comput. complex. **13** (2004), 91 – 130
1016-3328/04/030091–40
DOI 10.1007/s00037-004-0185-3

computational complexity

ON THE COMPLEXITY OF COMPUTING DETERMINANTS

Erich Kaltofen and Gilles Villard

To B. David Saunders
on the occasion of his 60th birthday

Abstract. We present new baby steps/giant steps algorithms of asymptotically fast running time for dense matrix problems. Our algorithms compute the determinant, characteristic polynomial, Frobenius normal form and Smith normal form of a dense $n \times n$ matrix A with integer entries in $(n^{3.2} \log \|A\|)^{1+o(1)}$ and $(n^{2.697263} \log \|A\|)^{1+o(1)}$ bit operations; here $\|A\|$ denotes the largest entry in absolute value and the exponent adjustment by "$+o(1)$" captures additional factors $C_1(\log n)^{C_2}(\log\log \|A\|)^{C_3}$ for positive real constants C_1, C_2, C_3. The bit complexity $(n^{3.2} \log \|A\|)^{1+o(1)}$ results from using the classical cubic matrix multiplication algorithm. Our algorithms are randomized, and we can certify that the output is the determinant of A in a Las Vegas fashion. The second category of problems deals with the setting where the matrix A has elements from an abstract commutative ring, that is, when no divisions in the domain of entries are possible. We present algorithms that deterministically compute the determinant, characteristic polynomial and adjoint of A with $n^{3.2+o(1)}$ and $O(n^{2.697263})$ ring additions, subtractions and multiplications.

Keywords. Integer matrix, matrix determinant, characteristic polynomial, Smith normal form, bit complexity, division-free complexity, randomized algorithm, multivariable control theory, realization, matrix sequence, block Wiedemann algorithm, block Lanczos algorithm.
Subject classification. 68W30, 15A35.

1. Introduction

The computational complexity of many problems in linear algebra has been tied to the computational complexity of matrix multiplication. If the result is to be exact, for example the exact rational solution of a linear system, the lengths of the integers involved in the computation and the answer affect the

running time of the used algorithms. A classical methodology is to compute the results via Chinese remaindering. Then the standard analysis yields a number of fixed radix, i.e. bit operations for a given problem that is essentially (within polylogarithmic factors) bounded by the number of field operations for the problem times the maximal scalar length in the output. The algorithms at times use randomization, because not all modular images may be usable. For the determinant of an $n \times n$ integer matrix A one thus gets a running time of $(n^4 \log \|A\|)^{1+o(1)}$ bit operations (von zur Gathen & Gerhard 1999, Chapter 5.5), because the determinant can have at most $(n \log \|A\|)^{1+o(1)}$ digits; by $\|A\|$ we denote the largest entry in absolute value. Here and throughout this paper the exponent adjustment by "$+o(1)$" captures additional factors $C_1(\log n)^{C_2}(\mathrm{loglog}\,\|A\|)^{C_3}$ for positive real constants C_1, C_2, C_3 ("soft-O"). Via an algorithm that can multiply two $n \times n$ matrices in $O(n^\omega)$ scalar operations the time is reduced to $(n^{\omega+1} \log \|A\|)^{1+o(1)}$. We can set $\omega = 2.375477$ (Coppersmith & Winograd 1990).

First, it was recognized that for the problem of computing the exact rational solution of a linear system the process of Hensel lifting can accelerate the bit complexity beyond the Chinese remainder approach (Dixon 1982), namely to cubic in n without using fast matrix multiplication algorithms. For the determinant of an $n \times n$ integer matrix A, an algorithm with $(n^{3.5} \log \|A\|^{1.5})^{1+o(1)}$ bit operations is given by Eberly *et al.* (2000).[1] Their algorithm computes the Smith normal form via the binary search technique of Villard (2000).

Our algorithms combine three ideas.

(I) The first is an algorithm by Wiedemann (1986) for computing the determinant of a sparse matrix over a finite field. Wiedemann finds the minimum polynomial for the matrix as a linear recurrence on a corresponding Krylov sequence. By preconditioning the input matrix, that minimum polynomial is the characteristic polynomial, and the determinants of the original and preconditioned matrix have a direct relation.

(II) The second is by Kaltofen (1992) where Wiedemann's approach is applied to dense matrices whose entries are polynomials over a field. Kaltofen achieves speedup by employing Shank's baby steps/giant steps technique for the computation of the linearly recurrent scalars (cf. Paterson & Stockmeyer 1973). For integer matrices the resulting randomized algorithm is of the Las Vegas kind—always correct, probably fast—and has worst case

[1] Eberly *et al.* (2000) give an exponent for $\log \|A\|$ of 2.5, but the improvement to 1.5 based on fast Chinese remaindering (Aho *et al.* 1974) is immediate.

bit complexity $(n^{3.5} \log \|A\|)^{1+o(1)}$ and again can be speeded with subcubic time matrix multiplication (Kaltofen & Villard 2001). A detailed description of this algorithm, with an early termination strategy in case the determinant is small (cf. Brönnimann *et al.* 1999; Emiris 1998), is presented by Kaltofen (2002).

(III) By considering a bilinear map using two blocks of vectors rather than a single pair of vectors, Wiedemann's algorithm can be accelerated (Coppersmith 1994; Kaltofen 1995; Villard 1997a,b). Blocking can be applied to our algorithms for dense matrices and further reduces the bit complexity.

The above ingredients yield a randomized algorithm of the Las Vegas kind for computing the determinant of an $n \times n$ integral matrix A in $(n^{3+1/3} \times \log \|A\|)^{1+o(1)}$ expected bit operations, that with a standard cubic matrix multiplication algorithm. If we employ fast FFT-based Padé approximation algorithms for matrix polynomials, for example the so-called half-GCD algorithm (von zur Gathen & Gerhard 1999) and fast matrix multiplication algorithms, we can further lower the expected number of bit operations. Under the assumption that two $n \times n$ matrices can be multiplied in $O(n^\omega)$ operations in the field of entries, and an $n \times n$ matrix by an $n \times n^\zeta$ matrix in $n^{2+o(1)}$ operations, we obtain an expected bit complexity for the determinant of

$$(n^\eta \log \|A\|)^{1+o(1)} \quad \text{with} \quad \eta = \omega + \frac{1-\zeta}{\omega^2 - (2+\zeta)\omega + 2}. \tag{1.1}$$

The best known values $\omega = 2.375477$ (Coppersmith & Winograd 1990) and $\zeta = 0.2946289$ (Coppersmith 1997) yield $\eta = 2.697263$. For $\omega = 3$ and $\zeta = 0$ we have $\eta = 3 + 1/5$ as given in the abstract above.

Our techniques can be further combined with the ideas by Giesbrecht (2001) to produce a randomized algorithm for computing the integer Smith normal form of an integer matrix. The method becomes Monte Carlo—always fast and probably correct—and has the same bit complexity (1.1). In addition, we can compute the characteristic polynomial of an integer matrix by Hensel lifting (Storjohann 2000b). Again the method is Monte Carlo and has bit complexity (1.1). Both results utilize the fast determinant algorithm for matrix polynomials (Storjohann 2002, 2003).

The algorithm by Kaltofen (1992) (see case II above) was originally put to a different use, namely that of computing the characteristic polynomial and adjoint of a matrix without divisions, counting additions, subtractions, and multiplications in the commutative ring of entries. Serendipitously, blocking (see case III above) can be applied to our original 1992 division-free algorithm,

and we obtain a deterministic algorithm that computes the determinant and characteristic polynomial of a matrix over a commutative ring in $n^{\eta+o(1)}$ ring additions, subtractions and divisions, where η is given by (1.1). The exponent $\eta = 2.697263$ seems to be the best that is known today for the division-free determinant problem. By the technique of Baur and Strassen (1983) we obtain the adjoint of a matrix in the same division-free complexity.

Kaltofen and Villard (2004) have identified other algorithms for computing the determinant of an integer matrix. Those algorithms often perform at cubic bit complexity on what we call propitious inputs, but they have a worst case bit complexity that is higher than our methods. One such method is Clarkson's algorithm (Brönnimann & Yvinec 2000; Clarkson 1992), where the number of mantissa bits in the intermediate floating point scalars that are necessary for obtaining a correct sign depends on the orthogonal defect of the matrix. If the matrix has a large first invariant factor, Chinese remaindering can be employed in connection with computing the solution of a random linear system via Hensel lifting (Abbott *et al.* 1999; Pan 1988).

Notation. By $S^{m\times n}$ we denote the set of $m \times n$ matrices with entries in the set S. The set $\mathbb{Z}$ are the integers. For $A \in \mathbb{Z}^{n\times n}$ we denote by $\|A\|$ the matrix height (Kaltofen & May 2003, Lemma 2):

$$\|A\| = \|A\|_{\infty,1} = \max_{x\neq 0} \|Ax\|_\infty / \|x\|_1 = \max_{1\leq i,j\leq n} |a_{i,j}|.$$

Hence the maximal bit length of all entries in A and their signs is, depending on the exact representation, at least $2 + \lfloor \log_2 \max\{1, \|A\|\} \rfloor$. In order to avoid zero factors or undefined logarithms, we shall simply define $\|A\| > 1$ whenever it is necessary.

Organization of the paper. Section 2 introduces Coppersmith's block Wiedemann algorithm and establishes all necessary mathematical properties of the computed matrix generators. In particular, we show the relation of the determinants of the generators with the (polynomial) invariant factors of the characteristic matrix (Theorem 2.12), which essentially captures the block version of the Cayley–Hamilton property. In addition, we characterize when short sequences are insufficient to determine the minimum generator. Section 3 deals with the computation of the block generator. We give the generalization of the Knuth/Schönhage/Moenck algorithm for polynomial quotient sequences to matrix polynomials and show that in our case by randomization all leading coefficients stay non-singular (Lemma 3.10). Section 4 presents our new determinant algorithm for integer matrices and gives the running time analysis

when cubic matrix multiplication algorithms are employed (Theorem 4.2). Section 5 presents the division-free determinant algorithm. Section 6 contains the analysis for versions of our algorithms when fast matrix multiplication is introduced. The asymptotically best results are derived there. Section 7 presents the algorithms for the Smith normal form and the characteristic polynomial of an integer matrix. We give concluding thoughts in Section 8.

2. Generating polynomials of matrix sequences

Coppersmith (1994) first introduced blocking to the Wiedemann method. In our description we also take into account the interpretation by Villard (1997a; 1997b), where the relevant literature from linear control theory is cited. Our algorithms rely on the notion of minimum linear generating polynomials (generators) of matrix sequences. This notion is introduced below in Section 2.1. We also see how generators are related to block Hankel matrices and recall some basic facts concerning their computation. In Section 2.2 we then study determinants and Smith normal forms of generators and see how they will be used for solving our initial problem. All the results are given over an arbitrary commutative field $\mathbb{K}$.

2.1. Generators and block Hankel matrices. For the "block" vectors $X \in \mathbb{K}^{n\times l}$ and $Y \in \mathbb{K}^{n\times m}$ consider the sequence of $l \times m$ matrices

$$B^{[0]} = X^{\mathrm{Tr}}Y,\ B^{[1]} = X^{\mathrm{Tr}}AY,\ B^{[2]} = X^{\mathrm{Tr}}A^2Y,\ \ldots,\ B^{[i]} = X^{\mathrm{Tr}}A^iY,\ \ldots \tag{2.1}$$

As in the unblocked Wiedemann method, we seek linear generating polynomials. A vector polynomial $\sum_{i=0}^{d} c^{[i]}\lambda^i$, where $c^{[i]} \in \mathbb{K}^m$, is said to *linearly generate the sequence* (2.1) *from the right* if

$$\forall j \geq 0\colon \quad \sum_{i=0}^{d} B^{[j+i]}c^{[i]} = \sum_{i=0}^{d} X^{\mathrm{Tr}}A^{i+j}Yc^{[i]} = 0^l. \tag{2.2}$$

For the minimum polynomial of A, $f^A(\lambda)$, and for the μ-th unit vector in $\mathbb{K}^m$, $e^{[\mu]}$, $f^A(\lambda)e^{[\mu]} \in \mathbb{K}[\lambda]^m$ is such a generator because it already generates the Krylov sequence $\{A^iY^{[\mu]}\}_{i\geq 0}$, where $Y^{[\mu]}$ is the μ-th column of Y. We can now consider the set of all such right vector generators. This set forms a $\mathbb{K}[\lambda]$-submodule of the $\mathbb{K}[\lambda]$-module $\mathbb{K}[\lambda]^m$ and contains m linearly independent (over the field of rational functions $\mathbb{K}(\lambda)$) elements, namely all $f^A(\lambda)e^{[\mu]}$. Furthermore, the submodule has an ("integral") basis over $\mathbb{K}[\lambda]$, namely any set of m linearly independent generators such that the degree in λ of the determinant

of the matrix formed by those basis vector polynomials as columns is minimal. The matrices corresponding to all integral bases clearly are right equivalent with respect to multiplication from the right by any unimodular matrix in $\mathbb{K}[\lambda]^{m\times m}$, whose determinant is by definition of unimodularity a non-zero element in $\mathbb{K}$. Thus we can pick a matrix canonical form for this right equivalence, say the Popov form (Popov 1970) (see also Kailath 1980, §6.7.2) to get the following definition.

Definition 2.3. The unique matrix generating polynomial for (2.1) in Popov form, denoted by $F_X^{A,Y} \in \mathbb{K}[\lambda]^{m\times m}$, is called the *minimum matrix generating polynomial* (*generator*).

As we will show below, $\deg(\det F_X^{A,Y}) \leq n$. The computation of the minimum matrix generating polynomial from the matrix sequence (2.1) can be accomplished by several interrelated approaches. One is a sophisticated generalization of the Berlekamp/Massey algorithm (Coppersmith 1994; Dickinson *et al.* 1974; Rissanen 1972). Another generalizes the theory of Padé approximation (Beckermann & Labahn 1994; Forney, Jr. 1975; Giorgi *et al.* 2003; Van Barel & Bultheel 1992). The interpretation of the Berlekamp/Massey algorithm as a specialization of the extended Euclidean algorithm (Dornstetter 1987; Sugiyama *et al.* 1975) can be carried over to matrix polynomials (Coppersmith 1994; Thomé 2002) (see also Section 3 below). All approaches solve the classical Levinson–Durbin problem, which for matrix sequences becomes a block Toeplitz linear system (Kaltofen 1995). The relation to Toeplitz/Hankel matrices turns out to be a useful device for establishing certain properties.

For a degree d and a length e we consider the $l\cdot e$ by $m\cdot(d+1)$ block Hankel matrix

$$\mathrm{Hk}_{e,d+1}(A,X,Y) = \begin{bmatrix} B^{[0]} & B^{[1]} & \dots & B^{[d-1]} & B^{[d]} \\ B^{[1]} & B^{[2]} & & B^{[d]} & B^{[d+1]} \\ \vdots & & \ddots & \vdots & \vdots \\ B^{[e-1]} & \dots & \dots & \dots & B^{[d+e-1]} \end{bmatrix} \tag{2.4}$$

For any vector generator $\sum_{i=0}^{d} c^{[i]}\lambda^i \in \mathbb{K}[\lambda]^m$ we must have

$$\mathrm{Hk}_{e,d+1} \cdot \begin{bmatrix} c^{[0]} \\ \vdots \\ c^{[d]} \end{bmatrix} = 0 \quad \text{for all } e > 0.$$

By considering the rank of (2.4) we can infer the reverse. If

$$\mathrm{Hk}_{n,d+1} \cdot \begin{bmatrix} c^{[0]} \\ \vdots \\ c^{[d]} \end{bmatrix} = 0 \tag{2.5}$$

then $\sum_{i=0}^{d} c^{[i]}\lambda^i$ is a vector generator of (2.1). The claim follows from the fact that $\operatorname{rank}\mathrm{Hk}_{n,d+1} = \operatorname{rank}\mathrm{Hk}_{n+e',d+1}$ for all $e' > 0$. The latter is justified by observing that any row in the $(n+e')$th block row of $\mathrm{Hk}_{n+e',d+1}$ is linearly dependent on corresponding previous rows via the minimum polynomial f^A, which has degree $\deg f^A \leq n$.

We observe that $\operatorname{rank}\mathrm{Hk}_{e,d} \leq n$ for all $d > 0, e > 0$ by considering the factorization

$$\mathrm{Hk}_{e,d} = \begin{bmatrix} X^{\mathrm{Tr}} \\ X^{\mathrm{Tr}}A \\ X^{\mathrm{Tr}}A^2 \\ \vdots \\ X^{\mathrm{Tr}}A^{e-1} \end{bmatrix} \cdot \begin{bmatrix} Y & AY & A^2Y & \dots & A^{d-1}Y \end{bmatrix}$$

and noting that either matrix factor has rank at most n.

Therefore, when $d \geq \deg F_X^{A,Y}$, the module over $\mathbb{K}[\lambda]$ generated by solutions to (2.5) is the module of vector generators, with the columns of $F_X^{A,Y}(\lambda)$ as basis. In this case, if the column degrees of the minimum generator are $\delta_1 \leq \cdots \leq \delta_m$, the dimension of the right nullspace of $\mathrm{Hk}_{e,d+1}$ in (2.5) over $\mathbb{K}$ is $(d-\delta_1+1)+\cdots+(d-\delta_m+1)$. Hence for $d \geq \deg F_X^{A,Y}$ and $e \geq n$ we have $\operatorname{rank}\mathrm{Hk}_{e,d+1} = \delta_1 + \cdots + \delta_m = \deg(\det F_X^{A,Y}) \leq n$, the latter because $F_X^{A,Y}(\lambda)$ is in Popov form. Since the last block column in $\mathrm{Hk}_{e,d+1}$ with $d \geq \deg F_X^{A,Y}$ is generated by previous block columns, via shifting lower degree columns of $F_X^{A,Y}(\lambda)$ as necessary by multiplying with powers of λ, we have

$$\operatorname{rank}\mathrm{Hk}_{e,d} = \deg(\det F_X^{A,Y}) \quad \text{for } d \geq \deg F_X^{A,Y} \text{ and } e \geq n. \tag{2.6}$$

One may now define the minimum $e_{\min}$ such that the matrix $\mathrm{Hk}_{e_{\min},d}$ for $d = \deg F_X^{A,Y}$ has full rank $\deg(\det F_X^{A,Y})$. Any algorithm for computing the minimum generator requires the first $\deg F_X^{A,Y} + e_{\min}$ elements of the sequence (2.1).

We give an example over $\mathbb{Q}$ (Turner 2002). Let

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 \end{bmatrix}, \quad X = Y = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Then

$$B^{[0]} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad B^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B^{[2]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B^{[3]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$B^{[4]} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \quad B^{[5]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B^{[6]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B^{[7]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Therefore

$$\mathrm{Hk}_{4,5}(A, X, Y) = \left[\begin{array}{cc|cc|cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

and since the nullspace of $\mathrm{Hk}_{4,5}(A, X, Y)$ is generated by the vectors

$$\begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix},$$

we get

$$F_X^{A,Y}(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \lambda^4 + \begin{bmatrix} -2 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \lambda^4 - 2 & 0 \\ 0 & 1 \end{bmatrix}.$$

Now let X be as above and let $\overline{Y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}^{\mathrm{Tr}}$. Then

$$\overline{B}^{[0]} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad \overline{B}^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \overline{B}^{[2]} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},$$
$$\overline{B}^{[3]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \overline{B}^{[4]} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}, \quad \overline{B}^{[5]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

Therefore

$$\mathrm{Hk}_{4,3}(A, X, \overline{Y}) = \left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right],$$

and since the nullspace of $\mathrm{Hk}_{4,3}(A, X, \overline{Y})$ is generated by the vectors

$$\left[\begin{array}{c} 0 \\ -2 \\ \hline 0 \\ 0 \\ \hline 1 \\ 0 \end{array}\right], \quad \left[\begin{array}{c} -1 \\ 0 \\ \hline 0 \\ 0 \\ \hline 0 \\ 1 \end{array}\right],$$

we get

$$F_X^{A,\overline{Y}}(\lambda) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \lambda^2 - \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} \lambda^2 & -1 \\ -2 & \lambda^2 \end{bmatrix}.$$

Note that in both cases the determinant of the minimum generator is $\lambda^4 - 2$, which is $\det(\lambda I - A)$.

The second example above, where $e_{\min} = 4 > \deg F_X^{A,Y} = 2$, shows that more than $2 \deg F_X^{A,Y}$ sequence elements may be necessary to compute the generator, in contrast to the scalar Berlekamp/Massey theory. The last block row of $\mathrm{Hk}_{4,3}(A, X, \overline{Y})$ is required to restrict the right nullspace to the two generating vectors.

However, for random X and Y both $\deg F_X^{A,Y}$ and $e_{\min}$ are small. Let us define, for fixed l and m,

$$\nu = \max_{d \geq 1,\, e \geq 1,\, X \in \mathbb{K}^{n \times l},\, Y \in \mathbb{K}^{n \times m}} \operatorname{rank} \mathrm{Hk}_{e,d}(A, X, Y). \tag{2.7}$$

Indeed, the probabilistic analysis in (Kaltofen 1995, Section 5) and (Villard 1997b, Corollary 6.4) shows the existence of matrices $W \in \mathbb{K}^{n\times l}$ and $Z \in \mathbb{K}^{n\times m}$ such that $\operatorname{rank} \mathrm{Hk}_{e_0,d_0}(A,W,Z) = \nu$ with $d_0 = \lceil \nu/m \rceil$ and $e_0 = \lceil \nu/l \rceil$. Moreover, ν is equal to the sum of the degrees of the first $\min\{l,m\}$ invariant factors of $\lambda I - A$ (see Theorem 2.12 below), and hence X, Y can be taken from any field extension of $\mathbb{K}$. Then due to the existence of W, Z, for symbolic entries in X, Y and therefore, by (DeMillo & Lipton 1978; Schwartz 1980; Zippel 1979), for random entries, the maximal rank is preserved for block dimensions e_0, d_0. Note that the degree of the minimum matrix generating polynomial is now $\deg F_X^{A,Y} = d_0 < n/m + 1$ and the number of sequence elements required to compute the minimum generator is $d_0 + e_0 = \lceil \nu/l \rceil + \lceil \nu/m \rceil < n/l + n/m + 2$. If $\mathbb{K}$ is a small finite field, Wiedemann's analysis has been generalized by Villard (1997b) (see also Brent *et al.* 2003).

As with the unblocked Wiedemann projections, unlucky projection block vectors X and Y may cause a drop in the determinantal degree $\deg(\det F_X^{A,Y})$. They may also increase the length of the sequence required to compute the generator $F_X^{A,Y}$.

2.2. Smith normal forms of matrix generating polynomials. In this section we study how the invariant structure of $F_X^{A,Y}$ partly reveals the structure of A and $\lambda I - A$. Our algorithms in Sections 4 and 5 pick random block vectors X, Y or use special projections and compute a generator from the first $d_0 + e_0$ elements of (2.1). Under the assumption that $\operatorname{rank} \mathrm{Hk}_{e,d} = \nu$ (see (2.7)) for sufficiently large d, e, we prove here that $\det F_X^{A,Y}$ is the product of the first $\min\{l,m\}$ invariant factors of $\lambda I - A$. These are well-studied facts in the theory of realizations of multivariable control theory; for instance see Kailath (1980). The basis is the matrix power series

$$X^{\mathrm{Tr}}(\lambda I - A)^{-1} Y = X^{\mathrm{Tr}} \left(\sum_{i\geq 0} \frac{A^i}{\lambda^{i+1}} \right) Y = \sum_{i\geq 0} \frac{B^{[i]}}{\lambda^{i+1}}.$$

LEMMA 2.8. *One has the fraction description*

$$X^{\mathrm{Tr}}(\lambda I - A)^{-1} Y = N(\lambda) D(\lambda)^{-1} \tag{2.9}$$

if and only if there exists $T \in \mathbb{K}[\lambda]^{m\times m}$ *such that* $D = F_X^{A,Y} T$.

PROOF. For the necessary condition, since every polynomial numerator in $X^{\mathrm{Tr}}(\lambda I - A)^{-1} Y$ has degree strictly less than the corresponding denominator,

every column of N has degree strictly less than that of the corresponding column of D. Thus it can be checked that the columns of D satisfy (2.2) and D must be a multiple of $F_X^{A,Y}$. Conversely, let $D = F_X^{A,Y}T$ in $\mathbb{K}[\lambda]^{m\times m}$ be an invertible matrix generator for (2.1). Using (2.2) for its m columns it can be seen that we have

$$X^{\mathrm{Tr}}(\lambda I - A)^{-1}YD(\lambda) = N(\lambda) \in \mathbb{K}[\lambda]^{l\times m},$$

where the column degrees of N are lower than those of D. This yields the matrix fraction description (2.9). □

Clearly, for $D = F_X^{A,Y}$, the minimum polynomial $f^A(\lambda)$ is a common denominator of the rational entries of the matrices on both sides of (2.9). If the least common denominator of the left side matrix is actually the characteristic polynomial $\det(\lambda I - A)$, then it follows from degree considerations that $\det F_X^{A,Y} = \det(\lambda I - A)$. Our algorithm uses the matrix preconditioners discussed in Section 4 and random or ad hoc projections (Section 5) to achieve this determinantal equality. We shall make the relationship between $\lambda I - A$ and $F_X^{A,Y}$ more explicit in Theorem 2.12 whose proof will rely on the structure of the matrix denominator D in (2.9) and on the following.

For a square matrix M over $\mathbb{K}[\lambda]$ we consider the Smith normal form (Newman 1972), which is an equivalent diagonal matrix over $\mathbb{K}[\lambda]$ with diagonal elements $s_1(\lambda), \ldots, s_\phi(\lambda), 1, \ldots, 1, 0, \ldots, 0$, where the s_i's are the non-trivial invariant factors of M, that is, non-constant monic polynomials with the property that s_i is a (trivial or nontrivial) polynomial factor of s_{i-1} for all $2 \le i \le \phi$. Because the Smith normal form of the characteristic matrix $\lambda I - A$ corresponds to the Frobenius canonical form of A for similarity, the largest invariant factor of $\lambda I - A$, $s_1(\lambda)$, equals the minimum polynomial $f^A(\lambda)$.

LEMMA 2.10. *Let $M \in \mathbb{K}[\lambda]^{\mu\times\mu}$ be non-singular and let $U \in \mathbb{K}[\lambda]^{\mu\times\mu}$ be unimodular such that*

$$MU = \begin{bmatrix} H & H_{12} \\ 0 & H_{22} \end{bmatrix}, \tag{2.11}$$

where H is a square matrix. Then the i-th invariant factor of H divides the i-th invariant factor of M.

PROOF. Identity (2.11) may be rewritten as

$$MU = \begin{bmatrix} I & H_{12} \\ 0 & H_{22} \end{bmatrix} \begin{bmatrix} H & 0 \\ 0 & I \end{bmatrix}.$$

Since the invariant factors of two non-singular matrices divide the invariant factors of their product (Newman 1972, Theorem II.14), the largest invariant factors of $\mathrm{diag}(H, I)$ that are those of H, divide the corresponding invariant factors of MU and thus M. □

We can now see how the Smith form of $F_X^{A,Y}$ is related to that of $\lambda I - A$. Essentially, the result may be obtained, for instance, following the lines in (Kailath 1980, §6.4.2). Here we give a statement and a proof better suited to our purposes.

THEOREM 2.12. *Let $A \in \mathbb{K}^{n\times n}$, $X \in \mathbb{K}^{n\times l}$, $Y \in \mathbb{K}^{n\times m}$ and let $s_1, \ldots, s_\phi$ denote all invariant factors of $\lambda I - A$. The i-th invariant factor of $F_X^{A,Y}$ divides s_i. Furthermore, there exist matrices $W \in \mathbb{K}^{n\times l}$ and $Z \in \mathbb{K}^{n\times m}$ such that for all i, $1 \le i \le \min\{l, m, \phi\}$, the i-th invariant factor of $F_W^{A,Z}$ is equal to s_i and the $m - \min\{l, m, \phi\}$ remaining ones are equal to 1. Moreover, for fixed l and m,*

$$\begin{aligned} \deg_\lambda(\det F_W^{A,Z}(\lambda)) &= \max_{X,Y} \deg(\det F_X^{A,Y}(\lambda)) \\ &= \deg s_1 + \cdots + \deg s_{\min\{l,m,\phi\}} \\ &= \nu, \quad \textit{which is defined in } (2.7). \end{aligned} \tag{2.13}$$

PROOF. We prove the first statement for a particular denominator matrix D of a fraction description of $X^{\mathrm{Tr}}(\lambda I - A)^{-1}Y$. Indeed, if the i-th invariant factor of D divides s_i then, by Lemma 2.8 and using the product argument given in the proof of Lemma 2.10, the same holds by transitivity of division for $F_X^{A,Y}$. When Y has rank $r < m$, one may introduce an invertible transformation $Q \in \mathbb{K}^{m\times m}$ such that $YQ = [Y_1\ 0]$ with $Y_1 \in \mathbb{K}^{n\times r}$. From this, if $X^{\mathrm{Tr}}(\lambda I - A)^{-1}Y_1 = N_1 D_1^{-1}$ then

$$X^{\mathrm{Tr}}(\lambda I - A)^{-1}Y = \begin{bmatrix} N_1(\lambda) & 0 \end{bmatrix} \begin{bmatrix} D_1(\lambda) & 0 \\ 0 & I \end{bmatrix}^{-1} Q^{-1}$$

and the invariant factors of the denominator matrix $Q\ \mathrm{diag}(D_1, I)$ are those of D_1. We can thus without loss of generality assume that Y has full column rank. Let us now construct a fraction description of $X^{\mathrm{Tr}}(\lambda I - A)^{-1}Y$ with D as announced. Choose $Y_c \in \mathbb{K}^{n\times(n-m)}$ such that $T = [Y\ Y_c]$ is invertible in $\mathbb{K}^{n\times n}$ and let $D \in \mathbb{K}[\lambda]^{m\times m}$ be defined from a unimodular triangularization of $T^{-1}(\lambda I - A)$, that is,

$$T^{-1}(\lambda I - A)U(\lambda) = \begin{bmatrix} D(\lambda) & H_{12}(\lambda) \\ 0 & H_{22}(\lambda) \end{bmatrix} \tag{2.14}$$

with U unimodular. If V is the matrix formed by the first m columns of U we have the fraction descriptions $(\lambda I - A)^{-1}Y = VD^{-1}$ and $X^{\mathrm{Tr}}(\lambda I - A)^{-1}Y =$

$(X^{\mathrm{Tr}}V)D^{-1}$. Thus D is a denominator matrix for $X^{\mathrm{Tr}}(\lambda I - A)^{-1}Y$. By (2.14) and Lemma 2.10, its i-th invariant factor divides the i-th invariant factor s_i of $\lambda I - A$ and the first assertion is proven.

To establish the rest of the theorem we work with the associated block Hankel matrix $\mathrm{Hk}_{e,d}(A, X, Y)$. By definition of the invariant factors we know that

$$\dim \mathrm{span}(X, A^{\mathrm{Tr}}X, (A^{\mathrm{Tr}})^2X, \ldots) \leq \deg s_1 + \cdots + \deg s_{\min\{l,\phi\}}$$

and

$$\dim \mathrm{span}(Y, AY, A^2Y, \ldots) \leq \deg s_1 + \cdots + \deg s_{\min\{m,\phi\}},$$

thus

$$\mathrm{rank}\,\mathrm{Hk}_{e,d}(A, X, Y) \leq \mathrm{rank}\left(\begin{bmatrix} X^{\mathrm{Tr}} \\ X^{\mathrm{Tr}}A \\ X^{\mathrm{Tr}}A^2 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} Y & AY & A^2Y & \ldots \end{bmatrix}\right) \leq \bar{\nu},$$

where $\bar{\nu} = \deg s_1 + \cdots + \deg s_{\min\{m,l,\phi\}}$. Hence, from the specializations W and Z of X and Y given in (Villard 1997b, Corollary 6.4), we get

$$\mathrm{rank}\,\mathrm{Hk}_{e_0,d_0}(A, W, Z) = \max_{X,Y,d,e} \mathrm{rank}\,\mathrm{Hk}_{e,d+1}(A, X, Y) = \bar{\nu} \tag{2.15}$$

with $d_0 = \lceil \bar{\nu}/m \rceil$ and $e_0 = \lceil \bar{\nu}/l \rceil$ and thus $\bar{\nu} = \nu$. Using (2.6) we also have

$$\deg_\lambda(\det F_W^{A,Z}(\lambda)) = \max_{X,Y} \deg_\lambda(\det F_X^{A,Y}(\lambda)) = \bar{\nu}. \tag{2.16}$$

With (2.15) and (2.16) we have proven the two maximality assertions. In addition, since the i-th invariant factor $\bar{s}_i$ of $F_W^{A,Z}$ must divide s_i, the only way to get $\deg_\lambda(\det F_W^{A,Z}) = \nu$ is to take $\bar{s}_i = s_i$ for $1 \leq i \leq \min\{m, l, \phi\}$ and $\bar{s}_i = 1$ for $\min\{m, l, \phi\} < i \leq m$. □

As already noticed, the existence of such W, Z establishes maximality of the matrix generator for symbolic X and Y and, by the Schwartz/Zippel lemma, for random projection matrices. In next sections we will use $\det F_W^{A,Z}(\lambda) = \det(\lambda I - A)$ for computing the determinant and the characteristic polynomial of matrices A with the property $\phi \leq \min\{l, m\}$. For general matrices we will use $F_W^{A,Z}$ to determine the first $\min\{l, m\}$ invariant factors of A.

3. Normal matrix polynomial remainder sequences

As done for a scalar sequence (Brent *et al.* 1980; Dornstetter 1987; Sugiyama *et al.* 1975), the minimum matrix generating polynomial of a sequence can be computed *via* a specialized matrix Euclidean algorithm (Coppersmith 1994; Thomé 2002). Taking advantage of fast matrix multiplication algorithms requires extending these approaches. In Section 3.1 we propose a matrix Euclidean algorithm which combines fast matrix multiplication with the recursive Knuth/Schönhage half-GCD algorithm (von zur Gathen & Gerhard 1999; Knuth 1970; Moenck 1973; Schönhage 1971). This is applicable to computing the matrix minimum polynomial of a sequence $\{X^{\mathrm{Tr}}AY\}_{i\geq 0}$ if the latter leads to a normal matrix polynomial remainder chain. We show in Section 3.2 that this is satisfied, with high probability, by our random integer sequences. This will be satisfied by construction by the sequence in the division-free computation. For simplicity we work in the square case $l = m$, thus with a sequence $\{B^{[i]}\}_{i\geq 0}$ of matrices in $\mathbb{K}^{m\times m}$.

3.1. Minimum polynomials and half Euclidean algorithm. If $F = \sum_{i=0}^{d} F^{[i]}\lambda^i \in \mathbb{K}[\lambda]^{m\times m}$ is a generating matrix polynomial for $\{B^{[i]}\}_{i\geq 0}$ then, as we have seen with (2.5), we have

$$\begin{bmatrix} B^{[0]} & B^{[1]} & \dots & B^{[d]} \\ B^{[1]} & B^{[2]} & \dots & B^{[d+1]} \\ \vdots & \vdots & \ddots & \vdots \\ B^{[d-1]} & B^{[d+1]} & \dots & B^{[2d-1]} \end{bmatrix} \begin{bmatrix} F^{[0]} \\ F^{[1]} \\ \vdots \\ F^{[d]} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3.1}$$

The left side matrix was denoted by $\mathrm{Hk}_{d,d+1}$ in (2.4). We define $\hat{B}$ in $\mathbb{K}[\lambda]^{m\times m}$ by $\hat{B} = \sum_{i=0}^{2d-1} B^{[2d-i-1]}\lambda^i$. Identity (3.1) is satisfied if and only if there exist matrices S and T of degree less than $d-1$ in $\mathbb{K}[\lambda]^{m\times m}$ such that

$$\lambda^{2d}S(\lambda) + \hat{B}(\lambda)F(\lambda) = T(\lambda). \tag{3.2}$$

Thus $\lambda^{2d}I$ and $\hat{B}$ may be considered as the inputs of an extended Euclidean scheme. In the scalar case, the remainder sequence of the Euclidean algorithm is said to be *normal* when at each step the degree is decreased by 1 exactly. By the theorem of subresultants, the remainder sequence is normal if and only if the subresultants are non-zero (Brown & Traub 1971). In an analogous way we will identify normal matrix remainder sequences related to the computation of matrix generating polynomials. We use these remainder sequences to establish a recursive algorithm based on fast matrix polynomial multiplication.

For two matrices $M = \sum_{i=0}^{2d} M^{[i]}\lambda^i$ and $N = \sum_{i=0}^{2d-1} N^{[i]}\lambda^i$ in $\mathbb{K}[\lambda]^{m\times m}$, if the leading matrix $N^{[2d-1]}$ is invertible in $\mathbb{K}^{m\times m}$ then one can divide M by N in an obvious way to get:

$$\text{(3.3)}\qquad \begin{cases} M = NQ + R, \quad \text{with } \deg Q = 1, \deg R \le 2d-2, \\ Q = (N^{[2d-1]})^{-1}(M^{[2d]}\lambda + M^{[2d-1]} - N^{[2d-2]}(N^{[2d-1]})^{-1}M^{[2d]}). \end{cases}$$

If the leading matrix coefficient of R is invertible (matrix coefficient of degree $2d-2$), then the process can be continued. The remainder sequence is normal if all matrix remainders have invertible leading matrices; if so we define:

$$\text{(3.4)}\qquad \begin{cases} M_{-1} = M, \quad M_0 = N, \\ M_i = M_{i-2} - M_{i-1}Q_i, \quad 1 \le i \le d, \end{cases}$$

with $\deg M_i = 2d-1-i$. The above recurrence relations define matrices S_i and F_i in $\mathbb{K}[\lambda]^{m\times m}$ such that

$$\text{(3.5)}\qquad M_{-1}(\lambda)S_i(\lambda) + M_0(\lambda)F_i(\lambda) = M_i(\lambda), \qquad 1 \le i \le d,$$

S_i has degree $i-1$ and F_i has degree i. We also define $S_{-1} = I$, $S_0 = 0$, $F_{-1} = 0$ and $F_0 = I$. As shown below, the choice $M_{-1} = \lambda^{2d}I$ and $M_0 = \hat{B}$ leads to a minimum matrix generating polynomial $F = F_d$ for the sequence $\{B^{[i]}\}_{i\ge 0}$ (compare (3.5) and (3.2)).

THEOREM 3.6. *Let $\hat{B}$ be the matrix polynomial $\sum_{i=0}^{2d-1} B^{[2d-i-1]}\lambda^i \in \mathbb{K}^{m\times m}[\lambda]$. If for all $1 \le k \le d$ we have $\det \mathrm{Hk}_{k,k} \ne 0$, then the matrix half Euclidean algorithm with $M_{-1} = \lambda^{2d}I$ and $M_0 = \hat{B}$ works as announced. In particular:*

(I) *M_i has degree $2d-1-i$ $(0 \le i \le d)$ and its leading matrix $M_i^{[2d-1-i]}$ is invertible $(1 \le i \le d-1)$;*

(II) *F_i has degree i and its leading matrix $F_i^{[i]}$ is invertible $(0 \le i \le d)$; S_i has degree $i-1$ $(1 \le i \le d)$.*

The algorithm produces a minimum matrix generating polynomial $F_d(\lambda)$ for the sequence $\{B^{[i]}\}_{0\le i\le 2d-1}$ and $F = (F_d^{[d]})^{-1}F_d(\lambda)$ is the unique one in Popov normal form.

Furthermore, if in the matrix half Euclidean algorithm the conditions (I)–(II) *are satisfied for all i with $1 \le i \le d$, then $\det \mathrm{Hk}_{k,k} \ne 0$ for all $1 \le k \le d$.*

PROOF. We prove the assertions by induction. For $i = 0$, since by assumption $B^{[0]}$ is invertible, M_0 satisfies (I). By definition $F_0 = I$ and starting at $i = 1$, $S_1 = I$. Now assume that the properties are true for $i - 1$. Then, by (3.3),

$$Q_i = \tilde{Q}_i\lambda + \bar{Q}_i = (M_{i-1}^{[2d-i]})^{-1} M_{i-2}^{[2d-i+1]}\lambda + \bar{Q}_i \ ,$$

$\tilde{Q}_i$ is invertible by (I) at previous steps and $\bar{Q}_i$ is in $\mathbb{K}^{m\times m}$. The leading matrix of F_i is

$$F_i^{[i]} = -F_{i-1}^{[i-1]}\tilde{Q}_i,$$

and thus F_i satisfies (II). The same argument holds for S_i ($i - 1 \geq 1$). By construction M_i has a degree lower than $2d - 1 - i$, and hence, looking at the right side coefficient matrices of (3.5), we know that

$$\underbrace{\begin{bmatrix} B^{[0]} & B^{[1]} & \dots & B^{[i]} \\ B^{[1]} & B^{[2]} & \dots & B^{[i+1]} \\ \vdots & \vdots & \ddots & \vdots \\ B^{[i]} & B^{[i+1]} & \dots & B^{[2i]} \end{bmatrix}}_{\mathrm{Hk}_{i+1,i+1}} \begin{bmatrix} F_i^{[0]} \\ F_i^{[1]} \\ \vdots \\ F_i^{[i]} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ M_i^{[2d-1-i]} \end{bmatrix}. \tag{3.7}$$

By assumption of non-singularity of $\mathrm{Hk}_{i+1,i+1}$ and since we have proved that $F_i^{[i]}$ is invertible, the columns in the right side matrix of (3.7) are linearly independent, thus $M_i^{[2d-1-i]}$ is invertible. This proves (I). Identity (3.5) for $i = d$ also establishes (3.1), which means that F_d is a matrix generating polynomial for $\{B^{[i]}\}_{0\leq i\leq 2d-1}$ whose leading matrix $F_d^{[d]}$ its invertible. It follows that $F = (F_d^{[d]})^{-1}F_d(\lambda)$ is in Popov normal form. The minimality comes from the fact that $\mathrm{Hk}_{d,d}$ is invertible and hence no vector generator (column of a matrix generator) can be of degree less than d.

We finally prove that invertible leading coefficient matrices in the Euclidean algorithm guarantee non-singularity for all $\mathrm{Hk}_{k,k}$. To that end, we consider the range of $\mathrm{Hk}_{i+1,i+1}$ in (3.7). Clearly, the block vector $[0\ I_m]^{\mathrm{Tr}}$ is in the range, since $M_i^{[2d-1-i]}$ is invertible. By induction hypothesis for $\mathrm{Hk}_{i,i}$, we see that the first i block columns of $\mathrm{Hk}_{i+1,i+1}$ can generate $[I_{mi}\ 0]^{\mathrm{Tr}}$, where the block zero row at the bottom is achieved by subtraction of appropriate linear combinations of the previous block vector $[0\ I_m]^{\mathrm{Tr}}$. Hence the range of $\mathrm{Hk}_{i+1,i+1}$ has full dimension. □

For $B^{[i]} = X^{\mathrm{Tr}}AY$, $i \geq 0$, the next corollary shows that F is as expected.

COROLLARY 3.8. *Let A be in $\mathbb{K}^{n\times n}$, let $B^{[i]} = X^{\mathrm{Tr}}A^iY \in \mathbb{K}^{m\times m}$, $i \geq 0$, and let $\nu = md$ be the determinantal degree $\deg_\lambda(\det F_X^{A,Y})$. If the block Hankel matrix $\mathrm{Hk}_{d,d}(A,X,Y)$ satisfies the assumption of Theorem 3.6 then $F = F_X^{A,Y}$.*

PROOF. We know from (2.6) that ν is the maximum possible rank for the block Hankel matrices associated to the sequence, thus the infinite one $\mathrm{Hk}_{\infty,d+1}$ satisfies

$$\operatorname{rank}\mathrm{Hk}_{\infty,d+1} = \operatorname{rank}\begin{bmatrix} B^{[0]} & B^{[1]} & \dots & B^{[d]} \\ B^{[1]} & B^{[2]} & \dots & B^{[d+1]} \\ \vdots & \vdots & & \vdots \end{bmatrix} = \operatorname{rank}\mathrm{Hk}_{d,d+1} = \nu.$$

It follows that $\mathrm{Hk}_{\infty,d+1}$ and $\mathrm{Hk}_{d,d+1}$ have the same nullspace, and F, which by Theorem 3.6 is a matrix generator for the truncated sequence $\{B^{[i]}\}_{0\leq i\leq 2d-1}$, is a generator for the whole sequence. The argument used for the minimality of F remains valid, hence $F = F_X^{A,Y}$. □

REMARK 3.9. In Theorem 3.6 and Corollary 3.8 we have only addressed the case where the target determinantal degree is an exact multiple md of the blocking factor m. This can be assumed with no loss of generality for the algorithms in Sections 4 and 5 and the corresponding asymptotic costs in Section 6. Indeed, we will work there with $\nu = n$ and the input matrix A may be padded to $\mathrm{diag}(A, I)$.

In the general case or in practice to avoid padding, the Euclidean algorithm leads to $\operatorname{rank} M_{d-1}^{[d]} = \nu \bmod m \leq m$ and requires a special last division step. The minimum generator $F = F_X^{A,Y}$ has degree $d = \lceil \nu/m \rceil$, with column degrees $[\delta_1, \dots, \delta_m] = [d-1, \dots, d-1, d, \dots, d]$, where $d-1$ is repeated $m\lceil \nu/m \rceil - \nu$ times (Villard 1997b, Proposition 6.1).

The above method can be combined with the recursive Knuth (1970)/Schönhage (1971)/Moenck (1973) algorithm. If ω is the exponent of matrix multiplication then, as soon as the block Hankel matrix has the required rank profile, $F_X^{A,Y}$ may be computed with $(n^\omega d)^{1+o(1)}$ operations in $\mathbb{K}$. The required FFT-based multiplication algorithms for matrix polynomials are described by Cantor and Kaltofen (1991).

3.2. Normal matrix remainder sequences over the integers. The normality of the remainder sequence associated to a given matrix A essentially comes from the genericity of the projections. This may be partly seen in the scalar case for Lanczos algorithm from (Eberly & Kaltofen 1997, Lemma 4.1),

(Eberly 2002) or (Kaltofen & Lee 2003; Kaltofen *et al.* 2000) and in the block case from (Kaltofen 1995, Proposition 3) or (Villard 1997b, Proposition 6.1).

We show here that the block Hankel matrix has generic rank profile for generic projections, and then the integer case follows by randomization. We let $\mathcal{X}$ and $\mathcal{Y}$ be two $n \times m$ matrices with indeterminate entries $\xi_{i,j}$ and $\upsilon_{i,j}$ for $1 \leq i \leq n$ and $1 \leq j \leq m$. Let also ν be the maximum determinantal degree defined by (2.13) in Theorem 2.12.

LEMMA 3.10. *With $d = \lceil \nu/m \rceil$, the block Hankel matrix $\mathrm{Hk}_{d,d}(A, \mathcal{X}, \mathcal{Y})$ has rank ν and its principal minors of order i are non-zero for $1 \leq i \leq \nu$.*

PROOF. To simplify the presentation we only detail the case where ν is a multiple of m (see Remark 3.9). Let $\mathrm{Kr}_i(A, Z) \in \mathbb{K}^{n \times i}$ be the block Krylov matrix formed by the i first columns of $[Z \; AZ \; \ldots \; A^{d-1}Z]$ for $1 \leq i \leq \nu$. The specialization $Z \in \mathbb{K}^{n \times m}$ of $\mathcal{Y}$ given in (Villard 1997b, Proposition 6.1) satisfies

$$\operatorname{rank} \mathrm{Kr}_i(A, Z) = i, \quad 1 \leq i \leq \nu. \tag{3.11}$$

We now argue, by specializing $\mathcal{X}$ and $\mathcal{Y}$, that the target principal minors are non-zero. If $i \leq m$, using (3.11) one can find $X \in \mathbb{K}^{n \times i}$ such that the rank of $X^{\mathrm{Tr}}\mathrm{Kr}_i(A, Z)$ equals i. If $m < i \leq \nu$ then one can find $X \in \mathbb{K}^{n \times m}$ such that $X^{\mathrm{Tr}}\mathrm{Kr}_i(A, Z) = [0 \; J_m]$, where J_m is the $m \times m$ reversion matrix. Hence $\mathrm{Hk}_{d,d}(A, X, Z)$ has ones on its i-th anti-diagonal and zeros above, and the corresponding principal minor of order i is $(-1)^{\lfloor i/2 \rfloor}$. □

The polynomial $\prod_{k=1}^{d} \det \mathrm{Hk}_{k,k}(A, \mathcal{X}, \mathcal{Y})$ is non-zero of degree no more than $md(d+1)$ in $\mathbb{K}[\ldots, \xi_{i,j}, \ldots, \upsilon_{i,j}, \ldots]$. If the entries of X and Y are chosen uniformly and independently from a finite set $S \subset \mathbb{Z}$ then, by the Schwartz/Zippel lemma and Theorem 3.6, the associated matrix remainder sequence is normal with probability at least $1 - md(d+1)/|S|$.

4. The block baby steps/giant steps determinant algorithm

We shall present our algorithm for integer matrices. Generalizations to other domains, such as polynomial rings, are certainly possible. The algorithm follows the Wiedemann paradigm (Wiedemann 1986, Chapter V) and uses a baby steps/giant steps approach for computing the sequence elements (Kaltofen 1992). In addition, the algorithm blocks the projections (Coppersmith 1994). A key ingredient is that from the theory of realizations described in Section 2, it is possible to recover the characteristic polynomial of a preconditioning of the input matrix.

Algorithm *Block Baby Steps/Giant Steps Determinant.*
Input: a matrix $A \in \mathbb{Z}^{n\times n}$.
Output: an integer that is the determinant of A, or "failure;" the algorithm fails with probability no more than 1/2.

Step 0. Let $h = \log_2 \mathrm{Hd}(A)$, where $\mathrm{Hd}(A)$ is a bound on the magnitude of the determinant of A, for instance, Hadamard's bound (see, for example, von zur Gathen & Gerhard 1999). To guarantee the probability of a successful completion, the algorithm uses positive constants $\gamma_1, \gamma_1' \geq 1$.
Choose a random prime integer $p_0 \leq \gamma_1' h^{\gamma_1}$ and compute $\det A \bmod p_0$ by LU-decomposition over $\mathbb{Z}_{p_0}$.
If the result is zero, A is most likely singular, and the algorithm calls an algorithm for computing $x \in \mathbb{Z}^n \setminus \{0\}$ with $Ax = 0$; see Remark 4.7 on page 115 below. Note that the following steps would fail, for example, to certify the determinant of the zero matrix.

Step 1. Precondition A in such a way that with high probability $\det(\lambda I - A) = s_1(\lambda) \cdots s_{\min\{m,\phi\}}$, where $s_1, \ldots, s_\phi$ are the invariant factors of $\lambda I - A$ and where m is the blocking factor that will be chosen in Step 2. 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 (Chen *et al.* 2002, Theorem 4.3). The second by Turner (2001) is $A \leftarrow EA$, where

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

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 below for the first preconditioner. Both preconditioners achieve $s_1(\lambda) = \det(\lambda I - A)$ with probability $1 - O(n^2/|S|)$. Note that A is non-singular. We shall choose $S = \{i \mid -\lfloor \gamma_2' n^{\gamma_2} \rfloor \leq i \leq \lceil \gamma_2' n^{\gamma_2} \rceil\}$, where $\gamma_2 \geq 2$ and $\gamma_2' \geq 1$ are real constants.

Step 2. Let the blocking factors be $l = m = \lceil n^\sigma \rceil$, where $\sigma = 1/3$.
Select random $X, Y \in S^{n\times m}$.
We will compute the sequence $B^{[i]} = X^{\mathrm{Tr}} A^i Y$ for all $0 \leq i < \lceil 2n/m \rceil = O(n^{1-\sigma})$ by utilizing our baby steps/giant steps technique (Kaltofen 1992).

Let the number of giant steps be $s = \lceil n^\tau \rceil$, where $\tau = 1/3$, and let the number of baby steps be $r = \lceil 2\lceil n/m \rceil / s \rceil = O(n^{1-\sigma-\tau})$.

Substep 2.1. For $j = 0, 1, \ldots, r-1$ Do $V^{[j]} \leftarrow A^j Y$;

Substep 2.2. $Z \leftarrow A^r$;

Substep 2.3. For $k = 1, 2, \ldots, s-1$ Do $(U^{[k]})^{\mathrm{Tr}} \leftarrow X^{\mathrm{Tr}} Z^k$;

Substep 2.4. For $j = 0, 1, \ldots, r-1$ Do
For $k = 0, 1, \ldots, s-1$ Do $B^{[kr+j]} \leftarrow (U^{[k]})^{\mathrm{Tr}} V^{[j]}$.

Step 3. Compute the minimum matrix generator $F_X^{A,Y}(\lambda)$ from the initial sequence segment $\{B^{[i]}\}_{0 \le i < 2\lceil n/m \rceil}$. Here we can use the method from Section 3, padding the matrix so that m divides n (see Remark 3.9 on page 107), and return "failure" whenever the coefficient $F_i^{[i]}$ of the matrix remainder polynomial is singular. For alternative methods, we refer to Remark 4.1 below the algorithm.

Step 4. If $\deg(\det F_X^{A,Y}) < n$ return "failure" (this check may be redundant, depending on which method was used in Step 3). Otherwise, since $F_X^{A,Y}(\lambda)$ is in Popov form we know that its determinant is monic and by Theorem 2.12 we have $\det F_X^{A,Y}(\lambda) = \det(\lambda I - A)$. Return $\det A = \Delta(0)$, or a value adjusted according to the preconditioner used in Step 1. □

Remark 4.1. As we have seen in Section 2.1 there are several alternatives for carrying out Step 3 (Beckermann & Labahn 1994; Coppersmith 1994; Dickinson *et al.* 1974; Forney, Jr. 1975; Giorgi *et al.* 2003; Kaltofen 1995; Rissanen 1972; Thomé 2002; Van Barel & Bultheel 1992). In Step 4 we require that $\det F_X^{A,Y}(\lambda) = \det(\lambda I - A)$. In order to achieve the wanted bit complexity, we must stop any of the algorithms after having processed the first $2\lceil n/m \rceil$ elements of (2.1). The algorithm used must then return a candidate matrix polynomial $\widetilde{F}$. Clearly, if Step 4 exposes $\deg(\det \widetilde{F}) < n$ one knows that the randomizations were unlucky. However, if $\deg(\det \widetilde{F}) = n$ there still may be the possibility that $\widetilde{F} \neq F_X^{A,Y}$ due to a situation where the first $2\lceil n/m \rceil$ elements do not determine the generator, as would be the case in the two examples given in Section 2. In order to achieve the Las Vegas model of randomized algorithmic complexity, verification of the computed generator is thus necessary here. For example, the algorithm used could do so by establishing that $\mathrm{rank}\, \mathrm{Hk}_{\lceil n/m \rceil, \lceil n/m \rceil}(A, X, Y) = n$. Our algorithm from Section 3 implicitly does

so via Theorem 3.6 on page 105. One could do so explicitly by computing the rank of $\mathrm{Hk}_{\lceil n/m\rceil,\lceil n/m\rceil}$ modulo a random prime number.

We remark that the *arithmetic* cost of verifying that the candidate for $F_X^{A,Y}$ is a generator for the block Krylov sequence $\{A^iY\}_{i\geq 0}$ is the same as Step 2. The reduction is seen by applying the transposition principle (Kaltofen 2000, Section 6): note that computing all $B^{[i]}$ amounts to computing the block diagonal left product

$$\left[(X^{\mathrm{Tr}})_{1,*} \mid (X^{\mathrm{Tr}})_{2,*} \mid \ldots\right] \cdot \begin{bmatrix} \ldots A^iY \ldots & 0 & 0 & \cdots & 0 \\ 0 & \ldots A^iY \ldots & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & & \ldots A^iY \ldots \end{bmatrix},$$

where $(X^{\mathrm{Tr}})_{i,*}$ denotes the i-th row of X^{Tr}. Computing $\sum_i A^iYc^{[i]}$, where $c^{[i]} \in \mathbb{K}^{m\times m}$ are the coefficients of $F_X^{A,Y}$, amounts to computing the block diagonal right product

$$\begin{bmatrix} \ldots A^iY \ldots & 0 & 0 & \cdots & 0 \\ 0 & \ldots A^iY \ldots & 0 & \cdots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & & \ldots A^iY \ldots \end{bmatrix} \cdot \begin{bmatrix} (c^{[0]})_{*,1} \\ (c^{[1]})_{*,1} \\ \vdots \\ (c^{[0]})_{*,2} \\ (c^{[1]})_{*,2} \\ \vdots \end{bmatrix},$$

where $(c^{[i]})_{*,j}$ denotes the j-th column of the matrix $c^{[i]}$. One may also develop an explicit baby steps/giant steps algorithm for computing $\sum_i A^iYc^{[i]}$. However, because the integer lengths of the entries in $c^{[i]}$ are much larger than those of X and Y, we do not know how to keep the bit complexity low enough to allow verification of the candidate generator via verification as a block Krylov space generator.

We shall first give the bit complexity analysis for our block algorithm under the assumption that no subcubic matrix multiplication à la Strassen or subquadratic block Toeplitz solver/greatest common divisor algorithm à la Knuth/Schönhage is employed. We will investigate those best theoretically possible running times in Section 6.

THEOREM 4.2. *Our algorithm computes the determinant of any non-singular matrix $A \in \mathbb{Z}^{n\times n}$ with $(n^{3+1/3}\log\|A\|)^{1+o(1)}$ bit operations. Our algorithm utilizes $(n^{1+1/3} + n\log\|A\|)^{1+o(1)}$ random bits and either returns the correct determinant or returns "failure," the latter with probability of no more than* $1/2$.

In our analysis, we will use modular arithmetic. The following lemma will be used to establish the probability of getting a good reduction with prime moduli.

LEMMA 4.3. *Let $\gamma \geq 1$ and $\gamma' \geq 1$ be real constants. Then for all integers $H \in \mathbb{Z}_{\geq 2}$ that with $h = 2\log_e H \leq 1.89 \log_2 H$ satisfy $10 \leq h$, $h \notin [113, 113.6]$ and $\gamma' \leq h^\gamma$, we have the probability estimate*

$$\text{Prob}(p \text{ divides } H \mid p \text{ a prime integer}, 2 \leq p \leq \gamma' h^\gamma) \leq \frac{25}{8} \frac{\gamma}{\gamma' h^{\gamma-1}}. \tag{4.4}$$

PROOF. We have the following estimates for the distribution of prime numbers:

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

where C_1, C_2 and C_3 are positive constants. Explicit values for C_1, C_2 and C_3 have been derived. We may choose $C_1 = 0.5$ for $x \geq 10$ (Rosser & Schoenfeld 1962, Theorem 10 + explicit estimation for $10 \leq x < 101$), $C_2 = 0.8$ for $x \geq 5$ (Rosser & Schoenfeld 1962, Corollary 1 to Theorem 2 and explicit estimation for $10 \leq x < 17$), and $C_3 = 1.25$ for $x < 113$ and $x \geq 113.6$ (Rosser & Schoenfeld 1962, Corollary 2 to Theorem 2).

Since we have $\prod_{p \leq h} p > e^{C_1 h} = H$, there are at most $\pi(h) < C_3 h / \log_e h$ distinct prime factors in H. The number of primes $\leq \gamma' h^\gamma$ is more than $C_2 \gamma' h^\gamma / (\gamma \log_e h + \log_e \gamma')$, because from our assumptions we have $\gamma' h^\gamma \geq 10$. Therefore the probability for a random p to divide H is no more than, using $\log_e \gamma' \leq \gamma \log_e h$,

$$\frac{C_3 h / \log_e h}{C_2 \gamma' h^\gamma / (\gamma \log_e h + \log_e \gamma')} \leq \frac{C_3 h / \log_e h}{C_2 \gamma' h^\gamma / (2\gamma \log_e h)} \leq \frac{2C_3}{C_2} \frac{\gamma}{\gamma' h^{\gamma-1}}. \qquad \square$$

In the above Lemma 4.3 we have introduced the constant γ' so that it is possible to choose $\gamma = 1$ and have a positive probability of avoiding a prime divisor of H.

PROOF OF THEOREM 4.2. The unblocked version of the algorithm is fully analyzed by Kaltofen (2002) with the additional modification of early termination when the determinant is small. That analysis uses a residue number system (Chinese remaindering) for representing long integers, which we adopt for the blocked algorithm. This adds the bit cost of generating a stream of sufficiently large random primes (including p_0 in Step 0).

Step 0 has by $h = O(n \log(n\|A\|))$, which follows from Hadamard's bound, the bit complexity $(n^3 + n^2 \log \|A\|)^{1+o(1)}$, the latter term accounting for taking every entry of A modulo p_0. The failure probability of Step 0, that is when $\det A \equiv 0 \pmod{p_0}$ for non-singular A, is bounded by Lemma 4.3. Thus, for $H = \det A$ and appropriate choice of γ_1 and γ_1' in Step 0 all non-singular matrices will pass with probability no less than 9/10.

Step 1 increases $\log \|DA\|$ or $\log \|EA\|$ to no more than $O((\log n)^2 \log \|A\|)$ and has bit cost $(n^3 \log \|A\|)^{1+o(1)}$.

Steps 3 and 4 are performed modulo sufficiently many primes p_l so that $\det A$ can be recovered via Chinese remaindering. Using $p_l \geq 2$, we obtain the very loose count

$$1 \leq l \leq 2 \log_2(\mathrm{Hd}\, A) = 2h = O(n \log(n\|A\|)), \tag{4.5}$$

the factor 2 accounting for recovery of negative determinants. Modular arithmetic becomes necessary to avoid length growth in the scalars in $F_X^{A,Y}$ during Steps 3 and 4. We shall first estimate the probability of success, and then the bit complexity. The probabilistic analysis will also determine the size of the prime moduli.

The algorithm fails if:

(I) The preconditioners D or E in Step 1 do not yield $\det(\lambda I - A) = s_1(\lambda) \cdots s_{\min\{m,\phi\}}$, that with probability $\leq O(1/n^{\gamma_2 - 2})$. As for Step 0, we select the constant γ_2, γ_2' so that the preconditioners fail with probability $\leq 1/10$.

(II) The projections X, Y in Step 2 do not yield $\mathrm{rank}\, \mathrm{Hk}_{\lceil n/m \rceil, \lceil n/m \rceil}(A, X, Y) = n$. Since for $X = \mathcal{X}$ and $Y = \mathcal{Y}$ with variables $\xi_{i,j}$, $\upsilon_{i,j}$ as entries full rank is achieved (see Section 2), we can consider an $n \times n$ non-singular submatrix $\Gamma(\mathcal{X}, \mathcal{Y})$ of $\mathrm{Hk}_{\lceil n/m \rceil, \lceil n/m \rceil}(A, \mathcal{X}, \mathcal{Y})$. By (DeMillo & Lipton 1978; Schwartz 1980; Zippel 1979) we get

$$\mathrm{Prob}(\det \Gamma(X, Y) = 0 \mid X, Y \in S^{n \times m}) \leq \frac{\deg(\det \Gamma)}{|S|} \leq \frac{2n}{|S|} \leq \frac{1}{\gamma_2' n^{\gamma_2 - 1}}.$$

If we use the matrix polynomial remainder sequence algorithm of Section 3 for Step 3, we also fail if $\prod_{1 \leq k < \lceil n/m \rceil} \det \mathrm{Hk}_{k,k}(A, X, Y) = 0$, that with probability no more than $n(n/m + 1)/|S| \leq (n^{1-\sigma} + 1)/(2\gamma_2' n^{\gamma_2 - 1})$.

Again, the constants γ_2, γ_2' are chosen so that the probability is $\leq 1/10$.

(III) The computation modulo one of the moduli p_l fails for Step 3 or 4. Then p_l divides $\det \Gamma(A, X, Y)$. Since $\log |\det(\Gamma(A, X, Y))| = ((n^2/m) \times \log \|A\|)^{1+o(1)}$, we may select the random moduli in the range

$$2 \leq p_l \leq \gamma_3'(n^{2-\sigma} \log \|A\|)^{(1+o(1))\gamma_3} = q. \tag{4.6}$$

where $\sigma = 1/3$ and $\gamma_3 \geq 2, \gamma_3' \geq 1$ are constants. Note that in (4.6) the exponent $1 + o(1)$ captures derivable polylogarithmic factors $C_1(\log n)^{C_2}(\log \|A\|)^{C_3}$, where C_1, C_2, C_3 are explicit constants. By Lemma 4.3 the probability that any one of the $\leq 2h$ moduli fails, i.e. divides $\det \Gamma(A, X, Y)$, is no more than $2h/(n^{2-\sigma} \log \|A\|)^{(1+o(1))(\gamma_3-1)}$. By the Hadamard estimate (4.5) we can make this probability no larger than $1/10$ via selecting the constants γ_3, γ_3' sufficiently large.

If we must also avoid divisors of $\prod_{1 \leq k < \lceil n/m \rceil} \det \mathrm{Hk}_{k,k}(A, X, Y)$ for the matrix polynomial remainder sequence algorithm, the range (4.6) increases to $p_l \leq \gamma_3'(n^{3-2\sigma} \log \|A\|)^{(1+o(1))\gamma_3}$.

(IV) The algorithms fails to compute sufficiently many random prime moduli $p_l \leq q$ (see (4.6)). There is now a deterministic algorithm of bit complexity $(\log p_l)^{12+o(1)}$ for primality testing (Agrawal *et al.* 2002), which is not required but simplifies the theoretical analysis here. We pick $k = 4h \log q$ positive integers $\leq q$. The probability for each to be prime is $\geq 1/\log q = \psi$ (provided $q \geq 17$; Rosser & Schoenfeld 1962). By Chernoff bounds for the tail of the binomial distribution, the probability that fewer than $2h = (1 - 1/2)\psi k$ are prime is $\leq e^{-(1/2)^2 \psi k/2} = 1/e^{h/2}$. Thus for $h \geq 5$ the probability of failing to find $2h$ primes is $\leq 1/10$.

Cases (I)–(IV) together with Step 0 add up to a failure probability of $\leq 1/2$. We conclude by estimating the number of bit operations for Steps 2–4.

Step 2 computes $B^{[i]} \bmod p_l$ for $0 \leq i < 2\lceil n/m \rceil$ and $1 \leq l \leq 2h$ as follows. First, all $B^{[i]}$ are computed as exact integers. For Substeps 2.1 and 2.2, that requires $O(n^3 \log r)$ arithmetic operations on integers of length $(r \log \|A\|)^{1+o(1)}$, in total $(n^{4-\sigma-\tau} \log \|A\|)^{1+o(1)}$ bit operations (recall that $\sigma = \tau = 1/3$). Substeps 2.3 and 2.4 require $O(smn^2)$ arithmetic operations on integers of length $(rs \log \|A\|)^{1+o(1)}$, again $(n^{3+\tau} \log \|A\|)^{1+o(1)}$ bit operations. Then all $O((n/m)m^2)$ entries of all $B^{[i]}$ are taken modulo p_l with l in the range (4.5) and p_l in (4.6). Straightforward remaindering would yield a total of $(nmhrs \log \|A\|)^{1+o(1)}$ bit operations, which is $(n^3(\log \|A\|)^2)^{1+o(1)}$. The complexity can be reduced to $(n^3 \log \|A\|)^{1+o(1)}$ via a tree evaluation scheme (Aho *et al.* 1974; Heindel & Horowitz 1971, Algorithm 8.4).[2]

Steps 3 and 4 are performed modulo all $O(h)$ prime moduli p_l. For each prime the cost of extended Euclidean algorithm on matrix polynomials is $O(m^3(n/m)^2)$ residue operations. Overall, the bit complexity of Steps 3 and 4

[2]Note that this speedup comes at a cost of an extra log-factor.

is again $(n^{3+\sigma} \log \|A\|)^{1+o(1)}$. The number of required random bits in D or E, X and Y, and case (IV) above is immediate. □

It is possible to derive explicit values for the constants γ_1, γ_1', γ_2, γ_2', γ_3, and γ_3' so that Theorem 4.2 holds. However, any implementation of the algorithm would select reasonably small values. For example, all prime moduli would be chosen 32 or 64 bit in length. Since the method is Las Vegas, such choice only affects the probability of not obtaining a result.

If Step 3 uses a Knuth/Schönhage half-GCD approach with FFT-based polynomial arithmetic for the Euclidean algorithm on matrix polynomials of Section 3, the complexity for each modulus reduces to $(m^2 n)^{1+o(1)}$ residue operations. Thus, the overall complexity of Steps 3 and 4 reduces to $(n^{2+2\sigma} \times \log \|A\|)^{1+o(1)}$ bit operations. For $\sigma = 3/5$ and $\tau = 1/5$ the bit complexity of the algorithm is then $(n^{3+1/5} \log \|A\|)^{1+o(1)}$.

REMARK 4.7. In order to state a Las Vegas bit complexity for the determinant of a general square matrix, we need to consider the cost of certifying singularity in Step 0 on page 109 above. In order to meet the complexity of Theorem 4.2 on page 111 above we can use the algorithm by Dixon (1982). Reduction to a non-singular subproblem can be accomplished by methods of Kaltofen and Saunders (1991), and the rank is determined in a Monte Carlo manner via a random prime modulus; see also (Villard 1988, p. 102).

5. Improved division-free complexity

Our baby steps/giant steps algorithm with blocking of Section 4 can be employed to improve Kaltofen's (1992) division-free complexity of the determinant (see also Seifullin 2003). Here we consider a matrix $A \in R^{n \times n}$, where R is a commutative ring with a unit element. At task is to compute the determinant of A by ring additions, subtractions and multiplications. Blocking can improve the number of ring operations from $n^{3.5+o(1)}$ (Kaltofen 1992) to $n^{3+1/3+o(1)}$, that without subcubic matrix multiplication or subquadratic Toeplitz/GCD algorithms, and best possible from $O(n^{3.0281})$ (Kaltofen 1992)[3] to $O(n^{2.6973})$. Our algorithm combines the blocked determinant algorithm with the elimination of divisions technique of Kaltofen (1992). Our computational model is either a straight-line program/arithmetic circuit or an algebraic random access machine (Kaltofen 1988). Further problems are to compute the characteristic polynomial and the adjoint matrix of A.

[3]The proceedings paper gives an exponent 3.188; the smaller exponent is in a postnote added to the version posted on `www.kaltofen.us/bibliography`.

The main idea of Kaltofen (1992) follows Strassen (1973) and for the input matrix A computes the determinant of the polynomial matrix $L(z) = C + z(A - C)$, where $C \in \mathbb{Z}^{n\times n}$ is a special integral matrix whose entries are independent of the entries in A (see below). For $\Delta(z) = \det L(z)$ we have $\det A = \Delta(1)$. All intermediate elements are represented as polynomials in $R[z]$ or as truncated power series in $R[[z]]$ and the "shift" matrix C determines them in such a manner that whenever a division by a polynomial or truncated power series is performed the constant coefficients are ± 1. For the algorithm in Section 4 we not only pick a generalized shift matrix, now denoted by M, but also concrete projection block vectors $X \in \mathbb{Z}^{n\times m}$ and $Y \in \mathbb{Z}^{n\times m}$. No randomization is necessary, as M is a "good" input matrix ($\phi = m$) and X and Y are "good" projections, we have $\det F_X^{L(z),Y}(\lambda) = \det(\lambda I - L(z))$.

The matrices M, X and Y are block versions of the ones constructed by Kaltofen (1992). Suppose that the blocking factor m is a divisor of n, the dimension of A. This we can always arrange by padding A to $\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}$. Let $d = n/m$ and

$$a_i = \binom{i}{\lfloor i/2 \rfloor}, \quad c_i = -(-1)^{\lfloor (d-i+1)/2 \rfloor} \binom{\lfloor (d+i)/2 \rfloor}{i},$$

$$C = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & & 0 & 1 \\ c_0 & c_1 & \dots & c_{d-2} & c_{d-1} \end{bmatrix}, \quad v = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \\ a_{d-1} \end{bmatrix}.$$

We have shown (Kaltofen 1992) that for the sequence $a_i = e_1^{\mathrm{Tr}} C^i v$, where $e_1^{\mathrm{Tr}} = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \in \mathbb{Z}^{1\times d}$ is the first d-dimensional unit (row) vector, the Berlekamp/Massey algorithm divides by only ± 1. We now define

$$M = \begin{bmatrix} C & 0 & \dots & 0 \\ 0 & C & \ddots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & C \end{bmatrix} \in \mathbb{Z}^{n\times n},$$

$$X = \begin{bmatrix} e_1 & 0 & \dots & 0 \\ 0 & e_1 & \ddots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & & e_1 \end{bmatrix} \in \mathbb{Z}^{n\times m}, \quad Y = \begin{bmatrix} v & 0 & \dots & 0 \\ 0 & v & \ddots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & & v \end{bmatrix} \in \mathbb{Z}^{n\times m}.$$

By construction, the algorithm for computing the determinant of Section 4 performed now with the matrices X, M, Y results in a minimum matrix generator

$$F_X^{M,Y}(\lambda) = (\lambda^d - c_{d-1}\lambda^{d-1} - \cdots - c_0)I_m,$$

where I_m is the $m \times m$ identity matrix. Furthermore, this generator can be computed from the sequence of block vectors $B^{[i]} = a_i I_m$ by a matrix Euclidean algorithm (see Section 3) in which all leading coefficient matrices are equal to $\pm I_m$.

The arithmetic cost for executing the block baby steps/giant steps algorithm on the polynomial matrix $L(z) = M + z(A - M)$ is related to the bit complexity of Section 4. Now the intermediate lengths are the degrees in z of the computed polynomials in $R[z]$. Therefore, the matrices $X^{\mathrm{Tr}}L(z)^i Y \in R[z]^{m\times m}$ can be computed for all $0 \le i < 2d$ in $n^{3+1/3+o(1)}$ ring operations. In the matrix Euclidean algorithm for Step 3 we perform truncated power series arithmetic modulo z^{n+1}. The arithmetic cost is $(d^2m^3n)^{1+o(1)}$ ring operations for the classical Euclidean algorithm with FFT-based power series arithmetic. For the latter, we employ a division-free FFT-based polynomial multiplication algorithm (Cantor & Kaltofen 1991). Finally, to obtain the characteristic polynomial, we may slightly extend Step 4 on page 110 and compute the entire determinant of $F_X^{L(z),Y}(\lambda)$ division-free in truncated power series arithmetic over $R[z,\lambda]$ mod (z^{n+1}, λ^{n+1}) (a different approach is given at the end of Section 6). For this last step we can use our original division-free algorithm (Kaltofen 1992) and achieve arithmetic complexity $(m^{3.5}n^2)^{1+o(1)}$. We have proven the following theorem.

THEOREM 5.1. *Our algorithm computes the characteristic polynomial of any matrix $A \in R^{n\times n}$ with $(n^{3+1/3})^{1+o(1)}$ ring operations in R. By the results of Baur and Strassen (1983) the same complexity is obtained for the adjoint matrix, which can be symbolically defined as* $\det(A)A^{-1}$.

6. Using fast matrix multiplication

As stated in the Introduction, by use of subcubic matrix multiplication algorithms the worst case bit complexity of the block algorithms in Sections 4 and 5 can be brought below cubic complexity in n. We note that taking the n^2 entries of the input matrix modulo n prime residues is already a cubic process in n; our algorithms therefore proceed differently.

Now let ω by the exponent for fast matrix multiplication. By Coppersmith and Winograd (1990) we may set $\omega = 2.375477$. The considerations in this section are of a purely theoretical nature.

Substep 2.1 in Section 4 is done by repeated doubling as in

$$\left[A^{2^\mu}Y\ A^{2^\mu+1}Y\ldots\ A^{2^{\mu+1}-1}Y\right] = A^{2^\mu}\left[Y\ AY\ldots\ A^{2^\mu-1}Y\right] \quad \text{for } \mu = 0, 1, \ldots.$$

Therefore the bit complexity for Substeps 2.1 and 2.2 is $(n^\omega r \log \|A\|)^{1+o(1)}$ with an exponent $\omega + 1 - \sigma - \tau$ for n. Note that σ and τ determine the blocking factor and number of giant steps, and will be chosen later so as to minimize the complexity.

Substep 2.3 both splits the integer entries in $U^{[k]}$ into chunks of length $(r \log \|A\|)^{1+o(1)}$, which is the bit length of the entries in Z. There are at most $s^{1+o(1)}$ such chunks. Thus each block vector times the matrix product $(U^{[k]})^{\mathrm{Tr}}Z$ is a rectangular matrix product of dimensions $(ms)^{1+o(1)} \times n$ by $n \times n$. We now appeal to fast methods for rectangular matrices (Coppersmith 1997) (we seem not to need the results of Huang & Pan 1998), which show how to multiply an $n \times n$ matrix by an $n \times \nu$ matrix in $n^{\omega-\theta+o(1)}\nu^{\theta+o(1)}$ arithmetic operations (by blocking the $n \times n$ matrix into $(t \times t)$-sized blocks and the $n \times \nu$ matrix into $(t \times t^\zeta)$-sized blocks such that $n/t = \nu/t^\zeta$ and that the individual block products only take $t^{2+o(1)}$ arithmetic steps each), where $\theta = (\omega - 2)/(1 - \zeta)$ with $\zeta = 0.2946289$. There are s such products on integers of length $(r \log \|A\|)^{1+o(1)}$, so the bit complexity for Substep 2.3 is $(sn^{\omega-\theta}(ms)^\theta r \log \|A\|)^{1+o(1)}$ with an exponent $\omega + 1 - \sigma + (\sigma + \tau - 1)\theta$ for n.

Step 3 for each individual modulus can be performed by the method presented in Section 3 in $(m^\omega n/m)^{1+o(1)}$ residue operations. For all $\leq 2h$ moduli we get a total bit complexity for Step 3 of $(m^{\omega-1}n^2 \log \|A\|)^{1+o(1)}$ with an exponent $2 + \sigma(\omega - 1)$ for n.

The bit complexities of Substep 2.4 and Step 4 are dominated by the complexities of other steps.

All of the above bit costs lead to total bit complexity of $(n^\eta \log \|A\|)^{1+o(1)}$, where the exponent η depends on the use matrix multiplication exponents ω and ζ. Table 6.1 displays the optimal values of η for selected exponents together with the exponents for the blocking factor and giant stepping that attain the optimum. Line 1 is the symbolic solution, Line 2 gives the best exponent that we have achieved. Line 3 is the solution without appealing to faster rectangular matrix multiplication schemes. Line 4 corresponds to the comments before Remark 4.7 on page 115, and line 5 uses Strassen's original subquadratic matrix multiplication algorithm. Line 6 exhibits the slowdown without faster rectangular matrix multiplication algorithms. Line 7 is our complexity for a hypothetical quadratic matrix multiplication algorithm.

An issue arises whether the singularity certification in Step 0 of our algorithm can be accomplished at a matching or lower bit complexity than the ones

	ω	ζ	η	σ	τ
1	ω	ζ	$\omega + \frac{1-\zeta}{\omega^2-(2+\zeta)\omega+2}$	$1 - \frac{\omega-(1+\zeta)}{\omega^2-(2+\zeta)\omega+2}$	$\frac{\omega-2}{\omega^2-(2+\zeta)\omega+2}$
2	2.375477	0.2946289	2.697263	0.506924	0.171290
3	ω	0	$\omega + \frac{1}{(\omega-1)^2+1}$	$1 - \frac{\omega-1}{(\omega-1)^2+1}$	$\frac{\omega-2}{(\omega-1)^2+1}$
4	3	0	$3 + \frac{1}{5}$	$\frac{3}{5}$	$\frac{1}{5}$
5	$\log_2(7)$	0	3.041738	0.576388	0.189230
6	2.375477	0	2.721267	0.524375	0.129836
7	2	0	$2 + \frac{1}{2}$	$\frac{1}{2}$	0

Table 6.1: Determinantal bit/division-free complexity exponent η.

given above for the determinant. We refer to possible approaches by Mulders and Storjohann (2004) and Storjohann (2004).

The above analysis applies to our algorithm in Section 5 and yields for the determinant and adjoint matrix a division-free complexity of $O(n^{2.697263})$ ring operations. To our knowledge, this is the best known to date. For the division-free computation of the characteristic polynomial the homotopy in z is altered, because the computation of $\det F_X^{L(z),Y}(\lambda) \bmod (z^{n+1}, \lambda^{n+1})$ in Step 4 (see Section 5) seems to require too many ring operations. One instead computes

$$\det(M - zAM) = \det(I - zA) \det M = \pm z^n \det((1/z)I - A)$$

by replacing $A - M$ by AM in the original determinant algorithm. Since $\det M = \pm 1$ one thus gets (the reverse of) the characteristic polynomial in $O(n^{2.697263})$ ring operations as well.

A Maple 7 worksheet that contains our exponent calculations is posted at `http://www.kaltofen.us/bibliography`.

7. Integer characteristic polynomial and normal forms

As already seen in Sections 5 and 6 over an abstract ring R, our determinant algorithm also computes the adjoint matrix and the characteristic polynomial. In the case of integer matrices, although differently from the algebraic setting, the algorithm of Section 4 may also be extended to solving other problems. We briefly mention two extensions. For $A \in \mathbb{Z}^{n\times n}$ we shall first see that the algorithm leads to the characteristic polynomial of a preconditioning of A and consequently to the Smith normal form of A. We shall then see how $F_X^{A,Y}$ may

be used to compute the Frobenius normal form of A and hence its characteristic polynomial. Note that the exponents in our bit complexity are of the same order as those discussed for the determinant problem in Table 6.1.

7.1. Smith normal form of integer matrices. A randomized Monte Carlo algorithm for computing the Smith normal form $S \in \mathbb{Z}^{n\times n}$ of an integer matrix $A \in \mathbb{Z}^{n\times n}$ of rank r may be designed by combining the algorithm of Section 4 with the approach of Giesbrecht (2001). Here we improve on the best previously known randomized algorithm of Eberly *et al.* (2000). The current estimate for a deterministic computation of the form is $(n^{\omega+1}\log\|A\|)^{1+o(1)}$ (Storjohann 1996).

The Smith normal form over $\mathbb{Z}$ is defined in a way similar to what we have seen in Section 2.2 for polynomial matrices. The Smith form S is an equivalent diagonal matrix in $\mathbb{Z}^{n\times n}$, with diagonal elements $s_1, s_2, \ldots, s_r, 0, \ldots, 0$ such that s_i divides s_{i-1} for $2 \le i \le r$. The s_i's are the invariant factors of A (Newman 1972).

Giesbrecht's approach reduces the computation of S to the computation of the characteristic polynomials of the matrices $D_1^{(i)}T^{(i)}D_2^{(i)}A$ for $l = (\log n + \log\log\|A\|)^{1+o(1)}$ random choices of diagonal matrices $D_1^{(i)}$ and $D_2^{(i)}$ and of Toeplitz matrices $T^{(i)}$, $1 \le i \le l$. The invariant factors may be computed from the coefficients of these characteristic polynomials. The preconditioning $B \leftarrow D_1^{(i)}T^{(i)}D_2^{(i)}A$ ensures that the minimum polynomial f^B of B is squarefree (Giesbrecht 2001, Theorem 1.4) (see also Chen *et al.* 2002 for such preconditionings). Hence if $\bar{f}^B$ denotes the largest divisor of f^B such that $\bar{f}^B(0) \neq 0$, we have $r = \operatorname{rank} B = \deg \bar{f}^B$, which is $-1 + \deg f^B$ if A is singular. By Theorem 2.12, for random X and Y we shall have, with high probability, $\Delta(\lambda) = \det F_X^{B,Y}(\lambda) = \lambda^{k_1} f^B(\lambda) = \lambda^{k_2}\bar{f}^B(\lambda)$ for two positive integers k_1 and k_2 that depend on the rank and on the blocking factor m. The needed characteristic polynomials $\lambda^{n-r}\bar{f}^B$ and then the Smith form are thus obtained from the determinants of l matrix generating polynomials.

To ensure a high probability of success, the computations are done with $D_1^{(i)}, D_2^{(i)}$ and $T^{(i)}$ chosen over a ring extension $R_{\mathbb{Z}}$ of degree $O((\log n)^2)$ of $\mathbb{Z}$, in combination with Chinese remaindering modulo $(n\log\|A\|)^{1+o(1)}$ primes (Giesbrecht 2001, Theorem 4.2). For one choice of $B^{(i)}$, the cost overhead compared to Step 4 in Section 4 is the one of computing the entire determinant of the $m \times m$ matrix polynomial $F_X^{B^{(i)},Y}$ of degree $d = \lceil n/m \rceil$. Over a field, by (Storjohann 2002, Proposition 24) or (Storjohann 2003, Proposition 41) such a determinant is computed in $(m^\omega d)^{1+o(1)}$ arithmetic operations. Using the $(n\log\|A\|)^{1+o(1)}$ primes and the fact that the ring extension $R_{\mathbb{Z}}$ has degree

$O((\log n)^2)$, $\det F_X^{B^{(i)},Y} \in R_{\mathbb{Z}}[\lambda]$ is thus computed in $(n^{2+\sigma(\omega-1)} \log \|A\|)^{1+o(1)}$ bit operations.

From this we see that the cost of computing the l characteristic polynomials, which is the dominant cost in computing the Smith form, corresponds to the estimate already taken into account for Step 3 of the determinant algorithm. Hence the values of η in Table 6.1 remain valid for the computation of the Smith normal form using a randomized Monte Carlo algorithm.

7.2. Integer characteristic polynomial and Frobenius normal form. As used above, a direct application of Section 4 leads to the characteristic polynomial of a preconditioning of A. To compute the characteristic polynomial of A itself, we extend our approach using the Frobenius normal form and the techniques of Storjohann (2000b). The Frobenius normal form of $A \in \mathbb{Z}^{n\times n}$ is a block diagonal matrix in $\mathbb{Z}^{n\times n}$ similar to A. Its diagonal blocks are the companion matrices for the invariant factors $s_1(\lambda), \ldots, s_\phi(\lambda)$ of $\lambda I - A$. Hence the characteristic polynomial $\det(\lambda I - A) = \prod_{i=1}^{\phi} s_i(\lambda)$ is directly obtained from the normal form. Our result is a randomized Monte Carlo algorithm which improves on previous complexity estimates for computing the characteristic polynomial or the Frobenius normal form over $\mathbb{Z}$ (Storjohann 2000a, Table 10.1). The certified randomized algorithm of Giesbrecht and Storjohann (2002) uses $(n^{\omega+1} \log \|A\|)^{1+o(1)}$ bit operations.

By Theorem 2.12 on page 102, if we avoid the preconditioning step (Step 1) in the determinant algorithm on page 109 in Section 4, the computation leads to $F_X^{A,Y}(\lambda)$ and to

$$\det F_X^{A,Y}(\lambda) = \prod_{i=1}^{\min\{m,\phi\}} s_i(\lambda).$$

The first invariant factor $s_1(\lambda)$ is the minimum polynomial f^A of A, hence $\det F_X^{A,Y}$ is a multiple of f^A and a factor of the characteristic polynomial in $\mathbb{Z}[\lambda]$. Following the cost analysis of the previous Section 7.1 for the determinant of the matrix generating polynomial, the exponents in Table 6.1 are thus valid for the computation of $\det F_X^{A,Y}$. The square free part f^A_{sqfr} of $\det F_X^{A,Y}$ may be deduced in $(n^2 \log \|A\|)^{1+o(1)}$ bit operations (Gerhard 2001, Theorem 11).

From the Frobenius normal form of A modulo a random prime p, f^A_{sqfr} allows a multifactor Hensel lifting for reconstructing the form over $\mathbb{Z}$ (Storjohann 2000b). With high probability, $\lambda I - A$ also has ϕ invariant factors modulo p. We denote them by $\bar{s}_1, \ldots, \bar{s}_\phi$. They can be decomposed into ϕ products

$$\bar{s}_i = \bar{t}_1^{e_{i1}} \ldots \bar{t}_m^{e_{im}}, \quad 1 \le i \le \phi,$$

for a GCD-free family $\{\bar{t}_1, \ldots, \bar{t}_m\}$ of square free polynomials in $\mathbb{F}_p[\lambda]$ and for indices $(e_{i1}, \ldots, e_{im}) \in \mathbb{Z}^m_{>0}$, $1 \leq i \leq \phi$. This decomposition is computed in $(n^2 \log p)^{1+o(1)}$ bit operations (Bach & Shallit 1996, Section 4.8). With high probability we also have

$$\bar{t}_1 \bar{t}_2 \ldots \bar{t}_m = f^A_{\text{sqfr}} \bmod p.$$

The latter factorization can be lifted, for instance using the algorithm of (von zur Gathen & Gerhard 1999, §15.5), into a family $\{t_1, \ldots, t_m\}$ of polynomials modulo a sufficiently high power k of p. With high probability, the invariant factors of $\lambda I - A$ over $\mathbb{Z}$ and the Frobenius form of A may finally be obtained as the following combinations of the t_i's:

$$s_i = t_1^{e_{i1}} \ldots t_m^{e_{im}} \bmod p^k, \quad 1 \leq i \leq \phi,$$

with coefficients reduced in the symmetric range.

In addition to the computation of $F^{A,Y}_X(\lambda)$, the dominant cost is the cost of the lifting. Any divisor of the characteristic polynomial has a coefficient size of $(n \log \|A\|)^{1+o(1)}$ (for instance see Giesbrecht & Storjohann 2002, Lemma 2.1) hence one can take $k = (n \log \|A\|)^{1+o(1)}$. The polynomials $t_1, \ldots, t_m$ are thus computed in $(n^2 \log \|A\|)^{1+o(1)}$ bit operations (von zur Gathen & Gerhard 1999, Theorem 15.18). We may conclude that the values of the exponent of η in Table 6.1 are valid for the randomized computation of the Frobenius normal form and the characteristic polynomial of an integer matrix.

Theorem 5.4 by Pan (2002) states a Las Vegas bit complexity of $(n^{16/5} \times \log \|A\|)^{1+o(1)}$ for the Frobenius factors of a matrix $A \in \mathbb{Z}^{n \times n}$ by a different method. Victor Pan has told us on May 13, 2004 that his proof of his claim currently has a flaw.

8. Concluding remarks

Our baby steps/giant steps and blocking techniques apply to entry domains other than the integers, like polynomial rings and algebraic number rings. We would like to add that if the entries are polynomials over a possibly finite field, there are additional new techniques possible (Jeannerod & Villard 2004; Mulders & Storjohann 2003; Storjohann 2002, 2003). Storjohann (2004) has extended his 2003 techniques to construct a Las Vegas algorithm that computes $\det A$ where $A \in \mathbb{Z}^{n \times n}$ in $(n^\omega \log \|A\|)^{1+o(1)}$ bit operations, when $n \times n$ matrices are multiplied in $O(n^\omega)$ algebraic operations. The best known division-free complexity of the determinant remains at $O(n^{2.697263})$ as stated in Section 5 and Section 6. Furthermore, the best known bit-complexity of the characteristic

polynomial of an integer matrix is to our knowledge the one in Section 7.2, namely $(n^{2.697263} \log \|A\|)^{1+o(1)}$.

For the classical matrix multiplication exponent $\omega = 3$, the bit complexity of integer matrix determinants is thus proportional to $n^{\eta+o(1)}$ as follows: $\eta = 3 + \frac{1}{2}$ (Eberly *et al.* 2000; Kaltofen 1992, 2002), $\eta = 3 + \frac{1}{3}$ (Theorem 4.2 on page 111), $\eta = 3 + \frac{1}{5}$ (line 4 in Table 6.1 on page 119), $\eta = 3$ (Storjohann 2004). Together with the algorithms discussed in Section 1 on page 94 that perform well on propitious inputs, such a multitude of results poses a problem for the practitioner: which of the methods can yield faster procedures in computer algebra systems? With William J. Turner we have implemented our baby steps/giant steps algorithm (Kaltofen 1992, 2002) in Maple 6 with mixed results in comparison to Gaussian elimination and Chinese remaindering. The main problem seems the overhead hidden in the $n^{o(1)}$-factor. For example, for $n_1 = 10000$ one has $(\log_2 n_1)/n_1^{1/3} > 0.616$, which means that saving a factor of $n^{1/3}$ at the cost of a factor $\log_2 n$ may for practical considerations be quite immaterial. In addition, one also needs to consider other properties, such as the required intermediate space and whether the algorithm is easily parallelized. We believe that the latter may be the most important advantage in practice of our block approach (cf. Coppersmith 1994; Kaltofen 1995).

The reduction of the bit complexity of an algebraic problem below that of its known algebraic complexity times the bit length of the answer should raise important considerations for the design of generic algorithms with abstract coefficient domains (Jenks *et al.* 1988) and for the interpretation of algebraic lower bounds for low complexity problems (Strassen 1990). We demonstrate that the interplay between the algebraic structure of a given problem and the bits of the intermediately computed numbers can lead to a dramatic reduction in the bit complexity of a fundamental mathematical computation task.

Acknowledgements

We thank William J. Turner for his observations on the practicality of our method, Mark Giesbrecht for reporting to us the value of the smallest exponent in (Eberly *et al.* 2000) prior to its publication, Elwyn Berlekamp for comments on the Berlekamp/Massey algorithm, and the three referees for their comments.

This material is based on work supported in part by the National Science Foundation (USA) under Grants Nos. DMS-9977392, CCR-9988177 and CCR-0113121 (Kaltofen) and by CNRS (France) Actions Incitatives No 5929 et Stic LinBox 2001 (Villard).

An extended abstract of this paper is (Kaltofen & Villard 2001).

References

Note: many of the authors' publications cited below are accessible through links in their webpages listed under their addresses.

J. Abbott, M. Bronstein & T. Mulders (1999). Fast deterministic computation of determinants of dense matrices. In *ISSAC 99, Proc. 1999 Internat. Symp. Symbolic Algebraic Comput.*, S. Dooley (ed.), ACM Press, New York, 181–188.

M. Agrawal, N. Kayal & Nitin Saxena (2002). PRIMES is in P. Manuscript. Available from `http://www.cse.iitk.ac.in/news/primality.pdf`.

A. Aho, J. Hopcroft & J. Ullman (1974). *The Design and Analysis of Algorithms*. Addison and Wesley, Reading, MA.

E. Bach & J. Shallit (1996). *Algorithmic Number Theory. Volume 1: Efficient Algorithms*. The MIT Press, Cambridge, MA.

W. Baur & V. Strassen (1983). The complexity of partial derivatives. *Theoret. Comput. Sci.* **22**, 317–330.

B. Beckermann & G. Labahn (1994). A uniform approach for fast computation of matrix-type Padé approximants. *SIAM J. Matrix Anal. Appl.* **15**, 804–823.

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

R. P. Brent, S. H. Gao & A. G. B. Lauder (2003). Random Krylov spaces over finite fields. *SIAM J. Discrete Math.* **16**, 276–287.

H. Brönnimann, I. Emiris, V. Pan & S. Pion (1999). Sign determination in residue number systems. *Theoret. Comput. Sci.* **210**, 173–197.

H. Brönnimann & M. Yvinec (2000). Efficient exact evaluation of signs of determinants. *Algorithmica* **27**, 21–56.

W. S. Brown & J. F. Traub (1971). On Euclid's algorithm and the theory of subresultants. *J. ACM* **18**, 505–514.

D. G. Cantor & E. Kaltofen (1991). On fast multiplication of polynomials over arbitrary algebras. *Acta Inform.* **28**, 693–701.

L. Chen, W. Eberly, E. Kaltofen, B. D. Saunders, W. J. Turner & G. Villard (2002). Efficient matrix preconditioners for black box linear algebra. *Linear Algebra Appl.* **343–344**, 119–146.

K. L. Clarkson (1992). Safe and efficient determinant evaluation. In *Proc. 33rd Annual Sympos. Foundations of Comput. Sci.*, IEEE Comput. Soc. Press, Los Alamitos, CA, 387–395.

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

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

D. Coppersmith & S. Winograd (1990). Matrix multiplication via arithmetic progressions. *J. Symbolic Comput.* **9**, 251–280.

R. A. DeMillo & R. J. Lipton (1978). A probabilistic remark on algebraic program testing. *Inform. Process. Lett.* **7**, 193–195.

B. W. Dickinson, M. Morf & T. Kailath (1974). A minimal realization algorithm for matrix sequences. *IEEE Trans. Automat. Control* **AC-19**, 31–38.

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

J. L. Dornstetter (1987). On the equivalence between Berlekamp's and Euclid's algorithms. *IEEE Trans. Inform. Theory* **33**, 428–431.

W. Eberly (2002). Avoidance of look-ahead in Lanczos by random projections. Manuscript in preparation.

W. Eberly, M. Giesbrecht & G. Villard (2000). On computing the determinant and Smith form of an integer matrix. In *Proc. 41st Annual Sympos. Foundations of Comput. Sci.*, IEEE Comput. Soc. Press, Los Alamitos, CA, 675–685.

W. Eberly & E. Kaltofen (1997). On randomized Lanczos algorithms. In Küchlin (1997), 176–183.

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

G. D. Forney, Jr. (1975). Minimal bases of rational vector spaces, with applications to multivariable linear systems. *SIAM J. Control* **13**, 493–520.

J. von zur Gathen & J. Gerhard (1999). *Modern Computer Algebra*. Cambridge Univ. Press, Cambridge.

J. Gerhard (2001). Fast modular algorithms for squarefree factorization and Hermite integration. *Appl. Algebra Engrg. Comm. Comput.* **11**, 203–226.

M. Giesbrecht (2001). Fast computation of the Smith form of a sparse integer matrix. *Comput. Complexity* **10**, 41–69.

M. Giesbrecht & A. Storjohann (2002). Computing rational forms of integer matrices. *J. Symbolic Comput.* **34**, 157–172.

P. Giorgi, C.-P. Jeannerod & G. Villard (2003). On the complexity of polynomial matrix computations. In Sendra (2003), 135–142.

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

X. H. Huang & V. Y. Pan (1998). Fast rectangular matrix multiplication and applications. *J. Complexity* **14**, 257–299.

C.-P. Jeannerod & G. Villard (2004). Essentially optimal computation of the inverse of generic polynomial matrices. *J. Complexity*, to appear. Available from `http://perso.ens-lyon.fr/gilles.villard`.

R. D. Jenks, R. S. Sutor & S. M. Watt (1988). Scratchpad II: An abstract datatype system for mathematical computation. In *Mathematical Aspects of Scientific Software*, J. R. Rice (ed.), IMA Vol. Math. Appl. 14, Springer, New York, 157–182.

T. Kailath (1980). *Linear Systems*. Prentice-Hall.

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

E. Kaltofen (1992). On computing determinants of matrices without divisions. In *Proc. 1992 Internat. Sympos. Symbolic Algebraic Comput. (ISSAC'92)*, P. S. Wang (ed.), ACM Press, New York, 342–349.

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

E. Kaltofen (2000). Challenges of symbolic computation: my favorite open problems. *J. Symbolic Comput.* **29**, 891–919. With an additional open problem by R. M. Corless and D. J. Jeffrey.

E. Kaltofen (2002). An output-sensitive variant of the baby steps/giant steps determinant algorithm. In Mora (2002), 138–144.

E. Kaltofen & W.-S. Lee (2003). Early termination in sparse interpolation algorithms. *J. Symbolic Comput.* **36**, 365–400.

E. Kaltofen, W.-S. Lee & A. A. Lobo (2000). Early termination in Ben-Or/Tiwari sparse interpolation and a hybrid of Zippel's algorithm. In *Proc. 2000 Internat. Sympos. Symbolic Algebraic Comput. (ISSAC'00)*, C. Traverso (ed.), ACM Press, New York, 192–201.

E. Kaltofen & J. May (2003). On approximate irreducibility of polynomials in several variables. In Sendra (2003), 161–168.

E. Kaltofen & B. D. Saunders (1991). On Wiedemann's method of solving sparse linear systems. In *Proc. AAECC-9*, H. F. Mattson *et al.* (eds.) Lecture Notes in Comput. Sci. 539, Springer, Heidelberg, 29–38.

E. Kaltofen & G. Villard (2001). On the complexity of computing determinants. In *Proc. Fifth Asian Symposium on Computer Mathematics (ASCM 2001)*, K. Shirayanagi and K. Yokoyama (eds.), Lecture Notes Ser. Comput. 9, World Sci., Singapore, 13–27.

E. Kaltofen & G. Villard (2004). Computing the sign or the value of the determinant of an integer matrix, a complexity survey. *J. Comput. Appl. Math.* **162**, 133–146.

D. E. Knuth (1970). The analysis of algorithms. In *Congrès Int. Math., Nice,* Volume 3, 269–274.

W. Küchlin (ed.) (1997). *ISSAC 97, Proc. 1997 Internat. Sympos. Symbolic Algebraic Comput.* ACM Press, New York.

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

T. Mora (ed.) (2002). *ISSAC 2002 Proc. 2002 Internat. Sympos. Symbolic Algebraic Comput.* ACM Press, New York.

T. Mulders & A. Storjohann (2003). On lattice reduction for polynomial matrices. *J. Symbolic Comput.* **35**, 377–401.

T. Mulders & A. Storjohann (2004). Certified dense linear system solving. *J. Symbolic Comput.* **37** (2004), 485–510.

M. Newman (1972). *Integral Matrices.* Academic Press.

V. Y. Pan (1988). Computing the determinant and the characteristic polynomial of a matrix via solving linear systems of equations. *Inform. Process. Lett.* **28**, 71–75.

V. Y. Pan (2002). Randomized acceleration of fundamental matrix computations. In *Proc. STACS 2002*, Lecture Notes in Comput. Sci. 2285, Springer, Heidelberg, 215–226.

M. S. Paterson & L. J. Stockmeyer (1973). On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.* **2**, 60–66.

V. M. Popov (1970). Some properties of control systems with irreducible matrix-transfer functions. In *Seminar on Differential Equations and Dynamical Systems, II* (College Park, MD, 1969), Lecture Notes in Math. 144, Springer, Berlin, 169–180.

J. Rissanen (1972). Realizations of matrix sequences. Technical Report RJ-1032, IBM Research, Yorktown Heights, NY.

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

A. Schönhage (1971). Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Inform.* **1**, 139–144.

J. T. Schwartz (1980). Fast probabilistic algorithms for verification of polynomial identities. *J. ACM* **27**, 701–717.

T. R. Seifullin (2003). Acceleration of computation of determinants and characteristic polynomials without divisions. *Cybernet. Systems Anal.* **39**, 805–815.

J. R. Sendra (ed.) (2003). *ISSAC 2003, Proc. 2003 Internat. Sympos. Symbolic Algebraic Comput.* ACM Press, New York.

A. Storjohann (1996). Near optimal algorithms for computing Smith normal forms of integer matrices. In *ISSAC 96, Proc. 1996 Internat. Sympos. Symbolic Algebraic Comput.*, Y. N. Lakshman (ed.), ACM Press, New York, 267–274.

A. Storjohann (2000a). *Algorithms for matrix canonical forms*. Dissertation, Swiss Federal Institute of Technology (ETH), Zurich.

A. Storjohann (2000b). Computing the Frobenius form of a sparse integer matrix. Paper to be submitted.

A. Storjohann (2002). Higher-order lifting. In Mora (2002), 246–254.

A. Storjohann (2003). High-order lifting and integrality certification. *J. Symbolic Comput.* **36**, 613–648.

A. Storjohann (2004). The shifted number system for fast linear algebra on integer matrices. Technical Report CS-2004-18, School of Computer Science, University of Waterloo, `http://www.scg.uwaterloo.ca/~astorjoh/publications.html`.

V. Strassen (1973). Vermeidung von Divisionen. *J. Reine Angew. Math.* **264**, 182–202.

V. Strassen (1990). Algebraic complexity theory. In *Handbook of Theoretical Computer Science, Algorithms and Complexity*, J. van Leeuwen (ed.), Volume A, Elsevier, Amsterdam, 633–672.

Y. Sugiyama, M. Kasahara, S. Hirasawa & T. Namekawa (1975). A method for solving key equation for decoding Goppa codes. *Inform. and Control* **27**, 87–99.

E. Thomé (2002). Subquadratic computation of vector generating polynomials and improvements of the block Wiedemann method. *J. Symbolic Comput.* **33**, 757–775.

W. J. Turner (2001). A note on determinantal divisors and matrix preconditioners. Paper to be submitted.

W. J. Turner (2002). *Black box linear algebra with the LINBOX library*. Ph.D. thesis, North Carolina State Univ., Raleigh, NC, 193 pp.

M. Van Barel & A. Bultheel (1992). A general module theoretic framework for vector M-Padé and matrix rational interpolation. *Numerical Algorithms* **3**, 451–462.

G. Villard (1988). *Calcul Formel et Parallélisme : Résolution de Systèmes Linéaires*. Ph.D. thesis, Institut National Polytechnique de Grenoble.

G. Villard (1997a). Further analysis of Coppersmith's block Wiedemann algorithm for the solution of sparse linear systems. In Küchlin (1997), 32–39.

G. Villard (1997b). 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`.

G. Villard (2000). Computing the Frobenius normal form of a sparse matrix. In *CASC 2000, Proc. 3rd Internat. Workshop on Computer Algebra in Scientific Computing*, V. G. Ganzha *et al.* (eds.), Springer, 395–407.

D. Wiedemann (1986). Solving sparse linear equations over finite fields. *IEEE Trans. Inform Theory* **32**, 54–62.

R. Zippel (1979). Probabilistic algorithms for sparse polynomials. In *Proc. EUROSAM '79*, Lecture Notes in Comput. Sci. 72, Springer, Heidelberg, 216–226.

Manuscript received 23 August 2003

Erich Kaltofen
Department of Mathematics
North Carolina State University
Raleigh, NC 27695-8205, U.S.A.
kaltofen@math.ncsu.edu
http://www.kaltofen.us

Gilles Villard
Laboratoire LIP
École Normale Supérieure de Lyon
46, Allée d'Italie
69364 Lyon Cedex 07, France
Gilles.Villard@ens-lyon.fr
http://perso.ens-lyon.fr/gilles.villard/