Division-Free Algorithms for the Determinant and the Pfaffian: Algebraic and Combinatorial Approaches

Günter Rote

Institut für Informatik, Freie Universität Berlin, Takustraße 9, D-14195 Berlin
rote@inf.fu-berlin.de

1 Introduction

The most common algorithm for computing the determinant of an $n \times n$ matrix A is Gaussian elimination and needs $O(n^3)$ additions, subtractions, multiplications, and divisions. On the other hand, the explicit definition of the determinant as the sum of $n!$ products,

$$\det A = \sum_{\pi} \operatorname{sign} \pi \cdot a_{1\pi(1)} a_{2\pi(2)} \dots a_{n\pi(n)} \tag{1}$$

shows that the determinant can be computed *without* divisions. The summation is taken over the set of all permutations π of n elements. Avoiding divisions seems attractive when working over a commutative ring which is not a field, for example when the entries are integers, polynomials, or rational or even more complicated expressions. Such determinants arise in combinatorial problems, see [11].

We will describe an $O(n^4)$ algorithm that works without divisions, and we will look at this algorithm from a combinatorial and an algebraic viewpoint.

We will also consider the *Pfaffian* of a skew-symmetric matrix, a quantity closely related to the determinant. The results are in many ways analogous to those for the determinant, but the algebraic aspect of the algorithms is not explored yet.

This survey is yet another article which highlights the close connection between linear algebra and cycles, paths, and matchings in graphs [1,17,25]. Much of this material is based on the papers of Mahajan and Vinay [14,15] and of Mahajan, Subramanya, and Vinay [13].

2 Algorithms for the Determinant

2.1 Alternate Expressions for the Determinant

A direct evaluation of the expression (1) requires $n!(n-1)$ multiplications (plus a smaller number of additions and subtractions, which will be ignored in the sequel). One can try to rewrite this expression and factorize it in a clever way such that common subexpressions need to be evaluated only once. Expanding one row or columns leads to a recursive procedure which takes $T(n) = n \cdot (1 + T(n-1)) \approx n!(e-1)$ multiplications. Splitting the matrix into two equal parts and applying Laplace expansion needs $B(n) = \binom{n}{\lfloor n/2 \rfloor}(1 + B(\lfloor n/2 \rfloor) + B(\lceil n/2 \rceil))$ multiplications. This is between 2^n (for $n > 3$) and 2^{2n}, which is still exponential.

It will turn out that one has to first *extend* the expression by additional redundant terms. Only this will allow to *reduce* the time to evaluate the expression.

2.2 Cycle Decompositions

We first rewrite the product $a_{1\pi(1)}a_{2\pi(2)} \dots a_{n\pi(n)}$ in terms of the *cycle structure* of the permutation π. Every permutation corresponds to a partition of the set $\{1, \dots, n\}$ into disjoint cycles. For example, the cycle decomposition

$$\mathcal{C} = (8\ 7\ 4)(6)(5\ 3\ 1\ 2) = (\underline{1}\ 2\ 5\ 3)(\underline{4}\ 8\ 7)(\underline{6}) \tag{2}$$

corresponds to the following permutation.

$$\pi = \begin{pmatrix} i: & 1\,2\,3\,4\,5\,6\,7\,8 \\ \pi(i): & 2\,5\,1\,8\,3\,6\,4\,7 \end{pmatrix}$$

We can get a canonical form of a cycle decomposition C by letting each cycle start at its smallest element, which is called the *head* of the cycle and is underlined in (2), and by rearranging the cycles in increasing order of their heads. Formally, a *cycle* $(c_1, c_2, \dots, c_i)$ of length i can be defined as a sequence of $i \geq 1$ *distinct* elements from the ground set $\{1, \dots, n\}$, where c_1 is the smallest element and is called the *head* of C. The *weight* of C is $\text{weight}(C) = a_{c_1c_2}a_{c_2c_3} \dots a_{c_ic_1}$. This is the product of the weight of its arcs if C is regarded as a cycle in a graph.

A *cycle decomposition* $\mathcal{C}$ is a sequence $C_1, \dots, C_k$ of *disjoint* cycles with increasing heads, covering every element $1, \dots, n$ exactly once. The *weight* of $\mathcal{C}$ is the product of the weights of its cycles, and the *sign* of a cycle decomposition $\mathcal{C}$ with k cycles is defined as

$$\operatorname{sign}\mathcal{C} = \operatorname{sign}\pi := (-1)^{n+k}.$$

One can check that this coincides with the usual definition of the sign and parity of a permutation.

So we can rewrite (1) as follows:

$$\det A = \sum_{\text{cycle decompositions } \mathcal{C}} \operatorname{sign}\mathcal{C} \cdot \text{weight}(\mathcal{C}) \tag{3}$$

2.3 Clow Sequences

If we relax the requirement that all elements of a cycle are distinct, we arrive at the concept of a *closed ordered walk* or *clow* for short, in the terminology of Mahajan and Vinay [14]. A *clow* $(c_1, c_2, \dots, c_i)$ of length i is a sequence of $i \geq 1$

numbers where the *head* c_1 of C is the unique smallest element: $c_1 < c_2, \dots, c_i$. The *weight* of C is again $\text{weight}(C) = a_{c_1c_2}a_{c_2c_3}\dots a_{c_ic_1}$. (Valiant [24], who introduced this concept, called this a c_1-loop.)

A *clow sequence* $\mathcal{C}$ is a sequence $C_1, \dots, C_k$ of clows with a strictly increasing sequence of heads:

$$\mathcal{C} = (\underline{c_1}, \dots), (\underline{c_2}, \dots), \dots, (\underline{c_k}, \dots), \tag{4}$$

with $c_1 < c_2 < \dots < c_k$. The weight of $\mathcal{C}$ is the product of the weights of its clows, its length is the sum of the lengths of the clows, and the sign of $\mathcal{C}$ is just $(-1)^{n+k}$. It is clear that a cycle decomposition is a special case of a clow sequence for which the notions of weight and sign coincide with the one given above.

We can replace the summation over the set of cycle decompositions in formula (3) by the summation over the larger class of clow sequences.

Theorem 1.

$$\det A = \sum_{\text{clow sequences } \mathcal{C} \text{ of length } n} \operatorname{sign} \mathcal{C} \cdot \text{weight}(\mathcal{C}) \tag{5}$$

Proof. It must be shown that all “bad” clow sequences which are *not* cycle decompositions cancel because the corresponding weights occur equally often with a positive and a negative sign. This is done by pairing up the bad clow sequences such that clow sequences which are matched have opposite signs. This proof technique is typical of many combinatorial proof for theorems in linear algebra [20,18,23,25]. The following lemma is stated in Valiant [24]; its proof is given in full detail in [14,15].

Lemma 1. *Let $\mathcal{C}$ be a clow sequence that contains repeated elements. Then we can find a clow sequence $\bar{\mathcal{C}}$ with the same weight and opposite sign. Moreover, this mapping between $\mathcal{C}$ and $\bar{\mathcal{C}}$ is an involution, i. e., if we construct the corresponding other clow sequence for $\bar{\mathcal{C}}$, we obtain $\mathcal{C}$ again.*

Proof. Let $\mathcal{C} = C_1, \dots, C_k$. We successively add the clows C_k, C_{k-1}, $\dots$ until a repetition occurs. Assume that $C_{m+1}, \dots, C_k$ is a set of disjoint cycles but $C_m, C_{m+1}, \dots, C_k$ contains a repeated element, where $1 \le m \le k$. Let $C_m = (c_1, \dots, c_i)$, and let c_j be the first element in this clow which is either (A) equal to a previous element c_ℓ in the clow $(1 < \ell < j)$ or (B) equal to an element in one of the cycles $C_{m+1}, \dots, C_k$. Precisely one of these two cases will occur.

In case (A), we remove the cycle $(c_\ell{=}c_j, c_{\ell+1}, \dots, c_{j-1})$ from the clow C_m and turn it into a separate cycle, which we insert it into $\mathcal{C}$ in the proper position after cyclically shifting its head to the front. In case (B), we insert the cycle into the clow C_m after the element c_j. One easily checks that the operations in cases (a) and (b) are inverses of each other; they change the number of clows by one, and hence they invert the sign. This concludes the proof of the lemma and of Theorem 1. □

2.4 Dynamic Programming According to Length

In some sense, (1) and (3) are the most economical formulas for the determinant because all superfluous terms have been canceled and no further reduction is possible. However, we shall now see that the redundant form (5) is more amenable to an efficient algorithm. The idea is to construct clow sequences incrementally, arc by arc: We use dynamic programming to systematically compute the following quantities:

> $[l, c, c_0, s]$ is the sum of all *partial clow sequences* with length l, ending in the current vertex c, where the head of the current clow is c_0, and the sign $s = \pm 1$ gives the parity of the number of finished clows so far.

A partial clow sequence is an initial piece of the sequence given in (4): It consists of a number of completed clows, and one incomplete clow. For completing the clow sequence, we only have to know the head and the current last vertex of the incomplete clow. In the end, we need to know the parity of the total number of clows; and therefore we have to store $(-1)^k$, where k is the number of completed clows so far.

A dynamic programming algorithm can best be described by a dynamic programming graph. Our graph has $O(n^3)$ nodes corresponding to the quantities $[l, c, c_0, s]$, with $1 \le l \le n$, $1 \le c_0 \le c \le n$, and $s = \pm 1$.

A partial clow sequence can be grown either by extending the current clow or by closing the current clow and starting a new one. Correspondingly, the dynamic programming graph has two kinds of outgoing arcs from the node $[l, c, c_0, s]$: arcs to nodes $[l + 1, c', c_0, s]$ of cost $a_{cc'}$, for all $c' > c_0$, and arcs of cost a_{cc_0}, to all nodes $[l + 1, c'_0, c'_0, -s]$, for all $c'_0 > c_0$. The latter type of arcs will start a new clow with head c'_0 and no edges yet. We consider all nodes $[0, c_0, c_0, 1]$ as start nodes and add two artificial end nodes $[n, n + 1, n + 1, s]$ with $s = \pm 1$. The sum of all path lengths from start nodes to end nodes, taken with appropriate sign according to s, is the desired quantity in (5), i. e., the determinant.

The sum of all path weights in this acyclic graph can be computed in time proportional to the number of arcs of this graph, which is $O(n^4)$: there are $O(n)$ arcs leaving each of $O(n^3)$ nodes. For example, one can calculate node values in the order or increasing l values. The storage requirement in this case is $O(n^2)$ because only the nodes with two adjacent l values need to be stored at any time.

Mahajan and Vinay [14] used this algorithm to show that the computation of the determinant belongs to the complexity class GapL, a complexity class in which summation over path weights in an acyclic graphs can be done, but, very loosely speaking, subtraction is allowed only once at the very end. One can simplify the algorithm by collapsing corresponding nodes $[l, c, c_0, +1]$ and $[l, c, c_0, -1]$ with opposite sign fields into one node $[l, c, c_0]$ and introducing "negative" arc weights: the second type of arcs have then cost $-a_{cc_0}$ instead of a_{cc_0}.

2.5 Adding a Vertex at a Time: Combinatorial Approach

The previous algorithm considers all partial clow sequences in order of increasing length l. A different factorization is possible: we can first consider the clow with head $c_1 = 1$, if such a clow is present, and the remaining clow sequence, which is a sequence on the ground set $\{2, \dots, n\}$. For expressing the clows with head 1 we partition the matrix A into its first row r, its first column s, and the remaining matrix M.

$$A = \left(\begin{array}{c|c} a_{11} & r \\ \hline s & M \end{array}\right)$$

The sum of all clows of length l can be written as follows:

$$\begin{array}{ll} \text{length } l = 0\text{:} & 1 \\ \text{length } l = 1\text{:} & a_{11} \\ \text{length } l = 2\text{:} & \sum_{i=2}^{n} a_{1i}a_{i1} = rs \\ \text{length } l = 3\text{:} & \sum_{i=2}^{n}\sum_{j=2}^{n} a_{1i}a_{ij}a_{j1} = rMs \\ \vdots & \\ \text{length } l\text{:} & rM^{l-2}s \end{array}$$

The term 1 for length $l = 0$ corresponds to the "empty" clow consisting of no edges. It accounts for the case when the head of the first clow is larger than 1. We denote by q'_i the sum of signed weights of all clow-sequences of length $n-1-i$ which *don't contain* the element 1. Here we define the sign of a clow sequence with k clows as $(-1)^{n-1+k}$. Then we can write:

$$\det A = -1 \cdot q'_{-1} + a_{11} \cdot q'_0 + rs \cdot q'_1 + \cdots + rM^{i-1}s \cdot q'_i + \cdots \qquad (6)$$

The minus sign accounts for the change of the sign convention when going from q'_{-1} over a ground set of $n-1$ elements to the original ground set with n elements. In the other terms this minus sign is canceled because we have one additional clow.

By Lemma 1, the quantity q'_{-1} must be zero: q'_{-1} is the signed sum of all clow sequences of length n on the ground set $\{2, \dots, n\}$. Such sequences must contain repeated elements and hence all terms cancel. So we get

$$\det A = a_{11} \cdot q'_0 + rs \cdot q'_1 + \cdots + rM^{i-1}s \cdot q'_i + \cdots + rM^{n-2}s \cdot q'_{n-1}.$$

We intend to compute the values q'_i recursively, so we need to consider clow sequences of all possible lengths. Let us denote by q_i the sum of signed weights of all clow-sequences of length $n - i$ (with the original definition of the sign). So $q_0 = \det A$, and $q'_0 = \det M$. By the above argument, we get the following formula for computing $q_0, q_1, q_2, \dots, q_n$ from the sequence $q'_0, q'_1, \dots, q'_{n-1}$.

$$\begin{pmatrix} q_n \\ q_{n-1} \\ q_{n-2} \\ \vdots \\ q_2 \\ q_1 \\ q_0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & \cdots & 0 & 0 & 0 \\ a_{11} & -1 & \cdots & 0 & 0 & 0 \\ rs & a_{11} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ rM^{n-4}s & rM^{n-5}s & \cdots & a_{11} & -1 & 0 \\ rM^{n-3}s & rM^{n-4}s & \cdots & rs & a_{11} & -1 \\ rM^{n-2}s & rM^{n-3}s & \cdots & rMs & rs & a_{11} \end{pmatrix} \begin{pmatrix} q'_{n-1} \\ q'_{n-2} \\ \vdots \\ q'_2 \\ q'_1 \\ q'_0 \end{pmatrix} \qquad (7)$$

This matrix multiplication is nothing but a convolution between the sequence $q'_0, q'_1, \ldots$ and the sequence $-1, a_{11}, rs, rMs, \ldots$, which can most conveniently be expressed as a polynomial multiplication:

$$\begin{aligned} &q_n\lambda^n + q_{n-1}\lambda^{n-1} + \cdots + q_1\lambda + q_0 \\ &\quad = (-\lambda + a_{11} + rs\lambda^{-1} + rMs\lambda^{-2} + \cdots + rM^is\lambda^{-i-1} + \cdots) \\ &\qquad \times (q'_{n-1}\lambda^{n-1} + q'_{n-2}\lambda^{n-2} + \cdots + q'_1\lambda + q'_0) \qquad (8) \end{aligned}$$

Actually, these are not polynomials, but something like Laurent series in λ^{-1}. In explicit form, this means

$$\begin{pmatrix} q_n \\ q_{n-1} \\ q_{n-2} \\ \vdots \\ q_2 \\ q_1 \\ q_0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} -1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots \\ a_{11} & -1 & \cdots & 0 & 0 & 0 & 0 & \cdots \\ rs & a_{11} & \cdots & 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \\ rM^{n-4}s & rM^{n-5}s & \cdots & -1 & 0 & 0 & 0 & \cdots \\ rM^{n-3}s & rM^{n-4}s & \cdots & a_{11} & -1 & 0 & 0 & \cdots \\ rM^{n-2}s & rM^{n-3}s & \cdots & rs & a_{11} & -1 & 0 & \cdots \\ rM^{n-1}s & rM^{n-2}s & \cdots & rMs & rs & a_{11} & -1 & \cdots \\ rM^{n}s & rM^{n-1}s & \cdots & rM^2s & rMs & rs & a_{11} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \begin{pmatrix} q'_{n-1} \\ q'_{n-2} \\ \vdots \\ q'_1 \\ q'_0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

This is an extended version of (7) which reveals the "hidden parts" of (8). For example, the line corresponding to $q_{-1} = 0$ yields the relation

$$0 = rs \cdot q'_0 + rMs \cdot q'_1 + \cdots + rM^is \cdot q'_i + \cdots + rM^{n-1}s \cdot q'_{n-1},$$

which may be an interesting identity but is of little use for computing anything.

Mahajan and Vinay [14] have analyzed the recursive algorithm based on (7) more carefully, and they have shown that the quantities q_i, after omitting irrelevant terms like q'_{-1} from (6), actually represent a subset of the clow-sequences that are characterized by the so-called prefix property.

2.6 The Characteristic Polynomial

We have associated the polynomial

$$P_A(\lambda) = q_n\lambda^n + q_{n-1}\lambda^{n-1} + \cdots + q_1\lambda + q_0$$

to the matrix A, and we have shown how $P_A(\lambda)$ can be computed from the corresponding polynomial $P_M(\lambda)$ of the $(n-1)\times(n-1)$ submatrix M:

$$P_A(\lambda) = (-\lambda + a_{11} + rs\lambda^{-1} + rMs\lambda^{-2} + \cdots + rM^is\lambda^{-i-1} + \cdots) \cdot P_M(\lambda) \quad (9)$$

This allows us to inductively compute polynomials for larger and larger submatrices, starting from $P_{(a_{nn})}(\lambda) = a_{nn} - \lambda$, until we finally obtain $P_A(\lambda)$ and the determinant $\det A = q_0$.

We have seen that the inductive approach of adding a vertex at a time naturally leads us to consider not only clow sequences of length n as in (5), but also clow sequences of all other lengths. We have also found it convenient to use the corresponding sums of weights as coefficients of a polynomial $P_A(\lambda)$. It turns out that this polynomial is nothing but the *characteristic polynomial* of the matrix A:

$$P_A(\lambda) := \det(A - \lambda I) = q_n\lambda^n + q_{n-1}\lambda^{n-1} + \cdots + q_1\lambda + q_0 \quad (10)$$

The following well-known formula generalizes formula (3) for the determinant to all coefficients of the characteristic polynomial.

$$q_i = \sum_{\text{families } \mathcal{C} \text{ of disjoint cycles with length } n-i} \operatorname{sign}\mathcal{C} \cdot \operatorname{weight}(\mathcal{C}) \quad (11)$$

Here we consider not partitions into cycles, but families $\mathcal{C}$ of disjoint cycles with a total of $n-i$ elements. The sign of such a family with k cycles is defined as $(-1)^{n-i+k}$. Formula (11) comes from the fact that a cycle family which covers $n-i$ vertices leaves i vertices uncovered, at which one can place the diagonal entries $-\lambda$ when the determinant of $A - \lambda I$ is expanded.

The following theorem states that the quantities q_i defined in the previous section are indeed the coefficients of the characteristic polynomial (10) as we know it. It is proved in the same way as Theorem 1, see [14].

Theorem 2. *The coefficients q_i $(i = 0, 1, \ldots, n)$ of the characteristic polynomial of A can be expressed as*

$$q_i = \sum_{\text{clow sequences } \mathcal{C} \text{ of length } n-i} \operatorname{sign}\mathcal{C} \cdot \operatorname{weight}(\mathcal{C}), \quad (12)$$

where the sign of a clow sequence of length $n-i$ with k cycles is defined as $(-1)^{n-i+k}$. □

The dynamic programming algorithm of Section 2.4 can easily be extended to compute all coefficients q_i: Simply adding artificial terminal nodes $[n-i, n+1, n+1]$, with arcs of the correct sign, will do the job.

2.7 Adding a Vertex at a Time: Algebraic Approach

We will now give an algebraic proof of (7). The determinant of a bordered matrix

$$A = \left(\begin{array}{c|c} a_{11} & r \\ \hline s & M \end{array}\right) \quad (13)$$

can be expressed as

$$\det A = \det M \cdot (a_{11} - rM^{-1}s) \tag{14}$$

if the matrix M is invertible. This follows from the fact that $\det M$ is the cofactor of a_{11}, and hence $(A^{-1})_{11} = \det M / \det A$. The element $(1,1)$ of the inverse A^{-1} can be determined by block Gauss-Jordan elimination of (13), starting with the lower right block M. This gives $(A^{-1})_{11} = (a_{11} - rM^{-1}s)^{-1}$ which leads to (14).

Now we wish to express $P_A(\lambda) = \det(A - \lambda I)$ in terms of $P_M(\lambda) = \det(M - \lambda I)$. Applying (14) to the matrix $A - \lambda I$ gives

$$\begin{aligned} P_A(\lambda) = |A - \lambda I| &= \begin{vmatrix} a_{11} - \lambda & r \\ s & M - \lambda I \end{vmatrix} \\ &= |M - \lambda I| \cdot (a_{11} - \lambda - r(M - \lambda I)^{-1}s) \\ &= P_M(\lambda) \cdot (a_{nn} - \lambda + r(\lambda I - M)^{-1}s) =: P_M(\lambda) \cdot F(\lambda) \end{aligned}$$

For simplifying the expression $F(\lambda)$, which is the multiplier which turns P_M into P_A, we rewrite $(\lambda I - M)^{-1}$ as follows.

$$\begin{aligned} (\lambda I - M)^{-1} &= \lambda^{-1}(I - \lambda^{-1}M)^{-1} \\ &= \lambda^{-1}(I + \lambda^{-1}M + \lambda^{-2}M^2 + \cdots) \\ &= \lambda^{-1}I + \lambda^{-2}M + \lambda^{-3}M^2 + \cdots \end{aligned}$$

This gives

$$\begin{aligned} F(\lambda) &= a_{11} - \lambda + r(\lambda I - M)^{-1}s \\ &= -\lambda + a_{11} + rs\lambda^{-1} + rMs\lambda^{-2} + rM^2s\lambda^{-3} + rM^3s\lambda^{-4} + \cdots, \end{aligned} \tag{15}$$

which is the same factor as in (8) and (9).

Another proof of (7), based on the Cayley-Hamilton Theorem, is due to P. Beame, see [3].

2.8 Who Invented the Incremental Algorithm?

Berkowitz [3] used the algorithm based on the recursion (7) to derive a parallel procedure for evaluating the determinant. He called this algorithm *Samuelson's method*, giving a reference to Samuelson's original paper [19] and to the textbook of Faddeev and Faddeeva [8]. Valiant [24], who gave the first combinatorial interpretation of (7), referred to the algorithm as the Samuelson-Berkowitz algorithm. However, Samuelson's algorithm for computing the characteristic polynomial is an $O(n^3)$ algorithm that uses divisions. This is also the way in which it is described in [8, §45]. So the above algorithm can definitely not be attributed to Samuelson. However, a passage towards the end of Section §45 in [8] contains the system (7). Faddeyev and Faddeyeva state this as a possible stricter justification of Samuelson's method. The mention briefly that this formula follows from the consideration of bordered determinants. This calculation has been sketched above in Section 2.7.

The fact that (7) appears in the section about Samuelson's method in the book [8] may be the reason for the erroneous attribution. It seems that the method ought to be credited to Faddeyev and Faddeyeva. However, being numerical analysts, they would probably not have considered the implied $O(n^4)$ procedure as a good "method". Berkowitz [3] was apparently the first one to use the recursive formula (7) algorithmically.

3 The Pfaffian

The determinant of a skew-symmetric matrix ($a_{ij} = -a_{ji}$) is the square of another expression, which is called the *Pfaffian* [12,16]. For example,

$$\begin{vmatrix} 0 & a_{12} & a_{13} & a_{14} \\ -a_{12} & 0 & a_{23} & a_{24} \\ -a_{13} & -a_{23} & 0 & a_{34} \\ -a_{14} & -a_{24} & -a_{34} & 0 \end{vmatrix} = (a_{12}a_{34} - a_{13}a_{24} + a_{14}a_{23})^2$$

The determinant of a skew-symmetric matrix of odd order is always 0. Formally, the Pfaffian of a skew-symmetric matrix A with an even number of rows and columns can be defined as follows:

$$\operatorname{pf} A = \sum_{\text{perfect matchings } \mathcal{M}} \operatorname{sign} \mathcal{M} \cdot \operatorname{weight}(\mathcal{M}) \tag{16}$$

Here, a perfect matching $\mathcal{M}$ with $k = n/2$ edges is written as

$$\mathcal{M} = (\underline{i_1}, j_1), (\underline{i_2}, j_2), \dots, (\underline{i_k}, j_k), \tag{17}$$

where, by convention, $i_l < j_l$ for each l. The *sign* of the matching $\mathcal{M}$ is defined as the sign of

$$\begin{pmatrix} 1 & 2 & 3 & 4 & \dots & n-1 & n \\ i_1 & j_1 & i_2 & j_2 & \dots & i_k & j_k \end{pmatrix}$$

when this is regarded as a permutation of $\{1, \dots, n\}$. The weight of $\mathcal{M}$ is $a_{i_1 j_1} a_{i_2 j_2} \dots a_{i_k j_k}$. One can check that the sign of $\mathcal{M}$ does not depend on the order in which the edges are given. Moreover, exchanging a pair i_l, j_l changes the sign but not the signed weight since $a_{ji} = -a_{ij}$. This means that we are in fact free to choose any convenient permutation π as a representation of a given matching $\mathcal{M}$.

Because of their close connection with matchings, Pfaffians are of interest in combinatorics, see [12,21]. For example, for certain graphs, including planar graphs, it is possible to count the number of perfect matchings in polynomial time by using Pfaffians. Knuth [10] gives a short history of Pfaffians, and argues that Pfaffians are in some way more fundamental than determinants, to which they are closely related (see Section 4).

The Pfaffian can be computed in $O(n^3)$ steps by an elimination procedure that is similar to Gaussian elimination for symmetric matrices. Alternatively, the square root of the determinant gives the Pfaffian up to the correct sign. Both approaches may be undesirable if divisions or square roots should be avoided.

3.1 Alternating Cycle Decompositions

The combinatorial approaches to the determinant of Section 2 can be applied to Pfaffians as well. We consider the product of the Pfaffians of two skew-symmetric matrices A and B. Expanding the definitions (16) leads to an overlay of two matchings $\mathcal{M}_A$ and $\mathcal{M}_B$. The union of $\mathcal{M}_A$ with $\mathcal{M}_B$ decomposes into a disjoint union of *alternating cycles* with a total of n edges:

$$\mathcal{M}_A \uplus \mathcal{M}_B = \mathcal{C} = C_1 \cup C_2 \cup \cdots$$

An alternating cycle is a cycle of even length whose edges alternate between $\mathcal{M}_A$ and $\mathcal{M}_B$. Edges which are contained both in $\mathcal{M}_A$ and $\mathcal{M}_B$ must be kept separate the union $\mathcal{M}_A \uplus \mathcal{M}_B$; for each "common" edge, $\mathcal{M}_A \uplus \mathcal{M}_B$ contains two (distinguishable) parallel edges, forming an alternating cycle of length 2.

An alternating cycle can be traversed in two orientations. We specify the orientation by taking the vertex with the smallest number as the "first vertex" or *head* of the cycle and insisting that the first edge should belong to M_A.

Taking the union of two matchings brings us back into the realm of cycle decompositions which we studied in Section 2. Now, the weight of an alternating cycle $C = (c_1, c_2, \ldots, c_i)$ with $c_1 < c_2, \ldots, c_i$ is

$$\text{weight}(C) = a_{c_1c_2} b_{c_2c_3} a_{c_3c_4} b_{c_4c_5} \ldots a_{c_{i-1}c_i} b_{c_ic_1}. \tag{18}$$

As in Section 2.2, an *alternating cycle family* $\mathcal{C}$ is a set of disjoint alternating cycles. Its *length* is the total number of elements in its cycles. If the length is n, we speak of an alternating cycle *decomposition.* The *weight* of $\mathcal{C}$ is the product of the weights of its cycles, and the the sign of an alternating cycle decomposition $\mathcal{C}$ with k cycles is defined as $\operatorname{sign} \mathcal{C} = (-1)^k$.

Theorem 3.

$$\operatorname{pf} A \cdot \operatorname{pf} B = \sum_{\text{alternating cycle decompositions } \mathcal{C}} \operatorname{sign} \mathcal{C} \cdot \text{weight}(\mathcal{C})$$

Proof. The Pfaffian has been defined as a weighted sum of matchings; so the left side is a sum over all *pairs* of matchings $(\mathcal{M}_A, \mathcal{M}_B)$, where $\mathcal{M}_A$ is regarded as a matching for A and $\mathcal{M}_B$ is regarded as a matching for B. Now, there is a unique correspondence between pairs of matchings and alternating cycle decompositions. The only thing that has to be checked is that the signs are correct. Suppose that $\mathcal{M}_A \uplus \mathcal{M}_B$ decomposes into k disjoint cycles $\mathcal{C} = (c_1, c_2, \ldots, c_i), (d_1, d_2, \ldots, d_j), \ldots$. We may conveniently represent $\mathcal{M}_A$ and $\mathcal{M}_B$ by the permutations

$$\pi_A = \left(\begin{array}{ccccccc|cccc|cc} 1 & 2 & 3 & 4 & \cdots & i-1 & i & i+1 & i+2 & \cdots & i+j & i+j+1 \cdots & n \\ c_1 & c_2 & c_3 & c_4 & \cdots & c_{i-1} & c_i & d_1 & d_2 & \cdots & d_j & & \cdots \end{array}\right)$$

and

$$\pi_B = \left(\begin{array}{cccccc|cccc|ccc} 1 & 2 & 3 & 4 & \cdots & i-1 & i & i+1 & i+2 & \cdots & i+j & i+j+1 & \cdots & n \\ c_2 & c_3 & c_4 & c_5 & \cdots & c_i & c_1 & d_2 & d_3 & \cdots & d_1 & & \cdots & \end{array}\right),$$

respectively. This gives $\operatorname{sign} \mathcal{M}_A \cdot \operatorname{sign} \mathcal{M}_B = \operatorname{sign} \pi_A \cdot \operatorname{sign} \pi_B = \operatorname{sign}((\pi_A)^{-1} \circ \pi_B) = (-1)^k$, since the permutation $(\pi_A)^{-1} \circ \pi_B$ decomposes into k even-length cycles. □

As an easy consequence, we obtain the relation between the determinant and the Pfaffian that was mentioned at the beginning of this section.

Corollary 1.

$$\det A = (\operatorname{pf} A)^2$$

Proof. When we set $A = B$ in Theorem 3, we almost get the determinant formula (3): the difference between alternating cycle decompositions and cycle decompositions disappears except for the requirement that an alternating cycle must alway have even length.

Changing the orientation of an odd cycle in the expansion of the determinant of a skew-symmetric matrix reverses the weight to its negative. Thus it is easy to establish a sign-reversing bijection among all cycle decompositions containing odd cycles, for example by reversing the odd cycle with the largest head. So the cycle decompositions containing odd cycles cancel. □

If we want to use Theorem 3 to get an expression for the Pfaffian of a single matrix A, we must set B to a matrix B_0 whose Pfaffian is 1. So we select the skew-symmetric matrix

$$B_0 = \begin{pmatrix} 0 & 1 & & & & & \\ -1 & 0 & & & & & \\ & & 0 & 1 & & & \\ & & -1 & 0 & & & \\ & & & & \ddots & & \\ & & & & & 0 & 1 \\ & & & & & -1 & 0 \end{pmatrix} \tag{19}$$

that represents the matching $\mathcal{M}_0 = (1,2),(3,4),\ldots,(n-1,n)$, which we use as a *reference matching.* So we immediately get the following alternate expression for the Pfaffian.

Corollary 2.

$$\operatorname{pf} A = \sum_{\mathcal{M}_0\text{-alternating cycle decompositions } \mathcal{C}} \operatorname{sign} \mathcal{C} \cdot \operatorname{weight}(\mathcal{C}) \qquad \square$$

Here, the general concepts of alternating cycles, cycles families, etc., specialize as follows: An $\mathcal{M}_0$-*alternating cycle* $(c_1, c_2, \ldots, c_i)$ is a cycle of even length i. In addition, we must have $c_3 = c_2 - 1$ if c_2 is even and $c_3 = c_2 + 1$ if c_2 is odd, and the same relation holds between c_4 and c_5, … , c_i and c_1. Its weight is given by (18) with respect to B_0, i. e., $b_{2j-1,2j} = 1$ and $b_{2j,2j-1} = -1$. All other notions are unchanged.

3.2 Alternating Clow Sequences

Generalizing Theorem 1 in Section 2.3, Mahajan, Subramanya, and Vinay [13] have extended the notion of an alternating cycle cover to an *alternating clow sequence.* This gives the following theorem.

Theorem 4.

$$\operatorname{pf} A \cdot \operatorname{pf} B = \sum_{\text{alternating clow sequences } \mathcal{C} \text{ of length } n} \operatorname{sign} \mathcal{C} \cdot \operatorname{weight}(\mathcal{C}) \tag{20}$$

$$\operatorname{pf} A = \sum_{\mathcal{M}_0\text{-alternating clow sequences } \mathcal{C} \text{ of length } n} \operatorname{sign} \mathcal{C} \cdot \operatorname{weight}(\mathcal{C}) \tag{21}$$

Here, an *alternating clow* $(c_1, c_2, \dots, c_i)$ of length i is defined like a clow, with the additional restriction that i must be even. The condition on the clow head is unchanged: $c_1 < c_2, \dots, c_i$. The weight of alternating clows is defined like in (18) for alternating cycles. An $\mathcal{M}_0$*-alternating clow* $(c_1, c_2, \dots, c_i)$ must have every other edge (c_2, c_3), (c_4, c_5), $\dots$, (c_i, c_1) belonging to $\mathcal{M}_0$. Alternating clow *sequences*, their length, weight, and sign are defined just as for clows and alternating cycle families. (Our notation and terminology differs from [13].)

The proof of (20), and of (21) which is a special case of it, follows the lines of the proof of Lemma 1: a sign-reversing bijection among "bad" alternating clow sequences which contain repeated elements is established. The only change occurs in case (A), when a clow C_k forms a sub-cycle $(c_\ell{=}c_j, c_{\ell+1}, \dots, c_{j-1})$. If that sub-cycle has odd length, we cannot split it from the current clow because it does not form an alternating cycle. In this case, we simply reverse the orientation and replace this part of C_k by $(c_\ell{=}c_j, c_{j-1}, \dots, c_{\ell+1})$. The weight of C_k changes sign, and it can also be checked easily that the resulting mapping between bad alternating clow sequences remains an involution. $\square$

Mahajan, Subramanya, and Vinay [13] considered only the case (21) of a single matrix, and they gave a slightly different proof.

3.3 Incremental Algorithms for the Pfaffian

A dynamic programming algorithm for evaluating (21) by summing over all partial alternating clow sequences in order of increasing length can easily be formulated in analogy to Section 2.4, see [13]. However, we shall instead formulate another algorithm along the lines of the recursive algorithm based on (7) in Section 2.5.

Since Pfaffians only make sense for matrices of even order, we partition the given skew-symmetric matrix by splitting two rows and columns from it.

$$A = \left(\begin{array}{cc|c} 0 & a_{12} & r \\ -a_{12} & 0 & -s^{\mathrm{T}} \\ \hline -r^{\mathrm{T}} & s & M \end{array}\right)$$

Enumeration of alternating clow sequences with head $c_1 = 1$ according to length yields the power series

$$G(\lambda) = -\lambda^2 + a_{12} + rB_0 s\lambda^{-2} + rB_0 MB_0 s\lambda^{-4} + rB_0 MB_0 MB_0 s\lambda^{-6} + \cdots \tag{22}$$

B_0 is the "skew-symmetric unit matrix" (19) of appropriate dimension, here $(n-2) \times (n-2)$. The coefficient of λ^{-i+2}, for $i \geq 2$, represents the negative sum of all weights of all alternating clow sequences with head 1 and length i. Note that a clow starting with vertex 1 must terminate with the edge $(2,1)$, and hence the factor $b_{21} = -1$ is alway present in the above terms. We define the *Pfaffian-characteristic polynomial* of a skew-symmetric $n \times n$ matrix A as

$$\tilde{P}_A(\lambda) := \tilde{q}_n \lambda^n + \tilde{q}_{n-2}\lambda^{n-2} + \cdots + \tilde{q}_2\lambda^2 + \tilde{q}_0,$$

where the quantities $\tilde{q}_i$ denote the sum of signed weights of all $\mathcal{M}_0$-alternating clow-sequences of length $n-i$. This definition can also be used for odd n. Similarly as in Theorem 2, $\tilde{q}_i$ is equal to the sum of signed weights of all $\mathcal{M}_0$-alternating cycle decompositions of length $n-i$. It can also be interpreted the sum of signed weights of certain matchings with $(n-i)/2$ edges as in (16), with an appropriate definition of signs.

Note that an alternating clow cannot have head 2. So we split alternating clow sequences into the (possibly empty) clow with head 1 and the remaining clow sequence, which consists of elements $\{3, \ldots, n\}$ only, in analogy to Section 2.5. We obtain the recursive equation

$$\tilde{P}_A(\lambda) = G(\lambda) \cdot \tilde{P}_M(\lambda), \tag{23}$$

which immediately leads to a recursive division-free algorithm for computing $\tilde{P}_A(\lambda)$ and the Pfaffian pf $A = \tilde{q}_0$.

4 Open Questions

The relation

$$\det A = \operatorname{pf}\begin{pmatrix} 0 & A \\ -A^{\mathrm{T}} & 0 \end{pmatrix}$$

shows that determinants are just special cases of Pfaffians that correspond to bipartite graphs. In this sense, the Pfaffian is a more basic notion, which would deserve a more thorough understanding, despite the traditional prevalence of the determinant in the curriculum and in applications. We mention a few questions that arise from the results which we have surveyed.

4.1 The Pfaffian-Characteristic Polynomial: Algebraic Interpretation

As in the case of the determinant, our algorithm has naturally lead to a polynomial $\tilde{P}_A(\lambda)$ of degree n that is associated to a skew-symmetric matrix A of order n. In contrast to the case of the determinant, we have no idea what an algebraic interpretation of this polynomial might be, or how one might prove the relation (23) by algebraic means. It is not generally the case that the characteristic polynomial is the square of $\tilde{P}_A(\sqrt{(}\lambda))$, as the relation $q_0 = \tilde{q}_0^2$ between their constant terms would suggest.

In Theorems 3 and 4, it has turned out that it appears more natural to deal with the product of Pfaffians of *two* different matrices A and B: after all, Corollary 2 is only a special case of Theorem 3 which is by no means easier to formulate, and the same relation holds between (20) and (21) in Theorem 4. Many identities about Pfaffians involve products of two Pfaffians that are taken from submatrices of a single given matrix. One might speculate that taking two different matrices and their submatrices could lead to more general theorems and at the same time more transparent proofs is some other cases as well.

Instead of substituting the matrix B_0 from (19) into (20) of Theorem 4, one could also use the matrix

$$B_1 = \begin{pmatrix} 0 & 1 & 1 & 1 & \cdots & 1 \\ -1 & 0 & 1 & 1 & \cdots & 1 \\ -1 & -1 & 0 & 1 & \cdots & 1 \\ -1 & -1 & -1 & 0 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & -1 & \cdots & 0 \end{pmatrix},$$

which might be called the skew-symmetric analog of the all-ones matrix and has $\operatorname{pf} B_1 = 1$. This results in a different expression for $\operatorname{pf} A$ than (21) in Theorem 4 and to another algorithm. That algorithm is in a way simpler than the incremental algorithm given in Section 3.3, since it increments the matrix by one row and column at a time instead of two. However, it uses more computations. The matrix B_1 also leads to a different concept of a Pfaffian-characteristic polynomial.

There is an analogous algorithm for directly computing the product of two Pfaffians according to equation (20) of Theorem 4. It seems that the Pfaffian-characteristic polynomial ought to be associated to two matrices instead of one. The incremental algorithm corresponding to equation (20) leads to a polynomial $\tilde{P}_{A,B}(\lambda)$ that is related to the characteristic polynomial by the equation

$$\tilde{P}_{A,A}(\lambda) = P_A(\lambda),$$

for skew-symmetric matrices A.

Is there a way to introduce a second matrix B also into the computation of determinants in the same natural way? This might be related to the *generalized eigenvalue problem* that requires the solution of the characteristic equation

$$\det(A - \lambda B) = 0,$$

for an arbitrary matrix B instead of the usual identity matrix. The above characteristic equation is equal to the characteristic equation of AB^{-1} if B is invertible. This problem is important for some applications. Can any of the algorithms of Section 2 be adapted to compute the coefficients of this polynomial?

4.2 Complexity

All division-free methods for computing the determinant or the Pfaffian that we have described take $O(n^4)$ steps. This has already been mentioned for the dynamic programming algorithm of Section 2.4. In the recursive algorithms (7) and (23) of Sections 2.5 and 3.3, the step of the recursion from the $(n-1)\times(n-1)$ matrix M to the $n \times n$ matrix A takes $O(n^3)$ time for computing the coefficients of the factor $F(\lambda)$ or $G(\lambda)$ respectively, and $O(n^2)$ steps for multiplying the two polynomials, if this is done in the straightforward way. This amounts to a total of $O(n^4)$ steps.

The bottleneck is thus the computation of the iterated matrix-vector products in (15) and (22): for $F(\lambda)$ in (15), we successively compute the row vectors r, rM, rM^2, ... , and multiply these by the column vector s. When the matrix A is sparse, this computation will be faster: If A has m nonzero elements, each matrix-vector product will take only $O(m)$ steps instead of $O(n^2)$ (assuming $m \geq n$). This leads to an algorithm with a running time of $O(mn^2)$. Thus, when m is small, this algorithm may be preferable to ordinary Gaussian elimination even when divisions are not an issue. Due to its simplicity, the iterative algorithm gains an additional advantage over Gaussian elimination or other direct methods, which must be careful about the choice of pivots in order to avoid numerical problems or to exploit sparsity.

Sparsity will also improve the dynamic programming algorithm of Section 2.4 from $O(n^4)$ to $O(mn^2)$ steps. It is a challenging problem to devise a division-free $O(n^3)$ method for computing the determinant.

The algorithms discussed in this paper can also be used to design good parallel algorithms for determinants and Pfaffians [3,14,13].

Despite their advantages, the methods described in this article are not well-known. A straightforward implementation of the incremental algorithm of Section 2.5 in MAPLE started to beat the built-in MAPLE procedure for the characteristic polynomial for random integer matrices of order 20, and was clearly faster already for matrices of order 40. For a simple 5×5 matrix with rational entries like $(a_{ij}) = (\frac{1}{x_i+x_j})$, our algorithm for the characteristic polynomial was two times faster than MAPLE's algorithm for computing the determinant only.

4.3 Alternative Algorithms

In the introduction it was mentioned that division-free algorithms might be useful for evaluating determinants of integers. The most straightforward algorithm would be to carry out Gaussian elimination with rational arithmetic. This approach may result in an explosive growth in the lengths of the numbers unless fractions are reduced after every step. However, to avoid the excessive growth of numbers, it is not necessary to reduce fractions by their greatest common divisor: one can perform pivots involving divisions for which the result is known to be integral. This form of integral pivoting is attributed to Camille Jordan [5, p. 69], see also [6,2,9,7]. Edmonds [6] has shown that the numbers do not explode and hence, this algorithm runs in polynomial time. The same approach can be extended to Pfaffians [9].

Thus, for evaluating determinants or Pfaffians of integers, this approach is superior to our division-free algorithms because it takes only $O(n^3)$ operations. However, when dealing with multivariate polynomials, even "integer division" can be awkward, and division-free algorithms may prove to be worthwhile.

Strassen [22] has given a general recipe for converting an algorithm that uses divisions to obtain an integral final result into an algorithm without divisions. Divisions are simulated by expanding expressions into truncated power series. This is somehow reminiscent of the recursion (9) which also involves power series that can be truncated because the result is a polynomial. According to [4], Strassen's approach leads to an algorithm with $O(n^5)$ steps in the case of Gaussian elimination. It would be interesting to see if there is any relation of this algorithm to the algorithms presented here.

References

1. Martin Aigner, Lattice paths and determinants. In: *Computational Discrete Mathematics*, ed. Helmut Alt, (this volume), Lecture Notes Comput. Sci., Vol. 2122 2001, pp. 1–12.
2. Erwin H. Bareiss, Sylvester's identity and multistep integer-preserving Gaussian elimination. *Math. Comput.* **22** (1968), 565–578.
3. Stuart J. Berkowitz, On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.* **18** (1984), 147–150.
4. Allan Borodin, Joachim von zur Gathen, John Hopcroft, Fast parallel matrix and GCD computations. *Inf. Control* **52** (1982), 241–256
5. E. Durant, *Solution numérique des équations algébriques, tome II: systèmes des plusieurs équations*. Masson & Cie., Paris 1961.
6. Jack Edmonds, Systems of distinct representatives and linear algebra. *J. Res. Nat. Bur. Standards, Sect. B* **71** (1967), 241–245.
7. Jack Edmonds, J.-F. Maurras, Note sur les Q-matrices d'Edmonds. *RAIRO, Rech. opér.* **31** (1997), 203–209.
8. D. K. Faddeyev, V. N. Faddeyeva, *Vyčislitel′nye metody lineĭnoĭ algebry* (in Russian), Moscow, 1960. English translation: D. K. Faddeev, V. N. Faddeeva, *Numerical Methods of Linear Algebra*. Freeman, San Francisco 1963. German translation: D. K. Faddejew, W. N. Faddejewa, *Numerische Methoden der linearen Algebra*, several editions since 1964.
9. G. Galbiati and Franceso Maffioli, On the computation of pfaffians. *Discr. Appl. Math.* **51** (1994), 269–275.
10. Donald E. Knuth, Overlapping Pfaffians. *Electron. J. Comb.* **3** (1996), No. 2, article R5, 13 pp. Printed version: *J. Comb.* **3** (1996), No. 2, 147–159.
11. Christian Krattenthaler, Advanced determinant calculus. *Séminaire Lotharingien de Combinatoire* **B42q** (1999), 67 pp.
12. László Lovász, M. D. Plummer, *Matching Theory*. Ann. Discr. Math., Vol. 29. North-Holland Mathematics Studies, Vol. 121. Amsterdam 1986.
13. Meena Bhaskar Mahajan, P R Subramanya, V Vinay, A combinatorial algorithm for Pfaffians. In: *Computing and combinatorics. Proc. 5th annual international conference. (COCOON '99), Tokyo, July 1999*, ed. Takao Asano et al., Lecture

Notes Comput. Sci. 1627, Springer-Verlag, pp. 134–143 (1999). Extended version: DIMACS Technical Report 99-39, Rutgers University, July 1999.
14. Meena Bhaskar Mahajan, V Vinay, Determinant: Combinatorics, algorithms, and complexity. *Chicago J. Theor. Comput. Sci.*, Vol. 1997, Article no. 1997-5, 26 pp.
15. Meena Mahajan, V. Vinay, Determinant: Old algorithms, new insights. *SIAM J. Discrete Math.* **12** (1999), 474–490.
16. Thomas Muir, *A Treatise on the Theory of Determinants.* MacMillan and Co., London 1882; repr. Dover, New York 1960.
17. Günter Rote, Path problems in graphs. In: *Computational graph theory*, ed. Gottfried Tinhofer et al., Computing Suppl. **7**, 155–189, Springer-Verlag, Wien 1990.
18. D. E. Rutherford, The Cayley-Hamilton theorem for semi-rings. *Proc. Roy. Soc. Edinburgh*, Sect. A **66** (1961–64), 211–215 (1964).
19. Paul A. Samuelson, A method of determining explicitly the coefficients of the characteristic equation. *Ann. Math. Statist.* **13** (1942), 424–429.
20. Dennis Stanton, Dennis White, *Constructive combinatorics.* Springer-Verlag, New York 1986.
21. John R. Stembridge, Nonintersecting paths, pfaffians, and plane partitions. *Adv. Math.* **83** (1990), 96–113.
22. Volker Strassen, Vermeidung von Divisionen. *J. reine angew. Math.* **264** (1973), 184–202.
23. Howard Straubing, A combinatorial proof of the Cayley-Hamilton theorem. *Discrete Math.* **43** (1983), 273–279.
24. Leslie G. Valiant, Why is Boolean complexity theory difficult? In: *Boolean Function Complexity*, ed. M. S. Paterson, LMS Lecture Notes Series, Vol. 169, Cambridge Univ. Press, 1992, pp. 84–94.
25. Doron Zeilberger, A combinatorial approach to matrix algebra. *Discrete Math.* **56** (1985), 61–72.