# Fast Computation of Determinants


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

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

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


January 21, 1999


**Abstract**

In this paper we consider the exact computation of determinants of matrices of integers. We present a new practical algorithm with worst case complexity $O(n^4 \log n)$, where $n$ is the dimension of the matrix, but with expected complexity $O(\epsilon n^4 + n^3 \log(n)^2)$. We include timing data to compare our algorithm with existing ones. This data also shows that the cubic term dominates the growth for $n$ up to 400. 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 (see [6]). Moreover, many geometrical properties can be expressed as the sign of a particular determinant (see also [2]).

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 (see [8], [4]). These normal forms can be computed more efficiently modulo the determinant of a particular matrix (see [3], [5]).

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 $O(n^5)$. The Chinese remaindering method has complexity $O(n^4 \log n)$ (see [1]).

In [2] a modified Chinese remaindering method is used to determine the sign of the determinant. The complexity of this algorithm is also $O(n^4 \log n)$.

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, 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, for some matrices it is equivalent to the full Chinese remaindering method.

# 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 (see also [9]) that for a given matrix $M$ the solution to the linear system $Mx = b$, for $b$ random, typically has the determinant of $M$ as common denominator (except perhaps for some small prime factors). Experimentation soon showed that we almost always obtain $\pm\det(M)$ as the overall common denominator for solutions to three such random linear systems.

Now, a linear system over the integers can be solved in time $O(n^3(\log n)^2)$ using Hensel lifting (see [7]). The basis of our new algorithm is now obvious though a number of difficulties remain:

1. For some matrices $M$ we can obtain only a factor of $\det(M)$ by solving random linear systems (e.g. when $M$ is a multiple of the identity matrix).
2. Even for well-behaved matrices there is always a chance that the common denominator is only a factor of the determinant.

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

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

Now assuming the matrix $M$ is non-singular we obtain $S^{-1} = T/s_n$ for some diagonal matrix of integers, $T$. It is easy to see that when $Mx = b$ then $x = (VTUb)/s_n$. From this it follows that the common denominator of the solution to any linear system $Mx = b$ divides $s_n$.

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

Since $V, T, U$ and $b$ have all integer entries, we see from $x = (VTUb)/s_n$ that the common denominator $w$ 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, we have for a prime $p$ that $p|(s_n/w)$ if and only if $VTUb = 0$ modulo $p$. The following theorem gives a hint on the frequency that this might happen.

**Theorem 1** *Let $M$ be an $n \times n$ matrix over $\mathbb{Z}$ and $p \in \mathbb{Z}$ be prime. Then*

$$\#\{b \in \{0, 1, \ldots, p-1\}^n \mid Mb \equiv 0 \pmod{p}\} = p^{n-r}$$

*where $r$ is the rank of $M \pmod{p}$.*

*Proof:* Let $K$ be the kernel of $M \pmod{p}$ over $\mathbb{F}_p$. Then the dimension of $K$ is $n - r$. Since $Mb \equiv 0 \pmod{p} \iff b \pmod{p} \in K$, the theorem follows. •

Let $p|s_n$ be prime and $t$ such that $p^t|s_n$ and $p^{t+1} \not| s_n$. Since $U$ and $V$ are unimodular, they are invertible modulo $p$, and so the rank $\bar{r}$ of $VTU \pmod{p}$ is equal to the rank of $T \pmod{p}$, which is equal to the number of invariant factors $s_i$ such that $p^t|s_i$. In particular, when $M$ has trivial SNF, then $\bar{r} = 1$. From Theorem 1 it follows that for a randomly chosen $b \in \{0, 1, \ldots, p-1\}^n$, the probability that $p^t$ does not divide the common denominator of $M^{-1}b$ is $1/p^{\bar{r}}$.

Now choose $b$ with random coefficients from $\{0, 1, \ldots, N\}$. From the above it follows that for large $N$ and large $p$, there is only a low probability that the power of $p$ in the common denominator $w$ of $M^{-1}b$ is less than the power of $p$ in $s_n$. This means that the quotient $s_n/w$ will be small with high probability. By choosing several random $b$'s and taking the least common multiple $u$ of the common denominators obtained in that way, the quotient $s_n/u$ will be even smaller, with high probability.

The idea of our algorithm is to determine $\det(M)/u$ by using the well–known Chinese remaindering technique. From this we get $\det(M)$ immediately. We now present our algorithm:

**Input**: positive integer $n$, $n \times n$ matrix of integers $M$.
**Output**: The determinant of $M$.
(1) Using Hensel lifting compute solutions to two random linear systems. We use rational recovery to obtain the solutions.
(2) Put $D$ equal to the common denominator of these two solutions. We are now sure 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 (see [6]) for rows and columns.
(4) Now compute $\det(M)/D$ using Chinese remaindering techniques; we can start with the value and prime used in step (1). 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 $D(\det(M)/D)$.

Our algorithm can be viewed as a variant of the standard Chinese remaindering technique. The important point is that with very high probability the bound $H/D$ is far smaller than $H$ itself, 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.

The worst case for our method is when $D$ turns out to be 1. This must happen if the matrix is unimodular, and may happen with non-negligible probability if $s_n$ is small. In these cases our algorithm will be marginally slower than direct use of Chinese remaindering: the first two steps have overall complexity $O(n^3(\log n)^2)$ while the last two cost altogether $O(n^4 \log n)$ if $D$ is small.

### 2.1 Refinements and Implementation Details

As mentioned above, the worst case for our algorithm is when $D$ is small (compared to the bound). This will happen with certainty when $\det(M)$ 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.

When $D$ is small, we add 1 to a random entry $(i, j)$ of $M$, yielding the matrix $\hat{M}$. When $N$ is the matrix we get from $M$ by deleting row $i$ and column $j$, we have:

$$\det(M) = \det(\hat{M}) + (-1)^{ij} \det(N). \tag{1}$$

We now compute $\det(\hat{M})$ and $\det(N)$ using our algorithm and use (1) to obtain $\det(M)$.

When the reason for $D$ being small is that $\det(M)$ is small, then $\det(\hat{M})$ will be large whenever $\det(N)$ is large. Since $\det(N)$ is large in most cases, this will improve the performance of our algorithm when $\det(M)$ is small.

We compute the solution of the system $Mx = b$ using Hensel lifting. For this we choose a random prime $p$ and for which the system is non-singular modulo $p$. When the system happens to be singular modulo $p$, we cannot solve the system using this $p$ and we have to choose another prime. However, in this case we know that $\det(M) \equiv 0 \pmod{p}$ from which it follows that $\det(M)/D \equiv 0 \pmod{p}$. This information can then be used in step (4), so even "bad" primes are useful to us.

When $\det(M) = 0$, the above also applies. In that case we always find that $\det(M) \equiv 0 \pmod{p}$ and so we will never get a factor $D$ of the determinant. When this happens we can conclude that $\det(M) = 0$ when the product of the primes used exceeds $2H$. In that case the complexity is again $O(n^4 \log n)$.

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 $\{0, \ldots, p-1\}$ 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 $p$.

## 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 $10 \times 10$ 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 $|d|/H$ where $d$ is the determinant and $H$ is the row Hadamard bound; this value necessarily lies in the interval $[0, 1]$ and the closer it is to 1 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 $|d|/H$ 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 surface of the sphere. Thus our task is to compute $d_n$, the expected value of $|d|$ for $n \times n$ matrices having this distribution, for each integer $n$.

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 $v_1, \ldots, v_n$. Now we define $n-1$ angles as follows: let $\phi_1$ be the angle between $v_1$ and $v_2$, let $\phi_2$ be the angle between $\mathrm{span}(v_1, v_2)$ and $v_3$, and in general let $\phi_k$ 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

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

Let us focus our attention on $\phi_k$ for the moment. We begin by finding the density function for $\phi_k$, 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 $\phi_k$ 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 function proportional to $(\cos\phi_k)^{k-1}(\sin\phi_k)^{n-k-1}$. The formula for the expected value of $\sin\phi_k$ is now immediate:

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

Thanks to the symmetry of the sphere the distribution of $\phi_k$ is independent of the values of $\phi_1, \ldots, \phi_{k-1}$, 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 $I_{m,n}$ for the integral $\int_0^{\pi/2} (\sin\theta)^m\,(\cos\theta)^n\,d\theta$. So using this notation we have $\mathbf{E}(\sin\phi_k) = I_{n-k,k-1}/I_{n-k-1,k-1}$, and consequently

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

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

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

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, the right hand side is asymptotically $e^{-1}(1 + O(1/n^2))$, from which it easily follows that

$$d_n \in \theta(e^{-n/2}), \tag{2}$$

or equivalently there exist constants $C_1$ and $C_2$ such that $C_1 e^{-n/2} < d_n < C_2 e^{-n/2}$.

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 $200 \times 200$ with entries chosen independently and uniformly in an interval; while this distribution is significantly different from the uniform spherical one, we observed nonetheless that the ratio of the determinant to the Hadamard bound always lay close to $e^{-n/2}$.

### 3.1 Complexity issues

In this section we want to say a few words on the complexity of our algorithm. In the worst case, when $D = 1$, we perform in step (4) the full Chinese remaindering algorithm. In that case we have to compute the determinant for $O(n \log n)$ primes and the complexity is then $O(n^4 \log n)$.

However, from (2) it follows that the expected number of primes for which we have to compute the quotient of the determinant and $D$ is bounded by $O(n)$. From this it follows that we improve the complexity by a factor of $\log n$, i.e. the expected complexity of our algorithm is $O(n^4)$. Practical experimentations has shown however that for random matrices of size up to 400, the Hensel lifting step, which is cubic, dominates the final Chinese remaindering, which is quartic but with a smaller constant of proportionality.

## 4 Comparison with Existing Implementations

We have implemented our new algorithm using the programming language C, and using the GNUmp library for operations on big integers. Table 1 below presents a comparison of our implementation with a number of others: the column headed "Hensel" gives times for our new algorithm, "CRA" corresponds to our implementation of a pure Chinese Remaindering method, "NTL" refers to the determinant function for integer matrices supplied with the library NTL ([10]) while "Maple" and "REDUCE" refer to the respective computer algebra systems.

We considered four classes of matrix:

| matrix type | size | Hensel | CRA | NTL | REDUCE | Maple |
|---|---|---|---|---|---|---|
| small | $50 \times 50$ | 0.2 | 0.2 | 0.1 | 1.0 | 5.5 |
| | $100 \times 100$ | 1.1 | 1.5 | 2.1 | 18 | 150 |
| | $200 \times 200$ | 10 | 23 | 35 | $*$ | $*$ |
| | $400 \times 400$ | 100 | 380 | 570 | | |
| | $800 \times 800$ | 1100 | | | | |
| large | $50 \times 50$ | 1.1 | 2.1 | 1.1 | 20 | 140 |
| | $100 \times 100$ | 7.7 | 18 | 16 | $*$ | $*$ |
| | $200 \times 200$ | 64 | 220 | 240 | | |
| | $400 \times 400$ | 590 | 3400 | 3850 | | |
| unim | $50 \times 50$ | 0.5 | 0.3 | 0.2 | 0.2 | 1.9 |
| | $100 \times 100$ | 4.8 | 3.3 | 3.8 | 1.8 | 18 |
| | $200 \times 200$ | 55 | 48 | 64 | $*$ | 240 |
| | $400 \times 400$ | 760 | 760 | 1070 | | |
| snf | $50 \times 50$ | 0.7 | 0.5 | 0.3 | 0.7 | 3.7 |
| | $100 \times 100$ | 6.5 | 4.5 | 4.7 | $*$ | 110 |
| | $200 \times 200$ | 71 | 61 | 77 | | |
| | $400 \times 400$ | 940 | 920 | 1260 | | |

Table 1: experimental timings

**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.

We note that the implementation used for the timings did not include the heuristic given above for use when the common denominator of the random solution is small. The table entries are CPU times in seconds; an entry $*$ means that the computation did not complete within one hour (or memory was exhausted), and 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 $800 \times 800$ where only one matrix was used).

The better practical complexity of the new algorithm compared to the Chinese Remaindering technique is immediately evident from this table. Furthermore, we see that the method is already competitive for matrices as small as $50 \times 50$. We note that the system REDUCE had particularly impressive performance for the unimodular matrices; we believe this is due to especially favourable behaviour of their implementation of the Bareiss algorithm on the test matrices we constructed. Apart from that exception it is clear that NTL and our implementations are superior to those included with the two computer algebra systems (in terms of both speed and size of matrix which can be handled).

## 5 Conclusion

The improvement in the theoretical expected complexity of our new algorithm appears to be relatively minor; in contrast, we see that, from a practical standpoint, the new algorithm becomes competitive for quite small matrices, and actually exhibits sub-quartic behaviour over the range of examples chosen in our tests. It is clearly the method of choice for large dense matrices of integers.

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.

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 $i$th 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 $O(n^3(\deg(M))^2)$ (see [7]).

## References


[1] E.H. Bareiss. Computational Solutions of Matrix Problems Over an Integral Domain. *J. Inst. Maths Applics* **10**, 68–104, 1972.

[2] H. Brönnimann, I.Z. Emiris, V.Y. Pan and S. Pion. Computing Exact Geometric Predicates Using Modular Arithmetic with Single Precision. *Symposium on Computational Geometry '97*, 174–182, 1997.

[3] P.D. Domich, R. Kannan and L.E. Trotter. Hermite Normal Form Computation Using Modulo Determinant Arithmetic. *Math. Oper. Res.* **12**, 50–59, 1987.

[4] M.A. Frumkin. Polynomial Time Algorithms in the Theory of Linear Diophantine Equations. *LNCS* **56**, 386–392, 1977.

[5] J.L. Hafner and K.S. McCurley. Asymptotically Fast Triangularization of Matrices over Rings. *SIAM J. Comput.* **20**, 1068–1083, 1991.

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

[7] T. Mulders and A. Storjohann. Diophantine Linear System Solving. *Submitted to ISSAC'99*, 1999.

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

[9] V. Pan. Computing the Determinant and the Charactersitic Polynomial of a Matrix via Solving Linear Systems of Equations. *Inf. Proc. Letters* **28**, 71–75, 1988.

[10] V. Shoup. NTL: A Library for doing Number Theory. `http://www.cs.wisc.edu/~shoup/ntl`