Fast Deterministic Computation of Determinants of Dense Matrices

John Abbott
Dipartimento di Matematica
Università di Genova, Italy
abbott@dima.unige.it

Manuel Bronstein
INRIA – Projet CAFÉ
Sophia-Antipolis, France
Manuel.Bronstein@inria.fr

Thom Mulders
Institute of Scientific Computing
ETH Zurich, Switzerland
mulders@inf.ethz.ch

Abstract

In this paper we consider deterministic computation of the exact determinant of a dense matrix M of integers. We present a new algorithm with worst case complexity $O\left(n^4(\log n + \log ||M||) + n^3 \log^2 ||M||\right)$, where n is the dimension of the matrix and $||M||$ is a bound on the entries in M, but with average expected complexity $O\left(n^4 + n^3(\log n + \log ||M||)^2\right)$, assuming some plausible properties about the distribution of M. We will also describe a practical version of the algorithm and include timing data to compare this algorithm with existing ones. Our result does not depend on "fast" integer or matrix techniques.

1 Introduction

One of the most fundamental characteristics of a square matrix is its determinant. Its being 0 expresses the non–invertibility of the matrix, i.e. the fact that it has a non–trivial kernel. For a real matrix, its absolute value is the volume of the multi–dimensional parallelepiped with generating edges the rows (or columns) of the matrix (cf. [10]). Moreover, many geometrical properties can be expressed as the sign of a particular determinant (cf. [3]).

Another source of applications for determinant computations can be found in the area of normal forms of matrices. Normal forms of matrices, like the Hermite and Smith normal form, can be used to classify matrices and solve Diophantine systems (cf. [13], [8]). These normal forms can be computed more efficiently modulo the determinant of a particular matrix (cf. [7], [9]).

So faster determinant algorithms contribute to faster algorithms for solving several kinds of problems.

Perhaps the two most commonly used algorithms for determinant computations are fraction–free Gaussian elimination and the Chinese remaindering method. Using standard arithmetic, the fraction–free method has complexity $O\left(n^5(\log n + \log ||M||)^2\right)$, where n is the dimension of the matrix M and $||M||$ is a bound on the entries in M. The Chinese remaindering method has complexity $O\left(n(\log n + \log ||M||)(n^3 + n^2 \log ||M||)\right)$, or equivalently, $O\left(n^4(\log n + \log ||M||) + n^3 \log^2 ||M||\right)$; the cost of reducing M modulo a prime is included (cf. [1]).

In [5] there is an algorithm to compute the sign of the determinant which works by approximating the determinant safely. This algorithm is designed to use limited precision arithmetic and it has arithmetic complexity $O\left(n^3(\log n + \log ||M||)\right)$ on numbers of $3n/2 + 2\log ||M||$ bits.

In [3] a modified Chinese remaindering method is used to determine the sign of the determinant. Its complexity is the same as for the Chinese remaindering method and experiments show it to be slower than our algorithm on larger matrices.

In this paper we propose an algorithm that also uses Chinese remaindering. However, instead of computing the whole determinant d using Chinese remaindering we first compute a big factor D of d using Hensel lifting (cf. [14]), and then use Chinese remaindering only for computing d/D. The advantage is that we need far fewer primes in this way, thus improving the performance of the algorithm. Our algorithm will perform well for most matrices; however, in rare cases the factor D can be small and then the full cost of the Chinese remaindering method must be borne.

2 The Algorithm

The new algorithm stemmed from the desire to use Hensel lifting techniques which have already proved to yield excellent results in various other types of computation, and most notably in the factorization of polynomials. The most direct approach appears to be to lift an LU-decomposition but this does not improve upon the complexity of the Chinese remaindering method. Instead we use a slightly indirect approach inspired by the observation that one can solve a linear system using Hensel lifting, and that for a given matrix M the solution to the linear system $Mx = b$, for b random, typically has the absolute value of the determinant of M as least common denominator (abbrev. LCD) except perhaps for some small prime factors.

This idea has already been used in [14]. There the linear system $(A - \lambda E)x(\lambda) = u$, for matrices A and E and a vector u, is solved using Hensel lifting. The LCD of $x(\lambda)$ then typically equals $\det(A - \lambda E)$. When E is the identity matrix this gives the characteristic polynomial of A and substitution of $\lambda = 0$ gives $\det(A)$.

Although the LCD of a solution to a linear system $Mx = b$ typically equals $|\det(M)|$, this is not true in general. We identify two causes:

1. For some matrices M we can only ever obtain a proper factor of $\det(M)$ as the LCD of the solution to the linear system $Mx = b$, regardless of the choice of b (e.g. when M is a scalar multiple of the identity matrix).

2. Even for well-behaved matrices (i.e. for which $|\det(M)|$ is the LCD of the solution to the linear system $Mx = b$, for some b) there is always a chance that the particular b chosen leads to a solution whose LCD is only a proper factor of the determinant.

In either case we do obtain a (possibly trivial) factor of the determinant.

Now assume that M is non–singular. Those matrices for which it is never possible to obtain (the absolute value of) the determinant as LCD are characterized as those having a non-trivial Smith Normal Form (SNF). The SNF of M is the matrix S defined as follows: $UMV = S$, where U and V are unimodular and S is diagonal with each entry on the diagonal positive and dividing all those which follow it. These diagonal elements $s_1, \ldots, s_n$ are called the invariant factors of M. The SNF is said to be trivial if and only if $s_{n-1} = 1$.

Clearly $S^{-1} = T/s_n$, where T is the integer, diagonal matrix with ith diagonal entry s_n/s_i. It is easy to see that the solution to $Mx = b$ is given by $x = (VTUb)/s_n$. Whence it follows that the LCD of the solution divides s_n. Since $|\det(M)| = \det(S) = s_1 s_2 \cdots s_n$, we see that we can get $|\det(M)|$ as the LCD if and only if $s_1 = s_2 = \cdots = s_{n-1} = 1$, i.e. the SNF of M is trivial.

It is a well-known phenomenon that matrices with entries randomly chosen in a given range almost always have either trivial SNF, or besides s_n just a few small non–trivial invariant factors. From this it follows that for most matrices $\det(M)/s_n$ is indeed small.

We now study the relationship betwen s_n and the LCD of the solution x when the vector b is chosen randomly. Since V, T, U and b have all integer entries, we see from $x = (VTUb)/s_n$ that the LCD w of x is not equal to s_n if and only if all entries of $VTUb$ and s_n have a common factor. In particular, for N a positive factor of s_n we have that $N|(s_n/w)$, i.e. w is deficient by a factor of at least N relative to s_n, if and only if $VTUb = 0$ modulo N. We shall show that the chance of w being deficient by a large factor is small. Our result is encapsulated in the following two theorems.

Theorem 1 *Let T be an $n \times n$ diagonal matrix with diagonal entries $t_1, t_2, \ldots, t_n \in \mathbb{Z}$ and let U and V be $n \times n$ unimodular matrices over $\mathbb{Z}$. Let N be a positive integer. Then*

$$\#\left\{b \in \{0, 1, \ldots, N-1\}^n \mid VTUb \equiv 0 \pmod{N}\right\} = \prod_{i=1}^{n} \gcd(t_i, N).$$

Proof: Let I denote the interval $\{0, 1, \ldots, N-1\}$. Since V is unimodular we have for $b \in I^n$ that

$$VTUb \equiv 0 \pmod{N} \Leftrightarrow TUb \equiv 0 \pmod{N}.$$

Since U is unimodular the map $b \mapsto Ub \pmod{N}$ defines a bijection on the set I^n. It follows that the set in the theorem has the same number of elements as

$$\{b \in I^n \mid Tb \equiv 0 \pmod{N}\}.$$

Writing $b = (b_1, \ldots, b_n)$ and considering each component separately we have $Tb \equiv 0 \pmod{N}$ iff each b_i is a multiple of $N/\gcd(t_i, N)$. The theorem now follows. •

Corollary 1 *Let T, U, V and N be as in Theorem 1. Let I be the interval $\{m, m+1, \ldots, m+r-1\} \subset \mathbb{Z}$. Then*

$$\#\{b \in I^n \mid VTUb \equiv 0 \pmod{N}\} \leq \left(\left\lceil \frac{r}{N} \right\rceil\right)^n \prod_{i=1}^{n} \gcd(t_i, N),$$

with equality when $N|r$.

Proof: For $b \in I^n$ let $c \in \{0, 1, \ldots, N-1\}^n$ be such that $c \equiv b \pmod{N}$. Then $VTUb \equiv 0 \pmod{N} \Leftrightarrow VTUc \equiv 0 \pmod{N}$. The corollary follows. •

Now let N be a positive factor of s_n and let I be the interval $\{m, m+1, \ldots, m+r-1\} \subset \mathbb{Z}$. From Corollary 1 we see that when we choose a random $b \in I^n$, the probability that $N|(s_n/w)$, where w is the LCD of the solution to $Mx = b$, is at most

$$\left(\frac{\lceil r/N \rceil}{r}\right)^n \prod_{i=1}^{n} \gcd(s_n/s_i, N), \qquad (1)$$

with equality when $N|r$. When $N|r$, (1) reduces to

$$\prod_{i=1}^{n} \frac{\gcd(s_n/s_i, N)}{N},$$

and since the nth factor in this product is $1/N$, we see that the probability of missing a large factor N is very small in this case.

When r is much smaller than N, then formula (1) gives only a crude bound because of the ceiling function involved, but when r is much larger than N the bound is quite accurate. So to make good use of (1) for every factor N of s_n we must take an interval I of width r much larger than s_n. Although we don't know s_n, we do know that $s_n \leq |\det(M)|$ and $|\det(M)|$ can be bounded by Hadamard's bound H (cf. [10]), so we can take $r \in O(H)$. Now, H will be quite big for large matrices. In fact we have $\log H \in O(n(\log n + \log ||M||))$, so a typical $b \in I^n$ has size $O(n^2(\log n + \log ||M||))$ bits. From a theoretical point of view this would be no problem, since the solution of $Mx = b$ (even for a small b) also takes $O(n^2(\log n + \log ||M||))$ bits, and the asymptotic complexity for solving $Mx = b$ is the same for large and small b (cf. [12]). Here we neglect the cost of generating $O(n^2(\log n + \log ||M||))$ random bits — this is reasonable under the assumption that a random bit has unit cost. In practice however, using such a large interval would slow down the computations tremendously; we want an interval of more moderate size. The following theorem shows the way.

Theorem 2 *Let T be an $n \times n$ diagonal matrix with diagonal entries $t_1, t_2, \ldots, t_n \in \mathbb{Z}$ such that $t_n | t_{n-1} | \cdots | t_1$. Let U and V be $n \times n$ unimodular matrices over $\mathbb{Z}$. Let p be a prime and k a positive integer. Let I be the interval $\{m, m+1, \ldots, m+r-1\} \subset \mathbb{Z}$. Then*

$$\#\{b \in I^n \mid VTUb \equiv 0 \pmod{p^k}\} \leq \prod_{i=1}^{n} \left(\left\lceil \frac{r \gcd(t_i, p^k)}{p^k} \right\rceil\right),$$

with equality when $p^k | r$.

Proof: Since V is unimodular we have for $b \in I^n$ that

$$VTUb \equiv 0 \pmod{p^k} \quad \Leftrightarrow \quad TUb \equiv 0 \pmod{p^k}.$$

Let $\phi: \mathbb{Z} \to \mathbb{Z}/p^k$ be the canonical projection map. Then for $b \in I^n$:

$$TUb \equiv 0 \pmod{p^k} \quad \Leftrightarrow \quad \phi(TU)\phi(b) = 0.$$

We will first investigate the kernel of $\phi(TU)$.

Write $t_i = p^{e_i} q_i$, where $p \not| q_i$ (when $t_i = 0$ take $e_i = \infty$ and $q_i = 1$). Let $K \in \mathbb{Z}^{n \times n}$ be the diagonal matrix with ith diagonal entry p^{k_i}, where $k_i = \max(k - e_i, 0)$. Notice that $k_1 \leq k_2 \leq \cdots \leq k_n$. Then for $c \in (\mathbb{Z}/p^k)^n$ we have

$$\phi(T)c = 0 \quad \Leftrightarrow \quad c \in \mathrm{CS}(\phi(K)),$$

where $\mathrm{CS}(\phi(K))$ (*column span* of $\phi(K)$) denotes the $\mathbb{Z}/p^k$–module of all $\mathbb{Z}/p^k$–linear combinations of the columns of $\phi(K)$. So for $c \in (\mathbb{Z}/p^k)^n$ we have

$$\phi(TU)c = 0 \quad \Leftrightarrow \quad c \in \mathrm{CS}(\phi(U^{-1}K)).$$

Since an element from $\mathbb{Z}/p^k$ which is not a multiple of $\phi(p)$ is a unit in $\mathbb{Z}/p^k$, and U^{-1} is unimodular, we can perform Gaussian elimination on the columns of $\phi(U^{-1})$. This yields matrices $P, R, L \in (\mathbb{Z}/p^k)^{n \times n}$ where P is a permutation matrix, R is an invertible upper triangular matrix and L is a unit lower triangular matrix such that

$$P\phi(U^{-1})R = L.$$

When $R = (r_{ij})$, let $\tilde{R} = (\tilde{r}_{ij})$ such that $\tilde{r}_{ij} = \phi(p^{k_j - k_i}) r_{ij}$. Notice that when $r_{ij} \neq 0$, then $j \geq i$ and so $k_j \geq k_i$, implying $p^{k_j - k_i} \in \mathbb{Z}$. Then $R\phi(K) = \phi(K)\tilde{R}$ and $\tilde{R}$ is still an invertible matrix. Putting things together we get:

$$\begin{aligned}
VTUb \equiv 0 \pmod{p^k} \quad &\Leftrightarrow \quad TUb \equiv 0 \pmod{p^k} \\
&\Leftrightarrow \quad \phi(TU)\phi(b) = 0 \\
&\Leftrightarrow \quad \phi(b) \in \mathrm{CS}(\phi(U^{-1}K)) \\
&\Leftrightarrow \quad P\phi(b) \in \mathrm{CS}(P\phi(U^{-1}K)) \\
&\Leftrightarrow \quad P\phi(b) \in \mathrm{CS}(P\phi(U^{-1}K)\tilde{R}) \\
&\Leftrightarrow \quad P\phi(b) \in \mathrm{CS}(P\phi(U^{-1})R\phi(K)) \\
&\Leftrightarrow \quad P\phi(b) \in \mathrm{CS}(L\phi(K)).
\end{aligned}$$

Since P is a permutation matrix we can ignore P and count

$$\#\{b \in I^n \mid \phi(b) \in \mathrm{CS}(L\phi(K))\}$$

Let $L = (l_{ij})$, then

$$L\phi(K) = \begin{bmatrix} \phi(p^{k_1}) & 0 & 0 & \cdots \\ \phi(p^{k_1})l_{21} & \phi(p^{k_2}) & 0 & \cdots \\ \phi(p^{k_1})l_{31} & \phi(p^{k_2})l_{32} & \phi(p^{k_3}) & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}.$$

For $b = (b_1, \ldots, b_n) \in I^n$ such that $\phi(b) \in \mathrm{CS}(L\phi(K))$ let $x = (x_1 \ldots, x_n) \in (\mathbb{Z}/p^k)^n$ such that $\phi(b) = L\phi(K)x$. We will now bound the number of b such that $\phi(b) \in \mathrm{CS}(L\phi(K))$. Since $p^{k_1} | b_1$, there are at most $\lceil r/p^{k_1} \rceil$ possibilities for b_1, with equality when $p^{k_1} | r$. Fixing b_1, does not determine x_1 uniquely but it does determine x_1 times the first column of $L\phi(K)$ uniquely. Since $p^{k_2} | b_2 - x_1\phi(p^{k_1})l_{21}$, there are at most $\lceil r/p^{k_2} \rceil$ possibilities for b_2, with equality when $p^{k_2} | r$. Again, fixing b_2 does not determine x_2 uniquely but it does determine x_2 times the second column of $L\phi(K)$ uniquely. Proceeding in this way and noticing that $p^{k_i} = p^k / \gcd(t_i, p^k)$ proves the theorem. •

Remark: Counting the b with $\phi(b) \in \mathrm{CS}(L\phi(K))$ in the proof of Theorem 2 could be done more concisely by noticing that the column Howell normal form of $L\phi(K)$ (cf. [11], [16]) has the same diagonal entries as $L\phi(K)$.

Now let p be a prime and k a positive integer such that $p^k | s_n$. Let I be the interval $\{m, m+1, \ldots, m+r-1\}$. From Theorem 2 we see that when we choose a random $b \in I^n$, the probability that $p^k | (s_n/w)$, where w is the LCD of the solution to $Mx = b$, is at most

$$r^{-n} \prod_{i=1}^{n} \left\lceil \frac{r \gcd(s_n/s_i, p^k)}{p^k} \right\rceil. \tag{2}$$

Now (2) is never greater than

$$\frac{1}{r} \lceil r/p^k \rceil, \tag{3}$$

and when M has trivial SNF then (2) is equal to (3). When $r \leq p^k$, then (3) is equal to $1/r$ and when r is much larger than p^k, then (3) is near $1/p^k$. In fact, when $r < p^k$, the probability will in practice be much smaller than suggested by formula (2). This is because Theorem 2 gives an upper bound which is loose except in special cases (e.g. when $p^k | r$).

As an example let us assume that M has trivial SNF. Looking at the proof of Theorem 2 we see that then $k_1 = \cdots = k_{n-1} = 0$. Then

$$\mathrm{CS}(L\phi(K)) = \mathrm{CS} \begin{bmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ l_{n1} & \cdots & l_{n-1\,1} & \phi(p^k) \end{bmatrix}.$$

So for $(b_1, \ldots, b_n) \in I^n$ we have

$$\begin{aligned}
&\phi(b) \in \mathrm{CS}(L\phi(K)) \\
&\quad \Leftrightarrow \quad b_n - b_1 l_{n1} - \cdots - b_{n-1} l_{n-1\,1} \equiv 0 \pmod{p^k},
\end{aligned}$$

i.e. for $b_1, \ldots, b_{n-1} \in I$, $b_1 l_{n1} + \cdots + b_{n-1} l_{n-1\,1}$ should be in the interval I up to a multiple of p^k. When the width of I is small compared to p^k, this is unlikely to happen.

In the following example we compare the bound Q given by Theorem 2 to the average of the actual number of b's such that $\phi(b) \in \mathrm{CS}(L\phi(K))$, for 10 randomly chosen L.

Example We take $n = 5$ and $I = \{0, 1, 2, 3\}$. For $k_1 = \cdots = k_4 = 0, k_5 = k$ we get $Q = 256$ and the following values for the average:

$p \setminus k$	3	5	7	10	13
2	130.0	28.9	9.1	2.4	1.0
3	39.8	5.4	1.6	1.0	1.0

For $k_1 = \cdots = k_3 = 0, k_4 = 2, k_5 = k$ we get $Q = 64$ and the following values for the average:

$p \setminus k$	3	5	7	10	13
2	34.0	9.6	3.5	1.2	1.0
3	5.3	1.2	1.1	1.0	1.0

In summary, if we choose b uniformly in I^n for some interval I and solve $Mx = b$, then w, the LCD of the solution x, is close to s_n; i.e. s_n/w is small with high probability. By choosing several random b's and taking the LCD for all the solutions, D, we may get even closer to s_n. Next we determine $\det(M)/D$ by using the well–known Chinese remaindering technique. From this we get $\det(M)$ immediately.

We now present our algorithm:

ALGORITHM: DETERMINANT
Input: $n \times n$ matrix of integers M.
Output: $\det(M)$.

(1) Using Hensel lifting compute solutions to $Mx = b$ for several random vectors $b \in I^n$ for a suitable interval $I = \{m, m+1, \ldots, m+r-1\}$. We use rational recovery to obtain the solutions.

(2) Put D equal to the LCD of all the solutions found in step (1). We know that D is a divisor of the determinant.

(3) Compute, H, a bound for the absolute value of $\det(M)$. We can use the minimum of Hadamard's bound for rows and columns.

(4) Now compute $\det(M)/D$ using Chinese remaindering techniques; we can start with the values and primes used in step (1) by the Hensel lifting. We know that the absolute value of $\det(M)/D$ cannot exceed H/D, so we can use $2H/D$ as the bound for the Chinese remaindering modulus.

(5) Compute $\det(M)$ as $D \times (\det(M)/D)$.

Our algorithm can be viewed as a variant of the standard Chinese remaindering technique. The important point is that with high probability the bound H/D is far smaller than H itself (see Section 3), so that typically "most of the work" is done in steps (1) and (2) while the verification/rectification in steps (3) and (4) has much lower cost than computing solely using Chinese remaindering.

We point out that, except for $O(n)$ gcd computations, the only multiplications and divisions involving big integers have a small integer as the other parameter. So we have no need of "fast" arithmetic for matrices or big integers.

3 How Tight is Hadamard's Bound?

The only bound for the absolute value of the determinant of which we are aware is that due to Hadamard: $H = \prod_{i=1}^n |v_i|$ where v_i is the vector whose coordinates are given by row i of the matrix, and $|v|$ represents the euclidean length of the vector V. We can equally well use the columns of the matrix instead of its rows. Indeed, the smaller of these two possibilities is still a valid upper bound. Empirically we have observed that the row and column bounds rarely differ by a factor greater than 2 for matrices 10×10 or larger with independent uniformly distributed integer entries in the range -100 to 100. Nonetheless there do exist cases where the two values differ significantly: e.g. consider an identity matrix with all entries in one column replaced by a large integer K. Since both versions of the bound can be computed cheaply it is worth doing so, and then using the smaller.

For our analysis we shall assume that just one of the versions of Hadamard's bound is used, say the "row" version. Our aim is to compute the expected value of $\log(H/|d|)$ where d is the determinant and H is the row Hadamard bound; this value is necessarily positive and the closer it is to 0 the tighter Hadamard's bound is. The first point to establish is our assumed distribution on the matrices. We shall avoid all matrices containing a row of zeroes because these give $d = H = 0$. Now the value of $H/|d|$ is unaffected if we scale any row of the matrix by a non-zero value; thus we may assume that each row has euclidean length 1, and consequently that $H = 1$. Such matrices can be viewed as specifying n points on the surface of the unit n-sphere (where n is the size of the matrix). Our assumed distribution is that these points are independently and uniformly distributed on the surface of the sphere. Thus our task reduces to computing the expected value of $\log |d|$ for $n \times n$ matrices having this distribution, for each integer n.

We shall proceed by finding upper and lower bounds for $\mathbf{E}(\log |d|)$. From Jensen's inequality (see any book on basic probability, e.g. [2]) we see that:

$$\log\left(\mathbf{E}(|d|)\right) \geq \mathbf{E}(\log |d|) \geq -\log\left(\mathbf{E}(1/|d|)\right).$$

We can compute $\mathbf{E}(|d|)$ directly, but have to use a trick to get the lower bound (slightly different from that above). We deal with $\mathbf{E}(|d|)$ first, and shall write d_n to mean this value for matrices $n \times n$.

Now, viewing the absolute value of the determinant as the volume of a parallelepiped we see that it can be computed as a product of sines of certain angles. Let us call the n points $v_1, \ldots, v_n$. We define $n-1$ angles as follows: let ϕ_1 be the angle between v_1 and v_2, let ϕ_2 be the angle between $\mathrm{span}(v_1, v_2)$ and v_3, and in general let ϕ_k be the angle between $\mathrm{span}(v_1, \ldots, v_k)$ and v_{k+1} for k up to $n-1$. It is well-known that

$$|d| = \sin\phi_1 \, \sin\phi_2 \, \cdots \, \sin\phi_{n-1}.$$

Let us focus our attention on ϕ_k for the moment. We begin by finding the density function for ϕ_k, and by symmetry we can suppose that $\phi_k \in [0, \pi/2]$. Let $W_k = \mathrm{span}(v_1, \ldots, v_k)$, a subspace of dimension k. The vector v_{k+1} can be written uniquely as the sum of two components: $v_\parallel$ lying in W_k and $v_\perp$ perpendicular to W_k. Furthermore $|v_\parallel| = \cos\phi_k$ and $|v_\perp| = \sin\phi_k$ by definition. Thus all vectors forming an angle ϕ_k with W_k are generated as the sum of a vector lying on the k-sphere of radius $\cos\phi_k$ inside W_k with another vector lying on the $(n-k)$-sphere of radius $\sin\phi_k$ inside $W_k^\perp$. Whence we obtain the density of ϕ_k as a function of θ proportional to $(\cos\theta)^{k-1}(\sin\theta)^{n-k-1}$.

The formula for the expected value of $\sin\phi_k$ is now immediate:

$$\int_0^{\frac{\pi}{2}} (\cos\theta)^{k-1} (\sin\theta)^{n-k} \, d\theta \Bigg/ \int_0^{\frac{\pi}{2}} (\cos\theta)^{k-1} (\sin\theta)^{n-k-1} \, d\theta$$

Thanks to the symmetry of the sphere the distribution of ϕ_k is independent of the values of the other ϕ_j, and so the expected value of the product $\sin\phi_1 \sin\phi_2 \cdots \sin\phi_{n-1}$ is just the product of the individual expected values.

Before continuing, it will be handy to introduce some new notation. Let us write $I_{r,s}$ for the integral $\int_0^{\pi/2}(\sin\theta)^r(\cos\theta)^s\,d\theta$. So using this notation we have $\mathbf{E}(\sin\phi_k) = I_{n-k,k-1}/I_{n-k-1,k-1}$, and consequently

$$d_n = \prod_{r=1}^{n-1}\frac{I_{n-r,r-1}}{I_{n-r-1,r-1}} = \frac{I_{n-1,0}}{I_{n-2,0}}\cdot\frac{I_{n-2,1}}{I_{n-3,1}}\cdots\frac{I_{1,n-2}}{I_{0,n-2}}$$

It can easily be shown that $(r+s)I_{r,s} = (r-1)I_{r-2,s}$ whenever $r \geq 2$ using integration by parts. Whence $I_{r,0}$ can be expressed as a rational multiple of either $I_{0,0} = \pi/2$ or $I_{1,0} = 1$. Another elementary result is that $I_{r,1} = 1/(r+1)$, and this eventually allows the following simplifications:

$$\frac{d_{n+1}}{d_{n-1}} = \left(\frac{n-1}{n}\right)^n \frac{I_{n-2,0}}{I_{n-1,0}}$$

It is not hard to show that the n-th power has asymptotic form $e^{-1}(1 - 1/2n + O(1/n^2))$, and employing Stirling's asymptotic formula for the factorial we find that the other factor on the right hand side is $1 + 1/2n + O(1/n^2)$. Altogether then, we see that the right hand side is asymptotically $e^{-1}(1+O(1/n^2))$, from which it easily follows that $d_n \in \Theta(e^{-n/2})$ and in fact we get this upper bound:

$$\mathbf{E}(\log|d|) \leq \log d_n = -n/2 + O(1).$$

Now, we address the lower bound, and start with a "trick": we note that $\log 1/|d|$ can be written as:

$$\log 1/|d| = \log 1/(\sin\phi_1\cdots\sin\phi_{n-2}) + \log 1/\sin\phi_{n-1}. \quad (4)$$

Recall that the expectations of the two terms on the right can be computed separately and then summed. The expectation of the second term can easily be bounded:

$$\mathbf{E}(\log(1/\sin\phi_{n-1})) = \int_0^{\frac{\pi}{2}} -\log\sin\theta\,(\cos\theta)^{n-2}d\theta \Big/ I_{n-2,0}.$$

Using Stirling's approximation we see that the denominator $I_{n-2,0} = \sqrt{\pi/2n}(1+O(1/n)) \in \Theta(1/\sqrt{n})$, while the numerator remains bounded for all n:

$$0 \leq \int_0^{\frac{\pi}{2}} -\log\sin\theta\,(\cos\theta)^{n-2}d\theta \leq \int_0^{\frac{\pi}{2}} -\log\sin\theta\,d\theta \leq \pi/2$$

where the last inequality results from replacing $\sin\theta$ by the lower bound $\theta/(\pi/2)$ valid over the range of integration.

The expectation of the first term on the right of (4) can be bounded above using Jensen's inequality: the bound is $\log d'_n$ where $d'_n = \mathbf{E}(1/(\sin\phi_1\,\sin\phi_2\cdots\sin\phi_{n-2}))$. Reasoning much as we did for d_n, we compute each $\mathbf{E}(1/\sin\phi_k)$ separately as a ratio of two integrals of the form $I_{r,s}$, then we mimic the ensuing simplifications to obtain $d'_{n-1}/d'_{n+1} = e^{-1}\left(1+O(1/n^2)\right)$ and subsequently $\log d'_n = n/2 + O(1)$. Substituting these results into (4) gives us:

$$\mathbf{E}(\log 1/|d|) \leq n/2 + \Theta(\sqrt{n})$$

So putting everything together we have proved the following theorem:

Theorem 3 *Let $v_1,\ldots,v_n \in \mathbb{Z}^{1\times n}$ be non-zero random vectors such that $v_i/|v_i|$ is uniformly distributed on the surface of the unit n-sphere. Let M be the matrix whose rows are $v_1,\ldots,v_n$, and H be the "row" version of Hadamard's bound for M. Then*

$$n/2 + \Theta(\sqrt{n}) \geq \mathbf{E}\left(\log(H/|\det(M)|)\right) \geq n/2 + O(1).$$

Informally, this means that Hadamard's bound is too large by roughly a factor of $e^{n/2}$, a quantity which depends only on the matrix dimension and not on the size of the entries. Experimentally we have computed determinants of matrices up to 800×800 with entries chosen independently and uniformly in an interval (for convenience of implementation); while this distribution is significantly different from the uniform spherical one, we observed nonetheless that the ratio of Hadamard's bound to the determinant always lay close to $e^{n/2}$.

4 Complexity issues

In this section we will study the complexity of our algorithm. We consider both the worst case and the average case.

It is clear that the running time of the algorithm is dominated by steps (1) and (4). We start with step (1). There we solve several linear systems $Mx = b$ with b varying. The complexity of solving such a linear system is $O\left(n^3\log^2 n\right)$ for bounded M and b (cf. [6]) and $O\left(n^3(\log n + \log||M||)^2 + n\log^2||b||\right)$ for arbitrary M and b (cf. [12]), where $||M||$ is a bound on the entries in M and $||b||$ is a bound on the entries in b. Let A be the number of systems we solve, then the total cost of step (1) is

$$O\left(A\left(n^3(\log n + \log||M||)^2 + n\log^2||b||\right)\right). \quad (5)$$

Now we estimate the cost of step (4). In it we perform $O(\log(H/D))$ Chinese remaindering steps giving a complexity of

$$O\left(\log(H/D)\left(n^2\log||M|| + n^3\right)\right). \quad (6)$$

Writing

$$\frac{H}{D} = \frac{H}{|\det(M)|}\cdot\frac{|\det(M)|}{s_n}\cdot\frac{s_n}{D},$$

we see that $\log(H/D)$ is equal to

$$\log(H/|\det(M)|) + \log(|\det(M)|/s_n) + \log(s_n/D). \quad (7)$$

Notice that the factor $H/|\det(M)|$ is commonly called the *orthogonality defect* (e.g. in [5]). We will now investigate all three terms in (7) separately.

We have a number of choices for our algorithm. We can choose either of A and I to be small or big.

Lemma 1 *If we choose $A = 2$ and $r = \lceil(n+1)H\rceil$, for example with $I = \{1,2,\ldots,r\}$. Then step (1) has complexity $O\left(n^3(\log n + \log||M||)^2\right)$. Moreover,*

$$\mathbf{E}(\log(s_n/D)) \in O(1).$$

Proof: Assuming again that a random bit can be generated at unit cost the $2n$ random numbers in I can be chosen in time $O(n\log r) \in O\left(n^2(\log n + \log||M||)\right)$. The complexity result now follows from substitution in (5). Let N be a positive factor of s_n and put $q = \lfloor r/N\rfloor$. So $q \geq n$ and formula (1) is bounded above by

$$\left(\frac{q+1}{q}\right)^n \prod_{i=1}^n \frac{\gcd(s_n/s_i, N)}{N} \leq \frac{\exp(1)}{N}.$$

So, since $A = 2$, the probability that $s_n/D = N$ is bounded by $\exp(2)/N^2$. From this it follows that

$$\mathbf{E}(\log(s_n/D)) \quad\leq\quad \sum_{N|s_n}\log N\cdot\frac{\exp(2)}{N^2}$$

$$
\begin{aligned}
&\le \exp(2)\sum_{i=1}^{s_n}\frac{\log N}{N^2} \\
&\in O(1).
\end{aligned}
$$

•

Lemma 2 *If we choose $r \ge 2$ but small (e.g. $I = \{0, 1\}$) and $A = \lceil 2\log_2(n(\log n + \log ||M||))\rceil$. Then step (1) has complexity $O\left(n^3(\log n + \log ||M||)^3\right)$. Moreover,*

$$\mathbf{E}(\log(s_n/D)) \in O(1).$$

Proof: The complexity result follows from substitution in equation (5).

Let $L = \{\text{primes } p \ne 2, 3 \text{ dividing } s_n\} \cup \{4, 9\}$. Let $\mathbf{P}[x]$ denote the probability that x is true. From formula (3) it follows that for $q \in L$:

$$\mathbf{P}[q|(s_n/D)] \le (1/2)^A.$$

Now, it is easily verified that $\#L \le \kappa \log s_n$ for $\kappa = 1/\log 2$, so $\mathbf{P}[\exists q \in L\colon q|(s_n/D)] \le \kappa \log(s_n)(1/2)^A$. Since

$$(s_n/D)\not| 6 \quad \Leftrightarrow \quad \exists q \in L\colon q|(s_n/D),$$

we have $\mathbf{P}[(s_n/D)|6] \ge 1 - \kappa\log(s_n)(1/2)^A$. So

$$
\begin{aligned}
\sum_{N|s_n, N\not| 6} \mathbf{P}[s_n/D = N] &= 1 - \mathbf{P}[(s_n/D)|6] \\
&\le \kappa \log(s_n)(1/2)^A.
\end{aligned}
$$

Thus

$$
\begin{aligned}
\mathbf{E}(\log(s_n/D)) &\le \log 2 + \log 3 + \log 6 + \\
&\qquad \sum_{N|s_n, N\not| 6} \log(N)\cdot \mathbf{P}[s_n/D = N] \\
&\le O(1) + \log(s_n) \sum_{N|s_n, N\not| 6} \mathbf{P}[s_n/D = N] \\
&\le O(1) + \kappa \log^2(s_n)(1/2)^A.
\end{aligned}
$$

Since s_n is bounded by Hadamard's bound, the lemma follows. •

Now we tackle the first two terms in (7). From theorem 3 we know that when the rows v_i of M are chosen such that $v_i/|v_i|$ are uniformly distributed on the surface of the unit n-sphere, then $\mathbf{E}\left(\log(H/|\det(M)|)\right) = n/2 + o(n)$. Concerning the term $\log(|\det(M)|/s_n)$, we only know that, in practice for a random matrix M (with entries randomly chosen from a fixed range), $\log(|\det(M)|/s_n)$ is "always" much smaller than n. So, based on these two observations, we will make the following assumption on the distribution when choosing a random matrix M:

Assumption 1 *When choosing a random matrix M, we have:*

$$\mathbf{E}(\log(H/|\det(M)|) \in O(n);$$
$$\mathbf{E}(\log(|\det(M)|/s_n)) \in O(n).$$

It is clear that when the factor D, computed in our algorithm, is small, e.g. $D = 1$, we have to perform the full Chinese remaindering algorithm in step (4). The complexity is in that case

$$O\left(n^4(\log n + \log ||M||) + n^3 \log^2 ||M||\right),$$

which is the same as for the full Chinese remaindering algorithm.

However, in general the expected complexity will be much better. When we choose a random matrix M, we will call the expected value of the expected complexity the *average expected complexity*. We now get the following theorem:

Theorem 4 *Under Assumption 1 there is an algorithm to compute the determinant of an $n \times n$ integer matrix M with average expected bit-complexity*

$$O\left(n^4 + n^3(\log n + \log ||M||)^2\right),$$

where $||M||$ is a bound on the entries in M.

Proof: This follows from Lemma 1 and (6). •

5 Refinements and Implementation Details

In this section we will describe a practical version of our algorithm. Here our choice of interval and number of linear systems to solve is guided more by experience than the theory above.

The interval we use is $I = \{-99, -98 \ldots, +99\}$, and we solve just a single random linear system. Lemma 1 does not apply because the interval is too small, and Lemma 2 tells us to solve many random linear systems. Since Lemma 2 is very pessimistic in general, as observed in Section 2, our choice works well in practice.

The worst case for our algorithm is when D is small (compared to the bound H); this will happen with certainty when $\det(M)$ itself is small. Now in most cases, even when $\det(M)$ is small, the determinant of any $(n-1)\times(n-1)$ minor of M will not be small. This observation leads to the following possible modification of our algorithm.

We partition the matrix into an $(n-1)\times(n-1)$ submatrix N, two vectors v and w, and a single entry y:

$$M = \begin{bmatrix} N & v \\ w & y \end{bmatrix}.$$

Solve the system $xN = w$ for x; now form the inner product of x with the first $n-1$ rows of M and subtract the result from the last row of M (computationally we need only form the inner product of x and v). This yields

$$\det(M) = \det\left(\begin{bmatrix} N & v \\ 0 & y - xv \end{bmatrix}\right) = (y - xv)\det(N). \quad (8)$$

To compute $\det(N)$ we can use our original algorithm but skipping step (1) since we already have a solution x. For simplicity, we described the partition as splitting off the last row and column; analogously we can split off any row and column of our choice instead. If the system $xN = w$ has no solution then there is a linear relation on the rows of N which can be used instead of the solution x to annihilate all but one element in a row of M.

In fact, this variant can be used for any matrix, not just those yielding a small D in step (2) of the original algorithm.

matrix type	size	Maple	NTL	Pion	CRA	Hensel	$\log_2(d)$	$\log_2(d/D)$	$\log_2(H/D)$
small	50×50	3.8	0.2	0.1	0.2	0.1	400	1	38
	100×100	100	2.6	0.9	1.1	0.9	845	1	73
	200×200		42	15	12	7.3	1800	1	147
	400×400		700	310	230	69	3800	1	290
	800×800			5700	5200	780	8000	1	580
large	50×50	96	1.5	n/a	1.6	0.8	3400	2	40
	100×100		20	n/a	9.1	6.0	6800	1	74
	200×200		300	n/a	96	51	13750	1	145
	400×400		4900	n/a	2000	420	27700	0	290
unim	50×50	1.3	0.3	0.1	0.3	0.4	0	0	640
	100×100	12	4.9	1.6	2.1	3.0	0	0	1560
	200×200	160	79	27	23	29	0	0	3000
	400×400		1300	620	410	360	0	0	5000
snf	50×50	2.5	0.4	0.2	0.5	0.5	181	178	810
	100×100	72	6.2	1.9	2.6	3.7	366	363	1900
	200×200	2800	95	33	28	36	736	733	3700
	400×400		1600	730	530	460	1476	1473	6500

Table 1: experimental timings

For matrices which would produce a large D in step (2), this variant is essentially equivalent to the original, whereas for many matrices which would produce a small D the variant behaves much better. This latter is due to the empirical fact that an $(n-1) \times (n-1)$ minor chosen at random from a matrix with small s_n (compared to the determinant) typically has an "almost-trivial" SNF. The variant still will perform badly on some matrices e.g. totally unimodular ones whose minors all have determinants -1, 0 or 1.

To compute the solution of a system $Mx = b$ using Hensel lifting we choose a random prime p for which M is non-singular modulo p. If M is singular modulo p, we must choose another prime, but we also have the information that $\det(M) \equiv 0 \pmod p$ which implies $\det(M)/D \equiv 0 \pmod p$. This information can then be used in step (4), so even "bad" primes are useful to us.

If $\det(M) = 0$ (without modulus) then we will always find that $\det(M) \equiv 0 \pmod p$. When this happens we can conclude that $\det(M) = 0$ when the product of the primes tried exceeds H. In this case we have performed the full Chinese remaindering algorithm.

Another refinement is a general technique for computing modulo machine primes: if w is the number of bits of a machine word, and p is a half–word prime, then we can add k products of elements of $\{0, \dots, p-1\}$ without risking an overflow, where $k = (2^w - 1)/(p-1)^2$. This means that, during the computation of $\det(M)$ modulo p, we can eliminate k successive pivots without reducing the entries modulo p below the pivoting row (the row of the pivot has to be reduced however). If we select p so that k is larger than the matrix size then we only need a quadratic number of reductions modulo p rather than cubic. For a fixed prime, this also lets the algorithm decide when to reduce the remaining matrix modulo p.

6 Comparison with Existing Implementations

We have implemented our new algorithm using the programming language C, and using the GNUmp library (version 2.0.2) for operations on big integers: it will be included in the next release of CoCoA [4].

Table 1 presents a comparison of our implementation with a number of others: the column headed "Maple" refers to the computer algebra system Maple V release 4, "NTL" refers to the determinant function for integer matrices supplied with the library NTL (version 3.5a, [15]), "Pion" refers to the package[1] produced by Pion *et al.* (cf. [3]), "CRA" corresponds to our implementation of a pure Chinese Remaindering method, and "Hensel" gives times for our new algorithm in the guise of the variant described in Section 5. The platform is a 400MHz Pentium running Linux.

We considered four classes of matrix:

small entries are independently and uniformly distributed between -99 and 99;

large entries are independently and uniformly distributed between -10^{20} and 10^{20};

unim unimodular matrices formed by multiplying a lower-triangular and an upper-triangular matrix, each having only ones along the diagonal, and the off-diagonal entries being independently and uniformly distributed between -99 and 99;

snf matrices having non-trivial SNF with $s_1 = 1$ and $s_i = 13$ for $i > 1$, formed as a product LSU where L and U are lower- and upper-triangular matrices (as for our unimodular matrices), and S is the SNF.

The table entries are CPU times in seconds; a blank entry means that the computation was not attempted. All times are averages for five different matrices of the specified type (except for the matrix 800×800 where only one matrix was used). Memory consumption was reasonable: for the 800×800 matrix the maximum requirement was 28Mbytes including the space to hold the input matrix as GMP integers.

The better practical complexity of the new algorithm compared to the Chinese Remaindering technique is immediately evident from this table, particularly for the first two types of matrix. Furthermore, we see that the new method is

[1] This package computes only the sign of the determinant, although not much additional work is needed to compute the full determinant.

already competitive for matrices as small as 50×50. Pion's package for determining the sign of the determinant worked well for the smaller dimension matrices, but not for the larger ones; it was not applicable to matrices of type "large". NTL's implementation is based on Chinese Remaindering — it was the reference system during debugging. By default NTL offers a "probabilistic" determinant, for the tests we specified the option that the computation should be "deterministic".

Experimentally, while preparing the table of times above, we observed that the cost of the rational recovery to find the solution of the linear system was only ever a small fraction of the total time taken to compute the solution. The one exception was a matrix of moderate dimension, 52×52, with large entries (up to 1450 decimal digits) — the matrix came from a problem in chip design. In this case the rational recovery took as long as the Hensel lifting. Such behaviour is to be expected if $\log ||M||$ is large compared to n; in these cases it would be appropriate to find only some of coordinates in the solution vector, and take D to be the LCD of these coordinates.

7 Conclusions and future work

We have described an algorithm that improves the theoretical average expected complexity of determinant computation by a factor of $\log n$. We have also given a practical version of the algorithm that can compete with other algorithms.

The expected complexity depends on the tightness of a bound for the determinant. If a new bound is discovered which is tighter than Hadamard's then our algorithm can benefit directly with a consequent gain for the expected complexity — the expected complexity for Chinese remaindering methods cannot be improved by using better bounds, since they necessarily have to lift up to the determinant.

One can also transform our algorithm to a probabilistic one. In that case one only has to compute $\det(M)/D$ until it stabilizes for sufficiently many primes, depending on the chosen fault tolerance. Since $\det(M)/D$ will be small we need only a moderate number of primes. The average expected complexity is then $O\left(n^3(\log n + \log ||M||)^2\right)$.

We also want to point out that our algorithm is also applicable in the case when M is a matrix with polynomial entries over a field. In that case a bound on the degree of the determinant is $H = \sum_{i=1}^{n} d_i$, where d_i is the maximum degree in the ith row of M. Generally, this bound H is much tighter than Hadamard's bound in the integer case. This means that in the polynomial case, the expected complexity of our algorithm is close to the complexity of Hensel lifting, which is $O\left(n^3(\deg(M))^2\right)$ (cf. [12]).

There are some remaining interesting questions:

- Can we say anything more about Assumption 1? For instance, what are the exact values of $\mathbf{E}(\log(H/|\det(M)|))$ and $\mathbf{E}(\log(|\det(M)|/s_n))$ for any given distribution for M, e.g. when the entries in M are chosen randomly from a fixed range. Especially, any statement about $\mathbf{E}(\log(|\det(M)|/s_n))$ would be very interesting. This question is of the same nature as constructing a "random" matrix with prescribed SNF.
- Can we decrease the number of iterations in Lemma 2 by a moderate increase of r? Is $r \in O(n)$ and $A \in O(\log(\log n + \log ||M||))$ a possible candidate?
- Can we combine our algorithm with numerical methods that will give a better bound on $\det(M)$ than Hadamard's bound?

Acknowledgement We would like to thank Arne Storjohann for some fruitful discussions. We also thank the referees for their useful observations.

References

[1] Bareiss, E. Computational solutions of matrix problems over an integral domain. *J. Inst. Maths Applics 10* (1972), 68–104.

[2] Breiman, L. *Probability.* Addison–Wesley, 1968.

[3] Brönnimann, H., Emiris, I., Pan, V., and Pion, S. Computing exact geometric predicates using modular arithmetic with single precision. In *Symposium on Computational Geometry '97* (1997), pp. 174–182.

[4] Capani, A., Niesi, G., and Robbiano, L. Cocoa: Computations in commutative algebra. `http://cocoa.dima.unige.it/`.

[5] Clarkson, K. Safe and effective determinant evaluation. In *Proc. 33rd Ann. IEEE Symp. Foundations of Comp. Science* (1992), pp. 387–395.

[6] Dixon, J. Exact solution of linear equations using p-adic expansions. *Numer. Math. 40* (1982), 137–141.

[7] Domich, P., Kannan, R., and Trotter, L. Hermite normal form computation using modulo determinant arithmetic. *Math. Oper. Res. 12* (1987), 50–59.

[8] Frumkin, M. Polynomial time algorithms in the theory of linear diophantine equations. In *Proc. first International FCT-conference* (1977), M. Karpinski, Ed., vol. 56 of *LNCS*, Springer, pp. 386–392.

[9] Hafner, J., and McCurley, K. Asymptotically fast triangularization of matrices over rings. *SIAM J. Comput. 20* (1991), 1068–1083.

[10] Horn, R., and Johnson, C. *Matrix Analysis.* Cambridge University Press, 1985.

[11] Howell, J. Spans in the module $(\mathbb{Z}_m)^s$. *Linear and Multilinear Algebra 19* (1986), 67–77.

[12] Mulders, T., and Storjohann, A. Diophantine linear system solving. In *These proceedings* (1999).

[13] Newman, M. *Integral Matrices.* Academic Press, 1972.

[14] Pan, V. Computing the determinant and the charactersitic polynomial of a matrix via solving linear systems of equations. *Inf. Proc. Letters 28* (1988), 71–75.

[15] Shoup, V. Ntl: A library for doing number theory. `http://www.cs.wisc.edu/~shoup/ntl`.

[16] Storjohann, A., and Mulders, T. Fast algorithms for linear algebra modulo n. In *Proc. 6th Ann. European Symp. on Algorithms* (1998), vol. 1461 of *LNCS*, Springer, pp. 139–150.