On Computing Determinants of Matrices Without Divisions*

Erich Kaltofen

Department of Computer Science, Rensselaer Polytechnic Institute
Troy, New York 12180-3590; Inter-Net: kaltofen@cs.rpi.edu

An algorithm is given that computes the determinant of an $n \times n$ matrix with entries from an arbitrary commutative ring in $\widetilde{O}(n^3\sqrt{n})$ ring additions, subtractions, and multiplications; the "soft-O" $\widetilde{O}$ indicates some missing $\log n$ factors. The exponent in the running time can be reduced further by use of asymptotically fast matrix multiplication. The same results hold for computing all n^2 entries of the adjoint matrix of an $n \times n$ matrix with entries from a commutative ring.

1. Overview

1.1. Statement of Result

For any n^2, $n \geq 1$, elements $y_{i,j}$, $1 \leq i, j \leq n$, of an *arbitrary commutative ring* (with multiplicative unit), consider the determinant of the corresponding $n \times n$ matrix Y,

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

where S_n is the set of all permutations on the column indices $\{1, 2, \ldots, n\}$. Furthermore, consider the adjoint matrix of Y,

$$\mathrm{Adj}(Y) = \Big[\, (-1)^{i+j} \mathrm{Det}(Y{\downarrow}i, j) \,\Big]^{\mathrm{T}}_{1 \leq i,j \leq n},$$

where $Y{\downarrow}i, j$ is that $(n-1) \times (n-1)$ submatrix which is derived from Y by removing the i^{th} row and j^{th} column.

We present algorithms that can compute both $\mathrm{Det}(Y)$ and $\mathrm{Adj}(Y)$ in

$$O(n^3\sqrt{n}\,\log n\,\mathrm{loglog}\,n) \textit{ ring operations,}$$

i.e., $+$, $-$, and $\times$. Our method can also be speeded by using any of the asymptotically fast matrix multiplication algorithms. With matrix-multiplication exponent $\omega < 3$ we may compute determinant and adjoint in

$$O(n^{\omega/2+2}\log n\,\mathrm{loglog}\,n) \text{ ring operations.}$$

For the best-known exponent $\omega \lessapprox 2.3755$ (Coppersmith and Winograd 1990) this yields $O(n^{3.188})$ ring operations. Note that if the ring does not have a multiplicative unit, it may be embedded into its Dorroh superring, which then does (Jacobson 1974, §2.17).

Our model of computation may either be *algebraic circuits* (straight-line programs, directed acyclic computation graphs) or the more programming-oriented *algebraic random access machines*, which are formally defined in (Kaltofen 1988).

1.2. Discussion of Earlier Work

Removing divisions from programs that compute quantities that are in polynomial relation to the inputs, but which use rational functions in intermediate results, is first addressed in Strassen's (1973) seminal paper. Strassen shows that divisions do not help for the asymptotically fast matrix multiplication problem, but with his approach the determinant problem can be solved by a division-free computation of $O(n^{\omega+1}\log n\,\mathrm{loglog}\,n)$ ring operations. It is supposed here that one can multiply polynomials of degree n over the ring of entries in $O(n \log n\,\mathrm{loglog}\,n)$ ring additions, subtractions, and

*This material is based on work supported in part by the National Science Foundation under Grant No. CCR-90-06077 and under Grant No. CDA-88-05910.

In Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '92, P. S. Wang, ed., ACM Press, pp. 342–349 (1992).

multiplications. An algorithm with that complexity is, however, much more recent (Cantor and Kaltofen 1991). Strassen's result can also be derived explicitly for the determinant from the so-called exact-division version of Gaussian elimination (Sasaki and Murao 1982, §5). Note that the standard way of computing the determinant of a matrix by Gaussian elimination makes use of division by non-zero pivot elements.

Division-free computation for determinant and adjoint have played a significant rôle in estimating the parallel complexity of basic linear algebra problems, such as matrix inversion. The sequential division-free algebraic complexity then corresponds to the *work* the parallel algorithm performs, which usually is within a poly-log factor of the processor count. It is in this setting that improvements have been made. If the coefficient domain is a field with inverses for the integers $2, 3, \ldots, n$, the determinant of an $n \times n$ matrix can be computed in a little less than $O(n^{\omega+1/2})$ additions, subtractions, multiplications, and divisions by $2, 3, \ldots, n$ (Preparata and Sarwate 1978 and Galil and Pan 1989). Recently, the parallel processor count for matrix inversion has been reduced to optimal (within a log-factor), but in that algorithm divisions (and randomizations) are performed (Kaltofen and Pan 1991).

For fields of characteristic 2, say, the study of parallel methods (Berkowitz 1984; see also Chistov 1985) merely shaved of a $\log\log n$ factor from Strassen's approach. The results in this paper improve the best-known division-free complexity asymptotically by a factor of $n^{\omega/2-1}$. We note that any method for computing the determinant can be automatically converted to one for computing the adjoint while increasing the complexity by no more than a constant factor. This fact follows easily from the generic program transformation for computing all partial derivatives by Baur and Strassen (1983) (see also Linnainmaa 1976), since $\partial \operatorname{Det}(Y)/\partial y_{i,j} = (-1)^{i+j}\operatorname{Det}(Y\downarrow i, j)$.

1.3. Relevance of Result to Symbolic Computation

Computing determinant and adjoint over arbitrary commutative rings is significant for the categorical approach to computer algebra itself (Davenport and Trager 1990). Our algorithm also removes divisions from the determinant problem even when the entries are field elements. This is particularly important when manipulating matrices, their determinants, and their inverses, whose entries are *symbolic*, that is, contain parameters whose values remain unspecified. Due to exponential expression-size explosion such determinants cannot always be represented in expanded, explicit, form. A possible alternative is the implicit representation by *straight-line programs* (see Kaltofen 1988 and 1989). The straight-line representation of a determinant by a program encoding a Gaussian elimination process contains divisions, that is, it is not valid for certain parameter settings. With the results here that problem is avoided.

One can also represent determinants and inverses of symbolic matrices by so-called *black boxes* that evaluate the objects for values of their parameters (Kaltofen and Trager 1990). These black boxes can, of course, test intermediately computed elements for zero. We wish to point out, however, that such tests are not always easy. Difficulties arise, for example, when the inputs to the black box are not precise, e.g., numeric, or truncated p-adic or power series. Then with the algorithm of this paper the black box can always produce a numeric or truncated p-adic or power series result. Furthermore, zero testing can be affected by the characteristic of the coefficient field. It can be desirable to have black boxes that are oblivious to the characteristic (cf. Duval 1989). Finally, for certain function models zero-testing may become even undecidable (Richardson 1968).

1.4. Outline of Approach

In his seminal paper, Wiedemann (1986) has taken so-called Krylov subspace methods for sparse linear system manipulation one step further by reducing the arising "smaller" linear problems to the Berlekamp/Massey (1969) algorithm. As it turns out, Wiedemann's approach is applicable to abstract fields (Kaltofen and Saunders 1991, Kaltofen 1992), and the approach has yielded new insight into the problem of solving general, dense, linear systems (Kaltofen and Pan 1991). Here we combine that approach with Strassen's (1973) avoidance of divisions. The latter requires a concrete input on which an algorithm with divisions does not divide by zero. For Gaussian elimination without pivot-search this would be the identity matrix. However, for Wiedemann's determinant algorithm, which not only divides but is also randomized, the construction of this concrete input is more involved. We need a matrix and vectors for projection that produce a linearly generated sequence such that the recursion equation can be determined without ever having to divide, say, when applying the Berlekamp/Massey algorithm.

By a stroke of some luck, such a sequence can be "reversely engineered." We inspect the Berlekamp/Massey algorithm and its relation to the extended Euclidean algorithm (Dornstetter 1987), and instead of computing discrepancies, by some of which is divided, we determine the next sequence elements such that any division is by the unit 1. There are several more conditions that need to be enforced, but that is also possible. In the end, all needed constants can be defined by explicit formulas with binomial coefficients (§2). Our concrete matrix is then chosen the companion matrix of the linear generator of that sequence, and the projection vectors

i	$f_i(x)$	$t_i(x)$
0	$126 + 70x + 35x^2 + 20x^3 + 10x^4 + 6x^5 + 3x^6 + 2x^7 + x^8 + x^9$	1
1	$126 - 56x - 35x^2 - 15x^3 - 10x^4 - 4x^5 - 3x^6 - x^7 - x^8$	$-x + 1$
2	$126 + 196x - 21x^2 - 15x^3 - 5x^4 - 4x^5 - x^6 - x^7$	$-x^2 + x + 1$
3	$126 - 182x - 231x^2 + 6x^3 + 5x^4 + x^5 + x^6$	$x^3 - x^2 - 2x + 1$
4	$126 + 322x - 203x^2 - 246x^3 + x^4 + x^5$	$x^4 - x^3 - 3x^2 + 2x + 1$
5	$126 - 308x - 553x^2 + 209x^3 + 251x^4$	$-x^5 + x^4 + 4x^3 - 3x^2 - 3x + 1$
6	$\frac{7932834}{63001} + \frac{20267632}{63001}x - \frac{12688669}{63001}x^2 - \frac{15368221}{63001}x^3$	$\frac{1}{251}x^6 - \frac{209}{63001}x^5 + \frac{61955}{63001}x^4 - \frac{62416}{63001}x^3 - \frac{188124}{63001}x^2 + \frac{125877}{63001}x + \frac{62959}{63001}$

Figure 1: Special remainder sequence for $n = 5$.

are taken such that the Wiedemann algorithm produces precisely that sequence. We then employ Strassen's technique. However, additional improvements in the innards of that method, not unlike those in (Preparata and Sarwate 1978), are required to obtain the stated complexity (§3).

Over fields, our result has asymptotically at least a factor $n^{2-\omega/2}$ more operations than can be accomplished with divisions. For $\omega > 2$, the efficient solution of a plausible sub-problem (§3, Step 1 in $n^{\omega}(\log n)^{O(1)}$ ring operations) would remove that extra factor.

Used Notation

The symbols $\mathbb{Z}$ and $\mathbb{Q}$ denote the set of integers and rational numbers, respectively. In appropriate context, $\mathbb{Z}$ shall also denote the sub-ring generated by the multiplicative unit of the ring of entries. By capital letters, e.g., C, Y, Z, etc., we shall always denote matrices, while vectors are denoted by lower case letters, e.g., u and v, and which are then always column vectors. Row vectors are denote by transposition of column vectors, as in u^{T}.

2. Normal Linear Recurrences

In this section we establish that the infinite sequence of integers $a_i \in \mathbb{Z}$ for $i \geq 0$,

$$a_0 = 1, a_1 = 1, a_2 = 2, a_3 = 3, \ldots, a_i = \binom{i}{\lfloor i/2 \rfloor}, \ldots$$

has the following properties (1) and (2). For any $n \geq 1$ consider the polynomials

$$f_{-1}(x) = x^{2n},\ f_0(x) = a_0 x^{2n-1} + \cdots + a_{2n-1},$$

and their polynomial remainder/quotient sequence $f_i(x)$, $q_i(x) \in \mathbb{Q}[x]$ for $i \geq 1$,

$$f_i(x) = f_{i-2}(x) - q_i(x) f_{i-1}(x),\ \deg(f_i) < \deg(f_{i-1}).$$

Then it is true that for all $1 \leq i \leq n-1$,

$$f_i(x) = \pm x^{2n-1-i} + \text{lower order monomials}. \qquad (1)$$

It immediately follows by induction from (1) that

$$f_i(x) \in \mathbb{Z}[x] \text{ for all } -1 \leq i \leq n,$$

and that for the extended Euclidean schemes

$$s_i(x) f_{-1}(x) + t_i(x) f_0(x) = f_i(x),\ s_i(x), t_i(x) \in \mathbb{Q}(x)$$

for $i \geq 1$, we must also have for all $i \leq n$ that $s_i(x), t_i(x) \in \mathbb{Z}[x]$. Our sequence also guarantees that

$$t_n(x) = \pm x^n + \text{intermediate monomials} + 1. \qquad (2)$$

The arising polynomials for the case $n = 5$ are given in the Figure 1, including the polynomials f_{n+1} and t_{n+1}, which are not integral any more.

We shall explain the significance of properties (1) and (2), before we prove them. These stem from the way that one computes the determinant of an $n \times n$ matrix by the Wiedemann (1986) method. Briefly, that method for a matrix C with minimum = characteristic polynomial finds for random vectors u and v a linear recurrence for the sequence of domain elements

$$a_i = u^{\mathrm{T}} C^i v, \quad i \geq 0.$$

The polynomial associated with the linear recurrence turns out to be, at least with high probability, equal to the minimum polynomial of C. Thus the determinant of C is the constant coefficient of that polynomial. As it follows from the theory of linearly generated sequences (see, e.g., Kaltofen 1992), our sequence restricted to the first $2n$ elements,

$$a_0 = 1, a_1 = 1, a_2 = 2, \ldots, a_{2n-1},$$

is *linearly generated* by the polynomial

$$\pm t_n(x) = x^n - c_{n-1} x^{n-1} - \cdots - c_0 \qquad (3)$$

with

$$c_i = -(-1)^{\lfloor (n-i+1)/2 \rfloor} \binom{\lfloor (n+i)/2 \rfloor}{i}.$$

$\Lambda_0(x) \leftarrow 1; \Sigma_0(x) \leftarrow 0; l_0 \leftarrow 0; \delta \leftarrow 1;$
For $r = 1, 2, 3, \ldots$ Do {
At this point, $a_0, \ldots, a_{r-2} \in \mathbb{Z}$ are determined.
With $\Lambda_{r-1}(x) = c_{r-1,0}x^{d_{r-1}} + c_{r-1,1}x^{d_{r-1}-1} + \cdots + c_{r-1,d_{r-1}}$, $c_{r-1,0} \neq 0$, the r-th discrepancy is $\delta_r \leftarrow c_{r-1,d_{r-1}}a_{r-1} + c_{r-1,d_{r-1}-1}a_{r-2} + \cdots + c_{r-1,0}a_{r-d_{r-1}-1}$; note that always $c_{r-1,d_{r-1}} = 1$.
Note that for the coefficients in (3) we have $c_i = -c_{2n,i}$.
We now choose a_{r-1} such that (1) and (2) is true.
If $2l_{r-1} < r$ or $r = 2$ Then {Determine a_{r-1} such that $\delta_r = 1$;
this branch occurs for $r = 1, 2, 3, 5, 7, 9, \ldots$ }
Else {Determine a_{r-1} such that $\delta_r = 0$;
this branch occurs for $r = 4, 6, 8, 10, \ldots$ }
The rest of the loop body is the Berlekamp/Massey algorithm; note that always $\delta = 1$.
If $\delta_r \neq 0$ Then $\{\Lambda_r(x) \leftarrow \Lambda_{r-1}(x) - \frac{\delta_r}{\delta}x\Sigma_{r-1}(x)$;
If $2l_{r-1} < r$ Then $\{\Sigma_r(x) \leftarrow \Lambda_{r-1}(x)$; $l_r \leftarrow r - l_{r-1}$; $\delta \leftarrow \delta_r$;
this branch occurs for odd r }
Else $\{\Sigma_r(x) \leftarrow x\Sigma_{r-1}(x)$; $l_r \leftarrow l_{r-1}$;
this branch occurs for $r = 2$ }}
Else $\{\Lambda_r(x) \leftarrow \Lambda_{r-1}(x)$; $\Sigma_r(x) \leftarrow x\Sigma_{r-1}(x)$; $l_r \leftarrow l_{r-1}$;
this branch occurs for even $r \geq 4$ }}
Return $\{a_0, a_1, \ldots, a_{r-1}, \ldots\}$.

Figure 2: Algorithm to generate special sequence.

Being a linear generator means that

$$a_i = c_{n-1}a_{i-1} + c_{n-2}a_{i-2} + \cdots + c_0 a_{i-n} \quad \text{for } n \leq i < 2n.$$

If we now choose C the companion matrix of (3), i.e.,

$$C = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & 0 \\ & & \ddots & \ddots & \\ 0 & & & 0 & 1 \\ c_0 & c_1 & \ldots & c_{n-2} & c_{n-1} \end{bmatrix}, \tag{4}$$

then

$$a_i = \underbrace{[1 \quad 0 \quad 0 \quad \ldots \quad 0]}_{e_1^{\mathrm{T}}} \times C^i \times v, \quad v = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix}.$$

Therefore, if we perform Wiedemann's algorithm on C using $u = e_1$, which is the 1-st unit vector, and v as it is defined, the arising linear recurrence problem is the one described above. Solving the linear recurrence by the extended Euclidean algorithm, and subsequently making the computed characteristic polynomial monic, necessitates by the properties (1) and (2), respectively, no divisions. We will use this special case as the point of translation for Strassen's elimination of divisions scheme.

We can actually construct our sequence by exploiting the interpretation of the polynomial remainder sequence of f_{-1} and f_0 as the Berlekamp/Massey algorithm (Dornstetter 1987; see also Kaltofen 1992). Essentially, we can choose the next a_i in such a way that the discrepancy, which corresponds to the leading coefficient of the next remainder is equal to 1. This is possible because the so-called error-locator polynomials have always the constant coefficient 1. Thus one may enforce condition (1), but it does not yet guarantee condition (2). Indeed, for a polynomial remainder sequence without degree gaps, a so-called normal PRS, one more discrepancy computation completes the linear quotient. By making that discrepancy 0, the current error-locator polynomial remains untouched, thus cannot drop in degree. Therefore, throughout the Berlekamp/Massey algorithm performed on the constructed sequence, the leading (and trailing) coefficients of the error locator polynomials are ± 1. The same property must then hold also for the reverse of those polynomials, the multipliers $t_i(x)$. Since in the initial phase of the algorithm the situation is somewhat different, we give the complete construction.

A few more comments are in order. The case $r = 4, 6, 8, \ldots$ corresponds to the computation of the constant coefficient of the quotients in the polynomial remainder sequence. Since the discrepancies for $r = 1, 2, 3, 5, 7, \ldots$ are all equal to 1, the quotients are all linear, and $\Lambda_1 = 1$, $\Lambda_2 = 1 - x$, $\Lambda_{2m-1} = \Lambda_{2m}$ with $\deg(\Lambda_{2m-1}) = m$ for all $m \geq 2$. The reason for treating $r = 2$ exceptionally is so that the process gets initialized correctly. By setting $\delta_4 = \delta_6 = \cdots = 0$, one guarantees that always $c_0 = \pm 1$. This condition is essential because at the end of our application c_0 corresponds to the leading coefficient of the polynomial that is an associate of the characteristic polynomial. One also has to divide by that coefficient in order to obtain the determinant

correctly.

One of the referees for this paper discovered the explicit formulas for both a_i and $c_{2n,i}$ given earlier, which now can be derived from the algorithm given in Figure 2. The decisions in that algorithm are entirely known. In particular, $d_r = \lfloor (r+1)/2 \rfloor$ for all $r \geq 2$, and $l_r = \lfloor (r+1)/2 \rfloor$ for all $r \geq 0$, and $\delta_r = \delta = 1$ for all $r \geq 0$. Therefore, $\Lambda_r(x) = \sum_{i=0}^{d_r} c_{r,i} x^{d_r - i}$ is updated only for odd $r = 2m+1$ (and $r = 2$, which we need not consider here). In that case,

$$\begin{aligned}\Lambda_{2m+1} &= \Lambda_{2m} - x\Sigma_{2m} \\ &= \Lambda_{2m} - x \cdot x\Sigma_{2m-1} \\ &= \Lambda_{2m} - x^2 \Lambda_{2m-2}.\end{aligned}$$

Comparing the coefficient of x^j in this equation, noting that $d_{2m+1} = m+1$, $d_{2m} = m$, and $d_{2m-2} = m-1$, we obtain

$$c_{2m+1,m+1-j} = c_{2m,m-j} - c_{2m-2,m+1-j}.$$

Therefore, it can be established by induction on m, using properties of Pascal's triangle, that for all $m \geq 1$ and $0 \leq i \leq m$

$$c_{2m-1,i} = c_{2m,i} = (-1)^{\lfloor (m-i+1)/2 \rfloor} \binom{\lfloor (m+i)/2 \rfloor}{i}. \quad (5)$$

Furthermore, we have the following identities for a_r: for even $r = 2m$

$$0 = \sum_{i=0}^{m} c_{2m-1,i} a_{m+i-1}, \quad (6)$$

and for odd $r = 2m-1$

$$1 = \sum_{i=0}^{m-1} c_{2m-2,i} a_{m+i-1}. \quad (7)$$

These identities yield for all $r \geq 1$

$$a_r = \binom{r}{\lfloor r/2 \rfloor}. \quad (8)$$

Here we shall prove only that (5) and (8) yield (7); identity (6) follows similarly. Plugging in (5) and (8) for the product under the summation sign, and applying the identity

$$\binom{\mu}{\kappa}\binom{\rho}{\mu} = \binom{\rho-\kappa}{\mu-\kappa}\binom{\rho}{\kappa},$$

as well as replacing $m-1$ by m, we have for the right-hand-side of (7)

$$\sum_{i=0}^{m} (-1)^{\lfloor (m-i+1)/2 \rfloor} \binom{m}{\lfloor (m-i)/2 \rfloor} \binom{m+i}{m}. \quad (9)$$

We consider only the subcase where $m = 2k+1$ is odd: then for even i,

$$\begin{aligned}\binom{m}{\lfloor (m-i)/2 \rfloor} &= \binom{m}{\lfloor (m-(i+1))/2 \rfloor} \\ &= \binom{2k+1}{k-(i/2)} \end{aligned} \quad (10)$$

and

$$\begin{aligned}(-1)^{\lfloor (m-i+1)/2 \rfloor} &= -(-1)^{\lfloor (m-(i+1)+1)/2 \rfloor} \\ &= (-1)^{k+1}(-1)^{i/2}.\end{aligned}$$

Therefore, in (9) the two coefficents to (10) can be combined as

$$\begin{aligned}\binom{m+i}{m} - \binom{m+i+1}{m} &= -\binom{m+i}{m-1} \\ &= -\binom{2k+1+i}{2k},\end{aligned}$$

which yields for $j = k - i/2$ the following sum for (9):

$$\sum_{j=0}^{k} (-1)^j \binom{2k+1}{j} \binom{4k+1-2j}{2k}. \quad (11)$$

We may extend the summation to the full range of integers by adding 1, since

$$\binom{4k+1-2j}{2k} = \begin{cases} 0 & \text{for } k+1 \leq j \leq 2k, \\ 1 & \text{for } j = 2k+1. \end{cases}$$

If we expand the first binomial coefficient,

$$\binom{2k+1}{j} = \binom{2k}{j} + \binom{2k}{j-1},$$

then (11) is equal to

$$\begin{aligned}1 &+ \sum_{j=-\infty}^{\infty} (-1)^j \binom{2k}{j} \binom{4k+1-2j}{2k} \\ &- \sum_{j=-\infty}^{\infty} (-1)^{j-1} \binom{2k}{j-1} \binom{4k-1-2(j-1)}{2k}. \end{aligned} \quad (12)$$

However, both sums in (12) are 2^{2k}, since generally

$$\sum_{\iota=-\infty}^{\infty} (-1)^{\iota} \binom{\kappa}{\iota} \binom{\mu - \sigma\iota}{\kappa} = \sigma^{\kappa} \quad \text{for } \kappa \geq 0$$

(Graham et al. 1989, (5.53)). Therefore (12) = (11) = (9) = 1, proving (7) for odd m. The case where m is even follows similarly.

3. Division-Free Determinant Computation

The main idea of the algorithm is the application of Strassen's (1973) general method of avoiding divisions to Wiedemann's (1986) determinant algorithm. There are, of course, several obstacles that need to be overcome. For one, Wiedemann's algorithm is randomized. Also, on arbitrary matrices it requires $O(n^3 \log n)$ steps, which after elimination of divisions generally would require by a factor of $O(n \log n \operatorname{loglog} n)$ more time. And finally, there still would be divisions by elements in the algebraic closure of the coefficient field. We shall give the solutions to all these problems by outlining our algorithm right away.

Step 1: Let Y by an $n \times n$ matrix with indeterminate entries $y_{i,j}$.
For $i = 0, 1, \ldots, 2n-1$ Do
{Compute the coefficients of z^j, $0 \le j \le i$, of

$$\alpha_i(z) = e_1^{\mathrm{T}}(C + zY')^i v_0, \quad v_0 = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix},$$

where C is the companion matrix (4) of §2, $Y' = Y - C$, and a_j are the entries in the sequence of §2. Note that the computed quantities are homogeneous polynomials of degree j in the entries $y'_{i,j}$ of Y'. }

Step 2: Compute the Euclidean scheme

$$s_n(x)x^{2n} + t_n(x)(\alpha_{2n-1}x^{2n-1} + \cdots + \alpha_0) = f_n(x),$$

such that $\deg(f_n) = n-1$, with coefficients in the domain of truncated power series in z, $\mathbb{Q}(\ldots, y'_{i,j}, \ldots)[[z]]$ mod z^{n+1}. Note that the coefficients of $z^l x^k$ in the remainders f_i and the corresponding "Bezout" multipliers t_i, $1 \le i \le n$, are actually elements in $\mathbb{Z}[\ldots, y'_{i,j}, \ldots] = \mathbb{Z}[\ldots, y_{i,j}, \ldots]$, since for $z = 0$ all leading coefficients of the intermediate polynomial remainders are ± 1 (see also the remarks below). It is these elements in $\mathbb{Z}[\ldots, y'_{i,j}, \ldots]$ that are computed here.

Step 3: Make

$$t_n(x) = (\pm 1 + z \cdot (\ldots))\, x^n + \text{lower order terms}$$

monic by dividing by the truncated power series inverse of the leading coefficient. Let the constant coefficient of the resulting polynomial, which is the characteristic polynomial of $C + zY'$, be

$$\sum_{j=1}^{n} \tau_j(\ldots, y'_{i,j}, \ldots) z^j + O(z^{n+1}),$$

where we so far have computed all τ_j. The determinant of Y is then

$$\operatorname{Det}(Y) = \sum_{j=1}^{n} \tau_j(\ldots, y_{i,j} - c_{i,j}, \ldots),$$

where $c_{i,j}$ are the entries in C. Note that the translation of $y_{i,j}$ occurs in Step 1 at the beginning of the computation, so one can return the sum of the τ's as the determinant.

Before going through the complexity analysis of the algorithm, we shall give further explanation on the workings of our method. First, it is crucial to observe that for $z = 0$ the algorithm represents Wiedemann's method on the input matrix C. No randomization is necessary, since for the matrix C the minimum polynomial is equal to the characteristic polynomial, which in turn is the minimum linear generator for the sequence $\{a_0 = \alpha_0(0), a_1 = \alpha_1(0), \ldots\}$. The algorithm makes Strassen's elimination of divisions explicit using the special input C as the point at which the algorithm never divides by zero. Indeed, at that special point no division occurs at all, due to the construction in §2. The leading coefficients of all arising remainders are ± 1, and so is the leading coefficient of $t_n(x)$ (again setting $z = 0$). It may be helpful for the reader to consult the paper (Kaltofen 1988, Theorem 7.1) for an alternate description of Strassen's method.

We now analyze each step, in reverse order of occurrence. Step 3 is simply the division of a truncated power series by another, hence requires $O(n \log n \operatorname{loglog} n)$ ring operations (Sieveking 1972, Kung 1974, and Cantor and Kaltofen 1991). Step 2 can be taken care of in $O(n^3 \log n \operatorname{loglog} n)$ ring operations, since the extended Euclidean algorithm requires $O(n^2)$ domain operations, which in our case are truncated power series operations. We note that by the construction of the case $z = 0$ the remainder sequence is normal. Step 2 can be done faster, namely in $O(n^2 (\log n)^3 (\operatorname{loglog} n)^2)$ ring operations using the Knuth (1970)/Schönhage (1971) "half-gcd" approach. Divisions are then again only by the leading coefficients of the remainders, which is most readily seen from Strassen's (1983, Proof of Theorem 4.2) explicit statement of that algorithm (see also Moenck 1973 and Brent et al. 1980).

We are thus left with Step 1. Let $r = \lceil \sqrt{2n} \rceil$ and $s = \lceil 2n/r \rceil$. We compute in the given order:

Substep 1.1. For $j = 1, 2, \ldots, r-1$:
$v_j(z) = (C + zY')^j v_0$;
Substep 1.2. $Z(z) = (C + zY')^r$;
Substep 1.3. For $k = 1, 2, \ldots, s$: $u_k^{\mathrm{T}}(z) = e_1^{\mathrm{T}} Z(z)^k$;
Substep 1.4. For $j = 0, 1, \ldots, r-1$:
For $k = 0, 1, \ldots, s$:
$\alpha_{kr+j}(z) = u_k^{\mathrm{T}}(z) v_j(z)$.

Substep	Standard matrix multipl.	Fast matrix multipl.
1.1. $(C + zY')^j v_0$:	$r \cdot O(n^2) \cdot O(r)$	$\sum_{j<\log r} O(n^\omega) \cdot O(M(2^{j+1}))$
1.2. $(C + zY')^r$:	$\sum_{j<\log r} O(n^3) \cdot O(M(2^{j+1}))$	Same as in 1.1.
1.3. $e_1^{\mathrm{T}} Z(z)^k$:	$s \cdot O(n^2) \cdot M(n)$	$s \cdot O(r^2 s^\omega) \cdot M(r)$
1.4. $u_k^{\mathrm{T}}(z) v_j(z)$:	$rs \cdot O(n) \cdot M(n)$	$(r^2/s) \cdot O(s^\omega) \cdot M(n)$
Total	$O(n^3\sqrt{n}\,\log n\,\operatorname{loglog} n)$	$O(n^{\omega/2+2}\log n\,\operatorname{loglog} n)$

Figure 3: Summary of running times for Substeps of Step 1.

Note that

$$\begin{aligned} v_j(z) &\in (\mathbb{Z}[\ldots, y'_{i,j}, \ldots][z])^n, \\ Z(z) &\in (\mathbb{Z}[\ldots, y'_{i,j}, \ldots][z])^{n\times n}, \\ u_k^{\mathrm{T}}(z) &\in (\mathbb{Z}[\ldots, y'_{i,j}, \ldots][z])^{1\times n}, \\ \alpha_{kr+j}(z) &\in \mathbb{Z}[\ldots, y'_{i,j}, \ldots][z]. \end{aligned}$$

Figure 3 summarizes the cost, in terms of ring operations, of each substep. We shall give the measures for both standard $O(n^3)$ and fast $O(n^\omega)$ matrix multiplication time. The function $M(n) = O(n \log n \operatorname{loglog} n)$ measures the cost of the truncated power series arithmetic. In order to make the entries more intuitive, we decompose our running time measures, first counting the number of iterations, then the cost in terms of matrix or vector operations, and finally the cost for the individual truncated power series operation. Note that for substeps 1.1 and 1.2 the maximum degree is z^r, which is one of the reasons why things run faster.

Both entries for substep 1.2 are validated by exponentiation with repeated squaring. A similar matrix A times vector b doubling argument, sketched as

$$\begin{aligned} A^{2^i} \times \Big[\, b \;\Big|\; Ab \;\Big|\; \ldots \;\Big|\; A^{2^i-1} b \,\Big] \\ = \Big[\, A^{2^i} b \;\Big|\; A^{2^i+1} b \;\Big|\; \ldots \;\Big|\; A^{2^{i+1}-1} b \,\Big], \end{aligned}$$

justifies the second entry in substep 1.1. Aside from the second entry in substep 1.4, which is derived by blocking the arising matrix times matrix problem into $(s \times s)$-sized blocks, the only other entry using nonstandard techniques is the dominating second entry in substep 1.3. The argument for the complexity measure stated there considers the problem of computing

$$u_{k+1}(z) = Z(z)^{\mathrm{T}} \times \underbrace{\sum_{l=0}^{s} u_{k,l}(z)(z^r)^l}_{u_k(z)},$$

where the entries of the vectors $u_{k,l}(z)$ have degree less than r in z. One first computes the matrix product

$$Z(z)^{\mathrm{T}} \times \Big[\, u_{k,0}(z) \;\Big|\; u_{k,1}(z) \;\Big|\; \ldots \;\Big|\; u_{k,s}(z) \,\Big]$$

by $(s \times s)$-sized blocking. Thus, we are required to do $O(r^2)$ block-products, each block costing $O(s^\omega)$ power series operations truncated at $O(z^{2r})$. Summing the resulting vector polynomials up is dominated by that count.

Note added on June 14, 1995: Thomas H. Spencer has observed that by choosing $r = \lceil (2n)^{1-\beta} \rceil$ and $s = \lceil (2n)^\beta \rceil$ with $\beta = (\omega-2)/(\omega-1) \lessapprox 0.273$, the time with fast matrix multiplication is overall reduced to $O(n^{\omega+1-\beta})$, which with the currently best $\omega \lessapprox 2.3755$ becomes $O(n^{3.103})$. Using in Substep 1.3 a fast method for multiplying an $n \times n$ matrix by an $n \times s$ matrix in $O(n^{\omega-\gamma+o(1)} s^{\gamma+o(1)})$ arithmetic operations (by blocking the $n \times n$ matrix into $(t \times t)$-sized blocks and the $n \times s$ matrix into $(t \times t^\delta)$-sized blocks such that $n/t = s/t^\delta$ and that the individual block product only take $O(t^{2+o(1)})$ arithmetic steps each), where $\gamma = (\omega-2)/(1-\delta)$ with $\delta = 0.2946289$ (Coppersmith, private communication), the overall running time can be improved to $O(n^{\omega+1/(\gamma+1)+o(1)})$, which with $\omega \lessapprox 2.3755$ yields $O(n^{3.0281})$.

Acknowledgement: The explicit formulas for the integers a_i and c_i in §2 were pointed out to me by one of the referees. Another referee made me aware of the work by Sasaki and Murao. Furthermore, my colleague M. S. Krishnamoorthy provided the proof for (9) = 1. I am grateful for all these contributions.

Literature Cited

Baur, W. and Strassen, V., "The complexity of partial derivatives," *Theoretical Comp. Sci.* **22**, pp. 317–330 (1983).

Berkowitz, S. J., "On computing the determinant in small parallel time using a small number of processors," *Inform. Process. Letters* **18**, pp. 147–150 (1984).

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**, pp. 259–295 (1980).

Cantor, D. G. and Kaltofen, E., "On fast multiplication of polynomials over arbitrary rings," *Acta Inform.* **28**/7, pp. 693–701 (1991).

Chistov, A. L., "Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic," *Proc. FCT '85, Springer Lec. Notes Comp. Sci.* **199**, pp. 63–69 (1985).

Coppersmith, D. and Winograd, S., "Matrix multiplication via arithmetic progressions," *J. Symbolic Comput.* **9**/3, pp. 251–280 (1990).

Davenport, J. H. and Trager, B. M., "Scratchpad's view of algebra I: basic commutative algebra," in *Design and Implementation of Symbolic Computation Systems*, Springer Lect. Notes Comput. Sci. **429**, edited by A. Miola; pp. 40–54, 1988.

Dornstetter, J. L., "On the equivalence between Berlekamp's and Euclid's algorithms," *IEEE Trans. Inf. Theory* IT-**33**/3, pp. 428–431 (1987).

Duval, D., "Simultaneous computations in fields of different characteristics," in *Computers and Mathematics*, edited by E. Kaltofen and S. M. Watt; Springer-Verlag, New York, pp. 321–326, 1989.

Galil, Z. and Pan, V., "Parallel evaluation of the determinant and of the inverse of a matrix," *Inform. Process. Letters* **30**, pp. 41–45 (1989).

Graham, R. L., Knuth, D. E., and Patashnik, O., *Concrete Mathematics*; Addison-Wesley Publ. Comp., Reading, Massachusetts, 1989.

Jacobson, N., *Basic Algebra I*; W. H. Freeman & Co., San Francisco, 1974.

Kaltofen, E., "Greatest common divisors of polynomials given by straight-line programs," *J. ACM* **35**/1, pp. 231–264 (1988).

Kaltofen, E., "Factorization of polynomials given by straight-line programs," in *Randomness and Computation*, Advances in Computing Research **5**, edited by S. Micali; JAI Press, Greenwhich, Connecticut, pp. 375–412, 1989.

Kaltofen, E., "Efficient Solution of Sparse Linear Systems," *Lect. Notes*, Dept. Comput. Sci., Rensselaer Polytech. Inst., Troy, New York, 1992.

Kaltofen, E. and Pan, V., "Processor efficient parallel solution of linear systems over an abstract field," in *Proc. 3rd Ann. ACM Symp. Parallel Algor. Architecture*; ACM Press, pp. 180–191, 1991.

Kaltofen, E. and Saunders, B. D., "On Wiedemann's method of solving sparse linear systems," in *Proc. AAECC-5*, Springer Lect. Notes Comput. Sci. **536**; pp. 29–38, 1991.

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, pp. 301–320 (1990).

Knuth, D. E., "The analysis of algorithms," *Actes du congrès international des Mathématiciens* **3**, pp. 269–274 (1970).

Kung, H. T., "On computing reciprocals of power series," *Numer. Math.* **22**, pp. 341–348 (1974).

Linnainmaa, S., "Taylor expansion of the accumulated rounding error," *BIT* **16**, pp. 146–160 (1976).

Massey, J. L., "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory* **IT-15**, pp. 122–127 (1969).

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

Preparata, F. P. and Sarwate, D. V., "An improved parallel processor bound in fast matrix inversion," *Inform. Process. Letters* **7**/3, pp. 148–150 (1978).

Richardson, D., "Some undecidable problems involving elementary functions of a real variable," *J. Symbolic Logic* **33**/4, pp. 511–520 (1968).

Sasaki, T. and Murao, H., "Efficient Gaussian elimination method for symbolic determinants and linear systems," *ACM Trans. Math. Software* **8**/3, pp. 277–289 (1982).

Schönhage, A., "Schnelle Kettenbruchentwicklungen," *Acta Inform.* **1**, pp. 139–144 (1971). In German.

Sieveking, M., "An algorithm for division of power series," *Computing* **10**, pp. 153–156 (1972).

Strassen, V., "Vermeidung von Divisionen," *J. reine u. angew. Math.* **264**, pp. 182–202 (1973). In German.

Strassen, V., "The computational complexity of continued fractions," *SIAM J. Comput.* **12**/1, pp. 1–27 (1983).

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