The M4RIE library for dense linear algebra over small fields with even characteristic

Martin R. Albrecht (martinralbrecht@googlemail.com)

POLSYS Team, UPMC, Paris, France

ISSAC 2012, Grenoble, France

Outline

- Multiplication
  - Precomputation Tables
  - Karatsuba Multiplication
  - Results
- Elimination

The M4RIE Library

- handles $\mathbb{F}_{2^e}$ for $2 \leq e \leq 10$; $e \leq 16$ planned.
- available under the GPL Version 2 or later (GPLv2+)
- provides basic arithmetic (addition, equality testing, stacking, augmenting, sub-matrices, randomisation, etc.)
- implements asymptotically fast multiplication (this talk)
- implements asymptotically fast elimination (this talk)
- Linux, Mac OS X (x86 and PPC), OpenSolaris, and Windows (Cygwin)

http://m4ri.sagemath.org

Representation of Elements I

Elements in $\mathbb{F}_{2^e} \cong \mathbb{F}_2[x]/f$ can be written as

$$a_0\alpha^0 + a_1\alpha^1 + \cdots + a_{e-1}\alpha^{e-1}.$$

We identify the bitstring $a_0, \ldots, a_{e-1}$ with

- the element $\sum_{i=0}^{e-1} a_i\alpha^i \in \mathbb{F}_{2^e}$ and
- the integer $\sum_{i=0}^{e-1} a_i 2^i$.

Representation of Elements II

In the datatype `mzed_t` we pack several of those bitstrings into one machine word:

$$a_{0,0,0},\dots,a_{0,0,e-1},\ a_{0,1,0},\dots,a_{0,1,e-1},\ \dots,\ a_{0,n-1,0},\dots,a_{0,n-1,e-1}.$$

Additions are cheap, scalar multiplications are expensive.

Representation of Elements III

- Instead of representing matrices over $\mathbb{F}_{2^e}$ as matrices over polynomials we may represent them as polynomials with matrix coefficients.
- For each degree we store matrices over $\mathbb{F}_2$ which hold the coefficients for this degree.
- The data type `mzd_slice_t` for matrices over $\mathbb{F}_{2^e}$ internally stores e-tuples of M4RI matrices, i.e., matrices over $\mathbb{F}_2$.

Additions are cheap, scalar multiplications are expensive.

Representation of Elements IV

$$
\begin{aligned}
A &= \begin{pmatrix} \alpha^2+1 & \alpha \\ \alpha+1 & 1 \end{pmatrix} \\
&= \begin{bmatrix} \square 101 & \square 010 \\ \square 011 & \square 001 \end{bmatrix} \\
&= \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \right)
\end{aligned}
$$

Figure: 2×2 matrix over $\mathbb{F}_8$

Outline

Multiplication
- Precomputation Tables
- Karatsuba Multiplication
- Results

Elimination

Outline

Multiplication
- Precomputation Tables
- Karatsuba Multiplication
- Results

Elimination

The idea I

Input: A – $m \times n$ matrix
Input: B – $n \times k$ matrix
begin
    for $0 \leq i < m$ **do**
        for $0 \leq j < n$ **do**
            $C_j \longleftarrow C_j + A_{j,i} \times B_i$;
    return C;

The idea II

Input: A – $m \times n$ matrix
Input: B – $n \times k$ matrix

begin
  for $0 \leq i < m$ **do**
    for $0 \leq j < n$ **do**
      $C_j \longleftarrow C_j + A_{j,i} \times B_i$; // **cheap**
  return C;

The idea III

Input: A – $m \times n$ matrix
Input: B – $n \times k$ matrix

begin
    for $0 \leq i < m$ **do**
        for $0 \leq j < n$ **do**
            $C_j \longleftarrow C_j + A_{j,i} \times B_i$; // **expensive**
    return C;

The idea IV

Input: A – $m \times n$ matrix
Input: B – $n \times k$ matrix

begin
  for $0 \leq i < m$ **do**
    for $0 \leq j < n$ **do**
      $C_j \longleftarrow C_j + A_{j,i} \times B_i$; // **expensive**
  return C;

But there are only 2^e possible multiples of B_i.

The idea V

begin

  Input: A – $m \times n$ matrix

  Input: B – $n \times k$ matrix

  for $0 \leq i < m$ **do**

    for $0 \leq j < 2^e$ **do**

      $T_j \longleftarrow j \times B_i$; // **now this is expensive**

    for $0 \leq j < n$ **do**

      $x \longleftarrow A_{j,i}$;

      $C_j \longleftarrow C_j + T_x$;

  return C;

$m \cdot n \cdot k$ additions, $m \cdot 2^e \cdot k$ multiplications.

Optimisation: Computing Precomputation Tables

- Computing precomputation tables naively costs 2^e multiplications, one for each entry.
- We can reduce this to e multiplication and 2^e additions, by
  - computing the e products $\alpha^j \cdot B_i$ for $0 \leq j < e$
  - and forming all linear combinations thereof
- For the second step we can use Gray codes [Gra53] similar to the "Method of the Four Russians".

$m \cdot (n + 2^e) \cdot k$ additions, $m \cdot e \cdot k$ multiplications.

Optimisation: Multiple Precomputation Tables

Now, that we have eliminated most scalar multiplications,

- the actual arithmetic is quite cheap compared to memory reads and writes and
- the cost of memory accesses greatly depends on where in memory data is located.
  1. If our tables T are in cache that is cheap,
  2. we have to read/write the row C_j anyway,
  3. accessing $A_{j,i}$ does not seem to cost must in practice.
- So we try to fill our cache with precomputation code tables.

In our implementation we use 8 such tables.

Strassen-Winograd [Str69] Multiplication

- fastest known pratical algorithm
- complexity: $\mathcal{O}\left(n^{\log_2 7}\right)$
- The algorithm just described can be used as base case for small dimensions

Outline

- Multiplication
  - Precomputation Tables
  - Karatsuba Multiplication
  - Results
- Elimination

Karatsuba: the idea

- Consider $\mathbb{F}_{2^2}$ with the primitive polynomial $f = x^2 + x + 1$.
- We want to compute $C = A \cdot B$.
- Rewrite A as $A_1 x + A_0$ and B as $B_1 x + B_0$.
- The product is

$$C = A_1 B_1 x^2 + (A_1 B_0 + A_0 B_1)x + A_0 B_0.$$

- Reduction modulo f gives

$$C = (A_1 B_1 + A_1 B_0 + A_0 B_1)x + A_0 B_0 + A_1 B_1.$$

- This last expression can be rewritten as

$$C = ((A_1 + A_0)(B_1 + B_0) + A_0 B_0)x + A_0 B_0 + A_1 B_1.$$

Thus this multiplication costs 3 multiplications and 4 adds over $\mathbb{F}_2$.

Implementation

- We use the M4RI library to provide multiplications and additions over $\mathbb{F}_2$.
- LinBox now implements a generalisation of this for dense matrices over $\mathbb{F}_{p^e}$ for larger p.

Outline

- Multiplication
  - Precomputation Tables
  - Karatsuba Multiplication
  - Results
- Elimination

Results: Multiplication I

e	Magma 2.15-10	GAP 4.4.12	SW-NJ	SW-NJ/ M4RI	[Mon05]	Bitslice	Bitslice/ M4RI
1	0.100s	0.244s	–	1	1	0.071s	1.0
2	1.220s	12.501s	0.630s	8.8	3	0.224s	3.1
3	2.020s	35.986s	1.480s	20.8	6	0.448s	6.3
4	5.630s	39.330s	1.644s	23.1	9	0.693s	9.7
5	94.740s	86.517s	3.766s	53.0	13	1.005s	14.2
6	89.800s	85.525s	4.339s	61.1	17	1.336s	18.8
7	82.770s	83.597s	6.627s	93.3	22	1.639s	23.1
8	104.680s	83.802s	10.170s	143.2	27	2.140s	30.1

Table: Multiplication of $4,000 \times 4,000$ matrices over $\mathbb{F}_{2^e}$ on 2.66 Ghz Intel i7

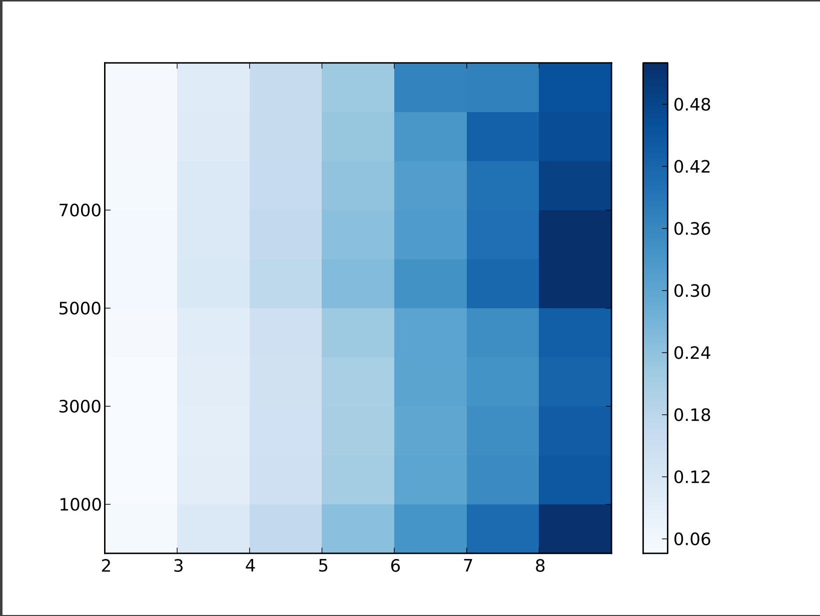
Results: Multiplication II

	CPU time in seconds						
n	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$	$e = 8$
1000	0.005	0.011	0.016	0.024	0.033	0.041	0.051
2000	0.032	0.067	0.100	0.149	0.209	0.245	0.309
3000	0.102	0.206	0.312	0.453	0.651	0.753	0.951
4000	0.230	0.468	0.687	1.013	1.475	1.653	2.061
5000	0.489	0.919	1.322	2.042	2.784	3.168	3.931
6000	0.872	1.798	2.681	3.895	5.189	6.375	7.885
7000	1.311	2.682	3.962	5.758	7.549	9.450	12.153
8000	1.873	3.869	5.611	8.145	10.862	13.617	16.615

	CPU cycles / $n^{2.807}$						
n	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$	$e = 8$
1000	0.055	0.113	0.169	0.247	0.336	0.414	0.518
2000	0.046	0.096	0.145	0.215	0.302	0.354	0.446
3000	0.047	0.095	0.144	0.209	0.301	0.348	0.439
4000	0.047	0.096	0.141	0.208	0.304	0.340	0.424
5000	0.053	0.101	0.145	0.224	0.306	0.348	0.433
6000	0.057	0.118	0.177	0.257	0.342	0.420	0.520
7000	0.056	0.114	0.169	0.246	0.323	0.404	0.520
8000	0.055	0.113	0.165	0.239	0.319	0.400	0.489

Table: Multiplication on 2.66 Ghz Intel i7

Results: Multiplication III

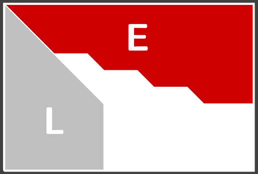


Outline

Multiplication
- Precomputation Tables
- Karatsuba Multiplication
- Results

Elimination

PLE Decomposition I



Definition (PLE)

Let A be a $m \times n$ matrix over a field K. A PLE decomposition of A is a triple of matrices P, L and E such that P is a $m \times m$ permutation matrix, L is a unit lower triangular matrix, and E is a $m \times n$ matrix in row-echelon form, and

$$A = PLE.$$

PLE decomposition can be in-place, that is L and E are stored in A and P is stored as an m-vector.

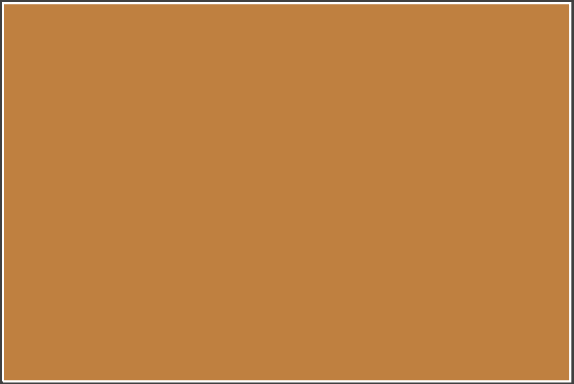
PLE Decomposition II

From the PLE decomposition we can

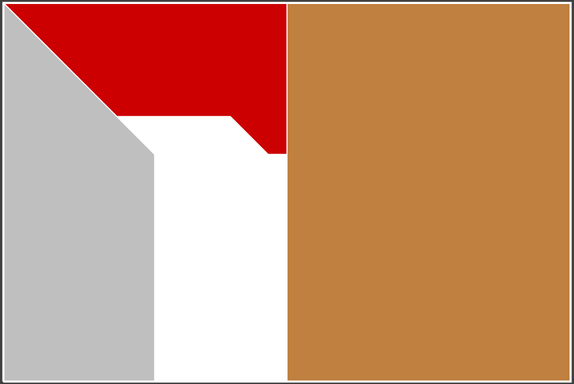
- read the rank r,
- read the row rank profile (pivots),
- compute the null space,
- solve $y = Ax$ for x and
- compute the (reduced) row echelon form.

C.-P. Jeannerod, C. Pernet, and A. Storjohann.
Rank-profile revealing Gaussian elimination and the CUP matrix decomposition.
arXiv:1112.5717, 35 pages, 2012.

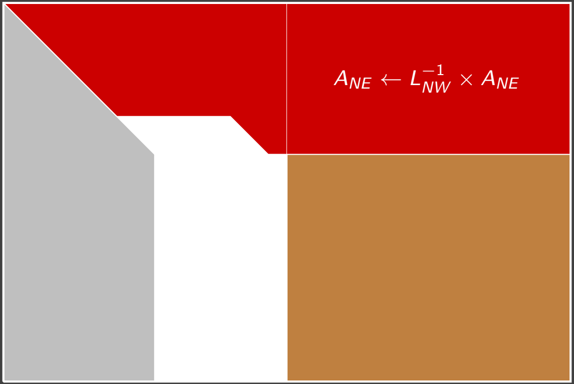
Block Recursive PLE Decomposition $\mathcal{O}(n^\omega)$ I



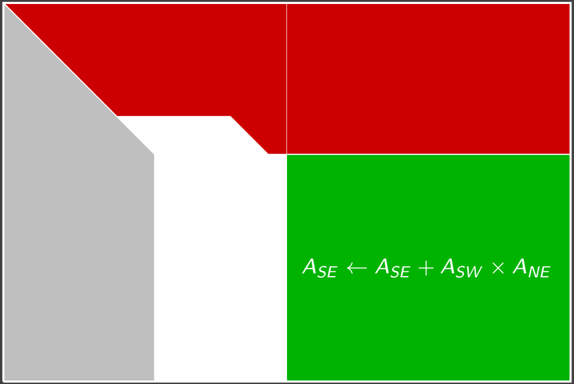
Block Recursive PLE Decomposition $\mathcal{O}(n^\omega)$ II



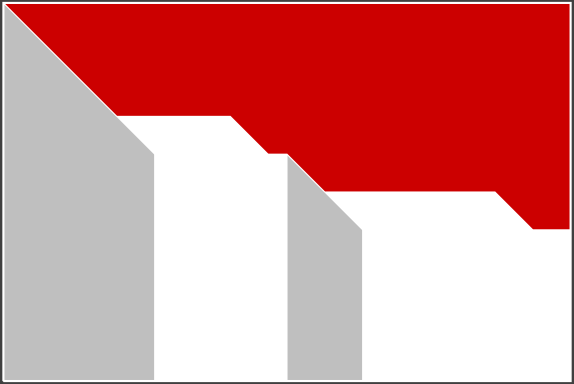
Block Recursive PLE Decomposition $\mathcal{O}(n^{\omega})$ III



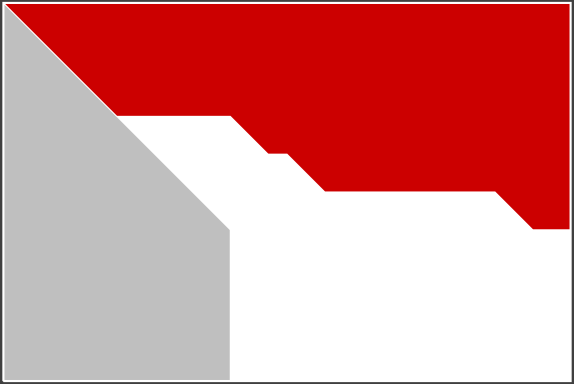
Block Recursive PLE Decomposition $\mathcal{O}(n^\omega)$ IV



Block Recursive PLE Decomposition $\mathcal{O}(n^\omega)$ V



Block Recursive PLE Decomposition $\mathcal{O}(n^\omega)$ VI



Ingredients

We need

1. efficient matrix-matrix products (**DONE**),
2. an efficient implementation of triangular system solving with matrices (**NEXT**),
3. an efficient PLE base case (**NEXT**).

Triangular System Solving

Triangular system solving with matrices also reduced to matrix-matrix multiplication, so all we need is an efficient base-case.

For that we use precomputation tables again.

begin
    for $m > i \geq 0$ **do**
        $B_i \leftarrow U_{i,i}^{-1} \cdot B_i$;
        $T \leftarrow \textsc{MakeTable}(B_i)$;
        for $0 \leq j < i$ **do**
            $x \leftarrow U_{j,i}$;
            $B_j \leftarrow B_j + T_x$;

Gaussian elimination/PLE base case

```
Input: A – m × n matrix
begin
    r ⟵ 0;
    for 0 ≤ j < n do
        for r ≤ i < m do
            if A_{i,j} = 0 then continue;
            rescale row i of A such that A_{i,j} = 1;
            swap the rows i and r in A;
            T ⟵ multiplication table for row r of A;
            for r + 1 ≤ k < m do
                x ⟵ A_{k,j};
                A_k ⟵ A_k + T_x;
            r ⟵ r + 1;
        return r;
```

Results: Reduced Row Echelon Forms I

e	Magma 2.15-10	GAP 4.4.12	LinBox (mod p) 1.1.6	M4RIE 6b24b839a46f	
2	6.04s	162.65s	49.52s	3.31s	PLE
3	14.47s	442.52s	49.92s	5.33s	PLE
4	60.37s	502.67s	50.91s	6.33s	PLE
5	659.03s	N/A	51.20s	10.51s	PLE
6	685.46s	N/A	51.61s	13.08s	PLE
7	671.88s	N/A	53.94s	17.29s	PLE
8	840.22s	N/A	64.24s	20.25s	PLE
9	1630.38s	N/A	76.18s	260.77s	Gauss elim.
10	1631.35s	N/A	76.45s	291.30s	Gauss elim.

Table: Elimination of $10,000 \times 10,000$ matrices on 2.66Ghz i7

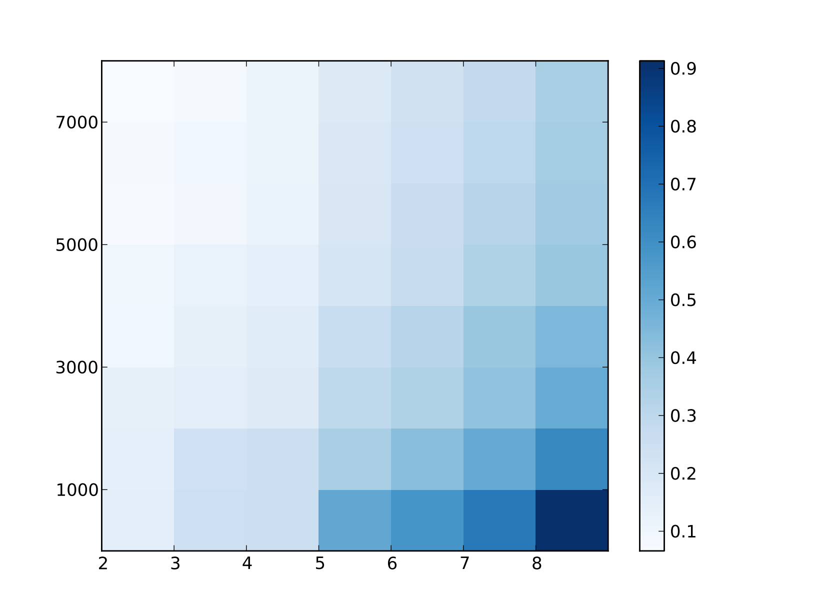
Results: Reduced Row Echelon Forms II

	CPU time in seconds						
n	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$	$e = 8$
1000	0.015	0.024	0.025	0.051	0.058	0.067	0.090
2000	0.100	0.167	0.177	0.244	0.298	0.349	0.435
3000	0.310	0.338	0.380	0.647	0.730	0.896	1.089
4000	0.483	0.691	0.792	1.304	1.558	1.928	2.194
5000	0.918	1.158	1.327	1.954	2.476	3.063	3.589
6000	1.123	1.448	1.836	2.995	4.012	4.869	5.691
7000	1.854	2.279	2.777	4.471	5.738	7.085	8.648
8000	2.258	2.738	4.032	6.102	8.049	9.835	12.091

	CPU cycles / $n^{2.807}$						
n	$e = 2$	$e = 3$	$e = 4$	$e = 5$	$e = 6$	$e = 7$	$e = 8$
1000	0.154	0.245	0.253	0.515	0.584	0.678	0.913
2000	0.144	0.240	0.254	0.351	0.429	0.503	0.625
3000	0.142	0.156	0.175	0.298	0.336	0.413	0.501
4000	0.098	0.142	0.162	0.267	0.320	0.396	0.450
5000	0.100	0.127	0.145	0.214	0.271	0.336	0.394
6000	0.072	0.095	0.120	0.197	0.264	0.320	0.374
7000	0.078	0.097	0.118	0.190	0.244	0.302	0.369
8000	0.066	0.080	0.118	0.179	0.236	0.288	0.354

Table: Gaussian elimination on 2.66 Ghz Intel i7

Results: Reduced Row Echelon Forms III



Sensitivity to Sparsity

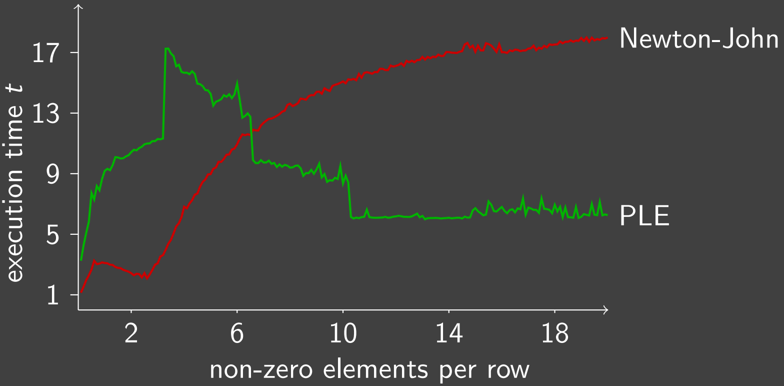


Figure: Gaussian elimination of $10,000 \times 10,000$ matrices over $\mathbb{F}_{16}$ on 2.67GHz Intel Core i7 M 620.

Fin


John
Travolta
Olivia
Newton-John
GREASE
STARTS JUNE 16th AT A THEATRE NEAR YOU
©1978 PARAMOUNT PICTURES CORPORATION

Frank Gray.
Pulse code communication, March 1953.
US Patent No. 2,632,058.

Peter L. Montgomery.
Five, six, and seven-term Karatsuba-like formulae.
IEEE Trans. on Computers, 53(3):362–369, 2005.

Volker Strassen.
Gaussian elimination is not optimal.
Nummerische Mathematik, 13:354–256, 1969.