An output-sensitive variant of the baby steps/giant steps determinant algorithm*

Erich Kaltofen

Dept. of Mathematics, North Carolina State University, Raleigh, North Carolina 27695-8205 USA

kaltofen@math.ncsu.edu; http://www.kaltofen.net

1. INTRODUCTION

In the first half of 2000, two new algorithms were discovered for the efficient computation of the determinant of a (dense) matrix with integer entries. Suppose that the dimension of the matrix is $n \times n$ and that the maximum bit length of all entries is b. The algorithm by [10] requires $(n^{3.5}b^{2.5})^{1+o(1)}$ fixed precision, that is, bit operations. Here and in the following we use the exponent "$+o(1)$" to capture missing polylogarithmic factors $O((\log n)^{C_1}(\log b)^{C_2})$, where C_1, C_2 are constants ("soft-O"). As it has turned out an algorithm in [15], which in turn is based on one by [31] and which uses the baby steps/giant steps algorithm design technique, can be adapted to the dense integer matrix determinant problem and then has bit complexity $(n^{3.5}b)^{1+o(1)}$ [20, Section 2]. Both algorithms use randomization and the algorithm in [10] is Monte Carlo—always fast and probably correct—and the one in [20] is Las Vegas—always correct and probably fast. Both algorithms can be speeded by asymptotically fast subcubic matrix multiplication algorithms à la Strassen [8, 7, 14]. By blocking [6, 16, 29, 30] the baby steps/giant steps algorithm can be further improved, which yields the currently fastest known algorithms [20, Section 3] of bit complexity $(n^{3+1/3}b)^{1+o(1)}$, that without subcubic matrix multiplication and without the FFT-based polynomial "half" GCD procedures à la Knuth [23; 2, Chapter 8], and of bit complexity $n^{2.698}b^{1+o(1)}$ with subcubic matrix multiplication and FFT-based polynomial GCD procedures.

However, under certain favorable circumstances other algorithms can be faster. In our survey [21] we list Gaussian elimination combined with Chinese remaindering, Hensel lifting [1] and the above cited result [10] as propitious. For example, if the the determinant Δ is a small integer, Chinese remaindering can employ what is know as "early termination." One chooses random moduli and stops as soon as the (balanced) residue does not change any further [4]. The bit complexity of the resulting Monte Carlo algorithm is then $((\log|\Delta|)n^3 + bn^2)^{1+o(1)}$, that with standard matrix operations. If the input matrix has a large first invariant factor in the Smith form, Hensel lifting can reconstruct the determinant in cubic time. A small number of distinct invariant factors is conducive for the method of [10].

This paper provides an adaptive version of the unblocked baby steps/giant steps algorithm [20, Section 2]. The result is most easily stated when $b \le \log|\Delta| = (n^{1-\eta}b)^{1+o(1)}$ where Δ is the determinant to be computed and η with $0 \le \eta \le 1$ is not known. Note that by Hadamard's bound $\log_2|\Delta| \le n(b + \log_2(n)/2)$, so $\eta = 0$ covers the worst case. We describe a Monte Carlo algorithm that produces Δ in $(n^{3+1/2-\eta/2}b)^{1+o(1)}$ bit operations, again with standard matrix arithmetic. The corresponding bit complexity of the early termination Gaussian elimination method is $(n^{4-\eta}b)^{1+o(1)}$, which is always more, and that of the algorithm by [10] is $(n^{3+1/2-\eta/2}b^{1+1/2})^{1+o(1)}$.

Our adaptive determinant algorithm can be speeded by use of subcubic matrix multiplication algorithms so as to outperform an early termination Gaussian elimination algorithm that employs subcubic matrix multiplication. Such results seem, however, of purely theoretical interest; see Section 4 for a more in-depth discussion. Here we add that the exponent "$+o(1)$" in the version that uses cubic matrix multiplication and that has bit complexity $(n^{3+1/2-\eta/2}b)^{1+o(1)}$ is introduced (except when $b \gg n$) because the moduli of the Chinese remainder algorithm cannot be chosen of fixed magnitude. However, for all practical purposes primes with 32 or 64 bit will suffice to recover determinants of any reasonable length, say of fewer 10^{10} binary digits. Therefore the polylogarithmic factors in our complexity estimates do not degrade the practical performance of our method.

In section 2 we shall give the probability and complexity analyses for Chinese remaindering with early termination. Our bounds are valid for any algorithm that uses Chinese remaindering and thus of broader interest. For this reason, we carry out the analyses allowing for so-called thresholds. Thresholds are useful when the probability becomes too high that a single new modulus falsely triggers the early termination condition, which checks that the current residue candidate remains unaffected by the new modulus. Instead one requires that the residue does not change when adding in the Chinese remainder algorithm ζ new moduli. We call ζ the

*This material is based on work supported in part by the National Science Foundation under Grants Nos. DMS-9977392, CCR-9988177 and CCR-0113121.

threshold of early termination. For polynomial interpolation the benefit of higher thresholds, both theoretically and practically, are discussed in [17, 22]. We give the corresponding analyses for Chinese remaindering.

Section 3 presents the adaptive determinant algorithm and section 4 mentions improvements by subcubic matrix multiplication.

2. CHINESE REMAINDERING WITH EARLY TERMINATION

Let us suppose that we perform the Chinese remainder algorithm with a list of distinct prime module $p_1, p_2, \dots, p_m$ and for an integer $M \neq 0$ we reconstruct an integer $N \equiv M \pmod{p_1 \cdot p_2 \cdots p_m}$ from the residues $M \bmod p_i$, where $1 \leq i \leq m$. The Newton interpolation (method of divided differences) algorithm applied to the Chinese remainder problem computes the mixed radix representation for N, namely the integer coefficients c_i such that

$$\begin{aligned} N = c_0 + c_1 p_1 + c_2 p_1 p_2 + \cdots + c_{\delta-1} p_1 \cdots p_{\delta-1}, \\ \text{where } 1 \leq \delta \leq m, c_{\delta-1} \neq 0, |c_i| < p_{i+1} \\ \text{and } \operatorname{sign}(c_i) = \operatorname{sign}(N) \text{ for } 0 \leq i \leq \delta - 1. \quad (1) \end{aligned}$$

The algorithm is iterative and uses the formula

$$\begin{aligned} c_i = (N - c_0 - c_1 p_1 - \cdots - c_{i-1} p_1 \cdots p_{i-1}) \\ \cdot (p_1 \cdots p_i)^{-1} \bmod p_{i+1}. \quad (2) \end{aligned}$$

The required number of residue operations, including divisions, are of order $O(m^2)$. As is indicated in (1), the sign of the mixed radix coefficients c_i is dependent on the sign of N. The number δ of coefficients is determined by the magnitude of the absolute value of N as that index value satisfies $p_1 \dots p_{\delta-1} \leq |N| < p_1 \cdots p_\delta$. If $N < 0$ the mixed radix representation for $N' = p_1 \cdots p_m + N = c'_0 + c'_1 p_1 + \cdots + c'_{m-1} p_1 \cdots p_{m-1}$ yields the one for N,

$$\begin{aligned} N = (-p_1 + c'_0) + (-p_2 + c'_1 + 1) p_1 + \cdots \\ + (-p_m + c'_{m-1} + 1) p_1 \cdots p_{m-1}. \quad (3) \end{aligned}$$

The early termination strategy determines δ in (1) by first picking random prime moduli $p_1, \dots, p_m$ of a certain magnitude and then stopping when $\zeta \geq 1$ many consecutive residues are found to be zero for the first time, that is $c_i = \cdots = c_{i+\zeta-1} = 0$. In that case the algorithm asserts δ to be with high probability equal to i. Note that in our formulation the test is only valid for $i \leq m - \zeta$, because we always stop after m moduli. Following [17] we call ζ the threshold of early termination. The strategy is justified in the sense of a Monte Carlo algorithm because with high probability the condition cannot occur for $i < \delta$. We shall give the probabilistic analysis now.

THEOREM 1. *Let $p_1, \dots, p_m$ be distinct prime numbers that are uniformly and randomly selected from the interval $2 \leq p_i \leq m^\gamma \log(m)$ for all i with $1 \leq i \leq m$, where γ is a real constant with $\gamma > 1$ and ζ a threshold with $\zeta \geq 1$. For the coefficients c_i we then have the following estimate on the probability*

$$\begin{aligned} \operatorname{Prob}(\forall i, 0 \leq i \leq \delta - \zeta - 1 \colon \exists j, 0 \leq j \leq \zeta - 1 \colon c_{i+j} \neq 0) \\ \geq 1 - O(1/m^{\zeta(\gamma-1)-1}). \quad (4) \end{aligned}$$

Proof. We first prove that the probability that in (1) $c_i = 0$ for a given $i < \delta$ is $O(1/m^{\gamma-1})$. By (2) the prime p_i must divide $N_i = N - c_0 - c_1 p_1 - \cdots - c_{i-1} p_1 \cdots p_{i-1}$. We have $|N_i| < p_1 \cdots p_m \leq B^m$, where $B = m^\gamma \log(m)$ is the bound on the magnitude of the moduli. We shall compare two counts: the maximum number of prime divisors of N_i versus the number of primes from which p_i can be chosen. Inequalities for these counts can be given from properties of the distribution of prime numbers, namely

$$\prod_{\substack{p \text{ prime} \\ p \leq x}} p > e^{C_1 x}, \pi(x) = \sum_{\substack{p \text{ prime} \\ p \leq x}} 1 > \frac{C_2 x}{\log_e x}, \pi(x) < \frac{C_3 x}{\log_e x} \quad (5)$$

where C_1, C_2 and C_3 are positive constants. Explicit values for C_1, C_2 and C_3 have been derived. It is shown in [25] that for $x \geq 101$ we may choose $C_1 = 0.84$, for $x \geq 17$ we may choose $C_2 = 1$, and for $x \geq 114$ we may choose $C_3 = 1.25$. From (5) it follows that if x_0 is such that $e^{C_1 x_0} = |N_i|$ then the number of distinct prime factors of N_i is less than $\pi(x_0)$. We have $x_0 = O(m \log B)$ where $\log(B) = O(\log m)$. Hence $\pi(x_0) < C_3 x_0 / \log(x_0) = O(m)$. The number of available primes for p_i is $\pi(B) - i + 1$, the difference by $i - 1$ excluding the primes $p_1, \dots, p_{i-1}$ to be picked. From the magnitude of B, $\gamma > 1$ and (5) we infer the existence of a real constant $C_4 > 0$ such that $\pi(B) - i + 1 > C_4 m^\gamma$. We have established our initial claim.

We conclude the proof similarly to [17, Theorem 4]. The probability that $0 = c_i = \cdots = c_{i+\zeta-1}$ is $O(m^{\zeta(\gamma-1)})$. The events are dependent for $i = 0, 1, \dots, \delta - \zeta - 1$. Nonetheless, the probability that at least for one i a sequence of ζ zero coefficients occurs is $O((\delta - \zeta) m^{\zeta(\gamma-1)})$, which estimates the sum of the probabilities. The theorem gives the probability for the complementary event with the looser bound $m > \delta - \zeta$. $\boxtimes$

Next we shall discuss several issues of practicality. It is important to realize that the estimate (4) is lower bound for the stated probability. Even for $\gamma = \zeta = 1$ the actual probability may be bounded away from zero and the early termination algorithm may correctly terminate at $c_{\delta+\zeta} = 0$. We note that the constant implied by the big-O in the lower bound (4) can be explicitly calculated if one wishes to guarantee the termination with a given probability, although that claim would also require true random choices for the prime moduli. In practice, it seems important to work with thresholds $\zeta \geq 2$. One reason is already exhibited by the estimate (4): higher thresholds permit smaller moduli. For instance, if $\gamma = 2$ we obtain a probability bound away from 0 for threshold $\zeta = 2$, namely $1 - O(1/m)$. Another reason is that in the Chinese remainder setting it is advantageous to preselect the stream of prime moduli $p_1, p_2, \dots$ so that one can precompute constants that are independent of the input residues $M \bmod p_i$, for instance $(p_1 \cdots p_i)^{-1} \bmod p_{i+1}$ in (2). Such moduli are not random, but with a higher threshold one can avoid a premature termination for many inputs. An additional postcheck of the result at one or several additional random moduli is often possible and further reduces the chance of producing an erroneous result. An empirical study on how the threshold and postcheck count parameters improve the success rate for small moduli, in the setting of polynomial interpolation, can be found in [17, 22].

The early termination strategy seems to belong to the "folklore" of computer algebra. We have used early termination in the mid-1980s [12, 19] for purpose of determining the degree of a straight-line and black box polynomial. Our algorithms perform Newton interpolation at non-random points and test whether the interpolant agrees with the input polynomial at a random point, thus allowing for preconditioning in the interpolation process while guaranteeing a given probability of success. Chinese remaindering with early termination is applied to exact computations in geometry by [11]. Our discussion above introduces thresholds. We formulate the problem without a bound on the magnitude M, but note that the lack of a bound on $|M|$ prevents probability guarantees to be made about the correct recovery of M (as N) when early termination has occurred. Nonetheless, the test seems to be a reasonable heuristic even under those circumstances.

We now turn to bit complexity considerations of the Chinese remainder problem. Suppose that the moduli are bounded as in Theorem 1, namely $p_i = m^{O(1)}$ for all $1 \le i \le m$. As stated above, the Newton interpolation formula (2) costs $O(m^2)$ residue operations and therefore $m^{2+o(1)}$ bit operations. Here we shall remark that the exponent "$+o(1)$" is for most practical purposes equal to 0. If we choose moduli of 32 bits we have by (5) at least $1.98 \cdot 10^8$ primes at our disposal, for 64 bits at least $4.20 \cdot 10^{17}$. However, the method is still quadratic in m, and for later asymptotically fast analysis we shall discuss how to reduce the complexity to $m^{1+o(1)}$ bit operations. We note that these fast methods may not be of practical significance. For Chinese remaindering by the Lagrangian interpolation formula the tree-like evaluation schemes, which yield a bit complexity of $O(L(\log L)^2 \operatorname{loglog} L)$ where $L = \log(p_1 \cdots p_m)$ are discussed in [13; 2, Chapter 8]. The Lagrangian formula produces N in binary representation, which could be converted to mixed radix representation with an additional $O(L(\log L)^2 \operatorname{loglog} L)$ bit operations. The straight-forward divide-and-conquer technique seems to yield $O(L(\log L)^3 \times \operatorname{loglog} L)$ bit complexity when applied directly to the Newton interpolation approach.

However, the Lagrangian algorithm is "offline," meaning that for an additional new modulus and residue one starts over at the beginning. Asymptotically, the bit complexity will not be affected in our baby steps/giant steps algorithm, as we explain now. The determinant algorithm performs Chinese remaindering for

$$\mu_i = b(4 + 4^2 + \cdots + 4^i) = (4/3)b(4^i - 1), \quad i = 1, 2, \ldots \quad (6)$$

moduli at a time. In the above equation (6) the integer $b \ge 1$ is an additional parameter. The early termination criterion now checks for each i if

$$N < p_1 \cdots p_{\mu_i - \zeta} \quad \text{or} \quad p_1 \cdots p_{\mu_i} - N < p_1 \cdots p_{\mu_i - \zeta}, \quad (7)$$

so the mixed radix representation needs not to be computed.

When the point is reached where

$$|M| < p_1 \cdots p_{\mu_k - \zeta}, \quad (8)$$

the algorithm stops because then the early termination criterion is satisfied for sure for the moduli $p_1, \ldots, p_{\mu_k}$. Here k is the last value of i in the iteration assuming no false early termination has occurred. Since we have Hadamard's bound for the determinant, we shall continue the complexity analysis in terms of $|M|$, a bound $H \ge |M|$, and the threshold ζ. Thus the probability of correct recovery can be properly estimated. We note that with a bound $H \ge |M|$ for large M one may be able to stop when $2H < p_1 \cdots p_m$ thereby avoiding to verify the threshold condition for $p_{m+1}, \ldots$ We can make the following simplifying assumption. Because we now know that $m = 1 + \lceil \log_2 H \rceil$ is a sufficient number of moduli ($p_j \ge 2$), by (4) in Theorem 1 we can for all moduli choose the bound

$$p_j \le m^\gamma \log(m) = O(B_H) \quad \text{with } B_H = (\log H)^\gamma \operatorname{loglog} H. \quad (9)$$

We shall show that for random moduli of such size the bit complexity of the early termination strategy with the iteration (6) and using fast Lagrangian Chinese remaindering and the termination test (7) is

$$O(L(\log L)^2 \log L) \quad \text{where } L = (\zeta + b + \log |M|) \operatorname{loglog} H. \quad (10)$$

For each i the interpolation including the early termination test (7)—the products $p_1 \cdots p_{\mu_i - \zeta}$ and $p_1 \cdots p_{\mu_i}$ are computed via a tree [2, Algorithm 8.4]—has bit complexity $O(L_i(\log L_i)^2 \operatorname{loglog} L_i)$ with $L_i = \mu_i \log(B_H)$. By summing $\sum_{i=1}^{k} \mu_i = O(\mu_k)$ (see (6)) the total bit complexity remains $O(L_k(\log L_k)^2 \operatorname{loglog} L_k)$.

Finally, if for an integer ν with

$$\mu_{\nu-1} = 4/3b(4^{\nu-1} - 1) \le \zeta + \log_2 |M| < \mu_\nu = 4/3b(4^\nu - 1), \quad (11)$$

we have $k \le \nu$. The latter follows from $p_1 \cdots p_{\mu_\nu - \zeta} \ge 2^{\mu_\nu - \zeta} > |M|$, which satisfies (8). We conclude that $\mu_k \le \mu_\nu = 4\mu_{\nu-1} + 4b \le 4(\zeta + b + \log_2 |M|)$ and hence by (9) the bit length satisfies $\log(p_1 \cdots p_{\mu_k}) \le \mu_k \log(B_H) = O((\zeta + b + \log |M|) \operatorname{loglog} H)$, as is stated in (10).

3. BABY STEPS/GIANT STEPS WITH EARLY TERMINATION

The baby steps/giant steps algorithms of [20] can utilize the early termination property of Chinese remaindering with random prime moduli that was discussed in the previous section. The idea is to double the baby step range r and quadruple the number of prime moduli until the determinant stabilizes. We assume that we have chosen a stream of random prime residues $p_1, p_2, p_3, \ldots$ and that the mixed radix representation of $\det(A)$ with respect to those primes satisfies the early termination condition of Section 2, that is has no zero coefficients or, more formally, for

$$\det(A) = c_0 + c_1 p_1 + c_2 p_1 p_2 + \cdots + c_\delta p_1 \cdots p_{\delta-1} \quad (12)$$

either $0 < c_i < p_{i+1}$ for all i or $-p_{i+1} < c_i < 0$ for all p_i (cf. (1)). The assumption can be weakened to excluding a block of ζ consecutive zero coefficients, formally $\forall 0 \le i \le \delta - \zeta - 1\colon \exists 0 \le j \le \zeta - 1\colon c_{i+j} \ne 0$. We then call ζ the used threshold of termination. As we have shown in Theorem 1 above, a threshold $\zeta > 1$ increases the probability of correct early termination.

For the remainder of this section, we shall discuss early termination for our baby steps/giant steps version [20, section 2] of Wiedemann's original determinant algorithm [31, section V]. Improvements by fast matrix multiplication are deferred to section 4. While Wiedemann intended his algorithm for sparse matrices over finite fields, we apply his method to dense matrices with integer entries. First, we describe the method in its entirety.

Algorithm *Adaptive Baby Steps/Giant Steps Determinant*
Input: a matrix $A \in \mathbb{Z}^{n\times n}$ and an early termination threshold $\zeta \ge 1$.
Let b be the maximum bit length of all the entries in A, and let $H = 2^{bn} n^{n/2}$ be Hadamard's determinant bound $H \ge |\det A|$, and let $h = \log(H) = (nb)^{1+o(1)}$.
Output: an integer N such that

$$N = \det(A) \text{ with probability } 1 - O(1/h^{\zeta(\gamma-1)-1}),$$

or "failure." Here $\gamma > 1$ is a constant that controls the magnitude of the moduli (cf. Theorem 1 and (9)). The probability that failure arises is given in the proof of Theorem 2 below.

Step 1. Precondition A such that the minimum polynomial of the new matrix is equal its characteristic polynomial, provided A was non-singular on input. We have two very efficient preconditioners at our disposal. The first is $A \leftarrow DA$ where D is a random diagonal matrix with the diagonal entries chosen uniformly and independently from a set S of integers [5, Theorem 4.3]. The second is $A \leftarrow EA$ where

$$E = \begin{bmatrix} 1 & w_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & 1 & & w_{n-1} \\ 0 & \dots & 0 & & 1 \end{bmatrix}, \quad w_i \in S.$$

[28]. The product DA is slightly cheaper than EA, but recovery of $\det(A)$ requires division by $\det(D)$. Thus, all moduli that divide $\det(D)$ would have to be discarded from the Chinese remainder algorithm for the first preconditioner. The desired property, namely that the minimum polynomial is the characteristic polynomial in the nonsingular case, is achieved for both preconditioners with probability $1 - O(n^2/|S|)$. We shall choose $S = \{i \mid -n^{\gamma'} \le i \le n^{\gamma'}\}$, where $\gamma' > 2$ is a real constant.

Step 2. Initialize $r \leftarrow 1$; $Z \leftarrow A$;
assign to u, v random vectors in S^n. Now with probability $1 - O(n/|S|)$ [18, Lemma 2] the minimum linear generator of $u^{Tr} A^i v$ for $i = 0, 1, \dots$ is equal to the minimum polynomial of A. By Step 1 this will be with high probability the characteristic polynomial, from which the determinant of the input matrix is extracted. We note that if either the preconditioning in Step 1 or the bilinear projections u, v are unlucky, the degree of the minimum generator will be less than n.

While early termination has not occurred in Step 8
Do Steps 3–8;

Step 3. Set the baby steps length $r \leftarrow 2r$; note that the i-th time in the loop we have $r = 2^i$. $s \leftarrow \lceil 2n/r \rceil$.
We pick a new list of $d = r^2 b$ distinct random primes $p_{t+1}, \dots, p_{t+d}$ no greater than $h^\gamma \log h$ (see (9)). Note that at this point we already have Chinese remaindered $\det(A)$ with as many as $t = \mu_{i-1} = (4+16+\dots+4^{i-1})b$ prime moduli (cf. (6)) as we reach this point for the i-th time, for $i \ge 2$; for $i = 1$ we have $t = 0$.

Step 4. For $j = 1, 2, \dots, r-1$ Do $v^{[j,l]} \leftarrow A^j v \bmod p_{t+l}$ for all $1 \le l \le d$;

Step 5. $Z \leftarrow Z^2$; note that now $Z = A^r$.

Step 6. For $k = 1, 2, \dots, s$ Do $(u^{[k,l]})^{Tr} \leftarrow u^{Tr} Z^k \bmod p_{t+l}$ for all $1 \le l \le d$;

Step 7.
```
For $j = 0, 1, \dots, r-1$ Do
    For $k = 0, 1, \dots, s$ Do
        $(u^{Tr} A^{kr+j} v \bmod p_{t+l}) = a^{[l]}_{kr+j} \leftarrow$
                $(u^{[k,l]})^{Tr} v^{[j,l]} \bmod p_{t+l}$;
                        for all $1 \le l \le d$;
```

Step 8. For $l \leftarrow 1, 2, \dots$ Until $l = d$ Do
From $[a_i^{[l]}]_{0 \le i < 2n}$ try to compute $\det(\lambda I - A) \bmod p_{t+l}$. If the degree if the minimum generator is below n and the constant coefficient is not 0, discard p_{t+l}. If the preconditioning of Step 1 or the projections of Step 2 were unlucky, no modulus will succeed except possibly those dividing the determinant. We therefore cannot skip over an arbitrary number of moduli here, but shall terminate the algorithm with "failure," again after a certain threshold ζ' of encounters of that situation.

If the the sequence $a_i^{[l]}$ has revealed $(\det(A) \bmod p_{t+l})$ extend the mixed radix representation for $\det(A)$ with the obtained value. Check if ζ many mixed radix coefficients c_i in a row are zero for either $N = (\det(A) \bmod p_1 \cdot p_2 \cdots p_{t+l})$ or for $p_1 \cdot p_2 \cdots p_{t+l} - N$. The mixed radix coefficients for the latter can be computed via (3). An alternative check is (7). If the check succeeds, exit the loop and return N. ⊠

We shall give the running time in terms of n, b and $\delta' = \log_2 |\det A| \ge \delta$, where δ is defined in (12), The point is that after no more than $\delta + \zeta$ many primes the early termination condition must be triggered. It remains to determine how many bit operations are performed when the above algorithm examines $\delta + \zeta$ many primes.

Theorem 2. *The adaptive baby steps/giant steps determinant algorithm requires on any matrix A an order of*

$$\left(\sqrt{b(b + \zeta + \log|\det A|)} \cdot n^3\right)^{1+o(1)}$$

bit operations, where the $+o(1)$ exponent captures poly-logarithmic factors and big-O constants. The algorithm does not return "failure" with probability greater than $1/2$.

Proof. We first analyze the bit complexity. Capturing polylogarithmic factors via "$+o(1)$" exponents, for each individual iteration Steps 5 and 6 perform $(rn^3b)^{1+o(1)}$ bit operations. In Step 5 a single matrix multiplication is done on integers of length $(rb\log n)^{1+o(1)}$. For Step 6 we have $dsn^2 = O(r^2b(n/r)n^2)$ operations modulo the selected small primes from the stream. The cost of Step 6 includes taking Z modulo $d = r^2b$ primes, which costs $(r^2bn^2)^{1+o(1)}$ by remaindering modulo many primes via tree evaluation [2, Algorithm 8.4].

Step 4 is bounded by $(r^2bn^2)^{1+o(1)}$. We first compute the vectors A^jv with exact integer entries of length no more than $(rb\log n)^{1+o(1)}$. We then take the rn entries modulo the $d = r^2b$ primes in $(r^2brn)^{1+o(1)}$ bit operations. Step 7 is bounded by $(drsn)^{1+o(1)}$ bit operations, which is like in Step 4 $(r^2bn^2)^{1+o(1)}$.

Step 8 first does Berlekamp/Massey iterations modulo each p_{t+l}, which costs in total $(r^2bn^2)^{1+o(1)}$ bit operations. With classical Newton iteration, the Chinese remainder updates cost $O(d^2)$ residue operations, which introduces a b^2 term in the bit complexity. The bit complexity can be reduced to $(r^2b\log h)^{1+o(1)}$ by the asymptotically fast methods discussed at the end of section 2.

We now sum up all individual running times over the duration of the while-loop. The algorithm stops for $i = k$ with

$$\mu_k = (4 + 16 + \cdots + 4^k)b \geq \delta + \zeta > \mu_{k-1}$$

(cf. (11)). Thus $1/3b(4^k - 4) < \delta + \zeta$ which implies $4^k = O((\delta + \zeta + b)/b)$. The overall cost is

$$(2n^3b + 4n^3b + 8n^3b + \cdots + 2^kn^3b)^{1+o(1)},$$

which summed up is $(2^kbn^3)^{1+o(1)}$ and by the bound on $4^k = (2^k)^2$ the total bit complexity $(\sqrt{b(\delta + \zeta + b)} \cdot n^3)^{1+o(1)}$.

The bit operation counts for Steps 1–3 are dominated by the bit complexity $(bn^3)^{1+o(1)}$. From the bound H we know that $\delta + \zeta = (bn)^{1+o(1)}$, which is proportional to the number of primes needed. Each random prime has bit length $O(\log h)$ and can be computed by any of the randomized Monte Carlo algorithms in $(\log h)^{O(1)}$ bit operations.

It remains to estimate the probability of failure. Our algorithm returns "failure" under several circumstances. First, the algorithm may fail to establish a stream of distinct random primes. Second, the Wiedemann-style randomizations in Steps 1 and 2 may be unlucky. And third, in Step 8 we may encounter unusable moduli even in the presence of good choices in Steps 1 and 2. The first condition is avoided with probability no less than $1 - (bn\log h)^{1+o(1)}/2^w$. Here the quantity $bn\log h$ is proportional to the number of integers tested, the $\log h$ factor representing the prime density (5). The integer w is the number of witnesses for compositness that are probed for each integer. We shall tacitly assume that we have a random number generator for distinct integers in a given range. Wiedmann-style randomizations are successful with probability no less than $1 - O(1/n^{\gamma'-2})$.

Finally, we give a condition that excludes unlucky moduli in Step 8. Let φ be the degree of the minimum polynomial of the preconditioned matrix A, and let $a_i = u^{Tr}A^iv \in \mathbb{Z}$. Consider the Toeplitz matrix

$$T = \begin{bmatrix} a_{\varphi-1} & a_{\varphi-2} & \ldots & a_1 & a_0 \\ a_\varphi & a_{\varphi-1} & \ldots & a_2 & a_1 \\ \vdots & a_\varphi & \ddots & \vdots & a_2 \\ & \vdots & & & \vdots \\ a_{2\varphi-3} & & & a_{\varphi-1} & \\ a_{2\varphi-2} & a_{2\varphi-3} & \ldots & a_\varphi & a_{\varphi-1} \end{bmatrix}.$$

By the theory of linearly generated sequences [18, Lemma 1], any prime p_{t+l} that does not divide the determinant of T has the modular image of the rational linear generator as its modular generator. We have $\log|\det T| = (n^2b)^{1+o(1)}$ and hence by arguments similar as in the proof of Theorem 1, none of the selected primes divide $\det(T)$ with probability no less than $1 - (\delta + \zeta)(\log|\det T|)^{1+o(1)}/h^\gamma$, or in terms of n and b, no less than $1 - (n^3b^2)^{1+o(1)}/(nb)^\gamma$. ⊠

For example, if $\log|\det A| = O(n^{1-\eta}b)$, where $0 \leq \eta \leq 1$, and $\zeta = O(1)$, we obtain bit complexity $(n^{3+1/2-\eta/2}b)^{1+o(1)}$. The algorithm is randomized of the Monte Carlo kind. If the early termination condition is falsely discovered by a choice of unlucky moduli, an incorrect value for the determinant is returned. For very small determinants $|\det A| + \zeta < b$ it is faster to perform early termination Chinese remaindering with Gaussian elimination with a complexity $(n^3\log|\det A| + n^2b)^{1+o(1)}$.

We end this section with a historical remark. The notion of the "baby steps/giant steps" algorithm design paradigm was coined by Daniel Shanks in the late 1960s for several number theoretic algorithms. Our application of this paradigm is reminiscent of the algorithm by [24] for evaluating a scalar polynomial at a matrix.

4. SPEEDUP BY FAST MATRIX MULTIPLICATION

The non-adaptive baby steps/giant steps determinant algorithm was speeded by several techniques. One, which is already proposed in [15], employs asymptotically faster matrix multiplication algorithms à la Strassen. In fact, there are two improvements, one based on fast square matrix multiplication [8] and one in a "Note added in proof" posted in the web copy of the paper at http://www.math.ncsu.edu/~kaltofen, which also utilizes the special complexities of rectangular matrix multiplication [7, 14]. A more significant improvement is the use of blocks of vectors in place of the single projection vectors u, v [20, Section 3]. Again, one obtains running times using standard matrix multiplication and faster ones using asymptotically fast matrix multiplication. Finally, one may employ the half GCD algorithm for the Berlekamp/Massey step [3], which computes the minimum polynomial of the sequence of projected matrix powers, but which in the blocked case operates on polynomials with matrix coefficients (see, e.g., [27]). We note that our asymptotically fast Chinese remainder method of section 2 already made use of the half GCD algorithm.

Fast matrix multiplication allows for improvement of our adaptive techniques similar to the ones discussed in [15], which may only have a theoretical relevance. Asymptotically fast matrix multiplication is not entirely impractical. One referee has pointed us to [9], where is is shown that for $n \geq 128$ Strassen's algorithm can outperform the classical cubic method for matrices with floating point entries on an Apple G4 computer. Unfortunately, except in Step 5, the baby steps/giant steps algorithm of Section 3 employs Strassen's matrix multiplication algorithm to matrices of dimension $n^{0.45} \times n^{0.45}$ [15, Note added in proof in 1995] in order to lower the asymptotical complexity, which becomes practical only for very large n.

Let ω be the exponent for fast square matrix multiplication. The dominant costs are in Steps 5 and 6 of the adaptive baby steps/giant steps determinant algorithm. Clearly, Step 5 can be executed in $(rbn^\omega)^{1+o(1)}$ bit operations. Step 6 can be implemented in

$$\left(s \left(\frac{n}{d/(rb)}\right)^2 \left(\frac{d}{rb}\right)^\omega rb \right)^{1+o(1)} \tag{13}$$

bit operations as follows. We compute s products

$$(u^{[k+1]})^{Tr} \leftarrow (u^{[k]})^{Tr} Z \bmod p_{t+1} \cdots p_{t+d}.$$

The vectors $u^{[k,l]}$ are obtained by taking each entry of the resulting vectors modulo all p_{t+l} by a tree-based remaindering algorithm. Each matrix times vector product is performed by splitting the integer entries in $(u^{[k]})^{Tr}$, which are $O(d)$ bits long, into blocks of br digits, which is the asymptotic length of the entries in Z. We then perform the $d/(br) \times n$ times $n \times n$ matrix product with square $d/(br) \times d/(br)$ blocks. We note that asymptotically fast rectangular matrix algorithms can slightly speed the complexity here.

If one chooses $d = r^{(\omega-1)/(\omega-2)}\, b/n^{(3-\omega)/(\omega-2)}$ the complexity (13) again is $(rbn^\omega)^{1+o(1)}$. The rest of the analysis is as in section 3 and leads for $\log_2 |\det A| = O(n^{1-\eta} b)$ to a total bit complexity of

$$(n^{(\omega^2-\omega+1-\eta(\omega-2))/(\omega-1)} b)^{1+o(1)}$$

which for $\omega = 2.375477$ is $n^{3.1025-0.2730\eta} b^{1+o(1)}$. For $0 \leq \eta \leq 0.3754$ the algorithm out-performs Gaussian elimination with early termination Chinese remaindering, which has bit complexity $(n^{\omega+1-\eta} b)^{1+o(1)}$.

The asymptotically fast analysis above shows that fast matrix multiplication algorithms yield reduced exponents when applied to the baby steps/giant steps method in conjunction with the early termination strategy. By blocking, the bit complexity of the non-adaptive baby steps/giant steps determinant algorithm can be reduced to $n^{2.698} b^{1+o(1)}$ [20], again using asymptotically fast polynomial and matrix algorithms. A further reduction to $(n^\omega b)^{1+o(1)}$ may be possible by entirely different and new techniques [26]. On January 25, 2002, Victor Pan sent me a hand-written draft by fax that sketches how to apply the early termination strategy to the blocked version of the baby steps/giant steps determinant algorithm by [20] (for the case where cubic time matrix multiplication is used). Which among these methods, the unblocked or blocked adaptive baby steps/giant steps determinant algorithms or Storjohann's algorithm, which certifies the determinant, yields faster solutions in practice remains to be studied.

Acknowledgements: I thank the referees for their comments. Gilles Villard first brought the widely different categories of propitious inputs of the known determinant algorithms to my attention, which has inspired this work.
Correction made September 6, 2007 on page 140, third paragraph: loglog L replaces log L in several places.

Many of my papers can be retrieved on the Internet through links from my homepage.

5. REFERENCES

[1] Abbott, J., Bronstein, M., and Mulders, T. Fast deterministic computation of determinants of dense matrices. In *ISSAC 99 Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 1999), S. Dooley, Ed., ACM Press, pp. 181–188.

[2] Aho, A., Hopcroft, J., and Ullman, J. *The Design and Analysis of Algorithms.* Addison and Wesley, Reading, MA, 1974.

[3] Brent, R. P., Gustavson, F. G., and Yun, D. Y. Y. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *J. Algorithms 1* (1980), 259–295.

[4] Brönnimann, H., Emiris, I., Pan, V., and Pion, S. Sign determination in residue number systems. *Theoretical Comput. Sci. 210*, 1 (1999), 173–197. Special issue on real numbers and computers.

[5] Chen, L., Eberly, W., Kaltofen, E., Saunders, B. D., Turner, W. J., and Villard, G. Efficient matrix preconditioners for black box linear algebra. *Linear Algebra and Applications 343–344* (2002), 119–146. Special issue on *Structured and Infinite Systems of Linear Equations*, edited by P. Dewilde, V. Olshevsky and A. H. Sayed.

[6] Coppersmith, D. Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Math. Comput. 62*, 205 (1994), 333–350.

[7] Coppersmith, D. Rectangular matrix multiplication revisited. *J. Complexity 13* (1997), 42–49.

[8] Coppersmith, D., and Winograd, S. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput. 9*, 3 (1990), 251–280. Special issue on complexity theory.

[9] Crandall, R., and Klivington, J. Fast matrix algebra on Apple G4. http://developer.apple.com/hardware/ve/pdf/g4matrix.pdf, Mar. 2000.

[10] Eberly, W., Giesbrecht, M., and Villard, G. On computing the determinant and Smith form of an integer matrix. In *Proc. 41stAnnual Symp. Foundations of Comp. Sci.* (Los Alamitos, California, 2000), IEEE Computer Society Press, pp. 675–685.

[11] Emiris, I. Z. A complete implementation for computing general dimensional convex hulls. *Int. J. Comput. Geom. Appl. 8*, 2 (1998), 223–254.

[12] Freeman, T. S., Imirzian, G., Kaltofen, E., and Lakshman Yagati. Dagwood: A system for manipulating polynomials given by straight-line programs. *ACM Trans. Math. Software 14*, 3 (1988), 218–240.

[13] Heindel, L. E., and Horowitz, E. On decreasing the computing time for modular arithmetic. In *Conference Record, IEEE 12th Annual Symp. on Switching and Automata Theory* (1971), pp. 126–128.

[14] Huang, X., and Pan, V. Fast rectangular matrix multiplications and improving parallel matrix computations. In *Proc. Second Internat. Symp. Parallel Symbolic Comput. PASCO '97* (New York, N. Y., 1997), M. Hitz and E. Kaltofen, Eds., ACM Press, pp. 11–23.

[15] Kaltofen, E. On computing determinants of matrices without divisions. In *Proc. 1992 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'92)* (New York, N. Y., 1992), P. S. Wang, Ed., ACM Press, pp. 342–349.

[16] Kaltofen, E. Analysis of Coppersmith's block Wiedemann algorithm for the parallel solution of sparse linear systems. *Math. Comput. 64*, 210 (1995), 777–806.

[17] Kaltofen, E., Lee, W.-s., and Lobo, A. A. Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm. In *Proc. 2000 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'00)* (New York, N. Y., 2000), C. Traverso, Ed., ACM Press, pp. 192–201.

[18] Kaltofen, E., and Pan, V. Processor efficient parallel solution of linear systems over an abstract field. In *Proc. SPAA '91 3rd Ann. ACM Symp. Parallel Algor. Architecture* (New York, N.Y., 1991), ACM Press, pp. 180–191.

[19] Kaltofen, E., and Trager, B. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symbolic Comput. 9*, 3 (1990), 301–320.

[20] Kaltofen, E., and Villard, G. On the complexity of computing determinants. In *Proc. Fifth Asian Symposium on Computer Mathematics (ASCM 2001)* (Singapore, 2001), K. Shirayanagi and K. Yokoyama, Eds., vol. 9 of *Lecture Notes Series on Computing*, World Scientific, pp. 13–27. Invited contribution; extended abstract.

[21] Kaltofen, E., and Villard, G. Computing the sign or the value of the determinant of an integer matrix, a complexity survey. *J. Computational Applied Math.* (2002). Submitted to the special issue on Congrès International Algèbre Linéaire et Arithmétique: Calcul Numérique, Symbolique et Parallèle, held in Rabat, Morocco, May 2001, 17 pages.

[22] Lee, W. *Early termination strategies in sparse interpolation algorithms.* PhD thesis, North Carolina State Univ., Raleigh, North Carolina, Dec. 2001. 107 pages.

[23] Moenck, R. T. Fast computation of GCDs. In *Proc. 5th ACM Symp. Theory Comp.* (1973), pp. 142–151.

[24] Paterson, M. S., and Stockmeyer, L. J. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comp. 2* (1973), 60–66.

[25] Rosser, J. B., and Schoenfeld, L. Approximate formulas of some functions of prime numbers. *Illinois J. Math. 6* (1962), 64–94.

[26] Storjohann, A. Higher-order lifting. In *Proc. 2002 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'02)* (New York, N. Y., 2002), T. Mora, Ed., ACM Press, pp. 246–254.

[27] Thomé, E. Fast computation of linear generators for matrix sequences and application to the block Wiedemann algorithm. In *ISSAC 2001 Proc. 2001 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 2001), B. Mourrain, Ed., ACM Press, pp. 323–331.

[28] Turner, W. J. A note on determinantal divisors and matrix preconditioners. Paper to be submitted, Oct. 2001.

[29] Villard, G. Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In *ISSAC 97 Proc. 1997 Internat. Symp. Symbolic Algebraic Comput.* (New York, N. Y., 1997), W. Küchlin, Ed., ACM Press, pp. 32–39.

[30] Villard, G. A study of Coppersmith's block Wiedemann algorithm using matrix polynomials. Rapport de Recherche 975 IM, Institut d'Informatique et de Mathématiques Appliquées de Grenoble, www.imag.fr, Apr. 1997.

[31] Wiedemann, D. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory* IT-*32* (1986), 54–62.