first non-zero in the \((i, j)\) leading sub-matrix

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>All RPs</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td>✓</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Transposition</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

C. Pernet (Habilitation defense)
High Perf. and Reliable Algebraic Computing
November 25, 2014 19 / 39
### Pivoting strategies revealing rank profiles

#### Pivot Search

Pivot’s \((i, j)\) position minimizes some pre-order:

- **Row/Col order:** any non-zero on the first non-zero row/col
- **Lex/RevLex order:** first non-zero on the first non-zero row/col
- **Product order:** first non-zero in the \((i, j)\) leading sub-matrix

### Permutation

- Transpositions
- Cyclic Rotations

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>CoIRP</th>
<th>All RPs</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Transposition</td>
<td>Rotation</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
### Pivoting strategies revealing rank profiles

#### Pivot Search

Pivot's \((i, j)\) position minimizes some pre-order:

- **Row/Col order:** any non-zero on the first non-zero row/col
- **Lex/RevLex order:** first non-zero on the first non-zero row/col
- **Product order:** first non-zero in the \((i, j)\) leading sub-matrix

#### Permutation

- Transpositions
- Cyclic Rotations

<table>
<thead>
<tr>
<th>Search</th>
<th>Row perm.</th>
<th>Col. perm.</th>
<th>RowRP</th>
<th>ColRP</th>
<th>All RPs</th>
<th>Tiles possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Col. order</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Lexico.</td>
<td>Transposition</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Transposition</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Rev. lex.</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Transposition</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Transposition</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Product</td>
<td>Rotation</td>
<td>Rotation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique \( \mathcal{R}_A \in \{0, 1\}^{m \times n} \) such that any pair of \( (i, j) \)-leading sub-matrix of \( \mathcal{R}_A \) and of \( A \) have the same rank.

\[ A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 5 & 8 \\ 1 & 2 & 3 & 4 \\ 3 & 5 & 9 & 12 \end{bmatrix}, \quad \mathcal{R} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \]
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique $R_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $R_A$ and of $A$ have the same rank.

Theorem

- RowRP and ColRP read directly on $R(A)$
- Same holds for any $(i, j)$-leading submatrix.

RowRP = $\{1\}$
ColRP = $\{1\}$
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique $R_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $R_A$ and of $A$ have the same rank.

Theorem

- RowRP and ColRP read directly on $R(A)$
- Same holds for any $(i, j)$-leading submatrix.

RowRP = $\{1,2\}$
ColRP = $\{1,3\}$

$A = \begin{bmatrix}
1 & 2 & 3 & 4 \\
2 & 4 & 5 & 8 \\
1 & 2 & 3 & 4 \\
3 & 5 & 9 & 12
\end{bmatrix}$

$R = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}$
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique $R_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $R_A$ and of $A$ have the same rank.

Theorem

- RowRP and ColRP read directly on $R(A)$
- Same holds for any $(i, j)$-leading submatrix.

RowRP = \{1,4\}
ColRP = \{1,2\}
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique $R_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $R_A$ and of $A$ have the same rank.

Theorem

- RowRP and ColRP read directly on $R(A)$
- Same holds for any $(i, j)$-leading submatrix.

$$A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U & V \\ I_{n-r} & 0 \end{bmatrix} Q$$
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique $\mathcal{R}_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $\mathcal{R}_A$ and of $A$ have the same rank.

Theorem

- RowRP and ColRP read directly on $\mathcal{R}(A)$
- Same holds for any $(i, j)$-leading submatrix.

$$A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T P \begin{bmatrix} I_r & 0 \\ & 0 \end{bmatrix} QQ^T \begin{bmatrix} U & V \\ & I_{n-r} \end{bmatrix} Q$$

RowRP = $\{1,4\}$
ColRP = $\{1,2\}$
Computing all rank profiles at once

Dumas, P. and Sultan 13

Definition (Rank Profile matrix)

The unique $\mathcal{R}_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $\mathcal{R}_A$ and of $A$ have the same rank.

Theorem

- **RowRP and ColRP read directly on** $\mathcal{R}(A)$
- **Same holds for any** $(i, j)$-leading submatrix.

$$A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T P \begin{bmatrix} I_r & 0 \\ \Pi_{P,Q} & U \end{bmatrix} Q Q^T \begin{bmatrix} U & V \\ I_{n-r} \end{bmatrix} Q$$

RowRP = $\{1, 4\}$
ColRP = $\{1, 2\}$
Computing all rank profiles at once

Dumas, P. and Sultan 13

**Definition (Rank Profile matrix)**

The unique $\mathcal{R}_A \in \{0, 1\}^{m \times n}$ such that any pair of $(i, j)$-leading sub-matrix of $\mathcal{R}_A$ and of $A$ have the same rank.

**Theorem**

- **RowRP and ColRP read directly on $\mathcal{R}(A)$**
- **Same holds for any $(i, j)$-leading submatrix.**

\[ A = PLUQ = P \begin{bmatrix} L & 0 \\ M & I_{m-r} \end{bmatrix} P^T \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix} Q Q^T \begin{bmatrix} U & V \\ I_{n-r} \end{bmatrix} Q \]

With appropriate pivoting: $\Pi_{P,Q} = \mathcal{R}(A)$
A tiled recursive algorithm

Dumas, P. and Sultan 13

2 × 2 block splitting
A tiled recursive algorithm

Recursive call

Dumas, P. and Sultan 13
A tiled recursive algorithm

Dumas, P. and Sultan 13

\[
\text{TRSM: } B \leftarrow BU^{-1}
\]
A tiled recursive algorithm

Dumas, P. and Sultan 13

TRSM: $B \leftarrow L^{-1}B$
A tiled recursive algorithm

Dumas, P. and Sultan 13

MatMul: $C \leftarrow C - A \times B$
A tiled recursive algorithm

Dumas, P. and Sultan 13

MatMul: $C \leftarrow C - A \times B$
A tiled recursive algorithm

Dumas, P. and Sultan 13

MatMul: $C \leftarrow C - A \times B$
A tiled recursive algorithm

Dumas, P. and Sultan 13

2 independent recursive calls
A tiled recursive algorithm

Dumas, P. and Sultan 13

TRSM: $B \leftarrow BU^{-1}$
A tiled recursive algorithm

Dumas, P. and Sultan 13

**TRSM:** $B \leftarrow L^{-1}B$
A tiled recursive algorithm

Dumas, P. and Sultan 13

MatMul: $C \leftarrow C - A \times B$
A tiled recursive algorithm

Dumas, P. and Sultan 13

\[
\text{MatMul: } C \leftarrow C - A \times B
\]
A tiled recursive algorithm

Dumas, P. and Sultan 13

\[
\text{MatMul: } C \leftarrow C - A \times B
\]
A tiled recursive algorithm

Dumas, P. and Sultan 13

Recursive call
A tiled recursive algorithm

Dumas, P. and Sultan 13

Puzzle game (block cyclic rotations)
A tiled recursive algorithm

Dumas, P. and Sultan 13

- $O(mnr^{\omega-2})$ (degenerating to $2/3n^3$)
- computing col. and row rank profiles of all leading sub-matrices
- fewer modular reductions than slab algorithms
- rank deficiency introduces parallelism
Computing the characteristic polynomial

Motivation

- Connection with the Frobenius normal form
- Krylov methods at large
- Graph invariants
- Crucial step in modular form computations

The last missing reduction

- [Danilevskii 37], [Hessenberg 42] CharPoly, deterministic \( O(n^3) \)
- [Keller-Gehrig 85] CharPoly, deterministic \( O(n^\omega \log n) \)
- [Giesbrecht 93] Frobenius form, Las-Vegas probabilistic \( O(n^\omega \log n) \)
- [Augot, Camion 94] Frobenius form, deterministic \( O(n^3 \#\text{inv factors}) \)
- [Storjohann 00] Frobenius form, deterministic \( O(n^3) \) or \( O(n^\omega \log n \log \log n) \)
Characteristic polynomial

$k$-shifted form:

\[
\begin{array}{cccc}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
\end{array}
\]

\[k \leq k\]
Characteristic polynomial

\[ \begin{align*}
&\text{\textit{k} + 1-shifted form:} \\
&\begin{array}{ccc}
0 & 1 & \ldots \\
\vdots & \ddots & \ddots \\
& & \ddots \\
& & & 0 & 1 \\
& & & \vdots & \ddots \\
& & & & \ddots & \ddots \\
& & & & & 0 & 1 \\
& & & & & \vdots & \ddots \\
& & & & & & \ddots & \ddots \\
& & & & & & & 0 & 1 \\
& & & & & & & \vdots & \ddots \\
& & & & & & & & \ddots & \ddots \\
& & & & & & & & & 0 & 1 \\
& & & & & & & & & \vdots & \ddots \\
& & & & & & & & & & \ddots & \ddots \\
& & & & & & & & & & & 0 & 1 \\
& & & & & & & & & & & \vdots & \ddots \\
& & & & & & & & & & & & \ddots & \ddots \\
\end{array}
\end{align*} \]

From \( k \) to \( k + 1 \)-shifted in \( O(n^\omega - 1) \)

- Compute iteratively from a \( 1 \)-shifted form
- Invariant factors appear by increasing degree
- Until the Hessenberg polycyclic form

\[ \begin{align*}
&n^\omega \sum_{k=1}^{n^\omega} (1^k) \omega - 1 \leq \zeta(\omega - 1) n^\omega = O(n^\omega) \\
&\text{Generalized to the Frobenius form as well}
\end{align*} \]

- Transformation matrix in \( O(n^\omega \log \log n) \)

\begin{tabular}{c|c|c|c}
\hline
magma-v2.19-9 & 1.38s & 24.28s & 332.7s \\
fflas-ffpack & 0.532s & 2.936s & 32.71s \\
\hline
\end{tabular}
\( k + 1 \)-shifted form:

\[
\begin{array}{cccc}
0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{array}
\]

- From \( k \) to \( k + 1 \)-shifted in \( O(n(\frac{n}{k})^{\omega-1}) \)
- Compute iteratively from a \( 1 \)-shifted form
- Invariant factors appear by increasing degree
Hessenberg polycyclic:

- From $k$ to $k + 1$-shifted in $O(n\left(\frac{n}{k}\right)^{\omega-1})$
- Compute iteratively from a 1-shifted form
- Invariant factors appear by increasing degree
- Until the Hessenberg polycyclic form

\[
\begin{array}{cccc}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0
\end{array}
\]
From $k$ to $k + 1$-shifted in $O\left(n\left(\frac{n}{k}\right)^{\omega-1}\right)$

- Compute iteratively from a $1$-shifted form
- Invariant factors appear by increasing degree
- Until the Hessenberg polycyclic form

$$n^\omega \sum_{k=1}^{n} \left(\frac{1}{k}\right)^{\omega-1} \leq \zeta(\omega - 1)n^\omega = O(n^\omega)$$
Hessenberg polycyclic:

- From $k$ to $k+1$-shifted in $O(n \left(\frac{n}{k}\right)^{\omega-1})$
- Compute iteratively from a 1-shifted form
- Invariant factors appear by increasing degree
- Until the Hessenberg polycyclic form

\[
n^\omega \sum_{k=1}^{n} \left(\frac{1}{k}\right)^{\omega-1} \leq \zeta(\omega - 1)n^{\omega} = O(n^{\omega})
\]

- Generalized to the Frobenius form as well
- Transformation matrix in $O(n^{\omega} \log \log n)$
From $k$ to $k + 1$-shifted in $O(n \left(\frac{n}{k}\right)^{\omega-1})$

Compute iteratively from a 1-shifted form

Invariant factors appear by increasing degree

Until the Hessenberg polycyclic form

$$n^{\omega} \sum_{k=1}^{n} \left(\frac{1}{k}\right)^{\omega-1} \leq \zeta(\omega - 1)n^{\omega} = O(n^{\omega})$$

Generalized to the Frobenius form as well

Transformation matrix in $O(n^{\omega} \log \log n)$

<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>
Hessenberg polycyclic:

- From $k$ to $k+1$-shifted in $O(n\left(\frac{n}{k}\right)^{\omega-1})$
- Compute iteratively from a 1-shifted form
- Invariant factors appear by increasing degree
- Until the Hessenberg polycyclic form

\[ n^\omega \sum_{k=1}^{n} \left(\frac{1}{k}\right)^{\omega-1} \leq \zeta(\omega - 1)n^\omega = O(n^\omega) \]

- Generalized to the Frobenius form as well
- Transformation matrix in $O(n^\omega \log \log n)$

<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>ff1as-ffpack</td>
<td>0.532s</td>
<td>2.936s</td>
<td>32.71s</td>
<td>219.2s</td>
</tr>
</tbody>
</table>

C. Pernet (Habilitation defense) High Perf. and Reliable Algebraic Computing November 25, 2014 23 / 39
Outline

1. Design of High Performance Exact Linear Algebra Kernels
   - Matrix multiplication
   - Gaussian elimination
   - Rank profiles
   - Characteristic polynomial

2. Coding Theory for Fault Tolerant Computer Algebra
   - Approximation problems
   - Dense polynomial evaluation codes
   - Rational function codes
   - Sparse evaluation codes
## Fault Tolerance

### Reliability of large scale distributed computing

<table>
<thead>
<tr>
<th>System</th>
<th>Peak</th>
<th>Mean Time To Error</th>
<th>Mean Time To Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Waters</td>
<td>14 Pflops</td>
<td>15min</td>
<td>≈ 1/2 day</td>
</tr>
<tr>
<td>Tsubame 2</td>
<td>2.3 Pflops</td>
<td>?</td>
<td>15.8h</td>
</tr>
</tbody>
</table>

- Disk crash, hardware/software failures $\leadsto$ hard errors
- Bitflip in main or cache memory $\leadsto$ soft/silent errors
Fault Tolerance

Reliability of large scale distributed computing

<table>
<thead>
<tr>
<th>System</th>
<th>Peak</th>
<th>Mean Time To Error</th>
<th>Mean Time To Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Waters</td>
<td>14 Pflops</td>
<td>15 min</td>
<td>≈ 1/2 day</td>
</tr>
<tr>
<td>Tsubame 2</td>
<td>2.3 Pflops</td>
<td>?</td>
<td>15.8h</td>
</tr>
</tbody>
</table>

- Disk crash, hardware/software failures → hard errors
- Bitflip in main or cache memory → soft/silent errors
Fault Tolerance

Reliability of large scale distributed computing

<table>
<thead>
<tr>
<th></th>
<th>Peak</th>
<th>Mean Time To Error</th>
<th>Mean Time To Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Waters</td>
<td>14 Pflops</td>
<td>15min</td>
<td>≈ 1/2 day</td>
</tr>
<tr>
<td>Tsubame 2</td>
<td>2.3 Pflops</td>
<td>?</td>
<td>15.8h</td>
</tr>
</tbody>
</table>

- Disk crash, hardware/software failures \(\leadsto\) hard errors
- Bitflip in main or cache memory \(\leadsto\) soft/silent errors

Trust in outsourced computations (P2P, Cloud, Volunteer, etc)

Byzantine error model:

- a corrupted node is not always wrong
- black-listing is not an option
Fault Tolerance

Reliability of large scale distributed computing

<table>
<thead>
<tr>
<th></th>
<th>Peak</th>
<th>Mean Time To Error</th>
<th>Mean Time To Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Waters</td>
<td>14 Pflops</td>
<td>15min</td>
<td>≈ 1/2 day</td>
</tr>
<tr>
<td>Tsubame 2</td>
<td>2.3 Pflops</td>
<td>?</td>
<td>15.8h</td>
</tr>
</tbody>
</table>

- Disk crash, hardware/software failures \(\mapsto\) hard errors
- Bitflip in main or cache memory \(\mapsto\) soft/silent errors

Trust in outsourced computations (P2P, Cloud, Volunteer, etc)

Byzantine error model:

- a corrupted node is not always wrong
- black-listing is not an option

Algorithm Based Fault Tolerance:

exploit the algebraic specificity of the algorithm to embed redundancy.
ABFT using error correcting codes

Computations ⇔ Communication

Unsecure Computations ⇔ Noisy Communication

Choice of the parallelization algorithm determines

▶ the communication channel
▶ the error model
ABFT using error correcting codes

Unsecure Computations ⇔ Noisy Communication

Choice of the parallelization algorithm determines
- the communication channel
- the error model
Evaluation-interpolation schemes

Polynomial evaluation

\[ \text{Ev}(x_0, \ldots, x_{n-1}) : \ K^{<n}[X] \longrightarrow K^n \]

\[ f \mapsto (f(x_0), \ldots, f(x_{n-1})) \]

for \( x_0, \ldots, x_{n-1} \) distinct.
Evaluation-interpolation schemes

Polynomial evaluation

\[ \text{Ev}^{(x_0,\ldots,x_{n-1})} : \ K^<_n[X] \longrightarrow K^n \]

\[ f \longmapsto (f(x_0), \ldots, f(x_{n-1})) \]

for \( x_0, \ldots, x_{n-1} \) distinct.

\[ A \in K[X]^{m \times m} \]

\[ \det(A) \]

\[ A(x_0) \]

\[ A(x_{n-1}) \]

\[ d_0 \]

\[ d_{n-1} \]
Evaluation-interpolation schemes

**Chinese Remainder Theorem**

\[
\text{Ev}_{(p_0,...,p_{n-1})} : \mathbb{Z}^{\langle p_0 \times \cdots \times p_{n-1}} \longrightarrow \mathbb{Z}_{p_0} \times \cdots \times \mathbb{Z}_{p_{n-1}}
\]

\[
m \mapsto (m \mod p_0, \ldots, m \mod p_{n-1})
\]

for \( p_1, \ldots, p_n \) pairwise co-prime.
Making evaluation-interpolation schemes fault tolerant

\[ x_i \in F \quad \rightarrow \quad f(?) \quad \rightarrow \quad f(x_i) \]

**Problem**

Recover an unknown function \( f \), given as a black-box, from its evaluations.
Making evaluation-interpolation schemes fault tolerant

\[ x_i \in F \quad \Rightarrow \quad f(x_i) \quad \Rightarrow \quad f = \sum_{i=0}^{d_f} c_i X^i \]

**Problem**

Recover an unknown function \( f \), given as a black-box, from its evaluations.

**Additional knowledge on the model**

Dense polynomial: degree bound
Making evaluation-interpolation schemes fault tolerant

\[ x_i \in F \quad \rightarrow \quad f(x_i) \quad \rightarrow \quad f = \sum_{i=1}^{t} c_i X^{d_i} \]

**Problem**

*Recover an unknown function \( f \), given as a black-box, from its evaluations.*

**Additional knowledge on the model**

- **Dense polynomial**: degree bound
- **Sparse polynomial**: support unknown, bound on sparsity
Making evaluation-interpolation schemes fault tolerant

\[ x_i \in F \quad \frac{f}{g}(x_i) \quad f = \sum_{i=0}^{d_f} f_i X^i, \quad g = \sum_{i=0}^{d_g} g_i X^i \]

Problem

*Recover an unknown function* \( f \), given as a black-box, *from its evaluations.*

Additional knowledge on the model

**Dense polynomial:** degree bound

**Sparse polynomial:** support unknown, bound on sparsity

**Dense rational function:** degree bounds
Making evaluation-interpolation schemes fault tolerant

\[ x_i \in F \rightarrow f ? \rightarrow f(x_i) + e_i \]

**Problem**

Recover an unknown function \( f \), given as a black-box, from its evaluations.

**Additional knowledge on the model**

- **Dense polynomial**: degree bound
- **Sparse polynomial**: support unknown, bound on sparsity
- **Dense rational function**: degree bounds

**Trust in the evaluations**

- errors (outliers)
- approximations: numerical noise
Making evaluation-interpolation schemes fault tolerant

\[ x_i \in F \quad \rightarrow \quad f \quad \rightarrow \quad f(x_i) + e_i \]

Problem

Recover an unknown function \( f \), given as a black-box, from its evaluations.

Additional knowledge on the model

Dense polynomial: degree bound

Sparse polynomial: support unknown, bound on sparsity

Dense rational function: degree bounds

Trust in the evaluations

- errors (outliers)
- approximations: numerical noise
Rational function reconstruction

**Problem (RFR: Rational Function Reconstruction)**

Given \( A, B \in K[X] \) with \( \deg B < \deg A = n \),

Find \( f \in K_{\leq d_f}[X], \; g \in K_{\leq n-d_f-1}[X] \) such that

\[
f = gB \mod A.
\]

**Fact**

The Extended Euclidean Algo. run on \( (A, B) \) and terminated when \( \deg f_j \leq d_f < \deg f_{j-1} \), produces \( f_j = u_j A + v_j B \) s.t.

1. \( (f_j, v_j) \) is a solution to the RFR problem.
2. it is minimal: any other solution \( (f, g) \) is of the form

\[
f = qf_j, \quad g = qv_j \text{ for } q \in K[X].
\]
Dense polynomial interpolation

\[ x_i \in F \quad \rightarrow \quad f(x_i) \quad \rightarrow \quad f = \sum_{i=0}^{d_f} c_i X^i \]

without error: polynomial interpolation (Lagrange, Newton, etc).
Dense polynomial interpolation with errors

\[ x_i \in F \quad \text{?} \quad f(x_i) + e_i = y_i \quad f = \sum_{i=0}^{df} c_i X^i \]

without error: polynomial interpolation (Lagrange, Newton, etc).
with errors: Reed-Solomon decoding

- Erroneous interpolant: \( h = \text{Interp}((y_i, x_i)) \)
- Error locator polynomial: \( \Lambda = \prod_{e_i \neq 0} (X - x_i) \)

\[ \Lambda f = \Lambda h \mod \prod_{i=0}^{n-1} (X - x_i) \]
Dense polynomial interpolation with errors

\[ x_i \in F \implies f \] \[ f(x_i) + e_i = y_i \quad f = \sum_{i=0}^{d_f} c_i X^i \]

without error: polynomial interpolation (Lagrange, Newton, etc).
with errors: Reed-Solomon decoding

- Erroneous interpolant: \( h = \text{Interp}((y_i, x_i)) \)
- Error locator polynomial: \( \Lambda = \prod_{e_i \neq 0} (X - x_i) \)

\[ \Lambda f \equiv \frac{\Lambda h \mod \prod_{i=0}^{n-1} (X - x_i)}{N/D} \]

Rational Reconstruction Problem:

\( (\Lambda f, \Lambda) \) is a minimal solution \( \rightsquigarrow \) computed by Ext. Euclidean Algorithm

\[ f = f_j / v_j. \]
Correction capacity

Unique decoding of $t$ errors whenever:

$n \geq \deg f + 2E + 1$

Bounding the degree

- $\deg f$ rarely known \textit{a priori}; bound $d_f \geq \deg f$ often pessimistic
- Early termination:
  - \textbf{without errors}: add evaluations until interpolant stabilizes
  - \textbf{with errors}: no stabilization
Correction capacity

Unique decoding of $t$ errors whenever:

$$n \geq \deg f + 2E + 1$$

Bounding the degree

- $\deg f$ rarely known \textit{a priori}; bound $d_f \geq \deg f$ often pessimistic
- Early termination:
  - without errors: add evaluations until interpolant stabilizes
  - with errors: no stabilization

Parameter oblivious decoding

Khonji, P., Roch, Roche and Stalinsky 10

\implies how to use all available redundancy?

Effective redundancy available

Upper bound on $\deg f$ redundancy used with RS codes
Correction capacity

Unique decoding of $t$ errors whenever:

\[ n \geq \deg f + 2E + 1 \]

Bounding the degree

- $\deg f$ rarely known \textit{a priori}; bound $d_f \geq \deg f$ often pessimistic
- Early termination:
  - without errors: add evaluations until interpolant stabilizes
  - with errors: no stabilization

Parameter oblivious decoding

Khonji, P., Roch, Roche and Stalinsky 10

\[ \Rightarrow \text{how to use all available redundancy?} \]

\[ \Rightarrow \text{list decoder exploring all length } n \text{ Reed-Solomon codes} \]
Dense rational function interpolation with errors

\[ x_i \in F \quad \xrightarrow{\frac{f}{g}} \quad \frac{f(x_i)}{g} + e_i \quad \xrightarrow{f} \quad f = \sum_{i=0}^{d_f} f_i X^i, \ g = \sum_{i=0}^{d_g} g_i X^i \]

\[ \Lambda f = \Lambda g \ \mod \ \prod_{y_i \neq \infty} (X - x_i) \]

Rational Reconstruction Problem

\((\Lambda f, \Lambda g)\) is a minimal solution \(\leadsto\) computed by Ext. Euclidean Algorithm.
Dense rational function interpolation with errors

\[ x_i \in F \quad \xrightarrow{f/g} \quad \frac{f}{g}(x_i) + e_i \quad \xrightarrow{f = \sum_{i=0}^{d_f} f_i X^i, \ g = \sum_{i=0}^{d_g} g_i X^i} \]

\[ \Lambda f = \bar{\Lambda} g \mod \prod_{y_i \neq \infty} (X - x_i) \]

Rational Reconstruction Problem

\((\Lambda f, \bar{\Lambda} g)\) is a minimal solution \(\rightsquigarrow\) computed by Ext. Euclidean Algorithm.

Correction capacity

Unique decoding of \(E\) errors whenever

\[ n \geq d_f + d_g + 2E + 1 \]

- smoothly supports evaluations at poles (even erroneous ones)
- parameter oblivious decoding applies
Sparse interpolation

\[ x_i \in F \quad \xrightarrow{f} \quad f(x_i) \quad \xrightarrow{f} \quad f = \sum_{i=1}^{t} c_i X^{d_i} \]

Without error: [Prony 1795] [Ben-Or and Tiwari 88]

- sample in a geometric progression: \( y_i = f(\alpha^i) \)
Sparse interpolation

\[ x_i \in F \quad \xrightarrow{f} \quad f(x_i) \quad \xrightarrow{f} \quad f = \sum_{i=1}^{t} c_i X^{d_i} \]

**Without error:** [Prony 1795] [Ben-Or and Tiwari 88]
- sample in a geometric progression: \( y_i = f(\alpha^i) \)
- [Blahut’84]: the seq. \((y_0, y_1, \ldots)\) has linear complexity \( t \)
- and is generated by
  \[
  \Lambda(X) = \prod_{i=1}^{t} (X - \alpha^{d_i})
  \]
- Berlekamp-Massey algo. on \( 2t \) terms
- Vandermonde system
Sparse interpolation with errors

\[ x_i \in F \rightarrow f ? \rightarrow f(x_i) + e_i f = \sum_{i=1}^{t} c_i X^{d_i} \]

**Without error:** [Prony 1795] [Ben-Or and Tiwari 88]
- sample in a geometric progression: \( y_i = f(\alpha^i) \)
- [Blahut’84]: the seq. \((y_0, y_1, \ldots)\) has linear complexity \(t\)
- and is generated by \( \Lambda(X) = \prod_{i=1}^{t} (X - \alpha^{d_i}) \)
- Berlekamp-Massey algo. on \(2t\) terms
- Vandermonde system

**With errors:** rule of thumb:
- find a clean sub-sequence of \(2t\) terms free of error
Unique decoding by majority rule Berlekamp-Massey

Comer, Kaltofen and P. 12

Necessary condition for unique decoding:

\[ n \geq 2t(2E + 1) \]

\[
\begin{array}{c|c|c|c}
(t-1)\text{zeros} & (a_i) & \Lambda(z) & f(z) \\
0, 1, \bar{0}, 1, \ldots, \bar{0}, 1 & \Lambda(z) & z^t - 1 \quad \frac{1}{t} \sum_{i=0}^{t-1} z^{2i \frac{m}{2t}}
\end{array}
\]
### Unique decoding by majority rule Berlekamp-Massey

**Comer, Kaltofen and P. 12**

#### Necessary condition for unique decoding:

\[ n \geq 2t(2E + 1) \]

### Example

<table>
<thead>
<tr>
<th>((a_i))</th>
<th>(\Lambda(z))</th>
<th>(f(z))</th>
</tr>
</thead>
<tbody>
<tr>
<td>((t-1)) zeros</td>
<td>((t-1)) zeros</td>
<td>((t-1)) zeros</td>
</tr>
<tr>
<td>((0, 1, \bar{0}, 1, \ldots, \bar{0}, 1))</td>
<td>((0, 1, \bar{0}, -1, \ldots, \bar{0}, -1))</td>
<td>((0, 1))</td>
</tr>
<tr>
<td>(z^t - 1)</td>
<td>(z^t + 1)</td>
<td>(\frac{1}{t} \sum_{i=0}^{t-1} z^{2i\frac{m}{2t}})</td>
</tr>
</tbody>
</table>
Unique decoding by majority rule Berlekamp-Massey

Comer, Kaltofen and P. 12

Necessary condition for unique decoding:

\[ n \geq 2t(2E + 1) \]

\[
\begin{align*}
    x = (0, 1, 0, 1, \ldots, 0, 1) \\
y = (0, 1, 0, -1, \ldots, 0, -1)
\end{align*}
\]

\[
\begin{array}{c|c|c}
    (a_i) & \Lambda(z) & f(z) \\
    \hline
    (t-1)\text{zeros} & \Lambda(z) & f(z) \\
    \hline
    1, \ldots, 1 \quad 0 \quad 0 \quad \ldots \quad 0 \\
    1, \ldots, 1 \quad 0 \quad 0 \quad \ldots \quad 0 \\
    \hline
    z^t - 1 & \frac{1}{t} \sum_{i=0}^{t-1} z^{2i \frac{m}{2t}} \\
    z^t + 1 & -\frac{1}{t} \sum_{i=0}^{t-1} z^{(2i+1) \frac{m}{2t}}
\end{array}
\]

Sufficient condition for unique decoding:

\[ n \leq 2t(2E + 1) \]

\[
\begin{array}{c}
\Lambda_1 \quad \Lambda_2 \quad \Lambda_3 \quad \Lambda_4 \quad \Lambda_5 \quad \Lambda_1 \quad \Lambda_2 \quad \Lambda_3 \quad \Lambda_4 \quad \Lambda_5
\end{array}
\]

\[
\begin{align*}
    2t & \quad E=2 & \quad n=2t(2E+1)
\end{align*}
\]
Unique decoding by majority rule Berlekamp-Massey

Comer, Kaltofen and P. 12

Necessary condition for unique decoding:

\[ n \geq 2t(2E + 1) \]

\[
\begin{array}{c|c|c|c}
(a_i) & \Lambda(z) & f(z) \\
\hline
t-1 \text{zeros} & \Lambda(z) & f(z) \\
x = (0, 1, 0, 1, \ldots, 0, 1) & z^t - 1 & \frac{1}{t} \sum_{i=0}^{t-1} z^{2i \frac{m}{2t}} \\
y = (0, 1, 0, -1, \ldots, 0, -1) & z^t + 1 & \frac{-1}{t} \sum_{i=0}^{t-1} z^{(2i+1) \frac{m}{2t}} \\
\hline
\end{array}
\]

Sufficient condition for list decoding:

\[ n \leq 2t(E + 1) \]
### List decoding: using affine sub-sequences

**Kaltofen and P. 14**

<table>
<thead>
<tr>
<th></th>
<th>$f(\alpha^0)$</th>
<th>$f(\alpha^1)$</th>
<th>$f(\alpha^2)$</th>
<th>$f(\alpha^3)$</th>
<th>$f(\alpha^4)$</th>
<th>$f(\alpha^5)$</th>
<th>$f(\alpha^6)$</th>
<th>$f(\alpha^7)$</th>
<th>$f(\alpha^8)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(\alpha^0)$</td>
<td>$f(\alpha^2)$</td>
<td>$f(\alpha^4)$</td>
<td>$f(\alpha^6)$</td>
<td>$f(\alpha^8)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^1)$</td>
<td>$f(\alpha^3)$</td>
<td>$f(\alpha^5)$</td>
<td>$f(\alpha^7)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^0)$</td>
<td></td>
<td>$f(\alpha^3)$</td>
<td>$f(\alpha^6)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^1)$</td>
<td></td>
<td></td>
<td>$f(\alpha^4)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^2)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
List decoding: using affine sub-sequences

Kaltofen and P. 14

<table>
<thead>
<tr>
<th>$f(\alpha^0)$</th>
<th>$f(\alpha^1)$</th>
<th>$f(\alpha^2)$</th>
<th>$f(\alpha^3)$</th>
<th>$f(\alpha^4)$</th>
<th>$f(\alpha^5)$</th>
<th>$f(\alpha^6)$</th>
<th>$f(\alpha^7)$</th>
<th>$f(\alpha^8)$</th>
<th>$= \text{Ev}(f, \alpha)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(\alpha^0)$</td>
<td>$f(\alpha^2)$</td>
<td>$f(\alpha^4)$</td>
<td>$f(\alpha^6)$</td>
<td>$f(\alpha^8)$</td>
<td>$= \text{Ev}(f, \alpha^2)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^1)$</td>
<td>$f(\alpha^3)$</td>
<td>$f(\alpha^5)$</td>
<td>$f(\alpha^7)$</td>
<td>$= \text{Ev}(f \circ (\alpha x), \alpha^2)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^0)$</td>
<td>$f(\alpha^3)$</td>
<td>$f(\alpha^6)$</td>
<td>$= \text{Ev}(f, \alpha^3)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^1)$</td>
<td>$f(\alpha^4)$</td>
<td>$f(\alpha^7)$</td>
<td>$= \text{Ev}(f \circ (\alpha x), \alpha^3)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^2)$</td>
<td>$f(\alpha^5)$</td>
<td>$f(\alpha^8)$</td>
<td>$= \text{Ev}(f \circ (\alpha^2 x), \alpha^3)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
List decoding: using affine sub-sequences

Kaltofen and P. 14

<table>
<thead>
<tr>
<th>$f(\alpha^0)$</th>
<th>$f(\alpha^1)$</th>
<th>$f(\alpha^2)$</th>
<th>$f(\alpha^3)$</th>
<th>$f(\alpha^4)$</th>
<th>$f(\alpha^5)$</th>
<th>$f(\alpha^6)$</th>
<th>$f(\alpha^7)$</th>
<th>$f(\alpha^8)$</th>
<th>$= \text{Ev}(f, \alpha)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(\alpha^0)$</td>
<td>$f(\alpha^2)$</td>
<td>$f(\alpha^4)$</td>
<td>$f(\alpha^6)$</td>
<td></td>
<td>$f(\alpha^8)$</td>
<td></td>
<td></td>
<td>$= \text{Ev}(f, \alpha^2)$</td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^1)$</td>
<td>$f(\alpha^3)$</td>
<td>$f(\alpha^5)$</td>
<td>$f(\alpha^7)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$= \text{Ev}(f \circ (\alpha x), \alpha^2)$</td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^0)$</td>
<td>$f(\alpha^3)$</td>
<td>$f(\alpha^6)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$= \text{Ev}(f, \alpha^3)$</td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^1)$</td>
<td>$f(\alpha^4)$</td>
<td>$f(\alpha^7)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$= \text{Ev}(f \circ (\alpha x), \alpha^3)$</td>
<td></td>
</tr>
<tr>
<td>$f(\alpha^2)$</td>
<td></td>
<td>$f(\alpha^5)$</td>
<td></td>
<td>$f(\alpha^8)$</td>
<td></td>
<td></td>
<td></td>
<td>$= \text{Ev}(f \circ (\alpha^2 x), \alpha^3)$</td>
<td></td>
</tr>
</tbody>
</table>
List decoding: using affine sub-sequences

Kaltofen and P. 14

\[
\begin{array}{cccccccc}
  f(\alpha^0) & f(\alpha^1) & f(\alpha^2) & f(\alpha^3) & f(\alpha^4) & f(\alpha^5) & f(\alpha^6) & f(\alpha^7) & f(\alpha^8) \\
  f(\alpha^0) & f(\alpha^2) & f(\alpha^4) & f(\alpha^6) & f(\alpha^8) & = & \text{Ev}(f, \alpha) \\
  f(\alpha^1) & f(\alpha^3) & f(\alpha^5) & f(\alpha^7) & = & \text{Ev}(f \circ (\alpha x), \alpha^2) \\
  f(\alpha^0) & f(\alpha^3) & f(\alpha^6) & = & \text{Ev}(f, \alpha^3) \\
  f(\alpha^1) & f(\alpha^4) & f(\alpha^7) & = & \text{Ev}(f \circ (\alpha x), \alpha^3) \\
  f(\alpha^2) & f(\alpha^5) & f(\alpha^8) & = & \text{Ev}(f \circ (\alpha^2 x), \alpha^3)
\end{array}
\]
List decoding: using affine sub-sequences

Kaltofen and P. 14

\[ f(\alpha^0) \ f(\alpha^1) \ f(\alpha^2) \quad f(\alpha^3) \ f(\alpha^4) \ f(\alpha^5) \ f(\alpha^6) \ f(\alpha^7) \ f(\alpha^8) = \text{Ev}(f, \alpha) \]

\[ f(\alpha^0) \quad f(\alpha^1) \quad f(\alpha^2) \quad f(\alpha^3) \quad f(\alpha^4) \quad f(\alpha^5) \quad f(\alpha^6) \quad f(\alpha^7) = \text{Ev}(f \circ (\alpha x), \alpha^2) \]

\[ f(\alpha^0) \quad f(\alpha^1) \quad f(\alpha^2) \quad f(\alpha^3) \quad f(\alpha^4) \quad f(\alpha^5) \quad f(\alpha^6) \quad f(\alpha^7) = \text{Ev}(f \circ (\alpha x), \alpha^3) \]

Difficult worst case analysis

\[ \text{size of the largest subset of } \{1, \ldots, n\} \text{ not containing } k \text{ terms in arithmetic progression} \]

\[ n - n^{(\log \log n)^{1/2}} \leq E \leq n^k - 2 \log k n^k - 2. \]
List decoding: using affine sub-sequences

Kaltofen and P. 14

\[
\begin{align*}
  f(\alpha^0) & \quad f(\alpha^1) & \quad f(\alpha^2) & \quad f(\alpha^3) & \quad f(\alpha^4) & \quad f(\alpha^5) & \quad f(\alpha^6) & \quad f(\alpha^7) & \quad f(\alpha^8) = \text{Ev}(f, \alpha) \\
  f(\alpha^0) & \quad f(\alpha^2) & \quad f(\alpha^4) & \quad f(\alpha^6) & \quad f(\alpha^8) = \text{Ev}(f, \alpha^2) \\
  f(\alpha^1) & \quad f(\alpha^3) & \quad f(\alpha^5) & \quad f(\alpha^7) = \text{Ev}(f \circ (\alpha x), \alpha^2) \\
  f(\alpha^0) & \quad f(\alpha^3) & \quad f(\alpha^6) = \text{Ev}(f, \alpha^3) \\
  f(\alpha^1) & \quad f(\alpha^4) & \quad f(\alpha^7) = \text{Ev}(f \circ (\alpha x), \alpha^3) \\
  f(\alpha^2) & \quad f(\alpha^4) & \quad f(\alpha^5) & \quad f(\alpha^7) = \text{Ev}(f \circ (\alpha^2 x), \alpha^3)
\end{align*}
\]

Difficult worst case analysis

- [Erdős and Turan 36]: size of the largest subset of \{1 \ldots n\} not containing \(k\) terms in arithmetic progression
- [Szeremedi 75]: arithmetic prog. are dense

\[
\begin{align*}
  n - \frac{n}{(\log \log n)^{1/2k + 9}} \leq E \leq \frac{n}{k-2} \log_k \frac{n}{k-2}.
\end{align*}
\]
Towards better decoding capacities

Unique decoding:

List decoding (basic):

List decoding (affine subsequence):
Towards better decoding capacities

Unique decoding:

List decoding (basic):

List decoding (affine subsequence):

Improved conditions for unique decoding

Descartes’ rule of signs: Over $K = \mathbb{R}_{>0}$:

Irreducibility of cyclotomic polynomials:

- Over $K = \mathbb{C}$:

- Over $\mathbb{F}_{q}^{(p_1)} \times \cdots \times \mathbb{F}_{q}^{(p_n)}$:

- No known decoding algorithm

- Makes the list decoding algo. a unique decoding one
Conclusion

Design framework for high performance exact linear algebra

Asymptotic reduction $\succ$ algorithm tuning $\succ$ building block implementation

- Favor **tiled recursive** algorithms
  $\leadsto$ **architecture oblivious vs aware** algorithms [Gustavson 07]
Conclusion

Design framework for high performance exact linear algebra

Asymptotic reduction > algorithm tuning > building block implementation

- Favor tiled recursive algorithms
  ⇔ architecture oblivious vs aware algorithms [Gustavson 07]
- New pivoting strategies revealing all rank profile informations
  ⇔ tournament pivoting? [Demmel, Grigori and Xiang 11]
  ⇔ $O(r^\omega + (m + n + |A|)^{1+o(n)})$? [Storjohann and Yang 14]
Conclusion

Design framework for high performance exact linear algebra

Asymptotic reduction $>$ algorithm tuning $>$ building block implementation

- Favor tiled recursive algorithms
  $\Rightarrow$ architecture oblivious vs aware algorithms [Gustavson 07]
- New pivoting strategies revealing all rank profile informations
  $\Rightarrow$ tournament pivoting? [Demmel, Grigori and Xiang 11]
  $\Rightarrow O(r^\omega + (m + n + |A|)^{1+o(n)})$? [Storjohann and Yang 14]
- Recursive tasks and coarse grain parallelization
  $\Rightarrow$ Light weight task workstealing management required (libkomp)
  $\Rightarrow$ Need for an improved recursive dataflow scheduling
Conclusion

Fault tolerance based on evaluation codes

- RS and CRT codes extended to rational fractions
  \(\rightsquigarrow\) smooth generalization

- Parameter oblivious decoding for early termination schemes
  \(\rightsquigarrow\) parameter oblivious \textbf{list-decoding}? [Wu 08]

- Sparse evaluation codes
  \(\rightsquigarrow\) Gap between best correction radius and existing algorithm
Perspectives

Large scale distributed exact linear algebra

- reducing communications [Demmel, Grigori and Xiang 11]
- sparse and hybrid (Boyer and Vialla) [Faugère and Lachartre 10]
- combine genericity and efficiency to attack crypto. challenges
Perspectives

Large scale distributed exact linear algebra

- reducing communications [Demmel, Grigori and Xiang 11]
- sparse and hybrid (Boyer and Vialla) [Faugère and Lachartre 10]
- combine genericity and efficiency to attack crypto. challenges

Polynomial matrix arithmetic for coding theory

- State of the art implementations in LinBox [Giorgi and Lebreton 14]
- Coding theory tools in Sage (Lucas)
- Further joint developments
Perspectives

Large scale distributed exact linear algebra

- reducing communications [Demmel, Grigori and Xiang 11]
- sparse and hybrid (Boyer and Vialla) [Faugère and Lachartre 10]
- combine genericity and efficiency to attack crypto. challenges

Polynomial matrix arithmetic for coding theory

- State of the art implementations in LinBox [Giorgi and Lebreton 14]
- Coding theory tools in Sage (Lucas)
- Further joint developments

Symbolic-numeric computation

- smooth transition between noise and errors for sparse codes [Comer Kaltofen P. 12], [Kaltofen Yang 13-14]
  ↦ improve decoding capacities and efficiency
  ↦ extend to larger classes of codes
- High precision floating point linear algebra via exact rational arithmetic
Perspectives

Large scale distributed exact linear algebra

- reducing communications [Demmel, Grigori and Xiang 11]
- sparse and hybrid (Boyer and Vialla) [Faugère and Lachartre 10]
- combine genericity and efficiency to attack crypto. challenges

Polynomial matrix arithmetic for coding theory

- State of the art implementations in LinBox [Giorgi and Lebreton 14]
- Coding theory tools in Sage (Lucas)
- Further joint developments

Symbolic-numeric computation

- smooth transition between noise and errors for sparse codes [Comer Kaltofen P. 12], [Kaltofen Yang 13-14]
  \[ \leadsto \] improve decoding capacities and efficiency
  \[ \leadsto \] extend to larger classes of codes
- High precision floating point linear algebra via exact rational arithmetic

Thank you