NEARLY OPTIMAL ALGORITHMS FOR CANONICAL MATRIX FORMS

MARK GIESBRECHT *

Abstract. A Las Vegas type probabilistic algorithm is presented for finding the Frobenius canonical form of an $n \times n$ matrix T over any field K. The algorithm requires $O\tilde{~}(\mathrm{MM}(n)) = \mathrm{MM}(n) \cdot (\log n)^{O(1)}$ operations in K, where $O(\mathrm{MM}(n))$ operations in K are sufficient to multiply two $n \times n$ matrices over K. This nearly matches the lower bound of $\Omega(\mathrm{MM}(n))$ operations in K for this problem, and improves on the $O(n^4)$ operations in K required by the previously best known algorithms. We also demonstrate a fast parallel implementation of our algorithm for the Frobenius form, which is processor-efficient on a PRAM. As an application we give an algorithm to evaluate a polynomial $g \in \mathsf{K}[x]$ at T which requires only $O\tilde{~}(\mathrm{MM}(n))$ operations in K when $\deg g \leq n^2$. Other applications include sequential and parallel algorithms for computing the minimal and characteristic polynomials of a matrix, the rational Jordan form of a matrix, for testing whether two matrices are similar, and for matrix powering, which are substantially faster than those previously known.

Key words. Frobenius form, Jordan form, evaluating polynomials at matrices, matrix powering, matrix multiplication, processor-efficient parallel algorithms.

AMS subject classifications. 15-04, 15A21, 65Y05, 65Y20

Computing a canonical or normal form of an $n \times n$ matrix T over any field K is a classical mathematical problem with many practical applications. A fundamental theorem of linear algebra states that any $T \in \mathsf{K}^{n \times n}$ is similar to a unique matrix $S \in \mathsf{K}^{n \times n}$ of the block diagonal form

$$S = \mathrm{diag}(C_{f_1}, C_{f_2}, \ldots, C_{f_k}) = \begin{pmatrix} \boxed{C_{f_1}} & & & \mathbf{0} \\ & \boxed{C_{f_2}} & & \\ & & \ddots & \\ \mathbf{0} & & & \boxed{C_{f_k}} \end{pmatrix} \in \mathsf{K}^{n \times n}, \quad (0.1)$$

where each C_{f_i} is the companion matrix of some monic $f_i \in \mathsf{K}[x]$ for $1 \leq i \leq k$, and $f_i \mid f_{i-1}$ for $2 \leq i \leq k$ (we write $T \sim S$ to denote similarity). Recall that the companion matrix C_g of a monic $g = \sum_{0 \leq j \leq r} b_i x^i \in \mathsf{K}[x]$ has the form

$$C_g = \begin{pmatrix} 0 & 0 & & & -b_0 \\ 1 & 0 & & & -b_1 \\ & 1 & & \mathbf{0} & \vdots \\ & & \ddots & & \vdots \\ & \mathbf{0} & & \ddots & \\ & & & 1 & -b_{r-1} \end{pmatrix} \in \mathsf{K}^{r \times r}.$$

* Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, R3T 2N2. Research was supported in part by Natural Sciences and Engineering Research Council of Canada research grant OGP0155376.

A matrix S with these properties, called the *Frobenius form* of T, always exists and is unique and *rational*, i.e., it remains the same even if the entries of T are allowed to lie in an algebraic closure of K. The polynomials $f_1, \ldots, f_k \in \mathsf{K}[x]$ are the *invariant factors* of T, and the first of these, f_1, is the *minimal polynomial* of T, that is f_1 is the polynomial of smallest degree in $\mathsf{K}[x] \setminus \{0\}$ such that $f_1(T) = 0$. The product $f_1 \cdots f_k$ is the characteristic polynomial of T. Since T is similar to S, by definition there exists an invertible $U \in \mathsf{K}^{n\times n}$ such that $U^{-1}TU = S$. Our approach to finding the Frobenius form is to find such a matrix U, from which we find S. Two excellent references for the background information are Gantmacher (1990), Chapter 7, and Hoffman & Kunze (1971), Chapter 7.

In Sections 1 through 4 we link the cost of computing the Frobenius form to the cost of multiplying two matrices. Specifically, if $\mathrm{MM}(n)$ operations in K are sufficient to multiply two $n \times n$ matrices over a field K, we show that $O\tilde{~}(\mathrm{MM}(n)) = \mathrm{MM}(n) \cdot (\log n)^{O(1)}$ operations in K suffice to compute the Frobenius form of an $n \times n$ matrix over K. This algorithm is of the Las Vegas type — it is allowed to choose elements randomly and uniformly from a finite subset of K at unit cost, and with probability at least 1/4 it returns the correct answer, otherwise it reports failure. An incorrect answer is never returned. Our algorithm only works as stated when $\#\mathsf{K} \geq n^2$. When K has $q < n^2$ elements, we embed K in a field F of degree $O(\log n)$ over K which does possess n^2 elements. The Frobenius form of T lies in $\mathsf{K}^{n\times n}$ since it is rational, but $U \in \mathsf{F}^{n\times n}$ may have entries in $\mathsf{F} \setminus \mathsf{K}$. In this case the running time of our algorithms is multiplied by a small power of $\log_q n$. Details are presented in Section 4. In Section 5 we show how to implement our algorithm for the Frobenius form in a processor-efficient manner on a PRAM. As a by-product we obtain the first processor-efficient parallel algorithm for the characteristic polynomial of a matrix.

The best previously known algorithms for finding the Frobenius form of a matrix in $\mathsf{K}^{n\times n}$, by Ozello (1987) and Lüneburg (1987), require $O(n^4)$ operations in K (see also Kannan & Bachem 1979, and Kannan 1985). Kaltofen *et al.* (1987, 1990) present probabilistic algorithms for finding the Frobenius form of a matrix in the parallel complexity class RNC^2. They achieve their result through the use of a more general algorithm to compute the Smith normal form of a polynomial matrix. From the Smith normal form of the matrix $\lambda I - T \in \mathsf{K}[\lambda]^{n\times n}$ (where λ is an indeterminate) the Frobenius form S of T can be derived easily. A (deterministic) NC^2 algorithm for computing the Frobenius form is demonstrated by Villard (1994).

Our algorithm for computing the Frobenius form is nearly optimal in that there is a lower bound for the problem of $\Omega(\mathrm{MM}(n))$ operations in K. If we can compute the Frobenius form, then we can find the characteristic polynomial $f \in \mathsf{K}[x]$ of T with $O(n^2)$ additional operations in K. The determinant of T is $f(0)$, and it is shown in Baur & Strassen (1982) that computing the determinant requires $\Omega(\mathrm{MM}(n))$ operations in K, whence computing the Frobenius form of T requires $\Omega(\mathrm{MM}(n))$ operations in K (one must be careful of the model of computation here as Baur & Strassen's result is for the arithmetic circuit model – see Giesbrecht 1993, Section 1.1 for details).

As applications of our algorithm for computing the Frobenius form, we obtain fast algorithms for a number of interesting problems. One of the most striking results is for the problem of evaluating a polynomial at a matrix. In Section 6 we show that a polynomial $g \in \mathsf{K}[x]$, of degree r, can be evaluated at any matrix $T \in \mathsf{K}^{n\times n}$ with $O\tilde{~}(\mathrm{MM}(n) + r)$ operations in K. We also demonstrate a lower bound for evaluating a fixed non-linear polynomial at a matrix of $\Omega(\mathrm{MM}(n))$ operations in K, so in fact our algorithm is nearly optimal. The algorithm we present here improves upon the previously fastest algorithm of Paterson & Stockmeyer (1973), which requires $O(\mathrm{MM}(n)\sqrt{r})$ operations in K. More generally, we show that an arithmetic circuit or straight-line program can be evaluated at a matrix in nearly optimal time sequentially, and processor-efficiently in parallel. In brief, the Frobenius form provides a mechanism to transform the problem of evaluating a polynomial in $\mathsf{K}[x]$ at a matrix in $\mathsf{K}^{n\times n}$ from the multiplicative semi-group of $\mathsf{K}^{n\times n}$ to the multiplicative semi-group of a modular polynomial ring, where computation can be performed much more quickly.

As noted above, the companion matrix of the minimal polynomial forms the first block in the Frobenius form. Also, any two similar matrices have the same Frobenius form. Thus, our algorithm for computing the Frobenius form yields Las Vegas algorithms to determine the similarity of any two matrices in $\mathsf{K}^{n\times n}$, and to find the minimal polynomial of a matrix, which require $O\tilde{~}(\mathrm{MM}(n))$ operations in K. Our algorithm for computing the minimal polynomial is also nearly optimal: it is shown by Wiedemann (see also Kaltofen 1992) that if we can compute the minimal polynomial of an $n \times n$ matrix over a field K with $t(n)$ operations in K, then there is a Las Vegas algorithm to compute the determinant of any matrix in $\mathsf{K}^{n\times n}$ for which $O(t(n))$ operations in K are sufficient (see Giesbrecht 1993 for details). The best previously known algorithm for computing the minimal polynomial of an $n \times n$ matrix is the Monte Carlo algorithm of Wiedemann (1986), and requires an expected $O(n^3)$ operations (a Monte Carlo type algorithm has the ability to select random elements uniformly from a finite subset of K and with constant probability returns the correct answer, but with some controllably small probability may return an incorrect answer). To determine similarity, the best previously known sequential algorithms are the Frobenius form algorithms of Lüneburg (1987) and Ozello (1987), which require $O(n^4)$ field operations. An algorithm by Zalcstein & Garzon (1987), based on a very different method, runs in the parallel complexity class NC^2.

The Jordan normal form is probably the most commonly encountered of all canonical matrix forms, and is also one of the more difficult to compute. In particular, it requires that we determine both the geometric structure of the matrix (as captured in the Frobenius form) and the factorization of the minimal polynomial of the matrix into linear factors. In Section 7 we introduce the rational Jordan form as a slight generalization of the usual Jordan form. The rational Jordan form always exists and coincides with the usual Jordan form whenever that form exists. We show that computing the rational Jordan form of an $n \times n$ matrix over any field K can be accomplished by a Las Vegas type algorithm with an expected number of $O\tilde{~}(\mathrm{MM}(n))$ operations in K, given the complete factorization (into irreducible factors in $\mathsf{K}[x]$) of the minimal polynomial of that matrix. A simple lower bound of $\Omega(\mathrm{MM}(n))$ operations in K is shown for this problem. The requirement of a complete factorization of the minimal polynomial is also necessary, given that the complete factorization of any polynomial can be read off the rational Jordan form of the companion matrix of that polynomial.

Computational Model and Complexity Assumptions. The algorithms presented in this paper are generally given for both sequential and parallel models of computation. For the sequential algorithms we employ the arithmetic RAM, described more formally in von zur Gathen (1993). Informally, this is just a standard (Boolean) Random Access Machine (or RAM, see Aho *et al.* 1974, Section 1.2), with an additional memory for holding elements of some field K. The instructions of the usual

RAM are supplemented with instructions for basic field operations in K ($+$, $\times$, $-$, $\div$), as well as a test for the zero element in K, and appropriate input and output instructions for elements in K. We report the cost of our algorithms as the number operations in K required as a function of the number of elements of K in the input.

For parallel algorithms we employ the arithmetic Parallel Random Access Machine (arithmetic PRAM) model over a field K (see Karp & Ramachandran 1990). This is a collection of arithmetic RAM's (over K) communicating by means of a set of shared memory cells. All accesses to global (and local) memory require unit time. The time $t(n)$ required by an arithmetic PRAM algorithm is the maximum of the number of field operations required by each of the component arithmetic RAM's in the PRAM on input size n. Also of interest is $p(n)$, the largest number of processors involved in the computation on any input of size n, as is the work $w(n) = t(n) \cdot p(n)$. It is useful to compare the work required by an algorithm running on a PRAM with the time required by an optimal sequential algorithm. Assume that any sequential algorithm for some problem requires time $\Omega(t(n))$ on input size n. A PRAM algorithm for this problem is called *processor-efficient* if it requires time $(\log n)^{O(1)}$ when run on $t(n) \cdot (\log n)^{O(1)}$ processors. That is, it is a fast parallel algorithm for which the work is within a poly-logarithmic factor of the optimal.

It is occasionally convenient, especially in summarizing results, to ignore logarithmic factors using the "soft O" notation: for any $g, h\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$, $g = O\tilde{}(h)$ if and only if there exists a constant $k \geq 0$ such that $g = O(h(\log h)^k)$.

We isolate the cost of algorithms for polynomial arithmetic and linear algebra as named functions (M and MM, respectively) of their input sizes when their costs appear in the cost of some algorithm using these operations. Over a ring F, we assume $O(\mathrm{M}_{\mathsf{F}}(n))$ operations in F are sufficient to multiply two polynomials in $\mathsf{F}[x]$ of degree at most n. The fast integer multiplication algorithms of Schönhage & Strassen (1971) can be recast as polynomial multiplication algorithms (see Schönhage 1977 and Nussbaumer 1980), and allow us to choose $\mathrm{M}_{\mathsf{F}}(n) = n \log n \log\log n$ when F is a field. More generally, for any ring F, the algorithm of Cantor & Kaltofen (1991) allows us to choose $\mathrm{M}_{\mathsf{F}}(n) = n \log n \log\log n$. If K is a field, and $f, g \in \mathsf{K}[x]$ have degree at most n, we can compute the division with remainder of f by g, that is, we can find $Q, R \in \mathsf{K}[x]$ such that $f = Qg + R$ with $R = 0$ or $\deg R < \deg g$ in $O(\mathrm{M}_{\mathsf{F}}(n))$ operations in K. We can compute $\gcd(f, g)$ with $O(\mathrm{M}_{\mathsf{F}}(n) \log n)$ operations in K (see Aho *et al.* 1974, Section 8.9). Generally we will simply write $\mathrm{M}(n)$ for $\mathrm{M}_{\mathsf{F}}(n)$ when F is clear from context.

For any field K, let $\mathsf{K}^{n\times n}$ be the ring of $n \times n$ matrices with entries in K. We assume that $O(\mathrm{MM}(n))$ operations in K are sufficient to multiply two matrices in $\mathsf{K}^{n\times n}$. Currently, the asymptotically best algorithm for matrix multiplication is by Coppersmith & Winograd (1990), with $\mathrm{MM}(n) = n^{2.376}$. For convenience we assume throughout this paper that $\mathrm{MM}(n) = \Omega(n^{2+\epsilon})$ for some $\epsilon > 0$. $O(\mathrm{MM}(n))$ operations in K are also sufficient to compute the rank and determinant of an $n \times n$ matrix over a field K, as well as to invert a non-singular matrix in $\mathsf{K}^{n\times n}$. Also, we can solve a system of n linear equations in n unknowns over K (which may be singular) with $O(\mathrm{MM}(n))$ operations in K (Bunch & Hopcroft 1974). Corresponding parallel algorithms for the polynomial and matrix operations above are discussed in Section 5.

1. Finding a Modular Cyclic Decomposition. The Frobenius normal form of the matrix $T \in \mathsf{K}^{n\times n}$ corresponds to a decomposition of the vector space $\mathsf{K}^{n\times 1}$, called the *cyclic decomposition* of $\mathsf{K}^{n\times 1}$ with respect to T. It is convenient to simply consider the linear map T as a K-endomorphism of a finite dimensional vector space

V over K (i.e., $T \in \mathrm{End}_{\mathsf{K}} V$), ignoring for now its representation as a matrix (in the case of $T \in \mathsf{K}^{n\times n}$ we have $V = \mathsf{K}^{n\times 1}$). V is a $\mathsf{K}[x]$-module as follows: any $g = \sum_{0\leq i\leq r} b_i x^i \in \mathsf{K}[x]$ acts on V as the endomorphism $\sum_{0\leq i\leq r} b_i T^i \in \mathrm{End}_{\mathsf{K}} V$. We write $f \circ v = f(T)v \in V$. An important role is played in this theory by polynomials which annihilate (i.e., map to $0 \in V$) vectors and subspaces of V. For any $v \in V$, define the *annihilator* $\mathrm{Ann}(T; v) \subseteq \mathsf{K}[x]$ of V as the ideal in $\mathsf{K}[x]$ of all $f \in \mathsf{K}[x]$ such that $f \circ v = 0$. Since $\mathsf{K}[x]$ is a principal ideal domain, $\mathrm{Ann}(T; v)$ is generated by a unique monic polynomial in $\mathsf{K}[x]$, the *minimal polynomial* $\min(T; v) \in \mathsf{K}[x]$ of the vector $v \in V$. The subspace $\mathrm{Orb}(T; v) \subseteq V$ spanned by $v, Tv, T^2v, \ldots \in V$ is the *cyclic subspace* of V generated by v. $\mathrm{Orb}(T; v)$ is T-invariant, that is, $Tw \in \mathrm{Orb}(T; v)$ for all $w \in \mathrm{Orb}(T; v)$, and $\dim \mathrm{Orb}(T; w) = \deg(\min(T; v))$. For any subspace $W \subseteq V$, the set of polynomials $\mathrm{Ann}(T; W) \subseteq \mathsf{K}[x]$ which annihilate *every* vector in W is also an ideal in $\mathsf{K}[x]$, called the *annihilator* of W. As an ideal, it too is generated by a unique monic polynomial, the *minimal polynomial* $\min(T; W) \in \mathsf{K}[x]$ of T on W.

The following very important theorem is shown in Hoffman & Kunze (1971), Section 7.2, Theorem 3, and Gantmacher (1990), Section 7.5.

FACT 1.1 (CYCLIC DECOMPOSITION THEOREM). *Let V be a finite dimensional vector space over a field K and $T \in \mathrm{End}_{\mathsf{K}} V$. There exist vectors $v_1, v_2, \ldots, v_k \in V$ such that*

$$V = V_1 \oplus V_2 \oplus \cdots \oplus V_k,$$

where $V_i = \mathrm{Orb}(T; v_i)$, and

$$\min(T; v_i) = \min(T; V_i \oplus \cdots \oplus V_k) \in \mathsf{K}[x]$$

for $1 \leq i \leq k$.

Given a cyclic decomposition of V under T as $V = V_1 \oplus \cdots \oplus V_k$, where $V_i = \mathrm{Orb}(T; v_i)$ for some $v_i \in V$ for $1 \leq i \leq k$, $f_i = \min(T; v_i) \in \mathsf{K}[x]$ is the *ith invariant factor* of T, for $1 \leq i \leq k$. Recall that the companion matrices of the invariant factors are found, in order, along the diagonal of the Frobenius form of T.

We once again consider T as a matrix in $\mathsf{K}^{n\times n}$, and the relationship between the cyclic decomposition of $\mathsf{K}^{n\times 1}$ under T and T's Frobenius form. Suppose $\deg f_i = d_i$ for $1 \leq i \leq k$. The matrix

$$U = \left[v_1 \,\middle|\, Tv_1 \,\middle|\, T^2v_1 \,\middle|\, \cdots \,\middle|\, T^{d_1-1}v_1 \,\middle|\, \cdots \,\middle|\, v_k \,\middle|\, Tv_k \,\middle|\, \cdots \,\middle|\, T^{d_k-1}v_k \right] \in \mathsf{K}^{n\times n} \quad (1.2)$$

has the property that $S = U^{-1}TU$ is in Frobenius normal form (see, for example, Hoffman & Kunze 1971, Section 7.5).

Our algorithm for finding the Frobenius normal form S of a matrix $T \in \mathsf{K}^{n\times n}$ proceeds by constructing the matrix U as in (1.2), and then computing $S = U^{-1}TU$. To do this we must somehow find the invariant factors $f_1, \ldots, f_k \in \mathsf{K}[x]$ and an associated set of vectors $v_1, \ldots, v_k \in V$ generating the cyclic components. Unfortunately, such vectors are very rare, and the approach of choosing them randomly will fail miserably. Ozello (1987), and Lüneburg (1987) construct them deterministically, but using more time than we allow ourselves. Our approach will be to first choose randomly a list of vectors $w_1, \ldots, w_k \in V$. With high probability these generate a "modular cyclic decomposition" of V (which we define later), from which the invariant factors and the Frobenius form S of T can be determined. The vectors $w_1, \ldots, w_k$ are then "purified" to obtain vectors $v_1, \ldots, v_k$ generating the components of the cyclic decomposition.

Our construction follows approximately the geometric proof of the Cyclic Decomposition Theorem by Gantmacher (1990), Chapter 7 (see also Hoffman & Kunze 1971, Section 7.2)

A weaker form of the cyclic decomposition theorem can be stated in terms of modular T-invariant vector spaces. If W is a T-invariant subspace of V then the space V/W, or V modulo W, is a K-vector space of dimension $\dim V - \dim W$. For $w_1, w_2 \in V$ we denote by $w_1 + W$ and $w_2 + W$ the images of w_1 and w_2 in V/W respectively, and say $w_1 \equiv w_2 \bmod W$ when $w_1 - w_2 \in W$. Since W is T-invariant, V/W has a $\mathsf{K}[x]$-module structure induced by the action of $\mathsf{K}[x]$ on V. Let $T_W: V/W \to V/W$ be T reduced modulo W, carrying $v + W$ to $Tv + W$ for any $v \in V$. Then V/W is a $\mathsf{K}[x]$-module, where $f \in \mathsf{K}[x]$ acts as $f(T_W): V/W \to V/W$, and we write $f \circ (v + W) = f(T_W)(v + W) = f(T)v + W$. Thus, $\min(T_W; v + W) \in \mathsf{K}[x]$ is the minimal polynomial of $v + W \in V/W$ in $\mathsf{K}[x]$, that is, the monic $f \in \mathsf{K}[x] \setminus \{0\}$ of minimal degree such that $f \circ (v + W) \equiv 0 \bmod W$.

An important role is played in the decomposition theory of a vector space by *maximal* vectors $v \in V$, those satisfying $\min(T; v) = \min(T; V)$. A proof of the following lemma can be found in Gantmacher (1990), Section 7.4:

Fact 1.2. *Let V be any vector space over K and $T: V \to V$ a linear map with invariant factors $f_1, \ldots, f_k \in \mathsf{K}[x]$, where $f_i \mid f_{i-1}$ for $2 \le i \le k$. There exists a maximal $v_1 \in V$ such that $\min(T; v_1) = \min(T; V) = f_1$, and for all such v_1 there exists a T-invariant subspace $V' \subseteq V$ such that $V = \mathrm{Orb}(T; v_1) \oplus V'$. Furthermore, the invariant factors of T on V' are $f_2, \ldots, f_k$.*

The utility of this lemma comes in the realization that, for v_1 and V' as above, V' is isomorphic as a $\mathsf{K}[x]$-module to $V/\mathrm{Orb}(T; v_1)$. This follows since every element $w \in V$ can be written uniquely as $w = \bar{w} + w'$ for some $\bar{w} \in \mathrm{Orb}(T; v_1)$ and $w' \in V'$, whence $w \equiv w' \bmod \mathrm{Orb}(T; v_1)$. Applying Fact 1.2 recursively to V' yields a decomposition of V as follows (see Gantmacher 1990, Section 7.4).

Fact 1.3 (Modular Cyclic Decomposition). *For $i \ge 1$ let $w_i \in V$ and define $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$ and $T_i: V/V_i \to V/V_i$ as T reduced modulo V_i (with $V_0 = \{0\}$ and $T_0 = T$). Assume that k is the smallest integer such that $V_k = V$, and that for $1 \le i \le k$ we have $\min(T_{i-1}; w_i + V_{i-1}) = \min(T_{i-1}; V/V_{i-1})$. Then $\min(T_{i-1}; w_i + V_{i-1}) = f_i$, the ith invariant factor of T on V, for $1 \le i \le k$.*

If $w_1, \ldots, w_k$ are as in the above theorem, then we say that they generate a *modular cyclic basis*

$$w_1, Tw_1, \ldots, T^{d_1 - 1} w_1, w_2, Tw_2, \ldots, T^{d_2 - 1} w_2, \ldots, w_k, Tw_k, \ldots, T^{d_k - 1} w_k$$

for V. Of course the summation $V = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_k)$ in Fact 1.3 is not direct, and once we have found $w_1, \ldots, w_k$, we must somehow "purify" them to get a direct sum. This is accomplished in Section 3.

Our approach to finding $w_1, \ldots, w_k \in V$ is to choose them randomly. This will only work when $\#\mathsf{K} \ge n^2$. When $\#\mathsf{K} < n^2$, we choose a small extension field F of K such that $\#\mathsf{F} \ge n^2$. This will be discussed in Section 4. For the remainder of this section, assume that $\#\mathsf{K} \ge n^2$.

Lemma 1.4. *Let W be any vector space of dimension at most n over K, spanned by $u_1, \ldots, u_n \in W$, and let $T: W \to W$ be a K-linear map. Let L be a subset of K containing at least n^2 elements. Then*

$$\operatorname*{Prob}_{(a_1, \ldots, a_n) \in L^n} \left\{ \min(T; \sum_{1 \le i \le n} a_i u_i) = \min(T; W) \right\} \ge 1 - \frac{1}{n}.$$

Proof. Let $y_1, \ldots, y_l \in W$ be such that

$$W = \mathrm{Orb}(T; y_1) \oplus \mathrm{Orb}(T; y_2) \oplus \cdots \oplus \mathrm{Orb}(T; y_l)$$

is the cyclic decomposition of W with respect to T, where T has l invariant factors. For $1 \le i \le l$, let $g_i = \min(T; y_i) \in \mathsf{K}[x]$ be the ith invariant factor of T, so $g_i \mid g_{i-1}$ for $2 \le i \le l$.

We choose random elements $a_1, \ldots, a_n \in L$ and assign $v = \sum_{1 \le i \le n} a_i u_i \in W$. In what follows we only consider the component of v in $\mathrm{Orb}(T; y_1)$. Let $h_i \in \mathsf{K}[x]$ be such that the component of u_i in $\mathrm{Orb}(T; y_1)$ is $h_i \circ y_1$, with $\deg h_i < \deg g_1$, for $1 \le i \le l$. The component of v in $\mathrm{Orb}(T; y_1)$ is then $v_1 = \sum_{1 \le i \le n} a_i h_i \circ y_1$. Let $T_1 \colon \mathrm{Orb}(T; y_1) \to \mathrm{Orb}(T; y_1)$ be the restriction of T to $\mathrm{Orb}(T; y_1)$. Certainly $\min(T; v) = g_1$ when $\min(T_1; v_1) = g_1$; $\min(T; v)$ must annihilate *all* of v's components. When is $\min(T_1; v_1) = g_1$? Let $h = \sum_{1 \le i \le n} a_i h_i$, so that $v_1 = h \circ y_1$. Clearly $g_1 \circ v_1 = (g_1 h) \circ y_1 = 0$ so $g = \min(T; v_1)$ divides g_1. Also, $(gh) \circ y_1 = 0$, which is true only when $gh \equiv 0 \bmod g_1$. If $\gcd(h, g_1) = 1$ then $g \equiv 0 \bmod g_1$, whence $g = g_1$. Conversely, if $\gcd(h, g_1) \ne 1$ then g is a proper divisor of g_1. In summary, $\min(T; v_1) = g_1$ if and only if $\gcd(h, g_1) = 1$.

To determine the probability that, for randomly chosen $a_1, \ldots, a_n \in L$, we have $\gcd(\sum_{1 \le i \le n} a_i h_i, g_1) = 1$, consider the polynomial $\rho = \sum_{1 \le i \le n} x_i h_i \in \mathsf{K}[x_1, \ldots, x_n][x]$, in the indeterminates $x_1, \ldots, x_n, x$. To prove the theorem it is sufficient to show that

$$\operatorname*{Prob}_{(a_1,\ldots,a_n)\in L^n} \Big\{ \gcd(\rho(a_1, \ldots, a_n)(x), g_1(x)) = 1 \Big\} \ge 1 - \frac{1}{n}.$$

This is accomplished with the aid of resultants (see van der Waerden 1970, Section 5.8). The resultant of ρ and g_1, considered as polynomials in x with coefficients in the integral domain $\mathsf{K}[x_1, \ldots, x_n]$ is a polynomial $R \in \mathsf{K}[x_1, \ldots, x_n]$ of degree at most n. Moreover, $\gcd(\rho(a_1, \ldots, a_n)(x), g_1(x)) = 1$ if and only if $R(a_1, \ldots, a_n) \ne 0$.

The polynomial R is non-zero as follows. The component of y_1 in $\mathrm{Orb}(T; y_1)$ is y_1 itself, and hence has minimal polynomial g_1. Assume $y_1 = \sum_{1 \le j \le n} b_j u_j$ for some $b_1, \ldots, b_n \in \mathsf{K}$. Then $\sum_{1 \le j \le n} b_j h_j \equiv 1 \bmod g_1$, and since each h_j has degree less than g_1 by definition, $\sum_{1 \le j \le n} b_j h_j = 1$. Thus, $R(b_1, \ldots, b_n) \ne 0$.

We now apply Corollary 1 of Schwartz (1980) to obtain

$$\operatorname*{Prob}_{(a_1,\ldots,a_n)\in L^n} \Big\{ R(a_1, \ldots, a_n) \ne 0 \Big\} \ge 1 - \frac{n}{\#L} \ge 1 - \frac{1}{n}.$$

□

A modular cyclic decomposition is now constructed by simply choosing the generating vectors "randomly" from V. Actually, since we do not know the number k of invariant factors beforehand, we choose $w_1, \ldots, w_n \in V$ and consider the probability that the first k of these generate V in the desired manner.

THEOREM 1.5. *Let L be a subset of K containing at least n^2 elements. For $1 \le i \le n$, randomly choose $w_i \in L^{n \times 1} \subseteq V$ and define $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$ and $T_i \colon V/V_i \to V/V_i$ as T reduced modulo V_i (with $V_0 = \{0\}$ and $T_0 = T$). With probability at least $1/4$, for all $1 \le i \le k$ (where k is the number of invariant factors of T) the ith invariant factor f_i of T satisfies $f_i = \min(T_{i-1}; w_i + V_{i-1}) = \min(T_{i-1}; V/V_{i-1})$.*

Proof. First we note, for $1 \le i \le n$, that V_i is a T-invariant subspace, so V/V_i is well defined as a $\mathsf{K}[T]$-module. We show that for each $1 \le i \le n$ that the probability that

$\min(T_{i-1}; w_i + V_{i-1}) = \min(T_{i-1}; V/V_{i-1})$ is at least $1 - 1/n$. To see this, consider the standard basis $e_1, \ldots, e_n \in \mathsf{K}^{n\times 1}$ for V, i.e., e_i is the vector with a one in the ith row and zeros in all other rows. We have chosen $w_i = \sum_{1\le i\le n} a_i e_i$ for some randomly selected $a_1, \ldots, a_n \in L$. The vectors $e_1 + V_{i-1}, \ldots, e_n + V_{i-1}$ span V/V_{i-1} and

$$w_i + V_{i-1} = \sum_{1\le i\le n} a_i e_i + V_{i-1} = \sum_{1\le i\le n} a_i(e_i + V_{i-1}) \in V/V_{i-1}.$$

Thus, by Lemma 1.4,

$$\mathrm{Prob}\Big\{\min(T_{i-1}; w_i + V_{i-1}) = \min(T_{i-1}; V/V_{i-1})\Big\} \ge 1 - 1/n.$$

The number of components k in the cyclic decomposition is certainly at most n, whence

$$\prod_{1\le i\le k} \mathop{\mathrm{Prob}}_{w_i\in L^{n\times 1}} \Big\{\min(T_{i-1}; w_i + V_{i-1}) = \min(T_{i-1}; V/V_{i-1})\Big\} \ge \left(1 - \frac{1}{n}\right)^n \ge \frac{1}{4}.$$

If all the w_i are chosen correctly then we also know, from Fact 1.3, that $f_i = \min(T_{i-1}; w_i + V_{i-1})$ for $1 \le i \le k$. $\square$

2. Computing the Invariant Factors of T on V. Assume now that we are given vectors $w_1, \ldots, w_n \in V$ for V such that $w_1, \ldots, w_k$ generate a modular cyclic basis for V, and let V_i, T_i be as in Fact 1.3, for $0 \le i \le k$. How do we find k and the invariant factors $f_1, \ldots, f_k$? We adapt an algorithm of Keller-Gehrig (1985) to accomplish this with $O(\mathrm{MM}(n)\log n)$ operations in K.

FACT 2.1 (KELLER-GEHRIG 1985). *Let $T \in \mathsf{K}^{n\times n}$ and $u_1, \ldots, u_n \in V$. Matrices $H_i \in \mathsf{K}^{n\times d_i}$ for $1 \le i \le n$ with the following properties can be computed with $O(\mathrm{MM}(n)\log n)$ operations in K:*

(i) $\sum_{1\le i\le n} d_i = n$, where $d_i \in \mathbb{N}$ may equal 0 for notational convenience;
(ii) for $1 \le i \le n$, $H_i = [u_i|Tu_i|\cdots|T^{d_i-1}u_i]$ if $d_i > 0$;
(iii) the columns of $H_1, \ldots, H_i$ form a basis of the $\mathsf{K}[x]$-module generated by $u_1, \ldots, u_i$, for $1 \le i \le n$ (that is, for $\mathrm{Orb}(T; u_1) + \cdots + \mathrm{Orb}(T; u_i)$).

Keller-Gehrig's (1985) algorithm is essentially an asymptotically fast version of the algorithm of Danilevsky (1937) for computing the characteristic polynomial of a matrix (see Faddeev & Faddeeva 1963). Applying Fact 2.1 to the vectors $w_1, \ldots, w_n$ yields k, bases for $V_1, \ldots, V_k$, and the degrees of the invariant factors $f_1, \ldots, f_k$. That is, the columns of $H_1, \ldots, H_i$ form a basis for $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$ and $\deg f_i = d_i$ for $1 \le i \le k$. We know $d_i = 0$ for $k < i \le n$ since $w_1, \ldots, w_k$ are assumed to generate V as a $\mathsf{K}[x]$-module.

Let $H = [H_1|H_2|\cdots|H_k] \in \mathsf{K}^{n\times n}$, which we call a *modular cyclic transition matrix* for T. Combining Fact 2.1 with Theorem 1.5 gives a probabilistic algorithm to find a modular cyclic transition matrix, which is correct with probability at least $1/4$. We summarize this algorithm below.

Algorithm: `FindModCyc`

Input: $T \in \mathsf{K}^{n\times n}$, where $\#\mathsf{K} \geq n^2$ and L is a subset of K with $\#L \geq n^2$;

Output: a modular cyclic transition matrix $H = [H_1|\cdots|H_k] \in \mathsf{K}^{n\times n}$ for T,
where $H_i = [w_i|\cdots|T^{d_i-1}w_i] \in \mathsf{K}^{n\times d_i}$ for some $w_i \in V$ $(1 \leq i \leq k)$;

(1) Choose $w_1, \ldots, w_n$ randomly from $L^{n\times 1}$.

(2) Find $H = [H_1|\cdots|H_k]$ such that $H_i = [w_i|\cdots|T^{d_i-1}w_i] \in \mathsf{K}^{n\times d_i}$, where
- $\sum_{1\leq i\leq k} d_i = n$
- for $1 \leq i \leq k$, the columns of $H_1, \ldots, H_i$ form a basis of the $\mathsf{K}[x]$-module generated by $w_1, \ldots, w_i$,

using Keller-Gehrig's algorithm.

End.

THEOREM 2.2. *Let $\#\mathsf{K} \geq n^2$. Given $T \in \mathsf{K}^{n\times n}$, the algorithm* `FindModCyc` *returns $w_1, \ldots, w_k \in V$ and matrices $H_i \in \mathsf{K}^{n\times d_i}$ for $1 \leq i \leq k$ defined as follows:*

(i) $\sum_{1\leq i\leq k} d_i = n$ *with* $d_i > 0$ *for* $1 \leq i \leq k$,

(ii) $H_i = [w_i|\cdots|T^{d_i-1}w_i]$ *for* $1 \leq i \leq k$,

(iii) The columns of $H_1, \ldots, H_i$ *form a basis for* $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$,

(iv) $f_i = \min(T \bmod V_{i-1}; w_i + V_{i-1}) = \min(T \bmod V_{i-1}; V/V_{i-1})$ *for* $1 \leq i \leq k$,

i.e., H is a modular cyclic transition matrix for T. The algorithm is probabilistic, and returns the correct answer with probability at least $1/4$ (it may produce an incorrect answer). It requires $O(\mathrm{MM}(n)\log n)$ operations in K.

When H is a modular cyclic transition matrix it determines a change of basis on V under which T has the form

$$G = H^{-1}TH = \begin{pmatrix} C_{f_1} & B_2 & \cdots & B_k \\ & C_{f_2} & & \\ & & \ddots & \\ 0 & & & C_{f_k} \end{pmatrix} \in \mathsf{K}^{n\times n}, \qquad (2.3)$$

where $C_{f_i} \in \mathsf{K}^{d_i\times d_i}$ is the companion matrix of f_i, the ith invariant factor of T for $1 \leq i \leq k$ (see Keller-Gehrig 1985, Section 5). Each matrix $B_i \in \mathsf{K}^{t_i\times d_i}$, where $t_i = \sum_{1\leq j<i} d_j$, is zero except for its last column; suppose this last column is

$$\vec{b}_i = (b_{i,1,0}, \ldots, b_{i,1,d_1-1}, b_{i,2,0}, \ldots, b_{i,2,d_2-1}, \ldots, b_{i,i-1,0}, \ldots, b_{i,i-1,d_{i-1}-1})^t \in \mathsf{K}^{t_i\times 1},$$

where $b_{ijl} \in \mathsf{K}$ for $2 \leq i \leq k$, $1 \leq j < i$, and $0 \leq l < d_j$. Under the change of basis induced by H, the vector $e_{t_i+j+1} \in \mathsf{K}^{n\times 1}$ (the column vector of all zeros except for a one in the $(t_i + j + 1)$'st row) in the basis given by the columns of H is the image of $T^j w_i$ in the standard basis for V.

The column $\vec{b}_i$ in B_i and the last column $(-a_{i,0}, \ldots, -a_{i,d_i-1})^t \in \mathsf{K}^{d_i\times 1}$ of C_{f_i} give the dependency of $T^{d_i}w_i$ on $T^l w_j$ for $1 \leq j \leq i$ and $0 \leq l < d_i$ by

$$Ge_{t_i+d_i} = H^{-1}T^{d_i}w_i = H^{-1}\sum_{0\leq l<d_i}(-a_{il})T^l w_i + H^{-1}\sum_{1\leq j<i}\sum_{0\leq l<d_j} c_{ijl}T^l w_j,$$

whence

$$T^{d_i}w_i + \sum_{0\le l<d_i} a_l T^l w_i = \sum_{1\le j<i}\sum_{0\le l<d_j} c_{ijl}T^l w_j,$$

or more simply

$$f_i \circ w_i = \sum_{1\le j<i} g_{ij}\circ w_j,$$

where $f_i = \sum_{0\le l\le d_i} a_{il}x^l$, and $g_{ij} = \sum_{0\le l<d_j} c_{ijl}x^l$ with $a_{il}, c_{ijl} \in \mathsf{K}$ for $1 \le j < i$ and $1 \le i \le k$. Thus $f_i \circ w_i \in V_{i-1}$, and since it has minimal degree, is the ith invariant factor of T on V. Note that by construction, $\sum_{1\le j<i} \deg g_{ij} \le n$ for all $1 \le i \le k$, a fact which will be required in the next section.

THEOREM 2.3. *Let $w_1, \ldots, w_n \in V$ be a basis for V such that $w_1, \ldots, w_k$ generate a modular cyclic basis for V, with $f_i = \min(T_{i-1}; w_i + V_{i-1})$ the ith invariant factor of T, for $1 \le i \le k$ and V_i, T_i as in Fact 1.3. Suppose also that we have computed a modular cyclic transition matrix for T. Then we can determine $k \in \mathbb{N}$, $f_1, \ldots, f_k \in \mathsf{K}[x]$, and $g_{ij} \in \mathsf{K}[x]$ for $1 \le j < i \le k$ such that $f_i \circ w_i = \sum_{1\le j<i} g_{ij}\circ w_j$, with $O(\mathrm{MM}(n))$ operations in K. Furthermore, $\sum_{1\le j<i} \deg g_{ij} \le n$ for all $1 \le i \le k$.*

Combining this theorem with Theorem 2.2 gives a Monte Carlo type probabilistic algorithm for computing the Frobenius form of T. Simply use the algorithm `FindModCyc` to find a modular cyclic transition matrix $H \in \mathsf{K}^{n\times n}$ for T. With probability at least 1/4 the companion blocks of the the invariant factors $f_1, \ldots, f_k \in \mathsf{K}[x]$ of T are along the diagonal of $H^{-1}TH$, from which we can construct the Frobenius form S of T. Of course, with positive probability we will get an erroneous Frobenius form, and since we do not get an invertible matrix $U \in \mathsf{K}^{n\times n}$ such that $S = U^{-1}TU$ from this procedure, we have no way of verifying that S is correct. This problem is addressed in the next section.

3. Purifying the Modular Decomposition. Once again, let $w_1, \ldots, w_k \in V$ generate a modular cyclic basis for V, where V_i and T_i are as in Fact 1.3 for $0 \le i \le n$, and $f_i = \min(T_{i-1}; w_i + V_{i-1})$ is the ith invariant factor of T on V for $1 \le i \le k$. Furthermore, assume that $f_i \circ w_i = \sum_{1\le j<i} g_{ij}\circ w_j$ for some $g_{ij} \in \mathsf{K}[x]$ for all $1 \le j < i \le k$. This information can all be computed with the algorithm of the previous section (see Theorem 2.3), along with the matrix H defined there whose columns give a modular decomposition basis for V. In this section vectors $v_1, \ldots, v_k \in V$ are constructed such that

$$V = \mathrm{Orb}(T; v_1) \oplus \mathrm{Orb}(T; v_2) \oplus \cdots \oplus \mathrm{Orb}(T; v_k),$$

and $f_i = \min(T; v_i)$, for $1 \le i \le k$. The key fact required is from Hoffman & Kunze (1971), Section 7.2, Theorem 3, Step 2:

FACT 3.1. *If $w_i \in V$, and $f_i, g_{ij} \in \mathsf{K}[x]$ are as above, then $f_i \,|\, g_{ij}$ for $1 \le j < i \le k$.*

Applying Fact 3.1, assume that $g_{ij} = f_i h_{ij}$ for some $h_{ij} \in \mathsf{K}[x]$, for all $1 \le j < i \le k$. We claim that the vectors $v_i = w_i - \sum_{1\le j<i} h_{ij}\circ w_j \in V$, for $1 \le i \le k$ are the ones desired.

THEOREM 3.2. *Let $w_1, \ldots, w_k \in V$ and $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$ such that $V_k = V$ and $V_{k-1} \ne V$. Also, let $f_i = \min(T_{i-1}; w_i + V_{i-1})$ such that $f_i \circ w_i = \sum_{1\le j<i} g_{ij}\circ w_j$ and $g_{ij} = f_i h_{ij}$ for $g_{ij}, h_{ij} \in \mathsf{K}[x]$ with $1 \le j < i \le k$. If $v_i = w_i - \sum_{0\le j<i} h_{ij}\circ w_j$, then for $1 \le i \le k$,*

(i) $f_i = \min(T; v_i)$,
(ii) $V_i = \mathrm{Orb}(T; v_1) \oplus \mathrm{Orb}(T; v_2) \oplus \cdots \oplus \mathrm{Orb}(T; v_i)$.

Proof. To show (i), note that for any i with $1 \le i \le k$ we have

$$f_i \circ v_i = f_i \circ w_i - f_i \circ \left(\sum_{1 \le j < i} h_{ij} \circ w_j \right) = f_i \circ w_i - \sum_{1 \le j < i} f_i h_{ij} \circ w_j$$
$$= f_i \circ w_i - \sum_{1 \le j < i} g_{ij} \circ w_j = 0.$$

Furthermore, since f_i is the polynomial of least degree such that $f_i \circ w_i \in V_{i-1}$, it follows that $f_i = \min(T; v_i)$.

To show (ii), we must show that $V_{i-1} \cap \mathrm{Orb}(T; v_i) = \{0\}$, or equivalently that if $g \circ v_i \in V_{i-1}$ then $g \circ v_i = 0$ for any $g \in \mathsf{K}[x]$. Suppose $g \circ v_i \in V_{i-1}$ for some polynomial $g \in \mathsf{K}[x]$. This is true only if $g \circ w_i \in V_{i-1}$, since $v_i \equiv w_i \bmod V_{i-1}$. But $f_i = \min(T_{i-1}; w_i + V_{i-1})$, so $f_i \,|\, g$, and $g \circ v_i = 0$. Then $V_{i-1} \cap \mathrm{Orb}(T; v_i) = \{0\}$ and $V_i = V_{i-1} \oplus \mathrm{Orb}(T; v_i)$. □

Note that this theorem does not assume that $w_1, \ldots, w_k$ generate a modular cyclic basis for V, only that they generate V as a $\mathsf{K}[x]$-module.

To compute $v_1, \ldots, v_k$, first compute $h_{ij} = g_{ij}/f_i \in \mathsf{K}[x]$ for all $1 \le j < i \le k$. For each $1 \le i \le k$ we note that $\sum_{1 \le j < i} \deg g_{ij} \le n$, so we can compute all g_{ij} for $1 \le j < i$ with $O(\mathrm{M}(n))$ operations in K. The number k of invariant factors is at most n, so all the g_{ij}'s can be found with $O(n\mathrm{M}(n))$ operations in K.

Computing the h_{ij}'s gives a representation of the v_i's in the modular decomposition basis for V given by the columns of H, not in the original (standard) basis for V. To find v_i in the standard basis, recall

$$v_i = w_i - \sum_{1 \le j < i} h_{ij} \circ w_j = w_i - \sum_{1 \le j < i} \sum_{0 \le l < d_j} c_{ijl} T^l w_j.$$

We compute v_i by the matrix-vector product $v_i = H\bar{v}_i \in V$, where

$$\bar{v}_i = (-c_{i,1,1}, \ldots, -c_{i,1,d_2-1}, \ldots, -c_{i,i-1,1}, \ldots, -c_{i,i-1,d_{i-1}}, 1, 0, \ldots, 0)^t \in \mathsf{K}^{n \times 1}.$$

All of $v_1, \ldots, v_k$ can now be computed by a single matrix product

$$H \cdot [\bar{v}_1 | \cdots | \bar{v}_k] = [v_1 | \cdots | v_k],$$

with $O(\mathrm{MM}(n))$ operations in K.

THEOREM 3.3. *Given $w_i \in V$ and $f_i, g_{ij} \in \mathsf{K}[x]$ for $1 \le i \le k$ and $1 \le j < i$ as above, we can compute $v_1, \ldots v_k \in V$ such that*

$$V = \mathrm{Orb}(T; v_1) \oplus \cdots \oplus \mathrm{Orb}(T; v_k)$$

and $f_i = \min(T; v_i)$ with $O(\mathrm{MM}(n) + n\mathrm{M}(n))$ operations in K.

4. Computing the Frobenius Form. The complete algorithm for computing the Frobenius normal form of any $T \in \mathsf{K}^{n \times n}$, where $\#\mathsf{K} \ge n^2$, can now be stated. A modification which works over smaller fields is presented in the sequel.

Algorithm: `FrobeniusForm`

Input: $T \in \mathsf{K}^{n\times n}$, where $\#\mathsf{K} \geq n^2$;

Output: – The Frobenius form $S \in \mathsf{K}^{n\times n}$ of T,
– an invertible $U \in \mathsf{K}^{n\times n}$ such that $S = U^{-1}TU$;

(1) Using `FindModCyc`, compute a modular cyclic transition matrix $H = [H_1|\cdots|H_k] \in \mathsf{K}^{n\times n}$, where
- $H_i = \left[w_i|Tw_i|\cdots|T^{d_i-1}w_i\right] \in \mathsf{K}^{n\times d_i}$ for $1 \leq i \leq k$,
- the columns of $H_1, \ldots, H_i$ span the $\mathsf{K}[x]$-module generated by the vectors $w_1, \ldots, w_i$ for $1 \leq i \leq k$;

An erroneous computation of H will be detected in step (2) or (4).

(2) If H is not invertible then quit, returning "failure";

(3) Compute $f_i, g_{ij} \in \mathsf{K}[x]$ from $G = H^{-1}TH$ such that $f_1 \circ w_1 = 0$ and $f_i \circ w_i = \sum_{1\leq j<i} g_{ij} \circ w_j$ for $2 \leq i \leq k$ and $1 \leq j < i$;

(4) If $f_i \nmid f_{i-1}$ for some $2 \leq i \leq k$, or $f_i \nmid g_{ij}$ for some $1 \leq j < i$ and $1 \leq i \leq k$, then return "failure" and quit;

(5) Compute $h_{ij} = g_{ij}/f_i$ for $1 \leq j < i$ and $1 \leq i \leq k$;

(6) Compute $v_i = w_i - \sum_{1\leq j<i} h_{ij} \circ w_j$ for $1 \leq i \leq k$;

(7) Compute the matrix $U \in \mathsf{K}^{n\times n}$ as in (1.2);

(8) Output the Frobenius form $S = \mathrm{diag}(C_{f_1}, \ldots, C_{f_k})$ and the transition matrix U (where C_{f_i} is the companion matrix of f_i for $1 \leq i \leq k$).

End.

THEOREM 4.1. *Let $T \in \mathsf{K}^{n\times n}$ over any field K with at least n^2 elements. With probability at least $1/4$,* `FrobeniusForm` *returns the Frobenius form S of T and an invertible $U \in \mathsf{K}^{n\times n}$ such that $S = U^{-1}TU$. Otherwise the algorithm reports "failure". In either case $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ operations in K are sufficient.*

Proof. First we prove the correctness of the output. We employ the algorithm `FindModCyc`, analyzed in Theorem 2.2, to compute a modular cyclic transition matrix H, which will be correct with probability at least $1/4$. Assume for now that H is computed correctly. By Fact 3.1 we know $f_i \,|\, g_{ij}$ for $1 \leq j < i$ and $1 \leq i \leq k$, and certainly $f_i \,|\, f_{i-1}$ for $2 \leq i \leq k$. Theorem 3.2 guarantees that $v_1, \ldots, v_k$ generate the cyclic composition factors of V and Gantmacher (1990), Section 7.5.2, shows that $S = U^{-1}TU$ is in Frobenius normal form.

If H is incorrectly computed, the algorithm reports failure in step (2) or (4). Certainly if H is singular then it is incorrect, and this is detected in step (2). Otherwise, assume that the tests in step (4) are passed, but $w_1, \ldots, w_k$ do not generate a modular cyclic basis. By Theorem 3.2, $V = \mathrm{Orb}(T; v_1) \oplus \cdots \oplus \mathrm{Orb}(T; v_k)$. Let $l \geq 1$ be the smallest integer such that $f_l \neq \bar{f}_l$, where $\bar{f}_l$ is the the actual lth invariant factor of T on V. In the vector space

$$\bar{V}_l = \mathrm{Orb}(T; v_l) \oplus \cdots \oplus \mathrm{Orb}(T; v_k) \cong V/V_{l-1}$$

there exists a vector $w \in \bar{V}$ with $\min(T_{l-1}; w + V_{l-1}) = \bar{f}_l$ (where $T_{l-1}\colon V/V_{l-1} \to V/V_{l-1}$ is T reduced modulo V_{l-1}). Since $f_j \,|\, f_l$ for $l \leq j \leq k$, we know $f_l = \min(T_{l-1}; \bar{V}_l) = \bar{f}_l$, a contradiction.

By Theorem 2.2, step (1) requires $O(\mathrm{MM}(n)\log n)$ operations in K. Using Theorem 2.3, steps (2)-(4) require $O(\mathrm{MM}(n))$ operations in K (H is singular exactly when its last column is zero). By Theorem 3.3 steps (5)-(6) require $O(\mathrm{MM}(n))$ operations in K to complete. The matrix U in step (7) can be found using Fact 2.1, with $O(\mathrm{MM}(n)\log n)$ operations in K. $\square$

The algorithm `FrobeniusForm` requires that the field K contain at least n^2 elements. If this is not the case, and $q = \#\mathsf{K} < n^2$, the chances of choosing $w_1, \ldots, w_k$ correctly may be very low. To remedy this we construct a field extension F of K containing n^2 elements. We then run the algorithm `FrobeniusForm` on $T \in \mathsf{F}^{n\times n}$. By Theorem 4.1 this correctly returns the Frobenius form $S \in \mathsf{F}^{n\times n}$ and an invertible $U \in \mathsf{F}^{n\times n}$ such that $S = U^{-1}TU$, with probability at least 1/4. However, the Frobenius form of T is a unique rational invariant of T, regardless of the field in which the elements of T are embedded. Thus $S \in \mathsf{K}^{n\times n}$ and this is precisely what we are looking for. The only drawback to this technique, aside from the slightly increased cost of operations in F, is that U is not necessarily a matrix over K.

To construct the field $\mathsf{F} \supseteq \mathsf{K}$, we use the deterministic algorithm of Shoup (1993) to find a polynomial $\psi \in \mathsf{K}[x]$ of degree $\lceil 2\log_q n\rceil$. Shoup's algorithm requires $O\tilde{~}(n)$ operations in K. Let $\mathsf{F} = \mathsf{K}[x]/(\psi)$, a finite field with at least n^2 elements, where each element is represented by a polynomial of degree less than $\lceil 2\log_q n\rceil$. Elements of K are simply represented as constant polynomials, giving a trivial embedding of K into F. Additions and subtractions in F require $O(\log_q n)$ operations in K, while multiplications in F require $O(\mathrm{M}(\log_q n))$ operations in K and divisions in F require $O(\mathrm{M}(\log_q n)\log\log_q n)$ operations in K.

THEOREM 4.2. *Let $T \in \mathsf{K}^{n\times n}$, where $q = \#\mathsf{K} < n^2$. The modification of* `FrobeniusForm` *described above computes the Frobenius form $S \in \mathsf{K}^{n\times n}$ of T, and an invertible matrix $U \in \mathsf{F}^{n\times n}$ such that $S = U^{-1}TU$. The field $\mathsf{F} = \mathsf{K}[x]/(\psi)$ is an extension of K, given by an irreducible monic $\psi \in \mathsf{K}[x]$ of degree $\lceil 2\log_q n\rceil$. The algorithm succeeds with probability at least 1/4, and otherwise reports "failure". It requires $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot \mathrm{M}(\log_q n)\log\log_q n)$ or $O\tilde{~}(\mathrm{MM}(n))$ operations in K.*

5. A Processor-Efficient Parallel Algorithm. In this section we exhibit a processor-efficient parallel algorithm for finding the Frobenius form of a matrix. Recall that computing the Frobenius form of a $T \in \mathsf{K}^{n\times n}$ is at least as hard as matrix multiplication, since from the Frobenius form we can quickly compute the determinant of T, which is known to be as hard as multiplying matrices (see Baur & Strassen, 1982). We demonstrate that our algorithm for computing the Frobenius form of any $T \in \mathsf{K}^{n\times n}$, can be implemented in a processor-efficient manner, and can be executed in (Las Vegas) time $(\log n)^{O(1)}$ using $O(\mathrm{MM}(n))$ processors.

Actually, the only part of the algorithm `FrobeniusForm` which requires modification, other than specifying an implementation technique for each step from the "toolkit" of known parallel algorithms, is the subroutine `FindModCyc` used in step (1). The routine of Keller-Gehrig's (1985), used in step (2) of `FindModCyc` in Section 2, is the heart of his asymptotically fast version of Danilevsky's (1937)'s algorithm for finding the characteristic polynomial of a matrix. It is not immediately clear that Keller-Gehrig's algorithm can be implemented in a processor-efficient manner. In this section we exhibit fast parallel versions of Keller-Gehrig's algorithm for computing the characteristic polynomial of a matrix and of our routine `FindModCyc`, both of which are processor-efficient. We then give the implementation details of a processor-efficient version of `FrobeniusForm`.

We begin by collecting a number of useful parallel algorithms which we require as subroutines.

FACT 5.1. *Suppose* K *is a field with characteristic* p.

(i) *Suppose that there is a bilinear algorithm which, given* $A, B \in \mathsf{K}^{n\times n}$ *computes* AB *with* $O(\mathrm{MM}(n))$ *operations in* K *sequentially (for a definition of bilinear algorithm see Borodin & Munro 1975), then there exists a processor-efficient parallel algorithm to compute* AB *on* $O(\mathrm{MM}(n)n^{\epsilon})$ *processors in time* $O(\log n)$, *for any* $\epsilon > 0$.

(ii) *Suppose* $\#\mathsf{K} \geq n$ *and* $A \in \mathsf{K}^{n\times n}$ *is invertible. If* $p \geq n$ *or* $p = 0$ *then we can compute* A^{-1} *with* $O(\mathrm{MM}(n))$ *processors in time* $O(\log^2 n)$. *If* $2 \leq p \leq n$ *there is a Las Vegas type probabilistic algorithm to compute* A^{-1} *with time* $O(\log^3 n/\log p)$ *on* $O(\mathrm{MM}(n))$ *processors.*

(iii) *If* $\#\mathsf{K} \geq n$ *then given vectors* $v_1, \ldots, v_m \in \mathsf{K}^{n\times 1}$, *where* $m \leq 2n$, *we can find the lexicographically first subset of* $v_1, \ldots, v_m$ *which form a basis for the vector space spanned by* $v_1, \ldots, v_m$. *If* $p \geq n$ *or* $p = 0$ *this can be accomplished with* $O(\log^4 n)$ *time on* $O(\mathrm{MM}(n)/\log n)$ *processors. If* $2 \leq p < n$ *this can be done with a Las Vegas algorithm requiring time* $O(\log^5 n/\log p)$ *on* $O(\mathrm{MM}(n)/\log n)$ *processors.*

(iv) *Given two polynomials* $f, g \in \mathsf{K}[x]$ *of degree at most* n, *we can compute* fg *in time* $O(\log n)$ *with* $O(n \log\log n)$ *processors.*

(v) *Given two polynomials* $f, g \in \mathsf{K}[x]$ *of degree at most* n, *we can compute the unique* $Q, R \in \mathsf{K}[x]$ *such that* $\deg R < \deg g$ *and* $f = Qg + R$ *in time* $O(\log n)$ *on* $O(n \log n)$ *processors.*

Proof. Part (i) is shown by Pan & Reif (1985) Theorem A.1. A processor-efficient algorithm for matrix inversion is given by Kaltofen & Pan (1991) and Kaltofen & Pan (1992). Eberly (1991) shows (iii). The fast polynomial multiplication algorithms of Cantor & Kaltofen (1991) also have parallel time and processor bounds as stated in Part (iv). Part (v) is shown by Bini & Pan (1992). □

We make the assumption, based on Fact 5.1(i), that there exists an algorithm which requires time $O(\log n)$ on $O(\mathrm{MM}(n))$ processors to multiply two $n \times n$ matrices over a field K.

Assume that we are given vectors $u_1, \ldots, u_n \in V$. As in Fact 2.1, we want to find matrices $H_i \in \mathsf{K}^{n\times d_i}$ for $1 \leq i \leq n$ with the following properties:

(i) $\sum_{1\leq i\leq n} d_i = n$, where $d_i \in \mathbb{N}$ may equal 0 for notational convenience;

(ii) for $1 \leq i \leq n$, $H_i = [u_i \mid Tu_i \mid \cdots \mid T^{d_i-1}u_i]$ if $d_i > 0$;

(iii) the columns of $H_1, \ldots, H_i$ form a basis of the $\mathsf{K}[x]$-module generated by $u_1, \ldots, u_i$, for $1 \leq i \leq n$.

We compute $H = [H_1 \mid \cdots \mid H_n]$ by computing matrices $H^{(0)}, H^{(1)}, \ldots, H^{(r)} \in \mathsf{K}^{n\times n}$ in sequence, where $r = \lceil \log_2 n \rceil$, $H^{(0)} = T$ and $H^{(r)} = H$. At step 0 we have $H^{(0)} = [H_1^{(0)} \mid \cdots \mid H_n^{(0)}]$, where $H_i^{(0)}$ is the ith column of T for $1 \leq i \leq n$. The idea is that at the end of stage j we will have computed $H^{(j)} = [H_1^{(j)} \mid \cdots \mid H_n^{(j)}]$, where

$$H_i^{(j)} = \left[u_i \mid Tu_i \mid \cdots \mid T^{d_i^{(j)}-1}u_i\right] \in \mathsf{K}^{n\times d_i^{(j)}}, \quad \text{for } 1 \leq i \leq n.$$

Here $d_i^{(j)}$ is the smallest integer less than 2^j such that $T^{d_i^{(j)}}u_i$ lies in the vector space spanned by $u_i, Tu_i, \ldots, T^{d_i^{(j)}-1}u_i$ along with all the columns of $H_1^{(j)}, \ldots, H_{i-1}^{(j)}$. If

$u_i, Tu_i, \ldots, T^{d_i^{(j)}} u_i$ and the columns of $H_1^{(j)}, \ldots, H_{i-1}^{(j)}$ are linearly independent then $d_i^{(j)} = 2^j$. This can be expressed more concisely in the language of modular vector spaces. Let $V_{i-1}^{(j)}$ be the vector space spanned by the columns of $H_1^{(j)}, \ldots, H_{i-1}^{(j)}$. Then $d_i^{(j)}$ is the smaller of 2^j and $\dim \mathrm{Orb}(T \bmod V_{i-1}^{(j)}; u_i \bmod V_{i-1}^{(j)})$. Recall that we allow $d_i^{(j)}$ to be zero, meaning that u_i is linearly dependent on the columns of $H_1^{(j)}, \ldots, H_{i-1}^{(j)}$. Since $2^r \geq n$, it is clear that $H^{(r)}$ will have the desired properties.

To find $H^{(j)}$ after computing $H^{(j-1)}$ (for $1 \leq j \leq r$), we first compute matrices $J_1^{(j)}, \ldots, J_n^{(j)}$ as follows:

$$J_i^{(j)} = \begin{cases} H_i^{(j-1)} & \text{if } H_i^{(j-1)} \text{ has fewer than } 2^j \text{ columns,} \\ [H_i^{(j-1)} \mid T^{2^j} H_i^{(j-1)}] & \text{otherwise.} \end{cases}$$

This doubles the number of iterates of u_i under T if this is required. Note that if $T^l u_i$ is linearly dependent upon V_i together with $u_i, Tu_i, \ldots, T^{l-1}u_i$ then so is $T^{l+1}u_i$. Thus, to find $d_1^{(j)}, \ldots, d_n^{(j)}$ we identify and mark the lexicographically first set of n linearly independent columns of

$$J^{(j)} = \left[J_1^{(j)} \mid \cdots \mid J_n^{(j)} \right].$$

The last column marked in $J_i^{(j)}$ is indexed by $d_i^{(j)}$. If none of the identified columns are in $J_i^{(j)}$ for some i then $d_i^{(j)} = 0$. Now let $H_i^{(j)} \in \mathsf{K}^{n \times d_i}$ consist of those columns of $J_i^{(j)}$ marked in this process.

This can be implemented in a processor-efficient manner using Fact 5.1. At stage j we compute $T^{2^j} = T^{2^{j-1}} \cdot T^{2^{j-1}}$, then we compute $J^{(j)}$ as described above from the product $T^{2^j} H^{(j-1)}$. The lexicographically first subset of the columns of $J^{(j)}$ are identified with the algorithm of Eberly (1991) (Fact 5.1(iii) in this paper), and $H^{(j)} \in \mathsf{K}^{n \times n}$ is constructed as above. This is repeated $r = \lceil \log_2 n \rceil$ times.

THEOREM 5.2. *If $p \geq n$ or $p = 0$ we can compute $H \in \mathsf{K}^{n \times n}$ and $d_1, \ldots, d_n \in \mathbb{N}$ as in Fact 2.1 in time $O(\log^4 n)$ on $O(\mathrm{MM}(n))$ processors. If $2 \leq p \leq n$ this algorithm requires time $O(\log^5 n / \log p)$ on $O(\mathrm{MM}(n))$ processors.*

Combining Theorem 5.2 with Theorem 1.5 allows us to implement the algorithm `FindModCyc` in Section 2 in a processor-efficient manner.

THEOREM 5.3. *Let $\#\mathsf{K} \geq n^2$. Given $T \in \mathsf{K}^{n \times n}$, the algorithm* `FindModCyc` *implemented in a processor-efficient manner as described above, returns $w_1, \ldots, w_k \in V$ and matrices $H_i \in \mathsf{K}^{n \times d_i}$ for $1 \leq i \leq k$ defined as follows:*

(i) $\sum_{1 \leq i \leq k} d_i = n$ with $d_i > 0$ for $1 \leq i \leq k$,
(ii) $H_i = [w_i | \cdots | T^{d_i - 1} w_i]$ for $1 \leq i \leq k$,
(iii) The columns of $H_1, \ldots, H_i$ form a basis for $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$,
(iv) $f_i = \min(T \bmod V_{i-1}; w_i + V_{i-1}) = \min(T \bmod V_{i-1}; V/V_{i-1})$ for $1 \leq i \leq k$.

The algorithm is probabilistic, and returns the correct answer with probability at least 1/4 (it may produce an incorrect answer). If $p \geq n$ or $p = 0$ it requires time $O(\log^4 n)$ on $O(\mathrm{MM}(n))$ processors. If $2 \leq p \leq n$ it requires time $O(\log^5 n / \log p)$ on $O(\mathrm{MM}(n))$ processors.

We now examine the parallel cost of the algorithm `FrobeniusForm`.

THEOREM 5.4. *The Las Vegas algorithm* `FrobeniusForm` *can be implemented in parallel in a processor-efficient manner. Let $T \in \mathsf{K}^{n \times n}$, where K is a field with $\#\mathsf{K} \geq n^2$.*

(i) *If* $p \geq n$ *or* $p = 0$ *then* `FrobeniusForm` *requires time* $O(\log^4 n)$ *on* $O(\mathrm{MM}(n))$ *processors.*
(ii) *If* $2 \leq p < n$ *then* `FrobeniusForm` *requires time* $O(\log^5 n / \log p)$ *on* $O(\mathrm{MM}(n))$ *processors.*

Proof. The statement of the parallel algorithm does not vary from the description in Section 4. Only the manner in which each step is executed changes, and we address these changes now. Step (1) is accomplished using the processor-efficient implementation of `FindModCyc` described in Theorem 5.3. The matrix H is singular exactly when its last column is zero, and this can be tested with constant time and work to perform step (2). As in the sequential algorithm, the coefficients of f_i and g_{ij} for $2 \leq i \leq k$ and $1 \leq j < i$ are read from $G = H^{-1}TH$ as in Theorem 2.3, and G can be computed within the desired parallel time and processor bounds by Fact 5.1. Steps (4)–(5) can be done using processor-efficient polynomial division algorithms, as specified in Fact 5.1 parts (iv) and (v). Step (6) can be performed by a single matrix product as in Theorem 3.3, and so can be done in a processor-efficient manner. Applying Theorem 5.3 gives us U in step (7). □

When $\#\mathsf{K} < n^2$ similar theorems hold, with running times and processor requirements multiplied by a small power of $\log_q n$.

THEOREM 5.5. *Let* K *be a field with* $q < n^2$ *and* $T \in \mathsf{K}^{n\times n}$. *We can implement the algorithm* `FrobeniusForm`, *as modified in Theorem 4.2, in a processor-efficient manner.*

(i) *If* $p \geq n$ *and* $p = 0$ *it requires time* $O(\log^4 n \log\log_q n)$ *on* $O(\mathrm{MM}(n) \cdot \mathrm{M}(\log_q n))$ *processors.*
(ii) *If* $2 \leq p < n$ *it requires time* $O(\log^5 n \log\log_q n / \log p)$ *on* $O(\mathrm{MM}(n)\cdot\mathrm{M}(\log_q n))$ *processors.*

Proof. The proof is the same as that of Theorem 5.4 except that we work in an extension F of K of algebraic degree $\lceil \log_q n \rceil$ over K, which does contain $O(n^2)$ elements (see Theorem 4.2). Each operations in F requires $O(\log\log_q n)$ operations in K on $O(\mathrm{M}(\log_q n))$ processors. □

Theorem 5.3 also has the following important corollary, essentially a processor-efficient version of the algorithms of Danilevsky (1937) and Keller-Gehrig (1985) for computing the characteristic polynomial of a matrix.

COROLLARY 5.6. *There is a processor-efficient Las Vegas algorithm for computing the characteristic polynomial* $f \in \mathsf{K}[x]$ *of any matrix* $T \in \mathsf{K}^{n\times n}$ *over any field* K. *If* $p \geq n$ *or* $p = 0$ *we can compute* f *in time* $O(\log^4 n)$ *on* $O(\mathrm{MM}(n))$ *processors. If* $2 \leq p < n$ *this algorithm requires time* $O(\log^5 n/\log p)$ *on* $O(\mathrm{MM}(n))$ *processors.*

Proof. We first find a matrix $H \in \mathsf{K}^{n\times n}$ as in Theorem 5.2 such that $G = H^{-1}TH$ is as in (2.3). The inverse of H and product $f_1 \cdots f_k$ can be computed within the desired time and processor bounds using the methods described in Fact 5.1. □

Note that the characteristic polynomial of $T \in \mathsf{K}^{n\times n}$ can also be computed as the product of the invariant factors of T, which can be read from the Frobenius form of T. The above corollary is a slight improvement on this for small fields. It relies on the fact that we do not need $H^{-1}TH$ to be in Frobenius form; the product of the polynomials whose companion matrices are on the diagonal of $H^{-1}TH$ always equals the characteristic polynomial of T.

6. Fast Evaluation of Matrix Functions. We now consider applications of our algorithm for finding the Frobenius form to evaluating polynomials at matrices, determining matrix similarity and finding the minimal polynomial of matrices. New algorithms requiring almost optimal time are presented. We also give lower bounds for the problems of evaluating a polynomial at matrix, and for finding the minimal polynomial of a matrix, which matches our upper bounds within poly-logarithmic factors.

Evaluating Polynomials at Matrices. Computing the Frobenius form $S \in \mathsf{K}^{n\times n}$ of a matrix $T \in \mathsf{K}^{n\times n}$ allows quick evaluation of $g(T) \in \mathsf{K}^{n\times n}$ for any polynomial $g \in \mathsf{K}[x]$. The problem of evaluating polynomials at matrices is certainly not new. Paterson & Stockmeyer (1973) give an algorithm to evaluate a polynomial $g \in \mathsf{K}[x]$ at any point in a ring extension $\Re$ of K, which requires $O(\sqrt{r})$ non-scalar multiplications in $\Re$, where $r = \deg g$. They apply their algorithm to evaluate $g(T)$ at any $T \in \mathsf{K}^{n\times n}$ with $O(\mathrm{MM}(n)\sqrt{r})$ operations in K (see also Brent & Kung 1978).

Theorem 6.2 gives a substantial improvement over this, especially when r is large, allowing evaluation of $g(T)$ with $O\tilde{~}(\mathrm{MM}(n) + r)$ operations in K. The well known observation required is that if $S = U^{-1}TU$ for some invertible $U \in \mathsf{K}^{n\times n}$, and $S = \mathrm{diag}(C_{f_1}, \ldots, C_{f_k})$ as in (0.1), then

$$g(T) = U \cdot g(S) \cdot U^{-1} = U \cdot \mathrm{diag}(g(C_{f_1}), \ldots, g(C_{f_k})) \cdot U^{-1}.$$

Furthermore, each $g(C_{f_i})$ can be evaluated quickly once $g \bmod f_i$ has been computed, and $g(S)$ can be computed with $O(n^2)$ operations in K when $\deg g \le n$.

LEMMA 6.1. *Let $C \in \mathsf{K}^{s\times s}$ be the companion matrix of $h \in \mathsf{K}[x]$, where $\deg h = s$. Also, let $g \in \mathsf{K}[x]$ have degree less than s. Then we can compute $g(C)$ with $O(s^2)$ operations in K, or in parallel time $O(\log s)$ on $O(s\mathrm{M}(s))$ processors.*

Proof. The vector space $\mathsf{K}^{s\times 1}$ has a $\mathsf{K}[x]$-module structure, where any $f \in \mathsf{K}[x]$ acts on $\mathsf{K}^{s\times 1}$ as $f(C) \in \mathsf{K}^{s\times s}$. As a $\mathsf{K}[x]$-module, $\mathsf{K}^{s\times 1}$ is isomorphic to $\mathsf{K}[x]/(h)$ — simply map $v = (b_0, \ldots, b_{s-1}) \in \mathsf{K}^{s\times 1}$ to $\varphi(v) = \sum_{0\le i<s} b_i x^i \bmod h$. It is easy to show that $x\varphi(v) \equiv \varphi(Cv) \bmod h$, and therefore that $f \cdot \varphi(v) = \varphi(f(C)v)$ for any $f \in \mathsf{K}[x]$. In particular, $\varphi(e_i) = x^{i-1}$, where $e_i \in \mathsf{K}^{n\times 1}$ is zero except for a one in the ith row, and hence $\varphi(g(C)e_i) \equiv g \cdot x^{i-1} \bmod h$, for $1 \le i \le s$. Since $g(C)e_i$ is the ith column of $g(C)$, we can find $g(C)$ by computing $g \cdot x^i \bmod h$ for $0 \le i < s$. This can be accomplished with $O(s^2)$ operations in K by realizing that computing $C \cdot v$ for any $v \in \mathsf{K}^{s\times 1}$, and hence multiplying by x modulo h, requires only $O(s)$ operations in K (C has only $2s$ non-zero entries). It can be done in time $O(\log s)$ on $O(s\mathrm{M}(s))$ processors using the polynomial multiplication algorithm of Cantor & Kaltofen (1991). $\square$

THEOREM 6.2. *Given $g \in \mathsf{K}[x]$ of degree r and $T \in \mathsf{K}^{n\times n}$, we can compute $g(T) \in \mathsf{K}^{n\times n}$ with a Las Vegas type probabilistic algorithm requiring $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n) + \mathrm{M}(r))$ operations in K. Here K is any field with at least n^2 elements.*

Proof. First compute the Frobenius form $S \in \mathsf{K}^{n\times n}$ of T, and an invertible $U \in \mathsf{F}^{n\times n}$ such that $U^{-1}TU = S$. This requires $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ operations in K using Theorem 4.1. For each invariant factor $f_i \in \mathsf{K}[x]$ of T, determine $g_i \equiv g \bmod f_i$, where $\deg g_i < \deg f_i$, for $1 \le i \le k$. This can be done with $O(\mathrm{M}(r) + n\mathrm{M}(n))$ operations in K by first reducing g modulo f_1, since all the other invariant factors divide f_1. Next compute the matrices $g_i(C_{f_i}) = g(C_{f_i})$ for $1 \le i \le k$. Using Lemma 6.1 above, this can be accomplished with $O(n^2)$ operations in K. Now $g(S) = \mathrm{diag}(g(C_{f_1}), \ldots, g(C_{f_k}))$ and $g(T) = Ug(S)U^{-1}$. $\square$

A more general theorem can be obtained using arithmetic circuits or straight-line programs (see Aho *et al.* 1974, Section 1.5 or von zur Gathen 1986 for a formal definition). This model gives a very natural computational representation of rational functions, nicely formalizing and generalizing such notions as sparseness and ease of computation. Briefly, an algebraic circuit or straight-line program $\wp$ over a field K consists of a finite list $s_1, s_2, \ldots, s_m$ of statements, each of the form $a_i := b_i \operatorname{op}_i c_i$, where a_i is a variable (assigned only at step i), $\operatorname{op}_i \in \{+, -, \times, /\}$, and each of b_i, c_i is either equal to a_j for some $1 \le j < i$, or is a constant in K, or is the *input* x. The output is the value of a_m upon evaluation of each of the steps in sequence (this semantics glosses over a number of subtleties — see von zur Gathen 1986). The sequential cost of $\wp$ is m. To define the parallel cost, note that $\wp$ defines a directed acyclic graph on the nodes $x, s_1, \ldots, s_k$, where there is an edge from s_i to s_j (or x to s_j) if s_j references a_i (resp. x). The parallel cost is the depth, or maximum path length from the node x to the node s_m.

Now let $\wp$ be algebraic circuit over K. Let F be an extension ring of K, so $\wp$ can be thought of as computing a function from $\mathsf{F} \to \mathsf{F}$. If $\wp$ has length m and depth d, then we can evaluate $\wp$ at a point $a \in \mathsf{F}$ on a sequential arithmetic RAM over F with $O(m)$ operations in F, or on an arithmetic PRAM over F in time $O(d)$ on $O(m)$ processors. We obtain for straight-line programs a theorem similar to Theorem 6.2.

THEOREM 6.3. *Let $\wp$ be a straight-line program over K as described above, with length m and depth d. Assume that K has characteristic p and $\#\mathsf{K} \ge n^2$. For any $T \in \mathsf{K}^{n\times n}$, we can evaluate $\wp$ at T in time as follows:*

(i) with $O(\mathrm{MM}(n)\log n + (n+m)\mathrm{M}(n))$ operations in K sequentially;
(ii) in parallel time $O(\log^4 n + d\log n)$ when $p \ge n$ or $p = 0$, or $O(\log^5 n/\log p + d\log n)$ when $2 \le p < n$, on $O(\mathrm{MM}(n)\log n + (n+m)\mathrm{M}(n))$ processors.

The algorithm is of the Las Vegas type, returns the correct answer with probability at least $1/4$ and otherwise reports "failure".

Proof. First compute the Frobenius form $S \in \mathsf{K}^{n\times n}$ of T, and an invertible $U \in \mathsf{F}^{n\times n}$ such that $U^{-1}TU = S$. Assume that $S = \operatorname{diag}(C_{f_1}, \ldots, C_{f_k})$, where $f_1, \ldots, f_k \in \mathsf{K}[x]$ are the invariant factors of T, and $C_{f_1}, \ldots, C_{f_k}$ are their respective companion matrices. Then evaluate $\wp$ at $x \bmod f_1$ in the ring $\mathsf{K}[x]/(f_1)$. If this evaluation fails, then we report that the circuit is undefined at T. Otherwise, assume the result of this evaluation yields $h \bmod f_1$ for some $h \in \mathsf{K}[x]$. Next compute $h_i \equiv h \bmod f_i$ for $1 \le i \le k$. The final result is then $U^{-1}\operatorname{diag}(h_1(C_{f_1}), \ldots, h_k(C_{f_k}))U \in \mathsf{K}^{n\times n}$. Correctness is clear from the ring-isomorphism between between $\mathsf{K}[T]$ and $\mathsf{K}[x] \bmod f_1$.

The cost of the algorithm is measured as follows. Computing the Frobenius form requires $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ operations in K using Theorem 4.1. In parallel, we can use Theorem 5.4, which requires $O(\log^4 n)$ time on $O(\mathrm{MM}(n))$ processors when $p \ge n$ or $p = 0$, or time $O(\log^5 n/\log p)$ on $O(\mathrm{MM}(n))$ processors when $2 \le p < n$. We can evaluate $\wp$ at x in $\mathsf{K}[x]/(f_1)$ with $O(m \cdot \mathrm{M}(n))$ operations in K sequentially, or time $O(d\log n)$ on $O(m \cdot \mathrm{M}(n))$ processors in parallel. The time required by the the remaining steps is dominated by the time required by these first two. $\square$

As an application of this straight-line program evaluation technique we consider computing high powers of matrices, an operation performed in some cryptographic systems (see, for example, Chuang & Dunham 1990). Using linear recurrences and companion matrices to compute powers of matrices is certainly not new, dating at least to Ranum (1911).

COROLLARY 6.4. *Given $s \geq 0$ and $T \in \mathsf{K}^{n\times n}$, we can compute $T^s \in \mathsf{K}^{n\times n}$ with a Las Vegas type probabilistic algorithm requiring $O(\mathrm{MM}(n)\log n + (n+\log s)\mathrm{M}(n))$ or $O\tilde{~}(\mathrm{MM}(n) + n\log s)$ operations in K, where K is any field with at least n^2 elements.*

Proof. The repeated squaring method of computing high powers of elements in any group yields a straight-line program of depth and length $O(\log s)$. Apply Theorem 6.3 to obtain the result. □

We now summarize the above results for fields with fewer than n^2 elements. These follow immediately from the use of Theorems 4.2 (sequential) and 5.5 (parallel) instead of Theorems 4.1 and 5.4 respectively.

THEOREM 6.5. *Let K be any field with $q < n^2$ elements and characteristic p, $T \in \mathsf{K}^{n\times n}$, and F the smallest extension field of K containing n^2 elements. Las Vegas type probabilistic algorithms exist as follows:*

(i) given $g \in \mathsf{K}[x]$ of degree r we can evaluate $g(T) \in \mathsf{K}^{n\times n}$ with $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot \mathrm{M}(\log_q n)\log\log_q n + \mathrm{M}(r))$ or $O\tilde{~}(\mathrm{MM}(n)+r)$ operations in K sequentially. In parallel we can do this in time $O(\log^4 n \log\log_q n + \log r)$ when $p = 0$ or $p \geq n$, or in time $O(\log^5 n\log\log_q n/\log p + \log r)$ when $2 \leq p < n$, on $O(\mathrm{MM}(n)\mathrm{M}(\log_q n) + \mathrm{M}(r))$ processors.

(ii) given a straight-line program $\wp$ with length m and depth d, we can evaluate $\wp$ at T with $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot\mathrm{M}(\log_q n)\log\log_q n + m\mathrm{M}(n))$ operations in K. In parallel we can do this in time $O(\log^4 n\log\log_q n + d\log n)$ when $p = 0$ or $p \geq n$, or in time $O(\log^5 n \log\log_q n/\log p + d\log n)$ when $2 \leq p < n$, on $O(\mathrm{MM}(n)\mathrm{M}(\log_q n) + m\cdot \mathrm{M}(n))$ processors.

A Lower Bound on Evaluating Polynomials at Matrices. In this subsection we show that evaluating a non-linear polynomial at a matrix is at least as hard as matrix multiplication. Specifically, if $g \in \mathsf{K}[x]$ has degree $r \geq 2$ and we can evaluate $g(T)$ at any $T \in \mathsf{K}^{n\times n}$ with at most $t(n)$ operations in K, then we can multiply two $n\times n$ matrices over K with $O(t(n))$ operations in K. We make a technical assumption about the cost function $t\colon \mathbb{R}_{>0} \to \mathbb{R}_{>0}$:

$$\forall a \in \mathbb{R}_{>0}\ \ \forall b \in \mathbb{R}_{\geq 1} \qquad b^2 t(a) \leq t(ab) \leq b^{3.5} t(a).$$

The problem of evaluating a polynomial at a matrix has a trivial lower bound of $\Omega(n^2)$ operations in K and, as discussed above, a deterministic upper bound of $O(n^{3.5})$ operations in K, making this assumption reasonable.

We will assume here that $\#\mathsf{K} \geq r-1$. A similar argument is presented in Giesbrecht (1993) for smaller fields. Let $A, B \in \mathsf{K}^{n\times n}$. We show how to use an algorithm for evaluating g at a matrix to multiply A and B. Consider the matrix

$$T = \begin{pmatrix} I_n & A & 0_n \\ 0_n & cI_n & B \\ 0_n & 0_n & I_n \end{pmatrix} \in \mathsf{K}^{3n\times 3n},$$

where $c \in \mathsf{K}$ and $0_n, I_n \in \mathsf{K}^{n\times n}$ are the zero and identity matrices respectively.

LEMMA 6.6. *For $j \geq 1$,*

$$T^j = \begin{pmatrix} I_n & (\sum_{0\leq i\leq j-1} c^i)\cdot A & (\sum_{0\leq i\leq j-2}(j-i-1)c^i)\cdot AB \\ 0_n & c^j I_n & (\sum_{0\leq i\leq j-1} c^i)\cdot B \\ 0_n & 0_n & I_n \end{pmatrix} \in \mathsf{K}^{3n\times 3n}.$$

Proof. We proceed by induction on j. Clearly the lemma is true for $j = 1$. Assume the lemma holds for $j-1$ (we assume $j \geq 2$). Then

$$T^j = T \cdot T^{j-1} = \begin{pmatrix} I_n & A & 0_n \\ 0_n & cI_n & B \\ 0_n & 0_n & I_n \end{pmatrix} \begin{pmatrix} I_n & (\sum_{0\leq i\leq j-2} c^i)\cdot A & (\sum_{0\leq i\leq j-3} (j-i-2)\, c^i)\cdot AB \\ 0_n & c^{j-1} I_n & (\sum_{0\leq i\leq j-2} c^i)\cdot B \\ 0_n & 0_n & I_n \end{pmatrix}$$

$$= \begin{pmatrix} I_n & (\sum_{0\leq i\leq j-1} c^i)\cdot A & (\sum_{0\leq i\leq j-2} (j-i-1)\, c^i)\cdot AB \\ 0_n & c^j I_n & (\sum_{0\leq i\leq j-1} c^i)\cdot B \\ 0_n & 0_n & I_n \end{pmatrix}.$$

□

Let $C \in \mathsf{K}^{n\times n}$ be the top right $n \times n$ block of $g(T)$. If $g = \sum_{0\leq i\leq r} b_i x^i$, then

$$C = \sum_{0\leq i\leq r} b_i \sum_{0\leq j\leq i-2} c^j AB = \left(\sum_{0\leq i\leq r-2} c^j \left(\sum_{i+2\leq j\leq r} (j-i-1) b_i \right)\right) \cdot AB = h(c)\cdot AB,$$

where $h = \sum_{0\leq i\leq r-2} z^i (\sum_{i+2\leq j\leq r} (j-i-1) b_j) \in \mathsf{K}[z]$ and z is an indeterminate. Since the coefficient of z^{r-2} in h is $b_r \neq 0$, we know $\deg h = r-2$. Now, we have assumed that $\#\mathsf{K} > r-2$, so there exists a $c \in \mathsf{K}$ such that $h(c) \neq 0$ because h has at most $r-2$ roots. Such a c can be found by evaluating h at $r-1$ distinct points in K, and using fast multi-point evaluation (see Aho *et al.* 1974, Section 8.5), can be found with $O(\mathrm{M}(r)\log r)$ operations in K. Now $AB = h(c)^{-1}C$, which can be computed with an additional $O(t(n))$ operations in K.

THEOREM 6.7. *Let $g \in \mathsf{K}[x]$ have degree $r \geq 2$. Suppose we can evaluate $g(T)$ for any matrix $T \in \mathsf{K}^{n\times n}$ with $t(n)$ operations in K. If $\#\mathsf{K} \geq r-1$ we can multiply two $n \times n$ matrices with $O(t(n) + \mathrm{M}(r)\log r)$ operations in K.*

In fact, for any given n, we need only compute a single $c \in \mathsf{K}$ such that $h(c) \neq 0$. After such a c has been found, evaluating g at any matrix in $\mathsf{K}^{n\times n}$ requires only $O(t(n))$ operations in K.

7. Computing the Rational Jordan Form. The rational Jordan form of a matrix $T \in \mathsf{K}^{n\times n}$ is a generalization of the usual Jordan form. Whereas a Jordan form of T exists only if K contains all eigenvalues of T, the rational Jordan form is a matrix in $\mathsf{K}^{n\times n}$. The eigenvalues of T, which may generate an algebraic extension of K of exponential degree over K, are replaced by the companion matrices of their minimal polynomials in $\mathsf{K}[x]$. For any monic irreducible $g \in \mathsf{K}[x]$ of degree r, and any

$m > 0$, define the *rational Jordan block* $J_g^{(m)} \in \mathsf{K}^{mr \times mr}$ as

$$J_g^{(m)} = \begin{pmatrix} \boxed{C_g} & \boxed{I_r} & & & \mathbf{0} \\ & \boxed{C_g} & \boxed{I_r} & & \\ & & \ddots & \ddots & \\ & \mathbf{0} & & \ddots & \boxed{I_r} \\ & & & & \boxed{C_g} \end{pmatrix} \in \mathsf{K}^{mr \times mr}, \tag{7.4}$$

where I_r is the $r \times r$ identity matrix, and $C_g \in \mathsf{K}^{r \times r}$ is the companion matrix of g. It is easily shown that both the minimal and characteristic polynomials of $J_g^{(m)}$ are equal to $g^m \in \mathsf{K}[x]$ (see 7.1 below). Any $T \in \mathsf{K}^{n \times n}$ is similar to its *rational Jordan form*, a matrix $Q = \operatorname{diag}(Q_1, Q_2, \ldots, Q_l) \in \mathsf{K}^{n \times n}$, where $Q_1, \ldots, Q_l$ are rational Jordan blocks. Notice that when all eigenvalues of T lie in K, then Q is in fact the usual Jordan form of T. The rational Jordan form is unique up to the order of the (rational) Jordan blocks, as is the usual Jordan form. Rational forms akin to the rational Jordan form were investigated at least as early as Frobenius (1911). Kaltofen *et al.* (1990) exhibit fast probabilistic parallel algorithms for a somewhat different rational Jordan form, as well as a "symbolic Jordan form" of a matrix. Roch & Villard (1994) demonstrate fast deterministic parallel algorithms, in the complexity class NC^3, for the symbolic Jordan form of a matrix.

Let K be any field with at least n^2 elements. The problem of finding the rational Jordan form $Q \in \mathsf{K}^{n \times n}$ of $T \in \mathsf{K}^{n \times n}$ is reducible, with $O(\mathrm{MM}(n) \log n + n\mathrm{M}(n))$ operations in K, to factoring a single polynomial in $\mathsf{K}[x]$ of degree at most n, namely the minimal polynomial of T which is found from the Frobenius form of T. The necessary observation is that once we know the Frobenius form of T and the complete factorization of its minimal polynomial f_1, we can immediately determine the rational Jordan form of T.

LEMMA 7.1. *Let $g \in \mathsf{K}[x]$ be monic and irreducible of degree r, and $m \geq 1$. The rational Jordan block $J_g^{(m)} \in \mathsf{K}^{mr \times mr}$ has minimal polynomial g^m.*

Proof. If $C_g \in \mathsf{K}^{r \times r}$ is the companion matrix of g, then $\mathsf{K}[C_g]$ is a field isomorphic to $\mathsf{L} = \mathsf{K}[z]/(g)$ for an indeterminate z (see Lidl & Niederreiter 1983, Section 2.5). This isomorphism is realized by mapping $C_g \mapsto z \bmod g$, and can be extended to an isomorphism of matrices in $\mathsf{K}[C_g]^{m \times m}$ with $\mathsf{L}^{m \times m}$. Observe that $J_g^{(m)} \in \mathsf{K}[C_g]^{m \times m}$, and that its image $\mathcal{J}_g^{(m)} \in \mathsf{L}^{m \times m}$ is a Jordan block, with its eigenvalue $\gamma = z \bmod g$ on the diagonal and ones on the superdiagonal. The minimal polynomial of $\mathcal{J}_g^{(m)}$ is $(x - \gamma)^m \in \mathsf{L}[x]$, so the minimal polynomial of $J_g^{(m)}$ in $\mathsf{K}[x]$ is the multiple of $(x - \gamma)^m$ in $\mathsf{K}[x] \setminus \{0\}$ of lowest degree. This is precisely g^m. $\square$

LEMMA 7.2. *Let $f \in \mathsf{K}[x]$ be monic of degree n and $C_f \in \mathsf{K}^{n \times n}$ the companion matrix of f. If $f = g_1^{m_1} \cdots g_s^{m_s}$ for distinct monic irreducible $g_1, \ldots, g_s \in \mathsf{K}[x]$, and $m_1, \ldots, m_s > 0$, then C_f has rational Jordan form $B = \operatorname{diag}(J_{g_1}^{(m_1)}, \ldots, J_{g_s}^{(m_s)})$.*

Proof. It is well known that the minimal polynomial of a block diagonal matrix is the least common multiple of the minimal polynomials of its blocks. Thus, $\min(B; \mathsf{K}^{n \times 1}) =$

$\mathrm{lcm}(g_1^{e_1}, \ldots, g_s^{e_s}) = g_1^{e_1} \cdots g_s^{e_s} = f$, since the minimal polynomial of $J_{g_i}^{(e_i)}$ is $g_i^{m_i}$ for $1 \le i \le s$ by Lemma 7.1. Since $\deg f = n$ and f is the minimal polynomial of B, B is similar to the companion matrix C_f of f. □

THEOREM 7.3. *Suppose $T \in \mathsf{K}^{n\times n}$ has invariant factors $f_1, \ldots, f_k \in \mathsf{K}[x]$, where f_i has companion matrix $C_{f_i} \in \mathsf{K}^{d_i\times d_i}$, with $\deg f_i = d_i$, for $1 \le i \le k$. If $B_i \in \mathsf{K}^{d_i\times d_i}$ is the rational Jordan form of C_{f_i} for $1 \le i \le k$, then $Q = \mathrm{diag}(B_1, \ldots, B_k) \in \mathsf{K}^{n\times n}$ is the rational Jordan form of T.*

Proof. By Lemma 7.2 there exists a non-singular $P_i \in \mathsf{K}^{d_i\times d_i}$ such that $B_i = P_i^{-1} C_{f_i} P_i$ for $1 \le i \le k$. The matrix $P = \mathrm{diag}(P_1, \ldots, P_k) \in \mathsf{K}^{n\times n}$ satisfies $Q = P^{-1}TP$, and Q is in rational Jordan form. □

THEOREM 7.4. *Let K be any field with at least n^2 elements. Given $T \in \mathsf{K}^{n\times n}$, there is a Las Vegas reduction from the problem of finding the rational Jordan form $Q \in \mathsf{K}^{n\times n}$ of T and an invertible $P \in \mathsf{K}^{n\times n}$ such that $Q = P^{-1}TP$ to the problem of factoring a single polynomial in $\mathsf{K}[x]$ of degree at most n (namely the minimal polynomial $f_1 \in \mathsf{K}[x]$ of T). This reduction requires $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ or $O\tilde{~}(\mathrm{MM}(n))$ operations in K.*

Proof. We find the Frobenius form $S \in \mathsf{K}^{n\times n}$ of T and an invertible $U_0 \in \mathsf{K}^{n\times n}$ such that $U_0^{-1}TU_0 = S$ using the algorithm `FrobeniusForm`. We factor the first invariant factor $f_1 \in \mathsf{K}[x]$ (determined by the Frobenius form and identical to the minimal polynomial of T) and construct the rational Jordan form $Q \in \mathsf{K}^{n\times n}$ of T using Theorem 7.3. An invertible $U_1 \in \mathsf{K}^{n\times n}$ such that $U_1^{-1}QU_1 = S$ is constructed using `FrobeniusForm` (we know Q is similar to S by its construction). Then $Q = U_1U_0^{-1}TU_0U_1^{-1}$ and $P = U_0U_1^{-1}$ satisfies $Q = P^{-1}TP$. □

To find the rational Jordan form of $T \in \mathsf{K}^{n\times n}$ when K has fewer than n^2 elements we embed K into a slightly larger field F to find the Frobenius form of T, using Theorem 4.2. As in Theorem 4.2, $\mathsf{F} = \mathsf{K}[x]/(\psi)$, where $\psi \in \mathsf{K}[x]$ is monic and irreducible of degree $\lceil 2\log_q n\rceil$.

THEOREM 7.5. *Let K be a field with $q = \#\mathsf{K} < n^2$. Given $T \in \mathsf{K}^{n\times n}$, there is a Las Vegas reduction from the problem of finding the rational Jordan form $Q \in \mathsf{K}^{n\times n}$ of T, and an invertible $P \in \mathsf{F}^{n\times n}$ such that $Q = P^{-1}TP$, to factoring a single polynomial in $\mathsf{K}[x]$ of degree at most n (namely the minimal polynomial $f_1 \in \mathsf{K}[x]$ of T). This reduction requires $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot \mathrm{M}(\log_q n)\log\log_q n)$ or $O\tilde{~}(\mathrm{MM}(n))$ operations in K.*

Using the Las Vegas algorithm of Berlekamp (1970) for factoring polynomials over finite fields we obtain an algorithm for computing the rational Jordan form of a matrix, given only that matrix.

COROLLARY 7.6. *Let K be a finite field of size q and $T \in \mathsf{K}^{n\times n}$. The rational Jordan form of T can be computed by a Las Vegas algorithm with $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n) + n\log q)$ or $O\tilde{~}(\mathrm{MM}(n) + n\log q)$ operations in K if $\#\mathsf{K} \ge n^2$, and $O((\mathrm{MM}(n)\cdot \log n + n\mathrm{M}(n)) \cdot \mathrm{M}(\log_q n)\log\log_q n + n\log q)$ or $O\tilde{~}(\mathrm{MM}(n) + n\log q)$ operations in K if $\#\mathsf{K} < n^2$.*

Acknowledgement. The author wishes to thank Joachim von zur Gathen and Erich Kaltofen for their enthusiasm and help while writing this paper.

REFERENCES

A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading MA, 1974.

W. BAUR AND V. STRASSEN, *The complexity of partial derivatives*, Theor. Comp. Science, 22 (1982), pp. 317–330.

E. R. BERLEKAMP, *Factoring polynomials over large finite fields*, Math. Comp., 24 (1970), pp. 713–735.

D. BINI AND V. PAN, *Improved parallel polynomial division*, in Proc. 33rd IEEE Symp. Foundat. Comput. Sci., Pittsburgh, USA, 1992, pp. 131–136.

A. BORODIN AND I. MUNRO, *The Computational Complexity of Algebraic and numeric Problems*, American Elsevier, New York NY, 1975.

R. P. BRENT AND H. T. KUNG, *Fast algorithms for manipulating formal power series*, J. Assoc. Comput. Mach., 25 (1978), pp. 581–595.

J. BUNCH AND J. HOPCROFT, *Triangular factorization and inversion by fast matrix multiplicaton*, Math. Comp., 28 (1974), pp. 231–236.

D. CANTOR AND E. KALTOFEN, *Fast multiplication of polynomials over arbitrary algebras*, Acta Informatica, 28 (1991), pp. 693–701.

C. CHUANG AND J. DUNHAM, *Matrix extensions of the RSA algorithm*, in Proc. Crypto 1990, 1990, pp. 137–153.

D. COPPERSMITH AND S. WINOGRAD, *Matrix multiplication via arithmetic progressions*, J. Symb. Comp., 9 (1990), pp. 251–280.

A. DANILEVSKY, *The numerical solution of the secular equation*, Matem. sbornik, 44 (1937), pp. 169–171. In Russian.

W. EBERLY, *Efficient parallel independent subsets and matrix factorizations*, in Proc. 3rd IEEE Symp. Parallel and Distributed Processing, Dallas, USA, 1991, pp. 204–211.

D. FADDEEV AND V. FADDEEVA, *Computational methods of linear Algebra*, W. H. Freeman and Company, San Francisco, 1963.

G. FROBENIUS, *Über den Rang einer Matrix*, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin, (1911), pp. 54–65.

F. R. GANTMACHER, *The Theory of Matrices, Vol. I*, Chelsea Publishing Co., New York NY, 1990.

M. GIESBRECHT, *Nearly Optimal Algorithms for Canonical Matrix Forms*, PhD thesis, University of Toronto, 1993. 196 pp.

K. HOFFMAN AND R. KUNZE, *Linear Algebra*, Prentice-Hall, Englewood Cliffs, N.J., 1971.

E. KALTOFEN, *Efficient solution of sparse linear systems*, 1992. Lecture notes, Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA.

E. KALTOFEN, M. S. KRISHNAMOORTHY, AND B. D. SAUNDERS, *Fast parallel computation of Hermite and Smith forms of polynomial matrices*, SIAM J. Algebraic and Discrete Methods, 8 (1987), pp. 683–690.

———, *Parallel algorithms for matrix normal forms*, Linear Algebra and its Applications, 136 (1990), pp. 189–208.

E. KALTOFEN AND V. PAN, *Processor-efficient parallel solution of linear systems over an abstract field*, in Proc. 3rd Annual ACM Symp Parallel Algorithms and Architectures, ACM Press, 1991, pp. 180–191.

———, *Processor-efficient parallel solution of linear systems II: The general case*, in Proc. 33rd IEEE Symp. Foundations of Comp. Sci., Pittsburgh, USA, 1992, pp. 714–723.

R. KANNAN, *Polynomial-time algorithms for solving systems of linear equations over polynomials*, Theoretical Computer Science, 39 (1985), pp. 69–88.

R. KANNAN AND A. BACHEM, *Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix*, SIAM J. Comp., 8 (1979), pp. 499–507.

R. M. KARP AND V. RAMACHANDRAN, *Parallel algorithms for shared-memory machines*, in Handbook of Theoretical Computer Science, J. van Leeuwen, ed., vol. A: Algorithms and Complexity, MIT Press/Elsevier, Cambridge, Mass., USA, 1990, pp. 869–932.

W. KELLER-GEHRIG, *Fast algorithms for the characteristic polynomial*, Theor. Computer Science, 36 (1985), pp. 309–317.

R. LIDL AND H. NIEDERREITER, *Finite Fields*, vol. 20 of Encyclopedia of Mathematics and its Applications, Addison-Wesley, Reading MA, 1983.

H. LÜNEBURG, *On Rational Normal Form of Endomorphisms: a Primer to Constructive Algebra*, Wissenschaftsverlag, Mannheim, 1987.

H. J. NUSSBAUMER, *Fast polynomial transform algorithms for digital convolutions*, IEEE Trans. ASSP, 28 (1980), pp. 395–398.

P. OZELLO, *Calcul Exact Des Formes De Jordan et de Frobenius d'une Matrice*, PhD thesis, Université Scientifique Technologique et Medicale de Grenoble, 1987.

V. Pan and J. Reif, *Efficient parallel solutions of linear systems*, in Proc. 17th Annual Symp. Theory of Computing, Providence, Rhode Island, USA, 1985, pp. 143–152.

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

A. Ranum, *The general term of a recurring series*, Bull. American Math. Society, 17 (1911), pp. 457–461.

J. Roch and G. Villard, *Calcul formel et parallélisme: étude de la forme normal de Jordan*, Tech. Report 7, Institut IMAG, Laboratoire LMC, 1994.

A. Schönhage, *Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2*, Acta Informatica, 7 (1977), pp. 395–398.

A. Schönhage and V. Strassen, *Schnelle Multiplikation großer Zahlen*, Computing, 7 (1971), pp. 281–292.

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

V. Shoup, *Fast construction of irreducible polynomials over finite fields*, in Proc. 4th Annual ACM-SIAM Symp. on Discrete Algorithms, Austin, USA, 1993, pp. 484–492.

B. L. van der Waerden, *Algebra*, vol. 1, Frederick Ungar Publishing Co., New York, 7th ed., 1970.

G. Villard, *Calcul formel et parallélisme: étude de la forme normal de Smith*, Tech. Report 8, Institut IMAG, Laboratoire LMC, 1994.

J. von zur Gathen, *Parallel arithmetic computations: a survey*, in Proc. 12th Int. Symp. Math. Foundations of Computer Science, vol. 233 of Springer Lecture Notes in Computer Science, Bratislava, 1986, pp. 93–112.

———, *Parallel linear algebra*, in Synthesis of Parallel Algorithms, J. Reif, ed., Morgan Kaufmann, San Mateo, USA, 1993, pp. 573–617.

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

Y. Zalcstein and M. Garzon, *An NC^2 algorithm for testing similarity of matrices*, Information Processing Letters, (1987), pp. 253–254.