Efficient linear algebra algorithms in symbolic computation

Erich Kaltofen
North Carolina State University
www.kaltofen.net

Factorization of an integer N
(continued fraction, quadratic sieves, number field sieves)

Compute a solution to the congruence equation

$$X^2 \equiv Y^2 \pmod{N}$$

via r relations on b basis primes

$$X_1^2 \cdot X_2^2 \cdots X_r^2 \equiv (p_1^{e_1})^2 \cdot (p_2^{e_2})^2 \cdots (p_b^{e_b})^2 \pmod{N}$$

Then N divides $(X+Y)(X-Y)$, hence

$$\mathrm{GCD}(X+Y, N) \text{ divides } N$$

Relation computation

Step 1: Compute $s > r$ relations on b basis primes

$$\forall 1 \leq i \leq s\colon Y_i^2 \equiv p_1^{c_{i,1}} \cdot p_2^{c_{i,2}} \cdots p_b^{c_{i,b}} \pmod{N}$$

Step 2: select r relations $X_1 = Y_{i_1}, \ldots, X_r = Y_{i_r}$ such that

$$\forall 1 \leq j \leq b\colon c_{i_1,j} + c_{i_2,j} + \cdots + c_{i_r,j} \equiv 0 \pmod{2}$$

One must compute non-zero solutions to the

sparse homogeneous linear system modulo 2

$$\begin{bmatrix} x_1 & \ldots & x_s \end{bmatrix} \begin{bmatrix} c_{1,1} \bmod 2 & \ldots & c_{1,b} \bmod 2 \\ c_{2,1} \bmod 2 & \ldots & c_{2,b} \bmod 2 \\ \vdots & & \vdots \\ c_{2,1} \bmod 2 & \ldots & c_{2,b} \bmod 2 \end{bmatrix} \equiv \begin{bmatrix} 0 & \ldots & 0 \end{bmatrix} \pmod{2}$$

LDDMLtR's RSA-120 matrix modulo 2

Row nr.	Columns with non-zero entries
1	0 1 481 1355 3b42 5cf6 c461 eda1 f0e7 15d19 199e0 2c317 33a5
2	0 1 9b4 f26 3214 7f99 a146 bc7e 10087 175c5 1953a 320b5 394
⋮	⋮
245 811	0 1 2 3 4 6 8 9 b c d f 10 12 13 14 16 17 18 19 1d 1e 1f 20 25 2
	. . . 3624a 36473 36905 37727 395

There are 10 − 217 non-zero entries/column, with 252 222 columns and 11 037 745 non-zero entries total; in the above format the matrix occupies 48 Mbytes of disc space.

```
challenge-rsa-honor-roll@rsa.com
RSA-155
Factors:
1026395928297411057720541965739916759007165678080380668033419335217907113
*
1066034883801684548209272203600128786792079585759892915222706082371930628
Date:    August 22, 1999
Method:  the General Number Field Sieve,
         with a polynomial selection method of Brian Murphy
         and Peter L. Montgomery,
         with lattice sieving (71%) and with line sieving (29%),
         and with  Peter L. Montgomery's blocked Lanczos and
         square root algorithms;
Time:    * Polynomial selection:
         The polynomial selection took approximately 100 MIPS years,
         equivalent to 0.40 CPU years on a 250 MHz processor.
         ...
         * Sieving: 35.7 CPU-years in total,
         ...
           124 722 179 relations were collected by eleven different sites
         ...
         * Filtering the data and building the matrix took about a month
```

```
  * Matrix: 224 hours on one CPU of the Cray-C916 at SARA, Amsterda
            the matrix had  6 699 191 rows and  6 711 336 columns,
            and weight  417 132 631 ( 62.27 nonzeros per row);
            calendar time: ten days
  * Square root:  Four jobs assigned one dependency each were run
                  in parallel on separate 300 MHz R12000 processors
                  within a 24-processor SGI Origin 2000 at CWI.
                  One job found the factorisation after 39.4 CPU-ho
  ...
  * The total calendar time for factoring RSA-155 was 5.2 months
    (March 17 - August 22)
    (excluding polynomial generation time)
    We could reduce this to one month sieving time and
    one month processing time if we had more sievers and
    had more experience with matrix-generation strategies.
Address: (of contact person)
Email:   Herman.te.Riele@cwi.nl
```

Sparse interpolation in Chebyshev basis

Chebyshev basis:

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$$

Sparse polynomial in Chebyshev basis:

$$f(x) = \sum_{j=1}^{t} c_j T_{\delta_j}(x), \quad 0 \leq \delta_1 < \delta_2 < \cdots < \delta_t$$

Interpolation problem: find $t, \delta_j \in \mathbb{Z}_{\geq 0}$ and $c_j \in \mathbb{F}$

For some $p \in \mathbb{F}$, define an auxiliary polynomial $\Lambda(z)$

$$\begin{aligned}\Lambda(z) &= \prod_{j=1}^{t} (z - T_{\delta_i}(p)) \\ &= T_t(z) + \lambda_{t-1} T_{t-1}(z) + \cdots + \lambda_0 T_0(z).\end{aligned}$$

Key idea [Lakshman & Saunders 1992]: Let $a_i = f(T_i(p))$

$$\forall i\colon \sum_{j=0}^{t-1} \lambda_j (a_{j+i} + a_{|j-i|}) = -(a_{t+i} + a_{|t-i|}).$$

The λ_j are solutions to a symmetric Toeplitz+Hankel system

$$\begin{bmatrix} 2a_0 & 2a_1 & \ldots & 2a_{t-1} \\ 2a_1 & a_2 + a_0 & \ldots & a_t + a_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 2a_{t-1} & a_t + a_{t-2} & \ldots & a_{2t-2} + a_0 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \\ \vdots \\ \lambda_{t-1} \end{bmatrix} = - \begin{bmatrix} 2a_t \\ a_{t+1} + a_{t-1} \\ \vdots \\ a_{2t-1} + a_1 \end{bmatrix}.$$

How to make leading principal submatrices non-singular?

Wen-shin Lee [2001]: Pick a random $p \in S \subset \mathbb{F}$
Gohberg-Koltracht [1989] algorithm finds t, λ_j
in $O(t^2)$ arithmetic steps

Kaltofen & Saunders [1991, 2001]: Precondition coefficient matrix:

$$\begin{bmatrix} 1 & v_2 & v_3 & \dots & v_n \\ 0 & 1 & v_2 & \dots & v_{n-1} \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & v_2 \\ 0 & \dots & & & 1 \end{bmatrix} \cdot (\text{Toeplitz} + \text{Hankel}) \cdot \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ w_2 & 1 & 0 & & 0 \\ w_3 & w_2 & 1 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \\ w_n & w_{n-1} & \dots & w_2 & 1 \end{bmatrix}$$

is for random v_j, w_j Toeplitz+Hankel-like with generic rank profile.

Black box matrix concept

$A \in \mathbb{F}^{n \times n}$ singular

$\mathbb{F}$ an arbitrary, e.g., finite field

Perform linear algebra operations, e.g., $A^{-1}b$ [Wiedemann 86] with

$O(n)$	black box calls and
$n^2(\log n)^{O(1)}$	arithmetic operations in $\mathbb{F}$ and
$O(n)$	intermediate storage for field elements

Black box model is useful for dense, structured matrices

$$\begin{bmatrix} 1 & \cdots & & \cdots & \frac{1}{n} \\ & & \vdots & & \\ & & \frac{1}{i+j-1} & & \\ & & \vdots & & \\ \frac{1}{n} & \cdots & & \cdots & \frac{1}{2n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

(Hilbert matrix)

Savings may be in space, not time: $O(1)$ vs. $O(n^2)$.

Idea for Wiedemann's algorithm

$A \in \mathbb{F}^{n\times n}$, $\mathbb{F}$ a (possibly finite) field

$\phi^A(\lambda) = c'_0 + \cdots + c'_m \lambda^m \in \mathbb{F}[\lambda]$ **minimum polynomial** of A

$\forall\, u, v \in \mathbb{F}^n\colon\ \forall\, j \geq 0\colon$

$$u^{Tr} A^j \phi^A(A) v = 0$$

$$\Updownarrow$$

$$c'_0 \cdot \underbrace{u^{Tr} A^j v}_{a_j} + c'_1 \cdot \underbrace{u^{Tr} A^{j+1} v}_{a_{j+1}} + \cdots + c'_m \cdot \underbrace{u^{Tr} A^{j+m} v}_{a_{j+m}} = 0$$

$$\Updownarrow$$

$\{a_0, a_1, a_2, \ldots\}$ is generated by a linear recursion

Theorem [Wiedemann 1986]: *For* ***random*** $u, v \in \mathbb{F}^n$, *a linear generator for* $\{a_0, a_1, a_2, \ldots\}$ *is one for* $\{I, A, A^2, \ldots\}$.

$$\forall\, j \geq 0:\ \ c_0 a_j + c_1 a_{j+1} + \cdots + c_d a_{j+d} = 0$$

$$\Downarrow \text{(with high probability)}$$

$$c_0 A^j v + c_1 A^{j+1} v + \cdots + c_d A^{j+d} v = \mathbf{0}$$

$$\Downarrow \text{(with high probability)}$$

$$c_0 A^j + c_1 A^{j+1} + \cdots + c_d A^{j+d} = \mathbf{0}$$

that is, with high probability $\phi^A(\lambda)$ divides $c_0 + c_1\lambda + \cdots + c_d\lambda^d$

Algorithm homogeneous Wiedemann

Input: $A \in \mathbb{F}^{n \times n}$ singular
Output: $w \neq \mathbf{0}$ such that $Aw = \mathbf{0}$

Step W1: Pick random $u, v \in \mathbb{F}^n$; $\quad b \leftarrow Av$;
for $i \leftarrow 0$ **to** $2n-1$ **do** $a_i \leftarrow u^{Tr} A^i b$.
(Requires $2n$ black box calls.)

Step W2: Compute a linear recurrence generator for $\{a_i\}$,
$c_\ell \lambda^\ell + c_{\ell+1}\lambda^{\ell+1} + \cdots + c_d \lambda^d, \quad \ell \geq 0, d \leq n, c_\ell \neq 0.$

Step W3: $\widehat{w} \leftarrow c_\ell v + c_{\ell+1} A v + \cdots + c_d A^{d-\ell} v$;
(With high probability $\widehat{w} \neq 0$ and $A^{\ell+1}\widehat{w} = 0$.)
Compute first k with $A^k \widehat{w} = 0$; **return** $w \leftarrow A^{k-1}\widehat{w}$.
(Requires $\leq n$ black box calls.)

Step W2 detail

Coefficients $c_0, \ldots, c_n$ can be found by computing a non-trivial solution to the Toeplitz system

$$\begin{bmatrix} a_n & a_{n-1} & \cdots & & a_1 & a_0 \\ a_{n+1} & a_n & & & a_2 & a_1 \\ \vdots & a_{n+1} & \ddots & & \vdots & a_2 \\ & \vdots & & & & \vdots \\ a_{2n-2} & & & \ddots & a_{n-1} & \\ a_{2n-1} & a_{2n-2} & \cdots & & a_n & a_{n-1} \end{bmatrix} \cdot \begin{bmatrix} c_n \\ c_{n-1} \\ c_{n-2} \\ \vdots \\ \\ c_0 \end{bmatrix} = \mathbf{0}$$

or by the Berlekamp/Massey algorithm.

Cost: $O(n(\log n)^2 \operatorname{loglog} n)$ arithmetic ops.

Flurry of recent results

Lambert [96], Teitelbaum [98], Eberly & Kaltofen [97]	relationship of Wiedemann and Lanczos approach
Villard [97]	analysis of **block** Wiedemann algorithm
Giesbrecht [97] and Mulders & Storjohann [99]	computation of **diophantine** solutions
Giesbrecht, Lobo & Saunders [98]	certificates for inconsistency
Chen, Eberly, Kaltofen, Saunders, Villard & Turner [2K]	butterfly network, sparse and diagonal preconditioners
Villard [2K] & Storjohann [01]	characteristic polynomial
Kaltofen & Villard [2K]	fast algorithm for determinant of a **dense** integer matrix

Life after Strassen matrix multiplication: bit complexity

Linear system solving $x = A^{-1}b$ where $A \in \mathbb{Z}^{n\times n}$ and $b \in \mathbb{Z}^n$:

With Strassen and Chinese remaindering [McClellan 1973]:

Step 1: For prime numbers $p_1, \ldots, p_k$ Do
Solve $Ax^{[j]} \equiv b \pmod{p_j}$ where $x^{[j]} \in \mathbb{Z}/(p_j)$

Step 2: Chinese remainder $x^{[1]}, \ldots, x^{[k]}$ to $A\overline{x} \equiv b \pmod{p_1 \cdots p_k}$

Step 3: Recover denominators of x_i by continued fractions of $\dfrac{\overline{x}_i}{p_1 \cdots p_k}$.

Length of integers: $k = (n \ \max\{\log \|A\|, \log \|b\|\} \ )^{1+o(1)}$

Bit complexity: $n^{3.38} \ \max\{\log \|A\|, \log \|b\|\}^{1+o(1)}$

With Hensel lifting [Moenck and Carter 1979, Dixon 1982]:

Step 1: For $j = 0, 1, \ldots, k$ and a prime p Do

Compute $\overline{x}^{[j]} = x^{[0]} + px^{[1]} + \cdots + p^j x^{[j]} \equiv x \pmod{p^{j+1}}$

1.a. $\widehat{b}^{[j]} = \dfrac{b - A\overline{x}^{[j-1]}}{p^j} = \dfrac{\widehat{b}^{[j-1]} - Ax^{[j-1]}}{p}$

1.b. $x^{[j]} \equiv A^{-1}\widehat{b}^{[j]} \pmod{p}$ reusing $A^{-1} \bmod p$

Step 2: Recover denominators of x_i by continued fractions of $\dfrac{\overline{x}_i^{[k]}}{p^k}$.

With classical matrix arithmetic:

Bit complexity of 1.a: $n(n \max\{\log\|A\|, \|b\|\})^{1+o(1)} + n^2(\log\|A\|)^{1+o(1)}$

Total bit complexity: $(n^3 \max\{\log\|A\|, \log\|b\|\})^{1+o(1)}$

Diophantine solutions
by Giesbrecht, Mulders&Storjohann:
Find several rational solutions.

$$A(\tfrac{1}{2}x^{[1]}) = b, \quad x^{[1]} \in \mathbb{Z}^n$$
$$A(\tfrac{1}{3}x^{[2]}) = b, \quad x^{[2]} \in \mathbb{Z}^n$$

$$\gcd(2,3) = 1 = 2 \cdot 2 - 1 \cdot 3$$
$$A(2x^{[1]} - x^{[2]}) = 4b - 3b = b$$

$\Longrightarrow$ Can compute **integral** solutions of **sparse** linear systems.

Matrix determinant definition

$$\det(Y) = \det\left(\begin{bmatrix} y_{1,1} & \cdots & y_{1,n} \\ y_{2,1} & \cdots & y_{2,n} \\ \vdots & & \vdots \\ y_{n,1} & \cdots & y_{n,n} \end{bmatrix}\right) = \sum_{\sigma \in S_n} \left(\operatorname{sign}(\sigma) \prod_{i=1}^{n} y_{i,\sigma(i)}\right),$$

where $y_{i,j}$ are from an *arbitrary commutative ring*,
and S_n is the set of all permutations on $\{1, 2, \ldots, n\}$.

Interesting rings: $\mathbb{Z}$, $\mathbb{K}[x_1, \ldots, x_n]$, $\mathbb{K}[x]/(x^n)$

Why the determinant complexity is important

Theorem [Giesbrecht 1992]
Suppose you have a Monte Carlo randomized algorithm on a random access machine that can compute the determinant of an $n \times n$ *matrix in* $D(n)$ *arithmetic operations.*

Then you have a Monte Carlo randomized algorithm on a random access machine that can multiply two $n \times n$ *matrices in* $O(D(n))$ *arithmetic operations.*

No proof is known for Las Vegas or deterministic algorithms.

Bit complexity of the determinant

With Chinese remaindering: $(n \log \|A\|)^{1+o(1)}$ times matrix multiplication complexity.

Sign of the determinant [Clarkson 92]: $n^{4+o(1)}$ if matrix is ill-conditioned.

Using denominators of linear system solutions [Abbott, Bronstein, Mulders 1999]: fast when large first invariant factor.

Using fast Smith form method $n^{3.5+o(1)}(\log \|A\|)^{2.5+o(1)}$ [Eberly, Giesbrecht, Villard 2000]

Baby steps/giant steps algorithm [Kaltofen 1992/2000]

Wiedemann randomly perturbs A and chooses random u and v; then $\det(\lambda I - A)$ = minimal recurrence polynomial of $\{a_i\}_{i=0,1,\ldots 2n-1}$.

Detail of sequence $a_i = u^T A^i v$ computation

Let $r = \lceil \sqrt{2n}\, \rceil$ and $s = \lceil 2n/r \rceil$.

Substep 1. For $j = 1, 2, \ldots, r-1$ Do $v^{[j]} \leftarrow A^j v$;

Substep 2. $Z \leftarrow A^r$;
[$O(n^3)$ operations; integer length $(\sqrt{n}\ \log \|A\|)^{1+o(1)}$]

Substep 3. For $k = 1, 2, \ldots, s$ Do $u^{[k]^T} \leftarrow u^T Z^k$;
[$O(n^{2.5})$ operations; integer length $(n\ \log \|A\|)^{1+o(1)}$]

Substep 4. For $j = 0, 1, \ldots, r-1$ Do
For $k = 0, 1, \ldots, s$ Do $a_{kr+j} \leftarrow \langle u^{[k]}, v^{[j]} \rangle$.

The state-of-the-art [Kaltofen & Villard 2001]

Theorem 1
The determinant of an integer matrix can be computed in $O(n^{2.698}(\log \|A\|)^{1+o(1)})$ *bit operations.*

Theorem 2
The determinant and adjoint of a matrix over a commutative ring can be computed with $O(n^{2.698})$ *ring additions, subtractions and multiplications.*

Problem 1 (from my 3ECM 2000 talk)
Improve the bit complexity of algorithms for the determinant, resultant, linear system solution, Toeplitz systems, over the integers.

Coppersmith's 1992 blocking

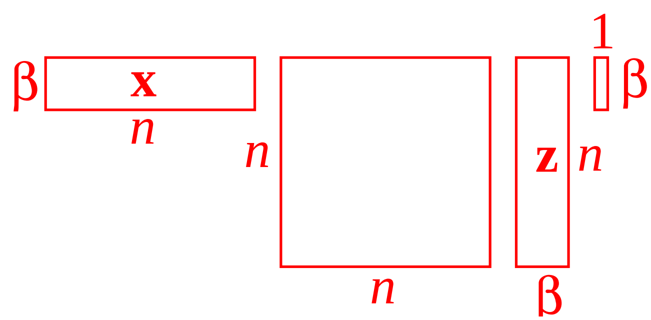
Use of the block vectors $\mathbf{x} \in \mathbb{F}^{n\times\beta}$ in place of u

$\mathbf{z} \in \mathbb{F}^{n\times\beta}$ in place of v

$$\mathbf{a}_i = \mathbf{x}^{Tr} A^{i+1} \mathbf{z} \in \mathbb{F}^{\beta\times\beta}, \quad 0 \le i < 2n/\beta + 2.$$

Find a **vector** polynomial $c_\ell \lambda^\ell + \cdots + c_d \lambda^d \in \mathbb{F}^\beta[\lambda]$, $d = \lceil n/\beta \rceil$:

$$\forall\, j \ge 0\colon \ \sum_{i=\ell}^{d} \mathbf{a}_{j+i} c_i = \sum_{i=\ell}^{d} \mathbf{x}^{Tr} A^{i+j}\, A \mathbf{z} c_i = \mathbf{0} \in \mathbb{F}^{\beta\times\beta}$$

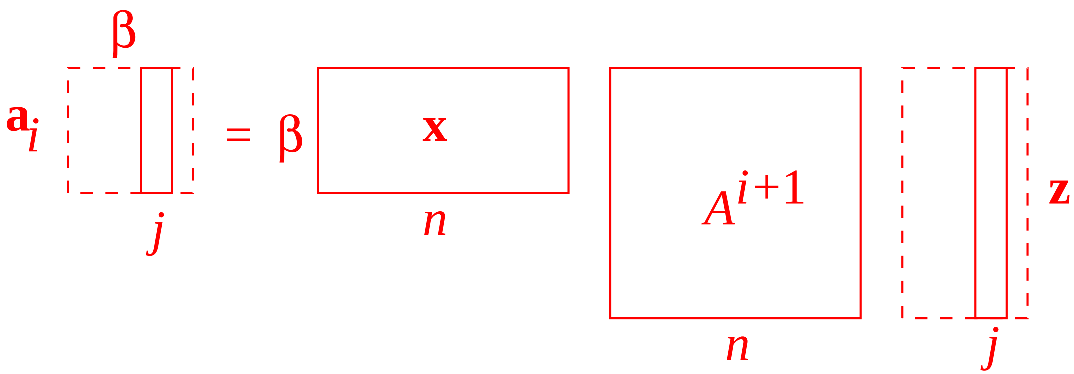


Then, analogously to before, with high probability

$$\widehat{w} = \sum_{i=\ell}^{d} A^{i-\ell} \mathbf{z} c_i \neq \mathbf{0}, \quad A^{\ell+1}\widehat{w} = \sum_{i=\ell}^{d} A^i A \mathbf{z} c_i = \mathbf{0} \in \mathbb{F}^n$$

Advantages of blocking

1. Parallel coarse- and fine-grain implementation



The j^{th} processor computes the j^{th} column of the sequence of (small) matrices.

2. Faster sequential running time:
   multiple solutions [Coppersmith; Montgomery 1994];
   $1 + \epsilon$ matrix times vector ops [Kaltofen 1995];
   determinant [Kaltofen & Villard 2000];
   charpoly of sparse matrix [Villard & Storjohann 2001]

3. Better probability of success [Villard 1997]

Analysis of blocking

All vector polynomials that generate $\{\mathbf{a}_i\}$ form a **module** over $\mathbb{F}[\lambda]$.

β vectors of minimal degree determinant from a $\mathbb{F}[\lambda]$-basis

A matrix canonical (Popov, Hermite) version of the basis defines a unique minimal polynomial $\mathbf{c}_0 + \mathbf{c}_1\lambda + \cdots + \mathbf{c}_d\lambda^d \in \mathbb{F}^{\beta\times\beta}[\lambda]$

From $(I - \lambda B)^{-1} = I + B\lambda + B^2\lambda^2 + \cdots$

$$\mathbf{x}^{Tr}(I - \lambda B)^{-1}\mathbf{y}(\mathbf{c}_d + \cdots + \mathbf{c}_0\lambda^d) = R(\lambda) \in \mathbb{F}[\lambda]^{\beta\times\beta}$$

we obtain a matrix Padé approximation (“realization”)

$$\mathbf{x}^{Tr}(I - \lambda B)^{-1}\mathbf{y} = \sum_i \mathbf{a}_i\lambda^i = R(\lambda)(\mathbf{c}_d + \cdots + \mathbf{c}_0\lambda^d)^{-1}$$

Computation of the matrix linear generator

Explicitly in Popov form by block Berlekamp/Massey algorithm [Coppersmith 1994, Thomé 2001] or implicitly in a block Lanczos version

Explicitly by a power Hermite-Padé approximation
[Beckermann & Labahn 1994]

By a block Toeplitz solver [Kaltofen 1995]

Software

By Coppersmith, Kaltofen & Lobo, Montgomery, Brent, W.-s. Lee,...

The LinBox group [Canada: Calgary, UWO, Waterloo; France: ENS Lyon, IMAG Grenoble, INRIA Sophia Antipolis; USA: Delaware, NCSU, Washington Coll. MD]: A generic C++ library for black box linear algebra, including integer problems

Open Problems

Black box methods: Compute the characteristic polynomial
Certify the minimal polynomial, rank

Structured methods: Superfast algorithms for resultant matrices
Subquadratic bit complexity for Toeplitz problems

Symbolic/numeric: How to interweave both methodologies?