On the complexity of computing determinants

Erich Kaltofen
North Carolina State University
www.kaltofen.net

Overview

1. Faster bit complexity without Strassen matrix multiplication

2. New speed-ups: the use of blocking
   With Gilles Villard (middle)

Matrix determinant definition

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

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

1. Bit complexity of linear algebra problems

Strassen's [1969] $O(n^{2.81})$ matrix multiplication algorithm

$$
\begin{array}{l|l}
m_1 \leftarrow (a_{1,2} - a_{2,2})(b_{2,1} - b_{2,2}) & \\
m_2 \leftarrow (a_{1,1} + a_{2,2})(b_{1,1} + b_{2,2}) & \\
m_3 \leftarrow (a_{1,1} - a_{2,1})(b_{1,1} + b_{1,2}) & \\
m_4 \leftarrow (a_{1,1} + a_{1,2})b_{2,2} & a_{1,1}b_{1,1} + a_{1,2}b_{2,1} = m_1 + m_2 - m_4 + m_6 \\
m_5 \leftarrow a_{1,1}(b_{1,2} - b_{2,2}) & a_{1,1}b_{1,2} + a_{1,2}b_{2,2} = m_4 + m_5 \\
m_6 \leftarrow a_{2,2}(b_{2,1} - b_{1,1}) & a_{2,1}b_{1,1} + a_{2,2}b_{2,1} = m_6 + m_7 \\
m_7 \leftarrow (a_{2,1} + a_{2,2})b_{1,1} & a_{2,1}b_{1,2} + a_{2,2}b_{2,2} = m_2 - m_3 + m_5 - m_7
\end{array}
$$

Problems reducible to matrix multiplication:

linear system solving [Bunch and Hopcroft 1974],...

Coppersmith and Winograd [1990]: $O(n^{2.38})$

Life after Strassen: 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 [McClellan 1973]:

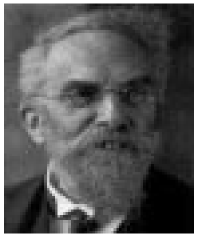
Step 1: For prime numbers $p_1, \ldots p_k$ Do
Solve $Ax^{[j]} \equiv b \pmod{p_j}$ where $x^{[j]} \in \mathbb{Z}/(p_j)$

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

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

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]:

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

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

1.a. $b^{[j]} = \dfrac{b - A\bar{x}^{[j-1]}}{p^j} = \dfrac{b - (A\bar{x}^{[j-2]} + Ap^{j-1}x^{[j-1]})}{p^j}$

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

Step 3: Recover denominators of x_i by continued fractions of $\dfrac{\bar{x}_i^{[k]}}{p^k}$.

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

Bit complexity of the determinant

With Chinese remaindering: $O(n \log \|A\|)$ times matrix multiplication complexity.

Sign [Clarkson early 1990s]: $O(n^3)$ floating point operations with $O(n)$ precision.

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

Wiedemann's [1986] determinant algorithm

For $u, v \in \mathbb{K}^n$ and $A \in \mathbb{K}^{n\times n}$ consider the sequence of field elements

$$a_0 = u^T v,\ a_1 = u^T A v,\ a_2 = u^T A^2 v,\ a_3 = u^T A^3 v,\ \ldots$$

The minimal polynomial of A linearly generates $\{a_i\}_{i=0,1,\ldots}$.

By the Berlekamp/Massey [1967] algorithm we can compute in $n^{1+o(1)}$ arithmetic operations a minimal linear generator for $\{a_i\}_{i=0,1,\ldots}$.

Wiedemann randomly perturbs A and chooses random u and v; then

$$\det(\lambda I - A) = \text{minimal recurrence polynomial of } \{a_i\}_{i=0,1,\ldots}$$

Detail of algorithm
[exactly like my division-free determinant algorithm ISSAC 92]

For $i = 0, 1, \ldots, 2n-1$ Do Compute the $a_i = u^T A^i v$;

Done by baby steps/giant steps: 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 $a_{kr+j} \leftarrow \langle u^{[k]}, v^{[j]} \rangle$.

Using fast rectangular matrix multiplication: $O(n^{3.064} \log \|A\|)$

Theorem 1
The determinant of an integer matrix can be computed in $O(n^{2.809} \log \|A\|)$ *bit operations.*
(Exponent 2.69 is a possibility but proofs need to be completed.)

Theorem 2
The determinant of a matrix over a commutative ring can be computed with $O(n^{2.69})$ *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, over the integers.

2. Coppersmith's blocking

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

$\mathbf{y} \in \mathbb{K}^{n\times\beta}$ in place of v

$$\mathbf{a}_i = \mathbf{x}^{\mathrm{Tr}} B^i \mathbf{y} \in \mathbb{K}^{\beta\times\beta}, \quad 0 \le i < \frac{2n}{\beta} + 2.$$

Find a matrix polynomial $\mathbf{c}_0 + \mathbf{c}_1\lambda + \cdots + \mathbf{c}_d\lambda^d \in \mathbb{K}^{\beta\times\beta}[\lambda]$, $d = \lceil n/\beta \rceil$, such that

$$\forall\, j \ge 0\colon \quad \sum_{i=0}^{d} \mathbf{a}_{j+i}\mathbf{c}_i = \sum_{i=0}^{d} \mathbf{x}^{\mathrm{Tr}} B^{i+j} \mathbf{y}\mathbf{c}_i = \mathbf{0} \in \mathbb{K}^{\beta\times\beta}$$

Probabilistic analysis

Theorem [K&V 2000]: *If B is nonsingular with distinct eigenvalues then we have for the **minimal** generating polynomial*

$$\det(\mathbf{c}_0 + \mathbf{c}_1\lambda + \cdots + \mathbf{c}_d\lambda^d) = \det(\lambda I - B)$$

*for **random** $\mathbf{x}$, $\mathbf{z}$ with probability*

$$\geq 1 - \frac{2n-1}{|\mathbb{K}|}.$$

Distinct eigenvalues can be obtained by preconditioning B à la [Wiedemann, 1986], for instance

$\widetilde{B} \leftarrow V \cdot B \cdot W \cdot G$ where V is randomized butterfly network
W is randomized butterfly network
G is random diagonal

Proof idea for probabilistic analysis

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

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

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

Use theorems from multivariable control theory (irreducible realizations) to show that polynomial denominators are the same.

Show run-time estimates in Maple worksheet

```
> beta := n^sigma; # blocking factor
```

$$\beta := n^\sigma$$

```
> s := n^tau; # number of giant steps
```

$$s := n^\tau$$

```
> r := simplify( (n/beta) / s); # number of baby steps
```

$$r := n^{(1-\sigma-\tau)}$$

Standard matrix arithmetic, quadratic B/M, Chinese remainder integer arithmetic

Step 1.1: Compute $B^j\, y$, $j = 0,...,r$

```
> substep1 := simplify( r * beta * n^2 * r );
```

$$\mathit{substep1} := n^{(4-\sigma-2\,\tau)}$$

Step 1.2: Compute $Z = B^r$ by repeated squaring

```
> substep2 := simplify( n^3 * r);
```

$$\mathit{substep2} := n^{(4-\sigma-\tau)}$$

Step 1.3: Compute $x^{Tr}\, Z^k$, $k = 0,...,s$

```
> substep3 := simplify( s * beta * n^2 * n/beta);
```

$$\mathit{substep3} := n^{(\tau+3)}$$

Step 1.4: Compute $(\, x^{Tr}\, Z^k)\, (\, B^j\, y)$

```
> substep4 := simplify( r * s * beta^2 * n * n/beta );
```

$$\mathit{substep4} := n^3$$

Step 2: Blocked Berlekamp/Massey for n moduli

```
> step2 := simplify( (n/beta)^2 * beta^3 * n );
```

$$step2 := n^{(3+\sigma)}$$

Step 3: Determinant of generator matrix polynomial for n moduli

```
> step3 := simplify( beta^3 * n * n );
```

$$step3 := n^{(3\,\sigma+2)}$$

Overall bit complexity

```
> eval([substep1, substep2, substep3, substep4, step2, step3],
>{sigma=1/3, tau=1/3});
```

$$[n^3,\ n^{(10/3)},\ n^{(10/3)},\ n^3,\ n^{(10/3)},\ n^3]$$

"*The asymptotically best algorithms frequently turn out to be worst on all problems for which they are used.*"

— D. G. CANTOR and H. ZASSENHAUS (1981)

Fast matrix multiplication O(n^ω), linear B/M, linear integer arithmetic

Step 1.1: Compute $B^j\,y$, $j = 0,\ldots,r$ by $B^{(2^i)}$ [$B^0\,y$ — ... — $B^{(2^i-1)}$ y]

```
> fastsubstep1 := simplify( n^omega * r );
```

$$fastsubstep1 := n^{(\omega+1-\sigma-\tau)}$$

Step 1.2: Compute $Z = B^r$ by repeated squaring

```
> fastsubstep2 := simplify( n^omega * r);
```

$$fastsubstep2 := n^{(\omega+1-\sigma-\tau)}$$

Step 1.3: Compute $x^{Tr}\,Z^k$, $k = 0,\ldots,s$ as *fat* vectors times a *thin* matrix

```
> fastsubstep3 := simplify( s * (n/(beta*s))^2 * (beta*s)^omega * r,
> 'power', 'symbolic' );
```

$$fastsubstep3 := n^{(-2\,\tau+3-3\,\sigma+(\sigma+\tau)\,\omega)}$$

Step 1.4: Compute $(x^{Tr} Z^k) (B^j y)$

```
> fastsubstep4 := simplify( r^2/s * (beta*s)^omega * n/beta, 'power',
> 'symbolic' );
```

$$fastsubstep4 := n^{(3-3\sigma-3\tau+(\sigma+\tau)\omega)}$$

Step 2: Blocked Berlekamp/Massey for n moduli

```
> faststep2 := simplify( beta^2 * n * n);
```

$$faststep2 := n^{(2\sigma+2)}$$

Step 3: Determinant of generator matrix polynomial for n moduli

```
> faststep3 := simplify( beta^omega * n, 'power', 'symbolic' );
```

$$faststep3 := n^{(\sigma\omega+1)}$$

Overall bit complexity

```
> total := eval([fastsubstep1, fastsubstep2, fastsubstep3,
> fastsubstep4, faststep2, faststep3]);
```

$$total := [n^{(\omega+1-\sigma-\tau)}, n^{(\omega+1-\sigma-\tau)}, n^{(-2\tau+3-3\sigma+(\sigma+\tau)\omega)}, n^{(3-3\sigma-3\tau+(\sigma+\tau)\omega)}, n^{(2\sigma+2)}, n^{(\sigma\omega+1)}]$$

```
> expos:=simplify(map(x -> log[n](x), total), 'symbolic');
```

$$expos := [\omega+1-\sigma-\tau, \omega+1-\sigma-\tau, -2\tau+3-3\sigma+\sigma\omega+\omega\tau, 3-3\sigma-3\tau+\sigma\omega+\omega\tau, 2\sigma+2, \sigma\omega+1]$$

```
> numexpos:=eval(expos, omega=2.3755);
```

$$numexpos := [3.3755-\sigma-\tau, 3.3755-\sigma-\tau, .3755\tau+3.-.6245\sigma, 3.-.6245\sigma-.6245\tau, 2\sigma+2, 2.3755\sigma+1]$$

```
> minexpo := solve({numexpos[2]=numexpos[3],
> numexpos[2]=numexpos[5]}, {sigma,tau});
```

$$minexpo := \{\sigma = .4042922554,\ \tau = .1626232338\}$$

```
> eval(numexpos, minexpo);
```

$$[2.808584511,\ 2.808584511,\ 2.808584510,\ 2.645961276,\ 2.808584511,\ 1.960396253]$$