On the complexity of computing determinants and other challenges in symbolic computation.

Erich Kaltofen
North Carolina State University
www.kaltofen.net
Matrix determinant definition

\[
\det(\begin{vmatrix} y_1 & \cdots & y_n \\ \vdots & \ddots & \vdots \\ y_1 & \cdots & y_n \end{vmatrix}) = \sum_{\sigma \in S_n} \text{sign}(\sigma) y_{\sigma(1)} \cdots y_{\sigma(n)}
\]

where \(y_{i,j}\) are from an arbitrary commutative ring, \(S_n\) is the set of all permutations on \(\{1, 2, \ldots, n\}\), and \(\text{sign}(\sigma)\) is the sign of the permutation \(\sigma\).

Interesting rings: \(\mathbb{Z}\), \(K[x_1, \ldots, x_n]\), \(K[x] = \langle x_n \rangle\).

Matrix determinant definition
Fast matrix multiplication

Strassen’s [1969] \( O(2^{\log_27}) \) algorithm

Problems reducible to matrix multiplication:

- Linear systems solving, determinants [Bunch and Hopcroft 1974],...

Coppersmith and Winograd [1990]: \( O(2^{\log_26}) \)

...
No proof is known for Las Vegas or deterministic algorithms.

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

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 $O(D(n))$ arithmetic operations. Why the determinant complexity is important?
Perform linear algebra operations, e.g., \[ Wiedemann \] with intermediate storage for field elements and arithmetic operations in \( \mathbb{F} \) and black box calls and (\[ u \]) \( O(\log n)^2 \) \( u \) \( O(1) \) \( O(n) \) arithmetic operations in \( \mathbb{F} \) and \( O(n) \) intermediate storage for field elements.
Blackbox model is useful for dense, structured matrices.

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

(Hilbert matrix)

\[
\begin{bmatrix}
u_1 \\ \vdots \\ u_n
\end{bmatrix} = \begin{bmatrix}1 & \cdots & \cdots & 1 \\ \vdots & \frac{u_1}{1} & \cdots & \frac{u_{n-1}}{1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{u_1} & \frac{1}{u_2} & \cdots & \frac{1}{u_n}
\end{bmatrix} \begin{bmatrix}1 \\ \vdots \\ 1
\end{bmatrix}
\]
Idea for Wiedemann's algorithm

is generated by a linear recursion

\[ 0 = a_{\omega+\ell_D} V_{\omega L}^n \cdot w + \cdots + a_{\ell_D} V_{\omega L}^n \cdot \ell + \sum_{m} a_{\ell_D} V_{\omega L}^n \cdot 0 \]

\[ 0 = a(V)_{\omega} \ell D V_{\omega L}^n \]

\[ 0 \geq \ell A : \nu \nu \omega \nu \in a, n \omega A \]

\vspace{1cm}

minimum polynomial of \( V \)

\[ [\lambda] \nu \nu \omega \nu \in u \nu \nu \omega \nu + \cdots + \nu \omega = (\lambda)_{\omega} V^\phi \]

\vspace{1cm}

a (possibly finite) field

\vspace{1cm}

Idea for Wiedemann's algorithm
Theorem [Wiedemann 1986]: For random \( u, v \in \mathbb{F}_p \), the vector \( \mathbf{v} \) has a linear generator for \( \mathbb{F}_p \) with high probability.

That is, with high probability, \( \mathbf{v} \) divides \( \chi \phi(\mathbf{v}) \).

\[
0 = p + c_0 \alpha^p + \cdots + c_d \alpha^d + \mathbf{v}^0 \\
(\text{with high probability}) \\
0 = v_p + c_0 \alpha^p + \cdots + c_d \alpha^d + \mathbf{v}^0 \\
(\text{with high probability}) \\
0 = p + c_0 \cdot 0 + \cdots + c_d \cdot 0 \\
\text{A linear generator for } \mathbb{F}_p \\
\{ \cdots, \mathbf{v}, \mathbf{v}, \mathbf{v} \} \text{ is one for } \{ v_0, \alpha v_1, \alpha^2 v_2 \} \text{ for random } u, v \in \mathbb{F}_p.\]
Algorithm homogenous Wiedemann

Input: $A \in \mathbb{F}_n$ singular

Output: $w \in \mathbb{F}_n$ such that $Aw = 0$

Step W1: Pick random $u, v \in \mathbb{F}_n$; $b v$.

for $i = 0$ to $2n - 1$ do

Compute a linear recurrence generator for $\{v_i\}$.

$$A_i = c_i + c_{i+1} x + \ldots + c_{i+p} x^p \in \mathbb{F}_n[u,v].$$

Step W2: Compute $w$ with high probability $b w \neq 0$ and $A w = 0$ and $b w \neq 0$.

$\mathcal{T}_{\mathbb{F}_n}$: \text{Compute first } k \text{ with } A_k b w = 0; \text{return } 0.$

Step W3: (Requires $2n$ black box calls.)

Compute first $k$ with $A_k w = 0$; return $w$.

$\mathcal{T}_{\mathbb{F}_n}$: Pick random $u \in \mathbb{F}_n$.

Output: $0 = w \in \mathbb{F}_n$ such that $Aw = 0$.
Coefficients $c_0, \ldots, c_n$ can be found by computing a non-trivial solution to the Toeplitz system.

Cost: $O(n \log^2 n \log \log n)$ arithmetic ops.

or by the Berlekamp/Massey algorithm.

\[
0 = \begin{bmatrix}
0 \\
\vdots \\
2^{u_0} \\
1^{u_0} \\
u_0
\end{bmatrix}
\cdot
\begin{bmatrix}
1^{u_0} & u_0 & \cdots & 2^{u_0} & 1^{u_0} \\
u_0 & 1 & \cdots & 2^{u_0} & 1^{u_0} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
2^{u_0} & 1 & \cdots & 1^{u_0} & 2^{u_0} \\
u_0 & 1 & \cdots & 1^{u_0} & 1^{u_0}
\end{bmatrix}
\]
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Characteristic polynomial</td>
<td>Villard [2K] &amp; Storjohann [01]</td>
<td></td>
</tr>
<tr>
<td>Diagonal preconditionings</td>
<td>[2K] Villard &amp; Turner</td>
<td></td>
</tr>
<tr>
<td>Butterfly network, sparse and preconditioning</td>
<td>Chen, Eberly, Kaltofen</td>
<td></td>
</tr>
<tr>
<td>Certificates for inconsistency</td>
<td>Giesbrecht, Lobo &amp; Saunders [98]</td>
<td></td>
</tr>
<tr>
<td>Solutions of diophantine equations</td>
<td>Mulders &amp; Storjohann [99]</td>
<td></td>
</tr>
<tr>
<td>Computation of diophantine algorithm</td>
<td>[67] Giesbrecht, and</td>
<td></td>
</tr>
<tr>
<td>Analysis of block Wiedemann algorithm</td>
<td>Villard [26]</td>
<td></td>
</tr>
<tr>
<td>and Lanczos approach</td>
<td>Eberly &amp; Kaltofen [67]</td>
<td></td>
</tr>
<tr>
<td>Relationship of Wiedemann</td>
<td>Lambert [96], Teitelbaum [98], Kaltofen [2K]</td>
<td></td>
</tr>
<tr>
<td>Flurry of recent results</td>
<td>[10] Turner</td>
<td></td>
</tr>
</tbody>
</table>

| 10 |
Life after Strassen: 2. Bit complexity

$$\max \{ \| q \| \log \| A \| \log \} \leq u \cdot 3.38$$

Length of integers: \( n \) = \( \log \| q \| \max \{ u \} + 1 \)

Step 1: For prime numbers \( p_1, \ldots, p_k \), do

- Solve \( Ax \equiv b \pmod{p_j} \) for \( x \)

Step 2: Chinese remainder \( x \). By continued fractions of \( x \) by \( \{ p_1, \ldots, p_k \} \), recover denominators of \( x \).

Step 3: \( q \equiv xA \pmod{p_1 \cdot \ldots \cdot p_k} \)

With Strassen and Chinese remaindering [McGillan 1973]:

- \( u \log \| q \| \log \| A \| \log \} \leq u \cdot 3.38 \)
- Linear system solving \( q_{\| A \| - 1} x = A \)
Step 1: For \( j = 0; 1; \ldots; k \) and a prime \( p \) Do

\[
\text{Compute } x[j] = x[0] + px[1] + \ldots + px^{j} \quad \text{mod } p^{j} + 1
\]

1.a.

\[
\frac{y[d]}{[y][x]}
\]

Step 2: Recover denominators of \( x \) by continued fractions of \( x \).

With classical matrix arithmetic:

\[
(1)_{o+1}(\|A\|, o+1) \cdot u + (1)_{o+1}(\|q\|, \|V\|, o+1) \cdot u
\]

Total bit complexity:

Bit complexity of 1.a.

With Hensel lifting [Moenck and Carter 1979, Dixon 1982]:
Can compute integral solutions of sparse linear systems.

\[ q = q_3 q_4 = ([\mathbb{Z}]x \cdot [1]x) V \]
\[ \gcd(2, 3) = 1 = 2 \cdot 2 - 1 \cdot 3 \]
\[ u \mathbb{Z} \ni [\mathbb{Z}]x \quad q = ([\mathbb{Z}]x \xi) V \]
\[ u \mathbb{Z} \ni [1]x \quad q = ([1]x \xi) V \]

Find several rational solutions.

By Giesbrecht, Hildebrandt, Storjohann: Diophantine solutions.
Bit complexity of the determinant

With Chinese remaindering:

\[ (\lfloor A \rfloor \log u) o + u \]

With fast Smith form method:

\[ o(1) (\lfloor A \rfloor \log u) \]

\[ \log k \]

Sign of the determinant [Clarkson 92]:

\[ n^4 (\lfloor A \rfloor \log u) \]

\[ o(1) \]

\[ k \]

Using fast Smith form method:

\[ o(1) (\lfloor A \rfloor \log u) \]

Using [Giesbrecht, Villard 2000]:

\[ \exp \left( \frac{3}{2} \log u \right) \left( \left\| V \right\| \right) \]

Using fast Smith form method:

\[ o(1) (\lfloor A \rfloor \log u) \]

Using denominators of linear system solutions [Abbott, Bronstein, Mulders 1999]:

\[ n^4 (\lfloor A \rfloor \log u) \]

\[ o(1) \]

Fast when large first invariant factor.

Using denominators of linear system solutions [Abbott, Bronstein, Mulders 1999]:

\[ n^4 (\lfloor A \rfloor \log u) \]

\[ o(1) \]

Using fast Smith form method:

\[ o(1) (\lfloor A \rfloor \log u) \]

Using fast Smith form method:

\[ o(1) (\lfloor A \rfloor \log u) \]
\begin{align*}
\langle [f]n, [g]n \rangle & \rightarrow f^+ + n \\
\text{For } \ell = 0, 1, \ldots, s \text { Do } n & \\
\text{Substep 4. For } \ell = 0, 1, \ldots, \ell - 1 \text { Do }
\end{align*}

\begin{align*}
[(I_0 + (||A|| \log u) O)] & \\
\text{Substep 3. For } \ell = 1, 2, \ldots, \ell \text { Do } n & \\
\end{align*}

\begin{align*}
\text{Substep 2. For } \ell = 1, 2, \ldots, \ell \text { Do } Z \rightarrow Z & \\
\end{align*}

\begin{align*}
\text{Substep 1. For } \ell = 1, 2, \ldots, \ell \text { Do } A & \\
\text{Let } r & \\
\text{Det of sequence } \det \{q_\ell \}_{\ell=0,1,2, \ldots} & = \det(A - I)X = (A^T \Omega - \Omega A) \text{ precondition of polynomial of degree } = 0, 1, \ldots, 2n - 1. \\
\text{Viehmann precondition } \Omega \text{ and choose random } \Omega \text{ and } \omega \text{ then }
\end{align*}

Baby steps/giant steps algorithm [Kaltofen 1992/2000]
The state-of-the-art \cite{KaltofenVillard2001}.

**Theorem 1**

The determinant of an integer matrix can be computed in $O(n^{2.698})$ 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, multiplications.

**Problem 1**

(from my 3EGM 2000 talk)

Improve the bit complexity of algorithms for the determinant, resultant, linear system solution, Toeplitz systems, over the integers.

**Problem 2**

Improve the bit complexity of algorithms for the determinant, resultant, linear system solution, Toeplitz systems, over the integers.
Then, analogously to before, with high probability

\[ u \in 0 = \text{vector polynomial} \quad \exists \mathbf{x} \quad \text{s.t.} \quad \mathbf{x} = \mathbf{A}^{-1} \mathbf{z} \]

Find a vector polynomial \( \mathbf{g} \times g \mathbb{F} \in 0 = \text{vector polynomial} \quad \exists \mathbf{x} \quad \text{s.t.} \quad \mathbf{x} = \mathbf{A}^{-1} \mathbf{z} \)

\[ \mathbf{g} \times g \mathbb{F} \in 0 = \text{vector polynomial} \quad \exists \mathbf{x} \quad \text{s.t.} \quad \mathbf{x} = \mathbf{A}^{-1} \mathbf{z} \]

\( \mathbf{g} \times g \mathbb{F} \in 0 = \text{vector polynomial} \quad \exists \mathbf{x} \quad \text{s.t.} \quad \mathbf{x} = \mathbf{A}^{-1} \mathbf{z} \)

Use of the block vectors in place of the block vectors in place of

Coppersmith's 1992 blocking
Advantages of blocking

1. Parallel coarse- and fine-grain implementation

\[ \beta = A_i + x \]

- The \( j \)th processor computes the \( j \)th column of the sequence of (small) matrices.

\[ \mathbf{g} = \begin{bmatrix} f \\ \mathbf{x} \\ \mathbf{u} \end{bmatrix} \]

2. Faster sequential running time:

- a. Multiple solutions [Coppersmith; Montgomery 1994];
- b. I + e matrix times vector ops [Kaltofen 1995];
- c. Determinant [Kaltofen & Villard 2000];
- d. Characteristic of sparse matrix [Villard & Storjohann 2001].
Analysis of blocking

All vector polynomials that generate \( f \) form a module over \( F \),

we obtain a matrix Padé approximation (realization)

\[
\begin{align*}
\mathcal{A}_I - (\forall \chi - I)_{\mathcal{A}} \mathbf{x} \\
\mathcal{A}_I - (\forall \chi - I)_{\mathcal{A}} \mathbf{x}
\end{align*}
\]

A unique minimal polynomial defines a matrix canonical (Popov, Hermite) version of the basis
denotes vectors of minimal degree determinant form an \( \mathcal{A} \)-basis

\[
\begin{align*}
[\chi]_{\mathcal{A}} \mathbf{x} &\in p\chi^p + \cdots + \chi^0 + 0 \\
\mathcal{A}_I - (\forall \chi - I)_{\mathcal{A}} \mathbf{x}
\end{align*}
\]

All vector polynomials that generate \( \{?a\} \) form a module over \( F \).

Analysis of blocking
Theorem

Let $A \in F_n^n$; $x, y \in F_n^n$ denote all invariant factors of $A$.

Then for all $i$, the $i$-th invariant factors of $c_p X^0 + \cdots + c_p$ divide $s_i$.

Furthermore, for random projection matrices $x \in S_n \times \mathbb{P}$, $y \in S_n \times \mathbb{P}$, $x \in S_n \times \mathbb{P}$ for all $y$, the $i$-th invariant factors of $c_p X^0 + \cdots + c_p$ are equal to $s_i$ with probability $1 - \frac{|S|}{2n}$.

Advantages of blocking continued:

3. Captures multiple invariant factors.

[Viillard 1997]
Computation of the matrix linear generator

Explicitly by a power Hermite-Padé approximation

By a block Toeplitz solver [Kaltofen 1995]

Explicitly in Popov form by block Berlekamp/Masseny algorithm

[Becker and Labahn 1994]

By a block Toeplitz solver [Kaltofen 1995]

Explicitly by a power Hermite-Padé approximation

[Becker and Labahn 1994]

Explicitly in Popov form by block Berlekamp/Masseny algorithm

Computation of the matrix linear generator
Factorization of nearby polynomials over the complex numbers.
Given is a polynomial \( f(x, y) \in \mathbb{Q}[x, y] \) with a factor of a constant degree and coefficient size if there is an \( \tilde{f}(x, y) \in \mathbb{C}[x, y] \) with coefficient size at most \( \epsilon \) and degree at most \( \delta \) such that \( \| f - \tilde{f} \| \leq \epsilon \),

for a reasonable coefficient norm.

We can compute in polynomial time in the degree and coefficient size.

**Theorem** [Hitz, Kaltofen, Lakshman ISSAC’99]

For a reasonable coefficient vector norm

\[
\| f \| \geq \delta \| \text{deg } f \| \quad \text{and} \quad \| \text{deg } \tilde{f} \| \geq \| f - \tilde{f} \|.
\]

Given is a polynomial \( f(x, y) \in \mathbb{Q}[x, y] \) and \( e \in \mathbb{Q} \) with a factorizable \( \tilde{f}(x, y) \in \mathbb{C}[x, y] \) with coefficient size at most \( \epsilon \) and degree at most \( \delta \) such that \( \| f - \tilde{f} \| \leq \epsilon \),

Problem 2 [Kaltofen LATIN’92]
Numerical algorithms

Conclusion on my exact algorithm [JSC 1985]:

“D. Izraelevitz at Massachusetts Institute of Technology has already implemented a version of algorithm 1 using complex floating point arithmetic. Early experiments indicate that the linear systems computed in step (L) tend to be numerically ill-conditioned. How to overcome this numerical problem is an important question which we will investigate.”


Stetter, Huang, Wu and Zhi [ISSAC’2K]: Hensel lift factor combinations numerically and eliminate extraneous factors early.

Corless, Giesbrecht, Kotsiras, van Hoeij, Watt [ISSAC’01]: sample curve by points and interpolate.
Will our systems guarantee their answers?

Basic polynomial algorithms with floating point coefficients are under development.

Maple 6 allows calls to NAG numeric library routines.

Will the system guarantee their answers?
\[ f := \frac{1}{\sin(x) + 2} \]

\[ g := \int f \, dx \]

\[ \text{plot}(g, x = -5..5); \]

\[ \forall x \in \mathbb{R}, \quad x = -5 \cdot 5 \]

\[ (\exists y \in \mathbb{R}, \ y = 5) \]

\[ \left( \exists \left( x \cdot \frac{z}{I} \right) \text{such that } \frac{z}{I} \right) \]

\[ \exists \left( x \cdot \frac{y}{I} \right) \text{ such that } \frac{y}{I} \]

\[ \exists \left( z + \frac{x \cdot \sin}{I} \right) = f \]

\[ \exists \left( z + \frac{(x \cdot \sin) / I}{I} \right) = f \]

Example by Corless and Jeffrey #
Early termination strategies

In order to obtain a better probability, we require $c_i = 0$ more than once before termination.

Threshold $u$

For $i = 1, 2, \ldots$

1. Pick random $p_i$ and from $f(p_i)$ compute

$$((1 + d - x) \cdots (1d - x) \mod) \quad (x)f \equiv$$

$$\cdots + (2d - x)(1d - x)c_2 + (1d - x)c_1 + c_0 \rightarrow (x)[i]f$$

2. If $c_i = 0$ stop.

End For

Pick random $f(p)$ and $f(0)$

For $i = 1, 2, \ldots$

Early termination in Newton interpolation

Early termination strategies
Kaltofen and Wen-shin Lee [2000] have developed early termination strategies for sparse interpolation algorithms.

Wen-shin Lee has a new heuristic to find a sparse shift, i.e., a number $a$ such that $(a - y) f$ has the minimum terms of the form $ye^i$.

Problem 3 [from my 2000 ECM talk]

Provide reasonable correctness specifications for our systems.