# NEARLY OPTIMAL ALGORITHMS FOR CANONICAL MATRIX FORMS

by

MARK WILLIAM GIESBRECHT

A thesis submitted in conformity with the requirements

for the Degree of Doctor of Philosophy,

Graduate Department of Computer Science,

in the University of Toronto

**Nearly Optimal Algorithms For Canonical Matrix Forms**

**Mark William Giesbrecht**

Doctor of Philosophy, 1993

Department of Computer Science, Univerity of Toronto

## Abstract

Computing canonical forms of matrices is a classical mathematical problem with many practical applications. We investigate the problem of computing the Frobenius canonical form and the more familiar Jordan canonical form of a matrix over any field. A probabilistic algorithm is exhibited which computes the Frobenius form of a matrix in about the same number of operations asymptotically as is required for matrix multiplication. It is also demonstrated that determining the Frobenius form is at least as hard as matrix multiplication, whence our algorithm is *nearly optimal.* Finding the Jordan form of a matrix is reduced to finding the Frobenius form and factoring the characteristic polynomial of that matrix.

We adapt these algorithms for computing canonical forms of matrices over general fields to practical algorithms for real and rational matrices. Forms such as these, which capture all the geometric information in a matrix, are numerically highly unstable, and lend themselves to exact (as opposed to constant precision floating point) arithmetic. We take this approach and demonstrate efficient probabilistic algorithms to find these canonical forms exactly.

Applications of our techniques to important problems such as matrix powering, evaluating a polynomial at a matrix, determining matrix similarity and finding special purpose bases for finite fields are also explored. Many of our algorithms match corresponding lower bounds for these problems, which we also demonstrate.

Finally, we consider the problems of decomposing finite associative algebras and factoring in non-commutative polynomial rings. We use our probabilistic algorithms for computing Frobenius forms of matrices, as well as new lower bounds on the density of elements with certain properties in any finite associative algebra, to obtain fast new algorithms for these difficult problems.

## Acknowledgement

I would like to thank my supervisor, Joachim von zur Gathen for continued support, ideas and interest over the past years. I would also like to thank Victor Shoup and Wayne Eberly for their contributions and for listening constructively to a lot of half-baked ideas, and Erich Kaltofen, who provided much inspiration and enthusiasm. As well, I'd like thank my other committee members, Allan Borodin, Christina Christara, Steve Cook, Charles Dyer and Charlie Rackoff, for their time and effort with my thesis and defenses.

To my office mates, Jim McInnes and Gordon Turpin, I owe many rounds at Grossman's for helping me to keep my feet on the ground. Cheers! I thank my mother for her unwaivering support and confidence, not to mention fine meals, over the last two decades of school. And I thank Tian Kou, for keeping me laughing all the way. Oh yeah, and I've got to use the word "rotisserie" again.

# Table of Contents

# Chapter 1.

# Introduction

In this thesis we present very fast probabilistic algorithms for the computation of canonical forms and decompositions in a number of algebraic domains, as well as some important applications of these algorithms. Chief among the problems we consider is that of computing the Frobenius normal form of a matrix over a field. This problem pervades nearly all sections of this thesis, either as the focus, or as a key subroutine in solutions to other problems.

A fundamental theorem of linear algebra states that any matrix $T$, with entries in a field $\mathsf{K}$, is similar to a unique matrix $S \in \mathsf{K}^{n\times n}$ in *Frobenius form*, that is,

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

and each $C_{f_i}$ is the companion matrix of a monic $f_i \in \mathsf{K}[x]$ for $1 \le i \le k$, and $f_i \mid f_{i-1}$ for $2 \le i \le k$. The companion matrix $C_g$ of a monic $g = \sum_{0\le j\le r} b_i x^i \in \mathsf{K}[x]$ has the form

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

Like the better known Jordan form, the Frobenius form is a unique representative of the set of all matrices similar to $T$. It captures completely the geometric structure of the matrix, and clearly displays such invariants as the minimal polynomial ($f_1$) and characteristic polynomial ($f_1 f_2 \cdots f_k$) of the matrix. We first present a nearly optimal solution to the problem of finding the Frobenius form of a matrix, on both sequential and parallel machines, and under a number of cost metrics. We then apply our algorithm, and the information that can be gleaned from the Frobenius form, to a number of applications.

Las Vegas type probabilistic algorithms (i.e., algorithms which have the ability to flip unbiased coins, but in no case produce erroneous output) are presented which compute the Frobenius form of a matrix. The variations on the basic algorithm address different goals: asymptotic and practical cost, sequential and parallel architectures, and efficiency on matrices over arbitrary fields in terms of field operations, and on matrices over the rational numbers in terms of bit operations. In Sections 2.1 through 2.4 we link the cost of computing the Frobenius form to the cost of multiplying two matrices. Specifically, if $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ are sufficient to multiply two $n \times n$ matrices over a field $\mathsf{K}$, we show that $O\tilde{~}(\mathrm{MM}(n)) = \mathrm{MM}(n) \cdot (\log n)^{O(1)}$ operations in $\mathsf{K}$ suffice to compute the Frobenius form of an $n \times n$ matrix over $\mathsf{K}$. This cost nearly matches a demonstrated lower bound of $\Omega(\mathrm{MM}(n))$ operations in $\mathsf{K}$ for this problem. The complexity of matrix multiplication is very well studied, and applying the current asymptotically best algorithm of Coppersmith & Winograd (1990) we can choose $\mathrm{MM}(n) = n^{2.376}$, whereas the standard and very practical "school" method gives $\mathrm{MM}(n) = n^3$. 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 $\mathsf{K}$. Using Kaltofen & Pan's (1991, 1992) processor efficient algorithms for matrix inversion, in Section 2.5 we implement our algorithm for the Frobenius form in a processor efficient parallel manner, for which time $(\log n)^{O(1)}$ on $O\tilde{~}(\mathrm{MM}(n))$ processors is sufficient. An extended abstract containing much of this work on computing the Frobenius form appears in (Giesbrecht, 1992b).

Unfortunately, achieving nearly optimal asymptotic cost does not necessarily imply a practical sequential algorithm, even when practical algorithms are used for basic subroutines such as matrix and polynomial multiplication. In Chapter 3 we develop algorithms for computing the Frobenius form which are designed for use with standard matrix and polynomial arithmetic; they in fact require fewer operations than the nearly optimal algorithms presented in Chapter 2 when $\mathrm{MM}(n) = n^3$ and $\mathrm{M}(n) = n^2$. We also suspect that these algorithms would be considerably faster in an actual implementation. A Las Vegas type probabilistic algorithm is exhibited which requires $O(n^3)$ operations in $\mathsf{K}$ to compute the Frobenius form of a matrix in $\mathsf{K}^{n\times n}$ using standard matrix and polynomial arithmetic. This algorithm relies on probabilistic techniques very much akin to those used by Wiedemann (1986) for solving sparse linear systems over finite fields, and also on a lower bound for the Euler totient function as generalized to polynomials over finite fields.

Much past practical work on the problem of computing Frobenius and other canonical forms of matrices has focused on real, rational, and integer matrices. In Chapter 4 we give an algorithm to compute the Frobenius form of an integer matrix. We also show how this can be used to find the Frobenius form of a rational or real matrix. For problems over $\mathbb{Z}$ we determine the cost in terms of bit operations, where the cost of basic arithmetic operations $(+, -, \times, /)$ is polynomial in the bit length of the operands. Given a matrix $T \in \mathbb{Z}^{n\times n}$, we present a Las Vegas type probabilistic algorithm to compute exactly the Frobenius form of $T$ which requires an expected number of $O\tilde{}(n^5 \log^2 \alpha)$ of bit operations, where $\alpha$ is greater than the absolute value of every entry in $T$. This cost assumes standard algorithms for matrix, polynomial, and integer arithmetic are used. Our algorithm improves substantially on the best previously known algorithms, by Ozello (1987) and Lüneburg (1987), both of which require $O\tilde{}(n^8 \log^2 \alpha)$ bit operations. Ozello's algorithm comes with the standard library of the Maple V computer algebra system. If we allow ourselves an exponentially small possibility of erroneous output (this type of probabilistic algorithm is generally referred to as a Monte Carlo type probabilistic algorithm) we can do better still, and can compute the Frobenius form of $T$ with an expected number of $O\tilde{}(n^4 \log \alpha)$ bit operations, also using standard arithmetic. Both our Las Vegas and Monte Carlo algorithms for integer matrices are modular: they compute the Frobenius form of $T$

modulo a collection of small primes and then use Chinese remaindering to recover the integer solution. The main problem is determining the number of primes required and selecting these primes correctly.

As an application of our algorithm for computing the Frobenius form, we obtain fast algorithms for a number of interesting problems. The most striking result is for the problem of evaluating a polynomial at a matrix. In Section 5.1 we show that a polynomial $g \in \mathsf{K}[x]$, of degree up to $n^2$, can be evaluated at any matrix $T \in \mathsf{K}^{n\times n}$ with $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$. More generally, for $g$ of any degree $r$ we can compute $g(T)$ with $O\tilde{}(\mathrm{MM}(n)+r)$ operations in $\mathsf{K}$. This improves upon the previously fastest algorithm of Paterson & Stockmeyer (1973) and Brent & Kung (1978), which requires $O(\mathrm{MM}(n)\sqrt{r})$ operations in $\mathsf{K}$. We also demonstrate a lower bound for evaluating a fixed non-linear polynomial at a matrix of $\Omega(\mathrm{MM}(n))$ operations in $\mathsf{K}$, so in fact our algorithm is nearly optimal. 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. These techniques extend to parallel architectures and yield processor efficient algorithms for evaluating polynomials and straight-line programs at matrices.

As noted above, the first block in the Frobenius form is the companion matrix of the minimal polynomial of that matrix. Also, any two similar matrices have the same Frobenius form. Thus, by showing that the Frobenius form can be determined with the same order of cost as matrix multiplication (at least up to the use of randomness and poly-logarithmic factors in its cost), we add two problems to the catalogue of functions which are no more difficult to compute than matrix multiplication. Better still, we show in Section 5.2 that computing the minimal polynomial is at least as hard as, and hence about the same complexity as, matrix multiplication. This follows by a reduction from the problem of computing the determinant of a matrix to that of computing the minimal polynomial of a matrix, since computing the determinant is known to be of the same asymptotic complexity as matrix multiplication (Baur & Strassen, 1982; see Theorem 1.3 of this thesis).

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 5.3 we introduce the rational Jordan form as a slight generalization of the usual Jordan form, which 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 $\mathsf{K}$ can be accomplished by a Las Vegas algorithm with an expected number of $O\tilde{~}(\mathrm{MM}(n))$ operations in $\mathsf{K}$, given the complete factorization of the minimal polynomial of that matrix. There is a corresponding lower bound of $\Omega(\mathrm{MM}(n))$ operations in $\mathsf{K}$ for the problem of computing the Jordan form. The requirement of a complete factorization of the minimal polynomial is also necessary, since the complete factorization of any polynomial can be read off the rational Jordan form of the companion matrix of that polynomial.

Of possibly more general interest is the problem of computing the rational Jordan form of a real, rational, or integer matrix. We consider only integer matrices, as we did for the Frobenius form. Let $T \in \mathbb{Z}^{n\times n}$, where each entry has absolute value at most $\alpha$. In Section 5.3 we exhibit a Las Vegas algorithm which, given a complete factorization of the minimal polynomial of $T$, finds the rational Jordan form $Q \in \mathbb{Z}^{n\times n}$ of that matrix with an expected number of $O\tilde{~}(n^5 \log^2 \alpha)$ bit operations, using standard integer, polynomial and matrix arithmetic. Our algorithms improve on the previously best known algorithm of Gil (1992), which requires $O\tilde{~}(n^{11} \log^2 \alpha)$ bit operations for this same problem.

A common way of implementing computation in a finite field, either in silicon or algorithmically, is by means of a normal basis over a smaller field (often a prime field). If $\mathsf{K}$ is a field with $q$ elements ($q$ is necessarily a prime power), and $\mathsf{F}$ is an algebraic extension field of degree $n$ over $\mathsf{K}$, then it was first shown by Hensel (1888) that there always exists a so called normal element $\alpha \in \mathsf{F}$ such that $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ form a (normal) basis for $\mathsf{F}/\mathsf{K}$. Such a basis has the advantage that the $q$th power of an element is simply a cyclic shift of the coordinates of that element with respect to the basis, and can be computed essentially for free. This property has been exploited in many implementations of finite field arithmetic, especially in applications such as cryptography where high powers of elements often need to be computed. For this reason a great deal of attention has recently been paid to the problem of finding normal elements. Normal bases of $\mathsf{F}/\mathsf{K}$ are intimately related to the Frobenius form of the so called Frobenius map, which maps each element of $\mathsf{F}$ to its $q$th power. Using our Frobenius form algorithm, we show in Section 6.1 how to construct such a normal element using $O\tilde{~}(\mathrm{MM}(n))$ operations in $\mathsf{K}$. However, for this problem it is actually possible to avoid the use of linear algebra entirely. We present what is currently the fastest known algorithm for finding a normal element in $\mathsf{F}/\mathsf{K}$, which requires an expected number of $O\tilde{~}(n^2 + n \log q)$ operations in $\mathsf{K}$. An initial version of this result, requiring an expected $O\tilde{~}(n^2 \log q)$ operations in $\mathsf{K}$, is presented in (von zur Gathen & Giesbrecht, 1990), and is subsequently improved to the current state of the art in (von zur Gathen & Shoup, 1992). The algorithm relies on a fast test for normalcy, which comes from Hensel's existence theorem, and a lower bound on the density of normal elements in any finite field. A variant of this algorithm, which works well when standard polynomial arithmetic is used, and makes use of linearly generated sequences and the Berlekamp-Massey algorithm, is also presented. We give further generalizations to finding elements of given additive order as well. Finally, we examine the much more difficult question of finding elements in $\mathsf{F}$ which are simultaneously normal over $\mathsf{K}$ and generate the multiplicative group of $\mathsf{F}$, called *primitive-normal* elements. Such elements are shown to always exist by Lenstra & Schoof (1987), and in Sections 6.2 and 6.3 we give a reduction from the problem of finding primitive-normal elements to that of finding primitive elements in $\mathsf{F}$, a problem generally thought to be not much easier than factoring integers.

Many problems in computer algebra, including factoring polynomials and modular representations of finite groups, can be phrased in terms of decomposing finite dimensional associative algebras over finite fields. These can be represented as subalgebras of $\mathsf{K}^{m\times m}$, where $\mathsf{K}$ is a finite field and $m \geq 1$. An important problem in computing with such algebras is that of finding zero divisors, and more generally, of decomposing algebras as a direct sum of simpler, canonically represented algebras. The complexity of decomposition and representation problems in finite dimensional associative algebras over finite fields is examined in depth by Rónyai (1990). He identifies the problem of finding zero divisors as one necessary step, and an algorithm is proposed to find such elements in polynomial time. In Chapter 7 we exhibit a very simple and practical Las Vegas type probabilistic algorithm for finding zero divisors in any finite dimensional associative algebra over a finite field. It relies on a rather difficult lower bound on the probability that a randomly selected element in such an algebra will have a reducible minimal polynomial, shown in Section 7.2. Applying our algorithm for fast evaluation of a polynomial at a matrix, presented in Section 5.1, we show that our algorithm has expected cost $O(\mathrm{MM}(m))$ in addition to the cost of factoring an expected number of $O(1)$ polynomials of degree at most $m$ in $\mathsf{K}[x]$.

Finally, we apply our algorithms for finding zero divisors in a finite dimensional associative algebras over a finite field to factoring in skew-polynomial rings, an interesting non-commutative generalization of usual polynomial factorization. Skew-polynomial rings have been examined since Ore (1933), and remain an active area of mathematical research. They appear in many diverse problems, such as coding theory, polynomial decomposition, and the study of semi-linear transformations. The ring of skew-polynomials over a finite field $\mathsf{F}$ form a non-commutative integral domain. While factorization in these rings is not unique, the set of complete factorizations of a polynomial has a well understood and quite intricate structure. We give an efficient algorithm for factorization in any skew-polynomial ring over a finite field. In Sections 8.1 through 8.4 we reduce the problem to that of finding zero-divisors in a small associative algebra, and apply our algorithm from Chapter 7 for this problem. In Section 8.5, we consider an application of factoring in skew-polynomial rings to the problem of functionally decomposing a class of "wild" polynomials, for which no polynomial time algorithm for functional decomposition was previously known. This work on factoring in skew-polynomial rings and decomposing additive polynomials also appears in (Giesbrecht, 1992a).

## 1.1 Model of Computation

The algorithms presented in this thesis are generally presented on two different types of computer, sequential and parallel, and their costs are measured with a number of metrics. However, they are presented in a relatively architecture-independent manner, and specified in a high-level Pascal-like language. Over the next three sections we make more formal the details of our models of computation, and cost measures, after which we leave many of the details of implementation on various models to the reader.

Solutions to most problems are presented initially in the sequential model of computation. We use a *binary Random Access Machine* (binary RAM) as defined in (Aho *et al.*, 1974), where it is simply called a RAM. This is an abstraction of a usual general purpose computer, with a read only input tape, a write only output tape, a main memory, and a program (not in main memory). We sketch the binary RAM model here; for details see (Aho *et al.* 1974, Section 1.2). Each memory location, as well as each item on the input and output tapes, can contain an arbitrary integer. A program consists of a list of instructions, which are possibly given a unique label. Instructions are executed in sequential order, except as modified by a set of control-flow instructions. An instruction which branches to a labeled instruction when a specified memory location contains a zero is a sufficient set of control-flow instructions.

Instructions are provided for reading the next value on the input tape into memory and writing a memory location onto the next location on the output tape. An indirection instruction is provided which has two addresses as parameters; the contents of the memory location pointed to by the contents of the memory location addressed by the first parameter are stored at the address specified by the second parameter. As well, instructions are provided for the basic arithmetic operations of $+$, $-$, $\cdot$ (multiplication) and $/$ (the floor of the rational quotient) between the (integral) contents of two memory locations, which place the result in a third location. Instructions are also provided to perform comparisons such as $=$, $<$, $>$, etc., between two memory locations, placing the result, a 1 if true and 0 if false, at a third memory location. Finally, an instruction is provided to terminate execution, after which the contents of the output tape is considered to be the output from the RAM program.

Time is measured in terms of instructions executed (all instructions are assumed to have unit cost) and is generally presented as a function of the number of integers in the input to the problem on the input tape. Space is measured as the maximum amount of space required by the program at any point in the computation.

Many of the problems for which we present solutions have input and output from some ring R (often a field in this thesis), and for these problems it is useful to write the cost of our algorithms in terms of operations in R. One way to consider this formally is to assume that our machine has a second set of *algebraic* memory locations, each capable of holding an element of R, as well as algebraic input and output tapes, to which we respectively read and write values of R (we refer to the usual RAM memory as the *binary memory*, imagining memory locations to contain strings of ones and zeros representing integers). In addition to the usual instructions of a binary RAM, instructions are provided for basic ring operations ($+$, $-$, $\cdot$, and division by a unit) between two algebraic memory locations and placing the result in a third algebraic memory location, as well as a zero test which places a 0 in a specified binary memory location if the ring element input is zero, and a 1 otherwise. We call such a machine an *arithmetic* RAM. The cost of an algorithm on an arithmetic RAM is measured in terms of the number of operations in R required, and this cost is usually presented in terms of the number of elements from R in the input. This model is open to potential abuse: if we could construct a representation of the elements of R (or of the subset of R required by the algorithm) in the binary memory, then we could work entirely with binary RAM operations, which we are assuming are free. We are careful not to abuse the model in this way, and when working with an arithmetic RAM, use binary RAM operations only when necessary for such supporting operations as indexing and addressing. In an actual implementation such costs would generally be dominated by the cost of the ring arithmetic.

We are also be interested in the parallel cost of our algorithms, and the model we choose to work with is the PRAM or Parallel Random-Access Machine (see Karp & Ramachandran 1990). This is one of the most general ways of connecting together a number of RAM's, ignoring for the most part questions of bus traffic, connectivity, and memory contention. It serves as a convenient abstraction, and captures many of the fundamental issues of parallel computation. A PRAM consists of a number of independent sequential RAM's, each with its own private memory as well access to a global memory through which all communication between processors is achieved. There is one input tape and one output tape, to which all processors have access. In one unit of time each processor can read one memory location (local or global), execute a single RAM operation, write a local or global memory location, or read from the input tape or write to the output tape. The sequential RAM's in a PRAM may be either of the binary type, or of the arithmetic type over a ring $\mathsf{R}$ described above. In the latter case we also assume the existence of another global shared memory and input and output tapes, each cell of which is capable of holding a value from the ring $\mathsf{R}$. We assume our PRAM is of the concurrent-read, concurrent-write priority type, where processors can read and write to a global memory location simultaneously, and the processor with the highest priority (a fixed and unique value for each processor) is successful when many processors write the same shared memory location simultaneously.

The time $t(n)$ required by a PRAM algorithm on input size $n$ is the maximum time required by any processor involved in the computation on any input of 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 especially 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 a 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.

## Probabilistic Algorithms

In this thesis one assumption we make to achieve faster algorithms is the availability of random bits or, in the case of an algebraic RAM or PRAM over a ring $\mathsf{R}$, random elements from a subset of $\mathsf{R}$. Formally we view these "probabilistic" algorithms as follows. Suppose a problem requires an input $I$ from some set of possible inputs. A *probabilistic algorithm* which solves this problem is simply a deterministic algorithm (for a binary or arithmetic RAM or PRAM) which takes as input $I$ along with an additional finite list of bits (integers 0 or 1) or elements of $\mathsf{R}$, the "random" inputs. The allowable output depends upon the type of probabilistic algorithm we are considering, and we next discuss two common variants. A *Las Vegas type algorithm* is a probabilistic algorithm as described above. On some positive constant fraction of the possible random inputs the correct answer is returned (we must be careful to specify what this means when $\mathsf{R}$ is infinite and we expect random inputs from $\mathsf{R}$; see below). On the remaining random inputs, "failure" is returned. An incorrect output is never given. A less demanding variety of probabilistic algorithm is the Monte Carlo type. This is a probabilistic algorithm as defined above, and, like a Las Vegas type algorithm, it returns the correct answer on some positive constant fraction of the random inputs. However, on the remaining random inputs it can produce either failure or an incorrect answer. It should be noted that if a problem allows only a single possible correct output for any input then we can make the probability of error in a Monte Carlo algorithm arbitrarily small by repeated application of the algorithm with different random inputs.

When a probabilistic algorithm requires random inputs from $\mathsf{R}^j$, where $\mathsf{R}$ is an infinite ring, we must be more careful in our specification of the probability of success. We say that an algorithm is of the Las Vegas type if for *any* finite subset $L$ of $\mathsf{R}$ of size greater than some specified lower bound, on any program input, the algorithm returns the correct answer on at least a constant fraction of random inputs from $L^j$, and "failure" on other random inputs. Similarly, we say that an algorithm is of the Monte Carlo type if for *any* finite subset $L$ of $\mathsf{R}$ of size greater than some specified lower bound, on any program input, the algorithm returns the correct answer on at least a constant fraction of random inputs from $L^j$, and either "failure" or an incorrect answer on the remaining random inputs.

The cost of a probabilistic algorithm can be stated in two ways. The first and principal way that we choose is to give its deterministic running time, its probability of reporting failure, and, in the case of a Monte Carlo algorithm, its probability of returning an incorrect answer. Sometimes this is inconvenient, especially when we are using probabilistic algorithms as subroutines which are called a variable number of times. An alternative, though somewhat less informative, approach is to state the expected running time of the algorithm. In this case we state the expected number of operations required before an output other than failure is computed. For example, if a Las Vegas algorithm requires $O(t(n))$ operations to complete, and returns an answer other than failure with probability at least $p$, then the algorithm requires an expected number of $O(t(n)/p)$ operations. If the algorithm is Monte Carlo, of course we must still state the probability of an erroneous answer.

## Lower Bounds and Arithmetic Circuits

Often it is convenient to give lower bounds for algorithms in the straight-line program or arithmetic circuit model of computation. This model also gives a very natural computational representation of rational functions, nicely formalizing and generalizing such notions as sparseness and ease of computation. We refer the reader to (von zur Gathen, 1986) for a careful definition and survey of results in algebraic complexity theory, many of them for the arithmetic circuit model of computation.

A *straight-line program* or *arithmetic circuit* $\wp$ over a field $\mathsf{K}$ is defined as follows. The straight-line program $\wp$ has inputs $x_1, \ldots, x_k$, specified as indeterminates over $\mathsf{K}$. It consists of a finite list $s_1, s_2, \ldots, s_m$ of statements. Statement $i$ (where $1 \leq i \leq m$) is 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 $1 \leq j < i$, or is a constant in $\mathsf{K}$, or is one of the inputs $x_1, \ldots, x_k$. For the output, we specify a list $(i_1, \ldots, i_l) \in \{1, \ldots, m\}^l$ of some length $l \geq 1$, which designates a list of variables $(a_{i_1}, \ldots, a_{i_l})$ to output.

The semantics we assign to our arithmetic circuits is the function semantics of von zur Gathen (1986). Let $\mathsf{R}$ be any ring extension of $\mathsf{K}$ and $\infty$ the symbol for "undefined". Intuitively, we want a straight-line program to compute a function from a subset of $\mathsf{R}^k$ to $\mathsf{R}^l$. More formally, the semantics assigns to each constant, input, and variable in $\wp$ a rational function in $\mathsf{K}(x_1, \ldots, x_k) \cup \{\infty\}$, and to the entire straight-line program $\wp$ either a list of rational functions in $\mathsf{K}(x_1, \ldots, x_k)$, or $\infty$. For constants and inputs this assignment is straightforward: a constant $c \in \mathsf{K}$ in the program is assigned the function $\phi(c) = c \in \mathsf{K}(x_1, \ldots, x_k)$, while an input $x_i$ $(1 \leq i \leq k)$ is assigned the function $\phi(x_i) = x_i \in \mathsf{K}(x_1, \ldots, x_k)$. To each variable $a_i$ in the program $\wp$, which is assigned only at step $s_i$ as $a_i = b_i \operatorname{op}_i c_i$ (where $\operatorname{op}_i \in \{+, -, \times, /\}$), we assign $\phi(a_i) = \phi(b_i) \operatorname{op}_i \phi(c_i) \in \mathsf{K}(x_1, \ldots, x_k) \cup \{\infty\}$, where $\phi(a_i) \operatorname{op}_i \phi(b_i) = \infty$ if $\phi(a_i) = \infty$ or $\phi(b_i) = 0$, or if $\operatorname{op}_i = /$ and $\phi(b_i) = 0$. To $\wp$ we assign

$$\phi(\wp) = \begin{cases} (\phi(a_{i_1}), \ldots, \phi(a_{i_l})) & \text{if } \phi(a_i) \neq \infty \text{ for } 1 \leq i \leq m, \\ \infty & \text{otherwise.} \end{cases}$$

Thus, an arithmetic circuit $\wp$ such that $\phi(\wp) \neq \infty$ can be thought of as computing a sequence of rational functions in $\mathsf{K}(x_1, \ldots, x_k)$ at all points in $\mathsf{R}$ except those in a set $Z \subset \mathsf{R}^k$ where the circuit attempts to divide by a non-unit in $\mathsf{R}$. If $\mathsf{R}$ is a field, as is assumed for many lower bounds, then the points in $Z$ are the roots of some polynomial $h \in \mathsf{K}[x_1, \ldots, x_k]$, i.e., $Z$ is an algebraic set.

The sequential cost or *size* of $\wp$ is defined as its length $m$. To define its parallel cost, we note that $\wp$ specifies a directed acyclic graph, where the nodes are the program statements $s_1, \ldots, s_m$, along with the nodes for the inputs $x_1, \ldots, x_k$ and for any constants used in the program. There is an edge from node $s_i$ to $s_j$ if $a_i$ is used in $s_j$, or from an input or a constant node to $s_j$ if that input or constant is referenced in $s_j$. We assume for each output node that there is either a path to it from at least one input node, or that output node is simply an assignment of a variable to a constant. The parallel cost of the arithmetic circuit is defined as the length of the longest such path, or equivalently as the depth of $\wp$.

A very striking reduction from matrix multiplication to computing the determinant for arithmetic circuits is given by Baur & Strassen (1982) and Strassen (1973):

**Fact 1.1.** *Let* $\mathsf{K}$ *be an infinite field. Assume for each* $n \in \mathbb{N}$ *there is an arithmetic circuit over* $\mathsf{K}$*, with size* $t(n)$*, which computes the determinant of a matrix in* $\mathsf{K}^{n\times n}$ *at all matrices except perhaps those in some non-trivial algebraic set. Then for each* $n \geq 1$ *there exists an arithmetic circuit (without division), with size* $O(t(n))$*, which multiplies* any *two matrices in* $\mathsf{K}^{n\times n}$.

**Proof.** Baur & Strassen (1982) show that given an arithmetic circuit of size $t(n)$ which computes the determinant of any matrix in $\mathsf{K}^{n\times n}$ except perhaps those in some non-trivial algebraic set, then there is an arithmetic circuit of size $O(t(n))$ which computes the inverse of all matrices in $\mathsf{K}^{n\times n}$ except perhaps those in some non-trivial algebraic set. Strassen (1973) shows that given such a circuit to compute the inverse of a matrix, we can construct a circuit (without divisions) of size $O(t(n))$ which multiplies any two matrices in $\mathsf{K}^{n\times n}$. $\square$

In this thesis we deal mostly with algorithms for arithmetic RAM's and not arithmetic circuits. What do bounds on the size of arithmetic circuits, such as those implied by Fact 1.1, say about algorithms on a RAM? The next lemma is not as strong as we might like, given its use of randomness, but is the best we know of. Similar ideas are discussed by Kaltofen (1987, 1988). This lemma is similar in spirit to Theorem 1 of Adleman (1978), for algebraic rather than Boolean circuits. He shows that for every polynomial-time Las Vegas type algorithm on a standard Turing machine there is a family of Boolean circuits of polynomial-size computing the same function. Here we are concerned with algebraic circuits in the model discussed above, and a general result such as Adleman's is not true (there are functions computable on an arithmetic RAM which are not computable everywhere by a family of arithmetic circuits). Also, we are interested in the efficiency of the actual construction of the circuit from the program, which for any input size is linear in the maximum running time of the program on input of that size.

**Lemma 1.2.** *Let* $\mathsf{K}$ *be an algebraically closed field. Suppose we are given a Monte Carlo type probabilistic algorithm running on an arithmetic RAM over* $\mathsf{K}$ *which computes a function* $f\colon \mathsf{K}^k \to \mathsf{K}^l$*, and requires at most* $t(k)$ *operations in* $\mathsf{K}$*. Suppose also that we have a mechanism for choosing random elements, at unit cost, from a subset* $L$ *of* $\mathsf{K}$ *which has size at least* $2^{c+t(k)}$ *for some* $c > 1$*. Then we can construct an algebraic circuit which computes* $f$ *at all points except perhaps those in some non-trivial algebraic set* $V \subseteq \mathsf{K}^k$*. With probability at least* $1 - 2^{1-c}$ *this circuit is correct at all points in* $\mathsf{K}^k - V$*. For any given* $k$ *the construction of the circuit requires* $O(t(k))$ *operations in* $\mathsf{K}$ *on an arithmetic RAM over* $\mathsf{K}$*.*

**Proof.** For a fixed $k$, assume our Monte Carlo algorithm takes input from $\mathsf{K}^k$, as well as a random input from $\mathsf{K}^j$. We represent the input and random input generically by indeterminates $\bar{x} = (x_1, \ldots, x_k)$ and $\bar{y} = (y_1, \ldots, y_j)$ respectively.

We begin by showing that on all input points, except perhaps those in some algebraic set, our RAM program simply computes a straight-line program in the input. To avoid a slight problem with the fact that we can conditionally branch either on whether an algebraic memory location contains zero or on whether a binary memory location contains zero, we consider a slight modification of the original program: after each assignment to an algebraic memory location insert a (rather trivial) conditional branch, which simply branches to the next instruction if the assigned memory location is zero (and falls through the conditional branch and executes that same next instruction if the assigned memory location is non-zero).

Now consider the decision tree corresponding to the modified RAM program (see Aho *et al.* 1974, Section 1.6.4 for a definition of decision tree). For now we will consider the RAM program as computing a function of both the program inputs $\bar{x}$ and the random inputs $\bar{y}$. The corresponding decision tree is a binary tree where nodes are labelled with algebraic or binary conditional branch instructions. To each distinct path from a root node to a leaf node in the decision tree corresponds a distinct flow of control through the program. As we execute the RAM program on any input, to each decision we encounter there is a corresponding node in decision tree labelled with that branch instruction. Each node $v$ in the decision tree has two edges out, one corresponding to the conditional branch being taken and one to the branch not being taken. The daughter node of $v$ on the branch-taken edge in the decision tree corresponds to the next branch encountered in the RAM program if the branch corresponding to $v$ is taken. Similarly, the daughter node of $v$ on the branch-not-taken edge corresponds to the next branch encountered in the RAM program if the branch corresponding to $v$ is not taken. We next show all choices of inputs in $\mathsf{K}^{k+j}$ except perhaps those in a non-trivial algebraic set follow one branch of the decision tree. At each node in the decision tree we make a branch on whether or not a memory location —either algebraic or binary—is zero. Call a node in the decision tree *constant* if for all inputs from $\mathsf{K}^{k+j}$ control always flows out the same edge. Which nodes in the decision tree are constant? All inputs are algebraic (that is, from $\mathsf{K}$), and because the only communication between the binary and algebraic component of the RAM is through zero tests of algebraic memory locations, the state of the binary memory at any point in the execution is completely determined by the state—zero or non-zero —of all the algebraic variables assigned up to that point. In particular, because after each assignment to an algebraic variable we have inserted a conditional branch on whether that variable is zero, all nodes in the decision tree containing tests on binary memory are constant. Algebraic nodes in the decision tree branch on whether or not a some rational function in $\mathsf{K}^{k+j}$ is zero. We also label as constant all conditional branches for which the corresponding rational function is a constant in $\mathsf{K}$.

At each non-constant node, all points in $\mathsf{K}^{k+j}$ except perhaps those in some algebraic set take the non-zero branch. The points in $\mathsf{K}^{k+j}$ following the non-zero branch at *every* non-constant node in the decision tree are also non-roots of a polynomial $h \in \mathsf{K}[\bar{x}, \bar{y}]$ — the product of all those polynomials tested for zero along this path. Since the polynomial in $x_1, \ldots, x_k, y_1, \ldots, y_j$ computable at step $i$ of the algorithm has degree at most $2^i$,

$$\deg h \leq \sum_{1 \leq i \leq t(k)} 2^i < 2^{1+t(k)}.$$

The polynomial $h$ is non-zero since we are only considering non-constant nodes in the decision tree.

Next we show how to construct, with high probability, the straight-line program (in the inputs) which is evaluated at all input points except perhaps those in some non-trivial algebraic set $V$. Randomly choose an input point $\bar{a} = (a_1, \ldots, a_k)$ to the RAM program, and a random input $\bar{b} = (b_1, \ldots, b_j)$ used to run the program at $\bar{a}$, uniformly from $L^k$ and $L^j$ respectively. By Corollary 1 of Schwartz (1980) the probability that $h(\bar{a}, \bar{b}) \neq 0$ is at least $1 - 2^{1-c}$. When $h(\bar{a}, \bar{b}) \neq 0$ the output from the RAM program must be correct for input $\bar{a}$. To see this we note that the output from the RAM program is a rational function in $\bar{a} \in \mathsf{K}^k$ only, and not in $\bar{b}$. Thus, if the algorithm is incorrect on input $\bar{a}$ with random input $\bar{b}$, it is incorrect on input $\bar{a}$ for all random inputs $\hat{b} \in \mathsf{K}^j$ such that $h(\bar{a}, \hat{b}) \neq 0$. Thus, the algorithm is incorrect on all random inputs except those in some non-trivial algebraic set, which contradicts the fact that the algorithm is of the Monte Carlo type (and must produce a correct answer on a positive constant fraction of the random inputs chosen from $L^j$, where $L$ is any finite subset of $\mathsf{K}$ of sufficiently large size).

We then simulate the execution of the program at $(\bar{a}, \bar{b})$, and output a straight-line program of computation along the path we took. At each step where we perform a field operation, output the instruction performed in terms of previously computed variables or the inputs $x_1, \ldots, x_k$ or random inputs $y_1, \ldots, y_j$, and assign the result to a new variable. Now fix the random inputs as $y_i = b_i$ for $1 \leq i \leq j$ — this set of random inputs do not cause a division by zero, and assuming that $h(\bar{a}, \bar{b}) \neq 0$ (and hence the program outputs correctly for input $\bar{a}$), $\bar{b}$ is a lucky random input for all inputs along this branch. The branching program produced is as specified by the theorem. $\Box$

We can use Lemma 1.2 to obtain a version of Fact 1.1 for RAM's:

**Theorem 1.3.** *Let $\mathsf{K}$ be an infinite field, $n \geq 1$ $t: \mathbb{R}_{>0} \to \mathbb{R}_{>0}$, and assume we have a way of choosing points randomly and uniformly from a subset $L$ of $\mathsf{K}$ with size at least $2^{c+t(n^2)}$ for some $c > 1$. If there exists a Monte Carlo algorithm which, for any $T \in \mathsf{K}^{n\times n}$ computes $\det(T)$ with at most $t(n)$ operations in $\mathsf{K}$, then there exists a Monte Carlo algorithm which, for any $A, B \in \mathsf{K}^{n\times n}$ computes $AB \in \mathsf{K}^{n\times n}$ with $O(t(n))$ operations in $\mathsf{K}$.*

**Proof.** From the algorithm for computing the determinant of a matrix, find the corresponding arithmetic circuit to compute the determinant of an $n \times n$ matrix using Lemma 1.2. This circuit is correct with probability at least $1 - 2^{1-c}$. Now use the algorithm of Baur & Strassen (1982), Corollary 5, to transform the circuit for computing the determinant to a circuit to compute the inverse of a matrix, at all points except perhaps those on some non-trivial algebraic set. This construction requires $O(t(n))$ operations in $\mathsf{K}$.

Strassen (1973), Section 3, Satz 4, demonstrates how to use a circuit to invert an $n \times n$ matrix to build a circuit which multiplies any two matrices, except perhaps those in some non-trivial algebraic set. This new circuit has size $O(t(n))$ and the construction requires time linear in the circuit size. We can remove all divisions from this circuit for multiplying matrices using the technique described by Strassen (1973), Section 1, Satz 5, yielding a circuit of size $O(t(n))$ without divisions, which multiplies any two matrices in $\mathsf{K}^{n\times n}$. Strassen's removal of divisions algorithm also requires time linear in the circuit size. This circuit for matrix multiplication can be evaluated on an algebraic RAM with $O(t(n))$ operations in $\mathsf{K}$. $\Box$

## 1.2 Complexity of Fundamental Algorithms

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. A main reason for this is that the cost of these problems is not a closed book. The upper bounds do not yet match the lower bounds asymptotically, and recent progress has been made on both problems. Also, the time required asymptotically does not entirely capture the practical cost of solving these problems. The ability to substitute an asymptotically slower algorithm whose (hidden) constant may be dramatically smaller allows us to more easily determine the practical efficiency of our algorithms.

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

Over any ring $\mathsf{R}$, we assume that $O(\mathrm{M}_\mathsf{R}(n))$ operations in $\mathsf{R}$ are sufficient to multiply two polynomials in $\mathsf{R}[x]$ of degree at most $n$. Using the practical "school" method we can choose $\mathrm{M}_\mathsf{R}(n) = n^2$, or, using the algorithms of Schönhage & Strassen (1971) and Schönhage (1977) when $\mathsf{R}$ is a field, or the algorithm of Cantor & Kaltofen (1991) for any ring $\mathsf{R}$, we can choose $\mathrm{M}_\mathsf{R}(n) = n \log n \log\log n$. If $\mathsf{K}$ is a field, and $f, g \in \mathsf{K}[x]$ have degree at most $n$, then 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 $O(\mathrm{M}_\mathsf{R}(n))$ operations in $\mathsf{K}$. We can compute $\gcd(f, g)$ with $O(\mathrm{M}_\mathsf{R}(n) \log n)$ operations in $\mathsf{K}$, or, if $\mathrm{M}_\mathsf{R}(n) = \Omega(n^{1+\epsilon})$ for some constant $\epsilon > 0$, with $O(\mathrm{M}_\mathsf{R}(n))$ operations in $\mathsf{K}$ (see Aho *et al.* 1974, Section 8.9). In the arithmetic PRAM model, we can compute the product of two polynomials in $\mathsf{R}[x]$ of degree at most $n$ in time $O(\log n)$ on $O(n \log\log n)$ processors using Cantor & Kaltofen's (1991) algorithm (this algorithm is actually given in the arithmetic circuit model, where it is implemented on a circuit of depth $O(\log n)$ and size $O(n \log n \log\log n)$). Division with remainder of $f \in \mathsf{K}[x]$ by $g \in \mathsf{K}[x]$ as above can be computed in time $O(\log n)$ on $O(n \log n)$ processors by Bini & Pan (1992).

For any field $\mathsf{K}$, let $\mathsf{K}^{n \times n}$ be the ring of $n \times n$ matrices with entries in $\mathsf{K}$. We assume that $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ are sufficient to multiply two matrices in $\mathsf{K}^{n \times n}$.

Using the standard (and practical) matrix multiplication algorithm gives $\mathrm{MM}(n) = n^3$, while the current asymptotically best method, due to Coppersmith & Winograd (1990), yields $\mathrm{MM}(n) = n^{2.376}$. For convenience we assume throughout this thesis that $\mathrm{MM}(n) = \Omega(n^{2+\epsilon})$ for some $\epsilon > 0$. $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ are sufficient to compute the rank and determinant of an $n \times n$ matrix over a field $\mathsf{K}$, 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 $\mathsf{K}$ (which may be singular) with $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ (Bunch & Hopcroft, 1974). In the parallel PRAM model we will be concerned almost exclusively with processor efficient algorithms. Pan & Reif (1985), Theorem A.1 show that for any bilinear algorithm for matrix multiplication requiring $O(\mathrm{MM}(n))$ operations to multiply two matrices in $\mathsf{K}^{n\times n}$, there exists a (processor efficient) PRAM algorithm requiring time $O(\log n)$ and work $O\tilde{}(\mathrm{MM}(n))$. Kaltofen & Pan (1991, 1992) describe processor efficient Las Vegas type algorithms for computing the determinant of a matrix, inverting a non-singular matrix, finding the rank of a matrix, and solving a system of $n$ linear equations in $n$ unknowns. We will discuss these algorithms in more detail when the need arises.

## 1.3 Computation over the Integers

We will consider the computation of a number of functions from $\mathbb{Z}^m \to \mathbb{Z}^m$ for some $m \geq 1$. While it is tempting to simply use the binary RAM model discussed above, this potentially allows us to perform arithmetic operations in unit time with integers whose bits cannot even be listed in constant time. Instead we employ an arithmetic RAM over the ring $\mathsf{R} = \mathbb{Z}$, supplemented with instructions for the comparison of integers stored in algebraic memory (a test for $\leq$ is sufficient). This has the effect of introducing two types of integers, those in the binary memory and those in the algebraic memory. The integers in the binary memory can be manipulated at unit cost, regardless of their size. For operations with the integers stored in the algebraic memory, which we will refer to as *long integers*, we charge cost proportional to the length in bits of the integers involved (see below). The reader is encouraged to think of the integers in the binary part of the RAM (and stored in the binary memory) as ones which can be stored in a single word of an actual machine. These are used for addressing and indexing, etc., and on an actual machine operations involving such integers generally require a single clock cycle. The long integers can be thought of as an abstract data type for potentially very large integers requiring multiple words of storage on a real computer. In the algorithms we consider for computing functions of integers, the cost of long integer arithmetic will be dominant, but it simplifies many analyses to count only the cost of long integer operations.

The currency in which we measure the cost of long integer arithmetic in our algorithms (on an algebraic RAM over $\mathbb{Z}$) is the *bit operation*, where a single bit operation reflects approximately the cost of computing a logical operation such as "and", "or" or "not" of single bit registers. We attempt to make the bit cost of a long integer operation on a RAM reflect the cost of executing this instruction on an actual machine. To this end, we assign the cost of long integer arithmetic as the number of bit operations required by a straight line program in the so called *bitwise computation model* of Aho *et al.* (1974), Section 1.4, to perform that operation. Most of the instructions on the RAM are thus assumed to require a number of bit operations linear in the length (in bits) of the operands. The two exceptions to this are long integer multiplication and division. We assign the function $\mathrm{M}_{\mathbb{Z}}(n) : \mathbb{N} \to \mathbb{N}$ to be the cost of long integer multiplication, where $O(\mathrm{M}_{\mathbb{Z}}(n))$ bit operations are sufficient for the RAM to multiply two long integers of absolute value less than $2^n$. In the bitwise computation model, we can set $\mathrm{M}_{\mathbb{Z}}(n) = n^2$ using the school method, while the best known algorithm of Schönhage & Strassen (1971) allows us to set $\mathrm{M}_{\mathbb{Z}}(n) = n \log n \log\log n$. We abuse notation here by our re-use of the function $\mathrm{M}_{\mathbb{Z}}$ (compare with $\mathrm{M}_{\mathsf{R}}$, the cost of multiplying polynomials over an arbitrary ring $\mathsf{R}$), but recall that the complexity of integer multiplication in bit operations is essentially the same as the complexity of polynomial multiplication in terms of field operations. From this point on we simply use the function $\mathrm{M} : \mathbb{N} \to \mathbb{N}$ to represent the cost of multiplying polynomials over a ring *and* the cost of multiplying integers. Let $a, b \in \mathbb{Z}$ be stored in two memory locations, with $|a|, |b| \le 2^n$.

**Operations executable with $O(n)$ bit operations:**

(i) read from the input tape or write to the output tape an integer of absolute value less than $2^n$;

(ii) perform the operations $a+b$, and $a-b$ as well as make comparisons such as $a \le b$;

(iii) load the contents of the memory location addressed by $a$ into a designated memory location (i.e., indirect addressing).

**Operations executable with $O(\mathrm{M}(n))$ bit operations:**

(i) perform the operations $a \cdot b$ or $\lfloor a/b \rfloor$ if $b \neq 0$.

We also note that under this cost model, we can compute $\gcd(a,b)$ with $O(\mathrm{M}(n)\log n)$ bit operations, or, if $\mathrm{M}(n) = \Omega(n^{1+\epsilon})$ for some $\epsilon > 0$, then we can compute $\gcd(a,b)$ with $O(\mathrm{M}(n))$ bit operations (see Aho *et al.* 1974, Section 8.8).

The whole approach of separating out regular or short integer arithmetic from long integer arithmetic is fraught with dangers. It is would be trivial to perform all of the long integer arithmetic in the binary RAM, where operations are all unit cost. This is clearly not our intent, but it is something to be aware of. Used honestly, we feel that this model best captures the actual complexity of long integer computation with the least amount of overhead when analyzing algorithms.

It is useful, when working with more complicated structures of integers, such as polynomials, vectors, and matrices, to introduce a size function or norm. We choose the infinity norm, the maximum absolute value of any integer in the structure. Formally, let $A \in \mathbb{Z}^{n\times m}$ with $(i,j)$th entry $a_{ij} \in \mathbb{Z}$, for $1 \le i \le n$ and $1 \le j \le m$, and $f = \sum_{0\le i\le n} b_i x^i \in \mathbb{Z}[x]$. Let

(i) $\|T\|_\infty = \max\{|a_{ij}| : 1 \le i \le n, 1 \le j \le m\}$,

(ii) $\|f\|_\infty = \max\{|b_i| : 0 \le i \le n\}$.

A number of inequalities relating the norms of sums and products of matrices will be useful in the sequel. The proof of the following is trivial.

**Theorem 1.4.** *Let $A \in \mathbb{Z}^{n\times m}$, $B \in \mathbb{Z}^{r\times s}$. Then*

*(i) $\|A+B\|_\infty \le \|A\|_\infty + \|B\|_\infty$ if $n=r$ and $m=s$,*

*(ii) $\|AB\|_\infty \le m \cdot \|A\|_\infty \cdot \|B\|_\infty$ if $m=r$.*

Inequalities relating the norms of sums, products, and quotients and remainders of polynomials will also be useful.

**Theorem 1.5.** *Let $f, g \in \mathbb{Z}[x]$ have degrees $n$ and $r$, respectively.*

*(i)* $\|f+g\|_\infty \le \|f\|_\infty + \|g\|_\infty$,

*(ii)* $\|fg\|_\infty \le (1+\min(r,n)) \cdot \|f\|_\infty \cdot \|g\|_\infty$,

*(iii) if $n \ge r$, $g$ is monic and non-zero, and $f = hg + t$ for some $h, t \in \mathbb{Z}[x]$ with* $\deg t \le r$*, then* $\|h\|_\infty \le 2^{n-r} \cdot \|f\|_\infty \cdot \|g\|_\infty^{n-r}$ *and* $\|t\|_\infty \le 2^{n-r+1} \cdot \|f\|_\infty \cdot \|g\|_\infty^{n-r+1}$.

**Proof.** Part (i) and (ii) are trivial. Part (iii) requires that we examine the division algorithm for $\mathbb{Z}[x]$ (see van der Waerden 1970, Section 3.4). Define $f_n = f$ and $f_{n-i} = \bar{a}_{n-i} x^{n-r-i} g + f_{n-i-1}$ for $0 \le i \le n-r$, where $\bar{a}_i$ is the coefficient of $x^{n-i}$ in $f_{n-i}$. In particular, $f_{n-i}$ has degree at most $n-i$ and $f = hg + t$ when $t = f_{r-1}$ and $h = \sum_{0 \le i \le n-r} \bar{a}_{n-i} x^{n-r-i}$. At each step $i$, for $0 \le i \le n-r$, we have $\|f_{n-i-1}\|_\infty \le \|f_{n-i}\|_\infty \cdot (1 + \|g\|_\infty)$, yielding

$$\|h\|_\infty \le \max\{\|f_n\|_\infty, \ldots, \|f_r\|_\infty\} \le \|f\|_\infty (1+\|g\|_\infty)^{n-r} \le 2^{n-r} \cdot \|f\|_\infty \cdot \|g\|_\infty^{n-r},$$

while

$$\|t\|_\infty = \|f_{r-1}\|_\infty \le \|f\|_\infty (1+\|g\|_\infty)^{n-r+1} \le 2^{n-r+1} \cdot \|f\|_\infty \cdot \|g\|_\infty^{n-r+1}.$$

□

## The Cost of Operations over the Integers

When doing analyses of computations over the integers, we would like our analyses to capture two somewhat different, yet very related notions of the cost of our algorithms. The first is the asymptotic bit cost. Here we desire the smallest running time as a function of the input size, ignoring a (possibly very large) constant factor. As discussed earlier, the costs of integer and polynomial multiplication and division are abstracted out by the function M, while the cost of matrix arithmetic is abstracted out by the function MM. In this mode of analysis we are not directly interested in the actual running time of our algorithm on a real machine, though it is certainly a sincere hope that this is captured at some level in this analysis. The second way we examine an algorithm is for its "practical" cost, that is, its cost assuming that standard integer, polynomial and matrix arithmetic are used (i.e., $\mathrm{M}(n) = n^2$ and $\mathrm{MM}(n) = n^3$). Clearly, this approach is open to abuse: we still make use of big-Oh notation and there is nothing stopping us from simply cheating and implementing faster algorithms with enormous hidden constants. We endeavour to avoid this.

## Modular Computation of Integer Functions

One of the techniques we use to obtain practical algorithms for problems over the integers is to compute the desired result modulo a set of small primes and reconstruct the answer by Chinese remaindering. We now analyse some of the costs and benefits of this technique. Suppose we want to compute a function $\Psi: \mathbb{Z}^m \to \mathbb{Z}^m$ for some $m > 0$, and for some set $\Lambda \subseteq \mathbb{N}$ of primes we can compute $\Psi(x) \bmod p$. If the product of the primes in $\Lambda$ its greater than $2 \cdot \|\Psi(x)\|_\infty$, then we can reconstruct $\Psi(x)$ uniquely from its residues modulo the primes in $\Lambda$. Unfortunately, we will see that not all primes can be used in some algorithms, and we will spend considerable effort identifying "good" primes $p$ for which we can efficiently and correctly evaluate the desired function modulo $p$. Typically we work with primes $p$ between $2^{\ell-1}$ and $2^\ell$, where $\ell > 0$ is a "word size", the length of integers we are comfortable performing basic operations on. On a real machine we would hope that $\ell$ is less than the length of a register on which arithmetic operations can be performed in a single machine cycle. Later in this section we give a simple lemma bounding the number of primes $p$ between $2^{\ell-1}$ and $2^\ell$.

We now summarize the cost of modular reduction and Chinese remaindering, both asymptotically and using standard arithmetic.

**Lemma 1.6.** *Let $x \in \mathbb{Z}^m$ with $\|x\|_\infty \le \alpha$. Also, let $p_1, \ldots, p_r \in \mathbb{N}$ be primes, with $2^{\ell-1} < p_i < 2^\ell$ for $1 \le i \le r$ and $p_1 \cdots p_r > 2\alpha$. Then we can compute $x \bmod p_i$ for all $1 \le i \le r$ with $O(m\mathrm{M}(r\ell) \cdot \log r)$ bit operations, or $O(mr\log(\alpha)\ell)$ bit operations using standard arithmetic.*

**Proof.** The algorithm presented by Aho *et al.* (1974), Theorem 8.9, requires $O(m\mathrm{M}(r\ell) \cdot \log r)$ bit operations. For the standard arithmetic case we can make the cost sensitive to the size of $\alpha$, which proves highly desirable. To reduce a single integer $y$, with $|y| \le \alpha$, modulo a prime $p$ between $2^{\ell-1}$ and $2^\ell$, we write $y$ in base $2^\ell$: it has $s = O(\log(|y|)/\ell)$ "digits", say $y = \sum_{0\le i<s} y_i 2^{i\ell}$ where $0 \le y_i < 2^\ell$ for $0 \le i < s$. Now compute $2^{i\ell} \bmod p$ for $0 \le i \le s$ and $y_i \bmod p$ for $0 \le i < s$. We can then compute $y$ modulo $p$ with $O(s)$ additional operations modulo $p$. Thus, we can compute $y \bmod p$ with $O(\log(\alpha) \cdot \mathrm{M}(\ell)/\ell)$ bit operations. Hence, to compute $x \bmod p$ by this method requires $O(mr\log(\alpha)\ell)$ bit operations using standard integer arithmetic. $\square$

The cost of recovering $\Psi(x)$ by Chinese remaindering is also well understood.

**Lemma 1.7.** *Let $y \in \mathbb{Z}^m$. Suppose we are given primes $p_1, \ldots, p_r$ satisfying $2^\ell < p_i < 2^\ell$ for $1 \le i \le r$, and with product greater than $2\|y\|_\infty$. If we are also given $y \bmod p_i$ for $1 \le i \le r$, then we can recover $y \in \mathbb{Z}^m$ with $O(m\mathrm{M}(r\ell)\cdot\log r)$ bit operations, or, using standard arithmetic, $O(mr^2\ell^2)$ bit operations.*

**Proof.** The algorithm presented by Aho *et al.* (1974), Theorem 8.12, requires $O(\mathrm{M}(r\ell)\log r)$ bit operations. By Lipson (1981), Section 8.2.2, recovering a single integer modulo a collection of $r$ primes requires $O(r^2\ell^2)$ bit operations using standard integer arithmetic. $\square$

Of course, all this assumes that we can indeed find a collection of $r$ primes $p$ such that we can efficiently and correctly compute $\Psi(x) \bmod p$. While this is largely dependent upon the problem we our solving, we present here a useful inequality bounding from below the number of primes $p$ with $2^{\ell-1} < p < 2^\ell$.

**Theorem 1.8.** *Let $\delta \ge 3$ and $\ell = 6 + \log\log\delta$. Then there exist at least $2\lceil\lceil\log_2(2\delta)\rceil/(\ell-1)\rceil$ primes $p$ such that $2^{\ell-1} < p < 2^\ell$.*

**Proof.** Let $\pi(x)$ be the number of primes greater than zero and at most $x$. Rosser & Schoenfeld (1962), (3.8), show that $\pi(2x) - \pi(x) \ge 3x/(5\log x)$ for $x \ge 21$, whence the number of primes $p$ such that $2^{\ell-1} < p < 2^\ell$ is at least $3\cdot 2^{\ell-1}/(5\cdot\log 2^{\ell-1})$ for all $\ell \ge 6$. It is easily derived that

$$\frac{3\cdot 2^{\ell-2}}{5\cdot\log 2^{\ell-2}} \ge 2\left(1 + \frac{2+\log_2\delta}{\ell-2}\right) \ge 2\left\lceil\frac{\lceil 1+\log_2\delta\rceil}{\ell-2}\right\rceil$$

for $\delta \ge 3$. $\square$

The principal use of this theorem is as follows. If $\ell = 6 + \log\log\delta$, then there are enough primes $p$ with $2^{\ell-1} < p < 2^{\ell}$ to modularly represent integers with absolute value up to $\delta$ ($\lceil \lceil \log_2(2\delta) \rceil / (\ell - 1) \rceil$ primes are sufficient for this), even assuming a set of "bad" primes, which we do not wish to use, whose product is at most $\delta$.

The cost of actually computing a set of $2\lceil \lceil \log_2(2\delta) \rceil / (\ell - 1) \rceil$ primes $p$ between $2^{\ell-1}$ and $2^{\ell}$ is considered pre-computation, though it is easily accomplished using the sieve of Eratosthenese. This requires $O(\log\delta \log\log\log\delta)$ bit operations: see (Knuth 1981, Section 4.5.4).

# Chapter 2.

# Computing the Frobenius Form

Computing a canonical or normal form of an $n \times n$ matrix $T$ over any field $\mathsf{K}$ is a classical mathematical problem with many practical applications. Recall from the introduction 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

$$S = \operatorname{diag}(C_{f_1}, C_{f_2}, \ldots, C_{f_k}) = \begin{pmatrix} \boxed{C_{f_1}} & & & \mathbf{0} \\ & \boxed{C_{f_2}} & & \\ & & \ddots & \\ \mathbf{0} & & & \boxed{C_{f_k}} \end{pmatrix} \in \mathsf{K}^{n\times n}, \qquad (2.1)$$

where each $C_{f_i}$ is the companion matrix of some monic $f_i \in \mathsf{K}[x]$ for $1 \le i \le k$, and $f_i \mid f_{i-1}$ for $2 \le i \le k$. A matrix $S$ with these properties, called the *Frobenius form* of $T$, always exists and is unique and *rational*, i.e., it remains unique even if the entries of $T$ are allowed to lie in any field extension of $\mathsf{K}$. 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 $U^{-1}TU = S$. Actually, our approach to finding the Frobenius form will be to find such a matrix $U$, from which we find $S$. Two excellent references for all the background information are (Gantmacher 1990, Chapter 7), and (Hoffman & Kunze 1971, Chapter 7).

In Sections 2.1 through 2.4 we develop a probabilistic algorithm which requires

$O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ or $O\tilde{~}(\mathrm{MM}(n))$ operations in $\mathsf{K}$ to compute $S, U \in \mathsf{K}^{n\times n}$ as above. This algorithm is of the Las Vegas type; with probability at least 1/4 it will return the correct answer, otherwise it will report failure. An incorrect answer will never be returned. Also, it only works as stated when $\#\mathsf{K} \geq n^2$. When $\mathsf{K}$ has $q < n^2$ elements, we embed $\mathsf{K}$ in a field $\mathsf{F}$ of algebraic degree $O(\log n)$ over $\mathsf{K}$ which does possess $n^2$ elements and compute the Frobenius form of the projection of $T$ in $\mathsf{F}^{n\times n}$. The Frobenius form of $T$ lies in $\mathsf{K}^{n\times n}$ since it is rational, but the matrix $U \in \mathsf{F}^{n\times n}$ constructed by our algorithm may have entries in $\mathsf{F} \setminus \mathsf{K}$. In this case the running time of our algorithms will be multiplied by a small power of $\log_q n$. Details are presented in Section 2.4.

Ozello (1987) presents two algorithms to find the Frobenius form of a matrix, the first of which works over any field while the second works only for rational matrices. Both of these require $O(n^4)$ operations in the ground field, though the second has better performance in terms of coefficient growth for integer matrices. A similar algorithm is presented by Lüneburg (1987), which also requires $O(n^4)$ operations in the ground field. These are apparently the fastest previously known sequential algorithms for this problem (see also Howell, 1973a & 1973b, Kannan & Bachem 1979, and Kannan 1985). Kaltofen *et al.* (1986, 1990) present a probabilistic algorithm 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 $xI - T \in \mathsf{K}[x]^{n\times n}$ the Frobenius form $S$ of $T$ can be derived easily, though it is not clear how to obtain the transition matrix $U$ such that $U^{-1}TU = S$ quickly (see Kaltofen *et al.* 1987).

Historically, the concept of invariant factors for integer matrices is first explored by Smith (1861). This is refined and extended considerable, in particular to matrices of complex polynomials, by Frobenius (1878). Much of Frobenius's work principally concerns bilinear forms, and results on canonical forms of matrices follow as corollaries. Many of these theorems are presented in a matrix setting in (Frobenius, 1894), and the theory of how a vector space decomposes as a direct sum of invariant subspaces under a linear map is further developed in (Frobenius, 1911). A more geometric approach to this theory, for matrices over an arbitrary field, is given by Gantmacher (1935) (see Gantmacher 1990, Chapter 7), though it is not clear if this is the first time such an approach is considered. We adapt many elements of Gantmacher's relatively constructive proof into our algorithm.

Our algorithm for computing the Frobenius form is nearly optimal in that if we can compute the Frobenius form with at most $t(n)$ operations in $\mathsf{K}$, then we can find the characteristic polynomial $f \in \mathsf{K}[x]$ of $T$ with $O(t(n))$ operations in $\mathsf{K}$ as well ($f$ is the product of $T$'s invariant factors). The determinant of $T$ is $(-1)^{\deg f} \cdot f(0)$, and it follows from Baur & Strassen (1982) (see Theorem 1.3 of this thesis) that this implies the existence of an algorithm to multiply any two matrices in $\mathsf{K}^{n\times n}$ which requires $O(t(n))$ operations in $\mathsf{K}$.

In Section 2.5 we examine the parallel cost of computing the Frobenius form of a matrix over any field $\mathsf{K}$. In particular, we exhibit a processor efficient algorithm for finding the Frobenius form of any $T \in \mathsf{K}^{n\times n}$. This means that if the best known *sequential* algorithm for multiplying two matrices in $\mathsf{K}^{n\times n}$ needs $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$, then we can find the Frobenius form of $T$ in time $(\log n)^{O(1)}$ on $O\tilde{~}(\mathrm{MM}(n))$ processors. Our method is essentially a processor efficient reduction to the problem of matrix inversion, for which we employ Kaltofen & Pan's (1991, 1992) processor efficient algorithm. En route we give a processor efficient version of Keller-Gehrig's(1985) algorithm for finding the characteristic polynomial of a matrix.

## 2.1 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 $\mathsf{K}$-endomorphism of a finite dimensional vector space $V$ over $\mathsf{K}$ (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 $g = \sum_{0\le i\le r} b_i x^i \in \mathsf{K}[x]$ acts on $V$ as the endomorphism $\sum_{0\le i\le r} b_i T^i \in \mathrm{End}_{\mathsf{K}} V$. We write $f \circ v = f(T)v \in V$. An important rôle 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 2.1** (Cyclic Decomposition Theorem). *Let $V$ be a finite dimensional vector space over a field $\mathsf{K}$ 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 $1 \le i \le k$.*

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 $1 \le i \le k$, $f_i = \min(T; v_i) \in \mathsf{K}[x]$ is the *ith invariant factor* of $T$, for $1 \le i \le k$. 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 $1 \le i \le k$. The matrix

$$U = \left[ v_1 \,\middle|\, Tv_1 \,\middle|\, T^2 v_1 \,\middle|\, \cdots \,\middle|\, T^{d_1 - 1} v_1 \,\middle|\, v_2 \,\middle|\, Tv_2 \,\middle|\, \cdots \,\middle|\, v_k \,\middle|\, \cdots \,\middle|\, T^{d_k - 1} v_k \right] \in \mathsf{K}^{n \times n} \quad (2.2)$$

has the property that $S = U^{-1} T U$ 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 (2.2), 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 (1935).

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 $\mathsf{K}$-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 rôle 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 2.2.** *Let $V$ be any vector space over $\mathsf{K}$ 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 2.2 recursively to $V'$ yields a decomposition of $V$ as follows (see Gantmacher 1990, Section 7.4).

**Fact 2.3** (Modular Cyclic Decomposition). *For* $i \geq 1$ *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* $1 \leq i \leq k$ *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* $1 \leq i \leq k$.

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 2.3 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 2.3.

Our approach to finding $w_1, \ldots, w_k \in V$ is to choose them randomly. This will only work when $\#\mathsf{K} \geq n^2$. When $\#\mathsf{K} < n^2$, we choose a small extension field $\mathsf{F}$ of $\mathsf{K}$ such that $\#\mathsf{F} \geq n^2$. This will be discussed in Section 5. For the remainder of this section, assume that $\#\mathsf{K} \geq n^2$.

**Lemma 2.4.** *Let* $W$ *be any vector space of dimension at most* $n$ *over* $\mathsf{K}$, *spanned by* $u_1, \ldots, u_n \in W$, *and let* $T : W \to W$ *be a* $\mathsf{K}$*-linear map. Let* $L$ *be a subset of* $\mathsf{K}$ *containing at least* $n^2$ *elements. Then*

$$\underset{(a_1,\ldots,a_n)\in L^n}{\mathrm{Prob}} \left\{ \min(T; \sum_{1 \leq i \leq n} a_i u_i) = \min(T; W) \right\} \geq 1 - \frac{1}{n}.$$

**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 $1 \leq i \leq l$, let $g_i = \min(T; y_i) \in \mathsf{K}[x]$ be the $i$th invariant factor of $T$, so $g_i \,|\, g_{i-1}$ for $2 \leq i \leq l$.

We choose random elements $a_1, \ldots, a_n \in L$ and assign $v = \sum_{1 \leq i \leq n} a_i u_i \in W$. 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 $1 \leq i \leq l$. The component of $v$ in $\mathrm{Orb}(T; y_1)$ is then $v_1 = \sum_{1 \leq i \leq n} a_i h_i \circ y_1$. 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 $h = \sum_{1 \leq i \leq n} a_i h_i$, 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 $\gcd(h, g_1) \neq 1$ 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 $\gcd(\sum_{1 \leq i \leq n} a_i h_i, g_1) = 1$, consider the polynomial $\rho = \sum_{1 \leq i \leq n} x_i h_i \in \mathsf{K}[x_1, \ldots, x_n][x]$, in the indeterminates $x_1, \ldots, x_n, x$. To prove the theorem it is sufficient to show that

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

This is accomplished with the aid of resultants (see van der Waerden 1970, Section 5.8). The resultant of $\rho$ 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 $R(a_1, \ldots, a_n) \neq 0$.

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 $y_1 = \sum_{1 \leq j \leq n} b_j u_j$ for some $b_1, \ldots, b_n \in \mathsf{K}$. Then $\sum_{1 \leq j \leq n} b_j h_j \equiv 1 \bmod g_1$, and since each $h_j$ has degree less than $g_1$ by definition, $\sum_{1 \leq j \leq n} b_j h_j = 1$. Thus, $R(b_1, \ldots, b_n) \neq 0$.

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

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

□

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 2.5.** *Let $L$ be a subset of $\mathsf{K}$ containing at least $n^2$ elements. For $1\leq i\leq n$, 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 $1\leq i\leq k$ (where $k$ is the number of invariant factors of $T$) the $i$th 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 $1\leq i\leq n$, 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 $1\leq i\leq n$ 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 $i$th row and zeros in all other rows. We have chosen $w_i=\sum_{1\leq i\leq n}a_ie_i$ 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

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

Thus, by Lemma 2.4,

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

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

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

If all the $w_i$ are chosen correctly then we also know, from Fact 2.3, that $f_i = \min(T_{i-1}; w_i + V_{i-1})$ for $1 \le i \le k$. $\square$

## 2.2 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 2.3, 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 $\mathsf{K}$.

**Fact 2.6** (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 $\mathsf{K}$:*

*(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 $\mathrm{Orb}(T; u_1) + \cdots + \mathrm{Orb}(T; u_i)$).*

Applying Fact 2.6 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 2.6 with Theorem 2.5 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 $\mathsf{K}$ 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
- $\sum_{1\leq i\leq n} d_i = n$ (any of $d_1, \ldots, d_k$ may be zero),
- for $1 \leq i \leq n$, 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 2.7.** *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 $\mathsf{K}$.*

The columns of $H$ are a modular cyclic basis for $V$, and $H$ determines a change of basis on $V$ (to this modular cyclic basis), under which $T$ has the form

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

where $C_{f_i} \in \mathsf{K}^{d_i\times d_i}$ is the companion matrix of $f_i$, the $i$th 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

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

or more simply

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

where $f_i = \sum_{0\leq l\leq d_i} a_{il}x^l$, and $g_{ij} = \sum_{0\leq l<d_j} c_{ijl}x^l$ with $a_{il}, c_{ijl} \in \mathsf{K}$ for $1 \leq j < i$ and $1 \leq i \leq k$. Thus $f_i \circ w_i \in V_{i-1}$, and since it has minimal degree, is the $i$th 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 2.8.** *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 $i$th invariant factor of $T$, for $1 \le i \le k$ and $V_i$, $T_i$ as in Fact 2.3. 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 $\mathsf{K}$. Furthermore, $\sum_{1\le j<i} \deg g_{ij} \le n$ for all $1 \le i \le k$.*

Combining this theorem with Theorem 2.7 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.

## 2.3 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 2.3 for $0 \le i \le n$, and $f_i = \min(T_{i-1}; w_i + V_{i-1})$ is the $i$th 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 2.8), 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 2.9.** *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 2.9, 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 2.10.** *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

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

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 thus $g \circ v_i = 0$. Therefore $V_{i-1} \cap \mathrm{Orb}(T; v_i) = \{0\}$ 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 $\mathsf{K}$. 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 $\mathsf{K}$.

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 $\mathsf{K}$.

**Theorem 2.11.** *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 $\mathsf{K}$.*

## 2.4 Computing the Frobenius Form

The complete algorithm for computing the Frobenius normal form of any $T \in \mathsf{K}^{n\times n}$, where $\#\mathsf{K} \geq n^2$, 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 $1 \leq i \leq k$;

$H$ need not be correct — an erroneous computation 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 (2.2);

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

**End.**

**Theorem 2.12.** *Let $T \in \mathsf{K}^{n\times n}$ over any field $\mathsf{K}$ 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 $\mathsf{K}$ are sufficient.*

**Proof.** First we prove the correctness of the output. We use the algorithm `FindModCyc`, analyzed in Theorem 2.7, 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 2.9 we know $f_i \,|\, g_{ij}$ for $1 \le j < i$ and $1 \le i \le k$, and certainly $f_i \,|\, f_{i-1}$ for $2 \le i \le k$. Once the tests in step (3) are passed, Theorem 2.10 guarantees that $v_1, \ldots, v_k$ generate the cyclic composition factors of $V$. 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 2.10, $V = \mathrm{Orb}(T; v_1) \oplus \cdots \oplus \mathrm{Orb}(T; v_k)$. Let $l \ge 1$ be the smallest integer such that $f_l \ne \bar{f}_l$, where $\bar{f}_l$ is the the actual $l$th 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 $l \le j \le k$, we know $f_l = \min(T_{l-1}; \bar{V}_l) = \bar{f}_l$, a contradiction.

By Theorem 2.7, step (1) requires $O(\mathrm{MM}(n)\log n)$ operations in $\mathsf{K}$. Using Theorem 2.8, steps (2)-(4) require $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ ($H$ is singular exactly when its last column is zero). By Theorem 2.11 steps (5)-(6) require $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ to complete. The matrix $U$ in step (7) can be found using Fact 2.6 with $O(\mathrm{MM}(n)\log n)$ operations in $\mathsf{K}$. $\square$

The algorithm `FrobeniusForm` requires that the field $\mathsf{K}$ 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 $\mathsf{F}$ of $\mathsf{K}$

containing $n^2$ elements. We then run the algorithm `FrobeniusForm` on $T \in \mathsf{F}^{n\times n}$. By Theorem 2.12 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 $\mathsf{F}$, is that $U$ is not necessarily a matrix over $\mathsf{K}$.

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 $\mathsf{K}$. 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 $\mathsf{K}$ are simply represented as constant polynomials, giving a trivial embedding of $\mathsf{K}$ into $\mathsf{F}$. Additions and subtractions in $\mathsf{F}$ require $O(\log_q n)$ operations in $\mathsf{K}$, while multiplications in $\mathsf{F}$ require $O(\mathrm{M}(\log_q n))$ operations in $\mathsf{K}$ and divisions in $\mathsf{F}$ require $O(\mathrm{M}(\log_q n)\log\log_q n)$ operations in $\mathsf{K}$.

**Theorem 2.13.** *Let $T \in \mathsf{K}^{n\times n}$, where $q = \#\mathsf{K} < n^2$. There is a Las Vegas type algorithm which 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 $\mathsf{K}$, 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 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 $\mathsf{K}$.*

## 2.5 A Processor Efficient Parallel Algorithm

In this section we exhibit a processor efficient parallel algorithm for finding the Frobenius form of a matrix (see Section 1.1 for the definition of a processor efficient algorithm). 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 (Baur & Strassen, 1982; see Theorem 1.3 of this thesis). 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, used in step (2) of `FindModCyc` in Section 2.2, is the heart of his asymptotically fast version of Danilevsky'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 a processor efficient parallel version of Danilevsky's algorithm for computing the characteristic polynomial of a matrix, and a processor efficient parallel version of `FindModCyc`. 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 2.14.** *Suppose* $\mathsf{K}$ *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* $\mathsf{K}$ *sequentially (for a definition of bilinear algorithm see Borodin & Munro 1975), then there exists a processor efficient parallel algorithm to compute* $AB$ *on* $O(\mathrm{MM}(n))$ *processors in time* $O(\log n)$.

*(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, 1992). A number of processor efficient algorithms are demonstrated by Eberly (1991), including (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). □

For this section we make the assumption, based on Fact 2.14(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 $\mathsf{K}$.

Assume that we are given vectors $u_1, \ldots, u_n \in V$. As in Fact 2.6, we want to find matrices $H_i \in \mathsf{K}^{n \times d_i}$ for $1 \le i \le n$ with the following properties:

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

(ii) for $1 \le i \le n$, $H_i = [u_i \mid Tu_i \mid \cdots \mid T^{d_i - 1} u_i]$ 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 $1 \le i \le n$.

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 $i$th column of $T$ for $1 \le i \le n$. 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

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

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 2.14. 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) (see Fact 2.14(iii)), and $H^{(j)} \in \mathsf{K}^{n \times n}$ is constructed as above. This is repeated $r = \lceil \log_2 n \rceil$ times.

**Theorem 2.15.** *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 2.6 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 2.15 with Theorem 2.5 allows us to implement the algorithm `FindModCyc` in Section 2.2 in a processor efficient manner.

**Theorem 2.16.** *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 $1 \le i \le k$ defined as follows:*

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

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

*(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* $1 \le i \le k$.

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

We now turn to the parallel cost of the algorithm `FrobeniusForm` presented in Section 2.4.

**Theorem 2.17.** *The Las Vegas algorithm* `FrobeniusForm` *can be implemented in a processor efficient manner. Let* $T \in \mathsf{K}^{n\times n}$, *where* $\mathsf{K}$ *is a field with* $\#\mathsf{K} \ge n^2$.

*(i) If* $p \ge n$ *or* $p = 0$ *then* `FrobeniusForm` *requires time* $O(\log^4 n)$ *on* $O(\mathrm{MM}(n))$ *processors.*

*(ii) If* $2 \le p < n$ *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 2.4. 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 2.16. 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 $2 \le i \le k$ and $1 \le j < i$ are read from $G = H^{-1}TH$ as in Theorem 2.8, and $G$ can be computed within the desired parallel time and processor bounds by Fact 2.14. Steps (4)–(5) can be done using processor efficient polynomial division algorithms, as specified in Fact 2.14 parts (iv) and (v). Step (6) can be performed by a single matrix product as in Theorem 2.11, and so can be done in a processor efficient manner. Applying Theorem 2.16 gives us $U$ in step (7). $\square$

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

**Theorem 2.18.** *Let* $\mathsf{K}$ *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 2.13, 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 2.17 except that we work in an extension $\mathsf{F}$ of $\mathsf{K}$ of algebraic degree $\lceil \log_q n \rceil$ over $\mathsf{K}$, which does contain $O(n^2)$ elements (see Theorem 2.13). Each operations in $\mathsf{F}$ requires $O(\log\log_q n)$ operations in $\mathsf{K}$ on $O(\mathrm{M}(\log_q n))$ processors. □

Theorem 2.16 also has the following important corollary, essentially a processor efficient version of the algorithm of Keller-Gehrig (1985) for computing the characteristic polynomial of a matrix.

**Corollary 2.19.** *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* $\mathsf{K}$. *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 $H$ as in Theorem 2.15. It is shown by Keller-Gehrig (1985) that

$$G = H^{-1}TH = \begin{pmatrix} C_{g_1} & B_2 & \cdots & B_l \\ & C_{g_2} & & \\ & & \ddots & \\ \mathbf{0} & & & C_{g_l} \end{pmatrix} \in \mathsf{K}^{n\times n},$$

where $C_{g_i} \in \mathsf{K}^{d_i \times d_i}$ is the companion matrix of some $g_i \in \mathsf{K}[x]$ of degree $d_i$ (for

$1 \leq i \leq l$), and $f = g_1 \cdots g_l$. Note that $g_1, \ldots, g_l$ are not necessarily invariant factors of $f$. The inverse of $H$ and product $g_1 \cdots g_k$ can be computed within the desired time and processor bounds using the methods described in Fact 2.14. $\square$

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 $T$.

# Chapter 3.

# Computing the Frobenius Form Using Standard Arithmetic

The algorithm `FrobeniusForm` presented in Chapter 2 for computing the Frobenius form of a matrix over any field is theoretically very pleasing. It is, up to a poly-logarithmic factor in its cost and the use of randomness, a nearly optimal algorithm for a classical problem. However, using practical polynomial and matrix arithmetic a somewhat stronger result can be stated, making use of techniques of Wiedemann (1986).

When $\mathrm{MM}(n) = n^3$ and $\mathrm{M}(n) = n^2$ our algorithm for computing the Frobenius form of an $n \times n$ matrix over a field $\mathsf{K}$ of size at least $n^2$ requires $O(n^3 \log n)$ operations in $\mathsf{K}$. In Section 3 we show that this can be accomplished with $O(n^3)$ operations in $\mathsf{K}$ using a modification of the algorithm `FrobeniusForm` presented in the preceding chapter. The only change required is the replacement of the subroutine of Keller-Gehrig (1985), described in Fact 2.6. The new version requires $O(n^3)$ operations in $\mathsf{K}$, as opposed to the $O(n^3 \log n)$ operations in $\mathsf{K}$ required by Keller-Gehrig's algorithm when standard arithmetic is used.

When $q = \#\mathsf{K} < n^2$, the algorithm `FrobeniusForm` also has the unfortunate deficiency that the transition matrix $U$ produced (such that $U^{-1}TU$ is in Frobenius form) may have entries in a small extension field of $\mathsf{K}$, and not in $\mathsf{K}$ as desired. As well, it requires $O(n^3(\log_q n)^2 \log\log_q n)$ operations in $\mathsf{K}$ using standard arithmetic. The inefficiencies stem from the the routine `FindModCyc`. In Section 3 we present a modification of `FrobeniusForm` which produces a transition matrix $U \in \mathsf{K}^{n\times n}$ such that $U^{-1}TU$ is in Frobenius form, and which requires only $O(n^3 \log_q n)$ operations in $\mathsf{K}$ using standard arithmetic. Aside from using the replacement for Keller-Gehrig's algorithm presented in Section 3, the only further modification consists of a new version of the subroutine `FindModCyc` to find a modular cyclic transition matrix with high probability, which we call `FindModCycDyn`. This subroutine utilizes a new, dynamic way of choosing maximal vectors in a modular vector space, based on a lower bound on an analogue of the Euler totient function for polynomials over finite fields, which we present in Section 3.

## 3.1 Computing the Frobenius Form over a Large Field with Standard Arithmetic

In this section we consider the problem of computing the Frobenius form of a matrix $T \in \mathsf{K}^{n\times n}$ over a field $\mathsf{K}$ with at least $n^2$ elements, using standard matrix and polynomial arithmetic. That is, we assume throughout this section that $\mathrm{MM}(n) = n^3$ and $\mathrm{M}(n) = n^2$. The Las Vegas type algorithm we present requires $O(n^3)$ operations in $\mathsf{K}$, as opposed to the $O(n^3 \log n)$ operations in $\mathsf{K}$ required by the algorithm `FrobeniusForm` presented in Theorem 2.12 (when standard arithmetic is used).

The only modification required to `FrobeniusForm` is the replacement of the algorithm of Keller-Gehrig (1985) used in step (2) of `FindModCyc` (called in step (1) of `FrobeniusForm`) and in step (7) of `FrobeniusForm` itself. The new version requires $O(n^3)$ operations in $\mathsf{K}$.

Once again, let $V = \mathsf{K}^{n\times 1}$ and suppose $v_1, \ldots, v_l \in V$ for some $l \geq 1$. We say that a non-singular $R^{(l)} \in \mathsf{K}^{n\times n}$ *row reduces* $C^{(l)} = [v_1|\cdots|v_l]$ if $R^{(l)}C^{(l)} \in \mathsf{K}^{n\times l}$ is upper triangular. An additional vector $v_{l+1} \in V$ is linear dependent upon $v_1, \ldots, v_l$ if and only if $R^{(l)}v_{l+1} = (c_1, \ldots, c_l, 0, \ldots, 0)^t \in V$ for some $c_1, \ldots, c_l \in \mathsf{K}$. Moreover, since $R^{(l)}$ is invertible, we find that $v_{l+1} = \sum_{1\leq i\leq l} c_i v_i$, that is, $R^{(l)}v_{l+1}$ gives the linear dependency of $v_{l+1}$ on $v_1, \ldots, v_l$.

If $v_{l+1} \in V$ is linearly independent of $v_1, \ldots, v_l$, then we can construct a matrix $R^{(l+1)} \in \mathsf{K}^{n\times l}$ such that $R^{(l+1)}[v_1|\cdots|v_{l+1}]$ is upper triangular. This is one step in the standard Gaussian elimination procedure and requires $O(n^2)$ operations in $\mathsf{K}$.

**Theorem 3.1.** *Let $v_1, \ldots, v_l \in V$ be linearly independent and $R^{(l)} \in \mathsf{K}^{n\times n}$ such that $R^{(l)}[v_1|\cdots|v_l] \in \mathsf{K}^{n\times(l+1)}$ is upper triangular.*

*(i) If $v_{l+1} \in V$ is linearly dependent on $v_1, \ldots, v_l$, then we can determine this fact and find $c_1, \ldots, c_l$ such that $v_{l+1} = \sum_{1\le i\le l} c_i v_i$ with $O(n^2)$ operations in $\mathsf{K}$.*

*(ii) If $v_1, \ldots, v_l, v_{l+1} \in V$ are linearly independent then we can compute an invertible $R^{(l+1)} \in \mathsf{K}^{n\times n}$ such that $R^{(l+1)}[v_1|\cdots|v_{l+1}] \in \mathsf{K}^{n\times l+1}$ is upper triangular with $O(n^2)$ operations in $\mathsf{K}$.*

Using this theorem we can find the minimal polynomial of a vector in a modular vector space efficiently.

**Theorem 3.2.** *Suppose $v_1, \ldots, v_l \in V$ form a basis for some $T$-invariant subspace $W$ of $V$, and that we are supplied with an invertible $R \in \mathsf{K}^{n\times n}$ such that $R[v_1|\cdots|v_l]$ is upper triangular. Given any $w \in V$, we can find*

*(i) $g = \min(T \bmod W; w + W) \in \mathsf{K}[x]$,*

*(ii) $w, Tw, \ldots, T^{d-1}w \in V$, where $d = \deg g$,*

*(iii) an invertible $\hat{R} \in \mathsf{K}^{n\times n}$ with $\hat{R}[v_1|\cdots|v_l|w|Tw|\cdots|T^{d-1}w]$ upper triangular,*

*with $O(n^2 d)$ operations in $\mathsf{K}$.*

**Proof.** The algorithm proceeds in stages $i = 0, 1, \ldots$ At stage 0, simply check whether $v \in W$ using $R$ and Theorem 3.1(i). If $v \in W$, return $g = 1$, $\hat{R} = R$ and quit. Otherwise, compute an invertible $R^{(1)} \in \mathsf{K}^{n\times n}$ such that $R^{(1)}[v_1|\cdots|v_l|w]$ is upper triangular.

At the beginning of stage $i > 0$ we assume $v_1, \ldots, v_l, w, Tw, \ldots, T^{i-1}w$ are linearly independent, and that $R^{(i)}[v_1|\cdots|v_l|w|\cdots|T^{i-1}w]$ is upper triangular. In stage $i$, first check if $T^i w$ is linearly dependent on $v_1, \ldots, v_l, w, \ldots, T^{i-1}w$ using Theorem 3.1(i) with $R^{(i)}$. If so we have

$$T^i = \sum_{0\le j<i} T^j w + \sum_{1\le j\le l} b_j v_j$$

for some $c_0, \ldots, c_{i-1}, b_1, \ldots, b_l \in \mathsf{K}$. Therefore,

$$g = x^i - \sum_{0\le j<i} c_j x^j = \min(T \bmod W; w + W),$$

$\hat{R} = R^{(i)}$, and we are done. Otherwise, find an invertible $R^{(i+1)} \in \mathsf{K}^{n\times n}$ such that $R^{(i+1)}[v_1|\cdots|v_l|w|\cdots|T^i w]$ is upper triangular using Theorem 3.1(ii), and continue with stage $i+1$.

Each stage requires $O(n^2)$ operations in $\mathsf{K}$ and there are at most $d$ stages. $\square$

This allows us to construct a version of Keller-Gehrig's algorithm (as described in Fact 2.6) which requires $O(n^3)$ operations in $\mathsf{K}$. The resulting algorithm is very similar to Danilevsky's (1937) algorithm for computing the characteristic polynomial of a matrix.

**Algorithm:** `ModuleBasis`

Input: $T \in \mathsf{K}^{n\times n}$, and $u_1, \ldots, u_n \in V = \mathsf{K}^{n\times 1}$;

Output: matrices $H_i \in \mathsf{K}^{n\times d_i}$ for $1 \le i \le n$ such that

- $\sum_{1\le i\le n} d_i = n$, where $d_i \in \mathbb{N}$ may equal 0;
- $H_i = [u_i|Tu_i|\cdots|T^{d_i-1}u_i]$ if $d_i > 0$;
- the columns of $H_1, \ldots, H_i$ form a basis of the $\mathsf{K}[x]$-module generated by $u_1, \ldots, u_i$;

(1) Let $H^{(0)} = I \in \mathsf{K}^{n\times n}$;

(2) For $i$ from 1 to $n$ do

(3) From $T$,$H_1, \ldots, H_{i-1}$, and $R^{(i-1)}$, compute $H_i = [u_i|\cdots|T^{d_i-1}u_i] \in \mathsf{K}^{n\times d_i}$ and $R^{(i)} \in \mathsf{K}^{n\times n}$ using Theorem 3.2 such that

- $H_i$ has rank $d_i$ and $T^{d_i}u_i$ is linearly dependent on the columns of $H_1, \ldots, H_i$ ($d_i$ may be zero);
- $R^{(i)}[H_1|\cdots|H_i]$ is upper triangular;

**End.**

**Theorem 3.3.** *The algorithm* `ModuleBasis` *above works as specified. On input $T \in \mathsf{K}^{n\times n}$ and $u_1, \ldots, u_n \in V$ it requires $O(n^3)$ operations in $\mathsf{K}$.*

**Proof.** At each iteration of the main loop, in step (3), we use Theorem 3.2 on $u_i$ with the ($T$-invariant) vector space with basis given by the columns of $H_1, \ldots, H_{i-1}$, and with the row-reducing matrix $R^{(i-1)}$, to obtain $H_i$ and a matrix $R^{(i)}$ which row reduces $[H_1|\ldots|H_i]$. When $i = 1$ we simply use Theorem 3.2 on the vector space $\{0\}$ and the identity matrix.

Iteration $i$ of the loop requires $O(e_i n^2)$ where $e_i = \deg\min(T \bmod W_{i-1}; u_i + W_{i-1})$ with $W_0 = \{0\}$ and $W_{i-1} = \mathrm{Orb}(T; u_1) + \cdots + \mathrm{Orb}(T; u_{i-1})$ for $i > 0$. Since $\sum_{1\le i\le n} e_i = n$, the total number of operations in $\mathsf{K}$ required is $O(n^3)$. $\square$

**Corollary 3.4.** *Let $\#\mathsf{K} \ge n^2$. Given $T \in \mathsf{K}^{n\times n}$, the algorithm* `FindModCyc`*, using* `ModuleBasis` *in step (2), finds vectors $w_1, \ldots, w_k \in V$ and matrices $H_i \in \mathsf{K}^{n\times d_i}$ for $1 \le i \le k$ defined as follows:*

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

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

*(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* $1 \le i \le k$.

*The algorithm is probabilistic, returns the correct answer with probability at least 1/4, but may return an incorrect answer. It requires $O(n^3)$ operations in $\mathsf{K}$.*

When $\#\mathsf{K} \ge n^2$, the subroutine `ModuleBasis` is the only additional tool required to make the algorithm `FrobeniusForm` in Section 2.4 run with $O(n^3)$ operations in $\mathsf{K}$ using conventional matrix and polynomial arithmetic. It is used in step (2) of `FindModCyc` to allow this subroutine to run with $O(n^3)$ operations in $\mathsf{K}$, as described in Corollary 3.4. As well, `ModuleBasis` is used directly in step (7) of `FrobeniusForm`. All other steps in `FrobeniusForm` can be performed with $O(\mathrm{MM}(n) + n\mathrm{M}(n))$ or $O(n^3)$ operations in $\mathsf{K}$ using standard matrix and polynomial arithmetic.

**Theorem 3.5.** *Let $T \in \mathsf{K}^{n\times n}$ over any field $\mathsf{K}$ with at least $n^2$ elements. With probability at least 1/4, the algorithm* `FrobeniusForm`*, using the subroutine* `ModuleBasis` *as described above, 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, $O(n^3)$ operations in $\mathsf{K}$ are sufficient, using standard matrix and polynomial arithmetic.*

## 3.2 Computing the Frobenius Form over a Small Field with Standard Arithmetic

When the field $\mathsf{K}$ has size $q < n^2$, the algorithm `FrobeniusForm` presented in Section 2.4 returns the Frobenius form $S \in \mathsf{K}^{n\times n}$ of $T$ as desired. Unfortunately, the transition matrix $U \in \mathsf{F}^{n\times n}$ it returns, such that $U^{-1}TU = S$, may have entries in some small extension field $\mathsf{F}$ of $\mathsf{K}$. Moreover, using standard matrix and polynomial arithmetic, it requires $O(n^3(\log_q n)^2 \log\log_q n)$ operations in $\mathsf{K}$. We can in fact do considerably better than this when we allow ourselves only standard arithmetic. In this section we give an algorithm to find the Frobenius form $S$ of $T$ and a transition matrix $U \in \mathsf{K}^{n\times n}$ such that $U^{-1}TU = S$, which requires $O(n^3 \log_q n)$ operations in $\mathsf{K}$ using standard matrix and polynomial arithmetic.

We make two modifications to the algorithm `FrobeniusForm` presented in Section 2.4. The major one is to the subroutine `FindModCyc` used in step (1), which gives a modular cyclic transition matrix for $T$ and a modular cyclic basis for $V = \mathsf{K}^{n\times 1}$ under $T$. The techniques used in `FindModCyc` to choose a modular cyclic basis which is correct with high probability does not necessarily work over a smaller field (i.e., Theorem 2.5 does not hold). When $q < n^2$, the best possible general lower bound on the probability of choosing a maximal vector in a $T$-invariant subspace of $V$ is $\Omega(1/\log_q n)$. A randomly chosen set of vectors $w_1, \ldots, w_k$ which generate a modular cyclic basis generates a modular cyclic basis with probability $\Omega(1/\log_q^k n)$, which is not a sufficiently strong lower bound to yield an efficient algorithm. In fact, for $q = 2$ it is not difficult to construct explicit examples where the probability of choosing a modular cyclic basis is $O(1/2^{n/2})$.

The new idea required is a more dynamic method of choosing maximal vectors in a vector space, which are correct with sufficiently high probability. These ideas are eventually applied in the algorithm `FindModCycDyn` below for generating a modular cyclic transition matrix which is correct with high probability. We first describe our technique informally. Let $W$ be any vector space over $\mathsf{K}$ with dimension $m \le n$. We use the following method to choose a maximal $w \in W$, that is, a $w$ which satisfies $\min(T; w) = \min(T; W)$. Fix $r = \lceil 2\log_q n \rceil$ and choose random vectors $w_1, \ldots, w_r \in$

$W$. Next compute $g_i = \min(T; w_i) \in \mathsf{K}[x]$ for $1 \le i \le r$ and $g = \mathrm{lcm}(g_1, \ldots, g_r)$. In Theorem 3.6 below, we show that $g = \min(T; W)$ with probability at least $1 - 1/n$. Now choose vectors $w \in W$ at random until one is found whose minimal polynomial is $g$. In Corollary 3.8 below we show that, for a randomly selected $w \in W$, $\min(T; w) = g$ with probability $\Omega(1/\log_q n)$. Thus, with an expected number of $O(\log_q n)$ choices of vectors in $W$ we can find a maximal vector with high probability. The only other change is the replacement of Keller-Gehrig's algorithm used in step (7) by the routine `ModuleBasis` presented in the previous section.

**Theorem 3.6.** *Let $\mathsf{K} = \mathbb{F}_q$ with $q < n^2$, $W$ any vector space over $\mathsf{K}$ with dimension $m \le n$, and $T: W \to W$ a $\mathsf{K}$-linear map. Let $r = \lceil 2 \log_q n \rceil$ and choose $u_1, \ldots, u_r \in W$ randomly and uniformly from $W$. Then*

$$\mathrm{Prob}\Big\{\mathrm{lcm}\,(\min(T; u_1), \ldots, \min(T; u_r)) = \min(T; W)\Big\} \ge 1 - \frac{1}{n}.$$

**Proof.** The idea of the proof is to show that Lemma 2.4 (which gives a lower bound on the probability of a randomly chosen vector in any vector space being maximal), applied over an extension field $\mathsf{F}$ of degree $r$ over $\mathsf{K}$, yields the desired result.

Let $\psi \in \mathsf{K}[z]$ be irreducible of degree $r$ and $\mathsf{F} = \mathsf{K}[z]/(\psi)$, a field of degree $r$ over $\mathsf{K}$. The map $T$ acts naturally on $W_0 = W \otimes \mathsf{F}$, with $T(\alpha w) = \alpha T w$ for any $w \in W$ and $\alpha \in \mathsf{F}$. Recall that if $w_1, \ldots, w_m \in W$ is a basis for $W$, the tensor product $W \otimes \mathsf{F}$ is a $\mathsf{K}$-vector space of dimension $mr$ with basis $\{w_i z^j | 1 \le i \le m,\ 0 \le j < r\}$. Since the minimal polynomial of $T$ is a rational invariant of $T$, $\min(T; W) = \min(T; W_0)$. In selecting $u_0, \ldots, u_r$ randomly from $W$, we have selected the vector

$$u = \sum_{0 \le i < r} u_i z^i \in W_0$$

randomly and uniformly from $W_0$. Since $\#\mathsf{F} \ge n^2$, by Lemma 2.4,

$$\mathrm{Prob}\,\{\min(T; u) = \min(T; W_0) = \min(T; W)\} \ge 1 - \frac{1}{n}.$$

What is $\min(T;u)$? Observe that for any $h \in \mathsf{K}[x]$,

$$h(T)u = h(T) \sum_{0 \le i < r} u_i z^i = \sum_{0 \le i < r} z^i h(T) u_i = 0$$

if and only if $h(T)u_i = 0$ for $0 \le i < r$. Thus

$$\min(T;u) = \operatorname{lcm}(\min(T;u_1), \ldots, \min(T;u_r)),$$

whence

$$\operatorname{Prob}\left\{\operatorname{lcm}(\min(T;u_1), \ldots, \min(T;u_r) = \min(T;W)\right\} \ge 1 - \frac{1}{n}.$$

□

Now we actually have to find a vector whose minimal polynomial is equal to the polynomial we have determined is $T$'s minimal polynomial with high probability. The technique we employ is essentially Wiedemann's (1986) method for finding the minimal polynomial of a sparse matrix and solving sparse systems of linear equations. We examine the probability that a randomly chosen $w \in W$ has $\min(T;w) = \min(T;W)$. To do this we link this probability with an analogue, for polynomials over finite fields, of Euler's totient function for integers. For any integer $n \in \mathbb{N}$, Euler's totient function $\phi(n)$ is the number of positive integers less than $n$ and relatively prime to $n$. For the remainder of this section we set $\mathsf{K} = \mathbb{F}_q$, where $q \in \mathbb{N}$ is a prime power. Denote by $\mathcal{I}_{\mathsf{K}} \subseteq \mathsf{K}[x]$ the set of monic irreducible polynomials of positive degree in $\mathsf{K}[x]$ and by $\mathcal{I}_{\mathsf{K}}(m) \subseteq \mathcal{I}_{\mathsf{K}}$ the set of monic irreducible polynomials in $\mathsf{K}[x]$ of degree $m$. We define the *size* $|f|$ of any $f \in \mathsf{K}[x] \setminus \mathsf{K}$ by $|f| = \#(\mathsf{K}[x]/(f)) = q^{\deg f}$, the number of elements of the residue class ring of $\mathsf{K}[x]$ modulo $f$. The analogue of Euler's totient function for integers is the number $\Phi_{\mathsf{K}}(f)$ of polynomials in $\mathsf{K}[x]$ of smaller degree than $f \in \mathsf{K}[x]$ which are relatively prime to $f$. This is the number of units in $\mathsf{K}[x]/(f)$, and (see Lidl & Niederreiter 1983, Lemma 3.69)

$$\Phi_{\mathsf{K}}(f) = |f| \prod_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ g \mid f}} (1 - |g|^{-1}).$$

The following is essentially the same as Proposition 4 of Wiedemann (1986).

**Theorem 3.7.** *Let $W$ be any vector space of dimension $m$ over $\mathsf{K} = \mathbb{F}_q$ and $T: W \to W$ a linear map with $f_1 = \min(T; W)$ and $\deg f_1 = d_1$. The number of vectors $w \in W$ such that $\min(T; w) = f_1$ is at least $\Phi_{\mathsf{K}}(f)q^{m-d_1}$.*

**Proof.** Consider the cyclic decomposition

$$W = W_1 \oplus W_2 \oplus \cdots \oplus W_k$$

of $W$ under $T$, where $f_i = \min(T; W_i)$ is the $i$th invariant factor of $T$ and $f_i \mid f_{i-1}$ for $2 \le i \le k$. The vector space $W_1$ is isomorphic to $\mathsf{K}[x]/(f_1)$ as a $\mathsf{K}[x]$-module, and the cyclic vectors in $\mathsf{K}[x]/(f_1)$ (those whose minimal polynomials are equal to $f_1$) are the images of polynomials in $\mathsf{K}[x]$ of degree less than $\deg f_1$ which are relative prime to $f_1$. Thus, there are $\Phi_{\mathsf{K}}(f_1)$ vectors in $w_1 \in W_1$ such that $\min(T; w_1) = f_1$. For any $u \in W_2 \oplus \cdots \oplus W_k$, we also have $\min(T; w_1 + u) = f_1$. Since $\#(W_2 \oplus \cdots \oplus W_k) = q^{m-d_1}$, the total number of $w \in W$ such that $\min(T; w) = f_1$ is at least $\Phi_{\mathsf{K}}(f_1)q^{m-s}$. □

In the next section we give asymptotically tight lower bounds on $\Phi_{\mathsf{K}}(f)$ for all fields $\mathsf{K}$ and all polynomials $f \in \mathsf{K}[x]$. For our purposes in this section we require only the bound from Theorem 3.15 which shows, for all finite fields $\mathsf{K} \cong \mathbb{F}_q$ and all $f \in \mathsf{K}[x]$ of degree $n$, that $\Phi_{\mathsf{K}}(f)/|f| > 1/e$ when $n \le q$ and $\Phi_{\mathsf{K}}(f)/|f| > 1/(5 \log_q n)$ when $n > q$. We denote this lower bound by a function

$$\rho(q, n) = \begin{cases} 1/(5 \log_q n) & \text{if } n > q, \\ 1/e & \text{if } n \le q, \end{cases}$$

so $\Phi_{\mathsf{K}}(f) \ge \rho(q, \deg f)$ for any $f \in \mathsf{K}[x] \setminus \{0\}$ by Theorem 3.15.

**Corollary 3.8** (to Theorem 3.7). *Let $\mathsf{K} \cong \mathbb{F}_q$, $W$ any vector space over $\mathsf{K}$ with dimension $m \le n$, and $T: W \to W$ a $\mathsf{K}$-linear map. If $w \in W$ is chosen randomly and uniformly from $W$ then*

$$\text{Prob}\,\{\min(T; w) = \min(T; W)\} \ge \rho(q, n).$$

We now give our algorithm `FindModCycDyn` for constructing a modular cyclic transition matrix for a $T \in \mathsf{K}^{n \times n}$, which is correct with high probability.

**Algorithm:** `FindModCycDyn`

Input: $T \in \mathsf{K}^{n\times n}$, where $\#\mathsf{K} \geq n^2$ and $L$ is a subset of $\mathsf{K}$ 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) Let $\tau := 0$;

(2) **For** $i$ from 1 to $n$ while $\sum_{1\leq j<i} d_j < n$ do

(3) Choose random vectors $u_1, \ldots, u_r \in V$, and compute $g_j := \min(T \bmod V_{i-1}; u_j + V_{i-1}) \in \mathsf{K}[x]$ for $1 \leq j \leq r$, and let $f_i := \mathrm{lcm}(g_1, \ldots, g_r)$;

**Repeat** while $\tau \leq 2n/\rho(q,n)$:

(4) Choose a random $w_i \in V$;

(5) Compute $g = \min(T \bmod V_{i-1}; w_i + V_{i-1})$,
$H_i = [w_i|Tw_i|\cdots|T^{d_i-1}w_i] \in \mathsf{K}^{n\times d_i}$, where $d_i = \deg g$,
$R^{(i+1)} \in \mathsf{K}^{n\times n}$ which row reduces $[H_1|\cdots|H_i]$;

(6) Let $\tau := \tau + d_i$;

(7) If $g = f_i$ then we have succeeded, goto step (8);

End Loop;

(8) If $\tau > 2n/\rho(q,n)$ Then report "Failure" and quit;

End Loop;

**End.**

**Theorem 3.9.** *Assume* $\mathsf{K} \cong \mathbb{F}_q$, *where* $q < n^2$. *Given* $T \in \mathsf{K}^{n\times n}$, *the algorithm* `FindModCycDyn` *finds* $w_1, \ldots, w_k \in V$ *and* $H_i \in \mathsf{K}^{n\times d_i}$ *for* $1 \leq i \leq k$ *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, returns the correct answer with probability at least 1/8, but may return an incorrect answer. It requires* $O(n^3 \log_q n)$ *operations in* $\mathsf{K}$.

**Proof.** The variable $\tau$, initialized to 0 in step 1, is a global counter of work done in the algorithm. It is incremented appropriately and the algorithm terminates with a message of failure if it gets too large.

Let $r = \lceil 2\log_q n \rceil$. The algorithm proceeds in stages $i = 1, 2, \ldots, k$. Prior to starting stage $i$, assume we have chosen $w_1, \ldots, w_{i-1} \in V$ correctly with probability at least $(1-1/n)^{i-1}$. Assume also that $H_1, \ldots, H_{i-1}$ satisfying (iii) and (iv) have been determined, as well as an invertible $R^{(i)}$ which row reduces $[H_1|\cdots|H_{i-1}]$ (where $R^{(1)}$ is the $n \times n$ identity matrix).

First, we show that iteration $i$ of the outer loop returns the correct answer or fails with probability at most $1 - 1/n$. A key observation here is that when we choose a vector $v$ uniformly and randomly from $V$, then $v + V_{i-1}$ is selected uniformly and randomly from $V/V_{i-1}$. This is because the map carrying any vector $v \in V$ to $v + V_{i-1}$ maps exactly $q^{\dim V_{i-1}}$ vectors to each vector in $V/V_{i-1}$. Thus, by Theorem 3.6, $f_i$ computed in step (1) equals $\min(T \bmod V_{i-1}; V/V_{i-1})$ with probability at least $1 - 1/n$. Therefore, assuming that we do not use too much work (i.e., we don't fall through the inner loop to step (8) where we report failure), the algorithm executed over all stages $i = 1 \ldots k$ returns the correct answer with probability $(1-1/n)^k \geq 1/4$.

At the beginning of iteration $i$ of the outer loop, assume $\tau$ has value $\tau_i$. Let $e_i$ be the number of times we go through the inner loop (steps (4)-(7)) in iteration $i$ of the outer loop. The expected value of $e_i$ is bounded below by $e_i/\rho(q,n)$. This follows from Corollary 3.8, since we choose vectors $w_i \in V$, and hence vectors $w_i + V_{i-1} \in V/V_{i-1}$, randomly and uniformly. Thus, the expected value of $\tau_{i+1} - \tau_i$ is at least $d_i/\rho(q,n)$, and the expected value of $\tau$ at the completion of all $k$ stages is at least $n/\rho(q,n)$. By Markov's inequality the probability that $\tau \geq 2n/\rho(q,n)$ is at most 1/2, so the probability of the algorithm returning the correct answer (and not failure) is at least 1/8.

To compute the total number of operations in $\mathsf{K}$ required by the algorithm, we first note that step (5) requires $O(n^2 d_i)$ operations in $\mathsf{K}$ by Theorem 3.2. Assume we enter iteration $i$ of the outer loop with $\tau_{i-1} = \tau$. After completing iteration $i$ of the outer loop, if we have made $l_i \geq 1$ iterations of the inner loop (4)-(7), then $\tau - \tau_{i-1} = l_i d_i$, and a total of $O(l_i n^2 d_i)$ operations are required for iterations $i$ of the outer loop. Thus the total number of operations required is

$$\sum_{1\le i\le k} n^2 l_i d_i = n^2\tau_k \le 2n^3/\rho(q,n) = O(n^3/\rho(q,n)),$$

where $\tau_k$ is the value of $\tau$ at the end of iteration $k$ of the outer loop. The total number of operations required by step (3) over all iterations is $O(n^3 r)$ or $O(n^3 \log_q n)$. $\square$

**Theorem 3.10.** *Let $T \in \mathsf{K}^{n\times n}$ over any field $\mathsf{K}$ with $q < n^2$ elements. The algorithm* `FrobeniusForm`*, using the subroutine* `FindModCycDyn` *in step (1) and* `ModuleBasis` *in step (7), returns the Frobenius form $S$ of $T$ and an invertible $U \in \mathsf{K}^{n\times n}$ such that $S = U^{-1}TU$. The algorithm succeeds with probability at least $1/8$, and otherwise reports "failure". The algorithm requires $O(n^3 \log_q n)$ operations in $\mathsf{K}$.*

**Proof.** All steps of `FrobeniusForm` except for the call to `FindModCyc` in step (1) and the use of Keller-Gehrig's (1985) algorithm in step (7) can be performed with $O(\mathrm{MM}(n) + n\mathrm{M}(n))$ or $O(n^3)$ operations in $\mathsf{K}$ using standard matrix and polynomial arithmetic. We use `FindModCycDyn` in step (1), which requires $O(n^3 \log_q n)$ operations in $\mathsf{K}$ as shown in Theorem 3.9, and `ModuleBasis` in step (7), which requires $O(n^3)$ operations in $\mathsf{K}$ as shown in Theorem 3.3. $\square$

## 3.3 The Euler Totient Function for Polynomials over Finite Fields

For any integer $n \in \mathbb{N}$, Euler's totient function $\phi(n)$ is the number of positive integers less than $n$ and relatively prime to $n$. The analogue for polynomials over a finite field of Euler's totient function for integers is the number $\Phi_{\mathsf{K}}(f)$ of polynomials in $\mathsf{K}[x]$ of smaller degree than $f \in \mathsf{K}[x]$ which are relatively prime to $f$. In this section we give a lower bound on $\Phi_{\mathsf{K}}(f)$ for any finite field $\mathsf{K}$ and any $f \in \mathsf{K}[x]$ which is the best possible in a strong sense. This lower bound is required in the algorithm of the previous section for computing a modular cyclic transition matrix of a matrix over a small field, and will also be applied in the construction of normal bases in Chapter 6.

We assume throughout this section that $\mathsf{K}$ is a field with a prime power $q$ number of elements. Denote by $\mathcal{I}_{\mathsf{K}} \subseteq \mathsf{K}[x]$ the set of monic irreducible polynomials of positive degree in $\mathsf{K}[x]$ and by $\mathcal{I}_{\mathsf{K}}(m) \subseteq \mathcal{I}_{\mathsf{K}}$ the set of monic irreducible polynomials in $\mathsf{K}[x]$ of degree $m$. We define the *size* $|f|$ of any $f \in \mathsf{K}[x] \setminus \mathsf{K}$ by $|f| = \#(\mathsf{K}[x]/(f)) = q^{\deg f}$ the number of elements in the residue class ring of $\mathsf{K}[x]$ modulo $f$. This is the number of units in $\mathsf{K}[x]/(f)$, and (see Lidl & Niederreiter 1983, Lemma 3.69)

$$\Phi_{\mathsf{K}}(f) = |f| \prod_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ g|f}} (1 - |g|^{-1}).$$

In this section we demonstrate an asymptotically tight lower bound on $\Phi_{\mathsf{K}}$ for any finite field $\mathsf{K}$. Such a bound was quite possibly known to Carlitz, who comments indirectly on its relatively large size in (Carlitz, 1952), though he presents no formal proof. Wiedemann (1986) exhibits an explicit and elegant lower bound on $\Phi_{\mathsf{K}}(f)$. He shows for any $f \in \mathsf{K}[x]$ that

$$\Phi_{\mathsf{K}}(f)/|f| \geq 1/(6l) \quad \text{when} \quad \sum_{1 \leq i \leq l} q^i \geq \deg f.$$

In (von zur Gathen & Giesbrecht, 1990) we independently prove a lower bound which is which is somewhat better than Wiedemann's for large fields, though somewhat worse for smaller ones. In this thesis we give a proof very much in the spirit of the one of Wiedemann, demonstrating an asymptotic lower bound which is the best possible in a strong sense, and which is valid for all polynomials over all finite fields. This theorem improves on the bounds of both Wiedemann (1986) and von zur Gathen & Giesbrecht (1990).

For $m \geq 1$, let $\mathrm{N}_q(m) = \#\mathcal{I}_{\mathsf{K}}(m)$. Lidl & Niederreiter (1983), Exercises 3.26 and 3.27 show

$$\frac{q^m}{m} - \frac{q(q^{m/2}-1)}{m(q-1)} \leq \mathrm{N}_q(m) \leq \frac{q^m - q}{m} \leq \frac{q^m}{m}. \tag{3.1}$$

We will require the following easily proven lemma on real functions.

**Lemma 3.11.** *Let $a: \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ be non-decreasing and $b: \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ be non-increasing, and $0 \leq x_i \leq 1$ for all $i \in \mathbb{N}$. For $k \in \mathbb{N}$ such that*

$$\sum_{1 \leq i} x_i a(i) \leq \sum_{1 \leq i \leq k} a(i), \quad \textit{we have} \quad \sum_{1 \leq i} x_i b(i) \leq \sum_{1 \leq i \leq k} b(i).$$

**Proof.** Suppose conversely that

$$\sum_{1\le i} x_i b(i) > \sum_{1\le i\le k} b(i),$$

whence

$$\sum_{k<i} x_i b(i) > \sum_{1\le i\le k} (1-x_i)b(i), \quad \text{and} \quad \sum_{k<i} x_i b(i)/b(1) > \sum_{1\le i\le k} (1-x_i)b(i)/b(1),$$

since $b(1) > 0$. Also, $b(i)/b(1) \le a(i)/a(1)$ for all $i \ge 1$ since $a$ is non-decreasing and $b$ non-increasing, so

$$\sum_{k<i} x_i a(i)/a(1) > \sum_{1\le i\le k} (1-x_i)a(i)/a(1), \quad \text{or} \quad \sum_{1\le i} x_i a(i) > \sum_{1\le i\le k} a(i),$$

a contradiction. □

The best way to state bounds on $\Phi_{\mathsf{K}}(f)$ for $f \in \mathsf{K}[x]$ is in terms of the least number $k$ such that the product $g \in \mathsf{K}[x]$ of all polynomials in $\mathcal{I}_{\mathsf{K}}(i)$ for $1 \le i \le k$ has degree at least $\deg f$. Informally, $g$ is the smoothest polynomial with a degree near to that of $f$ (by $g$ being smoothest we mean that it is the polynomial with the largest number of distinct irreducible factors), and $\Phi_{\mathsf{K}}(f)/|f|$ is minimized at smooth polynomials. The most exact result we obtain is stated in terms of the $k$th harmonic number $H_k = \sum_{1\le i\le k} 1/i$.

**Theorem 3.12.** *Let $q$ be a prime power, $\mathsf{K} = \mathbb{F}_q$, and let $f \in \mathsf{K}[x]$ have degree $n \ge 1$. Suppose $\sum_{1\le i\le k} i\mathrm{N}_q(i) \ge n$. Then $\Phi_{\mathsf{K}}(f)/|f| \ge e^{-H_k}$.*

**Proof.** For $i \ge 1$, let $x_i \in [0,1]$ be such that $x_i\mathrm{N}_q(i)$ is the number of polynomials in $\mathcal{I}_{\mathsf{K}}(i)$ dividing $f$. Thus

$$\Phi_{\mathsf{K}}(f)/|f| = \prod_{i\ge 1}(1-q^{-i})^{x_i \mathrm{N}_q(i)}.$$

Taking the logarithm of both sides we obtain

$$\begin{aligned}
\log \Phi_{\mathsf{K}}(f)/|f| &= \sum_{i\geq 1} \log(1-q^{-i}) x_i \mathrm{N}_q(i) \geq -\sum_{i\geq 1} (q^{-i}+q^{-2i}) x_i \mathrm{N}_q(i) \\
&\geq -\sum_{i\geq 1} \frac{x_i(q^i-q)}{iq^i} - \sum_{i\geq 1} \frac{x_i(q^i-q)}{iq^{2i}} \\
&= -\sum_{i\geq 1} \frac{x_i}{i} + \sum_{i\geq 1} \frac{x_i}{iq^{i-1}} - \sum_{i\geq 1} \frac{x_i}{iq^i} + \sum_{i\geq 1} \frac{x_i}{iq^{2i-1}} \geq -\sum_{i\geq 1} \frac{x_i}{i},
\end{aligned}$$

using (3.1) and the easily proven fact that $\log(1-y) \geq -y-y^2$ for $0 \leq y \leq 1/2$. We want to apply Lemma 3.11, with $a(i) = i\mathrm{N}_q(i)$ and $b(i) = 1/i$ for $i \geq 1$. To do this we require $i\mathrm{N}_q(i)$ is non-decreasing for $i \geq 1$; applying (3.1) for $i \geq 2$ we have

$$i\mathrm{N}_q(i) - (i-1)\mathrm{N}_q(i-1) \geq q^i - \frac{q^{i/2+1}}{q-1} - q^{i-1} + q = q^i \left(1 - \frac{1}{(q-1)q^{i/2-1}} - \frac{1}{q} + \frac{1}{q^{i-1}}\right) \geq 0$$

for all $q \geq 2$, so $i\mathrm{N}_q(i)$ is a non-decreasing function of $i$. Thus, by Lemma 3.11, $\log \Phi_{\mathsf{K}}(f)/|f| \geq -H_k$, or $\Phi_{\mathsf{K}}(f)/|f| \geq e^{-H_k}$. □

In fact we cannot do much better than this in general, as demonstrated by the following.

**Theorem 3.13.** *For every finite field $\mathsf{K} \cong \mathbb{F}_q$, there exists an infinite family $\mathcal{F} \subseteq \mathsf{K}[x]$ such that for each $f \in \mathcal{F}$, there exists a $k$ with $\deg f = \sum_{1\leq i\leq k} i\mathrm{N}_q(i)$, and*

$$\frac{\Phi_{\mathsf{K}}(f)}{|f|} \leq \frac{1}{e^{H_k}} \left(1 + \frac{1}{\sqrt{q}}\right)^{q/(q-1)}.$$

**Proof.** For each $k \geq 1$, let

$$f_k = \prod_{1\leq i\leq k} \prod_{g\in\mathcal{I}_{\mathsf{K}}(i)} g,$$

of degree $n_k = \sum_{1\leq i\leq k} i\mathrm{N}_q(i)$. Then

$$\begin{aligned}
\log \frac{\Phi_{\mathsf{K}}(f)}{|f|} &= \sum_{1\leq i\leq k} \mathrm{N}_q(i) \log(1-\frac{1}{q^i}) = -\sum_{1\leq i\leq k} \mathrm{N}_q(i) \sum_{j\geq 1} \frac{1}{jq^{ji}} \\
&\leq -\sum_{1\leq i\leq k} \left(\frac{q^i}{i} - \frac{q}{q-1} \cdot \frac{q^{i/2}-1}{i}\right) \cdot \frac{1}{q^i} \leq -\sum_{1\leq i\leq k} \frac{1}{i} + \frac{q}{q-1} \sum_{i\geq 1} \frac{1}{iq^{i/2}} - \frac{1}{iq^i} \\
&\leq -H_k - \frac{q}{q-1} \left(\log(1-q^{-1/2}) - \log(1-q^{-1})\right).
\end{aligned}$$

It follows immediately that

$$\frac{\Phi_{\mathsf{K}}(f)}{|f|} \le \frac{1}{e^{H_k}} \left( \frac{\sqrt{q}}{\sqrt{q}-1} \right)^{q/(q-1)} \left( 1 - \frac{1}{q} \right)^{q/(q-1)} = \frac{1}{e^{H_k}} \left( 1 + \frac{1}{\sqrt{q}} \right)^{q/(q-1)}.$$

□

For practical purposes it is desirable to state our lower bound from Theorem 3.12 in terms of $\log_q n$ as opposed to the $k$th harmonic number. We make use of the following lemma from number theory bounding harmonic numbers (see Apostol 1976, Theorem 3.2a).

**Lemma 3.14.** *For $k \ge 1$*

$$H_k = \sum_{1 \le i \le k} 1/i \le \log_e k + C + 1/k$$

*where $C$ is Euler's constant.*

**Proof.** Using Euler's summation formula (see Apostol 1976, Theorem 3.1), we find

$$\sum_{1 \le i \le k} \frac{1}{i} = \int_1^k \frac{dt}{t} - \int_1^k \frac{t - \lfloor t \rfloor}{t^2} dt + 1 - \frac{k - \lfloor k \rfloor}{k}$$

For the upper bound we use

$$\begin{aligned}
\sum_{1 \le i \le k} \frac{1}{i} \le{} & \log_e k - \int_1^k \frac{t - \lfloor t \rfloor}{t^2} dt + 1 = \log_e k + 1 - \int_1^\infty \frac{t - \lfloor t \rfloor}{t^2} dt + \int_k^\infty \frac{t - \lfloor t \rfloor}{t^2} dt \\
\le{} & \log_e k + 1 - \int_1^\infty \frac{t - \lfloor t \rfloor}{t^2} dt + \int_k^\infty \frac{1}{t^2} dt = \log_e k + 1 - \int_1^\infty \frac{t - \lfloor t \rfloor}{t^2} dt + \frac{1}{k} \\
={} & \log_e k + C + \frac{1}{k},
\end{aligned}$$

where

$$C = 1 - \int_1^\infty \frac{t - \lfloor t \rfloor}{t^2} dt = \lim_{k \to \infty} \left( \sum_{1 \le i \le k} \frac{1}{i} - \log_e k \right)$$

is Euler's constant (see Apostol 1976, pp. 53). □

We now state a lower bound for $\Phi_{\mathsf{K}}(f)$ in terms of $n = \deg f$ and $q = \#\mathsf{K}$.

**Theorem 3.15.** *Let $q$ be a prime power, $\mathsf{K} = \mathbb{F}_q$, and let $f \in \mathsf{K}[x]$ have degree $n \geq 1$. If $n \leq q$ then $\Phi_{\mathsf{K}}(f)/|f| \geq 1/e$, while for $n > q$ we have*

$$\frac{\Phi_{\mathsf{K}}(f)}{|f|} > \frac{c}{\lceil \log_q n \rceil} \cdot \left(1 - \frac{1}{\lceil \log_q n \rceil}\right), \quad \text{and} \quad \frac{\Phi_{\mathsf{K}}(f)}{|f|} > \frac{1}{5 \log_q n},$$

*where $c = e^{-C}$ and $C \approx 0.577216$ is Euler's constant.*

**Proof.** First we show for all $n \geq 1$ and $q \geq 2$ that when $k = \lceil \log_q n \rceil$ then

$$\sum_{1 \leq i \leq k} i \mathrm{N}_q(i) \geq n.$$

We begin with the cases $1 \leq k \leq 5$ separately.

- $k = \lceil \log_q n \rceil = 1:$ $\quad \mathrm{N}_q(1) = q \geq n$
- $k = \lceil \log_q n \rceil = 2:$ $\quad \mathrm{N}_q(1) + 2\mathrm{N}_q(2) = q^2 \geq n$
- $k = \lceil \log_q n \rceil = 3:$ $\quad \sum_{1 \leq i \leq 3} i\mathrm{N}_q(i) = q^3 + q^2 - q \geq q^3 \geq n$
- $k = \lceil \log_q n \rceil = 4:$ $\quad \sum_{1 \leq i \leq 4} i\mathrm{N}_q(i) = q^4 + q^3 - q \geq q^4 \geq n$
- $k = \lceil \log_q n \rceil = 5:$ $\quad \sum_{1 \leq i \leq 5} i\mathrm{N}_q(i) = q^5 + q^4 + q^3 - 2q \geq q^5 \geq n$

Now assume that $k = \lceil \log_q n \rceil \geq 6$. Then

$$\begin{aligned}
\sum_{1 \leq i \leq k} i\mathrm{N}_q(i) &\geq \sum_{1 \leq i \leq k} q^i - \frac{q(q^{i/2} - 1)}{q - 1} = \frac{q^{k+1} - q}{q - 1} - \frac{q}{q - 1} \left( \frac{q^{(k+1)/2} - q^{1/2}}{q^{1/2} - 1} \right) \\
&= q^k \left( \frac{q - q^{-k+1}}{q - 1} - \frac{1 - q^{-k/2}}{q^{(k-3)/2}(q - 1)(q^{1/2} - 1)} \right) \\
&\geq n \left( \frac{q - q^{-5}}{q - 1} - \frac{1}{q^{3/2}(q - 1)} \cdot \frac{1 - q^{-3}}{q^{1/2} - 1} \right) \geq n
\end{aligned}$$

for all $q \geq 2$.

By Theorem 3.12 $\Phi_{\mathsf{K}}(f)/|f| \geq e^{-H_k}$, when $\sum_{1 \leq i \leq k} i\mathrm{N}_q(i) \geq n$. When $n \leq q$ we find $k = H_k = 1$ and $\Phi_{\mathsf{K}}(f)/|f| \geq e^{-1}$. When $n > q$, then $k = \lceil \log_q n \rceil$ suffices, so

$$\frac{\Phi_{\mathsf{K}}(f)}{|f|} \geq e^{-H_k} \geq \frac{e^{-C-1/k}}{k} \geq \frac{c}{\lceil \log_q n \rceil} \left( 1 - \frac{1}{\lceil \log_q n \rceil} \right),$$

using Lemma 3.14.

We now show $\Phi_{\mathsf{K}}(f)/|f| > 1/(5\log_q n)$ for $n > q$. Let $k = \lceil \log_q n \rceil$. For $2 \le k \le 5$ we note that $q^{k-1}+1 \le n \le q^k$, and we must show

$$\Phi_{\mathsf{K}}(f)/|f| \ge e^{-H_k} \ge 1/(5\log_q n),$$

or equivalently that $\log_q n \ge \log_q(q^{k-1}+1) \ge e^{H_k}/5$, which is easily verified for $2 \le k \le 5$. For $k \ge 6$,

$$\begin{aligned}\frac{\Phi_{\mathsf{K}}(f)}{|f|} &\ge \frac{c}{\lceil \log_q n \rceil} \cdot \left(1 - \frac{1}{\lceil \log_q n \rceil}\right) \ge \frac{c}{\log_q n} \cdot \left(1 - \frac{1}{\log_q n}\right) \cdot \left(1 - \frac{1}{1+\log_q n}\right) \\ &\ge \frac{5c/7}{\log_q n} \ge \frac{1}{5\log_q n}.\end{aligned}$$

□

# Chapter 4.

# Computing the Frobenius Form of an Integer Matrix

Many "actual applications" of the Frobenius form, and more generally of finding the structure of the eigenvalues and eigenvectors of a matrix, are numerical in nature and concern matrices over the integers, rationals and constant-precision reals (i.e., floating point arithmetic on most current computers). In this chapter we give two new and efficient algorithms for computing the Frobenius form of an integer matrix. Both these algorithms improve substantially on the previous state of the art for this problem. It is relatively easy to see, and we show in the next section, that an algorithm to compute the Frobenius form of an integer matrix is sufficient for solving this problem for matrices over the rationals and constant-precision reals as well. Integer matrices have the added advantage that their Frobenius forms are also integer matrices, making them particularly amenable to modular techniques (see Section 1.3).

The cost of our algorithms for computing the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of an integer matrix $T \in \mathbb{Z}^{n\times n}$ will be measured in terms of $n$ and $\alpha = \|T\|_\infty$ (recall from Section 1.3 that $\|T\|_\infty$ is the largest absolute value of any entry in $T$). We present both a Las Vegas algorithm and a faster Monte Carlo algorithm. For each algorithm we give two different (but certainly related) cost analyses. The first is an asymptotic cost analysis in terms of bit operations, where the cost of multiplication of matrices is abstracted by the function MM, and that of multiplying polynomials and integers is abstracted by the function M (see Section 1.3). The second type of result we state is also in terms of bit operations, and only makes use of *standard* (matrix, polynomial, and integer) arithmetic, i.e., $\mathrm{MM}(n) = n^3$ and $\mathrm{M}(n) = n^2$. This is intended to capture the practical cost of solving a problem, a measure of the time required by some actual implementation of an algorithm. One of our goals is to have the result using standard arithmetic be as close as possible to the asymptotic results using the fastest possible arithmetic, and we usually achieve this within a multiplicative factor of about $n^3/\mathrm{MM}(n)$.

Asymptotically, our Las Vegas algorithm requires an expected number of $O\tilde{}(\mathrm{MM}(n) \cdot n^2 \log \alpha)$ bit operations to find the Frobenius form of $T$, where $\alpha = \|T\|_\infty$. When standard arithmetic is used, an expected number of $O(n^5(\log n + \log \alpha)\ell + n^4 \log^2 \alpha)$ bit operations are sufficient to execute this algorithm, where $\ell = \Theta(\log n + \log\log \alpha)$. We think of $\ell$ as the length of a word or short integer in bits; we perform most of the arithmetic in our algorithms with integers of this size. It is also possible to compute a transition matrix $U \in \mathbb{Q}$ to the Frobenius form (i.e., such that $U^{-1}TU = S$), within about the same time. Asymptotically, we can find $U$ with an expected number of $O\tilde{}(\mathrm{MM}(n) \cdot n^2 \log \alpha)$ bit operations, but due to the sizes of the entries in $U$—$O(n^2(\log n + \log \alpha))$ bits each—things are somewhat more difficult using standard arithmetic. We can compute a transition matrix $U \in \mathbb{Q}^{n\times n}$ with an expected number of $O(n^5(\log n + \log \alpha)\ell + n^4 \log^2 \alpha)$ bit operations using standard arithmetic, where the entries of a $U$ are represented modulo a set of $O(n^2(\log n + \log \alpha)/\ell)$ primes between $2^{\ell-1}$ and $2^\ell$. However, we know of no way to use Chinese remaindering to reconstruct the $n^2$ entries in $U$ within the time required by the rest of the algorithm (using standard arithmetic). Also, we know of no compelling reason why a transition matrix with a smaller infinity norm should exist.

We also exhibit a very fast Monte Carlo type probabilistic algorithm which requires an expected number of $O\tilde{}(n^4 \log \alpha)$ bit operations to find the Frobenius form of $T$. When implemented carefully with standard arithmetic, this algorithm allows us to compute the Frobenius form with an expected number of $O(n^4(\log n + \log \alpha)^2)$ bit operations. In this algorithm an incorrect Frobenius form is returned with probability at most $1/2^s$, where $s = \Omega(n(\log n + \log \alpha)/\ell)$. Once again, $\ell = \Theta(\log n + \log\log \alpha)$. We do not compute a transition matrix in the Monte Carlo algorithm.

Both our algorithms compute the Frobenius form of $T$ modulo a collection of small primes, and then recover the Frobenius form by means of Chinese remaindering. The main difficulty is that the Frobenius form of ($T \bmod p$) in $\mathbb{Z}_p^{n\times n}$ may not be ($S \bmod p$)

for some finite number of primes $p$. We describe a way of identifying these bad primes and show how to generate enough good primes, modulo which $T$ does have Frobenius form $S \bmod p$, to correctly compute the Frobenius form of $T$.

Other algorithms for computing the Frobenius form of an integer matrix have been proposed by Ozello (1987), Lüneburg (1987), and Mathieu & Ford (1990). Our algorithms are substantially faster than these, using the same and often even asymptotically slower — though more practical — algorithms for matrix, polynomial, and integer arithmetic. Asymptotically, the best previously known algorithm to solve this problem is by Mathieu & Ford (1990), and requires $O(n^6 \log n(\log n + \log \alpha))$ bit operations using fast integer arithmetic. Their cost analysis makes the (somewhat questionable) assumption that integer multiplication modulo $p^\mu$ requires $O(\mu \log \mu)$ bit operations, for a small prime $p$ (the best known algorithm requires $O(\mu \log \mu \log\log \mu)$ bit operations, assuming $p$ is constant). As well, it relies on an unproven (but experimentally justified) assumption which affects the time required, though not the correctness. The algorithm they propose finds the Frobenius form modulo a small prime $p$ and then lifts $p$-adically to obtain the integer Frobenius form $S \in \mathbb{Z}^{n\times n}$. While Mathieu & Ford's algorithm does make use of a subroutine to compute the Frobenius form of $T$ modulo a small prime $p$, which they assume requires $O(n^4)$ operations in $\mathbb{Z}_p$, the cost of their algorithm for the Frobenius form of $T$ is dominated by the cost of the $p$-adic lifting. Using our faster algorithm (in terms of field operations) from Theorem 2.12 for computing the Frobenius form modulo a prime does not yield an immediate improvement in the cost of their algorithm for integer matrices. Also note that Mathieu & Ford (1990) do not compute a transition matrix $U \in \mathbb{Q}^{n\times n}$ such that $U^{-1}TU = S$ within this time.

A modular algorithm is proposed by Ozello (1987) for finding the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of a matrix $T \in \mathbb{Z}^{n\times n}$. While this algorithm is not analyzed, it appears to require $O\tilde{}(n^5 \log \alpha)$ bit operations using standard arithmetic. However, Ozello does not address the question of choosing primes $p$ modulo which the Frobenius form of $T \bmod p$ is $S \bmod p$. For some matrices and specific selections of primes it does not appear that the algorithm works as stated. For example, $T$ may be equivalent to zero modulo a prime $p$, which certainly changes its Frobenius form modulo $p$. More subtle problems can occur when $p$ divides the discriminant of the minimal polynomial of $T$, where the multiplicity of elementary factors will be different modulo $p$, which potentially affects the invariant factors. We prove that similar assumptions to Ozello's are correct with high probability for randomly selected primes of suitable size. Ozello does not address the problem of computing a transition matrix $U \in \mathbb{Q}^{n\times n}$ such that $S = U^{-1}TU$ in his modular algorithm.

Ozello (1987) also proposes and analyzes an algorithm which makes extensive use of very large integers (e.g., integers containing $O(n^2 \log^2 \alpha)$ bits) for computing the Frobenius form of an integer matrix $T$ with $\alpha = \|T\|_\infty$. It requires $O\tilde{}(n^8 \log^2 \alpha)$ bit operations, at least up to a factor of $(\log n)^{O(1)}$. This algorithm also computes a transition matrix $U$ to the Frobenius form within this time. Finding a transition matrix efficiently is non-trivial since representing each entry in this matrix requires $O(n^2(\log n + \log \alpha))$ bits, at least by known constructions.

Howell (1973a, 1973b) first proposes using modular methods to compute what she refers to as the Frobenius form exactly, thereby avoiding problems with numerical stability, or at least trading them for problems of coefficient growth. Her definition of the Frobenius form does not match ours as it allows non-zeros in the entries above the diagonal blocks; it essentially the form produced by Danilevsky's (1937) algorithm, an upper Hessenberg matrix with companion matrices along the diagonal (see Wilkinson 1965, Sections 6.52 – 6.57). To avoid confusion we refer to this canonical form as the Danilevsky form of a matrix, and note that it is not unique, nor is it an invariant of the similarity class (it was proposed by Danilevsky (1937) as a way of computing the characteristic polynomial of a matrix; the polynomials corresponding to the companion blocks on the diagonal have product equal to the characteristic polynomial). She presents two algorithms, one computing the Danilevsky form modulo a single large prime, and one modulo many primes, from which the Danilevsky form of the integer matrix is determined by Chinese remaindering.

Problems related to that of computing the Frobenius form, such as computing the Jordan form and real Schur form, have received considerable attention in the numerical analysis literature; see (Golub & Van Loan, 1989) and (Wilkinson, 1965). From the numerical point of view, the problem of finding the Frobenius form (and

Jordan form) is inherently unstable, making computation using floating point arithmetic (with its inherent inaccuracies) very treacherous. Indeed, all rational matrices except those in a set of measure zero are *non-derogatory*, that is, have Frobenius forms equal to a single companion block. A slight perturbation of a rational matrix, such as that induced by an inexact representation of the entries (such as floating point numbers), will almost always be a non-derogatory matrix. Even finding the Frobenius form of a non-derogatory matrix (essentially the problem of finding the characteristic or minimal polynomial, if we do not insist on computing a transition matrix) is tricky, and implementations of algorithms such as Danilevsky's(1937), which could potentially generalize to algorithms for computing the Frobenius form of a general matrix, are highly unstable numerically (Frank, 1958). Attempts have been made to compensate for this instability, especially on special types of matrices, but with little success in general (see Golub & Van Loan 1989, Section 7.6.5). We regard this as good motivation to pursue our exact arithmetic techniques.

## 4.1 Frobenius Forms of Integer Matrices

In this section we first show that it is sufficient to have an algorithm to compute the Frobenius form of an integer matrix in order to compute the Frobenius form of a rational or constant-precision real matrix. This is accomplished by showing how to recover, from the Frobenius form of a scalar multiple of a matrix, the Frobenius form of the original matrix. This allows us to clear denominators or decimal digits by multiplication by a scalar to obtain an integer matrix. The Frobenius form of the original matrix can then be recovered from the Frobenius form of the scalar multiple.

Also in this section we examine the sizes of the entries which can occur in the Frobenius form of an integer matrix. As well, we consider the sizes of the integers encountered at various stages of the algorithm `FrobeniusForm`, assuming it was used as stated in Section 2.4, applied to an integer matrix (the algorithm `FrobeniusForm` works over any field, including $\mathbb{Q}$).

## The Frobenius Form of a Scalar Multiple of a Matrix

The Frobenius form of any matrix can easily be recovered from the Frobenius form of an integer multiple of it, as shown in the next lemma. Both rational matrices, and matrices of constant-precision real numbers have a scalar multiple which is an integer matrix, and we can compute the Frobenius form of this integer matrix, whence we then recover the Frobenius form of the original matrix.

**Lemma 4.1.** *Let $T \in \mathbb{Q}^{n\times n}$ and $c \in \mathbb{Z}\setminus\{0\}$ such that $T_0 = cT \in \mathbb{Z}^{n\times n}$. Suppose $T_0$ has Frobenius form $S_0 = \operatorname{diag}(C_{\bar f_1}, \ldots, C_{\bar f_k}) \in \mathbb{Z}^{n\times n}$, where $C_{\bar f_i} \in \mathbb{Z}^{d_i\times d_i}$ is the companion matrix of $\bar f_i \in \mathbb{Z}[x]$, the ith invariant factors of $T_0$, with $\deg \bar f_i = d_i$ for $1 \le i \le k$. The Frobenius form $S \in \mathbb{Q}^{n\times n}$ of $T$ is $S = c^{-1} D S_0 D^{-1}$, where*

$$D = \operatorname{diag}(1, c, \ldots, c^{d_1-1}, 1, \ldots, c^{d_2-1}, \ldots, 1, \ldots, c^{d_k-1}) \in \mathbb{Z}^{n\times n}.$$

**Proof.** Assume $U_0^{-1}(cT)U_0 = S_0$, with $U_0 \in \mathbb{Q}^{n\times n}$ invertible. Then $U_0^{-1}TU_0 = c^{-1}S_0$. For $1 \le i \le k$, let $D_i = \operatorname{diag}(1, c, \ldots, c^{d_i-1})$ and $\bar f_i = a_{i,0} + \cdots + a_{i,d_i-1}x^{d_i-1}$. Then $D_i(c^{-1}C_{\bar f_i})D_i^{-1} =$

$$\begin{pmatrix} 1 & & & \\ & c & & \\ & & \ddots & \\ & & & c^{d_i-1} \end{pmatrix} \begin{pmatrix} & & & -c^{-1}a_{i,0} \\ c^{-1} & & & -c^{-1}a_{i,1} \\ & \ddots & & \vdots \\ & & -c^{-1} & c^{-1}a_{i,d_i-1} \end{pmatrix} \begin{pmatrix} 1 & & & \\ & c^{-1} & & \\ & & \ddots & \\ & & & c^{-(d_i-1)} \end{pmatrix}$$

$$= \begin{pmatrix} & & & -c^{-d_i}a_{i,0} \\ 1 & & & -c^{-(d_i-1)}a_{i,1} \\ & \ddots & & \vdots \\ & & 1 & -c^{-1}a_{i,d_i-1} \end{pmatrix},$$

and $D_i(c^{-1}C_{\bar f_i})D_i^{-1}$ is the companion matrix of $f_i = c^{-d_i}\cdot \bar f_i(cx) \in \mathbb{Q}[x]$. Since $\bar f_i \mid \bar f_{i-1}$, we see $f_i \mid f_{i-1}$ for $2 \le i \le k$. Thus, if $D = \operatorname{diag}(D_1, \ldots, D_k) \in \mathbb{Q}^{n\times n}$ we have $D(c^{-1}S_0)D^{-1}$ in Frobenius form, so $S = c^{-1}DS_0D^{-1}$ is the Frobenius form of $T$. □

## Sizes of Integers in `FrobeniusForm` Over $\mathbb{Q}$

We next analyze the sizes of all intermediate results when running the algorithm `FrobeniusForm` on an integer matrix. When we say "intermediate result" here, we usually mean the input and output to each step in `FrobeniusForm`, though we refine a number of these steps. The steps themselves are treated as black boxes in which the sizes of intermediate computations will not be analysed. This is sufficient for our purposes.

Assume that we run `FrobeniusForm` on input $T \in \mathbb{Z}^{n\times n}$, where $\|T\|_\infty = \alpha$. In versions of the algorithm `FrobeniusForm` for an arbitrary field of size at least $n^2$, presented in Theorems 2.12 and 3.5, we choose the vectors $w_1, \ldots, w_n$ from $L^{n\times 1}$, where $L$ is a subset of $\mathsf{K}$ with size at least $n^2$. These algorithms certainly work on rational matrices (integer matrices are rational matrices), and we fix $L = \{0, \ldots, n^2 - 1\}$. In step (1) of `FrobeniusForm` we construct a modular cyclic transition matrix $H \in \mathbb{Z}^{n\times n}$ for $T$ using the subroutine `FindModCyc` discussed in Theorem 2.7 and Corollary 3.4 respectively. The maximum absolute value of an element in the $i$th column of $H$ is $\|T\|_\infty^{i-1} n^{i-1} \cdot n^2 = \alpha^{i-1}n^{i+1}$, for $i \geq 1$, and therefore $\|H\|_\infty \leq \alpha^{n-1}n^{n+1}$. Applying Hadamard's bound,

$$|\det(H)| \leq \left( \sum_{1\leq i\leq n} \alpha^{2i-2}n^{2i+2} \right)^{n/2} \leq \left( \frac{\alpha^{2n}n^{2n+4}}{\alpha^2 n^2 - 1} \right)^{n/2} < 2^{n/2}\alpha^{n^2-n}n^{n^2+n} = \delta, \quad (4.1)$$

where we have assigned $\delta = 2^{n/2}\alpha^{n^2-n}n^{n^2+n}$.

Next, we consider the sizes of the coefficients of the $f_i$'s $(1 \leq i \leq k)$ computed in step (3) of `FrobeniusForm`. These are all divisors of the characteristic polynomial of $T$. Mathieu & Ford (1990), Section 6, show that if $f \in \mathbb{Z}[x]$ is the characteristic polynomial of $T$ and $h = \sum_{0\leq i\leq k} c_i x^i \in \mathbb{Z}[x]$ divides $f$, then

$$\sum_{0\leq i\leq k} |c_i| \leq 2^k \prod_{1\leq i\leq n} (1 + s_j),$$

where $s_j$ is the 2-norm of the $j$th column of $T$ for $1 \leq j \leq n$. Since $s_j \leq \alpha\sqrt{n}$, we have

$$|c_i| \leq 2^k(1 + a\sqrt{n})^n = 2^k\alpha^n n^{n/2}(1 + 1/(a\sqrt{n}))^n < 2^k e^{n/2}\alpha^n n^{n/2},$$

for $1 \leq i \leq k$, where $e$ is the base of natural logarithms. We have the following:

**Lemma 4.2.** *Let $T \in \mathbb{Z}^{n\times n}$ with characteristic polynomial $f \in \mathbb{Z}[x]$ and $h \in \mathbb{Z}[x]$ dividing $f$ with $\deg h = d$. Then $\|f\|_\infty < \beta$ and $\|h\|_\infty < \beta$, where $\beta = 2^n e^{n/2}\alpha^n n^{n/2}$ and $\alpha = \|T\|_\infty$.*

In particular, all the invariant factors of $T$, and the $f_1, \ldots, f_k \in \mathbb{Z}[x]$ computed in `FrobeniusForm` (whether or not they are correct invariant factors), have infinity norms less than $\beta$.

Now consider the coefficients of the $g_{ij}$'s computed in step (3) of `FrobeniusForm`. These are all entries in $G = H^{-1}TH \in \mathbb{Q}^{n\times n}$. The adjoint $\det(H)H^{-1} \in \mathbb{Z}^{n\times n}$ of $H$ satisfies $\|\det(H)H^{-1}\|_\infty \le \delta$, so

$$\|\det(H)G\|_\infty \le n^2 \cdot \|\det(H)H^{-1}\|_\infty \cdot \|T\|_\infty \cdot \|H\|_\infty < 2^{n/2}\alpha^{n^2}n^{n^2+2n+3}.$$

Thus, the absolute values of the coefficients of $\det(H)g_{ij} \in \mathbb{Z}[x]$ are less than $2^{n/2}\alpha^{n^2}n^{n^2+2n+3}$, and this same upper bound holds for the numerators of the coefficients of $g_{ij}$. The denominators of the coefficients of $g_{ij}$ divide $\det(H)$ and so have absolute value less than $\delta$, for $1 \le j < i \le k$.

In step (4) of `FrobeniusForm` we determine if $f_i \mid g_{ij}$ for $1 \le j < i \le k$. This is accomplished by computing the division with remainder of $g_{ij}$ by $f_i$ to obtain $h_{ij}, t_{ij} \in \mathbb{Q}[x]$ such that $g_{ij} = h_{ij}f_i + t_{ij}$ and $\deg t_{ij} < \deg f_i$. We can then check to make sure $t_{ij} = 0$. Since $f_i$ and $\bar{g}_{ij} = \det(H)g_{ij}$ are both in $\mathbb{Z}[x]$, we can bound the numerators of coefficients of $h_{ij}$ and $t_{ij}$ by considering the division with remainder of $\bar{g}_{ij}$ by $f_i$: find $\bar{t}_{ij}, \bar{h}_{ij} \in \mathbb{Z}[x]$ such that $\bar{g}_{ij} = \bar{h}_{ij}f_i + \bar{t}_{ij}$, then $h_{ij} = \bar{h}_{ij}/\det(H)$ and $t_{ij} = \bar{t}_{ij}/\det(H)$.

To bound the sizes of the quotients and remainders we apply Theorem 1.5:

$$\begin{aligned}\|\bar{h}_{ij}\|_\infty \le& 2^{n-d_i} \cdot \left(2^n e^{n/2}\alpha^n n^{n/2}\right)^{n-d_i} \cdot \left(2^{n/2}n^{n^2+2n+3}\alpha^{n^2}\right)\\ \le& 2^{n^2+n/2-1}e^{(n^2-n)/2}\alpha^{2n^2-n}n^{3n^2/2+3n/2+3},\end{aligned}$$

and

$$\begin{aligned}\|\bar{t}_{ij}\|_\infty \le& 2^{n-d_i+1} \cdot \left(2^n e^{n/2}\alpha^n n^{n/2}\right)^{n-d_i+1} \cdot \left(2^{n/2}n^{n^2+2n+3}\alpha^{n^2}\right)\\ \le& 2^{n^2+3n/2}e^{n^2/2}\alpha^{2n^2}n^{3n^2/2+2n+3}.\end{aligned}$$

These bounds on $\|\bar{t}_{ij}\|_\infty$ and $\|\bar{h}_{ij}\|_\infty$ are also bounds on the maximum absolute values of the numerators of the coefficients of $t_{ij}$ and $h_{ij}$ respectively. The denominators of all coefficients of $t_{ij}$ and $h_{ij}$ are divisors of $\det(H)$, and hence their absolute values are bounded by $\delta$.

In step (6) of `FrobeniusForm` we compute $v_1, \ldots, v_k \in \mathbb{Q}^{n\times 1}$, where $v_i = w_i - \sum_{1\le j<i} h_{ij}w_j$ for $1 \le i \le k$. Now

$$\begin{aligned}\|(\det(H)h_{ij}) \circ w_j\|_\infty \le& \deg(h_{ij}) \cdot \|\det(H)h_{ij}\|_\infty \cdot \alpha^{\deg h_{ij}} n^{\deg h_{ij}} n^2 \\ \le& 2^{n^2+n/2-1} e^{(n^2-n)/2} \alpha^{2n^2} n^{3n^2/2+5n/2+6},\end{aligned}$$

whence

$$\begin{aligned}\|\det(H)v_i\|_\infty \le& \|w_i\|_\infty + \sum_{1\le j<i} \|(\det(H)h_{ij}) \circ w_j\|_\infty \\ \le& n^2 + (i-1) \cdot 2^{n^2+3n/2} e^{n^2/2} \alpha^{2n^2} n^{3n^2/2+2n+3} \\ <& 2^{n^2+3n/2} e^{n^2/2} \alpha^{2n^2} n^{3n^2/2+2n+4}.\end{aligned}$$

Thus, the absolute values of the numerators of all coefficients of $v_i$ are less than $2^{n^2+3n/2} e^{n^2/2} \alpha^{2n^2} n^{3n^2/2+2n+4}$, while the denominators all divide $\det(H)$ and hence their absolute values are less than $\delta$.

Finally, the absolute values of the numerators of coefficients of $U$ are less than

$$\max\{\|T^j v_i\|_\infty : 1 \le i \le k,\ 0 \le j < d_i\} < 2^{n^2+3n/2} e^{n^2/2} \alpha^{2n^2+n-1} n^{3n^2/2+3n+3},$$

and the denominators divide $\det(H)$ and have absolute values less than $2^{n/2}\alpha^{n^2-n} n^{n^2+n}$.

**Theorem 4.3.** *Let $T \in \mathbb{Z}^{n\times n}$ and consider the execution of* `FrobeniusForm` *on input $T$, assuming all rational numbers are represented by pairs of relatively prime integers (a numerator and denominator). The maximum absolute value of a numerator of any rational number encountered at the input or output to any step is less than*

$$\gamma = 2^{n^2+3n/2} e^{n^2/2} \alpha^{2n^2+n-1} n^{3n^2/2+3n+3},$$

*where $\alpha = \|T\|_\infty$. All denominators of rational numbers used during the execution are divisors of $\det(H)$ and have absolute value less than $\delta = 2^{n/2}\alpha^{n^2-n} n^{n^2+n}$.*

## 4.2 Modular Algorithms for the Frobenius Form

In the previous section, we show in Theorem 4.3 that if we choose the vectors $w_1, \ldots, w_n$ randomly from $L^{n\times 1}$, where $L = \{0, \ldots, n^2 - 1\}$, then intermediate values in the execution of `FrobeniusForm` do not get excessively large. From this, we might hope for an algorithm which requires $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot \mathrm{M}(\log\gamma))$ bit operations, where $\gamma$ is as in Theorem 4.3, and we represent rational numbers by two relatively prime integers (a numerator and a denominator). At least in a naïve implementation, this estimate of the cost also makes the assumption that intermediate values in the computation of *each step* is at most a constant times bigger than the size of the input and output to that step. This is probably reasonable using standard algorithms for matrix and polynomial arithmetic, but is less clear for asymptotically faster algorithms. Even if this assumption is true, using standard arithmetic, the algorithm `FrobeniusForm` requires $O\tilde{\ }(n^7 \log^2 \alpha)$ bit operations on input $T \in \mathbb{Z}^{n\times n}$, with $\alpha = \|T\|_\infty$ using this approach.

### A Modular Las Vegas Algorithm for the Frobenius Form

We can improve quite dramatically on an implementation of `FrobeniusForm` with arbitrary length integer arithmetic by the careful use of modular techniques (see Section 1.3). Our approach is to compute the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of $T$ modulo a set of small primes; for each prime $p$ we compute the Frobenius form $S^{(p)} \in \mathbb{Z}_p^{n\times n}$ of $T^{(p)} = (T \bmod p)$. A possible hazard in this technique is that for some of these primes $S^{(p)} \neq S \bmod p$. To solve this problem, we examine the degrees $(d_1^{(p)}, \ldots, d_{k_p}^{(p)}) \in \mathbb{Z}^{k_p}$ of the invariant factors of $T^{(p)}$. We show that if there is a set of primes $\Lambda$, such that the product of all primes in $\Lambda$ is at least $2\delta$, and such that for all primes $p, p' \in \Lambda$ we have $(d_1^{(p)}, \ldots, d_{k_p}^{(p)}) = (d_1^{(p')}, \ldots, d_{k_{p'}}^{(p')})$, then for all $p \in \Lambda$ we have $S^{(p)} = S \bmod p$. Once we have found such a set $\Lambda$ and computed the Frobenius form $S^{(p)}$ of $T^{(p)}$ for each $p \in \Lambda$, we use the Chinese remainder theorem to recover $S \in \mathbb{Z}^{n\times n}$ from $S^{(p)}$ for all $p \in \Lambda$.

We show now that the number of primes modulo which the degrees of the invariant factors do change is in fact quite small, and that when these degrees do not change modulo a prime $p$, that $S^{(p)} = S \bmod p$. Once again, let $L = \{0, \ldots, n^2 - 1\}$. In

Theorem 2.5 we show that there exist vectors $w_1, \ldots, w_k \in L^{n\times 1}$ which generate a modular cyclic basis for $V$. For $1 \le i \le k$, let $V_i = \mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$ and $T_i\colon V/V_i \to V/V_i$ as $T$ reduced modulo $V_i$ for $1 \le i \le k$ (with $V_0 = \{0\}$ and $T_0 = T$). Then $V = V_1 + \cdots + V_k$ and $f_i = \min(T_{i-1}; w_i + V_{i-1}) = \min(T_{i-1}; V/V_{i-1})$ is the $i$th invariant factor of $T$. Corresponding to this decomposition of $V$ is the matrix $H = [H_1|\cdots|H_n] \in \mathbb{Z}^{n\times n}$, as computed in Theorem 2.7, where $H_i \in \mathbb{Z}^{n\times d_i}$ satisfy (for $1 \le i \le k$)

(i) $d_i$ is the degree of the $i$th invariant factor of $T$ and $\sum_{1\le i\le n} d_i = n$;

(ii) for $1 \le i \le n$, $H_i = [w_i \mid Tw_i \mid \cdots \mid T^{d_i-1}w_i]$;

(iii) the columns of $H_1, \ldots, H_i$ form a basis of $\mathrm{Orb}(T; w_1) + \cdots + \mathrm{Orb}(T; w_i)$.

$H$ is a modular cyclic transition matrix for $T$. We call $(d_1, \ldots, d_k)$ the *degree sequence* of $T$.

Consider what happens to this modular cyclic basis modulo a prime $p$. Let $T^{(p)} = T \bmod p$, and $w_i^{(p)} = w_i \bmod p$ for $1 \le i \le k$. The vector space $V_1^{(p)} = \mathrm{Orb}(T^{(p)}; w_1^{(p)})$ has dimension $d_1^{(p)}$ which is at most $d_1$, since if integer vectors are linearly dependent over $\mathbb{Q}$ then they are certainly linearly dependent modulo $p$. Now let $V_i^{(p)} = \mathrm{Orb}(T^{(p)}; w_1^{(p)}) + \cdots + \mathrm{Orb}(T^{(p)}; w_i^{(p)})$ for $1 \le i \le k$. If $\dim V_{i-1}^{(p)} = \dim V_{i-1}$ for some $i \ge 2$, it similarly follows that $d_i^{(p)} = \dim V_i^{(p)} \le \dim V_i$. Moreover, if $p \nmid \det(H)$, then $(d_1, \ldots, d_k) = (d_1^{(p)}, \ldots, d^{(p_k)})$ since $H$ is non-singular modulo $p$ and $H^{(p)} = H \bmod p$ is a modular cyclic transition matrix for $T^{(p)}$. Thus

$$\min(T^{(p)} \bmod V_{i-1}; w_i^{(p)} + V_{i-1}^{(p)}) = \min(T \bmod V_{i-1}; w_i + V_{i-1}) \bmod p = f_i \bmod p$$

for $1 \le i \le k$ and $S^{(p)} = S \bmod p$. We have shown the following:

**Lemma 4.4.** *Let $T \in \mathbb{Z}^{n\times n}$ with a modular cyclic transition matrix $H \in \mathbb{Z}^{n\times n}$ and Frobenius form $S \in \mathbb{Z}^{n\times n}$.*

*(i) If for any prime $p$, the degree sequence of $T \bmod p$ equals that of $T$, then the Frobenius form of $T \bmod p$ over $\mathbb{Z}_p$ is $S \bmod p$.*

*(ii) If $p \nmid \det(H)$, then the Frobenius form of $T^{(p)} = T \bmod p$ is $S \bmod p$.*

Primes not dividing the determinant of a particular integral modular transition matrix are designated as *good*. Conversely, if $p \mid \det(H)$, it could be the case that there exists an $l \leq k$ such that for $1 \leq i < l$ we have $d_i^{(p)} = d_i$ while $d_l^{(p)} < d_l$. In particular, when $p \mid \det(H)$ the degree sequence of $T^{(p)}$ may be lexicographically smaller than the degree sequence of $T$. We designate such primes as *bad* and note that the product of all bad primes is at most $|\det(H)| < \delta$.

We compute the Frobenius form modulo primes $p_1, \ldots, p_r$, with $2^{\ell-1} < p_i < 2^\ell$ for $1 \leq i \leq r$, where $\ell = 6 + \log\log\delta$. By Lemma 1.8 we know that the number of primes $p$ with $2^{\ell-1} < p < 2^\ell$ is at least $2\lceil\lceil\log_2(2\delta)\rceil/(\ell-1)\rceil$, which is at least twice as large as the maximum number of bad primes between $2^{\ell-1}$ and $2^\ell$. Thus, we assign $r = 2\lceil\lceil\log_2(2\delta)\rceil/(\ell-1)\rceil$ primes, thereby ensuring that there are enough good ones. Modulo good primes, $T$ has a maximal degree sequence $\bar{d} \in \mathbb{N}^k$ under the lexicographic ordering of degree sequences, and this must in fact be the degree sequence of $T$ since we are guaranteed to have encountered at least one good prime. We throw out all primes $p$ modulo which $T$ has a lexicographically smaller degree sequence than $\bar{d}$. This leaves us with a set of good primes $\Lambda \subseteq \{p_1, \ldots, p_r\}$ with product at least $2\delta$, as well as $S^{(p)} \in \mathbb{Z}_p^{n\times n}$ for each $p \in \Lambda$. Since $\delta > \beta$ (where $\beta$ is as in Lemma 4.2 and bounds the infinity norms of the invariant factors of $T$), we can recover $S$ from its residues modulo the primes in $\Lambda$ by Chinese remaindering.

We now analyze the costs involved in this algorithm. To begin with, we must find $T \bmod p$ for $r = O(n^2(\log n + \log\alpha)/\ell)$ primes $p$. By Lemma 1.6 this requires $O(n^2\mathrm{M}(n^2(\log n + \log\alpha))\ell)$ bit operations asymptotically (note that $\ell = O(\log r)$). Using standard arithmetic, $O(n^2 r \log(\alpha)\ell)$ or $O(n^4(\log n + \log\alpha)\log\alpha)$ bit operations are sufficient. We then find the Frobenius form of $T$ modulo each of these primes $p$ using the algorithm `FrobeniusForm` over $\mathbb{Z}_p$. This algorithm works best for primes $p \geq n^2$, and since we use only primes $p > 2^{\ell-1}$, we insist that $2^{\ell-1} \geq n^2$ or $\ell \geq 1 + 2\log_2 n$. For a single prime $p_i$, with $1 \leq i \leq r$, we can compute $S^{(p_i)}$ with $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot \mathrm{M}(\ell))$ bit operations by Theorem 2.12, or $O(n^3\ell^2))$ bit operations with standard arithmetic by Theorem 3.5. To recover $S$ from its residues we use Chinese remaindering. We need only choose a subset of $\Lambda$ with product at least $2\beta = O(n(\log n + \log\alpha)/\ell)$ primes and recover $S$ modulo these primes. By Lemma 1.7 this requires $O(n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ bit operations, or, with standard arithmetic, $O(n^4(\log n + \log\alpha)^2)$ bit operations.

**Theorem 4.5.** *Let $T \in \mathbb{Z}^{n\times n}$ with $\alpha = \|T\|_\infty$. The Las Vegas algorithm discussed above to compute the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of $T$ requires an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot n^2(\log n + \log\alpha) \cdot \mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ or $O\tilde{~}(\mathrm{MM}(n) \cdot n^2 \log\alpha)$ bit operations. Using standard arithmetic, an expected number of $O(n^5(\log n + \log\alpha)\ell + n^4\log^2\alpha)$ bit operations are required. Here $\ell = \Theta(\log n + \log\log\alpha)$.*

**Computing Transition Matrices Efficiently**

In the above construction we did not find a transition matrix $U \in \mathbb{Q}^{n\times n}$ such that $U^{-1}TU = S$. The reason for this is that each entry in $U$ can be very large: since we have computed each transition matrix $U^{(p)}$ (for each $p \in \Lambda$) independently, that is, we picked new $w_1^{(p)}, \ldots, w_k^{(p)} \in \mathbb{Z}_p^{n\times 1}$ for each $p$, we have no good bound on the size of entries in $U$. However, once we know the Frobenius form $S$ of $T$, we can construct a modular cyclic transition matrix $H \in \mathbb{Z}^{n\times n}$ with relatively small entries without possibility of error. We can then proceed to construct a transition matrix $U$, whose entries require $O(n^2(\log n + \log\alpha))$ bits each.

First, assume that we have computed the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of $T$, and that $T$ has degree sequence $\bar{d} = (d_1, \ldots, d_k)$. Let $L = \{0, \ldots, n^2 - 1\}$ and choose vectors $w_1, \ldots, w_k \in L^{n\times 1}$ randomly. By Theorem 2.5 these generate a modular cyclic decomposition of $V$ with probability at least $1/4$. Let $H = [H_1|\cdots|H_n] \in \mathbb{Z}^{n\times n}$, where $H_i = [w_i|Tw_i|\cdots|T^{d_i-1}w_i] \in \mathbb{Z}^{n\times d_i}$ for $1 \le i \le k$. The matrix $H$ is non-singular if and only if $w_1, \ldots, w_k$ generate a modular cyclic basis for $V$. Assuming $H$ is non-singular, there corresponds a unique transition matrix $U \in \mathbb{Q}^{n\times n}$ such that $S = U^{-1}TU$. This $U$ is simply the matrix constructed by the algorithm `FrobeniusForm` in steps (2)-(8), when run over the rationals (these steps are completely deterministic). All numerators of entries in $U$ have absolute value less than $\gamma$, while denominators are all divisors of $\det(H)$, as shown in Theorem 4.3.

We now construct a set of good primes $\Lambda_0 \subseteq \mathbb{N}$ for this algorithm, though we postpone calculation of the size and number of primes required until later. For a prime $p \in \mathbb{N}$, first compute $H \bmod p$. If $H$ is non-singular modulo $p$, add $p$ to $\Lambda_0$ and compute $U^{(p)} \in \mathbb{Z}_p^{n\times n}$ such that $(U^{(p)})^{-1}T^{(p)}U^{(p)} = S \bmod p$. This can be done with an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot \mathrm{M}(\ell))$ bit operations using Theorem 2.12, or $O(n^3\ell^2)$ bit operations using Theorem 3.5 with standard arithmetic. When $p \nmid \det(H)$ we know that $S^{(p)} = S \bmod p$ by Lemma 4.4. The entries of $U$ are rational functions in the entries of $H$ and $T$ (from the algorithm `FrobeniusForm`) and the denominators in $U$ are all divisors of $\det(H)$ and hence non-zero modulo $p$. Thus $U^{(p)} \equiv U \bmod p$. Rational numbers whose numerators have absolute value less than $\gamma$ and whose denominators have absolute value less than $\delta$ can be represented uniquely modulo a set of primes with product at least $2\gamma\delta$. It is easy to see that $\lceil\lceil\log_2(2\gamma\delta)\rceil/(\ell-1)\rceil$ good primes between $2^\ell$ and $2^{\ell-1}$ suffices. By Lemma 1.8, if we let $\ell = 6+\log\log(\gamma\delta)$ there are at least $2\lceil\lceil\log_2(2\gamma\delta)\rceil/(\ell-1)\rceil$ primes in this range, and at least half of them are good (i.e., they don't divide $\det(H)$).

Since $\log(2\gamma\delta) = O(n^2(\log n+\log\alpha))$, the total cost of finding a set $\Lambda_0$ of good primes $p$ and matrices $U^{(p)} \in \mathbb{Z}_p^{n\times n}$, such that $\prod_{p\in\Lambda_0} p \geq 2\gamma\delta$, and for each $p \in \Lambda_0$ we have $U^{(p)} = U \bmod p$, is $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot n^2(\log n+\log\alpha)\cdot \mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n(\log n+\log\alpha))\ell)$ bit operations, or $O(n^5(\log n+\log\alpha)\ell + n^4\log^2\alpha)$ bit operations using standard arithmetic. To actually recover each entry of $U$ as an integer from its residues modulo the primes in $\Lambda_0$ is more expensive: by Lemma 1.7 it requires $O(n^2\mathrm{M}(n^2(\log n+\log\alpha))(\log n+\log\alpha))$ bit operations, or $O(n^6(\log n+\log\alpha)^2)$ bit operations using standard integer arithmetic.

**Theorem 4.6.** *Let $T \in \mathbb{Z}^{n\times n}$ with $\alpha = \|T\|_\infty$. The Las Vegas algorithm discussed above to compute the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of $T$, and a transition matrix $U \in \mathbb{Q}^{n\times n}$ such that $S = U^{-1}TU$ requires an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot n^2(\log n+\log\alpha)\cdot\mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n(\log n+\log\alpha))\ell)$ or $O\tilde{}(\mathrm{MM}(n)\cdot n^2\log\alpha)$ bit operations. Using standard arithmetic, an expected number of $O(n^5(\log n+\log\alpha)\ell + n^4\log^2\alpha)$ bit operations are required. In the standard arithmetic case, entries in $U$ are represented modulo a collection of $O(n^2(\log n+\log\alpha)/\ell)$ primes between $2^{\ell-1}$ and $2^\ell$. To reconstruct $U \in \mathbb{Q}^{n\times n}$ using standard arithmetic requires an additional $O(n^6(\log n+\log\alpha)^2)$ bit operations. Here $\ell = \Theta(\log n+\log\log\alpha)$.*

In some applications of Theorem 4.6 (see Theorem 5.28 below) it is useful to include even more primes in our collection $\Lambda_0$, though we never require more than $O(n^2(\log n+\log\alpha)/\ell)$ of them. We note now for future reference that the technique used above to find enough good primes to correctly represent the entries of $U$ can be used to find any number of good primes, provided of course that $\ell$ is made appropriately large. In particular, if $\xi \geq \gamma\delta$, and $\xi = O(n^2(\log n + \log \alpha)/\ell)$, we can choose $\ell = 6 + \log\log\xi$. By Lemma 1.8, this ensures that there are good primes $p$ with $2^{\ell-1} < p < 2^\ell$ with product at least $2\xi$.

**Corollary 4.7.** *Let $\xi = O(n^2(\log n + \log \alpha))$. Then we can compute $U$ as in Theorem 4.6 modulo a collection of primes $\Lambda_0$ (generated by the algorithm), such that for each $p \in \Lambda_0$ we have $2^{\ell-1} < p < 2^\ell$, and such that the product of all primes in $\Lambda_0$ is at least $2\xi$. The number of bit operations required is of the same order as in Theorem 4.6, where $\ell = \Theta(\log n + \log\log \alpha)$.*

## A Modular Monte Carlo Algorithm for the Frobenius Form

In the modular Las Vegas algorithm to compute the Frobenius form $S$ of $T \in \mathbb{Z}^{n\times n}$ we needed $\lceil\lceil\log_2(2\delta)\rceil/(\ell-1)\rceil = O(n^2(\log n + \log \alpha)/\ell)$ primes between $2^{\ell-1}$ and $2^\ell$ to ensure that we correctly identified the Frobenius form of $T$. However, to actually recover the Frobenius form from its residues requires only $s = \lceil\lceil\log_2(2\beta)\rceil/(\ell-1)\rceil = O(n(\log n + \log\alpha)/\ell)$ primes of this size. In fact, if we assume that $\ell = 6 + \log\log\delta$, and randomly choose primes $p$ with $2^{\ell-1} < p < 2^\ell$, the probability that all of them divide $\det(H)$ is less than $1/2^s$, since fewer than half the primes are bad.

Choose $s$ random primes $p$ such that $2^{\ell-1} < p < 2^\ell$, and compute $S^{(p)} \in \mathbb{Z}_p^{n\times n}$, the Frobenius form of $T^{(p)} = T \bmod p$: suppose $T^{(p)}$ has degree sequence $\bar{d}^{(p)}$. Once this is completed, let $\bar{d}$ be the lexicographically maximum degree sequence found. Unless all primes chosen divide $\det(H)$, and possibly even then, $\bar{d}$ is the degree sequence of $T$. Now randomly choose primes $p$ with $2^{\ell-1} < p < 2^\ell$ and compute the Frobenius form $S^{(p)}$ of $T^{(p)}$. If $T^{(p)}$ has degree sequence $\bar{d}$, add $p$ to the set $\Lambda$ of good primes. Continue this process until $s$ primes are in $\Lambda$. $S$ can be uniquely lifted from all the $S^{(p)}$ for $p \in \Lambda$, and assuming that we have correctly determined $\bar{d}$, $S$ is correct.

**Theorem 4.8.** *Let $T \in \mathbb{Z}^{n\times n}$ with $\alpha = \|T\|_\infty$. The Monte Carlo algorithm discussed above to compute the Frobenius form $S \in \mathbb{Z}^{n\times n}$ of $T$ requires an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot n(\log n + \log\alpha) \cdot \mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n^2(\log n + \log\alpha))\ell)$ or $O\tilde{~}(n^4 \log\alpha)$ bit operations. Using standard arithmetic, an expected number of*

$O(n^4(\log n + \log \alpha)^2)$ *bit operations are sufficient. The probability of error is at most* $1/2^s$, *where* $s = \Omega(n(\log n + \log \alpha)/\ell)$. *Here* $\ell = \Theta(\log n + \log \log \alpha)$.

It is suspected that the probability of failure is much lower than is stated in this theorem, and may in fact be zero. Notice that in this case both the asymptotic and practical costs are quartic in $n$; the asymptotic cost is not $O\tilde{}(n\mathrm{MM}(n))$. This is an artifact of our choice of algorithms for modular reduction: the asymptotic algorithm is not sensitive to the size of the integer being reduced, only the number of primes in the collection.

# Chapter 5.

# Applications of the Frobenius Form

In this chapter we consider a number of applications of computing the Frobenius form of a matrix. Chief among these is evaluating a polynomial at a matrix in nearly optimal time. In Section 5.1 we give a Las Vegas algorithm which, for any $g \in \mathsf{K}[x]$ of degree $r$, evaluates $g(T) \in \mathsf{K}^{n\times n}$ with $O\tilde{~}(\mathrm{MM}(n) + r)$ operations in $\mathsf{K}$. The main idea is to make use of the Frobenius form as a transformation from the multiplicative semi-group of matrices, where multiplication is expensive, to the multiplicative semi-group of polynomials, where multiplication can be computed in almost linear time. The idea of evaluating matrices at polynomials using canonical forms and polynomial arithmetic is not new, and dates at least as far back as (Ranum, 1911). The best previously known computational method of Paterson & Stockmeyer (1973) and Brent & Kung (1978) requires $O(\mathrm{MM}(n)\sqrt{r})$ operations in $\mathsf{K}$. Our technique is also applied to give fast algorithms for evaluating high powers of matrices, and more generally to the problem of evaluating arithmetic circuits, or equivalently straight-line programs, at matrices.

The cost of our algorithm to evaluate a non-linear polynomial at a matrix matches (up to a poly-logarithmic factor in its cost and the use of randomness) a lower bound for evaluating a polynomial at a matrix which we also prove in Section 5.1. Specifically we show, for sufficiently large fields $\mathsf{K}$, that if we can evaluate some fixed non-linear polynomial $g \in \mathsf{K}[x]$ at any matrix in $\mathsf{K}^{n\times n}$ with at most $t(n)$ operations in $\mathsf{K}$, then we can multiply any two matrices in $\mathsf{K}^{n\times n}$ with $O(t(n))$ operations in $\mathsf{K}$. A similar result holds for smaller fields, with an additional poly-logarithmic factor in $n$ in the running time. While it seems clear that matrix multiplication is required to evaluate a non-linear polynomial at a matrix, the proof is in fact not completely trivial.

Fast and practical computation of the Frobenius form also has consequences for practical evaluation of polynomials at matrices. Using asymptotically slower but practical "school" algorithms for matrix and polynomial arithmetic, our technique affords a Las Vegas algorithm to evaluate $g(T)$ with $O(n^3 + r^2)$ operation in $\mathsf{K}$ when $\#\mathsf{K} \geq n^2$, or $O(n^3 \log_q n + r^2)$ in $\mathsf{K}$ when $q = \#\mathsf{K} < n^2$.

We also examine the cost of evaluating an integer polynomial at an integer matrix. The algorithms we propose work modulo a set of small primes and use Chinese remaindering to recover the solution. If $T \in \mathbb{Z}^{n\times n}$ with $\alpha = \|T\|_\infty$, and $g \in \mathbb{Z}[x]$ has $\|g\|_\infty = \rho$, then we can compute $g(T)$ by a Las Vegas algorithm with an expected number of $O\tilde{~}(\mathrm{MM}(n) \cdot (\log \rho + r \log \alpha) + n^2(\log \rho + r \log \alpha) \log(\alpha))$ bit operations. Using standard integer, matrix, and polynomial arithmetic, this algorithm requires an expected number of $O((n^3+r^2)\cdot(\log\rho+r(\log n+\log\alpha))\ell+n^2(\log\rho+r(\log n+\log\alpha))^2)$ bit operations, where $\ell = \Theta(\log(\log \rho + r \log n + r \log \alpha))$

Two other quite straightforward applications of finding the Frobenius form $S \in \mathsf{K}^{n\times n}$ of $T \in \mathsf{K}^{n\times n}$ are finding the minimal polynomial of a matrix and testing matrix similarity. These are presented in Section 5.2. These are essentially just observations, but are worthy of mention because they are nearly optimal solutions to classical problems. Given $S$ we can immediately determine the minimal polynomial of $T$ and, indeed, all invariant factors of $T$. Our Las Vegas algorithm improves on the Monte Carlo probabilistic algorithm of Wiedemann (1986), which requires $O(n^3)$ operations in $\mathsf{K}$. We also show that finding the minimal polynomial of a matrix $T \in \mathsf{K}^{n\times n}$ is as hard multiplying two $n \times n$ matrices, at least up to randomness. Specifically, we give a Las Vegas type reduction from the problem of computing the determinant of $T$ to that of finding the minimal polynomial of a single matrix in $\mathsf{K}^{n\times n}$. Aside from computing this minimal polynomial, $O(n\mathrm{M}(n))$ additional operations in $\mathsf{K}$ are sufficient. That computing the determinant is as hard as multiplying matrices is demonstrated by Baur & Strassen (1982); see Theorem 1.3 of this thesis.

The cost of computing the minimal polynomial of an integer matrix is certainly no more than that of computing its Frobenius form. In fact, we do somewhat better than our algorithm for the Frobenius form of an integer matrix: we give a Las Vegas algorithm which requires an expected $O\tilde{~}(\mathrm{MM}(n) \cdot n \log \alpha)$ bit operations. This is about the same asymptotically as the cost of our Monte Carlo algorithm for computing the

Frobenius form of an integer matrix. This probably reflects more on our understanding of how to compute the Frobenius form than on any fundamental difference in the complexity of these problems. Using standard integer, polynomial, and matrix arithmetic, our algorithm requires an expected number of $O(n^4(\log n + \log \alpha)^2)$ bit operations.

Determining if two matrices are similar is certainly a classical problem, one which is made trivial given the ability to find the Frobenius form. This problem has also been looked at by Zalcstein & Garzon (1987), where an $NC^2$ algorithm is presented, and by Kaltofen *et al.* (1986, 1990), where an $RNC^2$ algorithm is given. Neither of these algorithms lead immediately to an efficient sequential algorithm. In Section 5.2 we also give an algorithm which determines if two matrices $T, A \in \mathsf{K}^{n\times n}$ are similar, and if so, returns a transition matrix $X$ such that $X^{-1}TX = A$. This algorithm is of the Las Vegas type and requires $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ operations in $\mathsf{K}$. It succeeds with probability at least 1/4.

In Section 5.3 we present an algorithm to find the rational Jordan form of a matrix $T \in \mathsf{K}^{n\times n}$. This canonical form is a generalization of the usual Jordan canonical form. Whereas a Jordan form of $T$ exists only if $\mathsf{K}$ contains all eigenvalues of $T$, the rational Jordan form always exists and lies in $\mathsf{K}^{n\times n}$. The eigenvalues of $T$, found along the diagonal of the usual Jordan form, are replaced in the rational Jordan form by the companion matrices of their minimal polynomials in $\mathsf{K}[x]$. If $\mathsf{K}$ contains all the eigenvalues of $T$, then the rational Jordan form equals the usual Jordan form. A straightforward reduction is given from finding the rational Jordan form to finding the Frobenius form and factoring the minimal polynomial. This reduction requires $O(n^2)$ operations in $\mathsf{K}$ and a constant number of calls to the routine for computing Frobenius forms of $n \times n$ matrices over $\mathsf{K}$. There is also a reduction in the other direction. Given the rational Jordan form of a matrix we can find the Frobenius form with $O(\mathrm{M}(n))$ operations in $\mathsf{K}$. Also, from the rational Jordan form of the companion matrix of a polynomial we can read off the complete factorization of that polynomial. Thus, we cannot expect to eliminate the requirement for factoring polynomials in any algorithm to compute the rational Jordan form.

When $\mathsf{K}$ is a finite field with $q$ elements, we can use Berlekamp's(1970) polynomial factoring algorithm along with Theorem 2.12 to obtain a Las Vegas algorithm to find the rational Jordan form of any $T \in \mathsf{K}^{n\times n}$ with $O\tilde{~}(\mathrm{MM}(n) + n \log q)$ operations in $\mathsf{K}$.

The Jordan form is also (possibly principally) of interest for rational and integer matrices. Numerically this problem is highly unstable. All $n \times n$ rational matrices, with the exception of a set of measure zero, have distinct eigenvalues and hence their Jordan form consists simply of the companion blocks of these eigenvalues along the diagonal. Any slight perturbation of a matrix (such as that incurred by rounding error when entries are represented by floating point numbers) will yield a matrix with possibly completely different Jordan block structure, usually with a single companion block. Some of these questions are addressed by Demmel (1983), where he attempts to classify possible Jordan forms which can occur with "uncertainly" specified data. Numerical algorithms for approximating the eigenvalues and eigenvectors of a real or complex matrix are plentiful. Most of these start from a Schur decomposition of the matrix (which is numerically stable but certainly not unique); see (Kågström & Ruhe, 1980a & 1980b), (Golub & Wilkinson, 1976), and (Bavely & Stewart, 1979) for algorithms, and (Golub & Van Loan, 1989) and (Wilkinson, 1965) for an overview. Because of this inherent instability of the problem, we concern ourselves only with exact arithmetic in $\mathbb{Z}$. The main problem in this paradigm of exact computation is not instability (as it is usually defined at least), but rather the sizes of the integers involved in the computation and the cost of multiplying and dividing them.

Computationally, it has been rather impractical to compute the rational Jordan form exactly for two reasons. Firstly, the minimal (or characteristic) polynomial of the matrix must be factored completely. Secondly, the sizes of the entries in the transition matrix generated by previously known algorithms are rather large. Algorithms for factoring polynomials in $\mathbb{Z}[x]$ have improved considerably over the years, and we make use of the algorithm of Schönhage (1984) to factor the minimal polynomial. Unlike other algorithms known for finding the rational Jordan form, factoring the minimal polynomial proves to be the bottleneck in our algorithm. We also address the issue of the size of the entries in the generated transition matrix to the rational Jordan form. Our algorithm generates a transition matrix to the Jordan form whose entries have the same order of length as our upper bounds on the length of the entries in the transition matrix to the Frobenius form. We have no reason to believe that a transition matrix with smaller entries should exist. Given a matrix $T$ with $\|T\|_\infty = \alpha$, we exhibit a Las Vegas type algorithm which requires an expected number of $O\tilde{}(\mathrm{MM}(n) \cdot n^2 \log \alpha)$ bit operations to find the rational Jordan form of $T$, given a complete factorization of the minimal polynomial $f_1 \in \mathbb{Z}[x]$ of $T$. Using standard integer, matrix, and polynomial arithmetic, our algorithm requires an expected number of $O(n^5(\log n + \log \alpha)\ell + n^4 \log^2 \alpha)$ bit operations, where $\ell = \Theta(\log n + \log\log \alpha)$. We also exhibit a Monte Carlo type algorithm which requires an expected number of $O\tilde{}(\mathrm{MM}(n) \cdot n \log \alpha)$ bit operations, but has a probability of at most $1/2^s$ of returning an incorrect answer, where $s = \Omega(n(\log n + \log \alpha)/\ell)$. Using standard arithmetic this Monte Carlo algorithm requires only an expected number of $O(n^4(\log n + \log \alpha)^2)$ bit operations. The best previously known algorithm for this problem is due to Gil (1992), and requires $O(n^{11}(\log n + \log \alpha)^2)$ bit operations.

The cost of factoring the minimal polynomial $f_1$ of $T$, using Schönhage's(1984) algorithm is $O(n^{6+\epsilon}(\log n + \alpha)^{2+\epsilon})$ for any $\epsilon > 0$ (using fast integer and polynomial arithmetic) which dominates the cost of our algorithm for finding the Frobenius form and a transition matrix given the factorization of $f_1$. Using standard arithmetic, Schönhage's algorithm requires $O(n^8(\log n + \log \alpha)^3)$ bit operations to factor $f_1$ and is clearly the bottleneck in finding the Frobenius form using standard arithmetic.

## 5.1 Evaluating Polynomials at Matrices

We now consider an application of the Frobenius form to evaluating polynomials at matrices. A new algorithm requiring almost optimal time is presented. This technique is generalized to evaluating a straight-line program at a matrix in almost optimal time as well. We consider the problem both for matrices over an abstract field $\mathsf{K}$, and for matrices over $\mathbb{Q}$.

We begin by looking at the evaluation of a polynomial in $\mathsf{K}[x]$ at a matrix in $\mathsf{K}^{n\times n}$. For most of this section, we assume that $\mathsf{K}$ is any field with at least $n^2$ elements. For fields $\mathsf{K}$ with fewer than $n^2$ elements similar results hold with running times multiplied by a small power of $\log_q n$. These results for smaller fields are summarized in Theorem 5.5 below. We also give a lower bound for evaluating a polynomial at matrix which matches the upper bound within a poly-logarithmic factor. Finally, we consider the bit cost of evaluating an integer polynomial at an integer matrix.

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 $g \in \mathsf{K}[x]$. Informally, it allows us to transform the problem into that of polynomial multiplication, for which fast algorithms are known. 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 $\mathfrak{R}$ of $\mathsf{K}$, which requires $O(\sqrt{r})$ non-scalar multiplications in $\mathfrak{R}$, 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 $\mathsf{K}$ (see also Brent & Kung 1978).

When $\deg g \geq n$, Paterson & Stockmeyer's algorithm can be improved considerably by reducing $g$ modulo a multiple $f \in \mathsf{K}[x]$ of the minimal polynomial of $T$. For example if $g = Qf + h$ for some $Q, h \in \mathsf{K}[x]$ and $\deg h < \deg g$, then $g(T) = h(T)$ because $f(T) = 0$. Since we can (deterministically) compute the characteristic polynomial of $T$ with $O(\mathrm{MM}(n)\log n)$ operations in $\mathsf{K}$ using Keller-Gehrig's (1985) algorithm, the characteristic polynomial is a a natural candidate for our multiple $f$ of the minimal polynomial. This yields a deterministic algorithm for evaluating $g(T)$ with $O(\mathrm{MM}(n)\sqrt{n} + n^3 + \mathrm{M}(r))$ or $O\tilde{\ }(n^3 + r)$ operations in $\mathsf{K}$, where $\mathrm{M}(r)$ operations in $\mathsf{K}$ are required to reduce $g$ modulo the characteristic polynomial of $T$.

Theorem 5.2 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 $\mathsf{K}$. The simple but key observation 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 (2.1), 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 $\mathsf{K}$ when $\deg g \leq n$.

**Lemma 5.1.** *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 $\mathsf{K}$, 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 $i$th 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 $i$th 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 $\mathsf{K}$ 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 $\mathsf{K}$ (since $C$ 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). $\Box$

The basic idea of evaluating a matrix at a polynomial using canonical forms and polynomial arithmetic is certainly not new, dating back at least to Ranum (1911).

**Theorem 5.2.** *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 $\mathsf{K}$, or $O(n^3 + r^2)$ operations in $\mathsf{K}$ using standard matrix and polynomial arithmetic. Here $\mathsf{K}$ 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 $\mathsf{K}$ using Theorem 2.12, or $O(n^3)$ operations in $\mathsf{K}$ using Theorem 3.5. 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 $\mathsf{K}$ 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 5.1 above, this can be accomplished with $O(n^2)$ operations in $\mathsf{K}$. Now $g(S) = \mathrm{diag}(g(C_{f_1}), \ldots, g(C_{f_k}))$ and $g(T) = Ug(S)U^{-1}$. $\Box$

A more general theorem can be obtained using arithmetic circuits or straight-line programs (see Section 1.1). Let $\wp$ be an algebraic circuit over $\mathsf{K}$, with one input $x$ and one output. Let $\mathsf{R}$ be an extension ring of $\mathsf{K}$, so $\wp$ computes a function from $\mathsf{R} \to \mathsf{R}$. If $\wp$ has length $m$ and depth $d$, then we can evaluate $\wp$ at a point $a \in \mathsf{R}$ on a sequential arithmetic RAM over $\mathsf{R}$ with $O(m)$ operations in $\mathsf{R}$, or on an arithmetic

PRAM over $\mathsf{R}$ in time $O(d)$ on $O(m)$ processors. As a concrete example, suppose $\mathsf{R} = \mathsf{K}[x]/(f)$ where $f \in \mathsf{K}[x]$ has degree $n$. Using (Cantor & Kaltofen, 1991) we can evaluate $\wp$ at any $g \in \mathsf{K}[x]/(f)$ with $O(m \cdot \mathrm{M}(n))$ operations in $\mathsf{K}$, and in parallel time $O(d \log n)$ on $O(m \cdot \mathrm{M}(n))$ processors.

We obtain for straight-line programs a theorem similar to Theorem 5.2.

**Theorem 5.3.** *Let $\wp$ be a straight-line program over $\mathsf{K}$ as described above, with length $m$ and depth $d$. Assume that $\mathsf{K}$ has characteristic $p$ and $\#\mathsf{K} \geq n^2$. 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 $\mathsf{K}$ sequentially;*

*(ii) in parallel time $O(\log^4 n + d\log n)$ when $p \geq n$ or $p = 0$, or $O(\log^5 n/\log p + d\log n)$ when $2 \leq p < n$, on $O(\mathrm{MM}(n)\log n + (n+m)\mathrm{M}(n))$ processors.*

*The algorithm is Las Vegas, and returns an answer with probability at least $1/4$.*

**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 $S = \mathrm{diag}(C_{f_1}, \ldots, C_{f_k})$, 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 $1 \leq i \leq k$. The final result is then $U^{-1}\mathrm{diag}(h_1(C_{f_1}), \ldots, h_k(C_{f_k}))U \in \mathsf{K}^{n\times n}$. 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 $\mathsf{K}$ using Theorem 2.12. In parallel, we can use Theorem 2.17, which requires $O(\log^4 n)$ time on $O(\mathrm{MM}(n))$ processors when $p \geq n$ or $p = 0$, or time $O(\log^5 n/\log p)$ on $O(\mathrm{MM}(n))$ processors when $2 \leq p < n$. We can evaluate $\wp$ at $x$ in $\mathsf{K}[x]/(f_1)$ with $O(m \cdot \mathrm{M}(n))$ operations in $\mathsf{K}$ 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).

**Corollary 5.4.** *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 $\mathsf{K}$, where $\mathsf{K}$ 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 5.3 to obtain the result. $\square$

We now summarize the above results for fields with fewer than $n^2$ elements. These follow immediately from the use of Theorems 2.13 (sequential) and 2.18 (parallel) instead of Theorems 2.12 and 2.17 respectively.

**Theorem 5.5.** *Let $\mathsf{K}$ be any field with $q < n^2$ elements and characteristic $p$, $T \in \mathsf{K}^{n\times n}$, and $\mathsf{F}$ the smallest extension field of $\mathsf{K}$ 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 $\mathsf{K}$ 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 $\mathsf{K}$. 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 $\mathsf{K}$, then we can multiply two $n\times n$ matrices over $\mathsf{K}$ with $O(t(n))$ operations in $\mathsf{K}$. 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 $\mathsf{K}$ and, as discussed above, a deterministic upper bound of $O(n^{3.5})$ operations in $\mathsf{K}$, making this assumption reasonable.

We begin with the case when $\#\mathsf{K} \geq r-1$. This is later generalized to all 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 5.6.** *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 $g = \sum_{0\le i\le r} b_i x^i$, then

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

where $h = \sum_{0\le i\le r-2} z^i (\sum_{i+2\le j\le r} (j-i-1)b_j) \in \mathsf{K}[z]$ and $z$ is an indeterminate. Since the coefficient of $z^{r-2}$ in $h$ is $b_r \ne 0$, 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 $h(c) \ne 0$ because $h$ has at most $r-2$ roots. Such a $c$ can be found by evaluating $h$ at $r-1$ distinct points in $\mathsf{K}$, 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 $\mathsf{K}$. Now $AB = h(c)^{-1}C$, which can be computed with an additional $O(t(n))$ operations in $\mathsf{K}$.

If $q = \#\mathsf{K} \le r-2$ things are somewhat trickier. It may be the case that there does not exist a $c \in \mathsf{K}$ such that $h(c) \ne 0$. To overcome this problem we work in an algebraic extension $\mathsf{F}$ of $\mathsf{K}$ of degree $\mu = \log_q \lceil r-1 \rceil$ over $\mathsf{K}$, which clearly does contain $r-1$ distinct elements. Since we are only given an algorithm which works for matrices over $\mathsf{K}$, we use the companion matrix representation of $\mathsf{F}$ as $\mu\times\mu$ matrices over $\mathsf{K}$ (see Lidl & Niederreiter 1983, Section 2.5). To do this, we first construct a monic irreducible polynomial $\gamma \in \mathsf{K}[x]$ of degree $\mu$. Such a polynomial can easily be found with $O\tilde{}(\sqrt{r})$ operations in $\mathsf{K}$ (see Shoup 1993). Suppose $\gamma$ has companion matrix $\Gamma \in \mathsf{K}^{\mu\times\mu}$. Then $\mathsf{F} = \mathsf{K}[\Gamma] \subseteq \mathsf{K}^{\mu\times\mu}$ is a field with $q^\mu \ge m-2$ elements. The field $\mathsf{K}$ is embedded in $\mathsf{F}$ as the $\mathsf{K}$-multiples of the $\mu\times\mu$ identity matrix. Now, if we can evaluate $g$ at any $n\times n$ matrix over $\mathsf{K}$ with $t(n)$ operations in $\mathsf{K}$, we can evaluate $g$ at an $n\times n$ matrix over $\mathsf{F}$ with $O(t(n\mu))$ or $O(t(n)\log_q^{3.5} r)$ operations in $\mathsf{K}$.

The field $\mathsf{F}$ has at least $m-1$ elements so the above reduction from multiplying matrices to evaluating $g$ at a matrix applies. A constant $c \in \mathsf{F}$ exists such that $h(c) \ne 0$, and can be found with $O(\mathrm{M}(\mu)\cdot \mathrm{M}(r)\log r)$ operations in $\mathsf{K}$. To multiply $A, B \in \mathsf{K}^{n\times n}$, we write $A, B$ as matrices in $\mathsf{F}^{n\times n} = \mathsf{K}[\Gamma]^{n\times n}$, then determine $AB \in \mathsf{F}^{n\times n}$ as above. This requires $O(t(n\mu))$ or $O(t(n)\log_q^{3.5} r)$ operations in $\mathsf{K}$. The product $AB \in \mathsf{K}^{n\times n}$ can be recovered by multiplying each element of $AB$ by $h(c)^{-1} \in \mathsf{K}[\Gamma]$, which can be done with $O(n^2 \log_q^3 r)$ additional operations in $\mathsf{K}$.

**Theorem 5.7.** *Let $g \in \mathsf{K}[x]$ have degree $r \ge 2$. Suppose we can evaluate $g(T)$ for* *any matrix $T \in \mathsf{K}^{n\times n}$ with $t(n)$ operations in $\mathsf{K}$. 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 $\mathsf{K}$. If $q = \#\mathsf{K} < r-1$ we can multiply two $n\times n$ matrices with $O(t(n)\log_q^{3.5} r + \mathrm{M}(r)\cdot \mathrm{M}(\log r)\log r)$ operations in $\mathsf{K}$.*

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 $\mathsf{K}$ when $\#\mathsf{K} \geq r-1$, and $O(t(n)\log_q^{3.5} r)$ operations in $\mathsf{K}$ when $\#\mathsf{K} < r-1$.

## Evaluating a Polynomial at an Integer Matrix

The cost in terms of bit operations of evaluating a polynomial $g \in \mathbb{Z}[x]$ at a matrix $T \in \mathbb{Z}^{n\times n}$ is determined in terms of four parameters: $n$, $\alpha = \|T\|_\infty$, $r = \deg g$, $\rho = \|g\|_\infty$. From these we can calculate an a priori upper bound on $\|g(T)\|_\infty$ of $\delta = 2\rho n^{r-1}\alpha^r$, since

$$\|g(T)\|_\infty \leq \sum_{0\leq i\leq r} |b_i|\|T^i\|_\infty \leq \sum_{0\leq i \leq r} \rho n^{i-1}\alpha^i < 2\rho n^{r-1}\alpha^r = \delta.$$

We compute $g(T)$ modulo a collection of small primes and then use Chinese remaindering to recover $g(T) \in \mathbb{Z}^{n\times n}$. Since the entries of $g(T)$ are all polynomials in the entries of $T$, this is in fact much easier than, say, computing the Frobenius form. In particular, for any prime $p$, $g(T \bmod p) = g(T) \bmod p$, so there are no bad primes, at least none which could lead to an incorrect answer. As in our algorithm for computing the Frobenius form of an integer matrix, we use primes $p$ between $2^{\ell-1}$ and $2^\ell$, where $\ell = 6 + \log\log\delta = O(\log(\log\rho + r\log n + r\log\alpha))$. By Lemma 1.8 there are at least $2\lceil\lceil\log_2(2\delta)\rceil/(\ell-1)\rceil$ of these primes, so in particular, there are enough to uniquely represent integers between $-\delta$ and $\delta$. Also, we make use of our algorithm for computing the Frobenius form (over a finite field), which works best for fields of size at least $n^2$, so we insist that $2^{\ell-1} \geq n^2$, or equivalently that $\ell \geq 1 + 2\log_2 n$.

First, find $T \bmod p$ for $\lceil\lceil\log_2(2\delta)\rceil/(\ell-1)\rceil$ primes $p$, which requires $O(n^2\mathrm{M}(\log\rho + r(\log n + \log\alpha))\ell)$ bit operations by Lemma 1.6. This can be computed using standard arithmetic with $O(n^2(\log\rho + r(\log n + \log\alpha))\log\alpha)$ bit operations. Now compute

$g(T) \bmod p$ for each of the selected primes $p$ using Theorem 5.2. This requires an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n) + \mathrm{M}(r))\cdot(\log\rho + r(\log n + \log\alpha))\cdot M(\ell)/\ell)$ bit operations, or an expected number of $O((n^3 + r^2)\cdot(\log\rho + r(\log n + \log\alpha))\ell)$ bit operations using standard arithmetic. Recovering $g(T)$ from its residues requires an additional $O(n^2\mathrm{M}(\log\rho + r(\log n + \log\alpha))\ell)$ bit operations, or $O(n^2(\log\rho + r(\log n + \log\alpha))^2)$ bit operations using standard arithmetic.

**Theorem 5.8.** *Let $T \in \mathbb{Z}^{n\times n}$ with $\alpha = \|T\|_\infty$, and $g \in \mathbb{Z}[x]$ with degree $r$ and $\rho = \|g\|_\infty$. Then we can compute $g(T)$ by a Las Vegas algorithm with an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n) + \mathrm{M}(r))\cdot(\log\rho + r(\log n + \log\alpha))\cdot M(\ell)/\ell + n^2\mathrm{M}(\log\rho + r(\log n + \log\alpha))\ell)$ or $O\tilde{}(\mathrm{MM}(n)\cdot(\log\rho + r\log\alpha) + n^2(\log\rho + r\log\alpha)\log\alpha)$ bit operations. Using standard arithmetic, we can compute $g(T)$ with an expected number of $O((n^3 + r^2)\cdot(\log\rho + r(\log n + \log\alpha))\ell + n^2(\log\rho + r(\log n + \log\alpha))^2)$ bit operations. Here $\ell = \Theta(\log(\log\rho + r\log n + r\log\alpha))$.*

## 5.2 Minimal Polynomials and Matrix Similarity

We next consider applications of the Frobenius form to finding the minimal polynomial (and all other invariant factors) of $T$, and to matrix similarity. Unless stated otherwise, we assume throughout this section that $\mathsf{K}$ is any field with at least $n^2$ elements. For fields $\mathsf{K}$ with fewer than $n^2$ elements similar results hold with running times multiplied by a small power of $\log_q n$. These results for smaller fields are summarized in Theorem 5.11 below.

Finding the minimal polynomial of a matrix $T \in \mathsf{K}^{n\times n}$, and in fact all the invariant factors, is trivial once the Frobenius form has been determined. The Monte Carlo type probabilistic algorithm of Wiedemann (1986) was previously the best known for this problem, and requires $O(n^3)$ operations in $\mathsf{K}$, with an arbitrarily small possibility of error. Wiedemann's algorithm actually regards $T$ as a "black box" and requires that $T$ be multiplied by a vector in $\mathsf{K}^{n\times 1}$ $O(n)$ times. In general each such matrix-vector product requires $O(n^2)$ operations in $\mathsf{K}$, but may require fewer operations when $T$ has a special form (for instance, if $T$ is "sparse" and has only $O(n)$ non-zero entries). Our algorithm requires $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$ to compute the minimal polynomial of $T$ and is of the Las Vegas type, that is, there is no possibility of erroneous output.

**Theorem 5.9.** *Given any $T \in \mathsf{K}^{n\times n}$, where $\mathsf{K}$ is any field with at least $n^2$ elements, there is a Las Vegas type probabilistic algorithm to determine all invariant factors $f_1, \ldots, f_k \in \mathsf{K}[x]$ of $T$ (where $f_1$ is the minimal polynomial of $T$), which requires $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ or $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$.*

Determining if two matrices in $\mathsf{K}^{n\times n}$ are similar has been considered in (Kaltofen *et al.*, 1986 & 1990), where an $RNC^2$ algorithm is presented. This is improved upon in (Zalcstein & Garzon, 1987), where an $NC^2$ algorithm is presented. The best sequential (deterministic) algorithms previously known for this problem are the Frobenius form algorithms of Ozello (1987) and Lüneburg (1987), both of which require $O(n^4)$ operations in $\mathsf{K}$.

**Theorem 5.10.** *Given $T, A \in \mathsf{K}^{n\times n}$ over a field $\mathsf{K}$ with at least $n^2$ elements, there is a Las Vegas type probabilistic algorithm to determine if $T$ is similar to $A$ with $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ or $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$. The algorithm also returns a non-singular $X \in \mathsf{K}^{n\times n}$ such that $A = X^{-1}TX$.*

**Proof.** Use the algorithm `FrobeniusForm` to find the Frobenius forms $S_1, S_2 \in \mathsf{F}^{n\times n}$ of $T$ and $A$ respectively, along with transition matrices $U_1, U_2 \in \mathsf{F}^{n\times n}$ such that $S_1 = U_1^{-1}TU_1$ and $S_2 = U_2^{-1}AU_2$. If $S_1 \neq S_2$ then $T$ is not similar to $A$, since two matrices are similar if and only if they have the same Frobenius form (see Gantmacher 1990, Chapter 7, Theorem 9″). If $S_1 = S_2$ then $U_2U_1^{-1}TU_1U_2^{-1} = A$, and we return $X = U_1U_2^{-1}$. $\square$

When $\mathsf{K}$ is a field with fewer than $n^2$ elements the above applications, which all use the algorithm `FrobeniusForm`, must work in an extension $\mathsf{F}$ of $\mathsf{K}$ which does contain $n^2$ elements. We use the same $\mathsf{F} = \mathsf{K}[x]/(\psi)$ as in Theorem 2.13, where $\psi \in \mathsf{K}[x]$ is irreducible of degree $\lceil \log_q n \rceil$.

**Theorem 5.11.** *Let $\mathsf{K}$ be any field with $q < n^2$ elements, $T, A \in \mathsf{K}^{n\times n}$, and $\mathsf{F}$ the smallest extension field of $\mathsf{K}$ containing $n^2$ elements. Las Vegas type probabilistic algorithms exist as follows:*

*(i) we can compute all invariant factors $f_1, \ldots, f_k \in \mathsf{K}[x]$ of $T$ (where $f_1$ is the minimal polynomial of $T$), with $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot \mathrm{M}(\log_q n)\log\log_q n)$ or $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$;*

(ii) *we can determine if $A$ is similar to $T$ and if so find an invertible $X \in \mathsf{F}^{n\times n}$ such that $A = X^{-1}TX$, with $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot \mathrm{M}(\log_q n)\log\log_q n)$ or $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$.*

**Proof.** These problems are solved directly from the Frobenius form $S$ of $T$, as in Theorems 5.9 and 5.10 respectively. When $q < n^2$, the Frobenius form $S$ can be computed with $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot \mathrm{M}(\log_q n)\log\log_q n)$ operations in $\mathsf{K}$ (see Theorem 2.13). $\square$

## A Lower Bound on Finding the Minimal Polynomial

We have demonstrated that finding the minimal polynomial of a matrix $T \in \mathsf{K}^{n\times n}$ can be accomplished by a Las Vegas type algorithm requiring $O\tilde{}(\mathrm{MM}(n))$ operations in $\mathsf{K}$. As the next theorem shows, we cannot expect to find an algorithm asymptotically much faster than this. Specifically, we show that if we can find the minimal polynomial of any $T \in \mathsf{K}^{n\times n}$ with at most $t(n)$ operations in $\mathsf{K}$, then there is a Las Vegas type algorithm to compute the determinant of $T$ with $O(t(n))$ operations in $\mathsf{K}$. It is derived from Baur & Strassen (1982) in Theorem 1.3 that such an algorithm to compute the determinant implies the existence of an algorithm to multiply any two matrices in $\mathsf{K}^{n\times n}$ for which $O(t(n))$ operations in $\mathsf{K}$ are sufficient.

We assume in the following that $\mathsf{K}$ contains at least $8n^2+1$ distinct elements, and hence $8n^2$ distinct units. Let $f \in \mathsf{K}[x]$ be the minimal polynomial of $T \in \mathsf{K}^{n\times n}$. We now show how to compute the determinant of $T$. One easy case is when $x \mid f$, so $\det T = 0$ and we are done. If $x \nmid f$ then $T$ is non-singular. A second easy case is when $\deg f = n$, and so $f$ is equal to the characteristic polynomial $\mathrm{char}(T) \in \mathsf{K}[x]$ of $T$. In this case $(-1)^{\deg f}\cdot f(0)$ is the determinant of $T$ and we are again finished. The more difficult case is when $T$ is non-singular and $f \neq \mathrm{char}(T)$. Here our reduction is essentially the same as Kaltofen's (1992) version of Wiedemann's(1986) algorithm for computing the determinant of a sparse matrix. Recall the definitions of an *upper triangular unit Toeplitz matrix* $A \in \mathsf{K}^{n\times n}$ and a *lower triangular Toeplitz matrix*

$B \in \mathsf{K}^{n\times n}$:

$$A = \begin{pmatrix} 1 & a_2 & a_3 & \cdots & a_n \\ & 1 & a_2 & \cdots & a_{n-1} \\ & & 1 & \ddots & \vdots \\ & & & \ddots & a_2 \\ & & & & 1 \end{pmatrix} \qquad B = \begin{pmatrix} b_1 & & & & \\ b_2 & b_1 & & & \\ b_3 & b_2 & b_1 & & \\ \vdots & & \ddots & \ddots & \\ b_n & b_{n-1} & \cdots & b_2 & b_1 \end{pmatrix}, \tag{5.1}$$

for $a_2, \ldots, a_n, b_1, \ldots, b_n \in \mathsf{K}$.

The following fact is shown by Kaltofen & Pan (1992), Proposition 1.

**Fact 5.12.** *Let $T \in \mathsf{K}^{n\times n}$ and $L$ a subset of $\mathsf{K} \setminus \{0\}$. Choose an upper triangular unit Toeplitz matrix $A \in \mathsf{K}^{n\times n}$ and a lower triangular Toeplitz matrix $B \in \mathsf{K}^{n\times n}$ as in (5.1), with entries $a_2, \ldots, a_n, b_1, \ldots, b_n$ selected randomly from $L$, with $b_1 \neq 0$. Then, with probability at least $1 - 4n^2/\#L$, the minimal polynomial* $\min(ATB) \in \mathsf{K}[x]$ *of $ATB \in \mathsf{K}^{n\times n}$ equals the characteristic polynomial* $\mathrm{char}(ATB) \in \mathsf{K}[x]$ *of $ATB$.*

We use this fact as follows. Observe that $\det(A) = 1$ and $\det(B) = b_1^n$. Thus, $\det(T) = \det(ATB)/b_1^n$. If $\min(ATB) = \mathrm{char}(ATB)$ then we know $\det(ATB)$ is the constant coefficient of its minimal polynomial multiplied by $(-1)^n$, and from this we can compute $\det(T)$ with $O(\log n)$ additional operations in $\mathsf{K}$ (the time required to compute $b_1^n$).

The product $ATB$ can be computed with $O(n\mathrm{M}(n)) = O(n^2 \log n \log\log n)$ operations in $\mathsf{K}$ as follows. We require only a standard observation about Toeplitz matrices (see Kaltofen & Saunders 1991): for $a_i, b_i, c_i \in \mathsf{K}$ for $1 \leq i \leq n$,

$$\begin{pmatrix} a_1 & & & & \\ a_2 & a_1 & & & \\ a_3 & a_2 & a_1 & & \\ \vdots & & \ddots & \ddots & \\ a_n & a_{n-1} & \cdots & a_2 & a_1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{pmatrix},$$

if and only if

$$(a_1 + a_2 x + \cdots + a_n x^n)(b_1 + b_2 x + \cdots + b_n x^n) \equiv (c_1 + c_2 x + \cdots c_n x^n) \bmod x^n.$$

Using fast polynomial multiplication, we can pre-multiply a vector in $\mathsf{K}^{n\times 1}$ by a lower triangular Toeplitz matrix in $\mathsf{K}^{n\times n}$ with $O(\mathrm{M}(n))$ operations in $\mathsf{K}$. Hence we can pre-multiply a lower triangular Toeplitz matrix in $\mathsf{K}^{n\times n}$ by any matrix in $\mathsf{K}^{n\times n}$ with $O(n\mathrm{M}(n))$ operations in $\mathsf{K}$. Similarly, for any upper triangular Toeplitz matrix $X \in \mathsf{K}^{n\times n}$ and any $v \in \mathsf{K}^{1\times n}$, we can compute $vX = X^t v^t$ with $O(\mathrm{M}(n))$ operations in $\mathsf{K}$, since $X^t$ is a lower triangular Toeplitz matrix. Hence we can post-multiply any matrix in $\mathsf{K}^{n\times n}$ by an upper triangular Toeplitz matrix with $O(n\mathrm{M}(n))$ operations in $\mathsf{K}$. We can, therefore, compute $ATB$ as in Lemma 5.12 with $O(n\mathrm{M}(n))$ or $O(n^2 \log n \log\log n)$ operations in $\mathsf{K}$.

**Theorem 5.13.** *Let $n \in \mathbb{N}$ and $\mathsf{K}$ be any field with at least $8n^2+1$ elements. Suppose we can compute the minimal polynomial of any matrix in $\mathsf{K}^{n\times n}$ with at most $t(n)$ operations in $\mathsf{K}$, where $t(n) = \Omega(n^{2+\epsilon})$ for $\epsilon > 0$. Then we can compute the determinant of any matrix with $O(t(n))$ operations in $\mathsf{K}$.*

## Computing the Minimal Polynomial of an Integer Matrix

In this subsection we present a Las Vegas type algorithm for computing the minimal polynomial $f \in \mathbb{Z}[x]$ of a matrix $T \in \mathbb{Z}^{n\times n}$. The coefficients of the minimal polynomial are not polynomials in the coefficients of $T$, and hence it is possible that modulo some primes $p$, $\min(T \bmod p) \neq \min(T) \bmod p$. However, we obtain a fast algorithm by using the following lemma.

**Lemma 5.14.** *Let $T \in \mathbb{Z}^{n\times n}$ with minimal polynomial $f \in \mathbb{Z}[x]$ and $\alpha = \|T\|_\infty$, and let $p_1, \ldots, p_r \in \mathbb{N}$ be primes with product at least $2^{n+1}e^{n/2}\alpha^{2n}n^{3n/2-1}$. If the degree of the minimal polynomial $f^{(p_i)} \in \mathbb{Z}_{p_i}[x]$ of $(T \bmod p_i) \in \mathbb{Z}_{p_i}^{n\times n}$ is the same for each $i$ with $1 \le i \le r$, then $f^{(p_i)} = f \bmod p_i$ for $1 \le i \le r$.*

**Proof.** For every prime $p$ it is the case that $f^{(p)}$ divides $(f \bmod p)$ as a polynomial in $\mathbb{Z}_p[x]$, and hence $\deg f^{(p)} \le \deg f$. Since $T$ has the same degree sequence as $T \bmod p$ for almost all primes $p$, we know that $\deg f^{(p)} = \deg f$ for almost all primes $p$ (see Lemma 4.4).

Lemma 4.2 implies that $\|f\|_\infty < \beta = 2^n e^{n/2} \alpha^n n^{n/2}$. For any polynomial $g \in \mathbb{Z}[x]$ of degree at most $n$ and $\|g\|_\infty < \beta$, we have

$$\|g(T)\|_\infty < \sum_{0 \le i \le n} \beta n^{i-1} \alpha^i < 2\beta n^{n-1} \alpha^n \le 2^{n+1} e^{n/2} \alpha^{2n} n^{3n/2-1}.$$

Let $m = p_1 \cdots p_r \in \mathbb{N}$. If $f(T) \bmod m = 0$ and $m \ge 2^{n+1} e^{n/2} \alpha^{2n} n^{3n/2-1}$ then it must be the case that $f(T) = 0$, since each entry of $f(T)$ has absolute value less than $2^{n+1} e^{n/2} \alpha^{2n} n^{3n/2-1}$. Therefore, the unique common pre-image in $\mathbb{Z}[x]$ of $f^{(p_i)}$ for $1 \le i \le r$ must be $f$. $\square$

By Lemma 4.4, for any prime $p$, if $T$ and ($T$ mod $p$) have the same degree sequence then certainly $\min(T) \bmod p = \min(T \bmod p) \in \mathbb{Z}_p[x]$. The degree sequence of $T$ mod $p$ equals that of $T$ for all primes except possibly those dividing a single integer $c$ of absolute value at most $\delta = 2^{n/2} n^{n^2+n} \alpha^{n^2-n}$ ($c = \det(H)$, where $H \in \mathbb{Z}^{n \times n}$ is a modular cyclic transition matrix of $T$; see Section 4.2). Fix $\ell = 6 + \log\log\delta = \Theta(\log n + \log\log\alpha)$): by Lemma 1.8 at most $1/2$ the primes between $2^{\ell-1}$ and $2^\ell$ divide $c$.

Our algorithm proceeds as follows. We consider the primes between $2^{\ell-1}$ and $2^\ell$ in increasing order, maintaining a list of good primes $\Lambda$. The set $\Lambda$ is initially empty. At all times we maintain a variable $d$ such that for all primes $p \in \Lambda$ we have $d = \deg\min(T \bmod p)$. For each prime $p > 2^{\ell-1}$ in sequence, compute $f^{(p)} = \min(T \bmod p)$ of degree $d^{(p)}$. If $d^{(p)} = d$ then add $p$ to $\Lambda$. If $d^{(p)} > d$ then $d$ must have been wrong, and hence so were all primes in $\Lambda$, and we reassign $\Lambda := \{p\}$ and $d = \deg(\min(T \bmod p))$. If $d^{(p)} < d$ then $p$ is a bad prime, and we do not add it to $\Lambda$. This process is continued until $\Lambda$ contains $\left\lceil \left\lceil \log_2(2^{n+1} e^{n/2} \alpha^{2n} n^{3n/2-1}) \right\rceil / (\ell - 1) \right\rceil$ primes, whose product must be at least $2^{n+1} e^{n/2} \alpha^{2n} n^{3n/2-1}$ (since all these primes have $\ell$ bits). By Lemma 5.14 we know that upon completion of this algorithm we have $\min(T \bmod p) = f \bmod p$ for all primes $p \in \Lambda$. Moreover, since $\prod_{p \in \Lambda} p \ge 2^{n+1} e^{n/2} \alpha^{2n} n^{3n/2-1} \ge 2\|f\|_\infty$, we can determine the coefficients of $f$ from its residues modulo these primes.

The cost of this algorithm is easy to determine. During this process we examine $O(n(\log n + \log\alpha)/\ell)$ primes. For each of these primes $p \in \Lambda$, we reduce $T$ modulo $p$. By Lemma 1.6 this requires $O(n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ bit operations, or, using standard arithmetic, $O(n^3(\log n + \log\alpha)\ell)$ bit operations. Next we compute the minimal polynomial of $T \bmod p$ for all $p \in \Lambda$. This requires an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot n(\log n + \log\alpha) \cdot \mathrm{M}(\ell)/\ell)$ bit operations, or, using standard arithmetic, an expected number of $O(n^4(\log n + \log\alpha)\ell)$ bit operations. Finally, to recover the minimal polynomial from its residues modulo the primes in $\Lambda$ requires $O(n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ bit operations, or, using standard arithmetic, $O(n^4(\log n + \log\alpha)^2)$ bit operations, by Lemma 1.7.

**Theorem 5.15.** *Let $T \in \mathbb{Z}^{n\times n}$ with $\alpha = \|T\|_\infty$. We can compute the minimal polynomial $f \in \mathbb{Z}[x]$ of $T$ with an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot n(\log n + \log\alpha) \cdot \mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ or $O\tilde{~}(\mathrm{MM}(n) \cdot n\log\alpha)$ bit operations. Using standard arithmetic, we can compute $f$ with an expected number of $O(n^4(\log n + \log\alpha)^2)$ bit operations. Here $\ell = \Theta(\log n + \log\log\alpha)$.*

## Determining Matrix Similarity

Determining whether two matrices $T_1, T_2 \in \mathsf{K}^{n\times n}$ are similar is trivial once we have computed their Frobenius forms: they have the same Frobenius form $S \in \mathsf{K}^{n\times n}$ if and only if they are similar. Moreover, if $S = U_1^{-1}T_1U_1 = U_2^{-1}T_2U_2$ for invertible $U_1, U_2 \in \mathsf{K}^{n\times n}$ then we can compute a transition matrix from $T_1$ to $T_2$: $T_2 = (U_1U_2^{(-1)})^{-1}T_1(U_1U_2^{-1})$. Using Theorems 2.12 and 3.5 we obtain the following:

**Theorem 5.16.** *Let $\mathsf{K}$ be a field with at least $n^2$ elements. Given $T_1, T_2 \in \mathsf{K}^{n\times n}$, we can determine if $T_1$ and $T_2$ are similar, and if so compute an invertible matrix $U_0 \in \mathsf{K}^{n\times n}$ such that $T_2 = U_0^{-1}T_1U_0$. The algorithm is of the Las Vegas type, and requires an expected number of $O(\mathrm{MM}(n)\log n + n\mathrm{M}(n))$ operations in $\mathsf{K}$, or $O(n^3)$ operations in $\mathsf{K}$ using standard arithmetic.*

When the field $\mathsf{K}$ has fewer than $n^2$ elements, similar results hold, using Theorems 2.13 and 3.10.

**Theorem 5.17.** *Let $\mathsf{K}$ be a field with $q = \#\mathsf{K} < n^2$. Given matrices $T_1, T_2 \in \mathsf{K}^{n\times n}$, we can determine by a Las Vegas algorithm if $T_1$ and $T_2$ are similar, and if so compute an invertible matrix $U_0 \in \mathsf{K}^{n\times n}$ such that $T_2 = U_0^{-1}T_1U_0$. The computation requires an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot \mathrm{M}(\log_q n)\log\log_q n)$ operations in*

*$\mathsf{K}$, and $U_0$ is given over a field $\mathsf{F}$ of algebraic degree $O(\log_q n)$ over $\mathsf{K}$ as in Theorem 2.13. We can perform the same computation using standard arithmetic, and returning the transition matrix in $\mathsf{K}^{n\times n}$, with an expected number of $O(n^3 \log_q n)$ operations in $\mathsf{K}$.*

## 5.3 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 $\mathsf{K}$ 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 $\mathsf{K}$ of exponential degree over $\mathsf{K}$, 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

$$J_g^{(m)} = \begin{pmatrix} \boxed{C_g} & \boxed{I_r} & & & \\ & \boxed{C_g} & \boxed{I_r} & & \mathbf{0} \\ & & \ddots & \ddots & \\ & \mathbf{0} & & \ddots & \boxed{I_r} \\ & & & & \boxed{C_g} \end{pmatrix} \in \mathsf{K}^{mr\times mr}, \tag{5.2}$$

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 5.18 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 $\mathsf{K}$, 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).

Let $\mathsf{K}$ 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 $\mathsf{K}$, 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 main observation is that once we know the Frobenius form of $T$ and the complete factorization of its minimal polynomial $f_1$, then we can immediately determine the rational Jordan form of $T$.

**Lemma 5.18.** *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 $(x - \gamma)^m \in \mathsf{L}[x]$ (see Gantmacher 1990, Section 7.7), so that 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 5.19.** *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 pairwise 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, $\min(B; \mathsf{K}^{n\times 1}) = \operatorname{lcm}(g_1^{e_1}, \ldots, g_s^{e_s}) = g_1^{e_1} \cdots g_s^{e_s} = f$, since the minimal polynomial of $J_{g_i}^{(e_i)}$ is $g_i^{m_i}$ for $1 \leq i \leq s$ by Lemma 5.18. Since $\deg f = n$ and $f$ is the minimal polynomial of $B$, $B$ is similar to the companion matrix $C_f$ of $f$. $\square$

**Theorem 5.20.** *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 $1 \leq i \leq k$. If $B_i \in \mathsf{K}^{d_i \times d_i}$ is the rational Jordan form of $C_{f_i}$ for $1 \leq i \leq k$, then $Q = \operatorname{diag}(B_1, \ldots, B_k) \in \mathsf{K}^{n \times n}$ is the rational Jordan form of $T$.*

**Proof.** By Lemma 5.19 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 $1 \leq i \leq k$. The matrix $P = \operatorname{diag}(P_1, \ldots, P_k) \in \mathsf{K}^{n \times n}$ satisfies $Q = P^{-1} T P$, and $Q$ is in rational Jordan form. $\square$

**Theorem 5.21.** *Let* $\mathsf{K}$ *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* $\mathsf{K}$.

**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 5.20. 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 $\mathsf{K}$ has fewer than $n^2$ elements we embed $\mathsf{K}$ into a slightly larger field $\mathsf{F}$ to find the Frobenius form of $T$, using Theorem 2.13. As in Theorem 2.13, $\mathsf{F} = \mathsf{K}[x]/(\psi)$, where $\psi \in \mathsf{K}[x]$ is monic and irreducible of degree $\lceil 2\log_q n\rceil$.

**Theorem 5.22.** *Let* $\mathsf{K}$ *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* $\mathsf{K}$.

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 5.23.** *Let* $\mathsf{K}$ *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* $\mathsf{K}$ *if* $\#\mathsf{K} \geq n^2$*, and* $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot \mathrm{M}(\log_q n)\log\log_q n + n\log q)$ *or* $O\tilde{}(\mathrm{MM}(n) + n\log q)$ *operations in* $\mathsf{K}$ *if* $\#\mathsf{K} < n^2$.

## Computing the Jordan Form of an Integer Matrix

The Jordan form $Q \in \mathbb{Z}^{n\times n}$ of an integer matrix $T$ is easily computed from the Frobenius form $S$ of $T$ if we are given the complete factorization of $T$'s minimal polynomial. In this subsection we show that this can be done by a Las Vegas algorithm requiring an expected number of $O\tilde{\ }(n^5 \log^2 \alpha)$ bit operations using standard arithmetic, where $\alpha = \|T\|_\infty$. We also present a Monte Carlo type algorithm which requires an expected number of $O\tilde{\ }(n^4 \log \alpha)$ bit operations, with an exponentially small probability of error. The question of finding the rational Jordan form given a factorization of the minimal polynomial of a matrix is also examined in Gil (1992), where an algorithm requiring $O\tilde{\ }(n^{11} \log^2 \alpha)$ bit operations is presented.

Our algorithm for integer matrices is simply the one presented above for general fields, using the algorithms for the Frobenius form of an integer matrix exhibited in Section 4.2. Given the Frobenius form of a matrix $T \in \mathbb{Z}^{n\times n}$ and a complete factorization of its minimal polynomial $f \in \mathbb{Z}[x]$, we can write down the Jordan form of $T$ as in Theorem 5.20. Finding a transition matrix to the Jordan form within the desired time is somewhat trickier, and we consider this in the sequel.

**Theorem 5.24.** *Given $T \in \mathbb{Z}^{n\times n}$ with $\|T\|_\infty \le \alpha$, and the complete factorization of its minimal polynomial $f_1 \in \mathbb{Z}[x]$ in $\mathbb{Z}[x]$, we can find the rational Jordan form of $T$ with*

*(i) a Las Vegas algorithm requiring an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n)) \cdot n^2(\log n + \log\alpha) \cdot \mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ or $O\tilde{\ }(\mathrm{MM}(n) \cdot n^2 \log\alpha)$ bit operations,*

*(ii) a Las Vegas algorithm using standard arithmetic requiring an expected number of $O(n^5(\log n + \log\alpha)\ell + n^4 \log^2\alpha)$ bit operations,*

*(iii) a Monte Carlo algorithm requiring an expected number of $O((\mathrm{MM}(n)\log n + n\mathrm{M}(n))\cdot n(\log n + \log\alpha)\cdot \mathrm{M}(\ell)/\ell + n^2\mathrm{M}(n(\log n + \log\alpha))\ell)$ or $O\tilde{\ }(\mathrm{MM}(n)\cdot n\log\alpha)$ bit operations and giving an erroneous answer with probability less than $1/2^s$, where $s = \Omega(n(\log n + \log\alpha)/\ell)$.*

*(iv) a Monte Carlo algorithm using standard arithmetic requiring an expected number of $O(n^4(\log n + \log\alpha)^2)$ bit operations and giving an erroneous answer with*

*probability less than $2^s$, where $s = \Omega(n(\log n + \log\alpha)/\ell)$.*

*Here $\ell = \Theta(\log n + \log\log\alpha)$.*

The algorithm of Schönhage (1984) will factor the minimal polynomial $f_1$ of $T$ with $O(n^{6+\epsilon} + n^4(\log\|f_1\|_\infty)^{2+\epsilon})$ or $O(n^{6+\epsilon}(\log n + \alpha)^{2+\epsilon})$ bit operations for any $\epsilon > 0$. Using standard arithmetic, Schönhage's algorithm requires $O(n^8 + n^5\log^3\|f_1\|_\infty)$ or $O(n^8(\log n + \log\alpha)^3)$ bit operations. This yields the following corollary:

**Corollary 5.25.** *Given $T \in \mathbb{Z}^{n\times n}$ with $\|T\|_\infty \le \alpha$, we can find the rational Jordan form of $T$ by a Las Vegas type probabilistic algorithm requiring an expected number of $O(n^{6+\epsilon}(\log n + \alpha)^{2+\epsilon})$ bit operations for any $\epsilon > 0$. Using standard arithmetic, an expected number of $O(n^8(\log n + \log\alpha)^3)$ bit operations are required.*

The method summarized above does not yield a transition matrix $P \in \mathbb{Z}^{n\times n}$ such that $P^{-1}TP = Q$ is in Frobenius form. Such a transition matrix can be constructed, but we must be very careful about the sizes of the entries. A direct implementation of the technique described above for matrices over general fields will give a transition matrix with entries requiring $O\tilde{~}(n^3\log\alpha)$ bits each, which is too large to manage within the allowed time. First, using Theorem 4.6, we find a matrix $U \in \mathbb{Q}^{n\times n}$ such that $S = U^{-1}TU$ is the Frobenius form of $T$. The numerator and denominator of each entry of $U$ requires $O(n^2(\log n + \log\alpha))$ bits to represent it (see Theorem 4.3). The existence of a transition matrix from the Frobenius form to the Jordan form, which has entries requiring $O(n^2(\log n + \log\alpha))$ bits each, is now demonstrated.

We first show a transition matrix from the Frobenius rational form to the "elementary rational form" (which we define below), and then exhibit a transition matrix from the elementary rational form to the rational Jordan form. Suppose $S = \operatorname{diag}(C_{f_1}, \ldots, C_{f_k})$, where $f_1, \ldots, f_k \in \mathbb{Z}[x]$ are the invariant factors of $T$ and $C_{f_i}$ is the companion matrix of $f_i$ for $1 \le i \le k$. Assume that the minimal polynomial $f_1$ of $T$ factors as $f_1 = g_1^{m_{11}} \cdots g_t^{m_{1t}}$, where $g_1, \ldots, g_t \in \mathbb{Z}[x]$ are distinct irreducible factors of $f_1$, and more generally that $f_i = g_1^{m_{i1}} \cdots g_t^{m_{it}}$ for $1 \le i \le k$. Corresponding to this factorization of $f_i$ is a block diagonal form of $C_{f_i}$:

$$C_{f_i} \text{ is similar to } E_i = \operatorname{diag}(C_{g_1^{m_{i1}}}, \ldots, C_{g_t^{m_{it}}}) \in \mathbb{Z}^{d_i\times d_i}$$

for $1 \le i \le k$ (see Gantmacher 1990, Section 7.6). A transition matrix to this block diagonal form is also easily described. For any polynomial $u = \sum_{0 \le j < d_i} c_i x^i \in \mathbb{Q}[x]$ of degree less than $d_i$, let $\Upsilon_i(u) = (c_0, \ldots, c_{d_i-1})^t \in \mathbb{Q}^{d_i \times 1}$. It is shown in (Gantmacher 1990, Section 7.6), that if

$$R_i = [\Upsilon_i(u_{i1})|\cdots|\Upsilon_i(x^{r_1 m_{i1}-1}u_{i1})|\cdots|\Upsilon_i(u_{it})|\cdots|\Upsilon_i(x^{r_t m_{it}-1}u_{it})] \in \mathbb{Z}^{n \times n},$$

where $u_{ij} = f_j / g_j^{m_{ij}} \in \mathbb{Z}[x]$ and $r_j = \deg g_j$ for $1 \le j \le t$, then $R_i^{-1} C_{f_i} R_i = E_i$. More generally, if $R = \mathrm{diag}(R_1, \ldots, R_k)$, then $E = R^{-1}SR$ is in block diagonal form, where each block is the companion of an elementary factor of $T$. We call $E$ the *elementary rational form* of $T$. This canonical form (or at least the corresponding decomposition of $\mathbb{Q}^{n\times 1}$ under $T$) is shown explicitly by Frobenius (1911). Note that the entries of $E$ are coefficients of divisors of $f$, and by Lemma 4.2 have absolute value at most $\beta = 2^n e^{n/2} \alpha^n n^{n/2}$. The entries of $R$ are also coefficients of divisors of $f_1$, and so are integers of absolute value at most $\beta$ as well.

Computing the entries of $R$ is very straightforward, requiring at most $n$ divisions of polynomials, each of degree at most $n$, or $O(n\mathrm{M}(n))$ operations in $\mathbb{Z}$. We ignore the cost in terms of bit operations of computing $R$ for now, since the ultimate algorithm is modular.

**Lemma 5.26.** *Given a matrix $S \in \mathbb{Z}^{n\times n}$ in Frobenius form, with $\|S\|_\infty \le \beta$, there exists a matrix $R \in \mathbb{Z}^{n\times n}$ such that $R^{-1}SR$ is in elementary rational form, and such that $\|R\|_\infty \le \beta$. The matrix $R$ can be computed from $S$ with $O(n\mathrm{M}(n))$ integer operations.*

Next we show the existence of an easily constructed transition matrix from the elementary rational form to the rational Jordan form. It is sufficient to show a transition matrix $D_g^{(m)} \in \mathbb{Q}^{dm \times dm}$ from the companion matrix of an elementary factor $g^m \in \mathbb{Z}[x]$ of $T$, where $g = \sum_{0 \le i \le d} b_i x^i \in \mathbb{Z}[x]$ is monic and irreducible of degree $d$ and $\|g\|_\infty \le \beta$, to its Jordan block $J_g^{(m)} \in \mathbb{Z}^{n \times n}$. We make use of the $\mathbb{Q}[x]$-module isomorphism between $\mathbb{Q}^{dm\times 1}$ and $\mathbb{Q}[x]/(g^{dm})$, which in this case is rather trivial, mapping a vector $v = (c_0, \ldots, c_{dm-1})^t \in \mathbb{Q}^{dm \times 1}$ to a polynomial $\sum_{0 \le i < dm} c_i x^i$. This allows us to speak in terms of a basis for $\mathbb{Q}[x]/(g^m)$ instead of $\mathbb{Q}^{dm\times 1}$, which is more convenient. An examination of the structure of $J_g^{(m)}$ reveals that a basis $\mathcal{B}$ for $\mathbb{Q}[x]/(g^m)$ satisfying the following conditions is required:

$$\mathcal{B} = (h_{1,1}, \ldots, h_{1,d-1}, \ldots, h_{m,0}, \ldots, h_{m,d-1}) \in \mathbb{Q}[x]^{dm},$$

where

(i) $\quad xh_{1,j-1} \equiv h_{1,j} \bmod g^m \qquad$ for $1 \leq j < d$,

(ii) $\quad xh_{1,d-1} \equiv -\sum_{0\leq j<d} b_j h_{1,j} \bmod g^m$,

(iii) $\quad xh_{i,j-1} \equiv h_{i,j} + h_{i-1,j-1} \bmod g^m \qquad$ for $2 \leq i \leq m$ and $1 \leq j < d$,

(iv) $\quad xh_{i,d-1} \equiv h_{i-1,d-1} - \sum_{0\leq j\leq d-2} b_j h_{i,j} \bmod g^m \qquad$ for $2 \leq i \leq m$.

Equivalent to (i) and (iii) respectively are

(i′) $\quad h_{1,j} \equiv xh_{1,j-1} \bmod g^m \qquad$ for $1 \leq j < d$,

(iii′) $\quad h_{i,j} \equiv xh_{i,j-1} - h_{i-1,j-1}$

$$\equiv x^j h_{i,0} - \sum_{0\leq l\leq j-1} x^{j-l-1} h_{i-1,l} \bmod g^m \qquad \text{for } 1 \leq j < d, \text{ and } 2 \leq i \leq m.$$

Thus, for $2 \leq i \leq m$ and $0 \leq j < d$, $h_{i,j}$ is uniquely determined by $h_{i,0}$. From (iii′) we obtain, for $1 \leq i \leq m$,

$$xh_{i,d-1} \equiv x^d h_{i,0} - \sum_{0\leq j<d} x^{d-1-j} h_{i-1,j} \bmod g^m,$$

while (iv) can be rewritten as

$$xh_{i,d-1} \equiv h_{i-1,d-1} - \sum_{0\leq j<d} b_j(x^j h_{i,0} - \sum_{0\leq j\leq j-1} x^{j-l-1} h_{i-1,j}) \bmod g^m.$$

Equating the right hand sides of the above two equations we get

$$\begin{aligned}\sum_{0\leq i\leq d} b_j x^j h_{i,0} = g \cdot h_{i,0} &\equiv \sum_{0\leq l<d} x^{d-1-l} h_{i-1,l} + \sum_{0\leq j<d} \sum_{0\leq l\leq j-1} b_j x^{j-l-1} h_{i-1,k} \\ &\equiv \sum_{0\leq j\leq d} \sum_{0\leq l\leq j-1} b_j x^{j-l-1} h_{i-1,l} \bmod g^m.\end{aligned}$$

As a solution to this system of modular polynomial equations we assign

$$
\begin{aligned}
h_{1,0} &= g^{m-1}, \\
h_{1,j} &= x^j g^{m-1}, && \text{for } 1 \le j < d, \\
h_{i,0} &= \left( \sum_{0\le j\le d} \sum_{0\le l\le j-1} b_j x^{j-l-1} h_{i-1,l} \right) \Big/ g && \text{for } 2 \le i \le m, \\
h_{i,j} &= x^j h_{i,0} - \sum_{0\le l\le j-1} x^{j-l-1} h_{i-1,l} && \text{for } 2 \le i \le m \text{ and } 1 \le j < d,
\end{aligned}
$$

and claim this satisfies the requirements on $\mathcal{B}$. Provided that

$$g \,\Big|\, \sum_{0\le j\le d} \sum_{0\le l\le j-1} b_j x^{j-l-1} h_{i-1,l},$$

i.e., that $h_{i,0}$ is a polynomial in $\mathbb{Z}[x]$, it is clear from the above derivation that our assignment of $h_{i,j}$ satisfies (i) through (iv). We now show that $h_{i,j} \in \mathbb{Z}[x]$. We show by induction that $g^{m-i} \mid h_{i,j}$ for $1 \le i \le m$ and $0 \le j < d$. For $i = 1$ this is obvious for all $0 \le j < d$. Assume it is true for $h_{i_0,j_0}$ with $1 \le i_0 < i$, and $0 \le j < d$. We see $g \cdot h_{i,0}$ is a $\mathbb{Z}[x]$-linear combination of $h_{i_0,j_0}$ with $1 \le i_0 < i$, so $g^{m-i+1}$ divides each term in this summation. Thus $g^{m-i} \mid h_{i,0}$. Similarly, $h_{i,j}$ is a $\mathbb{Z}[x]$-linear combination of $h_{i,0}$ and $h_{i_0,j_0}$ for $0 \le i_0 < i$, so $g^{m-i} \mid h_j$ for $0 \le j < d$. Therefore, by induction $g^{m-i} \mid h_{i,j}$ for $1 \le i \le m$ and $0 \le j < d$, and hence $h_{i,j} \in \mathbb{Z}[x]$.

A transition matrix $D_g^{(m)}$ from the companion matrix $C_{g^m} \in \mathbb{Z}^{dm\times dm}$ to its rational Jordan form, the rational Jordan block $J_g^{(m)}$ is thus

$$D_g^{(m)} = \Big[ \Upsilon(h_{1,0}) | \cdots | \Upsilon(h_{1,d-1}) | \cdots | \Upsilon(h_{m,0}) | \cdots | \Upsilon(h_{m,d-1}) \Big] \in \mathbb{Z}^{dm\times dm},$$

where $\Upsilon(\sum_{0\le i<dm} c_i x^i) = (c_0, \ldots, c_{dm-1})^t \in \mathbb{Z}^{dm\times 1}$.

We next examine the sizes of the entries of $D_g^{(m)}$, which are all coefficients of the polynomials $h_{i,j}$ for $1 \le i \le m$ and $0 \le j < d$. We prove that $\|h_{i,j}\|_\infty \le d^{2m-2}\beta^m n^{(m-1)/2} 2^{(n+1)(m-1)}$. Since $h_{1,0} = g^{m-1}$ is a divisor of $f_1$, each coefficient of $h_{1,j}$ has absolute value at most $\beta$ for $0 \le j < d$. For $2 \le i \le m$, $\|g \cdot h_{i,0}\|_\infty \le d^2\beta \cdot \max_{0\le l<d} \|h_{i-1,l}\|_\infty$. Dividing by $g$ and applying Mignotte's(1974) bound on the

size of coefficients of factors, it follows that

$$\|h_{i,0}\|_\infty \le d^2\beta 2^n\sqrt{n}\cdot \max_{0\le l<d}\|h_{i-1,l}\|_\infty.$$

More generally, for $1 \le i \le m$ and $0 \le j < d$,

$$\|h_{i,j}\|_\infty \le \|h_{i,0}\|_\infty + d\max_{0\le l<d}\|h_{i-1,l}\|_\infty \le d^2\beta 2^{n+1}\sqrt{n}\cdot \max_{0\le l<d}\|h_{i-1,l}\|_\infty.$$

By a trivial induction we obtain, for $1 \le i \le m$ and $0 \le j < d$, that

$$\|h_{i,j}\|_\infty \le (d^2\beta\sqrt{n}2^{n+1})^{m-1}\beta \le d^{2m-2}\beta^m n^{(m-1)/2}2^{(n+1)(m-1)}.$$

To actually compute the entries of $D_g^{(m)}$ is quite simple. We determine the cost in terms of operations in $\mathbb{Z}$, without regard for their sizes, since our ultimate goal is a modular algorithm. For $i = 1$ we can compute $h_{i,j} = x^j g^{m-1}$ with $O(\mathrm{M}(dm))$ operations in $\mathbb{Z}$ (we are given $g^m$). For $1 < i \le m$, assume we have computed $h_{i_0,j_0}$, with $1 \le i_0 < i$ and $0 \le j_0 < d$. First, compute $\sum_{0\le l<j} x^{j-l-1}h_{i-1,l}$ for $0 \le l < d$. This can be accomplished with $O(d^2m)$ operations in $\mathbb{Z}$. Computing $h_{i,0}$ can then be computed with $O(d^2m + \mathrm{M}(dm))$ additional operations in $\mathbb{Z}$. Moreover, using the precomputed $\sum_{0\le l<j} x^{j-l-1}h_{i-1,l}$ for $0 \le l < d$, we can find all of $h_{i,j}$, for $1 \le j < d$, with a total of $O(\mathrm{M}(dm) + d^2m)$ operations in $\mathbb{Z}$. Thus, we can determine all coefficients of all $h_{i,j}$'s, and hence all entries in the transition matrix $D_g^{(m)}$, with $O(m\mathrm{M}(dm)+d^2m^2)$ operations in $\mathbb{Z}$. We have shown the following.

**Lemma 5.27.** *Given a matrix $E \in \mathbb{Z}^{n\times n}$ in elementary normal form, with $\|E\|_\infty \le \beta$, there exists a matrix $D \in \mathbb{Z}^{n\times n}$ such that $D^{-1}ED$ is in rational Jordan form, and such that $\|D\|_\infty \le n^{(3n-1)/2}2^{n^2-1}\beta^n$. The matrix $D$ can be computed with $O(n\mathrm{M}(n))$ operations in $\mathbb{Z}$.*

By Lemma 5.26 there exists a matrix $R \in \mathbb{Z}^{n\times n}$ such that $E = R^{-1}SR = R^{-1}U^{-1}TUR$ is in elementary rational form, while, by Lemma 5.27 there is a $D \in \mathbb{Z}^{n\times n}$ such that $Q = D^{-1}ED$ is in rational Jordan from. Letting $P = URD$, we find $P^{-1}TP = Q$ is in rational Jordan form.

We now discuss how to actually compute $R$, $D$, and hence $P = URD$. Since the entries of $R$ and $D$ are *polynomials* in the entries of $S$, we can compute them modulo the same collection $\Lambda \subseteq \mathbb{N}$ of primes we used in the computation of the transition matrix $U$ and the Frobenius form $S$. The only stipulation is that the collection of primes have product at least $2\xi$, where $\xi$ is defined by

$$\begin{aligned} n^2 \cdot \| \det(H)U\|_\infty \det(H) \cdot \|R\|_\infty \|D\|_\infty &\leq 2n^2 \cdot \gamma\delta \cdot \beta \cdot n^{(3n-1)/2} 2^{n^2-1}\beta^n \\ &= 2^{3n^2+3} e^{n^2+n/2} \alpha^{4n^2+n-1} n^{3n^2+6n+9/2} \\ &\stackrel{\text{def}}{=} \xi. \end{aligned}$$

Here $H \in \mathbb{Z}^{n\times n}$ is a modular cyclic transition matrix as in Theorem 4.6, such that $\| \det(H)U\|_\infty \leq \gamma$ and all denominators of entries of $U$ divide $\det(H)$. Thus, all entries of $URD$ have entries with denominators dividing $\det(H)$, and numerators with absolute values at most $n^2\gamma \cdot \|R\|_\infty \cdot \|D\|_\infty$. Our choice of $\xi$ ensure that all entries of $P = URD$ can be represented uniquely modulo a collection of primes with product at least $2\xi$.

We compute the Frobenius form of $T$ using Theorem 4.6, but making use of a larger collection $\Lambda_0$ of primes. In particular, we insist that the primes in $\Lambda_0$ have product at least $2\xi$ (instead of $2\gamma\delta$ in Theorem 4.6). Since both $\log(\gamma\delta)$ and $\log(\xi)$ are in $O(n^2(\log n + \log \alpha))$, by Corollary 4.7 this does not change the asymptotic cost of Theorem 4.6. A choice of $\ell = 6 + \log\log \xi$ ensures that there are good primes $p$, with $2^{\ell-1} < p < 2^\ell$, which have product at least $2\xi$.

Modulo any prime $p \in \Lambda$, we can compute $R \bmod p$ with $O(n\mathrm{M}(n))$ operations in $\mathbb{Z}_p$ as shown in Lemma 5.26. The transition matrix $D$ from $E$ to $Q$ can be computed within this number of operations. Thus, the cost of computing $URD \bmod p$ is dominated by that of multiplying these matrices together, and hence can be done with $O(\mathrm{MM}(n))$ operations in $\mathbb{Z}_p$. If all the primes $p \in \Lambda$ satisfy $2^{\ell-1} < p < w^\ell$ then we can compute $URD$ modulo all these primes with $O(n^3\mathrm{M}(n)(\log n + \log \alpha) \cdot \mathrm{M}(\ell)/\ell)$ bit operations.

**Theorem 5.28.** *Let $T \in \mathbb{Z}^{n\times n}$ with $\|T\|_\infty \leq \alpha$. The Las Vegas algorithm discussed above to compute the rational Jordan form $Q \in \mathbb{Z}^{n\times n}$ of $T$, and a transition matrix $P \in \mathbb{Q}^{n\times n}$ such that $S = P^{-1}TP$ requires an expected number of* $O((\mathrm{MM}(n)\log n +$

*$n\mathrm{M}(n))\cdot n^2(\log n+\log\alpha)\cdot\mathrm{M}(\ell)/\ell+n^2\mathrm{M}(n(\log n+\log\alpha))\ell)$ or $O\tilde{~}(\mathrm{MM}(n)\cdot n^2\log\alpha)$ bit operations. Using standard arithmetic, an expected number of $O(n^5(\log n+\log\alpha)\ell+n^4\log^2\alpha)$ bit operations are required. In the standard arithmetic case, entries in $P$ are represented modulo a collection of $O(n^2(\log n+\log\alpha)/\ell)$ primes between $2^{\ell-1}$ and $2^\ell$. To reconstruct $P\in\mathbb{Q}^{n\times n}$ requires an additional $O(n^6(\log n+\log\alpha)^2)$ bit operations. Here $\ell=\Theta(\log n+\log\log\alpha)$.*

Finally, we note that the cost of both finding the rational Jordan form *and* a transition matrix to that form is still dominated by the cost of actually factoring the minimal polynomial of the matrix.

**Corollary 5.29.** *Given $T\in\mathbb{Z}^{n\times n}$ with $\|T\|_\infty\le\alpha$, we can find the rational Jordan form $S\in\mathbb{Z}^{n\times n}$ of $T$ and a transition matrix $U\in\mathbb{Q}^{n\times n}$ such that $S=U^{-1}TU$ by a Las Vegas type probabilistic algorithm requiring an expected number of $O(n^{6+\epsilon}(\log n+\alpha)^{2+\epsilon})$ bit operations for any $\epsilon>0$. Using standard arithmetic, an expected number of $O(n^8(\log n+\log\alpha)^3)$ bit operations are required.*

# Chapter 6.

# Bases for finite fields

In this chapter three efficient probabilistic algorithms to find a normal element in a finite field are presented. The first two of these algorithms are directly dependent upon a density estimate for normal elements in any finite field and a fast test for normalcy. This estimate is derived from Theorems 3.12 and 3.15 shown in Section 3, using a result of Ore (1934) linking the number of normal elements to the Euler totient function for polynomials over finite field. The third algorithm is based upon the algorithms for computing the Frobenius form of a matrix, presented in Chapter 2. We also examine the more difficult problem of finding primitive normal elements in finite fields. These are elements which are simultaneously normal over a subfield and generate the multiplicative group of the entire field. A density estimate similar to that in Theorem 3.15 yields a probabilistic polynomial-time reduction from finding primitive normal elements to finding primitive elements.

Let $q$ be a prime power and $\mathsf{K} = \mathbb{F}_q$, and for $n > 1$, let $\mathsf{F}$ be an algebraic extension field of $\mathsf{K}$, with $[\mathsf{F} : \mathsf{K}] = n$ so $\mathsf{F} \cong \mathbb{F}_{q^n}$. If $\alpha \in \mathsf{F}$, and the conjugates $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{n-1}}$ of $\alpha$ form a basis for $\mathsf{F}$ as a vector space over $\mathsf{K}$, then this is called a *normal basis*. We call $\alpha$ a *normal element* (of $\mathsf{F}$ over $\mathsf{K}$).

Normal bases are useful for implementing fast arithmetic in finite fields, in particular exponentiation. Of special interest is $q = 2$ and $n$ reasonably large; as an example, the Diffie & Hellman key exchange is based on exponentiation in $\mathbb{F}_{2^n}$. Algorithms and possible semiconductor implementations are given by Laws & Rushforth (1971), Wang *et al.* (1985), Beth *et al.* (1986), Agnew *et al.* (1988), Stinson (1990), and von zur Gathen (1992). The basic assumption in that work is that computing $q$th powers in $\mathbb{F}_{q^n}$ is for free (i.e., of negligible cost compared to a general multiplication in $\mathbb{F}_{q^n}$; only $q = 2$ is considered in all but the last of these papers). The assumption can be justified if a normal element is given, since then for an arbitrary $u = \sum_{0\le i<n} u_i\alpha^{q^i} \in \mathsf{F}$, with $u_0, \ldots, u_{n-1} \in \mathsf{K}$, we have

$$u^q = (\sum u_i\alpha^{q^i})^q = \sum_{0\le i<n} u_{i-1}\alpha^{q^i}$$

with $u_{-1} = u_{n-1}$. In other words, the coordinates of $u^q$ are a cyclic shift of those of $u$.

This chapter addresses the question: how can we find a normal element efficiently? More generally, we consider how to find an element of a given *additive order*; the additive order of an element is a natural generalization of the property of normalcy, first developed by Ore (1934) (see Section 6.1 for the definition). We give an algorithm to construct elements of a given additive order efficiently. Hensel (1888) pioneered the study of normal bases for finite fields and proved that they always exist. The first algorithm we present is essentially Hensel's algorithm (or rather, constructive proof) in Section 6.1. This algorithm is asymptotically very fast, but when using standard polynomial arithmetic one can do better. We adapt the methods of Wiedemann (1986), which he uses to solve sparse systems of linear equations, to obtain an algorithm which is considerably faster than our implementation of Hensel's method when using standard polynomial arithmetic. It is also easily modified to run with only linear space at the expense of multiplying the running time by the logarithm of the field size. Our adaptation of Hensel's algorithm requires quadratic space.

Eisenstein (1850) had already noted that normal bases always exist. Hensel, and also Ore (1934), demonstrate a formula for the exact number of these bases. Ore's approach is developed into more constructive proofs of the normal basis theorem in several textbooks (for example, van der Waerden 1966, Section 67, and Albert 1956, Section 4.15); these all use some linear algebra calculations. Schwarz (1988) has given a new proof along these lines, and several recent papers have translated this approach into algorithms. Sidel'nikov (1988) deals with the case where $n$ divides one of $p$ (the characteristic of $\mathsf{K}$), $q+1$, or $q-1$ and Stepanov & Shparlinsky (1987) with the case $\gcd(n, q) = 1$ or $q > n$. Semaev (1989) solves the general problem; also he observes that the general case reduces to the case where $n$ is a prime power. These three papers reduce, in deterministic polynomial time, the construction of a normal basis to factoring polynomials; hence, at the current state of the art for factoring, they yield polynomial-time algorithms (requiring $(n \log q)^{O(1)}$ operations in $\mathsf{K}$) that are probabilistic in general, and deterministic if $q$ is small (requiring $(nq)^{O(1)}$ operations in $\mathsf{K}$). Many of the results we present here were first shown (and submitted for publication) in 1987, and appear in (von zur Gathen & Giesbrecht, 1990).

The best general result to date is Lenstra's (1991) deterministic polynomial-time algorithm, given as a subroutine in his interesting solution of a more difficult problem, namely the construction of explicit isomorphisms between finite fields of the same cardinality in deterministic polynomial time. Lenstra's assumptions amount (within deterministic polynomial time) to requiring an irreducible polynomial of degree $n$ as input. His sufficient condition is also necessary since an irreducible polynomial is computed. We note that finding a normal element can also be accomplished using an algorithm for finding the Frobenius form of a matrix and a transition matrix to this Frobenius form. This yields an algorithm of the same generality and approximately the same (deterministic) cost as that of Lenstra's.

We present three algorithms for our problem in Section 6.1. All are based on well-known properties of normal bases. Our standard reference for material in this section is (Lidl & Niederreiter, 1983).

The first two algorithms simply choose a random element in $\mathsf{F}$ and test whether it is normal over $\mathsf{K}$. We rely on a density estimate of normal elements in finite fields which is provided by the lower bound on the totient function $\Phi_{\mathsf{K}}$ for $\mathsf{K}[x]$ discussed in Theorems 3.12 and 3.15. This shows that the probability of choosing a normal element is indeed very high: $\Omega(1/\log_q n)$ for $n > q$, and at least $1/e$ otherwise. The difference between the two methods is how we decide whether the chosen element is normal. The first test is from Hensel (1888) and requires $O\tilde{\ }(n^2 + n \log q)$ operations in $\mathsf{K}$ using fast polynomial arithmetic. However, when standard polynomial arithmetic is used this method requires $O(n^4 + n^2 \log q)$ operations in $\mathsf{K}$. The second test also requires $O\tilde{\ }(n^2 + n \log q)$ operations in $\mathsf{K}$ asymptotically, but using standard polynomial arithmetic, $O(n^3 + n^2 \log q)$ operations in $\mathsf{K}$. It also has the advantage of requiring space $O(n)$, as opposed to space $O(n^2)$ required by Hensel's test. This last advantage is rather marginal since we usual want a normal *basis*, which requires space $O(n^2)$ to write out, and not just a normal element. This test is based on the work of Wiedemann (1986), and is of the Las Vegas probabilistic type. Assuming our initial randomly selected element is normal, it will be recognized as such with probability at least 1/2.

We also adapt our algorithms for the Frobenius form of Chapter 2 to finding a normal basis for a finite field. We observe that this problem is reducible to that of constructing the $n \times n$ matrix over $\mathsf{K}$ which acts as the Frobenius map, and then finding the Frobenius form of this matrix and a transition matrix to this form. The columns of the transition matrix, appropriately mapped into $\mathsf{F}$, form a normal basis for $\mathsf{F}$ over $\mathsf{K}$. Using the algorithms of Chapter 2 this method requires an expected number of $O(n^3 + n^2 \log q)$ operations in $\mathsf{K}$ using standard polynomial arithmetic. The algorithm is of the Las Vegas type.

An element $\alpha \in \mathsf{F}$ is *primitive* if and only if every non-zero element of $\mathsf{F}$ is a power of $\alpha$. No (probabilistic) polynomial-time algorithm is known to test primitivity, or to generate a primitive element. For every $n$ and $q$, there exists a primitive element of $\mathsf{F}$ which is normal over $\mathsf{K}$: a *primitive normal element* (Lenstra & Schoof, 1987). Stepanov & Shparlinsky (1989) give a deterministic reduction from finding primitive normal elements to finding primitive elements; their reduction uses time linear in $\log q$, but exponential in $n$. In Section 6.3, we give a probabilistic polynomial-time (in $n \log q$) reduction from finding a primitive normal element in $\mathsf{F}$ to finding a primitive elements in $\mathsf{F}$. This is based on further estimates of finite field analogues of number-theoretic functions, given in Section 6.2.

## 6.1 Finding Normal Elements

As in the introduction to this chapter, we assume throughout this section that $q$ is a prime power and $\mathsf{K} = \mathbb{F}_q$. Let $h \in \mathsf{K}[x]$ be monic and irreducible of degree $n$, and $\mathsf{F} = \mathsf{K}[x]/(h)$, an algebraic extension field of degree $n$ over $\mathsf{K}$. Define $\alpha = x \bmod h \in \mathsf{F}$, and $(1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$ the standard power basis of $\mathsf{F}$ over $\mathsf{K}$. We want to find a normal basis of $\mathsf{F}$ over $\mathsf{K}$. For our first two algorithms at least, and arguably for the third in a rather indirect way, our approach is to choose an element $\beta \in \mathsf{F}$ at random and quickly determine whether or not it is normal over $\mathsf{K}$.

First, we show that the probability of choosing a normal element is quite high. We denote by $\nu(q,n)$ the number of normal elements in $\mathsf{F}$ over $\mathsf{K}$. Recall from Section 3 the Euler totient function for $\mathsf{K}[x]$: for any $f \in \mathsf{K}[x]$, $\Phi_{\mathsf{K}}(f) \in \mathbb{N}$ is the number of polynomial in $\mathsf{K}[x]$ with degree less than $\deg f$ and relatively prime to $f$. Ore (1934) shows that the number of normal elements in $\mathsf{F}$ over $\mathsf{K}$ is $\Phi_{\mathsf{K}}(x^n - 1)$ (see Lidl & Niederreiter 1983, Theorem 3.73). Applying Theorem 3.15, we immediately obtain the following theorem, which is the basis for our probabilistic algorithms for finding normal bases for finite fields.

**Theorem 6.1.** *For $q$ a prime power and $n > 1$, the probability $\nu(q,n)/q^n$ of an element chosen randomly from $\mathsf{F} \cong \mathbb{F}_{q^n}$ being normal over a subfield $\mathsf{K} \cong \mathbb{F}_q$ satisfies $\nu(q,n) \geq 1/e$ if $n \leq q$, and $\nu(q,n) > 1/(5\log_q n)$ if $n > q$.*

## Algorithm 1: Hensel's method

We couple Theorem 6.1 with fast tests for normalcy to obtain algorithms for finding normal elements in $\mathsf{F}$ over $\mathsf{K}$. The first such test we present is based on Hensel's (1888) criterion.

**Algorithm:** `NormalHensel`

Input: a monic irreducible polynomial $h \in \mathsf{K}[x]$ of degree $n$;

Output: An element $\mathsf{F} = \beta \in \mathsf{K}[x]/(h)$ normal over $\mathsf{K}$, or "failure";

(1) Set $k = 6$ if $n \leq q$ and $k = 10\log_q n$ if $n > q$, and set a counter $c$ to 1;

(2) Choose $\beta \in \mathsf{F} = \mathsf{K}[x]/(h)$ at random;

(3) For $0 \leq i < n$, compute $\beta_i = \beta^{q^i} \in \mathsf{F}$;

(4) Set $f = (\sum_{0 \leq i < n} \beta_i z^i) \in \mathsf{F}[z]$, where $z$ is an indeterminate, and compute $g = \gcd(f, z^n - 1) \in \mathsf{F}[z]$;

(5) If $g = 1$, return $\beta$ and stop. Otherwise increase $c$ by 1. If $c \leq k$, go to step (2), else return "failure" and stop;

**End.**

**Theorem 6.2.** *Algorithm* `NormalHensel` *works correctly as described in "Output"; failure occurs with probability at most 1/2. The algorithm uses* $O(M(n)^2 \log n + \mathrm{M}(n) \log q)$ *operations in* $\mathsf{K}$ *if* $n \le q$, *and* $O((M(n)^2 \log n + \mathrm{M}(n) \log q) \log_q n)$ *operations in* $\mathsf{K}$ *if* $n > q$.

**Proof.** Hensel (1888) proves that $\beta$ is normal if and only if $g = 1$ in step (4), and hence an eventual output $\beta$ is correct (see Lidl & Niederreiter 1983, Theorem 2.39).

By Theorem 6.1 the probability of choosing a normal element is at least $1/e$ if $n \le q$ and at least $1/5 \log_q n$ if $n > q$. In step (1) we set $k$ to be twice the lower bound on expected number of random choices to obtain a normal element, so by Markov's inequality we expect to find a normal element among any $k$ randomly chosen elements with probability at least 1/2.

We can compute all $\beta_i$ in step (3) using the algorithm of von zur Gathen & Shoup (1992) with $O(\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q)$ operations in $\mathsf{K}$. The greatest common divisor in step (4) can be calculated in $O(M(n) \log n)$ operations in $\mathsf{F}$ or $O(\mathrm{M}(n)^2 \log n)$ operations in $\mathsf{K}$. $\square$

## Algorithm 2: Applying Wiedemann's Techniques

When standard polynomial arithmetic is employed (i.e., $\mathrm{M}(n) = n^2$), the algorithm based on Hensel's test proposed above unfortunately requires an expected number of $O\tilde{}(n^4 + n^2 \log q)$ operations in $\mathsf{K}$ to find a normal element of $\mathsf{F}$ over $\mathsf{K}$. This is improved considerably when some techniques of Wiedemann (1986) are used. Recall the definition of a *linearly generated sequence* in a vector space $V$ over $\mathsf{K}$ as a sequence $\beta_0, \beta_1, \beta_2, \ldots \in V$ such that there exist $c_0, \ldots, c_m \in \mathsf{K}$ (not all zero) such that for all $j \ge 0$

$$c_0\beta_j + c_1\beta_{j+1} + \cdots + c_m\beta_{j+m} = 0.$$

For a good introduction to linearly generated sequences and computational methods associated with them, the reader is referred to (Kaltofen, 1992). The polynomial $\sum_{0 \le i \le m} c_i x^i \in \mathsf{K}[x]$ is called a *generating polynomial* of the sequence. If a sequence over $\mathsf{K}$ is linearly generated then all its generating polynomials, together with zero, form an ideal in $\mathsf{K}[x]$ (see Kaltofen 1992, Theorem 1). Thus, there is a unique monic polynomial of minimal degree which generates the sequence, called the *minimal polynomial* of the sequence.

The field $\mathsf{F}$ is certainly a vector space over $\mathsf{K}$ and for any $\beta \in \mathsf{F}$, the sequence $\beta_0, \beta_1, \ldots$ defined by $\beta_i = \sigma^i(\beta) = \beta^{q^i}$ is linearly generated: it satisfies $\beta_{i+n} - \beta_i = 0$, or, equivalently, is generated by the polynomial $x^n - 1$. The degree of the minimal polynomial of $\beta$ is the dimension of the vector space spanned by $\beta$ and its conjugates, and $\beta$ is normal exactly when this dimension is $n$. Since the minimal polynomial of the sequence generated by $\beta$ divides $x^n - 1$, $\beta$ is normal if and only if the minimal polynomial of the sequence $\beta, \sigma(\beta), \sigma^2(\beta), \ldots$ equals $x^n - 1$.

To avoid doing arithmetic in $\mathsf{F}$ we employ another device used by Wiedemann (1986). Suppose for now that $q \geq 2n$. Choose a vector $u = (u_0, \ldots, u_{n-1})$ randomly from $\mathsf{K}^n$. The vector $u$ induces a linear map from $\mathsf{F}$ to $\mathsf{K}$ via the inner product. If $\gamma = \sum_{0 \leq i < n} c_i \alpha^i$ (where $c_0, \ldots, c_{n-1} \in \mathsf{K}$ and $\alpha$ is a root of a monic irreducible $h \in \mathsf{K}[x]$; recall that we represent $\mathsf{F}$ by the standard power basis generated by $\alpha$) then let $\langle \gamma, u \rangle = \sum_{0 \leq i < n} c_i u_i \in \mathsf{K}$. Consider the sequence $b_0, b_1, b_2, \ldots \in \mathsf{K}$. It is easily seen that $x^n - 1$ is a generating polynomial for this sequence (see Kaltofen 1992, example 3), so in particular the minimal polynomial always divides $x^n - 1$. Moreover, Kaltofen (1992), page 19, Corollary 2, shows that with probability at least $1 - n/q \geq 1/2$ the minimal polynomial of $b_0, b_1, \ldots$ is equal to $x^n - 1$. We are now ready to state our algorithm for finding a normal element in $\mathsf{F}$ over $\mathsf{K}$.

**Algorithm:** `NormalWiedemann`

Input: a monic irreducible polynomial $h \in \mathsf{K}[x]$ of degree $n$;

Output: An element $\beta \in \mathsf{F} = \mathsf{K}[x]/(h)$ normal over $\mathsf{K}$, or "failure";

(1) Set $k = 24$ if $n \leq q$ and $k = 48 \log_q n$ if $n > q$, and set a counter $c$ to 1;

(2) Choose $\beta \in \mathsf{F} = \mathsf{K}[x]/(h)$ at random;

(3) Choose a vector $u \in \mathsf{K}^n$ at random;

(4) For $0 \leq i < 2n$, compute $b_i = \langle u, \beta^{q^i} \rangle \in \mathsf{K}$;

(5) Compute the minimal polynomial $f \in \mathsf{K}[x]$ of the linearly recurring sequence $b_0, b_1, \ldots \in \mathsf{K}$;

(5) If $f = x^n - 1$, return $\beta$ and stop. Otherwise increase $c$ by 1. If $c \leq k$, go to step (2), else return "failure" and stop;

**End.**

**Theorem 6.3.** *Algorithm* `NormalWiedemann` *works correctly as described in "Output"; when $q \geq 2n$, failure occurs with probability at most 1/2. The algorithm uses $O(n^3 + n^2 \log q)$ operations in $\mathsf{K}$ if $n \leq q$, or $O(n^3 \log_q n + n^2 \log q)$ operations in $\mathsf{K}$ if $n > q$. Space $O(n^2)$ is required. Alternatively, using space $O(n)$, the algorithm can be executed with $O(n^3 \log q)$ operations in $\mathsf{K}$ if $n \leq q$, or $O(n^3 \log n)$ operations in $\mathsf{K}$ if $n > q$. All these times assume standard polynomial multiplication, that is,* $\mathrm{M}(n) = n^2$.

**Proof.** As discussed above, $\beta$ is normal if and only if the minimal polynomial $f \in \mathsf{K}[x]$ of the linearly generated sequence $\beta, \beta^q, \ldots$ is $x^n - 1$. Moreover, the minimal polynomial of $b_0, b_1, \ldots$ is a divisor of $f$. Hence, if $b_0, b_1, \ldots$ has minimal polynomial $x^n - 1$ then $\beta$ is normal, so the output of the algorithm is always correct.

By Theorem 6.1, $\beta$ is normal with probability at least $1/(12 \log_q n)$ if $n > q$ or with probability at least 1/6 when $n \leq q$. Assuming $\beta$ is normal, then we have seen above that with probability at least 1/2 the minimal polynomial of the linearly generated sequence $b_0, b_1, \ldots$ is also $x^n - 1$ and $\beta$ is reported to be normal. Thus, the probability of success for each choice of $\beta$ and $u$ is at least $1/(12 \log_q n)$ if $n > q$, or at least 1/12 when $n \leq q$. We make a number of choices of $\beta$ equal to twice the lower bound on the expected number of random choices required to obtain a normal element, as specified by our choice of $k$ in step (1). This insures success with probability at least 1/2, by Markov's inequality.

The dominant cost of the algorithm is that of computing the sequence $b_0, b_1, \ldots,$ and more basically, of computing $\beta_{i+1} = \beta_i^q$ from $\beta_i$. Given $\beta_i$, we present two methods for computing $\beta_{i+1}$ as follows. In the first we compute a matrix representation $Q \in \mathsf{K}^{n\times n}$ of the Frobenius map $\sigma$ with respect to the basis $1, \alpha, \ldots, \alpha^{n-1}$. This is accomplished by computing $\alpha^{iq}$ for $0 \leq i < n$, which can be performed with $O(n^2 \log q)$ operations in $\mathsf{K}$. If $\alpha^{iq} = \sum_{0\leq j<n} a_{ij}\alpha^j$ for some $a_{ij} \in \mathsf{K}$ then let $Q_{i+1,j+1} = a_{ij}$ for $0 \leq i, j < n$. Then, for any $\gamma = \sum_{0\leq i<n} c_i\alpha^i \in \mathsf{F}$, we can compute

$$\gamma^q = \sum_{0\leq i<n} c_i\alpha^{qi} = \sum_{0\leq i<n} \bar{c}_i\alpha^i,$$

for $\bar{c}_0, \ldots, \bar{c}_{n-1} \in \mathsf{K}$, by the matrix vector product $Q(c_0, \ldots, c_{n-1})^t = (\bar{c}_0, \ldots, \bar{c}_{n-1})^t$.

The matrix $Q$ is computed once and for all at the beginning of the algorithm, after which we can compute $\beta_i$ from $\beta_{i+1}$ with $O(n^2)$ operations in $\mathsf{K}$. Alternatively, we can compute $\beta_{i+1}$ from $\beta_i$ with $O(n^2 \log q)$ operations in $\mathsf{K}$ by repeated squaring, which has the advantage of requiring only linear space.

To find the minimal polynomial of the linearly generated sequence $b_0, b_1, \ldots$, we make use of the Berlekamp-Massey algorithm (Berlekamp, 1968). The prefix $b_0, \ldots, b_{2n-1}$ uniquely determine the minimal polynomial of the sequence since we know the minimal polynomial has degree at most $n$ (see Kaltofen 1992, Corollary 1 to Theorem 4). The Berlekamp-Massey algorithm requires $O(n^2)$ operations in $\mathsf{K}$ to find this minimal polynomial, using space $O(n)$. $\square$

In the algorithm `NormalWiedemann` we have assumed that $q > 2n$ and used this fact to bound the probability that the minimal polynomial of $b_0, b_1, \ldots$ equals that of $\beta_0, \beta_1, \ldots$ When $\mathsf{K}$ is very small, this probability might in fact be very low. To remedy this, we can choose the entries of $u$ from an algebraic extension $\mathsf{F}$ of degree $\mu = \lceil \log_q 2n \rceil$ over $\mathsf{K}$, which does contain $2n$ elements. The extension field $\mu$ is represented as $\mathsf{K}[x]/(\psi)$ where $\psi \in \mathsf{K}[x]$ is monic and irreducible of degree $\mu$. The algorithm of Shoup (1990) can construct such a $\psi$ with $O\tilde{~}(n^{1/2})$ operations in $\mathsf{K}$. Thus, we compute $b_0, b_1, \ldots \in \mathsf{F}$ and determine the minimal polynomial of this sequence using the Berlekamp-Massey algorithm run over $\mathsf{F}$. The time required by the algorithm is asymptotically the same as that required by the algorithm run over larger fields.

**Theorem 6.4.** *Algorithm* `NormalWiedemann`, *as modified above for fields* $\mathsf{K}$ *of size* $q < 2n$, *works correctly as described in "Output"; failure occurs with probability at most* $1/2$. *The algorithm uses* $O(n^3 + n^2 \log q)$ *operations in* $\mathsf{K}$ *if* $n \le n$, *or* $O(n^3 \log_q n + n^2 \log q)$ *operations in* $\mathsf{K}$ *if* $n > q$. *Space* $O(n^2)$ *is required. Alternatively, using space* $O(n)$, *the algorithm can be executed with* $O(n^3 \log n)$ *operations in* $\mathsf{K}$ *if* $n \le q$, *or* $O(n^3 \log_q n)$ *operations in* $\mathsf{K}$ *if* $n > q$. *All these times assume standard polynomial multiplication, that is,* $\mathrm{M}(n) = n^2$.

**Proof.** The only difference is the cost is the time required to compute the inner products $b_i = (u, \beta_i) \in \mathsf{F}$ for $0 \le i < n$ and to run the Berlekamp-Massey algorithm over $\mathsf{F}$ instead of $\mathsf{K}$. Each of these steps require $O(n^2 \log_q n)$ or $O(n^2 \log n)$, which is dominated by the cost of computing the conjugates of $\beta$. $\square$

## Algorithm 3: Employing Frobenius Normal Forms

Finally, we note that algorithms (both ours and the slower but deterministic ones of Ozello (1987) and Lüneburg (1987)) for finding the Frobenius normal form of a matrix can be applied to find normal elements in a finite field. While these methods do not provide any particular advantage over the techniques described above, we present them here for completeness. We discuss them only using standard polynomial and matrix arithmetic.

As usual, we regard $\mathsf{F}$ as a $\mathsf{K}$-vector space of dimension $n$. $\mathsf{F}$ is assumed to be represented on a power basis $(1, \alpha, \alpha^2, \ldots, \alpha^{n-1})$, where $\alpha = x \bmod h$ and $h \in \mathsf{K}[x]$ is monic and irreducible of degree $n$. Thus we have the trivial isomorphism $\varphi : \mathsf{F} \to \mathsf{K}^{n \times 1}$ mapping $\gamma = \sum_{0 \le i < n} c_i \alpha^i$, to $\varphi(\gamma) = (c_0, \ldots, c_{n-1})^t \in \mathsf{K}^{n \times 1}$. As discussed in the proof of Theorem 6.3, the Frobenius map $\sigma$ on $\mathsf{F}$ acts via $\varphi$ on $\mathsf{K}^{n \times 1}$ by matrix-vector product by a matrix $Q \in \mathsf{K}^{n \times n}$, where $\varphi(\gamma^q) = Q\varphi(\gamma) \in \mathsf{K}^{n \times 1}$. A normal element $\beta \in \mathsf{F}$ over $\mathsf{K}$ is one whose conjugates form a basis for $\mathsf{F}$, or equivalently such that $\dim \mathrm{Orb}(Q; \varphi(\beta)) = n$. That is, $\varphi(\beta)$ is a cyclic vector in $\mathsf{K}^{n \times 1}$ under $Q$. The

Frobenius form $S \in \mathsf{K}^{n\times n}$ of $Q$ is a companion matrix of $x^n - 1$ ($Q$ has minimal and characteristic polynomial equal to $x^n - 1$). Also, all transition matrices $U \in \mathsf{K}^{n\times n}$ such that $U^{-1}TU = S$ have the structure

$$U = \left[v|Qv|Q^2v|\cdots|Q^{n-1}v\right] \in \mathbb{F}^{n\times n}$$

for some vector $v \in \mathsf{K}^{n\times 1}$. Moreover, since $U$ is non-singular (or, more fundamentally, since $Q$ has only one invariant factor), $v$ must be a cyclic vector under $Q$, whence $\varphi^{-1}(v) \in \mathsf{F}$ is a normal element in $\mathsf{F}$. Note that in this application it is very important that the transition matrix to the Frobenius form lie over $\mathsf{K}$, and not in an extension field, since we will read the normal element from the transition matrix.

**Theorem 6.5.** *Given an irreducible polynomial $h \in \mathsf{K}[x]$ of degree $n$, we can find a normal element in $\mathsf{F} = \mathsf{K}[x]/(h)$ over $\mathsf{K}$ with $O(n^4 + n^2 \log q)$ operations by a deterministic algorithm. If $q \geq n^2$ we can find a normal element in $\mathsf{F}$ over $\mathsf{K}$ by a Las Vegas probabilistic algorithm requiring $O(n^3)$ operations in $\mathsf{K}$, which returns the correct answer with probability at least $1/4$. If $q < n^2$, there is a Las Vegas algorithm requiring $O(n^3 \log_q n)$ operations in $\mathsf{K}$ are required and returning the correct answer with probability at least $1/8$. These times assume standard matrix and polynomial arithmetic.*

**Proof.** We construct the matrix $Q$ as in Theorem 6.3. This requires $O(n^2 \log q)$ operations in $\mathsf{K}$. We then compute the Frobenius form $S \in \mathsf{K}^{n\times n}$ of $Q$ and an invertible $U \in \mathsf{K}^{n\times n}$ such that $U^{-1}QU = S$. By the algorithm of Ozello (1987), this requires $O(n^4)$ operations in $\mathsf{K}$ deterministically. The fast Las Vegas algorithms rely on our results in Chapter 3. If $q \geq n^2$, Theorem 3.5 shows how to find the Frobenius form of $Q$ and a transition matrix $U$ with $O(n^3)$ operations in $\mathsf{K}$, where the correct answer is returned with probability at least $1/4$. If $q < n^2$, Theorem 3.10 shows how to compute the Frobenius form of $Q$ and $U \in \mathsf{K}^{n\times n}$ with $O(n^3 \log_q n)$ operations in $\mathsf{K}$, where the correct answer is returned with probability at least $1/8$. $\square$

## Finding Normal Bases

Combining the above theorems with a fast algorithm for finding irreducible polynomials yields a quick method of finding a normal basis for an algebraic extension field $\mathsf{F}$ of degree $n$ over $\mathsf{K}$.

**Corollary 6.6.** *Given an integer $n \geq 2$, we can construct a normal basis for $\mathsf{F} \cong \mathbb{F}_{q^n}$ over $\mathsf{K}$ with a probabilistic algorithm using an expected number $O((\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q) \log_q n)$ operations in $\mathsf{K}$ if $n > q$, or $O(\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q)$ operations in $\mathsf{K}$ otherwise.*

**Proof.** The algorithm of Shoup (1993) finds an irreducible polynomial of degree $n$ in $\mathsf{K}[x]$ with an expected number of $O(\mathrm{M}(n)^2 + \mathrm{M}(n) \log q)$ operations in $\mathsf{K}$. We then use such an irreducible polynomial with the algorithm `NormalHensel` above. This will find a normal element $\alpha \in \mathsf{F}$ over $\mathsf{K}$ in an expected number $O((\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q) \log_q n)$ operations in $\mathsf{K}$ when $n > q$, or $O(\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q)$ operations in $\mathsf{K}$ if $n \leq q$. Then $\alpha, \alpha^q, \ldots, \alpha^{q^{n-1}}$ forms the desired normal basis of $\mathsf{F}$ over $\mathsf{K}$, and this basis can be computed from $\alpha$ using the algorithm of von zur Gathen & Shoup (1992), with $O(\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q)$ operations in $\mathsf{K}$. $\square$

## Finding Elements of Given Additive Order

The *additive order* of an element is a naturally generalization of the property of normalcy, and is apparently first defined by Ore (1934); see also (van der Waerden, 1966) and (Lenstra & Schoof, 1987). Consider the *Frobenius automorphism* $\sigma : \mathsf{F} \to \mathsf{F}$ of $\mathsf{F}$ over $\mathsf{K}$ with $\sigma(\alpha) = \alpha^q$. For $f = \sum f_k x^k \in \mathsf{K}[x]$, we consider the $\mathsf{K}$-linear map $f(\sigma) = \sum f_k \sigma^k \colon \mathsf{F} \to \mathsf{F}$. The additive order $\mathrm{Ord}(\alpha) \in \mathsf{K}[x]$ of $\alpha$ is defined as the monic generator of the principal ideal $\{f \in \mathsf{K}[x] : f(\sigma)(\alpha) = 0\}$ in $\mathsf{K}[x]$. Since

$$(x^n - 1)(\sigma)(\alpha) = (\sigma^n - id)(\alpha) = \alpha^{q^n} - \alpha = 0$$

for all $\alpha \in \mathsf{K}$, $\mathrm{Ord}(\alpha)$ is a divisor of $x^n - 1$. An element $\alpha \in \mathsf{F}$ is normal over $\mathsf{K}$ if and only if $\mathrm{Ord}(\alpha) = x^n - 1$.

**Theorem 6.7.** *Given an irreducible polynomial $h \in \mathsf{K}[x]$ of degree $n$, an element $\alpha \in \mathsf{F} = \mathsf{K}[x]/(h)$ normal over $\mathsf{K}$, and $g \in \mathsf{K}[x]$ dividing $x^n - 1$, we can find an element $\beta \in \mathsf{F}$ of additive order $g$ with $O(\mathrm{M}(n)^2 \log n + \mathrm{M}(n) \log q)$ or $O\tilde{}(n^2 + n \log q)$ operations in $\mathsf{K}$.*

**Proof.** Simply compute $\beta = ((x^n - 1)/g)(\sigma)(\alpha) \in \mathsf{F}$. The element $\beta$ is certainly annihilated by $g(\sigma)$ since $g(\sigma)(\beta) = (x^n - 1)(\sigma)(\alpha) = 0$. To see $\mathrm{Ord}(\beta) = g$, suppose $h(\sigma)(\beta) = 0$ for some $h \in \mathsf{K}[x]$. This implies $(h(x^n - 1)/g)(\sigma)(\alpha) = 0$. Since all annihilators of $\alpha$ are divisible by $x^n - 1$, $g$ must divide $h$.

If we compute $\alpha^{q^i}$ for $0 \leq i \leq n - \deg g$ by the algorithm of von zur Gathen & Shoup (1992), then $\beta$ can be calculated with $O(n^2)$ additional operations in $\mathbb{F}_q$. $\square$

## 6.2 Bounds on Finite Field Functions

In this section we present a number of concrete and asymptotically tight bounds on some arithmetic functions of polynomials over finite fields. These will be used in the next section, and are of interest in their own right. Throughout we assume $\mathsf{K} = \mathbb{F}_q$ where $q \in \mathbb{N}$ is a prime power.

The Chebyshev $\vartheta$ function is defined for $z \geq 2$ by $\vartheta(z) = \sum_{p \leq z} \log_e p$, where the sum is extended over all primes $p \leq z$. From (Rosser & Schoenfeld 1962, Theorem 4), we get the following bounds on $\vartheta$.

**Fact 6.8.** *For $z \geq 563$,*

$$z\left(1 - \frac{1}{2\log_e z}\right) < \vartheta(z) < z\left(1 + \frac{1}{2\log_e z}\right).$$

We can define an analogue of the Chebyshev $\vartheta$ function for finite fields. For $z \geq 1$, let

$$\Theta_q(z) = \sum_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \leq z}} \deg g = \sum_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \leq z}} \log_q |g|.$$

$\Theta_q(z)$ remains constant when $q^b \leq z < q^{b+1}$, with $b \in \mathbb{N}$, and therefore we obtain sharp estimates for $\Theta_q(z)$ only when $z = q^b$ for some $b \in \mathbb{N}$.

**Theorem 6.9.** *For any prime power $q$ and $z = q^b$ for some integer $b \geq 1$, we have*

$$\frac{qz}{q-1} \cdot \left(1 - \frac{7}{2\sqrt{z}}\right) < \Theta_q(z) < \frac{qz}{q-1}.$$

*For any real $z \geq q$, we have*

$$\frac{z}{q-1} \cdot \left(1 - \frac{7\sqrt{q}}{2\sqrt{z}}\right) < \Theta_q(z) < \frac{qz}{q-1}.$$

**Proof.** Let $z = q^b$ for some integer $b \geq 1$. By definition,

$$\Theta_q(z) = \sum_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \leq z}} \deg g = \sum_{1 \leq d \leq b} d\mathrm{N}_q(d).$$

From the bounds on $\mathrm{N}_q$ given in equation (3.1) we get

$$\Theta_q(z) \leq \sum_{1 \leq d \leq b} q^d = \frac{q^{1+b}}{q-1} \cdot \left(1 - \frac{1}{q^{1+b}}\right) = \frac{qz}{q-1} \cdot \left(1 - \frac{1}{qz}\right) < \frac{qz}{q-1},$$

$$\begin{aligned}
\Theta_q(z) &\geq \sum_{1 \leq d \leq b} q^d - \frac{q}{q-1} \cdot \sum_{1 \leq d \leq b} q^{d/2} \\
&= \frac{qz}{q-1} \cdot \left(1 - \frac{1}{qz}\right) - \frac{q}{q-1} \cdot \frac{\sqrt{qz}}{\sqrt{q}-1} \cdot \left(1 - \frac{1}{\sqrt{qz}}\right) \\
&= \frac{qz}{q-1} - \frac{1}{q-1} - \frac{q}{q-1} \cdot \frac{\sqrt{qz}}{\sqrt{q}-1} + \frac{q}{(q-1)(\sqrt{q}-1)} \\
&> \frac{qz}{q-1} - \frac{q\sqrt{z}}{q-1} \cdot \frac{\sqrt{q}}{\sqrt{q}-1} > \frac{qz}{q-1} \cdot \left(1 - \frac{7}{2\sqrt{z}}\right),
\end{aligned}$$

since $q \geq 2$. For an arbitrary real $z \geq q$, let $z_0 = q^{\lfloor \log_q z \rfloor} \leq z$. Then $\Theta_q(z) = \Theta_q(z_0)$, and since the upper bound on $\Theta_q$ for powers of $q$ is an increasing function on $\mathbb{R}$, it holds for any $z \geq q$. The lower bound is an increasing function of $z$ for $z > 49/16$. Using the fact that $z/q < z_0$, when $z/q > 49/16$ the claimed lower bound follows. For $q < z \leq 49q/16$ the claim is easily verified. □

We also make use of the prime number function $\pi(z) = \#\{p \leq z \mid p \text{ prime}\}$. Rosser & Schoenfeld (1962), Theorem 1, give the following bounds on $\pi$.

**Fact 6.10.** *For any* $z \geq 17$,

$$\frac{z}{\log_e z} < \pi(z) < \frac{z}{\log_e z} \cdot \left(1 + \frac{3}{2\log_e z}\right).$$

The analogue of the number-theoretic function $\pi$ for a finite field $\mathbb{F}_q$ is $\Pi_q(z) = \#\{g \in \mathcal{I}_{\mathbb{F}_q} : |g| \leq z\}$, for any $z \geq 1$. We will make use of the following function in the analysis of $\Pi_q$:

$$\gamma_q(z) = \sum_{1 \leq d \leq \log_q z} \frac{q^d}{d}$$

for $q \geq 2$ and $z \geq q$. It can be bounded from above as follows:

**Lemma 6.11.** *For any* $z \geq q$ *and* $q \geq 2$,

$$\gamma_q(z) \leq \frac{q}{q-1} \cdot \frac{z}{\log_q z} \cdot \left(1 + \frac{2}{\log_q z}\right)$$

**Proof.** First, suppose $z = q^b$ for some integer $b \geq 1$. For $1 \leq b \leq 6$ the claim can be verified directly by considering each side of the inequality as a function of $q$ alone. We omit the details. We prove the claim by induction on $b \geq 7$:

$$\begin{aligned}\sum_{1 \leq d \leq b} \frac{q^d}{d} &= \frac{q^b}{b} + \sum_{1 \leq d \leq b-1} \frac{q^d}{d} \leq \frac{q^b}{b} + \frac{q}{q-1} \cdot \frac{q^{b-1}}{b-1} + \frac{2q}{q-1} \cdot \frac{q^{b-1}}{(b-1)^2} \\ &= \frac{q}{q-1}\frac{q^b}{b} + \frac{q}{q-1} \cdot \frac{q^b}{b^2} R(q,b),\end{aligned}$$

where

$$R(q,b) = \frac{b}{(b-1)q} + \frac{2b^2}{(b-1)^2 q}.$$

Since $R(q,b)$ is a decreasing function of $q$ and $b$, $R(q,b) \leq R(2,7) = 35/18 < 2$, so the claim is true for $q \geq 2$ and integer $b \geq 1$.

For any real $z \geq q$, let $z_0 = q^{\lfloor \log_q z \rfloor} \leq z$. Note first that

$$\frac{q}{q-1} \cdot \frac{z}{\log_q z} \cdot \left(1 + \frac{2}{\log_q z}\right)$$

is an increasing function of $z$ for $z \geq q^3$, so the claim, already proven for $z_0$, holds for real $z \geq q^3$. It is easily verified that the claim holds for real $z$ with $q < z < q^3$. $\square$

We will now show an analogue of the prime number theorem for $\mathsf{K}[x]$. $\Pi_q(z)$ remains constant when $q^b \le z < q^{b+1}$, with $b \in \mathbb{N}$, and therefore we obtain sharp estimates for $\Pi_q(z)$ only when $z = q^b$ for some $b \in \mathbb{N}$.

**Theorem 6.12.** *For any prime power $q$, and $z = q^b$ for some integer $b \ge 1$,*

$$\frac{q}{q-1} \cdot \frac{z}{\log_q z} \cdot \left(1 - \frac{7}{2\sqrt{z}}\right) < \Pi_q(z) < \frac{q}{q-1} \cdot \frac{z}{\log_q z} \cdot \left(1 + \frac{2}{\log_q z}\right).$$

*For any real $z \ge q$, we have*

$$\frac{1}{q-1} \cdot \frac{z}{\log_q z} \cdot \left(1 - \frac{7\sqrt{q}}{2\sqrt{z}}\right) < \Pi_q(z) < \frac{q}{q-1} \cdot \frac{z}{\log_q z} \cdot \left(1 + \frac{2}{\log_q z}\right).$$

**Proof.** The lower bound follows from

$$\Pi_q(z) \log_q z = \sum_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \le z}} \log_q z \ge \sum_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \le z}} \deg g = \Theta_q(z),$$

and the lower bounds in Theorem 6.9. For the upper bound, note that for arbitrary real $z \ge q \ge 2$,

$$\Pi_q(z) = \sum_{1 \le d \le \log_q z} \mathrm{N}_q(d) \le \sum_{1 \le d \le \log_q z} \frac{q^d}{d} \le \gamma_q(z)$$

by (3.1) and Lemma 6.11 proves the claim. $\square$

The function $\gamma_q(z)$ is a better approximation to $\Pi_q(z)$, with error only $O(\sqrt{z})$.

For $n \in \mathbb{N}$, let $\delta(n) = \#\{d \in \mathbb{N},\, d|n,\, d \text{ is squarefree}\} = 2^{\omega(n)}$, where $\omega(n)$ is the number of primes dividing $n$. Hardy & Wright (1962), Section 22.11 show the "normal order" of $\omega(n)$ is $\log\log n$. We can bound $\delta$ from above as follows.

**Theorem 6.13.** *For $n \ge 9$,*

$$\log_2 \delta(n) \le \frac{\log_e n}{\log_e \log_e n} \cdot \left(1 + \frac{1.25 \log_e \log_e \log_e n}{\log_e \log_e n}\right).$$

**Proof.** First, note that $\log_2 \delta(n)$ is the number of distinct prime divisors of $n$. Suppose $n = q_1^{e_1} q_2^{e_2} \cdots q_k^{e_k}$ where $k, e_1, e_2, \ldots, e_k$ are positive integers and $q_1 < q_2 < \cdots < q_k$ are primes. If $n_0 = q_1 q_2 \cdots q_k$, then $\delta(n_0) = \delta(n)$, while $n_0 \leq n$. Since we are trying to prove that $\delta$ is bounded above by an increasing function of $n$ for $n > e$, if the claim fails for $n$, it also fails for $n_0$ as well. We can therefore assume without loss of generality that $e_1 = e_2 = \cdots = e_k = 1$. Now suppose $p$ is a prime less than $q_k$ and $p \notin \{q_1, \ldots, q_{k-1}\}$. Let $n_1 = p q_1 q_2 \cdots q_{k-1}$. Once again $\delta(n_1) = \delta(n)$, and if the claim fails for $n$, it also fails for $n_1 < n$. Thus, we can assume without loss of generality that $n$ is the product of all primes less than some $z \geq 2$.

Let $z \geq 3989$ and let $n$ be the product of all primes less than or equal to $z$ (the number 3989 is chosen only for convenience in this proof). Then

$$\log_2 \delta(n) = \pi(z) < \frac{z}{\log_e z} \cdot \left(1 + \frac{3}{2\log_e z}\right)$$

by Fact 6.10. From Fact 6.8, $\log_e n = \vartheta(z) < z(1 + 1/(2\log_e z)) < 1.1z$ or $z > 0.9 \log_e n$, for $z \geq 3989$ or $n \geq \exp(\vartheta(3989))$. Also by Fact 6.8,

$$\log_e n = \vartheta(z) > z\left(1 - \frac{1}{2\log_e z}\right) > z\left(1 - \frac{1}{2\log_e \log_e n + 2\log_e 0.9}\right),$$

so that

$$z < \left(1 - \frac{1}{2\log_e \log_e n + 2\log_e 0.9}\right)^{-1} \log_e n < (1+u)\log_e n,$$

where $u = \log_e \log_e \log_e n/(2\log_e \log_e n) > 0$, since $z \geq 3989$ or $n \geq \exp(\vartheta(3989))$. Substituting this upper bound on $z$ into our upper bound for $\log_2 \delta(n)$ in terms of $z$, we get

$$\begin{aligned}
\log_2 \delta(n) &< \frac{\log_e n}{\log_e((1+u)\log_e n)} \cdot (1+u)\left(1 + \frac{3}{2\log_e((1+u)\log_e n)}\right) \\
&< \frac{\log_e n}{\log_e \log_e n} \cdot (1+u)\left(1 + \frac{3}{2\log_e \log_e n}\right) \\
&< \frac{\log_e n}{\log_e \log_e n} \cdot \left(1 + \frac{1.25 \log_e \log_e \log_e n}{\log_e \log_e n}\right)
\end{aligned}$$

for $n > \exp(\vartheta(3989))$.

For $i \geq 1$, let $P_i$ be the product of the $i$ smallest primes. The number 3989 is the 550th smallest prime, so the claim holds for $n \geq P_{550}$. Using the computer algebra program MAPLE 4.3, we verified that the claim holds for $P_i$, where $3 \leq i < 550$, and for $9 \leq n < P_3 = 30$. This shows the claim is true for all integers $n \geq 9$. $\square$

The inequality of this theorem is false for $n = 8$.

For polynomials over finite fields, the analogue of $\delta$ is as follows. For any prime power $q$, and any $f \in \mathsf{K}[x]$, let $\Delta_{\mathsf{K}}(f)$ be the number of distinct, monic, squarefree divisors of $f$ (including the divisor 1). We bound $\Delta_{\mathsf{K}}$ from above in the next theorem, and show that this bound cannot be improved much.

**Theorem 6.14.** *Let $f \in \mathsf{K}[x]$ of degree $n$. For $n > 1$, $\log_2 \Delta_{\mathsf{K}}(f) \leq n$. For $n > q$,*

$$\log_2 \Delta_{\mathsf{K}}(f) < \frac{n}{\log_q n} \cdot \left(1 + \frac{3.5 \log_e \log_e n}{\log_e n}\right).$$

*Furthermore, for any fixed prime power $q$, there exist an infinite number of $f \in \mathsf{K}[x]$ such that*

$$\log_2 \Delta_{\mathsf{K}}(f) \geq \frac{n}{\log_q n} \cdot \left(1 - \frac{5}{\log_q n}\right)$$

*where $n = \deg f$.*

**Proof.** We begin with the upper bound. First note that $\log_2 \Delta_{\mathsf{K}}(f)$ is the number of distinct divisors of $f$ in $\mathcal{I}_{\mathsf{K}}$. Suppose $f = g_1^{e_1} g_2^{e_2} \cdots g_k^{e_k}$ where $k, e_1, e_2, \ldots, e_k$ are positive integers and $g_1, g_2, \ldots, g_k \in \mathcal{I}_{\mathsf{K}}$ are pairwise distinct. If $f_0 = g_1 g_2 \cdots g_k$, then $\Delta_{\mathsf{K}}(f) = \Delta_{\mathsf{K}}(f_0)$, while $n_0 = \deg f_0 \leq n$. Since we are trying to prove $\log_2 \Delta_{\mathsf{K}}(f)$ is bounded above above by an increasing function of $n$ for $n \geq e$, if the claim fails for $f$, it also fails for $f_0$. We can therefore assume without loss of generality that $e_1 = \cdots = e_k = 1$. Now suppose $\deg g_1 \leq \deg g_2 \leq \cdots \leq \deg g_k$ and $h \notin \{g_1, \ldots, g_k\}$ for some $h \in \mathcal{I}_{\mathsf{K}}$ with $\deg h < \deg g_k$, and let $f_1 = h g_1 \cdots g_{k-1}$. Once again $\Delta_{\mathsf{K}}(f_1) = \Delta_{\mathsf{K}}(f)$, and if the claim fails for $f$, it also fails for $f_1$ (because $\deg f_1 < \deg f$). Therefore, we can assume without loss of generality that, for all $h | f$ with $h \in \mathcal{I}_{\mathsf{K}}$, all $h_0 \in \mathcal{I}_{\mathsf{K}}$ with $\deg h_0 < \deg h$ also divide $f$. In other words, $f$ is the product of the maximum number of distinct irreducible polynomials for its degree.

If $1 \leq n \leq q$, $f$ is the product of linear polynomials in $\mathsf{K}[x]$. Thus, $\log_2 \Delta_{\mathsf{K}}(f) = n$. If $n > q$, we write $\log_2 \Delta_{\mathsf{K}}(f) = P_1 + P_2$, where $f = g_1 \cdots g_k$ and $g_1, \ldots, g_k$ are irreducible,

$$P_1 = \sum_{\substack{|g_i| \le \beta \\ 1 \le i \le k}} 1, \qquad P_2 = \sum_{\substack{|g_i| > \beta \\ 1 \le i \le k}} 1,$$

and $\beta = n/\log_e n$. If $\beta < q$, then $P_1 = 0$. For $\beta > q$,

$$P_1 = \Pi_q(\beta) \le \frac{q\beta}{(q-1)\log_q \beta} \cdot \left(1 + \frac{2}{\log_q \beta}\right),$$

by Theorem 6.12. To bound $P_2$, we note that

$$q^n = |f| = \prod_{1 \le i \le k} |g_i| \ge \prod_{\substack{|g_i| > \beta \\ 1 \le i \le k}} \beta = \beta^{P_2},$$

or $P_2 \le n/\log_q \beta$. We set $\lambda_1 = \log_e n$ and $\lambda_2 = \log_e \log_e n$. If $\beta < q$, this gives

$$\begin{aligned}
\log_2 \Delta_{\mathsf{K}}(f) = P_2 < \frac{n}{\log_q \beta} &= \frac{n}{\log_q n} \cdot \left(\frac{\lambda_1}{\lambda_1 - \lambda_2}\right) \\
&= \frac{n}{\log_q n} \cdot \left(1 + \frac{\lambda_2}{\lambda_1 - \lambda_2}\right) \le \frac{n}{\log_q n} \cdot \left(1 + \frac{1.4\lambda_2}{\lambda_1}\right)
\end{aligned}$$

for $n \ge 1619 > e^{e^2}$, using the fact that $1/(\lambda_1 - \lambda_2) \le 1.38/\lambda_1$ for $n \ge 1619$. If $\beta \ge q$,

$$\begin{aligned}
\log_2 \Delta_{\mathsf{K}}(f) &< \frac{n}{\log_q \beta} + \frac{q\beta}{(q-1)\log_q \beta} \cdot \left(1 + \frac{2}{\log_q \beta}\right) \\
&= \frac{n}{\log_q n} \cdot \left(\frac{\lambda_1}{\log_e \beta} + \frac{q\lambda_1}{(q-1)n\log_e \beta} \cdot \left(1 + \frac{2\log_e q}{\log_e \beta}\right)\right) \\
&= \frac{n}{\log_q n} \cdot \left(\frac{\lambda_1 + h(q,n)}{\lambda_1 - \lambda_2}\right) = \frac{n}{\log_q n} \cdot \left(1 + \frac{h(q,n) + \lambda_2}{\lambda_1 - \lambda_2}\right),
\end{aligned}$$

where

$$h(q,n) = \frac{q}{q-1} \cdot \left(1 + \frac{2\log_e q}{\lambda_1 - \lambda_2}\right).$$

For $n \ge 1619$, $h(q,n)$ is a decreasing function of $n$. Also, for $n \ge 1619$, $h(q,n)$ is an increasing function of $q$ for $q \ge 7$, and achieves its maximum with $q \ge 7$. To maximize $h$, we choose $q$ as large as possible. Since $\beta = n/\lambda_1 > q$, we have

$h(q,n) < h(n/\lambda_1, n) = 3n/(n-\lambda_1) \le 3.02$ for $n \ge 1619$. It follows that

$$\begin{aligned}\log_2 \Delta_{\mathsf{K}}(f) &< \frac{n}{\log_q n} \cdot \left(1 + \frac{3.02 + \lambda_2}{\lambda_1 - \lambda_2}\right) < \frac{n}{\log_q n}\left(1 + \frac{2.51\lambda_2}{\lambda_1 - \lambda_2}\right) \\ &< \frac{n}{\log_q n}\left(1 + \frac{3.5\lambda_2}{\lambda_1}\right)\end{aligned}$$

for $n \ge 1619$. For $q < n < 1619$, we verified the claim using the computer algebra system Maple 4.3.

To show that this upper bound cannot be improved much, let $q$ be a fixed prime power, $z = q^m$ for some integer $m \ge 2$, and

$$f = \prod_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \le z}} g.$$

Then

$$\log_2 \Delta_{\mathsf{K}}(f) = \Pi_q(z) > \frac{qz}{(q-1)\log_q z} \cdot \left(1 - \frac{7}{2\sqrt{z}}\right),$$

by Theorem 6.12. This lower bound on $\Pi_q(z)$ is an increasing function of $z$ for $z \ge 4$. The degree $n$ of $f$ satisfies

$$n = \sum_{\substack{g \in \mathcal{I}_{\mathsf{K}} \\ |g| \le z}} \deg g = \Theta_q(z) < \frac{qz}{q-1}$$

by Theorem 6.9, which can be rewritten $z > n + n/q$. This shows

$$\begin{aligned}\log_2 \Delta_{\mathsf{K}}(f) &\ge \frac{n}{\log_q(n + n/q)} \cdot \left(1 - \frac{7}{2\sqrt{n + n/q}}\right) \\ &= \frac{n}{\log_q n}\left(1 - \frac{\log_q(1 + q^{-1})}{\log_q(n + n/q)}\right)\left(1 - \frac{7}{2\sqrt{n + n/q}}\right) \\ &\ge \frac{n}{\log_q n}\left(1 - \frac{1}{\log_q n} - \frac{7}{2\sqrt{n}}\right) \ge \frac{n}{\log_q n}\left(1 - \frac{5}{\log_q n}\right)\end{aligned}$$

using the fact that $7/(2\sqrt{n}) < 4/\log_q n$ for all $n \ge q \ge 2$. $\square$

## 6.3 Finding Primitive Normal Elements

Let $q$ be a prime power and $\mathsf{K} = \mathbb{F}_q$. For a positive integer $n$, let $\mathsf{E} = \mathbb{F}_{q^n}$, an algebraic extension field of degree $n$ over $q$. The multiplicative group of $\mathsf{E}$ is cyclic of order $q^n - 1$ (see Lidl & Niederreiter 1983, Theorem 2.8) and for any nonzero $\alpha \in \mathsf{E}$, we define the multiplicative order as $\mathrm{ord}(\alpha) = \min\{d \in \mathbb{N},\, d \geq 1,\, \alpha^d = 1\}$. An element of order $q^n - 1$ is called *primitive* and we denote by $\mathcal{P}$ the set of all these. It is well known that there are $\phi(q^n - 1)$ primitive elements in $\mathsf{E}$. There is no known general way to either generate or certify a primitive element in (probabilistic) polynomial time.

Let $\mathcal{N}$ be the set of normal elements in $\mathsf{E}$ over $\mathsf{K}$. Unless otherwise stated, we assume that a *normal* element in $\mathsf{E}$ means a normal element over $\mathsf{K}$. What is the probability $\varrho$ that a randomly chosen element $\alpha \in \mathsf{E}$ is simultaneously primitive and normal over $\mathsf{K}$? This question was first addressed by Carlitz (1952) who showed in his statement (4.7) that

$$\varrho = \frac{|\mathcal{P} \cap \mathcal{N}|}{q^n} \geq \frac{\phi(q^n - 1) \cdot \Phi_{\mathsf{K}}(x^n - 1)}{q^{2n}} - \frac{\delta(q^n - 1)\Delta_{\mathsf{K}}(x^n - 1)}{q^{n/2}}. \tag{6.1}$$

We will refer to the right hand side of this inequality as the Carlitz bound on the number of primitive normal elements. For sufficiently large fields $\mathsf{E}$, this tends towards $|\mathcal{P}| \cdot |\mathcal{N}|/q^{2n}$. It was later shown by Davenport (1968) that for $q$ prime and $n \geq 2$ there exists a primitive normal element in $\mathsf{E}$ over $\mathsf{K}$. Lenstra & Schoof (1987) showed that for all prime powers $q$ and $n \geq 2$ there exists a primitive normal element in $\mathsf{E}$ over $\mathsf{K}$. We give a positive lower bound for $\varrho$ for all but a finite number of pairs $(q, n)$.

**Theorem 6.15.** *Let $q$ be a prime power, $n \geq 2$, and $\varrho$ the probability that an element is primitive and normal in $\mathsf{E}$ over $\mathsf{K}$. Then*

*(i) if $n \geq 300$ and $n > q$, then $\varrho > 0.1/(\log_q n \cdot \log_e(n \log_e q))$,*

*(ii) if $n \geq 300$ and $n \leq q$, then $\varrho > 0.19/\log_e(n \log_e q)$,*

*(iii) if $300 \geq n \geq 2$ and $q \geq 10^7$, then $\varrho > 0.3/\log_e(n \log_e q)$.*

**Proof.** Since our bounds on $\Phi_{\mathsf{K}}$ and $\Delta_{\mathsf{K}}$ from Theorems 3.15 and 6.14 depend upon the relationship between $q$ and $n$, our lower bound on $\varrho$ must be divided into a number of cases. First we examine the bounds for $\delta(q^n-1)$ and $\phi(q^n-1)$, which are valid when $q^n-1 \geq 16$. We abbreviate $\lambda_1 = \log_e(q^n-1)$, $\lambda_2 = \log_e \lambda_1$ and $\lambda_3 = \log_e \lambda_2$. By Theorem 6.13,

$$\log_2 \delta(q^n-1) < \frac{\lambda_1}{\lambda_2} \cdot \left(1 + \frac{1.25\lambda_3}{\lambda_2}\right) < \frac{n \log_e q}{\lambda_2} \cdot \left(1 + \frac{1.25\lambda_3}{\lambda_2}\right),$$

since our upper bound on $\delta$ is an increasing function. This gives $\log_q \delta(q^n-1) < r \cdot n/\lambda_2$ where $r = (1+1.25\lambda_3/\lambda_2) \cdot \log_e 2$. We require the following lower bound on $\phi$, derived from (Rosser & Schoenfeld, 1962):

**Fact 6.16.** *For* $n \geq 3$*,*

$$\frac{\phi(n)}{n} > \frac{c}{\log_e \log_e n} \cdot \left(1 - \frac{1.41}{\log_e^2 \log_e n}\right),$$

*where* $c = e^{-C} \approx 0.56146$.

**Proof.** From Rosser & Schoenfeld (1962), (3.41) and (3.42), we have

$$\frac{\phi(n)}{n} > \frac{1}{\lambda/c + c_1/\lambda} = \frac{c}{\lambda} \cdot \left(1 - \frac{c_1 c}{\lambda^2 + c_1 c}\right) > \frac{c}{\lambda}\left(1 - \frac{c_1 c}{\lambda^2}\right),$$

where $\lambda = \log_e \log_e n$ and $c_1 = 2.5036$. $\square$

Using this fact, we have $\phi(q^n-1) > s \cdot q^n/\lambda_2$, where $s = c \cdot (1-1.41/\lambda_2^2) \cdot (1-q^{-n})$. Applying (6.1) we have

$$\varrho > (s/\lambda_2) \cdot (\Phi_{\mathsf{K}}(x^n-1)/q^n) - q^{rn/\lambda_2 + \log_q \Delta_{\mathsf{K}}(x^n-1) - n/2}. \tag{6.2}$$

By Theorem 6.14, for $n \geq q$ and $n \geq 300$ we have

$$\log_q \Delta_{\mathsf{K}}(x^n-1) < \frac{n \log_e 2}{\log_e n} \cdot \left(1 + \frac{3.5 \log_e \log_e n}{\log_e n}\right) < \frac{1.44n}{\log_e n}. \tag{6.3}$$

We first look at case (i), where $n \geq 300$ and $n > q$. Here, $\Phi_{\mathsf{K}}(x^n-1) \geq q^n/(5\log_q n)$ by Theorem 3.15. In this case we have $r < 0.97$ and $s > 0.53$. Using (6.2), and (6.3) we find

$$\varrho \geq \frac{0.53}{5\log_q n \cdot \lambda_2} - q^{n(0.97/\lambda_2+1.44/\log_e n-1/2)} > \frac{0.106}{\log_q n \cdot \lambda_2} - q^{-0.066n} > \frac{0.1}{\log_q n \cdot \lambda_2}.$$

For case (ii), when $300 \leq n \leq q$, $\Phi_{\mathsf{K}}(x^n - 1) \geq q^n/e$, and by Theorem 6.14, $\log_q \Delta_{\mathsf{K}}(x^n - 1) < n \log_q 2$. Also, $r \leq 0.93$ and $s \geq 0.54$, giving

$$\varrho \geq \frac{0.54}{e\lambda_2} - q^{n(0.93/\lambda_2+\log_q 2-1/2)} > \frac{0.198}{\lambda_2} - q^{-0.25n} > \frac{0.19}{\lambda_2}.$$

Finally, for case (iii), we use the fact that for all $q \geq 2$ and $n \geq 2$, $\log_q \Delta_q(x^n-1) \leq n \log_q 2$ by Theorem 6.14. Since $n < q$, $\Phi_q(x^n - 1) \geq q^n/e$ by Theorem 3.15. For $300 > n \geq 2$ and $q \geq 10^7$, $r \leq 1.1$ and $s \geq 0.496$, so that

$$\varrho \geq \frac{0.496}{e\lambda_2} - q^{n(1.003/\lambda_2+\log_e 2/\log_e q-1/2)} > \frac{0.368}{\lambda_2} - q^{-0.17n} > \frac{0.3}{\lambda_2}.$$

□

This theorem covers all but finitely many values of $(n, q)$. The exceptions, with $n < 300$ and $q < 10^7$, are about $2 \times 10^8$ in number. The following theorem cuts down this number significantly.

**Theorem 6.17.** *If* $9 \leq n < 300$ *and* $q \geq 300$, *then* $\varrho > 0.1/\log_e(n \log_e q)$.

**Proof.** We use the notation of Theorem 6.15. For $q \geq 300$ and $300 > n \geq 9$, we have $r \leq 1$ and $s \geq 0.5$. Also $\log_q \Delta_q(x^n - 1) \leq n \log_q 2$ by Theorem 6.14 and $\Phi_q(x^n - 1) > q^n/e$ by Theorem 3.15. Applying (6.2), this gives

$$\varrho \geq \frac{0.5}{e\lambda_2} - q^{n(1/\lambda_2+\log_q 2-1/2)} > \frac{0.183}{\lambda_2} - q^{-0.25n} > \frac{0.1}{\lambda_2}.$$

□

This leaves us with about $8.3 \times 10^6$ exceptional pairs of fields. For each of these pairs we have used Maple 4.3 to compute the Carlitz bound (6.1), or at least a good lower bound for it. We found that it is at least $1/200$ for all but 121 of the exceptional pairs of fields, and for the 121 remaining pairs it is negative. Thus, for all but 121 of the exceptional field pairs, the probability of finding a primitive normal element is at least $1/200$. We have $q \leq 2729$ and $n \leq 21$ for all 121 remaining cases. Note also that the existence proofs of Davenport (1968) and Lenstra & Schoof (1987) showing $\varrho > 0$ both consider a (smaller) number of special cases.

**Corollary 6.18.** *Let $q$ be a prime power and $n \geq 2$. There exists a probabilistic polynomial-time reduction from the problem of finding a primitive normal element in $\mathsf{E}$ over $\mathsf{K}$ to finding a primitive element in $\mathsf{E}$.*

**Proof.** Construct a lookup table of the (finitely many) exceptional cases not covered by Theorem 6.15, mapping a pair $(q, n)$ to a primitive normal element in $\mathsf{E}$ over $\mathsf{K}$. For a given input $(q, n)$, where $q$ is a prime power and $n \geq 2$, check if $(q, n)$ is in the table of exceptions. If it is, return the primitive element stored there. If not, find a primitive element $\beta \in \mathsf{E}$. Randomly select an integer $j$ between 1 and $q^n - 2$, compute $\alpha = \beta^j$, and test $\beta$ for primitivity and normalcy over $\mathsf{K}$. In the case of Theorem 6.15 (i), we require an expected number of $O(\log_e(n \log_e q) \cdot \log_q n)$ random choices. For the remaining two cases of Theorem 6.15 and the case of Theorem 6.17, we require an expected number of $O(\log_e(n \log_e q))$ random choices. To test primitivity, we need only check that $\gcd(q^n - 1, j) = 1$, which requires $O\tilde{\ }(n \log q)$ bit operations, while to test normalcy requires $O(\mathrm{M}(n)^2 + \mathrm{M}(n) \log_e q)$ operations in $\mathsf{K}$ by Theorem 6.2. $\square$

# Chapter 7.

# Zero Divisors in Associative Algebras

Finding zero divisors in a finite dimensional associative algebra over a finite field $\mathsf{K}$ is an important problem in computational symbolic algebra. The most frequently encountered example of such an algebra occurs in the factoring of a polynomial $f \in \mathsf{K}[x]$ into irreducible factors. This problem is equivalent to finding zero divisors in the associative $\mathsf{K}$-algebra $\mathsf{K}[x]/(f)$ and indeed, this is the view taken in many algorithms for factoring in $\mathsf{K}[x]$ (see Berlekamp 1970, Cantor & Zassenhaus 1981, and von zur Gathen & Shoup 1992). More generally, let $\mathcal{A}$ be any finite dimensional associative algebra over a finite field $\mathsf{K}$. That is, $\mathcal{A}$ is a finite dimensional vector space over $\mathsf{K}$, with a product $\times\colon \mathcal{A} \to \mathcal{A}$, such that $\mathcal{A}$ is a ring under $+$ and $\times$ (we write $ab$ for $a \times b$ for any $a, b \in \mathcal{A}$). Associative algebras have applications in linear algebra (see Gohberg 1986), and are central objects in the study of modular representations of groups (see Parker 1984, Michler 1987, and Rónyai 1990 for computational approaches). Many of these applications involve splitting or decomposing an associative algebra, an operation for which finding zero divisors is a key step. In Chapter 8 of this thesis, the problem of factoring in skew-polynomial rings, a natural non-commutative generalization of factoring in $\mathsf{K}[x]$, is reduced to the problem of finding zero divisors in an associative algebra of small dimension over $\mathsf{K}$.

Our goal in this chapter is to determine (efficiently) if $\mathcal{A}$ has any non-trivial (i.e., non-zero) zero divisors, and if it does, to produce a pair $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ such that $b_1 b_2 = 0$. Familiarity with the basic theorems and terminology of finite dimensional associative algebras is assumed. A good reference for this information is (Pierce 1982, Chapters 1–4). Throughout this chapter, the word "algebra" (over $\mathsf{K}$) will mean a finite dimensional associative algebra (over $\mathsf{K}$) with a multiplicative identity $1 \in \mathcal{A}$. Algebras may or may not be commutative. Our algorithm is of the Las

Vegas probabilistic type, that is, it is assumed to have at its disposal a source of random bits, accessible at unit cost. On any input, with probability at least 1/4, the algorithm returns the correct answer. Otherwise it reports "failure". In no instance is an incorrect answer returned.

An algebra $\mathcal{A}$ over $\mathsf{K}$ is described computationally as a $\mathsf{K}$-vector space of dimension $\nu > 0$ with a supplied basis $\mathcal{B} = (w_1, \ldots, w_\nu) \in \mathcal{A}^\nu$; elements of $\mathcal{A}$ are represented with respect to this basis as vectors in $\mathsf{K}^\nu$. A representation of the identity element in this basis is assumed to be supplied. In this computational representation of $\mathcal{A}$, addition is component-wise and requires $O(\nu)$ operations in $\mathsf{K}$ to perform. A *black box* algorithm to multiply two elements in $\mathcal{A}$ is assumed to be provided, which requires $\chi$ operations in $\mathsf{K}$. Our main theorem, which we prove over the next two sections, is as follows:

**Theorem 7.1.** *Let $\mathcal{A}$ be an algebra with dimension $\nu$ over $\mathsf{K} = \mathbb{F}_q$ represented as above. There is a Las Vegas type probabilistic algorithm which requires $O(\nu\chi + \mathrm{MM}(\nu) + \mathrm{M}(\nu)\log q)$ operations in $\mathsf{K}$ to determine whether $\mathcal{A}$ is a field extension of $\mathsf{K}$, or to produce $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ with $b_1 b_2 = 0$. The algorithm succeeds with probability at least 1/4.*

The proof of this theorem is broken into two parts. In Section 7.1 we exhibit a simple Las Vegas type algorithm which constructs two non-zero zero divisors multiplying to zero or, if $\mathcal{A}$ is a field extension of $\mathsf{K}$, provides an element generating a power basis for $\mathcal{A}$ over $\mathsf{K}$. Informally, the algorithm works as follows. Two elements $a_1, a_2 \in \mathcal{A}$ are selected randomly from $\mathcal{A}$. If $\mathcal{A}$ is a field then at least one of $a_1, a_2$ generates a power basis for $\mathcal{A}/\mathsf{K}$ with high probability. When $\mathcal{A}$ is not a field, the property required of the randomly selected elements is that their minimal polynomials in $\mathsf{K}[x]$ be reducible with high probability. Suppose $a_1$ has minimal polynomial $f \in \mathsf{K}[x]$. If $f = f_1 f_2$ for $f_1, f_2 \in \mathsf{K}[x] \setminus \mathsf{K}$ then $f_1(a_1) f_2(a_1) = 0$, and $f_1(a_1), f_2(a_1) \neq 0$ since $f$ is $a_1$'s minimal polynomial, i.e., $f_1(a_1)$ and $f_2(a_1)$ are non-zero zero divisors. In Section 7.2 we show, for any finite algebra which is not local (i.e., an algebra $\mathcal{A}$ such that $\mathcal{A}$ modulo its radical is a field; see Theorem 7.12), that the probability of two elements both having an irreducible minimal polynomial is at most 1/2. Unfortunately, this theorem does not necessarily hold when $\mathcal{A}$ is a non-commutative local algebra. In this case we show that, for randomly selected $a_1, a_2 \in \mathcal{A}$, with probability at least $1/4$, $a_1a_2 - a_2a_1$ has a reducible minimal polynomial. We do not need to know anything, nor do we compute anything, about the structure of $\mathcal{A}$ for our algorithm. Simply choose $a_1, a_2 \in \mathcal{A}$ at random and attempt to factor the minimal polynomials of $a_1$, $a_2$, and $a_1a_2 - a_2a_1$.

The problem of finding zero divisors was known previously to be polynomial-time reducible to factoring univariate polynomials over a finite field.

**Fact 7.2** (Rónyai, 1990). *Let $\mathcal{A}$ be an algebra over $\mathsf{K}$ with dimension $\nu$, presented as above. The problem of determining if there exist zero divisors in $\mathcal{A}$, and if so finding a pair $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ with $b_1b_2 = 0$, is reducible with $(\nu\eta \log p)^{O(1)}$ operations in $\mathsf{K}$ to the problem of factoring polynomials in $\mathbb{F}_p[x]$ of degree $(\nu\eta)^{O(1)}$.*

Making use of the factoring algorithms of Berlekamp (1970), Rónyai's theorem implies that the problem of determining whether non-zero zero divisors exist in $\mathcal{A}$, and finding a pair multiplying to zero if they do, is solvable with a deterministic algorithm requiring $(\nu\eta p)^{O(1)}$ operations in $\mathbb{F}_p$, or a Las Vegas type probabilistic algorithm requiring $(\nu\eta \log p)^{O(1)}$ operations in $\mathbb{F}_p$.

Rónyai's algorithm computes an explicit decomposition of $\mathcal{A}$, finding the radical of $\mathcal{A}$, computing a Wedderburn decomposition if $\mathcal{A}$ is semi-simple, and computing an isomorphism with a full matrix ring over an extension field of $\mathsf{K}$ if $\mathcal{A}$ is simple. Needless to say, this algorithm is quite complicated, and while not necessarily inefficient, Rónyai only shows it to be polynomial time and does not calculate the running time explicitly.

A common way of representing an algebra $\mathcal{A}$ of dimension $\nu$ over $\mathsf{K}$ is as a subalgebra of $\mathsf{K}^{m\times m}$, for some $m \geq 1$. Any $\nu$ dimensional algebra can be embedded into $\mathsf{K}^{m\times m}$ for any $m \geq \nu$, and possibly for some $m < \nu$. See Section 7.1 for a more formal discussion of this. When $\mathcal{A} \subseteq \mathsf{K}^{m\times m}$ is given with respect to some $\mathsf{K}$-basis $T_1, \ldots, T_\nu \in \mathsf{K}^{m\times m}$ we can employ the fast algorithms for finding the minimal polynomial of a matrix and evaluating a polynomial at a matrix from Chapters 2 and 5 to great advantage for finding zero divisors. A Las Vegas type random algorithm is given which requires an expected number of $O\tilde{~}(\mathrm{MM}(m) + \min(m, \nu) \log q)$ operations in $\mathsf{K}$ or an expected number of $O(m^3 + \min(m, \nu)^2 \log q)$ operations in $\mathsf{K}$ when standard polynomial and matrix arithmetic is used. These cost estimates assume $\#\mathsf{K} \geq m^2$, but similar results hold for smaller fields.

## 7.1 A Fast Algorithm for Finding Zero Divisors

Let $\mathcal{A}$ be any finite dimensional associative algebra (with identity) of dimension $\nu$ over a finite field $\mathsf{K} \cong \mathbb{F}_q$, where $q$ is a power of a prime $p$. As in the introduction, $\mathcal{A}$ is described computationally as a $\mathsf{K}$-vector space with a basis $\mathfrak{B} = (w_1, \ldots, w_\nu) \in \mathcal{A}^\nu$. A representation of $1 \in \mathcal{A}$ is assumed to be supplied. Addition in $\mathcal{A}$ is component-wise and a "black box" algorithm for multiplication in $\mathcal{A}$, which requires $\chi$ operations in $\mathsf{K}$, is assumed to be provided.

Our algorithm is based on finding and factoring the minimal polynomial of a randomly selected element $a \in \mathcal{A}$, then evaluating the factors at $a$. Recall that the *minimal polynomial* $\min_{\mathsf{K}}(b)$ of $b \in \mathcal{A}$ is the monic polynomial $f \in \mathsf{K}[x]$ of minimal degree such that $f(b) = 0$. It does not depend on how $\mathcal{A}$ is represented as an extension of $\mathsf{K}$, and has degree at most $\dim \mathcal{A} = \nu$.

**Algorithm:** `ZeroDivisor`

Input: an algebra $\mathcal{A}$ of dimension $\nu$ over $\mathsf{K}$ (see above);

Output: $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ with $b_1 b_2 = 0$, or a report that $\mathcal{A}$ is a field, or failure;

(1) Choose random $a_1, a_2 \in \mathcal{A}$;

   For $b \in \{a_1,\ a_2,\ a_1a_2 - a_2a_1\} \setminus \{0\}$ Do

(2)       Compute $f = \min_{\mathsf{K}}(b) \in \mathsf{K}[x]$;

(3)       Factor $f$ over $\mathsf{K}[x]$;

      If $f$ is reducible (say $f = gh$ for $g, h \in \mathsf{K}[x] \setminus \mathsf{K}$)

(4)           Return $g(b)$, $h(b)$;

      Else if $\deg f = \nu$ (and $f$ is irreducible)

(5)           Return "$\mathcal{A}$ is a field (and has no zero divisors)";

   End For;

(6) Return "Failure";

**End.**

To see that the algorithm is correct, examine two cases: when $\mathcal{A}$ is a (commutative) field and when it is not. In the latter case we know that $\mathcal{A}$ contains zero divisors by Wedderburn's Theorem (see Lidl & Niederreiter 1983, Section 2.6) which shows any finite algebra whose only zero divisor is zero, is a (commutative) field.

**$\mathcal{A}$ a field:** if some $b \in \mathcal{A}$ has a minimal polynomial $f \in \mathsf{K}[x]$ which is irreducible of degree $\nu$ then $\mathcal{A} = \mathsf{K}[b]$ and $\mathsf{K}[b]$ is isomorphic to the finite field $\mathsf{K}[x]/(f) \cong \mathbb{F}_{q^\nu}$ under the isomorphism carrying $b$ to $x \bmod f$. In the next section we show that when $b$ is chosen as in step (1) it will have an irreducible minimal polynomial of degree $\nu$ with high probability.

**$\mathcal{A}$ not a field:** let $b \in \mathcal{A}$ have a reducible minimal polynomial $f \in \mathsf{K}[x]$. If $f = gh$ for some $g, h \in \mathsf{K}[x] \setminus \mathsf{K}$, then $f(b) = 0 = g(b)h(b)$, and $g(b)$, $h(b)$ are non-zero since $f$ is the minimal polynomial of $b$. In the next section we show that when $b$ is chosen as in step (1) it will have a reducible minimal polynomial with high probability.

While determining the cost of this algorithm, assume failure probability $\varrho < 1$. In Theorems 7.5, 7.6 and 7.12 in the next section, we show that, in fact, $\varrho \leq 3/4$. Computing $f$ in step (2) can be accomplished by first computing the sequence $1, b, b^2, \ldots, b^\nu \in \mathcal{A}$, requiring $O(\nu\chi)$ operations in $\mathsf{K}$. Using linear algebra $f$ can then be found with $O(\mathrm{MM}(\nu))$ additional operations in $\mathsf{K}$. Factoring $f$ can be done using the Las Vegas type probabilistic algorithm of Berlekamp (1970), with $O(\mathrm{MM}(\nu) + \mathrm{M}(\nu)\log q)$ operations in $\mathsf{K}$. Evaluating $g(b)$ and $h(b)$ in step 4 can be done with $O(\nu^2)$ operations in $\mathsf{K}$, using the powers of $b$ computed in step (2). We have shown the following.

**Theorem 7.3.** *Let $\mathcal{A}$ be an algebra with dimension $\nu$ over $\mathsf{K} = \mathbb{F}_q$. The algorithm* `ZeroDivisor` *requires $O(\nu\chi + \mathrm{MM}(\nu) + \mathrm{M}(\nu)\log q)$ operations in $\mathsf{K}$ to determine whether $\mathcal{A}$ is a field extension of $\mathsf{K}$, or to produce $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ with $b_1 b_2 = 0$. The algorithm is probabilistic of the Las Vegas type.*

A standard way of representing an associative algebra $\mathcal{A}$ of dimension $\nu$ over $\mathsf{K}$ is to embed $\mathcal{A}$ into the algebra $\mathsf{K}^{\nu\times\nu}$. Such an embedding always exists, and is achieved, for example, by the *regular representation* of $\mathcal{A}$. Assume that $\mathcal{A}$ is represented, as above, with respect to some basis $\mathfrak{B} = (w_1, \ldots, w_\nu) \in \mathcal{A}^\nu$. Left multiplication by $w_i$ (for some $1 \le i \le \nu$) is a $\mathsf{K}$-linear map from $\mathcal{A}$ to $\mathcal{A}$. As such, it can be represented by a unique matrix $T_i \in \mathsf{K}^{\nu\times\nu}$, such that if

$$b = \sum_{1\le j\le\nu} \beta_j w_j \qquad \text{and} \qquad c = w_i b = \sum_{1\le j\le\nu} \gamma_j w_j$$

for $\beta_1, \ldots, \beta_\nu, \gamma_1, \ldots, \gamma_\nu \in \mathsf{K}$, then

$$T_i \cdot (\beta_1, \ldots, \beta_\nu)^t = (\gamma_1, \ldots, \gamma_\nu)^t.$$

It follows that an isomorphism $\varphi$ exists between $\mathcal{A}$ and the subalgebra $\mathcal{M}$ of $\mathsf{K}^{\nu\times\nu}$ with basis $T_1, \ldots, T_\nu \in \mathsf{K}^{\nu\times\nu}$, where

$$\varphi(b) = \varphi(\sum_{1\le j\le\nu} \beta_j w_j) = \sum_{1\le j\le\nu} \beta_j \varphi(w_j) = \sum_{1\le j\le\nu} \beta_j T_j \in \mathsf{K}^{\nu\times\nu}.$$

We call $\mathcal{M}$ the regular representation of $\mathcal{A}$ with respect to $\mathfrak{B}$. The matrix algebra $\mathcal{M}$ acts faithfully on $\mathcal{A}$ by matrix-vector product (where $\mathcal{A}$ is represented by vectors in $\mathsf{K}^\nu$ with respect to the basis $\mathfrak{B}$) and $\varphi(1) = I \in \mathsf{K}^{\nu\times\nu}$, the $\nu\times\nu$ identity matrix. Also, the minimal polynomial in $\mathsf{K}[x]$ of $b \in \mathcal{A}$ is equal to the minimal polynomial of the matrix $\varphi(b)$. This embedding has a number of useful consequences. First, if the matrix representation $B = \varphi(b)$ of $b$ is non-singular (as a matrix over $\mathsf{K}$), then $b$ has an inverse in $\mathcal{A}$, which is actually a power of $b$ since $\mathcal{A}$ is finite. Similarly, if $b \in \mathcal{A}$ and $f = \min_{\mathsf{K}}(b)$ is not divisible by $x$, then $B$ is invertible. Also, if $f = \min_{\mathsf{K}}(b)$ then $\mathsf{K}[b] \cong \mathsf{K}[x]/(f)$, and if $f$ is not divisible by $x$, then $b$ is invertible (all eigenvalues of $B$ are non-zero).

When the algebra $\mathcal{A}$ is represented as a subalgebra of $\mathsf{K}^{m\times m}$ for some $m \ge 1$, we can apply our algorithms from Chapters 2 and 5, for computing the minimal polynomial and evaluating a polynomial at a matrix, to improve the efficiency of the algorithm `ZeroDivisor`. We assume in the next theorem that the probability that the algorithm `ZeroDivisor` fails is an (absolute) constant $\varrho < 1$. It is shown in the next section that, in fact, $\varrho \leq 3/4$.

**Theorem 7.4.** *Let $\mathcal{A} \subseteq \mathsf{K}^{m\times m}$ be an associative algebra (with identity) of dimension $\nu$ over a finite field $\mathsf{K} \cong \mathbb{F}_q$. Assume $\mathcal{A}$ is given by a basis $T_1, \ldots, T_\nu \in \mathcal{A}$. Using the algorithm* `ZeroDivisor` *we can find a pair $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ such that $b_1 b_2 = 0$, represented with respect to the supplied basis, or, if $\mathcal{A}$ is a field, report this fact. Let $\mu = \min(m, \nu)$. The expected number of operations in $\mathsf{K}$ required is as follows:*

- *(i)* $O(\mathrm{MM}(m)\log m + m\mathrm{M}(m) + \mathrm{M}(\mu^2)\log\mu + \mathrm{M}(\mu)\log q)$ *or* $O\tilde{\ }(\mathrm{MM}(m) + \mu\log q)$ *when* $q \geq m^2$*;*
- *(ii)* $O\left((\mathrm{MM}(m)\log m + m\mathrm{M}(m)) \cdot \mathrm{M}(\log_q m)\log\log_q m + \mathrm{M}(\mu^2)\log\mu + \mathrm{M}(\mu)\log q\right)$ *or* $O\tilde{\ }(\mathrm{MM}(m) + \nu\log q)$ *when* $q < m^2$*;*
- *(iii)* $O(m^3 + \mu^2\log q)$ *using standard polynomial and matrix arithmetic when* $q \geq m^2$*;*
- *(iv)* $O(m^3\log_q m + \mu^2\log q)$ *using standard polynomial and matrix arithmetic when* $q < m^2$.

**Proof.** The algorithm is exactly as stated in `ZeroDivisor`, we simply make use of fast algorithms for computing the minimal polynomial of a matrix from Theorems 2.12, 2.13 and 3.5, and algorithms for evaluating a polynomial at a matrix from Theorems 5.2 and 5.5. The reason for introducing the variable $\mu = \min(m, \nu)$ is that the degree of the minimal polynomial of any element of $\mathcal{A}$ is at most $m$, since the degree of the minimal polynomial of any $m \times m$ matrix is at most $m$, and at most $\nu$, since the minimal polynomial of any element in an associative algebra is at most the dimension of that algebra. The degree of the minimal polynomial is thus at most $\mu$, and this is the polynomial we must factor completely. For the asymptotically fast algorithms of (i) and (ii) we use von zur Gathen & Shoup's (1992) factoring algorithm, which requires an expected number of $O(\mathrm{M}(\nu^2)\log\nu + \mathrm{M}(\nu)\log q)$ operations in $\mathsf{K}$ to completely factor a degree $\nu$ polynomial in $\mathsf{K}[x]$. For parts (iii) and (iv) we employ Berlekamp's (1970) algorithm for factoring polynomials in $\mathsf{K}[x]$. $\square$

## 7.2 The Density of Reducible Elements

In this section a lower bound of $1/4$ is shown on the probability that the algorithm `ZeroDivisor` presented in the previous section returns an answer (which is guaranteed to be correct — the algorithm is of the Las Vegas type). In fact, we show that except for non-commutative local algebras (to be defined later) that the probability of success is at least $1/2$. This proof is quite involved, the hardest case being when $\mathcal{A}$ has a non-trivial zero divisor. First, consider the case when $\mathcal{A}$ is a field extension of $\mathsf{K}$. We will assume throughout this section that $\mathsf{K}$ is a field of size $q \in \mathbb{N}$, where $q$ is a prime power.

**Theorem 7.5.** *Let $\mathcal{A}$ be field of dimension $\nu$ over $\mathsf{K}$. The algorithm* `ZeroDivisor` *with input $\mathcal{A}$ reports failure with probability $\varrho \le 1/4$.*

**Proof.** An element $b \in \mathcal{A}$ always has a minimal polynomial of degree dividing $\nu$, since $\mathsf{K}[b]$ is a subfield of $\mathcal{A}$, and $[\mathsf{K}[b] : \mathsf{K}]$ divides $[\mathcal{A} : \mathsf{K}]$. The elements $b \in \mathcal{A}$ such that $\deg \min_{\mathsf{K}}(b) = \nu$ are exactly those that satisfy an irreducible polynomial in $\mathsf{K}[x]$ of degree $\nu$, and $\nu$ distinct elements of $\mathcal{A}$ satisfy each such polynomial. Applying (3.1),

$$\mathrm{N}_q(\nu) \ge \frac{q^\nu}{\nu} - \frac{q}{q-1} \cdot \frac{q^{\nu/2}-1}{\nu} = \frac{q^\nu}{2\nu} \cdot \left(2 - 2 \cdot \frac{q^{\nu/2}-1}{(q-1)q^{\nu-1}}\right) \ge \frac{q^\nu}{2\nu},$$

where it is easily verified that $(q^{\nu/2}-1)/((q-1)q^{\nu-1}) \le 1/2$. The number of $b \in \mathcal{A}$ satisfying irreducible polynomials in $\mathsf{K}[x]$ of degree $\nu$ is $\nu \mathrm{N}_q(\nu) \ge \nu \cdot q^\nu/(2\nu) = q^\nu/2$. Thus, the number of $b \in \mathcal{A}$ with $\deg \min_{\mathsf{K}}(b) < \nu$ must be less than $q^\nu/2$, so $\varrho \le 1/4$ since we choose two elements $b \in \mathcal{A}$ independently and test each of them. □

Now let $\mathcal{A}$ be an algebra with at least one non-trivial zero divisor. We call an element $b \in \mathcal{A}$ *reducible* if its minimal polynomial in $\mathsf{K}[x]$ is reducible in $\mathsf{K}[x]$, and *irreducible* if its minimal polynomial in $\mathsf{K}[x]$ is irreducible in $\mathsf{K}[x]$. Define

$$R(\mathcal{A}) = \{b \in \mathcal{A} \mid b \text{ is irreducible over } \mathsf{K}\}, \quad \text{and} \quad P(\mathcal{A}) = \#R(\mathcal{A}).$$

The failure probability of the algorithm `ZeroDivisor` is at most $(P(\mathcal{A})/q^\nu)^2$, ignoring for now the possibility that $b = b_1 b_2 - b_2 b_1$ yields a zero divisor of $\mathcal{A}$ (this is only used when $\mathcal{A}$ is a local algebra over $\mathcal{A}$, that is, when $\mathcal{A}/\operatorname{Rad}(\mathcal{A})$ is a (finite) field, where $\operatorname{Rad}(\mathcal{A})$ is the Jacobson radical of $\mathcal{A}$ – see Theorem 7.12 below). Thus, we want to show that $P(\mathcal{A}) \leq q^{\nu}/\sqrt{2}$.

**Theorem 7.6.** *Let $\mathcal{A}$ be an algebra of dimension $\nu$ over $\mathsf{K}$ which is not local and possesses a non-trivial zero divisor. The probability $\varrho$ that the algorithm* `ZeroDivisor` *fails to find a non-trivial zero divisor in $\mathcal{A}$ is at most $1/2$.*

We begin by proving two theorems dealing with significant special cases. In Theorem 7.7 we prove Theorem 7.6 for $\mathcal{A}$ simple, and in Theorem 7.11 for $\mathcal{A}$ semi-simple.

**Theorem 7.7.** *If $\mathcal{A}$ is a simple algebra of dimension $\nu$ over $\mathsf{K}$, and with at least one non-trivial zero divisor, then $P(\mathcal{A}) \leq q^{\nu}/\sqrt{2}$.*

The proof is rather long and will be divided up into a number of lemmas. Since $\mathcal{A}$ is simple, it is isomorphic to a full matrix algebra $\mathsf{E}^{n\times n}$ of all $n\times n$ matrices over some algebraic extension field $\mathsf{E} \supseteq \mathsf{K}$ (see Lang 1984, Chapter 17). We set $\mu = [\mathsf{E} : \mathsf{K}]$, so $\mathsf{E} \cong \mathbb{F}_{q^{\mu}}$ and $\nu = \mu n^2$. One difficulty in the proof of Theorem 7.7 stems from the fact that we must prove the theorem true for *all* simple algebras $\mathcal{A}$ which have dimension $\nu$ over $\mathsf{K}$—an asymptotic bound will not suffice.

Let $a \in \mathcal{A}$ and $B \in \mathsf{E}^{n\times n}$ its image in $\mathsf{E}^{n\times n}$. The minimal polynomial in $\mathsf{K}[x]$ of $a \in \mathcal{A}$ is the monic $h \in \mathsf{K}[x]\setminus\{0\}$ of minimal degree such that $h(B) = 0$. This minimal polynomial is intimately related to the minimal polynomial $f \in \mathsf{E}[x]$ of the matrix $B$: $h$ is the monic polynomial of smallest degree in $\mathsf{K}[x] \setminus \{0\}$ such that $f \mid h$. We write $h = \min_{\mathsf{K}}(B) \in \mathsf{K}[x]$ and $f = \min_{\mathsf{E}}(B) \in \mathsf{E}[x]$. Theorem 7.7 is proven by showing that for at most a constant fraction of matrices $B \in \mathsf{E}^{n\times n}$ that $\min_{\mathsf{K}}(B) \in \mathsf{K}[x]$ is irreducible in $\mathsf{K}[x]$. First, we show that every matrix in $\mathsf{E}^{n\times n}$ similar to a companion matrix is similar to $q^{\mu n^2-\mu n}(1-q^{-\mu})^{1-q^{-\mu}}$ distinct matrices. This implies that a substantial fraction of all matrices in $\mathsf{E}^{n\times n}$ are similar to companion matrices in $\mathsf{E}^{n\times n}$. We then show for matrices similar to companion matrices that their minimal polynomials in $\mathsf{K}[x]$ are reducible almost as often as their minimal polynomials in $\mathsf{E}[x]$ are.

**Lemma 7.8.** *Let $\mathsf{E} = \mathbb{F}_{q^{\mu}}$ and $n \geq 2$. Let $B \in \mathsf{E}^{n\times n}$ be such that $f = \min_{\mathsf{E}}(B)$ and $\deg f = n$. Then $B$ is similar to at least $q^{\mu n^2-\mu n}(1-q^{-\mu})^{1-q^{-\mu}}$ distinct matrices in $\mathsf{E}^{n\times n}$, exactly one of which is a companion matrix.*

**Proof.** Matrices in $\mathsf{E}^{n\times n}$ whose minimal polynomials (in $\mathsf{E}[x]$) have degree $n$ are exactly those similar to the companion matrix of their minimal polynomials. As the minimal polynomial is the only invariant factor if it has degree $n$, it completely characterizes the similarity class. Since no two distinct companion matrices are similar, we know that $B$ is similar to exactly one companion matrix.

It is well known (see, for example, Hodges 1958) that the number of matrices similar to a given matrix $B \in \mathsf{E}^{n\times n}$ is the total number $\mathcal{L}(\mathsf{E}, n)$ of non-singular matrices in $\mathsf{E}^{n\times n}$ divided by the number of non-singular matrices in $\mathsf{E}^{n\times n}$ which commute with $B$. In the case of a $B \in \mathsf{E}^{n\times n}$ with $\deg \min_{\mathsf{E}}(B) = n$, it is shown by Gantmacher (1990), Section 8.2, that the only matrices commuting with $B$ are in $\mathsf{E}[B]$, whence there are at most $q^{\mu n}$ of them. From (Dickson 1901, Part II, Chapter 1), we have

$$\mathcal{L}(\mathsf{E}, n) = \prod_{0\le i<n} (q^{\mu n} - q^{\mu i}) = q^{\mu n^2} \prod_{1\le i\le n} (1 - 1/q^{\mu i}).$$

We bound $\prod_{1\le i\le n}(1 - q^{-\mu i})$ from below by considering its logarithm

$$\begin{aligned}\log \prod_{1\le i\le n} (1 - q^{-\mu i}) &= -\sum_{1\le i\le n} \sum_{j\ge 1} \frac{1}{j q^{\mu i j}} = -\sum_{j\ge 1} \frac{1}{j} \cdot \frac{1}{q^{\mu j - 1}} \left(1 - \frac{1}{q^{\mu j n}}\right) \\ &\ge -\sum_{j\ge 1} \frac{1}{j q^{\mu j}} \frac{q^{\mu j}}{q^{\mu j} - 1} = \log\left(1 - \frac{1}{q^{\mu}}\right)^{q^{\mu}/(q^{\mu}-1)}.\end{aligned}$$

Thus

$$\prod_{1\le i\le n} (1 - q^{-\mu i}) \ge \left(1 - \frac{1}{q^{\mu}}\right)^{1-1/(q^{\mu}-1)} \ge \left(1 - \frac{1}{q^{\mu}}\right)^{1-1/q^{\mu}},$$

and the number of matrices similar to $B$ is at least $q^{\mu n^2 - \mu n}(1 - q^{-\mu})^{1-q^{-\mu}}$. $\square$

We next bound the number of monic $f \in \mathsf{E}[x]$ of degree $n$ which have the property that the $h \in \mathsf{K}[x] \setminus \{0\}$ of smallest degree which $f$ divides, is irreducible in $\mathsf{K}[x]$.

First, we recall how an irreducible monic $g \in \mathsf{K}[x]$ of degree $m$ factors in $\mathsf{E}[x]$: $g = g_1 \cdots g_l$, where $g_1, \ldots, g_l \in \mathsf{E}[x]$ are distinct and monic, $d = \gcd(m, \mu)$ and $d = \deg g_1 = \cdots = \deg g_l = m/l$. The polynomials $g_1, \ldots, g_l$ are also in the same orbit under the action of the Galois group of $\mathsf{E}/\mathsf{K}$ on $\mathsf{E}[x]$ as follows. The Galois group of $\mathsf{E}/\mathsf{K}$ is generated by the Frobenius map $\sigma\colon \mathsf{E} \to \mathsf{E}$ carrying any $c \in \mathsf{E}$ to

$\sigma(c) = c^q$. An action on $\mathsf{E}[x]$ is also defined by $\sigma$, carrying $h = \sum_{0\leq i\leq r} c_i x^i$ to $\sigma(h) = \sum_{0\leq i\leq r} \sigma(c_i)x^i$, for $c_0, \ldots, c_r \in \mathsf{E}$. Define $\mathcal{I}_{\mathsf{E}}(d)$ as the set of monic irreducible polynomials of degree $d$ in $\mathsf{E}[x]$. The polynomials $g_1, \ldots, g_l$ form an orbit in $\mathcal{I}_{\mathsf{E}}(d)$ under $\sigma$, that is, there exist distinct integers $0 \leq j_1, \ldots, j_l < \mu$ such that $g_i = \sigma^{j_i}(g_1)$ for $1 \leq i \leq l$.

Now consider a polynomial $f \in \mathsf{E}[x]$ of degree $n$. The polynomial $h \in \mathsf{K}[x] \setminus \{0\}$, as defined above, is irreducible in $\mathsf{K}[x]$ if and only if all its irreducible factors in $\mathsf{E}[x]$ are distinct of some degree $d \mid n$, and all lie in the same $\sigma$-orbit of $\mathcal{I}_{\mathsf{E}}(d)$. For a fixed $d \mid n$, we next determine the number of monic polynomials of degree $n$ in $\mathsf{E}[x]$ whose irreducible factors in $\mathsf{E}[x]$ all lie in the same $\sigma$-orbit of $\mathcal{I}_{\mathsf{E}}(d)$. The size of each $\sigma$-orbit is a divisor of the order $\mu$ of $\sigma$, so we partition

$$\mathcal{I}_{\mathsf{E}}(d) = S_{\mu_1} \cup S_{\mu_2} \cup \cdots \cup S_{\mu_t},$$

where $\mu_1, \ldots, \mu_t \in \mathbb{N}$ are the distinct divisors of $\mu$ and $S_{\mu_i} \subseteq \mathcal{I}_{\mathsf{E}}(d)$ is the set of elements in $\mathcal{I}_{\mathsf{E}}(d)$ of orbit length $\mu_i$ under $\sigma$. Each $S_i$ consists of $\#S_i/\mu_i$ distinct orbits, and the polynomial $h \in \mathsf{K}[x]$ is irreducible in $\mathsf{K}[x]$ if and only if all its factors in $\mathsf{E}[x]$ are distinct and chosen from a single such orbit. Thus, if $k = n/d > \mu_i$ we cannot have $k$ distinct factors from the same orbit in $S_i$. If $k \leq \mu_i$ then there are $\binom{\mu_i}{k}$ ways of picking $k$ factors from any one orbit. Thus, when $k \leq \mu_i$, there are

$$\binom{\mu_i}{k}\frac{\#S_i}{\mu_i} \leq \binom{\mu}{k}\frac{\#S_i}{\mu}$$

polynomials $f \in \mathsf{E}[x]$ whose irreducible factors in $\mathsf{E}[x]$ are all distinct and come from a single $\sigma$-orbit in $S_i$. The total number of $f \in \mathsf{E}[x]$ whose irreducible factors in $\mathsf{E}[x]$ are all distinct and from the same $\sigma$-orbit in $\mathcal{I}_{\mathsf{E}}(d)$ is at most

$$\sum_{1\leq i\leq t}\binom{\mu_i}{k}\frac{\#S_i}{\mu_i} \leq \binom{\mu}{k}\frac{\mathrm{N}_{q^\mu}(d)}{\mu},$$

with equality when $\mu$ is prime and at least $k$. Summing this over all $d \mid n$, we obtain the total number of $f \in \mathsf{E}[x]$ whose irreducible factors in $\mathsf{E}[x]$ all come from the same

$\sigma$-orbit:

$$\begin{aligned}
\sum_{\substack{d \mid n \\ n/d \le \mu}} \binom{\mu}{n/d} \frac{\mathrm{N}_{q^\mu}(d)}{\mu} &< \sum_{\substack{d \mid n \\ n/d \le \mu}} \binom{\mu}{n/d} \frac{q^{\mu d}}{\mu d} = \sum_{\substack{k \mid n \\ k \le \mu}} \binom{\mu}{k} \frac{q^{\mu n/k} k}{\mu n} \\
&= \frac{q^{n\mu}}{n} + \sum_{\substack{k \mid n \\ 2 \le k \le \mu}} \binom{\mu}{k} \frac{q^{\mu n/k} k}{\mu n} \le \frac{q^{n\mu}}{n} + \sum_{\substack{k \mid n \\ 2 \le k \le \mu}} \binom{\mu}{2} \frac{q^{\mu n/2} \cdot 2}{\mu n} \\
&\le \frac{q^{n\mu}}{n} + \frac{(n-1)(\mu-1) q^{\mu n/2}}{\mu n}.
\end{aligned}$$

Here we make use of (3.1), which shows $\mathrm{N}_{q^\mu}(d) \le q^{\mu d}/d$. Also, in the second line of this derivation we use the fact that $\rho(k) = \binom{\mu}{k} q^{\mu n/k} k/(\mu n)$ is maximized on the interval $2 \le k < \mu$ at $k = 2$. This is demonstrated by showing that $\rho$ is strictly decreasing for $2 \le k \le \mu$, or equivalently that $\rho' = \partial\rho/\partial k$ is negative for $2 \le k \le \mu$. We derive

$$\rho' = -\frac{q^{\mu n/k} \mu! (k^2(\psi(k+1) - \psi(\mu - k + 1)) + \mu n \log q - k)}{k!(\mu - k)! \mu n k},$$

where $\psi(x) = -C + \sum_{1 \le i < x} i^{-1}$ is the poly-gamma function, the first derivative of $\log \Gamma(x) = \log((x-1)!)$, and $C \approx 0.57721566$ is Euler's constant (see Artin 1964). To show that $\rho(k) < 0$ for $2 \le k \le \mu$ we need only show that

$$\rho_0(k) = k^2(\psi(k+1) - \psi(\mu - k + 1)) + \mu n \log q - k > 0$$

for these $k$. This follows since $\mu \ge k$, whence $\mu n \log q - k > 0$, $\psi(k+1) - \psi(\mu - k + 1)$ is strictly increasing in $k$, and

$$\rho_0(2) = 4 - 4C - 4\psi(\mu - 1) + \mu n \log q > 4 - 4C - 4\log(\mu - 1) + \mu n \log q > 0$$

for all $\mu \ge 1$, using the fact that $\psi(x) \le \log x$ for all $x \ge 1$ (see Apostol 1976, Chapter 3).

We have proven the following:

**Lemma 7.9.** *Let $q$ be a prime power, $\mu \geq 1$, and $n \geq 2$. Define $\mathsf{K} = \mathbb{F}_q$ and $\mathsf{E}$ an algebraic extension of degree $\mu$ over $\mathsf{K}$, so $\mathsf{E} \cong \mathbb{F}_{q^\mu}$. The number of monic $f \in \mathsf{E}[x]$, such that the polynomial $h \in \mathsf{K}[x] \setminus \{0\}$ of smallest degree with $f \mid h$ is irreducible in $\mathsf{K}[x]$, is at most $q^{\mu n}/n + (n-1)(\mu-1)/(n\mu) \cdot q^{\mu n/2}$.*

We combine Lemmas 7.8 and 7.9 to count the number of matrices in $\mathsf{E}^{r\times r}$ whose minimal polynomials in $\mathsf{K}[x]$ are irreducible in $\mathsf{K}[x]$.

**Theorem 7.10.** *Let $q$ be a prime power, $\mu \geq 1$, and $n \geq 2$. Define $\mathsf{K} = \mathbb{F}_q$ and $\mathsf{E}$ an algebraic extension of degree $\mu$ over $\mathsf{K}$, so $\mathsf{E} \cong \mathbb{F}_{q^\mu}$. The number of matrices in $\mathsf{E}^{n\times n}$ whose minimal polynomials in $\mathsf{K}[x]$ are reducible in $\mathsf{K}[x]$ is greater than*

$$q^{\mu n^2}\left(1 - \frac{1}{n} - \frac{(n-1)(\mu-1)}{n\mu q^{\mu n/2}}\right)\cdot\left(1-\frac{1}{q^\mu}\right)^{1-1/q^\mu} \geq q^{\mu n^2}\cdot(1-1/\sqrt{2}).$$

**Proof.** By Lemma 7.9, there are at most

$$\frac{q^{\mu n}}{n} + \frac{(n-1)(\mu-1)q^{\mu n/2}}{n\mu}$$

polynomials $f \in \mathsf{E}[x]$ such that the $h \in \mathsf{K}[x] \setminus \{0\}$ of least degree such that $f \mid h$ is irreducible, and hence greater than

$$q^{\mu n} - \frac{q^{\mu n}}{n} - \frac{(n-1)(\mu-1)q^{\mu n/2}}{n\mu}$$

such that $f$ is reducible. A matrix in $\mathsf{E}^{n\times n}$ similar to a companion matrix of such an $f$ will have an reducible polynomial in $\mathsf{K}[x]$. By Lemma 7.8 a companion matrix $C_f \in \mathsf{E}^{n\times n}$ of such an $f$ is similar to at least $q^{\mu n^2 - \mu n}(1-q^{-\mu})^{1-q^{-\mu}}$ distinct matrices in $\mathsf{E}^{n\times n}$, of which $C_f$ is the only companion matrix. Thus there are at least

$$q^{\mu n^2}\left(1 - \frac{1}{n} - \frac{(n-1)(\mu-1)}{n\mu q^{\mu n/2}}\right)\cdot\left(1-\frac{1}{q^\mu}\right)^{1-1/q^\mu},$$

matrices in $\mathsf{E}^{n\times n}$ with reducible minimal polynomials in $\mathsf{K}[x]$. It is easily verified by analytic means that this formula is greater than $q^{\mu n^2}\cdot(1-1/\sqrt{2})$ for all $q \geq 2$, $\mu \geq 1$ and $n \geq 2$. $\square$

**Proof** (of Theorem 7.7). The number of elements $a \in \mathcal{A}$ with an irreducible polynomial in $\mathsf{K}[x]$ is equal to the number of matrices $B \in \mathsf{E}^{n\times n}$ with $\min_{\mathsf{K}}(B) \in \mathsf{K}[x]$ irreducible in $\mathsf{K}[x]$. By Theorem 7.10 this is at most $q^{\mu n^2}/\sqrt{2} = q^\nu/\sqrt{2}$. □

**Theorem 7.11.** *If $\mathcal{A}$ is a semi-simple algebra of dimension $\nu$ over $\mathsf{K}$ and possessing at least one non-trivial zero divisor, then $P(\mathcal{A}) \le q^\nu/\sqrt{2}$.*

**Proof.** If $\mathcal{A}$ is simple then Theorem 7.7 implies that this theorem is true, so assume $\mathcal{A}$ is not simple. The Wedderburn Decomposition Theorem (see Pierce 1982, Section 3.5) yields a decomposition of $\mathcal{A}$ as

$$\mathcal{A} \cong \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_k,$$

where $\mathcal{A}_i$ is a simple algebra of dimension $\nu_i \ge 1$ for $1 \le i \le k$. This is also a decomposition of $\mathcal{A}$ as an $\mathsf{K}$-vector space, so $\nu = \nu_1 + \nu_2 + \cdots + \nu_k$, with $k \ge 2$ since $\mathcal{A}$ is not simple. Each simple component $\mathcal{A}_i$ is isomorphic to $\mathsf{E}_i^{r_i \times r_i}$, where $\mathsf{E}_i$ is a finite extension field of $\mathsf{K}$. Under this isomorphism, each $b \in \mathcal{A}$ has an image $(b_1, \ldots, b_k) \in \mathcal{A}_1 \times \cdots \times \mathcal{A}_k$, and $\min_{\mathsf{K}}(b) = \operatorname{lcm}(\min_{\mathsf{K}}(b_1), \ldots, \min_{\mathsf{K}}(b_k))$. It should be clear that $P(\mathcal{A}) \le P(\mathcal{A}_1)P(\mathcal{A}_2)\cdots P(\mathcal{A}_k)$, since an element of $\mathcal{A}$ has an irreducible minimal polynomial only if the minimal polynomial of each of its components in $\mathcal{A}_1, \ldots, \mathcal{A}_k$ is irreducible (in fact, each component must have the *same* irreducible minimal polynomial, which is not reflected in this inequality). We consider two cases in this proof:

**Case (i):** $\mathcal{A}_1, \ldots, \mathcal{A}_k$ are all fields.

Here $r_1 = r_2 = \ldots = r_k = 1$. We need only consider $\mathcal{A}_1 \cong \mathbb{F}_{q^{\nu_1}}$ and $\mathcal{A}_2 \cong \mathbb{F}_{q^{\nu_2}}$, and show that at most half the elements of $\mathcal{A}_1 \times \mathcal{A}_2$ have an irreducible minimal polynomial.

Start by determining the number of elements of $\mathcal{A}_1 \times \mathcal{A}_2$ annihilated by a single irreducible polynomial in $\mathsf{K}[x]$. If $b \in \mathcal{A}$ and $\min_{\mathsf{K}}(b)$ is irreducible over $\mathsf{K}$, then $\min_{\mathsf{K}}(b_1) = \min_{\mathsf{K}}(b_2)$, since the minimal polynomial of a field element is always irreducible. Thus $\mathsf{K}(b_1)$ and $\mathsf{K}(b_2)$ are isomorphic as fields, and $d = \deg \min_{\mathsf{K}}(b)$ divides both $\nu_1$ and $\nu_2$ since $\mathsf{K}(b_1) \subseteq \mathcal{A}_1$ and $\mathsf{K}(b_2) \subseteq \mathcal{A}_2$. In particular $d \mid \gcd(\nu_1, \nu_2)$. Any one irreducible $f \in \mathsf{K}[x]$ of degree $d$ has $d^2$ roots in $\mathcal{A}_1 \times \mathcal{A}_2$.

We now count the number of elements $a \in \mathcal{A}_1 \times \mathcal{A}_2$ for which there exists a monic irreducible $f \in \mathsf{K}[x]$ with $f(a) = 0$. If $d \in \mathbb{N}$ divides $\gcd(\nu_1, \nu_2)$, exactly $d^2 \mathrm{N}_q(d)$ elements in $\mathcal{A}_1 \times \mathcal{A}_2$ are annihilated by polynomials in $\mathcal{I}_{\mathsf{K}}(d)$. On the other hand, if $d \nmid \gcd(\nu_1, \nu_2)$ then no element of $\mathcal{A}_1 \times \mathcal{A}_2$ is annihilated by an irreducible polynomial of degree $d$ in $\mathsf{K}[x]$. Making use of the fact that $\sum_{d \mid t} d\mathrm{N}_q(d) = q^t$ for any $t \geq 1$ (see Lidl & Niederreiter 1983 Corollary 3.21),

$$\begin{aligned} P(\mathcal{A}_1 \times \mathcal{A}_2) = \sum_{d \mid \gcd(\nu_1,\nu_2)} d^2 \mathrm{N}_q(d) &\leq \min(\nu_1, \nu_2) \sum_{d \mid \min(\nu_1,\nu_2)} d\mathrm{N}_q(d) \\ &\leq \min(\nu_1, \nu_2) q^{\min(\nu_1,\nu_2)} \leq q^{2\min(\nu_1,\nu_2)}/\sqrt{2} \leq q^{\nu_1+\nu_2}/\sqrt{2}, \end{aligned}$$

using the easily proven fact that $z \leq q^z/\sqrt{2}$ for all $z \geq 1$. Finally,

$$\begin{aligned} P(\mathcal{A}) &\leq P(\mathcal{A}_1 \times \mathcal{A}_2) P(\mathcal{A}_3 \times \cdots \times \mathcal{A}_k) \leq P(\mathcal{A}_1 \times \mathcal{A}_2) q^{\nu-\nu_1-\nu_2} \\ &\leq (q^{\nu_1+\nu_2})/\sqrt{2} \cdot q^{\nu-\nu_1-\nu_2} = q^{\nu}/\sqrt{2}. \end{aligned}$$

**Case (ii):** At least one of $\mathcal{A}_1, \ldots, \mathcal{A}_k$ is not a field.

Without loss of generality we can assume that $\mathcal{A}_1$ is not a field. Thus $\mathcal{A}_1 \cong \mathsf{E}_1^{r_1 \times r_1}$, where $r_1 \geq 2$, and $\mathsf{E}_1$ is an extension field of $\mathsf{K}$. By Theorem 7.7, $P(\mathcal{A}_1) \leq q^{\nu_1}/\sqrt{2}$, and

$$P(\mathcal{A}) \leq P(\mathcal{A}_1) P(\mathcal{A}_2) \cdots P(\mathcal{A}_k) \leq P(\mathcal{A}_1) q^{\nu-\nu_1} \leq (q^{\nu}/\sqrt{2}) \cdot q^{\nu-\nu_1} = q^{\nu}/\sqrt{2}.$$

□

The proof of Theorem 7.6 is completed by showing its validity when $\mathcal{A}$ is not semi-simple or local.

**Proof** (of Theorem 7.6). If $\mathcal{A}$ is semi-simple, Theorem 7.11 implies that the theorem is true, so assume $\mathcal{A}$ is not semi-simple, i.e., $\mathrm{Rad}(\mathcal{A}) \neq \{0\}$.

$\mathrm{Rad}(\mathcal{A})$ is a nil-potent subalgebra of $\mathcal{A}$; that is, for all $c \in \mathrm{Rad}(\mathcal{A})$ the minimal polynomial of $c$ in $\mathsf{K}[x]$ is a power of $x$. By the Wedderburn-Malcev Principal Theorem (see McDonald 1974, Theorem 8.28) $\mathcal{A} = \mathcal{O} + \mathrm{Rad}(\mathcal{A})$, where $\mathcal{O} \cong \mathcal{A}/\mathrm{Rad}(\mathcal{A})$ is a semi-simple subalgebra of $\mathcal{A}$, and $\mathcal{O} \cap \mathrm{Rad}(\mathcal{A}) = \{0\}$. Thus, every $a \in \mathcal{A}$ can be written uniquely as $a = b + c$, where $b \in \mathcal{O}$ and $c \in \mathrm{Rad}(\mathcal{A})$. If $a$ has minimal polynomial $f \in \mathsf{K}[x]$, observe that

$$0 = f(a) = f(b+c) = f(b) + [f(b+c) - f(b)],$$

and since every term in the expansion of $(b+c)^i - b^i$ for $i \geq 1$ contains a positive power of $c$, so too does every term in the expansion of $f(b+c) - f(b)$. Since $\mathrm{Rad}(\mathcal{A})$ is an ideal in $\mathcal{A}$ and $c \in \mathrm{Rad}(\mathcal{A})$, $f(b+c) - f(b) \in \mathrm{Rad}(\mathcal{A})$. Thus $f(b) = 0$.

If $\mathcal{O}$ has dimension $\tau > 0$ over $\mathsf{K}$, and is semi-simple and not a field, then the number of elements of $\mathcal{O}$ whose minimal polynomial over $\mathsf{K}$ is irreducible is at most $q^\tau/\sqrt{2}$ by Theorem 7.11 ($\mathcal{O}$ cannot be a field since $\mathcal{A}$ is not local). If $a \in \mathcal{A}$ has an irreducible minimal polynomial then its component in $\mathcal{O}$ must have an irreducible minimal polynomial as well, whence

$$P(\mathcal{A}) \leq q^\tau/\sqrt{2} \cdot q^{\nu-\tau} = q^\nu/\sqrt{2}.$$

Finally, if $\mathcal{A} = \mathrm{Rad}(\mathcal{A})$ then each element of $\mathcal{A}$ has minimal polynomial $x^i$ for some $i > 0$, and the only element with minimal polynomial $x$ is 0. In this case $P(\mathcal{A}) = 1$. $\square$

Now consider the case when $\mathcal{A}$ is a local algebra over $\mathsf{K}$, of dimension $\nu$.

**Theorem 7.12.** *Let $\mathcal{A}$ be a local algebra of dimension $\nu$ over $\mathsf{K}$, with at least one non-trivial zero divisor. If $\mathcal{A}$ is commutative then the probability $\varrho$ that the algorithm* `ZeroDivisor` *fails is less than $1/4$. If $\mathcal{A}$ is a non-commutative then $\varrho \leq 3/4$.*

**Proof.** First, we prove $P(\mathcal{A}) \leq q^\nu/\sqrt{2}$ when $\mathcal{A}$ is commutative. The Wedderburn-Malcev Principal Theorem gives a decomposition $\mathcal{A} = \mathcal{O} + \mathrm{Rad}(\mathcal{A})$, where $\mathcal{O}$ is a subalgebra of $\mathcal{A}$ isomorphic to $\mathcal{A}/\mathrm{Rad}(\mathcal{A})$, and $\mathcal{O} \cap \mathrm{Rad}(\mathcal{A}) = \{0\}$. Since $\mathcal{A}$ is local, $\mathcal{O}$ is a finite field of dimension $\tau$ over $\mathsf{K}$, say $\mathcal{O} \cong \mathbb{F}_{q^\tau}$ for some $\tau \geq 1$. The algebra $\mathcal{A}$ possesses a non-trivial zero divisor, so it is not a field and $\mathrm{Rad}(\mathcal{A}) \neq \{0\}$. Let $r > 1$ be the nullity of $\mathrm{Rad}(R)$, the smallest integer $r$ such that $\mathrm{Rad}(R)^r = \{0\}$.

For any $a \in \mathcal{A}$, suppose $f = \min_{\mathsf{K}}(a) \in \mathsf{K}[x]$ is irreducible of degree $n$. We prove this implies $a \in \mathcal{O}$. Consider the subalgebra $\mathsf{K}[a] \subsetneq \mathcal{A}$. Since the minimal polynomial of $a$ is irreducible, $\mathsf{K}[a] \cong \mathbb{F}_{q^n}$ is a field extension of $\mathsf{K}$, and by Fermat's Little Theorem

$a^{q^n} = a$. Let

$$t = \min_{t_0 > 0} \left\{ q^{t_0} \geq r \text{ and } t_0 \equiv 0 \bmod n \right\},$$

and consider the ring morphism $\phi : \mathcal{A} \to \mathcal{A}$ defined by $\phi(z) = z^{q^t}$ for any $z \in \mathcal{A}$ (raising an element to the $q$th power is an $\mathsf{K}$-linear map since $q$ is a power of the characteristic $p$ of $\mathcal{A}$, and $\mathcal{A}$ is commutative). Since $t \equiv 0 \bmod n$, we know $\phi(a) = a^{q^t} = a^{q^n} = a$. Suppose $a = b + c$ with $b \in \mathcal{O}$ and $c \in \mathrm{Rad}(\mathcal{A})$. Then

$$a = \phi(a) = \phi(b + c) = \phi(b) + \phi(c) = b^{q^t} + c^{q^t} = b^{q^t},$$

since $q^t$ is greater than the nullity of $\mathrm{Rad}(\mathcal{A})$ so $c^{q^t} = 0$. This implies $a \in \mathcal{O}$ since $a = b^{q^t}$ and $\mathcal{O}$ is a subalgebra of $\mathcal{A}$. Thus, the only elements of $\mathcal{A}$ with irreducible polynomials are those in $\mathcal{O}$. Since $\mathcal{O} \subsetneq \mathcal{A}$ it follows that $P(\mathcal{A}) \leq q^{\nu-1} \leq q^\nu/2$.

When $\mathcal{A}$ is non-commutative we note $b = a_1a_2 - a_2a_1 \in \mathrm{Rad}(\mathcal{A})$ since $\mathcal{A}/\mathrm{Rad}(\mathcal{A})$ is a field. Hence the minimal polynomial of $b$ is $x^i$ for some $i \geq 1$, and $i \geq 2$ if $b \neq 0$. We must show that at least $q^{2\nu}/4$ pairs $a_1, a_2 \in \mathcal{A}$ satisfy $a_1a_2 - a_2a_1 \neq 0$. The centre $C$ of $\mathcal{A}$ is a subalgebra of $\mathcal{A}$ with at most $q^{\nu-1}$ elements since $\mathcal{A}$ is non-commutative. For every $a_1 \in \mathcal{A} \setminus C$, the nullspace of the linear map $\varphi: \mathcal{A} \to \mathcal{A}$ defined by $\varphi(x) = a_1x - xa_1$ has at most $q^{\nu-1}$ elements since $a_1 \neq C$. Thus there are at least $(q^\nu - q^{\nu-1})^2$ pairs $a_1, a_2 \in \mathcal{A}$ with $a_1a_2 - a_2a_1 \neq 0$, and $(q^\nu - q^{\nu-1})^2/q^{2\nu} \geq 1/4$, so the probability $\varrho$ of the algorithm failing is at most $3/4$. □

For any algebra $\mathcal{A}$, the failure probability $\varrho$ of the algorithm `ZeroDivisor` is bounded by $\varrho \leq 3/4$, using Theorem 7.5 when $\mathcal{A}$ is a field and Theorems 7.6 and 7.12 when it is not. This yields the following corollary to Theorem 7.3.

**Corollary 7.13** (to Theorem 7.3). *Let $\mathcal{A}$ be an algebra of dimension $\nu$ over $\mathsf{K} = \mathbb{F}_q$. The algorithm* `ZeroDivisor` *requires $O(\nu\chi + \mathrm{MM}(\nu) + \mathrm{M}(\nu)\log q)$ operations in $\mathsf{K}$ to determine whether $\mathcal{A}$ is a field extension of $\mathsf{K}$, or to produce $b_1, b_2 \in \mathcal{A} \setminus \{0\}$ with $b_1b_2 = 0$ (where $\chi$ is the number of operations in $\mathsf{K}$ required for a single multiplication in $\mathcal{A}$), or to fail with probability at most $3/4$.*

# Chapter 8.

# Factoring in skew-polynomial rings

A central problem in computer algebra is factoring polynomials in $\mathsf{F}[x]$, where $\mathsf{F}$ is a finite field and $x$ is an indeterminate. In this chapter we present efficient factorization algorithms in a natural non-commutative generalization of the ring $\mathsf{F}[x]$, the skew-polynomial ring $\mathsf{F}[x;\sigma]$, where $\sigma\colon \mathsf{F} \to \mathsf{F}$ is a field automorphism. $\mathsf{F}[x;\sigma]$ is the ring of all polynomials in $\mathsf{F}[x]$ under the usual component-wise addition, and multiplication defined by $xa = \sigma(a)x$ for any $a \in \mathsf{F}$. For example, if

$$
\begin{aligned}
f &= x^2 + a_1x + a_0 \in \mathsf{F}[x;\sigma],\\
g &= x + b_0 \in \mathsf{F}[x;\sigma],
\end{aligned}
$$

then

$$
\begin{aligned}
f + g &= x^2 + (a_1 + 1)x + (a_0 + b_0),\\
fg &= x^3 + (a_1 + \sigma^2(b_0))x^2 + (a_1\sigma(b_0) + a_0)x + a_0b_0,\\
gf &= x^3 + (\sigma(a_1) + b_0)x^2 + (a_1b_0 + \sigma(a_0))x + a_0b_0,
\end{aligned}
$$

where $\sigma^2(a) = \sigma(\sigma(a))$ for any $a \in \mathsf{F}$. When $\sigma = id$, the identity automorphism on $\mathsf{F}$, the ring $\mathsf{F}[x;\sigma]$ is the usual ring of polynomials $\mathsf{F}[x]$ with $xa = ax$ for all $a \in \mathsf{F}$. Skew-polynomial rings have been studied since Ore (1933) and complete treatments are found in (Jacobson, 1943), (McDonald, 1974), and (Cohn, 1985). As far as we know this is the first examination of the computational aspects of these rings, and this work first appears in (Giesbrecht, 1992a).

Assume throughout this chapter that $\mathsf{F}$ has size $p^\delta$, where $p$ is a prime and $\delta \geq 1$. For any $f, g \in \mathsf{F}[x;\sigma]$ we find that $\deg(fg) = \deg f + \deg g$, where $\deg\colon \mathsf{F}[x;\sigma]\setminus\{0\} \to \mathbb{N}$ is the usual polynomial degree function. This implies $\mathsf{F}[x;\sigma]$ is integral (i.e., zero is the only zero divisor), and while not in general a unique factorization domain, it is a principal left ideal ring endowed with a right Euclidean algorithm (see Section 8.1). As in the commutative case, a non-zero $f \in \mathsf{F}[x;\sigma]$ is *irreducible* if whenever $f = gh$ for some non-zero $g, h \in \mathsf{F}[x;\sigma]$, then either $\deg g = 0$ or $\deg h = 0$. It follows that any $f \in \mathsf{F}[x;\sigma]$ can be written as $f = f_1 \cdots f_k$, where $f_1, \ldots, f_k \in \mathsf{F}[x;\sigma]$ are irreducible. This factorization may not be unique, and adjacent factors may not be interchangeable. Consider two factoring problems:

(i) The complete factorization problem: given any $f \in \mathsf{F}[x;\sigma]$, find irreducible $f_1, \ldots, f_k \in \mathsf{F}[x;\sigma]$ such that $f = f_1 \cdots f_k$.

(ii) The bi-factorization problem: given $f \in \mathsf{F}[x;\sigma]$ and an integer $s < \deg f$, determine if there exist $g, h \in \mathsf{F}[x;\sigma]$ with $f = gh$ and $\deg h = s$, and if so, find such $g$ and $h$.

These two notions of factorizations are computationally equivalent by polynomial-time reductions in a commutative unique factorization domain. However, when we have neither commutativity nor unique factorization (as is the case with skew-polynomial rings), this separation of the factoring problem into two cases more completely captures the full complexity of factoring.

In Sections 8.2 and 8.3 we give a reduction from the complete factorization problem for $f \in \mathsf{F}[x;\sigma]$ to the problem of determining whether a finite dimensional associative algebra $\mathcal{A}$ over a finite field $\mathsf{K}$ possesses any non-zero zero divisors, and if so, finding a pair multiplying to zero. This reduction is deterministic and requires a number of operations in $\mathsf{F}$ which is polynomial in $n = \deg f$ and $\delta \log p$. The problem of finding zero divisors in a finite algebra is examined in Chapter 7 of this thesis.

The bi-factorization problem in $\mathsf{F}[x;\sigma]$ is reduced in Section 8.4 to the complete factorization problem: given $f \in \mathsf{F}[x;\sigma]$ and $s < n = \deg f$, we can determine if there exist $g, h \in \mathsf{F}[x;\sigma]$ such that $f = gh$ and $\deg h = s$ with $(n\delta \log p)^{O(1)}$ operations in $\mathsf{F}$ plus the cost of completely factoring polynomials in $\mathsf{F}[x;\sigma]$ of total degree $O(n)$. This yields algorithms for bi-factorization which require $(n\delta p)^{O(1)}$ operations in $\mathsf{F}$, and Las Vegas type probabilistic algorithms which require $(n\delta \log p)^{O(1)}$ operations in $\mathsf{F}$. Applying the faster algorithm for solving the zero-divisor problem from Section 7.1 yields a significantly faster Las Vegas type algorithm for bifactorization, which requires $n^4 \cdot (\delta \log p \log n)^{O(1)}$ operations in $\mathsf{F}$.

## Applications of Skew-Polynomial Rings

An application of skew-polynomials is to the problem of functionally decomposing a class of polynomials which had previously defied polynomial-time decomposition algorithms. Algorithms which functionally decompose polynomials have received considerable attention lately. Given $f \in \mathsf{F}[\lambda]$ in an indeterminate $\lambda$, the problem is to determine polynomials $g, h \in \mathsf{F}[\lambda]$ of given degree such that $f = g \circ h = g(h(\lambda))$. Kozen & Landau (1989) and von zur Gathen, Kozen & Landau (1987) present polynomial-time (in $\deg f$) solutions to this problem in the "tame" case, when the characteristic $p$ of $\mathsf{F}$ does not divide $\deg g$ (see also von zur Gathen 1990a). In the "wild" case, when $p \mid \deg g$, no general algorithm is known, though partial solutions are given by von zur Gathen (1990b) and Zippel (1991). A very wild type of polynomial is the set of *linearized polynomials* over $\mathsf{F}$, those of the form $\sum_{0 \le i \le n} a_i \lambda^{p^i}$ (where $a_0, \ldots, a_n \in \mathsf{F}$). It turns out that whenever $g, h \in \mathsf{F}[\lambda]$ are such that $f = g \circ h$ then $\deg g = p^r$ for some $r \in \mathbb{N}$, i.e., all functional decompositions of linearized polynomials are wild! In Section 8.5 we present very fast algorithms for the functional decomposition of linearized polynomials, which run in time polynomial in $\log \deg f$.

## Representing Skew Polynomial Rings

We now characterize explicitly any skew-polynomial ring $\mathsf{F}[x; \sigma]$ over a finite field $\mathsf{F}$. The automorphism $\sigma \colon \mathsf{F} \to \mathsf{F}$ fixes some maximum subfield $\mathsf{K}$ of $\mathsf{F}$, and if $[\mathsf{K} : \mathbb{F}_p] = \eta$ then $\mathsf{K} \cong \mathbb{F}_q$ where $q = p^\eta$. The only automorphisms of $\mathsf{F}$ fixing $\mathsf{K}$ are iterates of the Frobenius map $\tau \colon \mathsf{F} \to \mathsf{F}$ of $\mathsf{F}/\mathsf{K}$, defined by $\tau(a) = a^q$ for all $a \in \mathsf{F}$. Thus $\sigma$ must have the form $\sigma(a) = \tau^\kappa(a) = a^{q^\kappa}$ for all $a \in \mathsf{F}$, where $\kappa < \mu = [\mathsf{F} : \mathsf{K}]$. Furthermore, since $\mathsf{K}$ is the largest subfield of $\mathsf{F}$ fixed by $\sigma$, $\gcd(\mu, \kappa) = 1$.

Part of the input to our algorithms is some auxiliary information to describe $\mathsf{F}[x; \sigma]$: a prime $p$, the integers $\eta$ and $\mu$ such that $[\mathsf{F} : \mathsf{K}] = \mu$ and $[\mathsf{K} : \mathbb{F}_p] = \eta$, and a *description* of the fields $\mathsf{K}$ and $\mathsf{F}$. The description of $\mathsf{K}$ consists of a polynomial $\Gamma_{\mathsf{F}} \in \mathbb{F}_p[x]$ of degree $\eta$ which is irreducible over $\mathbb{F}_p$. We identify $\mathsf{K} = \mathbb{F}_p[x]/(\Gamma_{\mathsf{F}}) \cong \mathbb{F}_q$, so that $\mathsf{K}$ has basis $\mathcal{B}_{\mathsf{K}} = \{1, \Theta_{\mathsf{F}}, \Theta_{\mathsf{F}}^2, \ldots, \Theta_{\mathsf{F}}^{\eta-1}\}$ as an $\mathbb{F}_p$-vector space, where $\Theta_{\mathsf{F}} = x \bmod \Gamma_{\mathsf{F}}$ and $\mathsf{K} = \mathbb{F}_p[\Theta_{\mathsf{F}}]$. The field $\mathsf{F}$ is described as an extension of $\mathsf{K}$ by a polynomial $\Gamma_{\mathsf{K}} \in \mathsf{K}[x]$ of degree $\mu$, which is irreducible over $\mathsf{K}$. Identify $\mathsf{F} = \mathsf{K}[x]/(\Gamma_{\mathsf{K}})$, so $\mathsf{F}$ has basis $\mathcal{B}_{\mathsf{F}} = \{1, \Theta_{\mathsf{K}}, \Theta_{\mathsf{K}}^2, \ldots, \Theta_{\mathsf{K}}^{\mu-1}\}$ as an $\mathsf{K}$-vector space, where $\Theta_{\mathsf{K}} = x \bmod \Gamma_{\mathsf{K}}$ and

$\mathsf{F} = \mathsf{K}[\Theta_{\mathsf{K}}]$. We also require the element $\Theta_{\mathsf{K}}^q = \tau(\Theta_{\mathsf{K}})$, represented with respect to this basis. This will allow us to make use of von zur Gathen & Shoup's (1992) algorithm to compute all conjugates of an element in $\mathsf{K}$ over $\mathsf{F}$ with $O(\mathrm{M}(\mu)^2 \log \mu)$ additional operations in $\mathsf{F}$. Such an element can be computed with $\log q$ operations in $\mathsf{F}$ by repeated squaring, though for convenience we consider it pre-computation and do not count this cost in algorithms using this technique. The cost of computing $\tau(\Theta_{\mathsf{K}})$ is dominated by other costs in our algorithms for both complete and bi-factorization. Note that $\mathsf{F}[x;\sigma]$ is an associative $\mathsf{K}$-algebra with basis $\{\Theta_{\mathsf{K}}^i x^j \mid 0 \le i < \mu,\ j \ge 0\}$. It is not in general a $\mathsf{F}$-algebra, since $\mathsf{F}$ is not, in general, in the centre of $\mathsf{F}[x;\sigma]$.

## 8.1 Basic Operations in $\mathsf{F}[x;\sigma]$

A brief development of the theory of skew-polynomial rings follows, along with algorithms implementing aspects of this theory when appropriate. We begin with an easy observation on the costs of addition and multiplication in $\mathsf{F}[x;\sigma]$. Let

$$f = \sum_{0 \le i \le n} a_i x^i, \qquad g = \sum_{0 \le j \le r} b_j x^{p^j}, \tag{8.1}$$

with $a_0, \ldots, a_n, b_0, \ldots, b_r \in \mathsf{F}$ and $a_n, b_r \ne 0$. Without loss of generality we can assume that $r \le n$. Obviously $f + g$ can be computed with $O(n)$ operations in $\mathsf{F}$ or $O(n\mu)$ operations in $\mathsf{K}$. To compute $fg$ we expand

$$fg = \sum_{0 \le i \le n} \sum_{0 \le j \le r} a_i x^i b_j x^j = \sum_{0 \le i \le n} \sum_{0 \le j \le r} a_i \sigma^i(b_j) x^{i+j}.$$

Compute $\sigma^i(b_j)$ for $0 \le i < \mu$ and $0 \le j \le r$ with $O\tilde{}(r\mathrm{M}(\mu)^2 \log \mu)$ operations in $\mathsf{K}$, as described in the introduction. Next compute the $n^2$ products in $\mathsf{F}$ to obtain $fg$.

**Lemma 8.1.** *Given $f, g \in \mathsf{F}[x;\sigma]$, each of degree at most $n$, we can compute $f + g$ with $O(n\mu)$ operations in $\mathsf{K}$, and $fg$ with $O(n^2\mathrm{M}(\mu) + n\mu\mathrm{M}(\mu)\log \mu)$ or $O\tilde{}(n^2\mu + n\mu^2)$ operations in $\mathsf{K}$.*

The skew-polynomial ring $\mathsf{F}[x;\sigma]$ has a right division algorithm, and in fact a (right) Euclidean algorithm. The right division algorithm is analogous to the usual one in $\mathsf{F}[x]$. Let $f, g \in \mathsf{F}[x;\sigma]$ be as in (8.1) with $g \ne 0$: we want to find $Q, R \in \mathsf{F}[x;\sigma]$

such that $f = Qg + R$ and $\deg R < \deg g$ or $R = 0$. The algorithm is trivial if $n < r$ —we know $Q = 0$ and $R = f$—so assume $n \geq r$. Let $f^{(n)} = f$, and for $n \geq i \geq r$ define $h^{(i)} = (\bar{a}_i/\sigma^{i-r}(b_r)) \cdot x^{i-r}$, where $\bar{a}_i$ is the coefficient of $x^i$ in $f^{(i)}$. Next define $f^{(i-1)} = f^{(i)} - h^{(i)}g \in \mathsf{F}[x;\sigma]$, whence $f^{(i)} = h^{(i)}g + f^{(i-1)}$ and $\deg f^{(i-1)} < \deg f^{(i)}$. Computing $h^{(n)}, f^{(n-1)}, h^{(n-1)}, f^{(n-1)}, \ldots, h^{(r)}, f^{(r-1)}$ in sequence, we get $f = Qg + R$ where $Q = h^{(n)} + h^{(n-1)} + \cdots + h^{(r)}$ and $R = f^{(r-1)}$, with $\deg R < \deg g$ or $R = 0$. The $Q$ and $R$ obtained in the division algorithm are unique, as they are in $\mathsf{F}[x]$.

**Lemma 8.2.** *If $f, g \in \mathsf{F}[x;\sigma]$ with $n = \deg f$, $r = \deg g$, and $g \neq 0$, then computing $Q, R \in \mathsf{F}[x;\sigma]$ such that $f = Qg + R$ and $\deg R < \deg g$ or $R = 0$ requires $O(r(n-r)\mathrm{M}(\mu) + r\mu\mathrm{M}(\mu)\log\mu)$ or $O\tilde{\ }(r(n-r)\mu + r\mu^2)$ operations in $\mathsf{K}$ when $r \leq n$.*

**Proof.** Start by computing $\sigma^i(b_j)$ for $0 \leq i < \mu$ and $0 \leq j \leq r$. This requires $O(r\mu\mathrm{M}(\mu)\log\mu)$ operations in $\mathsf{K}$. At stage $i$, computing $f^{(i)} - h^{(i)}g$ requires $r$ operations in $\mathsf{F}$. There are at most $n - r$ stages requiring a total of $O(r(n-r))$ operations in $\mathsf{F}$ or $O(r(n-r)\mathrm{M}(\mu))$ operations in $\mathsf{K}$. $\square$

Using the above division algorithm, modular equivalence can be meaningfully defined: Given $f_1, f_2, g \in \mathsf{F}[x;\sigma]$, we write $f_1 \equiv f_2 \bmod g$ if and only if there exists a $Q \in \mathsf{F}[x;\sigma]$ such that $f_1 - f_2 = Qg$. It is left as an exercise to the reader that "equivalence modulo $h$" is indeed an equivalence relation in $\mathsf{F}[x;\sigma]$.

Ore (1933) first proves the main structure theorem on complete factorizations in $\mathsf{F}[x;\sigma]$, a somewhat simplified version of which is stated below (this can also be proven as a consequence of the Krull-Schmidt theorem; see Jacobson 1943).

**Theorem 8.3** (Ore). *If $f \in \mathsf{F}[x;\sigma]$ factors completely as*

$$\begin{aligned} f &= f_1 f_2 \cdots f_k \\ &= g_1 g_2 \cdots g_t, \end{aligned}$$

*where $f_1, \ldots, f_k, \ldots, g_1, \ldots, g_t \in \mathsf{F}[x;\sigma]$ are irreducible, then $k = t$ and there exists a permutation $\varphi$ of $\{1 \ldots k\}$ such that for $1 \leq i \leq k$, $\deg f_i = \deg g_{\varphi(i)}$.*

## 8.2 Common Multiples and Divisors

From the existence of a right division algorithm in $\mathsf{F}[x;\sigma]$ follows the existence of a right Euclidean scheme in the usual way (see van der Waerden 1970, pp. 55). This implies the existence of *greatest common right divisors* (defined below), the non-commutative analogue of greatest common divisors in a commutative Euclidean domain. It also gives a fast algorithm for computing greatest common right divisors.

Assume $f_1, f_2 \in \mathsf{F}[x;\sigma]$ with $\deg f_1 \geq \deg f_2$. At each stage $i > 2$, let $f_i$ be the remainder of $f_{i-2}$ divided on the right by $f_{i-1}$. We obtain a Euclidean scheme defined by $f_{i-2} = Q_{i-2}f_{i-1} + f_i$ for $3 \leq i \leq t$, where $Q_i, f_i \in \mathsf{F}[x;\sigma]$, $\deg f_i < \deg f_{i-1}$, and $f_{t-1} = Q_{t-1}f_t$. The polynomial $w = af_t \in \mathsf{F}[x;\sigma]$, where $a \in \mathsf{F}$ is chosen such that $af_t$ is monic, is the Greatest Common Right Divisor (GCRD) of $f_1$ and $f_2$, denoted $\operatorname{gcrd}(f_1, f_2)$. The GCRD $w$ is the unique monic polynomial of highest degree such that there exist $u_1, u_2 \in \mathsf{F}[x;\sigma]$ with $f_1 = u_1w$ and $f_2 = u_2w$. In the usual polynomial ring $\mathsf{F}[x] = \mathsf{F}[x;id]$ we have $\operatorname{gcrd}(f_1, f_2) = \gcd(f_1, f_2)$, the usual greatest common divisor of $f_1, f_2 \in \mathsf{F}[x]$.

**Lemma 8.4.** *If $f_1, f_2 \in \mathsf{F}[x;\sigma]$ with $n = \deg f_1 \geq \deg f_2$, then we can compute $\operatorname{gcrd}(f_1, f_2)$ with $O(n^2\mathrm{M}(\mu)\mu\log\mu)$ or $O\tilde{~}(n^2\mu^2)$ operations in $\mathsf{K}$.*

**Proof.** For $1 \leq i \leq t$ assume $n_i = \deg f_i$, so $n = n_1 \geq n_2 > n_3 > n_4 > \cdots > n_t$. For $3 \leq i \leq t$ we can compute $f_i$ with $O((n_{i-2} - n_{i-1})\mathrm{M}(\mu) + n_{i-1}\mathrm{M}(\mu)\mu\log\mu)$ operations in $\mathsf{K}$. The total cost of the algorithm is therefore

$$\begin{aligned}
&\sum_{3\leq i\leq t} (n_{i-2} - n_{i-1}) \cdot n_{i-1}\mathrm{M}(\mu) + n_{i-1}\mathrm{M}(\mu)\mu\log\mu \\
&\qquad\leq \sum_{3\leq i\leq t} (n_{i-2}^2 - n_{i-1}^2)\mathrm{M}(\mu) + \sum_{3\leq i\leq t} n_{i-1}\mathrm{M}(\mu)\mu\log\mu \\
&\qquad\leq n_1^2\mathrm{M}(\mu) + n_2^2\mathrm{M}(\mu)\mu\log\mu,
\end{aligned}$$

or $O(n^2\mathrm{M}(\mu)\mu\log\mu)$ operations in $\mathsf{K}$. $\square$

The existence of a right Euclidean algorithm implies $\mathsf{F}[x;\sigma]$ is a principal left ideal ring, that is, each left ideal is generated by a single polynomial in $\mathsf{F}[x;\sigma]$. If $\mathsf{F}[x;\sigma]f$ and $\mathsf{F}[x;\sigma]g$ are the two left ideals generated by $f, g \in \mathsf{F}[x;\sigma]$ respectively, then the ideal $\mathsf{F}[x;\sigma]\operatorname{gcrd}(f,g) = \mathsf{F}[x;\sigma]f + \mathsf{F}[x;\sigma]g$ (see Jacobson 1943, Chapter 3).

The set $\mathsf{F}[x;\sigma]f \cap \mathsf{F}[x;\sigma]g$ is also a left ideal, consisting of all polynomials in $\mathsf{F}[x;\sigma]$ which are left multiples of both $f$ and $g$. Since this left ideal is principal, it is generated by a unique monic $h = \mathrm{lclm}(f,g) \in \mathsf{F}[x;\sigma]$, the Least Common Left Multiple (LCLM) of $f$ and $g$. The LCLM $h$ is the unique monic polynomial in $\mathsf{F}[x;\sigma]$ of lowest degree such that there exist $u_1, u_2 \in \mathsf{F}[x;\sigma]$ with $h = u_1 f$ and $h = u_2 g$. Ore (1933) shows that $\deg \mathrm{lclm}(f,g) = \deg f + \deg g - \deg \mathrm{gcrd}(f,g)$. In $\mathsf{F}[x] = \mathsf{F}[x;id]$ the LCLM is simply the usual least common multiple in $\mathsf{F}[x]$, and $\deg \mathrm{lcm}(f,g) = \deg f + \deg g - \deg \gcd(f,g)$.

The LCLM $h \in \mathsf{F}[x;\sigma]$ of $f, g \in \mathsf{F}[x;\sigma]$ is computed by first computing the degree $s$ of $\mathrm{lclm}(f,g)$ using Ore's formula and the algorithm for the GCRD. We use the fact that $h$ is the (non-zero) left multiple of $f$ (in $\mathsf{F}[x;\sigma]$) of lowest degree which is equivalent to zero modulo $g$. Assuming $f$ and $g$ have degrees $n$ and $r$ respectively, we compute the sequence $x^i f = Q_i g + R_i$ for $0 \le i \le s-n$, where $Q_i, R_i \in \mathsf{F}[x;\sigma]$ and $\deg R_i < \deg g = r$. If $u = \sum_{0\le i\le s-n} c_i x^i \neq 0$, for some $c_0, \ldots, c_{s-n} \in \mathsf{F}$ such that $\sum_{0\le i\le s-n} c_i R_i = 0$, then $uf \equiv 0 \bmod g$, and $uf$ is a scalar multiple of the LCLM $h$. We solve for $u$ with linear algebra to obtain the LCLM.

**Lemma 8.5.** *Given $f, g \in \mathsf{F}[x;\sigma]$ such that $n = \deg f \ge r = \deg g$, we can compute $h = \mathrm{lclm}(f,g)$ with $O(\mathrm{MM}(n)\cdot \mathrm{M}(\mu) + n^2\mu\mathrm{M}(\mu)\log\mu)$ or $O\tilde{\ }(\mathrm{MM}(n)\mu + n^2\mu^2)$ operations in $\mathsf{K}$.*

**Proof.** To compute $R_0, \ldots, R_{s-n}$ (as above) efficiently, we first compute $w \equiv x^r \bmod g$ and $R_0 \equiv f \bmod g$, where $w, R_0 \in \mathsf{F}[x;\sigma]$ both have degree less than $r$. This can be done with $O(n^2\mathrm{M}(\mu) + r\mu\mathrm{M}(\mu)\log\mu)$ operations in $\mathsf{K}$. Next we find all conjugates of all coefficients of both $R_0$ and $w$, which requires $O(r\mu\mathrm{M}(\mu)\log\mu)$ operations in $\mathsf{F}$.

The polynomials $R_1, R_2, \ldots, R_{s-n}$ are computed in sequence. At stage $i$ suppose we know $R_i = \sum_{0\le j<r} c_{ji}x^j \equiv x^i R_0 \bmod g$ for $c_{ji} \in \mathsf{F}$, as well as all conjugates of $c_{ji}$ for $1 \le j < r$. Then

$$x^{i+1} f \equiv xR_i = \sum_{0\le j<r} \sigma(c_{ji})x^{j+1} \equiv \left(\sum_{0\le j<r-1} \sigma(c_{ji})x^{j+1}\right) + \sigma(c_{j,r-1})w = R_{i+1}.$$

The polynomial $R_{i+1} \in \mathsf{F}[x;\sigma]$ has degree less than $r$, and since we know all conjugates of all coefficients of both $R_i$ and $w$, we can compute $R_{i+1}$, as well as all conjugates of all its coefficients, with $O((s-n)r\mu)$ operations in $\mathsf{F}$. To compute $R_0, \ldots, R_{s-n}$ then requires $O(r(s-n)\mu)$ additional operations in $\mathsf{F}$ or $O(n^2\mu\mathrm{M}(\mu))$ operations in $\mathsf{K}$. Solving for $u$ with linear algebra requires $O(\mathrm{MM}(n))$ operations in $\mathsf{F}$, or $O(\mathrm{MM}(n) \cdot \mathrm{M}(\mu))$ operations in $\mathsf{K}$. $\square$

A polynomial can also be "decomposed" with respect to LCLM's as follows. Two polynomials $f_1, f_2 \in \mathsf{F}[x;\sigma]$ are *co-prime* if $\mathrm{gcrd}(f_1, f_2) = 1$. Extending this to more polynomials, say $f_1, \ldots, f_l \in \mathsf{F}[x;\sigma]$ are *mutually co-prime* if

$$\mathrm{gcrd}(f_i, \mathrm{lclm}(f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_l)) = 1$$

for $1 \le i \le l$, i.e., each $f_i$ is co-prime to the LCLM of the remaining components. This is stronger than the usual pairwise co-primality often seen for $\mathsf{F}[x]$, though the two notions are equivalent in a commutative domain. An *LCLM-decomposition* of $f \in \mathsf{F}[x;\sigma]$ is a list $(f_1, \cdots, f_l) \in \mathsf{F}[x;\sigma]^l$ of mutually co-prime polynomials such that $f = \mathrm{lclm}(f_1, \cdots, f_l)$; $f$ is *LCLM-indecomposable* if it admits no non-trivial LCLM-decompositions.

The following result of Ore (1933) captures the uniqueness of polynomial decompositions in any skew-polynomial ring.

**Theorem 8.6** (Ore, 1933). *Let $f \in \mathsf{F}[x;\sigma]$ be monic such that $f = \mathrm{lclm}(f_1, \ldots, f_l)$, where $f_1, \ldots, f_l \in \mathsf{F}[x;\sigma]$ are LCLM-indecomposable and mutually co-prime.*

*(i) If $f = \mathrm{lclm}(g_1, \ldots, g_m)$, where $g_1, \ldots, g_m \in \mathsf{F}[x;\sigma]$ are LCLM-indecomposable and mutually co-prime, then $l = m$ and there exists a permutation $\varphi$ of $\{1, \ldots, l\}$ such that $\deg f_i = \deg g_{\varphi(i)}$ for $1 \le i \le l$.*

*(ii) If, for $1 \le i \le l$, $f_i = f_{i,1} f_{i,2} \cdots f_{i,s_i}$, where each $f_{i,j} \in \mathsf{F}[x;\sigma]$ is irreducible for $1 \le j \le s_i$, and $f = h_1 h_2 \cdots h_k$, where $h_1, \ldots, h_k \in \mathsf{F}[x;\sigma]$ are irreducible, then there exists a bijection $\varphi$ from $\{1, \ldots k\}$ to $\{(i,j) \,|\, 1 \le i \le l,\ 1 \le j \le s_i\}$ such that $\deg h_e = \deg f_{\varphi(e)}$ for $1 \le e \le k$.*

## 8.3 Finding Complete Factorizations

To completely factor any $f \in \mathsf{F}[x;\sigma]$, we construct a small finite associative algebra $\mathcal{D}$ over $\mathsf{K}$ with the property that each non-zero zero divisor in $\mathcal{D}$ yields a non-zero factorization of $f$. An associative algebra $\mathcal{D}$ over $\mathsf{K}$ is a $\mathsf{K}$-vector space with a product $\times\colon \mathcal{D} \to \mathcal{D}$ such that $\mathcal{D}$ is a ring under $+$ and $\times$ (we write $ab$ for $a \times b$ for $a, b \in \mathcal{D}$). A candidate for $\mathcal{D}$ is the quotient $\mathsf{F}[x;\sigma]/\mathsf{F}[x;\sigma]f$, but it is in general only a $\mathsf{F}[x;\sigma]$-module, and not an algebra. It is only an algebra when $\mathsf{F}[x;\sigma]f$ is a two-sided ideal in $\mathsf{F}[x;\sigma]$. To regain some of the desirable structure of finite algebras, we follow Cohn (1985), Section 0.7, and introduce the concept of an eigenring. For notational brevity, let $\mathfrak{S} = \mathsf{F}[x;\sigma]$ throughout this section. Define $I(\mathfrak{S}f) = \{u \in \mathfrak{S} \mid fu \equiv 0 \bmod f\}$, the *idealizer* of $\mathfrak{S}f$. The set $I(\mathfrak{S}f)$ is the largest subalgebra of $\mathfrak{S}$ in which $\mathfrak{S}f$ is a two-sided ideal. The *eigenring* $E(\mathfrak{S}f)$ of $\mathfrak{S}f$ is defined as the quotient $E(\mathfrak{S}f) = I(\mathfrak{S}f)/\mathfrak{S}f$, a finite $\mathsf{K}$-algebra since $\mathfrak{S}$ is an $\mathsf{K}$-algebra and $\mathfrak{S}f$ a two-sided ideal in $I(\mathfrak{S}f)$. If $\deg f = n$, the eigenring $E(\mathfrak{S}f)$ is isomorphic to the $\mathsf{K}$-algebra

$$\mathcal{D} = \{u \in I(\mathfrak{S}f) \mid \deg u < n\} = \{u \in \mathfrak{S} \mid fu \equiv 0 \bmod f \text{ and } \deg u < n\},$$

under addition in $\mathfrak{S}$ and multiplication in $\mathfrak{S}$ reduced modulo $f$ (i.e., each element in $E(\mathfrak{S}f)$ is represented by its unique residue modulo $f$). The key facts about $E(\mathfrak{S}f)$, which we shall prove in the sequel, are that it is a field if and only if $f$ is irreducible, and that non-zero zero divisors in $E(\mathfrak{S}f)$ allow us to compute non-zero factors of $f$ efficiently.

To prove the desired properties of the eigenring we need to characterize the centre $\mathfrak{C}$ of $\mathfrak{S}$, and the two-sided ideals in $\mathfrak{S}$. McDonald (1974), pages 24-25, shows $\mathfrak{C} = \mathsf{K}[x^\mu;\sigma] \subseteq \mathfrak{S}$, the polynomials in $x^\mu$ with coefficients in $\mathsf{K}$. This follows since the subset of $\mathfrak{S}$ commuting with $\Theta_\mathsf{K}$ is $\mathsf{F}[x^\mu]$, while the subset of $\mathfrak{S}$ commuting with $x$ is $\mathsf{K}[x]$. The elements $\Theta_\mathsf{K}$ and $x$ in $\mathfrak{S}$ generate $\mathfrak{S}$ as an $\mathsf{K}$-algebra, whence $\mathsf{K}[x^\mu] = \mathsf{F}[x^\mu] \cap \mathsf{K}[x]$ is the centre of $\mathfrak{S}$. Letting $y = x^\mu$, we identify $\mathfrak{C} = \mathsf{K}[y]$, the usual ring of polynomials over $\mathsf{K}$ in the indeterminate $y$, so in particular, $\mathfrak{C}$ is a commutative unique factorization domain. The degree (in $x$) of any element in $\mathfrak{C}$ will always be a multiple of $\mu$. Clearly, any $\hat{f} \in \mathsf{K}[y]$ generates a two-sided ideal $\mathfrak{S}\hat{f}$. In fact, the two-sided ideals in $\mathfrak{S}$ are exactly those of the form $\mathfrak{S}(\hat{f}x^s)$ for some $\hat{f} \in \mathsf{K}[y]$ and $s \in \mathbb{N}$. The maximal (non-zero) two-sided ideals in $\mathfrak{S}$ are $\mathfrak{S}x$, and $\mathfrak{S}\hat{u}$, where $\hat{u} \in \mathsf{K}[y] \setminus \{y\}$ is irreducible as a polynomial in $y$. An important characteristic of the left ideal $\mathfrak{S}f$ is the largest two sided ideal $\mathfrak{o}$ it contains, called the *bound* for $\mathfrak{S}f$ (see Jacobson 1943, Chapter 3, Sections 5 and 6). Closely related to the bound for $\mathfrak{S}f$ is the *minimal central left multiple* $\hat{f} \in \mathsf{K}[y]$ of $f$, the polynomial in $\mathsf{K}[y]$ of minimal degree which is a left multiple of $f$. Such a polynomial always exists (we show how to construct it efficiently in Lemma 8.13), and if $\mathrm{gcrd}(f, x) = 1$ then $\mathfrak{o} = \mathfrak{S}\hat{f}$. More generally, if $f = \mathrm{lclm}(f_0, x^s)$ for some $s \geq 0$ and some $f_0 \in \mathfrak{S}$ co-prime with $x$ and with minimal central left multiple $\hat{f}_0 \in \mathsf{K}[y]$, then $\mathfrak{o} = \mathfrak{S} \cdot \hat{f}_0 x^s$.

We recall some basic facts about associative algebras before we proceed. An algebra $\mathcal{A}$ is *simple* if its only two-sided ideals are $\{0\}$ and $\mathcal{A}$, and is *semi-simple* if it is a direct sum of simple algebras. Next, we summarize some well known facts about finite simple algebras (see, for example, Lang 1984, Chapter 17).

**Fact 8.7.** *Let $\mathcal{A}$ be a finite, simple algebra of dimension $d$ over $\mathsf{K}$, and let $L$ be a left ideal in $\mathcal{A}$.*

*(i) $\mathcal{A}$ is isomorphic to $\mathsf{E}^{m\times m}$, where $m \geq 1$, $\mathsf{E}$ is the centre of $\mathcal{A}$ and a finite extension field of degree $r$ over $\mathsf{K}$, and $d = m^2 r$.*

*(ii) There exist minimal left ideals $L_1, \ldots, L_m \subseteq \mathcal{A}$ and $l \leq m$ such that $L = L_1 \oplus \cdots \oplus L_l$ and $\mathcal{A} = L_1 \oplus \cdots \oplus L_m$. Furthermore, each minimal left ideal has dimension $rm$ as an $\mathsf{K}$-vector space.*

*(iii) There exist maximal left ideals $M_1, \ldots, M_m \subseteq \mathcal{A}$ and $k \leq m$ such that $L = M_1 \cap \cdots \cap M_k$, $M_1 \cap \cdots \cap M_m = \{0\}$, and $M_i + (M_1 \cap \cdots \cap M_{i-1} \cap M_{i+1} \cap \cdots \cap M_m) = \mathcal{A}$ for $1 \leq i \leq m$. Furthermore, each maximal left ideal has dimension $rm^2 - rm$ as a $\mathsf{K}$-vector space.*

A $\mathsf{K}$-algebra of particular interest is $\mathcal{A} = \mathfrak{S}/\mathfrak{S}\hat{f}$, where $\hat{f} \in \mathsf{K}[y] \setminus \{y\}$ is irreducible as a polynomial in $y$. Since $\mathfrak{S}\hat{f}$ is a maximal two-sided ideal in $\mathfrak{S}$, $\mathcal{A}$ is a simple algebra. From $\mathfrak{S}$, $\mathcal{A}$ inherits the property of being a left principal ideal ring. Suppose $g_1 + \mathfrak{S}\hat{f}$ and $g_2 + \mathfrak{S}\hat{f}$ are in some left ideal $J \subseteq \mathcal{A}$, where $g_1, g_2 \in \mathfrak{S}$. Then there exist $h_1, h_2 \in \mathfrak{S}$ such that $h_1 g_1 + h_2 g_2 = \mathrm{gcrd}(g_1, g_2)$ and

$$(h_1 + \mathfrak{S}\hat{f})(g_1 + \mathfrak{S}\hat{f}) + (h_2 + \mathfrak{S}\hat{f})(g_2 + \mathfrak{S}\hat{f}) = \mathrm{gcrd}(g_1, g_2) + \mathfrak{S}\hat{f} \in J.$$

Thus, left ideals are closed under GCRD's (of their pre-images in $\mathfrak{S}$) and each left ideal $J$ in $\mathcal{A}$ is generated by some unique $g + \mathfrak{S}\hat{f}$, where $g \in \mathfrak{S}$ is monic of minimal degree. Since $\mathrm{gcrd}(g, \hat{f}) + \mathfrak{S}\hat{f} \in J$ and $g$ has minimal degree, $g$ is a right factor of $\hat{f}$. We call such a $g$ the *minimal modular generator* of $J$. The following lemma relates left ideals in $\mathcal{A}$ with the left ideals in $\mathfrak{S}$ generated by their minimal modular generators.

**Lemma 8.8.** *Let $J_1, J_2 \subseteq \mathcal{A}$ be non-zero left ideals in $\mathcal{A}$, with respective minimal modular generators $g_1, g_2 \in \mathfrak{S}$.*

*(i) The left ideal $J_3 = J_1 \cap J_2$ in $\mathcal{A}$ has minimal modular generator $g_3 = \mathrm{lclm}(g_1, g_2)$ if $J_3 \neq \{0\}$. Otherwise $J_3 = \{0\}$ and $\hat{f} = \mathrm{lclm}(g_1, g_2)$.*

*(ii) The left ideal $J_4 = J_1 + J_2$ in $\mathcal{A}$ has minimal modular generator $g_4 = \mathrm{gcrd}(g_1, g_2)$.*

**Proof.** To prove (i) we note that $\mathrm{lclm}(g_1, g_2) + \mathfrak{S}\hat{f} \in J_3$, so we must show that $\mathrm{lclm}(g_1, g_2)$ is the minimal modular generator of $J_3$. Suppose $h + \mathfrak{S}\hat{f} \in J_3$ for some $h \in \mathfrak{S}$. Then $h \equiv w_1 g_1 \equiv w_2 g_2 \bmod \hat{f}$ for some $w_1, w_2 \in \mathfrak{S}$. It follows that since both $g_1$ and $g_2$ are right factors of $\hat{f}$, they are also both right factors of $h$ as well. Thus $h \equiv 0 \bmod \mathrm{lclm}(g_1, g_2)$, so the pre-image in $\mathfrak{S}$ of every element in $J_3$ is in $\mathfrak{S}\,\mathrm{lclm}(g_1, g_2)$. If $\mathrm{lclm}(g_1, g_2) \neq \hat{f}$ then $\mathrm{lclm}(g_1, g_2)$ is the minimal modular generator of $J_3$. If $\mathrm{lclm}(g_1, g_2) = \hat{f}$ then $J_3 = \{0\}$.

To prove (ii), we note

$$J_1 + J_2 = (\mathfrak{S}g_1 \bmod \mathfrak{S}\hat{f}) + (\mathfrak{S}g_2 \bmod \mathfrak{S}\hat{f}) = (\mathfrak{S}g_1 + \mathfrak{S}g_2) \bmod \hat{f} = \mathfrak{S}u \bmod \mathfrak{S}\hat{f},$$

where $u = \mathrm{gcrd}(g_1, g_2)$. Thus $u + \mathfrak{S}\hat{f}$ generates $J_4$ and $\hat{f} \equiv 0 \bmod u$ since both $g_1$ and $g_2$ are right factors of $\hat{f}$. For any $h \in \mathfrak{S}$ such that $h + \mathfrak{S}\hat{f} \in J_4$, $h \equiv Qu \bmod \hat{f}$ for some $Q \in \mathfrak{S}$, and since $u$ is a right factor of $\hat{f}$ and $Qu$, $u$ is a right factor of $h$. It follows that $u$ is the polynomial in $\mathfrak{S}$ of smallest degree such that $u + \mathfrak{S}$ generates $J_4$, that is, $u$ is the minimal modular generator of $J_4$. $\square$

The next theorem characterizes the LCLM-decompositions of those $f \in \mathfrak{S}$ whose minimal central left multiples are irreducible as polynomials in $y$.

**Theorem 8.9.** *For $f \in \mathfrak{S}$, the eigenring $E(\mathfrak{S}f)$ is a (finite) field if and only if $f$ is irreducible in $\mathfrak{S}$.*

**Proof.** If $f$ is irreducible then McDonald (1974), Exercise 2.24, shows $E(\mathfrak{S}f)$ is a finite field.

We now show that if $f$ is reducible then $E(\mathfrak{S}f)$ possesses zero divisors. If $f$ is reducible and LCLM-decomposable, then $f = \mathrm{lclm}(f_1, f_2)$, where $f_1, f_2 \in \mathfrak{S} \setminus \mathsf{F}$ and $g_1 f_1 + g_2 f_2 = 1$ for some $g_1, g_2 \in \mathfrak{S}$. Note that if $h \equiv 0 \bmod f_1$ and $h \equiv 0 \bmod f_2$ for any $h \in \mathfrak{S}$, then $h \equiv 0 \bmod f$. We now construct a pair of non-zero zero divisors in $E(\mathfrak{S}f)$. Let $h_1 = g_1 f_1$ and $h_2 = g_2 f_2$, neither of which are equivalent to zero modulo $f$. Then

$$f h_1 = f(1 - g_2 f_2) \equiv 0 \bmod f_2 \quad \text{and} \quad f h_1 = f g_1 f_1 \equiv 0 \bmod f_1,$$

so $f h_1 \equiv 0 \bmod f$. Similarly $f h_2 \equiv 0 \bmod f$, so $h_1, h_2 \in I(\mathfrak{S}f)$. Moreover, $h_1 h_2 = h_1 - h_1^2 = h_2 - h_2^2$, which is equivalent to zero modulo both $f_1$ and $f_2$, and hence modulo $f$. Thus $(h_1 + \mathfrak{S}f)(h_2 + \mathfrak{S}f) \equiv 0 \bmod f$ and $h_1 + \mathfrak{S}f$ and $h_2 + \mathfrak{S}f$ are non-zero zero divisors in $E(\mathfrak{S}f)$.

If $f$ is reducible but indecomposable then Jacobson (1943), Theorem 3.13, shows $\hat{f} = \hat{g}^e \in \mathsf{K}[y]$ is the minimal central left multiple of $f$, where $\hat{g} \in \mathsf{K}[y]$ is irreducible as a polynomial in $y$ and $e \geq 1$. If $\hat{g} = y$ then $f = x^d$ for some $d \geq 2$, and $\mathfrak{S}f$ is a two-sided ideal in $\mathfrak{S}$. Thus $E(\mathfrak{S}/\mathfrak{S}f) = \mathfrak{S}/\mathfrak{S}f$ and $x + \mathfrak{S}f$ is a zero-divisor in $E(\mathfrak{S}/\mathfrak{S}f)$. Now assume that $\hat{g} \neq y$. The set $f + \mathfrak{S}\hat{f}$ generates a left ideal $L$ in $\mathcal{A} = \mathfrak{S}/\mathfrak{S}\hat{f}$. We now show that $e > 1$ by contradiction. Suppose that $e = 1$ so that $\mathcal{A}$ is simple. Then by Fact 8.7, there exist maximal left ideals $M_1, \ldots, M_m \subseteq \mathcal{A}$ such that $M_1 \cap \cdots \cap M_k = L$ and $M_i + (M_1 \cap \cdots \cap M_{i-1} \cap M_{i+1} \cap \cdots \cap M_m) = \mathcal{A}$. Since $f$ is reducible we know $L$ is not maximal so $k \geq 2$. Each maximal left ideal $M_i$ has an irreducible minimal modular generator $h_i \in \mathfrak{S}$, for $1 \leq i \leq k$. By Lemma 8.8, $f = \mathrm{lclm}(h_1, \ldots, h_k)$. Moreover, since $M_i + (M_1 \cap \cdots \cap M_{i-1} \cap M_{i+1} \cap \cdots \cap M_m) = \mathcal{A}$ for $1 \leq i \leq m$, we know $M_i + (M_1 \cap \cdots \cap M_{i-1} \cap M_{i+1} \cap \cdots \cap M_k) = \mathcal{A}$ for $1 \leq i \leq k$, and by Lemma 8.8 it follows that $\mathrm{gcrd}(h_i, \mathrm{lclm}(h_1, \ldots, h_{i-1}, h_{i+1}, \ldots, h_k)) = 1$ for $1 \le i \le k$. Thus, $h_1, \ldots, h_k$ are pairwise co-prime. In particular, since $k \ge 2$, $f$ is decomposable, which is a contradiction. Assume then that $e \ge 2$. Note that $\hat{g} \in I(\mathfrak{S}f)$ and $\hat{g} \not\equiv 0 \bmod f$, so the image $\hat{g} + \mathfrak{S}f \in E(\mathfrak{S}f)$ of $\hat{g}$ in $E(\mathfrak{S}f)$ is non-zero. Since $\hat{g}^e \equiv 0 \bmod f$, we see that $(\hat{g} + \mathfrak{S}f)(\hat{g}^{e-1} + \mathfrak{S}f) \equiv 0 \bmod \mathfrak{S}f$ and $E(\mathfrak{S}f)$ is not a field. □

Next we show that left zero divisors in $\mathcal{D} \cong E(\mathfrak{S}f)$ allow us to split $f$.

**Theorem 8.10.** *For $f \in \mathfrak{S}$, if $u, v \in \mathcal{D}\backslash\{0\}$ with $uv \equiv 0 \bmod f$, then* $\mathrm{gcrd}(f, u) \ne 1$.

**Proof.** Suppose $\mathrm{gcrd}(f, u) = 1$. Then there exist $s, t \in \mathfrak{S}$ such that $sf + tu = 1$ and hence $sfv + tuv = v$. But $fv \equiv 0 \bmod f$ and $uv \equiv 0 \bmod f$ so $v \equiv 0 \bmod f$, a contradiction. □

The problem of finding complete factorizations in $\mathsf{F}[x; \sigma]$ is reduced to the problem of finding zero divisors in finite algebras by the following algorithm.

**Algorithm:** `Complete-Factorization`

Input: $f \in \mathsf{F}[x; \sigma]$ of degree $n$;

Output: $f_1, \ldots, f_k \in \mathsf{F}[x; \sigma]$ irreducible, with $f = f_1 \cdots f_k$;

(1) Compute a basis for $\mathcal{D}$ (above) as a $\mathsf{K}$-algebra;

(2) If $\mathcal{D}$ is a field Then Return $f$;

Else

(3) Find a non-zero left zero divisor $u \in \mathcal{D}$;

(4) Compute $h = \mathrm{gcrd}(f, u)$ and $g \in \mathsf{F}[x; \sigma]$ with $f = gh$;

(5) Recursively factor $g = g_1 \cdots g_r$ and $h = h_1 \cdots h_s$
with $g_1, \ldots, g_r, h_1, \ldots, h_s \in \mathsf{F}[x; \sigma]$ irreducible;

(6) Return $g_1, \ldots, g_r, h_1, \ldots h_s$;

**End.**

The polynomial $f \in \mathsf{F}[x; \sigma]$ is irreducible if and only if $\mathcal{D}$ is a field, and the algorithm halts correctly in this case. If $f \in \mathfrak{S}$ is reducible then Theorem 8.9 implies $\mathcal{D}$ is not a field, and therefore possesses non-zero zero divisors (Wedderburn's Theorem implies every finite algebra, whose only zero divisor is zero, is a field). By Theorem 8.10

any left zero divisor has a non-zero GCRD with $f$, yielding a proper factorization in step 4. The algorithm recurses on $g$ and $h$, each of which has degree less than $n$. Since there is no recursion when $f$ is irreducible, the procedure `Complete-Factorization` will be called at most $n$ times, each time on a polynomial of degree at most $n$.

The number of operations in $\mathsf{K}$ required by each step is now determined:

**Step 1.** A basis for $\mathcal{D}$ can be found as follows. Let $W \subseteq \mathsf{F}[x;\sigma]$ be the set of all $g \in \mathsf{F}[x;\sigma]$ with $\deg g < n$. As a $\mathsf{K}$-vector space $W$ is isomorphic to $\mathsf{F}[x;\sigma]/\mathsf{F}[x;\sigma]f$, with basis

$$\{\Theta_{\mathsf{K}}^i x^j \mid 0 \le i < \mu,\, 0 \le j < n\},$$

and dimension $n\mu$. Multiplication on the left by $f$ induces an $\mathsf{K}$-linear map $T\colon W \to W$: if $u \in W$ then $T(u) = v \equiv fu \bmod f$, for some $v \in W$. The elements of $\mathcal{D}$ are exactly those elements in the null space of $T$, a basis which is found by constructing a matrix for $T$ (an $n\mu \times n\mu$ matrix over $\mathsf{K}$) and then using linear algebra techniques to compute a basis for the null space. This matrix is computed by evaluating $T$ at each of the basis elements of $W$, i.e., finding $f\Theta_{\mathsf{K}}^i x^j \bmod f$ for $0 \le i < \mu$ and $0 \le j < n$, requiring a total of $O(n^3\mu\mathrm{M}(\mu) + n^2\mu^2\mathrm{M}(\mu)\log\mu)$ operations in $\mathsf{K}$. The linear algebra to find a basis for the null space of $T$, and hence for $\mathcal{D}$, requires $O(\mathrm{MM}(n\mu))$ additional operations in $\mathsf{K}$.

**Steps 2–3.** We have not yet shown how to determine if $\mathcal{D}$ is a field, and if it is not, produce a non-zero zero divisor in $\mathcal{D}$. In (Rónyai, 1987) it is shown that this problem is reducible, with $(n\mu\log q)^{O(1)}$ operations in $\mathsf{K}$, to factoring polynomials in $\mathbb{F}_p[x]$ of degree $(n\delta)^{O(1)}$ (recall $[\mathsf{F} : \mathbb{F}_p] = \delta$). A faster Las Vegas type probabilistic algorithm for this same problem is presented in Chapter 7 , and requires $O(n\mu\chi + \mathrm{MM}(n\mu) + n\mu\log q)$ operations in $\mathsf{K}$, where $\chi$ operations in $\mathsf{K}$ are required to multiply two elements of $\mathcal{D}$. A multiplication in $\mathcal{D}$ can be done with $O(n^2\mathrm{M}(\mu) + n\mu\mathrm{M}(\mu)\log\mu)$ operations in $\mathsf{K}$, so we can determine if $\mathcal{D}$ is a field, and if not, find a zero divisor in $\mathcal{D}$, with $O(n^3\mu\mathrm{M}(\mu) + n^2\mu^2\mathrm{M}(\mu)\log\mu + \mathrm{MM}(n\mu) + n\mu\log q)$ or operations in $\mathsf{K}$.

**Step 4.** Polynomials $g$ and $h$ can be computed with $O(n^2\mathrm{M}(\mu)\mu\log\mu)$ operations in $\mathsf{K}$ by Lemma 8.4.

As noted above, there are at most $n$ recursive calls, each on a polynomial of degree less than $n$. This yields the following theorem:

**Theorem 8.11.** *Let $f \in \mathsf{F}[x;\sigma]$ have degree $n$. The algorithm* `Complete-Factorization` *correctly finds a complete factorization of $f$ in $\mathsf{F}[x;\sigma]$, and proves:*

*(i) the complete factorization problem is deterministically reducible, with $(n\mu\log q)^{O(1)}$ operations in $\mathsf{K}$, to the problem of factoring polynomials in $\mathbb{F}_p[x]$ of degree $(n\delta)^{O(1)}$, and is solvable by a deterministic algorithm requiring $(n\delta p)^{O(1)}$ operations in $\mathsf{K}$.*

*(ii) the complete factorization problem is solvable by a Las Vegas type algorithm with $O(n^4\mu\mathrm{M}(\mu)+n^3\mu^2\mathrm{M}(\mu)\log\mu+n\mathrm{MM}(n\mu)+n^2\mu\log q)$ or $O\tilde{}(n^4\mu^2+n^3\mu^3+n\mathrm{MM}(n\mu)+n^2\mu\log q)$ operations in $\mathsf{K}$.*

## 8.4 Bi-Factorization With Two-Sided Ideals

Finding the minimal central left multiple $\hat{f} \in \mathsf{F}[y]$ of an $f \in \mathsf{F}[x;\sigma]$ provides the key to bi-factorization. The following theorem demonstrates how the factorization over $\mathsf{K}[y]$ of $\hat{f}$ yields a partial factorization of $f$. Once again, we let $\mathfrak{S} = \mathsf{F}[x;\sigma]$ throughout this section.

**Theorem 8.12.** *Let $f \in \mathsf{F}[x;\sigma]$ and $\hat{f} \in \mathsf{K}[y]\setminus\{0\}$ be such that $\hat{f} \equiv 0 \bmod f$. If $\hat{f} = \hat{f}_1\cdots\hat{f}_l$ for pairwise co-prime $\hat{f}_1,\ldots,\hat{f}_l \in \mathsf{K}[y]$, then $f = \mathrm{lclm}(h_1,\ldots,h_l)$, where $h_i = \mathrm{gcrd}(\hat{f}_i, f)$ for $1 \le i \le l$, and $h_1,\ldots,h_l$ are pairwise co-prime.*

**Proof.** From the definitions of GCRD and LCLM in Section 8.2, this theorem can be restated in terms of ideals: $\mathfrak{S}f = \mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_l$ and

$$\mathfrak{L}_i + (\mathfrak{L}_1 \cap \cdots \mathfrak{L}_{i-1} \cap \mathfrak{L}_{i+1} \cap \cdots \cap \mathfrak{L}_l) = \mathfrak{S},$$

for $1 \le i \le l$, where $\mathfrak{L}_i = \mathfrak{S}f + \mathfrak{S}\hat{f}_i = \mathfrak{S}h_i$.

We start by showing that $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_l = \mathfrak{S}f$. For any $u \in \mathfrak{S}f$, we know $u \equiv 0 \bmod f$ and hence $u \equiv 0 \bmod h_i$ and $u \in \mathfrak{L}_i$ for $1 \le i \le l$. Thus $\mathfrak{S}f \subseteq \mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_l$.

To show $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_l \subseteq \mathfrak{S}f$ assume $u \in \mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_l$. Thus $u = v_i f + w_i \hat{f}_i$ for some $v_i, w_i \in \mathfrak{S}$, and $u \equiv v_i f \bmod \hat{f}_i$, for $1 \leq i \leq l$. We know that $\mathfrak{S}/\mathfrak{S}\hat{f}$ is isomorphic as a ring to $\mathfrak{S}/\mathfrak{S}\hat{f}_1 \oplus \cdots \oplus \mathfrak{S}/\mathfrak{S}\hat{f}_l$. By the Chinese remainder theorem, since $u$ is a left multiple of $f$ modulo each $\hat{f}_i$, $u$ is a left multiple of $f$ modulo $\hat{f}$, i.e., $u \equiv vf \bmod \hat{f}$ for some $v \in \mathfrak{S}$. From this and the fact that $\hat{f} \equiv 0 \bmod f$ we see $u \equiv 0 \bmod f$, and therefore that $u \in \mathfrak{S}f$ and $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_l = \mathfrak{S}f$.

To show that

$$\mathfrak{L}_i + (\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_{i-1} \cap \mathfrak{L}_{i+1} \cap \cdots \cap \mathfrak{L}_l) = \mathfrak{S},$$

for $1 \leq i \leq l$, we observe that $\mathfrak{S}\hat{f} = \mathfrak{S}\hat{f}_1 \cap \cdots \cap \mathfrak{S}\hat{f}_l$, where

$$\mathfrak{S}\hat{f}_i + (\mathfrak{S}\hat{f}_1 \cap \cdots \cap \mathfrak{S}\hat{f}_{i-1} \cap \mathfrak{S}\hat{f}_{i+1} \cap \cdots \cap \mathfrak{S}\hat{f}_l) = \mathfrak{S}.$$

This follows since $\mathsf{K}[y]$ is a unique factorization domain. Thus, for $1 \leq i \leq l$, there exists $u_i \in \mathfrak{S}\hat{f}_i$ and $v_i \in \mathfrak{S}\hat{f}_1 \cap \cdots \cap \mathfrak{S}\hat{f}_l$ such that $u_i + v_i = 1$. Since $\mathfrak{L}_i \supseteq \mathfrak{S}\hat{f}_i$ for $1 \leq i \leq l$, we know $u_i \in \mathfrak{L}_i$ and $v_i \in \mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_{i-1} \cap \mathfrak{L}_{i+1} \cap \cdots \cap \mathfrak{L}_l$, so

$$1 \in \mathfrak{L}_i + (\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_{i-1} \cap \mathfrak{L}_{i+1} \cap \cdots \cap \mathfrak{L}_l) = \mathfrak{S}$$

for $1 \leq i \leq l$. $\square$

The above theorem is used to get a partial decomposition of $f$ by factoring its minimal central left multiple $\hat{f} \in \mathsf{K}[y]$, as a polynomial in $y$, into pairwise co-prime polynomials in $\mathsf{K}[y]$ and then taking GCRD's between $f$ and each of these factors. We now address the question of finding $\hat{f}$. First, compute the sequence $x^{i\mu} = Q_i f + R_i$ for $0 \leq i \leq n\mu$, where $Q_i, R_i \in \mathsf{F}[x;\sigma]$ and $\deg R_i < \deg f = n$. The set of all polynomials in $\mathsf{F}[x;\sigma]$ of degree less than $n$ forms a $\mathsf{K}$-vector space of dimension $n\mu$, where each coefficient in $\mathsf{F}$ is expanded with respect to the given basis of $\mathsf{F}/\mathsf{K}$. For $1 \leq i \leq n\mu$, where

$$R_i = \sum_{0 \leq j < n} \sum_{0 \leq l < \mu} \gamma_{jl} \Theta_{\mathsf{K}}^l x^j, \quad \text{let } \bar{R}_i = (\gamma_{0,0}, \gamma_{0,1}, \ldots, \gamma_{n-1,\mu-1})^t \in \mathsf{K}^{n\mu \times 1}.$$

Since there are $n\mu + 1$ polynomials $R_0, \ldots R_{n\mu}$, there exists a minimal $t \le n\mu$ and $\alpha_0, \ldots, \alpha_t \in \mathsf{K}$, not all zero, such that $\sum_{0\le i\le t} \alpha_i R_i = 0$ and hence that $\sum_{0\le i\le t} \alpha_i \bar{R}_i = 0$. The minimal central left multiple $\hat{f}$ of $f$ is then $\hat{f} = \alpha_t^{-1} \sum_{0\le i\le t} \alpha_i x^{\mu i}$.

Let $B$ be the $n\mu \times (n\mu + 1)$ matrix over $\mathsf{K}$ whose $i$th column is $\bar{R}_{i-1}$. Since $R_t$ is linearly dependent (over $\mathsf{K}$) on $R_0, \ldots, R_{t-1}$, and $R_{t+i} \equiv x^\mu R_{t+i-1} \bmod f$, it follows that $R_{t+i}$ is also linearly dependent (over $\mathsf{K}$) on $R_0, \ldots, R_{t-1}$, for $i \ge 0$. Thus $t = \operatorname{rank} B$, and if $v = (\alpha_0, \ldots, \alpha_t, 0, \ldots, 0) \in \mathsf{K}^{n\mu}$, then $Bv = 0$. Conversely, any non-zero $v \in \mathsf{K}^{n\mu}$ of the form $v = (\beta_0, \ldots, \beta_t, 0, \ldots, 0) \in \mathsf{K}^{n\mu}$, and in the null space of $B$, yields a scalar multiple $\sum_{0\le i\le t} \beta_i x^i$ of the minimal central left multiple of $\hat{f}$. Hence we can now solve for the minimal central left multiple of $\hat{f}$ with linear algebra over $\mathsf{K}$.

To determine the cost of this algorithm, start by computing $X_i \equiv x^i \bmod f$ with $\deg X_i < \deg f$, for $0 \le i \le n + \mu - 1$; this can be accomplished with $O(n^2\mu\mathrm{M}(\mu) + n\mu^2\mathrm{M}(\mu)\log\mu)$ operations in $\mathsf{K}$. Now for any

$$g = \sum_{0\le i<n} b_i x^i$$

with $b_0, \ldots, b_{n-1} \in \mathsf{F}$, we know

$$x^\mu g = \sum_{0\le i<n} b_i x^{i+\mu} \equiv \sum_{0\le i<n} b_i X_{i+\mu} \bmod f.$$

Using the fact that $R_i \equiv x^\mu R_{i-1} \bmod f$ for $i > 0$, we can compute $R_i$ from $R_{i-1}$ as an $\mathsf{F}$-linear combination of $X_\mu, \ldots, X_{\mu+n-1}$, with $O(n^2\mu)$ operations in $\mathsf{F}$. Finding $R_0, \ldots, R_{n\mu}$ then takes $O(n^3\mu\mathrm{M}(\mu))$ operations in $\mathsf{K}$, and the linear algebra to compute $\hat{f}$ from $\bar{R}_0, \ldots, \bar{R}_{n\mu}$ requires an additional $O(\mathrm{MM}(n\mu))$ operations in $\mathsf{K}$.

**Lemma 8.13.** *Given $f \in \mathsf{F}[x;\sigma]$ of degree $n$, we can find the minimal central left multiple of $f$ with $O(n^3\mu\mathrm{M}(\mu) + n^2\mu^2\mathrm{M}(\mu)\log\mu + \mathrm{MM}(n\mu))$ or $O\tilde{~}(n^3\mu^2 + n^2\mu^3 + \mathrm{MM}(n\mu))$ operations in $\mathsf{K}$.*

The next lemma characterizes the LCLM-decompositions of those $f \in \mathsf{F}[x;\sigma]$ whose minimal central left multiples are irreducible as polynomials in $y$.

**Theorem 8.14.** *Let $f \in \mathsf{F}[x;\sigma]$ and $\hat{f} \in \mathsf{K}[y]$ the minimal central left multiple of $f$ with $\deg \hat{f} = n\mu$. If $\hat{f}$ is irreducible as a polynomial in $y$ and $f = gh$ for some irreducible $h \in \mathsf{F}[x;\sigma]$, then $\deg h = n$.*

**Proof.** If $\hat{f} = y = x^\mu$ then $n = 1$. The only right factor of $\hat{f}$ in this case is $x$, which has degree 1.

Assume then that $\hat{f} \neq y$. The quotient $A = \mathfrak{S}/\mathfrak{S}\hat{f}$ is a simple algebra (since $\mathfrak{S}\hat{f}$ is a maximal left ideal in $\mathfrak{S}$) of dimension $n\mu^2$ over $\mathsf{K}$. By Fact 8.7, for some $m \geq 1$, $A$ is isomorphic to the ring of all $m \times m$ matrices over the centre $\mathsf{E}$ of $A$, where $\mathsf{E}$ is an extension field of $\mathsf{K}$. If $[\mathsf{E} : \mathsf{K}] = r$, then $n\mu^2 = rm^2$.

The centre $\mathsf{E}$ of $A$ is simply the image of $\mathsf{K}[y]$ in $A$. To see this, let $g \in \mathfrak{S}$ and $\bar{g}$ its image in $A$. If $g \in \mathsf{K}[y]$ then $\bar{g}$ is certainly in $\mathsf{E}$. Conversely, if $\bar{g} \in \mathsf{E}$, then we may assume $\deg g < \deg_x \hat{f}$, i.e., we choose the polynomial of least degree in $\mathfrak{S}$ which is equivalent to $\bar{g}$ modulo $\hat{f}$. Now $g\Theta_{\mathsf{K}} - \Theta_{\mathsf{K}} g \equiv 0 \bmod \hat{f}$ and $gx - xg \equiv 0 \bmod \hat{f}$, since $g$ is in the centre of $A$. The degrees of $g\Theta_{\mathsf{K}}$ and $\Theta_{\mathsf{K}} g$ are both less than $\deg_x \hat{f}$, so $g\Theta_{\mathsf{K}} - \Theta_{\mathsf{K}} g = 0$, which is only true if $g \in \mathsf{F}[x^\mu]$. Assume now that $\mu \geq 2$ (if $\mu = 1$ then $\mathfrak{S} = \mathsf{F}[x]$ and the theorem is trivially true). Since $g \in \mathsf{F}[x^\mu]$ it has degree less than $n\mu - 1$ and both $gx$ and $xg$ have degrees less than $n\mu$, whence $gx - xg = 0$. The elements $x$ and $\Theta_{\mathsf{K}}$ generate $\mathfrak{S}$ as a $\mathsf{K}$-algebra, and since $g$ commutes with both of them, $g$ must be in the centre of $\mathfrak{S}$. Therefore the centre $\mathsf{E}$ of $A$ is the image of $\mathsf{K}[y]$ in $A$, and has degree $n$ over $\mathsf{K}$. It follows that $r = [\mathsf{E} : \mathsf{K}] = n$, $m = \mu$, and $A \cong \mathsf{E}^{\mu \times \mu}$.

Maximal left ideals in $A$ are exactly those whose minimal modular generators are irreducible in $\mathfrak{S}$. In particular, the left ideal generated by $h + \mathfrak{S}$ is maximal. By Fact 8.7, each maximal left ideal in $A$ has dimension $n\mu^2 - n\mu$ as a $\mathsf{K}$-vector space. Since the left ideal in $A$ generated by $h + \mathfrak{S}$ is equal to the set of left multiples of $h$ of degree less than $n\mu$, reduced modulo $\hat{f}$, it has dimension $n\mu^2 - n \deg h$ as a $\mathsf{K}$-vector space. Thus $\deg h = n$. $\square$

A distinct degree factorization (in $\mathsf{K}[y]$), of the minimal central left multiple $\hat{f}$ of $f \in \mathsf{F}[x;\sigma]$, yields the degrees of all factors in any complete factorization of $f$ as shown in the next theorem and its corollary.

**Theorem 8.15.** *Let $f \in \mathsf{F}[x;\sigma]$ and $\hat{f} \in \mathsf{K}[y] \setminus \{0\}$ be such that $\hat{f} \equiv 0 \bmod f$. Furthermore, suppose $\hat{f} = \hat{g}^e$ for some $\hat{g} \in \mathsf{K}[y] \setminus \{0\}$ and $e \geq 1$, where $\hat{g}$ is irreducible as a polynomial in $\mathsf{K}[y]$, and $\deg_x \hat{g} = d\mu$. Then for all complete factorizations $f = f_1 \cdots f_l$, with $f_1, \ldots, f_l \in \mathsf{F}[x;\sigma]$ irreducible in $\mathsf{F}[x;\sigma]$, we have $\deg f_i = d$.*

**Proof.** Suppose $f = f_1 \cdots f_k$, where $f_1, \ldots, f_k \in \mathsf{F}[x;\sigma]$ are irreducible. We proceed by induction on $k$. If $k = 1$, then Jacobson (1943), Chapter 12, Theorem 13 shows $e = 1$, and by Theorem 8.14, $\deg f_1 = d$. Assume that the theorem is true for complete factorizations with fewer than $k$ irreducible factors. The minimal central left multiple of $f_k$ must be irreducible as a polynomial in $y$ and must divide $\hat{g}^e$, whence $\hat{g} \equiv 0 \bmod f_k$. By Theorem 8.14, $\deg f_k = d$. Moreover, by (Jacobson 1943, Chapter 12, Theorem 12), $\hat{g}^e \equiv 0 \bmod f_1 \cdots f_{k-1}$, so by induction then $\deg f_1 = \cdots = \deg f_{k-1} = d$. □

**Corollary 8.16.** *Let $f \in \mathsf{F}[x;\sigma]$ and $\hat{f} \in \mathsf{K}[y] \setminus \{0\}$ be such that $\hat{f} \equiv 0 \bmod f$. Furthermore, suppose $\hat{f} = \hat{g}_1^{e_1} \hat{g}_2^{e_2} \cdots \hat{g}_l^{e_l}$ where $e_1, \ldots, e_l \geq 1$ and $\hat{g}_1, \ldots, \hat{g}_l \in \mathsf{K}[y]$ are distinct and irreducible as polynomials in $\mathsf{K}[y]$, all with the same degree $d\mu$ in $x$. Then for any complete factorization $f = f_1 \cdots f_k$, with $f_1, \ldots, f_k \in \mathsf{F}[x;\sigma]$ irreducible, we have $\deg f_i = d$.*

**Proof.** By Theorem 8.12 we know $f = \mathrm{lclm}(h_1, \ldots, h_l)$, where $h_i = \mathrm{gcrd}(\hat{f}_i^{e_i}, f)$ for $1 \leq i \leq l$. Since $\hat{f}_i^{e_i} \equiv 0 \bmod h_i$ for $1 \leq i \leq l$, we know by Theorem 8.14 that every complete factorization

$$h_i = h_{i,1} h_{i,2} \cdots h_{i,s_i},$$

where each $h_{i,j} \in \mathfrak{S}$ is irreducible, is such that $\deg h_{i,j} = d$ for $1 \leq j \leq s_i$ and $1 \leq i \leq l$. Theorem 8.6 implies that if

$$f = f_1 f_2 \cdots f_k,$$

where $f_1, \ldots, f_k \in \mathfrak{S}$ are irreducible, then $\deg f_i = d$ for $1 \leq i \leq k$. □

Corollary 8.16 yields an efficient reduction from the bi-factorization problem to the complete factorization problem.

**Algorithm:** `Bi-Factorization`

Input: $f \in \mathsf{F}[x;\sigma]$ and $s \leq \deg f = n$;

Output: $g, h \in \mathsf{F}[x;\sigma]$ with $\deg h = s$, and $f = gh$, or a message that no such $h$ exists;

(1) Compute the minimal central left multiple $\hat{f} \in \mathsf{K}[y]$ of $f$;

(2) Find a distinct degree factorization of $\hat{f}$ as $\hat{f} = \hat{f}_1 \hat{f}_2 \cdots \hat{f}_n$, where $\hat{f}_i \in \mathsf{K}[y]$ is such that if $\hat{g} \in \mathsf{K}[y]$ divides $\hat{f}_i$, and $\hat{g}$ is irreducible as a polynomial in $\mathsf{K}[y]$, then $\deg_y \hat{g} = i$, for $1 \leq i \leq n$ (some $f_i$'s may have degree zero);

(3) Find $h_i = \mathrm{gcrd}(\hat{f}_i, f) \in \mathsf{F}[x;\sigma]$ for $1 \leq i \leq n$; Assume $\deg h_i = ie_i$ for some $e_i \in \mathbb{N}$;

(4) Factor each $h_i$ completely in $\mathsf{F}[x;\sigma]$ as $h_i = h_{i,1} h_{i,2} \cdots h_{i,e_i}$, where $\deg h_{i,j} = i$ for $1 \leq j \leq e_i$, and $1 \leq i \leq n$;

(5) Determine if there exists a set $d_1, \ldots, d_n \in \mathbb{N}$ with $d_i \leq e_i$ for $1 \leq i \leq n$, such that $\sum_{0 \leq i \leq n} i d_i = s$;

If such $d_1, \ldots, d_n$ exist then return $g, h \in \mathsf{F}[x;\sigma]$, where $h = \mathrm{lclm}(\bar{h}_1, \bar{h}_2, \ldots, \bar{h}_n)$, and $\bar{h}_i = h_{i,e_i - d_i + 1} h_{i,e_i - d_i + 2} \cdots h_{i,e_i}$ for $1 \leq i \leq n$, and $g \in \mathsf{F}[x;\sigma]$ is such that $f = gh$;

Otherwise, return "$f$ has no right factor of degree $s$ in $\mathsf{F}[x;\sigma]$";

**End.**

Any pair $g, h \in \mathsf{F}[x;\sigma]$ produced by the algorithm has $\deg h = s$ and $f = gh$, and if such a bi-factorization exists, this algorithm produces one. To see the former, we note that by Theorem 8.12, $f = \mathrm{lclm}(h_1, \ldots, h_n)$, where $h_i = \mathrm{gcrd}(\hat{f}_i, f)$ as computed in step 3. By Corollary 8.16, all complete factorizations of $h_i = h_{i,1} h_{i,2} \cdots h_{i,e_i}$ into irreducible $h_{i,j} \in \mathsf{F}[x;\sigma]$, are such that $\deg h_{i,j} = i$ for $1 \leq j \leq e_i$ and $1 \leq i \leq n$. If $h = \mathrm{lclm}(\bar{h}_1, \ldots, \bar{h}_n)$, then Theorem 8.6 implies $\deg h = s$. The computed $h$ is a right factor of $f$ since each $\bar{h}_i$ is a right factor of $h_i$ and each $h_i$ is a right factor of $f$ for $1 \leq i \leq n$.

If $f = uv$ for some $u, v \in \mathsf{F}[x;\sigma]$ and $\deg v = s$, this algorithm finds some right factor $h$ of $f$ of degree $s$. Suppose $v = v_1 v_2 \cdots v_t$, with $v_1, \ldots, v_t \in \mathsf{F}[x;\sigma]$ irreducible. If exactly $d_i$ of the factors $v_1, \ldots, v_t$ have degree $i$ for $1 \leq i \leq n$, then $d_i \leq e_i$ by

Theorem 8.6. Hence $h = \operatorname{lclm}(\bar{h}_1, \ldots, \bar{h}_n)$, computed in step 5, has degree $s$.

The the number of operations required by the algorithm `Bi-factorization` is now determined.

**Step 1.** Computing the minimal central left multiple of $f$ requires $O(n^3\mu\mathrm{M}(\mu) + \mathrm{MM}(n\mu))$ operations in $\mathsf{K}$ by Lemma 8.13.

**Step 2.** Distinct degree factorization can be computed with $O(\mathrm{M}(n\mu) \cdot \mathrm{M}(\sqrt{n\mu}) \cdot \sqrt{n\mu} \log n + \mathrm{M}(n\mu) \log q)$ operations in $\mathsf{K}$, using the algorithm of von zur Gathen & Shoup (1992).

**Step 3.** Use the fact that if $u \equiv v \bmod f$ then $\operatorname{gcrd}(f, u) = \operatorname{gcrd}(f, v)$, for any $u, v \in \mathsf{F}[x; \sigma]$. During the computation of the minimal central left multiple $\hat{f}$ in step 1 we found $R_j \equiv x^{\mu j} \bmod f$, where $\deg R_j < \deg f$ for $1 \leq j \leq n\mu$. Each polynomial $\hat{f}_i \bmod f$ is then just a linear combination of the $R_j$'s, so we can compute $\hat{f}_1 \bmod f, \hat{f}_2 \bmod f, \ldots, \hat{f}_n \bmod f$ within the time required for step 1. The required GCRD's can now be computed with $O(n^3\mathrm{M}(\mu)\mu \log \mu)$ operations in $\mathsf{K}$.

**Step 4.** Completely factoring $n$ polynomials in $\mathsf{F}[x; \sigma]$ of total degree $n$ is accomplished by the algorithm `Complete-Factorization` of Section 8.3. The cost of completely factoring $n$ polynomials in $\mathsf{F}[x; \sigma]$ of total degree $n$ is at most the cost of completely factoring a single polynomial in $\mathsf{F}[x; \sigma]$ of degree $n$. By Theorem 8.11 this can be done with a (Las Vegas) probabilistic algorithm requiring $O(n^4\mu\mathrm{M}(\mu) + n^3\mu^2\mathrm{M}(\mu) \log \mu + n\,\mathrm{MM}(n\mu) + n^2\mu \log q)$ operations in $\mathsf{K}$. By the same theorem, this problem is deterministically reducible, with $(n\mu \log p)^{O(1)}$ operations in $\mathsf{K}$, to the problem of factoring univariate polynomials in $\mathbb{F}_p[x]$ of degree $(n\mu)^{O(1)}$.

**Step 5.** Determining if $d_1, \ldots, d_n$ exist, and finding them if they do, while not performed with $\mathsf{K}$-operations (and hence not really "counted" in our model of computation), can be accomplished efficiently with a simple dynamic programming algorithm. The LCLM can be performed with $O(\mathrm{MM}(n) \cdot \mathrm{M}(\mu) + n^2\mu\mathrm{M}(\mu) \log \mu)$ operations in $\mathsf{K}$.

**Theorem 8.17.** *Let $f \in \mathsf{F}[x;\sigma]$ have degree $n$ and $s < n$. The above algorithm* `bi-factorization` *correctly solves the problem of determining if there exist $g, h \in \mathsf{F}[x;\sigma]$ with $f = gh$, and $\deg h = s$, and if so, find such $g, h$, and proves:*

*(i) the bi-factorization problem is deterministically reducible, with $(n\mu \log q)^{O(1)}$ operations in $\mathsf{K}$, to the problem of factoring polynomials in $\mathbb{F}_p[x]$ of degree $(n\delta)^{O(1)}$, and is solvable with a deterministic algorithm requiring $(n\delta p)^{O(1)}$ operations in $\mathsf{K}$;*

*(ii) the bi-factorization problem is solvable by a probabilistic algorithm with $O(n^4\mu \mathrm{M}(\mu) + n^3\mu^2\mathrm{M}(\mu)\log\mu + n\mathrm{MM}(n\mu) + n^2\mu\log q)$ operations in $\mathsf{K}$.*

## 8.5 Applications of Skew Polynomial Rings

The problem of functionally decomposing polynomials has received considerable attention recently, and there exist a number of classes of polynomials for which no polynomial-time solution has been found. We consider such a class—the linearized or additive polynomials—and show that it is isomorphic (in a computationally trivial way) to a skew polynomial ring. This allows us to employ our algorithms for complete and bi-factorization in skew polynomial rings to obtain very fast algorithms for the functional decomposition of linearized polynomials.

The linearized polynomials over $\mathsf{F}$, in an indeterminate $\lambda$, are those of the form $\sum_{0\le i\le n} a_i\lambda^{p^i}$ (where $a_0, \ldots, a_n \in \mathsf{F}$). The set $\mathbb{A}_\mathsf{K}$ of all linearized polynomials in $\mathsf{F}[\lambda]$ forms a ring under the usual polynomial addition $(+)$, and functional composition $(\circ)$; if $f, g \in \mathbb{A}_\mathsf{K}$ with

$$f = \sum_{0\le i\le n} a_i\lambda^{p^i}, \quad \text{and } g = \sum_{0\le j\le r} b_j\lambda^{p^j}, \quad \text{then } f\circ g = f(g(\lambda)) = \sum_{0\le i\le n}\sum_{0\le j\le r} a_i b_j^{p^i}\lambda^{p^{i+j}}.$$

Now consider the skew-polynomial ring $\mathsf{F}[x;\psi]$, where $\psi(a) = a^p$ for any $a \in \mathsf{F}$ and $xa = a^p x$ for all $a \in \mathsf{F}$. This skew-polynomial ring is isomorphic to the ring $\mathbb{A}_\mathsf{K}$ under the map $\Phi\colon \mathbb{A}_\mathsf{K} \to \mathsf{F}[x;\psi]$, which acts as the identity on $\mathsf{F}$ and sends $\lambda^{p^i}$ to $x^i$ for $i \ge 0$ (see McDonald 1974, Theorem 2.13). Note that if $f \in \mathbb{A}_\mathsf{K}$, then $\deg\Phi(f) = \log_p(\deg f)$, so this isomorphism removes some of the "sparseness" of linearized polynomials. Computationally, $\Phi$ just maps between two interpretations of the input, and is free of charge.

The functional decomposition problem for general polynomials in $\mathsf{F}[\lambda]$ comes in two flavours analogous to our complete factorization and bi-factorization problems for $\mathsf{F}[x;\sigma]$. Given a polynomial $f \in \mathsf{F}[\lambda]$ of degree $N$, the (functional) complete decomposition problem asks for functionally indecomposable $f_1, \ldots, f_k \in \mathsf{F}[\lambda]$ such that $f = f_1 \circ \cdots \circ f_k$ (any $h \in \mathsf{F}[\lambda] \setminus \mathsf{F}$ is functionally indecomposable if all its bi-decompositions contain a linear composition factor). When $p \nmid N$, the so-called "tame" case for complete decomposition, fast deterministic algorithms for complete decomposition are presented in (von zur Gathen *et al.*, 1987). When $p \mid N$, the "wild" case, an algorithm of Zippel (1991) apparently solves the complete decomposition problem in time $(\deg f)^{O(1)}$, although the exact running time is not calculated. All polynomials $f \in \mathbb{A}_{\mathsf{K}}$ have degree $p^n$ for some $n \in \mathbb{N}$, so the complete decomposition problem for linearized polynomials is certainly in the wild case. Given $f \in \mathsf{F}[\lambda]$ and $s \in \mathbb{N}$, the (functional) bi-decomposition problem asks if there exist $g, h \in \mathsf{F}[\lambda]$ such that $f = g \circ h$ and $\deg h = s$, and if so, find such $g$, $h$. The tame case, when $p \nmid (N/s)$, is solved efficiently by von zur Gathen *et al.* (1987). When $p \mid (N/s)$, the wild case, no algorithm is known to solve this problem in time $(\deg f)^{O(1)}$, though a partial solution is provided by von zur Gathen (1990b). All non-trivial bi-decompositions of linearized polynomials fall into the wild case, since, if $f \in \mathbb{A}_{\mathsf{K}}$ and $f = g \circ h$ for $g, h \in \mathsf{F}[\lambda]$, then Dorey & Whaples (1974) show that $\deg g = p^r$ for some $r \in \mathbb{N}$.

When $f \in \mathbb{A}_{\mathsf{K}}$, we can solve both the bi-decomposition and complete decomposition problems using our algorithms for complete factorization and bi-factorization in $\mathsf{F}[x;\psi]$. The key observation is that we need only consider decompositions of $f \in \mathbb{A}_{\mathsf{K}}$ into linearized polynomials: Dorey & Whaples (1974) show that if $f = \bar{f}_1 \circ \cdots \circ \bar{f}_k$ for any $\bar{f}_1, \ldots, \bar{f}_k \in \mathsf{F}[\lambda]$, then there exist $f_1, \ldots, f_k \in \mathbb{A}_{\mathsf{K}}$ such that $f = f_1 \circ \cdots \circ f_k$ and $\deg f_i = \deg \bar{f}_i$ for $1 \leq i \leq k$. A complete decomposition of any $f \in \mathbb{A}_{\mathsf{K}}$ of degree $p^n$ can be found by finding a complete factorization of $\Phi(f)$ in $\mathsf{F}[x;\psi]$. Similarly, the bi-decomposition problem on input $f \in \mathbb{A}_{\mathsf{K}}$ of degree $p^n$ and $s \in \mathbb{N}$, is equivalent to the bi-factorization problem in $\mathsf{F}[x;\psi]$ on inputs $\Phi(f) \in \mathsf{F}[x;\psi]$ and $\log_p s$.

**Theorem 8.18.** *Let* $f \in \mathbb{A}_{\mathsf{K}}$ *have degree* $N = p^n$ *and* $S = p^s < N$, *where* $\mathsf{F} = \mathbb{F}_{p^\delta}$ *for some prime* $p \in \mathbb{N}$ *and* $\delta \geq 1$. *We can produce a complete decomposition of* $f$ *in* $\mathbb{A}_{\mathsf{K}}$, *and determine if there exist* $g, h \in \mathbb{A}_{\mathsf{K}}$ *such that* $\deg h = S$ *and* $f = g \circ h$, *and if so, find such* $g, h$, *with a deterministic algorithm requiring* $(n\delta p)^{O(1)}$ *operations in* $\mathsf{F}$ *or a probabilistic algorithm requiring* $O(n^4\delta\,\mathrm{M}(\delta) + n^3\delta^2\,\mathrm{M}(\delta)\log\delta + n\,\mathrm{MM}(n\delta) + n^2\delta\log q)$ *operations in* $\mathsf{F}$.

# Chapter 9.

# Conclusion and Open Questions

In this thesis we consider a variety of problems related to canonical forms of matrices and their applications. Perhaps most importantly we show that the problem of finding the Frobenius form of a matrix has about the same asymptotic complexity as matrix multiplication. Of course, our algorithm is probabilistic of the Las Vegas type, and requires a poly-logarithmic number of matrix multiplications, leaving some room for improvement.

**Open Question 1.** *Does there exist a deterministic algorithm which finds the Frobenius form of a matrix $T \in \mathsf{K}^{n\times n}$, over a field $\mathsf{K}$, with $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$?*

We present our algorithm for computing the Frobenius form of a matrix on an arithmetic RAM and count all field operations. An alternative model of computation —the *black box* model—poses exciting new questions about algorithms for many problems in linear algebra. In the black box model (as it applies to linear algebra problems) we are given a black box or subroutine for multiplying some fixed matrix by any vector. The object is to solve a problem using as few applications of this black box as possible, while also minimizing the number of additional field operations and memory locations required as well. For example, Wiedemann (1986) shows (quite strikingly) that given a matrix $A \in \mathsf{K}^{n\times n}$ and a vector $b \in \mathsf{K}^{n\times 1}$ over any field $\mathsf{K}$, we can determine if there exists an $x \in \mathsf{K}^{n\times 1}$ such that $Ax = b$ and if so find one, with $O(n)$ matrix-vector multiplications by $A$ as well as $O(n^2)$ additional operations in $\mathsf{K}$ and $O(n)$ space (see also Kaltofen 1992). This black box model nicely generalizes the usual notion of sparseness, where the cost incurred by a matrix-vector multiplication is presumably low. It also recognizes the benefits of special types of matrices, such as Toeplitz matrices, where the cost of matrix-vector products is also low. The problem of computing the Frobenius form can be cast in this model, as can that of finding the characteristic polynomial of a matrix, since both the Frobenius form and characteristic polynomial of a matrix in $\mathsf{K}^{n\times n}$ can be represented with $O(n)$ elements in $\mathsf{K}$.

**Open Question 2.** *Can we compute the Frobenius form of a matrix $T \in \mathsf{K}^{n\times n}$, over a field $\mathsf{K}$, with $O(n)$ matrix-vector products by $T$, using $O(n)$ additional space and $O(n^2)$ additional operations in $\mathsf{K}$? Can we compute the characteristic polynomial of $T$ with $O(n)$ matrix-vector products by $T$, using $O(n)$ additional space and $O(n^2)$ additional operations in $\mathsf{K}$?*

In some applications we are required to compute the Frobenius form of an integer matrix $T \in \mathbb{Z}^{n\times n}$, where $\|T\|_\infty = \alpha$. We present a Las Vegas algorithm for this problem which requires an expected number or $O\tilde{~}(n^5 \log^2 \alpha)$ bit operations using standard arithmetic to accomplish this. This represents a significant improvement on the previous state of the art. The existence of a Monte Carlo algorithm which requires only an expected number of $O\tilde{~}(n^4 \log \alpha)$ bit operations suggests we can do better still.

**Open Question 3.** *Does there exist a Las Vegas algorithm which, given a matrix $T \in \mathbb{Z}^{n\times n}$ with $\|T\|_\infty = \alpha$, finds its Frobenius form with an expected number of $O\tilde{~}(n^4 \log \alpha)$ bit operations?*

A better upper bound on the number bad primes, or a fast way of rejecting theses bad primes in the modular computation is required. We expect that we are very close to a solution to this question, and have solved the problem for the case when the minimal polynomial is squarefree. It is interesting to note that we *have* exhibited a Las Vegas algorithm to determine the minimal polynomial of a $T \in \mathbb{Z}^{n\times n}$ with $\|T\|_\infty = \alpha$ which requires only an expected number of $O\tilde{~}(n^4 \log \alpha)$ bit operations.

We also demonstrate some applications of the Frobenius form. Principal among these is the rather striking result that a polynomial of degree up to $n^2$ in $\mathsf{K}[x]$ can be evaluated at an $n \times n$ matrix with only $O\tilde{~}(\mathrm{MM}(n))$ operations in the field $\mathsf{K}$, by a Las Vegas algorithm. This nearly matches a lower bound of $\Omega(\mathrm{MM}(n))$ operations in $\mathsf{K}$ which we also demonstrate.

**Open Question 4.** *Given a matrix $T \in \mathsf{K}^{n\times n}$ and a polynomial $g \in \mathsf{K}[x]$ of degree at most $n^2$, can we compute $g(T) \in \mathsf{K}^{n\times n}$ with $O(\mathrm{MM}(n))$ operations in $\mathsf{K}$ by a deterministic algorithm?*

Finding the Frobenius form of a matrix also yields the minimal polynomial of that matrix. We also show that if there is a Las Vegas or deterministic algorithm for computing the minimal polynomial of a matrix in $\mathsf{K}^{n\times n}$ which requires $t(n)$ operations in the field $\mathsf{K}$, then there is a Las Vegas type algorithm to compute the determinant of a matrix with $O(t(n))$ operations in $\mathsf{K}$. Since computing the determinant is known to require $\Omega(\mathrm{MM}(n))$ operations in $\mathsf{K}$, our algorithm for the minimal polynomial is nearly optimal. The use of randomness in this reduction is not completely satisfying.

We demonstrate that finding the Jordan form is essentially no harder than the two necessary sub-problems of finding the Frobenius form and of factoring the minimal polynomial. This problem presents no additional obstacles, neither over an arbitrary field nor over $\mathbb{Z}$.

Normal bases are an efficient representation for finite fields, especially for problems where high powers of elements must be computed. We give very fast algorithms for finding such bases, both asymptotically and practically. We show that the more difficult problem of finding primitive-normal elements in a finite field is really no harder than certifying primitive elements. Many questions remain about algorithms for working with normal bases, as fundamental as how to multiply and invert elements efficiently.

We also apply our algorithms for fast evaluation of polynomials at matrices, and for fast determination of the minimal polynomial of a matrix, to the problem of finding zero divisors in any associative algebra over a finite field. We give a very short and practical algorithm for accomplishing this, which relies on a rather difficult density estimate of the number of elements in an algebra with a reducible minimal polynomial. The solution to this problem represents a first step in what we know is a rich set of applications of our algorithms for computing the Frobenius form and for evaluating a polynomial at a matrix in the computational investigation of associative algebras and modular group representations.

**Open Question 5.** *Let $\mathcal{A}$ be an associative algebra over a finite field $\mathsf{K}$, with basis $T_1, \ldots, T_m \in \mathsf{K}^{n\times n}$. How fast can we find the radical of $\mathcal{A}$? How fast can we determine the Wedderburn-Malcev decomposition of $\mathcal{A} = \mathcal{B} + \mathrm{Rad}(\mathcal{A})$, where $\mathcal{B} \cong \mathcal{A}/\,\mathrm{Rad}(A)$? How fast can we decompose $\mathcal{B}$ as a direct sum of simple algebras, and compute an* *isomorphism between each of these simple components and a full matrix algebra over an extension of* $\mathsf{K}$*?*

All of these questions, except that of computing the Wedderburn-Malcev decomposition, are solved in polynomial time by Rónyai (1990), and his algorithms make extensive use of subroutines to evaluate polynomials at matrices and to compute minimal polynomials.

Finally, we examine the problem of factoring in skew-polynomial rings over finite fields. Our algorithm for factoring is a reduction to the problem of finding zero divisors in finite dimensional associative algebras over finite fields. We show that factoring in this non-commutative, non-unique factorization domain is not much more difficult than factoring in the usual polynomial ring, at least with respect to the time required asymptotically. As an application, we show how to efficiently functionally decompose additive polynomials, a class of polynomials which had previously defied polynomial-time functional decomposition algorithms.

# Bibliography


L. Adleman. Two theorems on random polynomial time. In *Proc. 19th IEEE Symp. Foundations of Computer Science*, pp. 75–83, Los Angeles CA, 1978.

G. B. Agnew, R. C. Mullin, and S. A. Vanstone. Fast exponentiation in $GF(2^n)$. In *Advances in Cryptology—EUROCRYPT '88*, ed. C. G. Günther, vol. 330 of *Lecture Notes in Computer Science*, pp. 251–255. Springer (Berlin), 1988.

A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley (Reading MA), 1974.

A. A. Albert. *Fundamental Concepts of Higher Algebra*. University of Chicago Press (Chicago, Illinois), 1956.

T. M. Apostol. *Introduction to Analytic Number Theory*. Springer-Verlag (New York), 1976.

E. Artin. *The gamma function*. Holt, Rinehart and Winston (New York), 1964.

W. Baur and V. Strassen. The complexity of partial derivatives. *Theor. Comp. Science* **22**, pp. 317–330, 1982.

C. Bavely and G. W. Stewart. An algorithm for computing reducing subspaces by block diagonalization. *SIAM J. Num. Anal.* **16**, pp. 359–367, 1979.

E. R. Berlekamp. *Algebraic coding theory*. McGraw Hill (New York), 1968.

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

T. Beth, B. M. Cook, and D. Gollmann. Architectures for exponentiation in $GF(2^n)$. In *Advances in Cryptology—CRYPTO '86*, ed. A. M. Odlyzko, vol. 263 of *Lecture Notes in Computer Science*, pp. 302–310. Springer (Berlin), 1986.

D. Bini and V. Pan. Improved parallel polynomial division. In *Proc. 33rd IEEE Symp. Foundat. Comput. Sci.*, pp. 131–136, Pittsburgh, USA, 1992.

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**, pp. 581–595, 1978.

J. Bunch and J. Hopcroft. Triangular factorization and inversion by fast matrix multiplicaton. *Math. Comp.* **28**, pp. 231–236, 1974.

D. Cantor and E. Kaltofen. Fast multiplication of polynomials over arbitrary algebras. *Acta Informatica* **28**, pp. 693–701, 1991.

D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.* **36**(154), pp. 587–592, 1981.

L. Carlitz. Primitive roots in a finite field. *Trans. Amer. Math. Soc.* **73**, pp. 373–382, 1952.

C. Chuang and J. Dunham. Matrix extensions of the RSA algorithm. In *Proc. Crypto 1990*, pp. 137–153, 1990.

P. Cohn. *Free Rings and their Relations.* Academic Press (London), 1985.

D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comp.* **9**, pp. 251–280, 1990.

A. Danilevsky. The numerical solution of the secular equation. *Matem. sbornik* **44**(2), pp. 169–171, 1937. In Russian.

H. Davenport. Bases for finite fields. *J. London Math. Soc.* **43**, pp. 21–39, 1968.

J. Demmel. *A Numerical Analyst's Jordan Canonical Form.* PhD thesis, California Institute of Technology, 1983.

L. E. Dickson. *Linear Groups with an Exposition of the Galois Field Theory.* Teubner, Leipzig, 1901; Dover, New York, 1958 (Leipzig), 1901. Dover, New York, 1958.

F. Dorey and G. Whaples. Prime and composite polynomials. *J. Algebra* **28**, pp. 88–101, 1974.

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

G. Eisenstein. Lehrsätze. *J. reine angew. Math.* **39**, pp. 180–182, 1850.

W. Frank. Computing eigenvalues of complex matrices by determinant evaluation and by methods of Daniliewski and Wielandt. *J. SIAM* **6**, pp. 378–392, 1958.

G. Frobenius. Theorie der linearen Formen mit ganzen Coefficienten. *J. reine angew. Math.* **86**, pp. 146–208, 1878.

G. Frobenius. Über die Elementartheiler der Determinanten. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin* pp. 31–44, 1894.

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

F. R. Gantmacher. Geometric theory of elementary divisors after Krull. *Trudy Odessa Gos. Univ. Mat.* **1**, pp. 89–108, 1935.

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

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*, pp. 93–112, Bratislava, 1986.

J. von zur Gathen. Functional decomposition of polynomials: the tame case. *J. Symb. Comp.* **9**, pp. 281–299, 1990a.

J. von zur Gathen. Functional decomposition of polynomials: the wild case. *J. Symb. Comp.* **10**, pp. 437–452, 1990b.

J. von zur Gathen. Efficient and optimal exponentiation in finite fields. *Computational Complexity* **1**, pp. 360–394, 1992.

J. von zur Gathen and M. Giesbrecht. Constructing normal bases in finite fields. *J. Symb. Comp.* **10**, pp. 547–580, 1990.

J. von zur Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Computational Complexity* **2**, pp. 187–224, 1992.

J. von zur Gathen, D. Kozen, and S. Landau. Functional decomposition of polynomials. In *Proc. 28th Ann. IEEE Symp. Foundations of Computer Science*, pp. 127–131, Los Angeles CA, 1987.

M. Giesbrecht. Factoring in skew-polynomial rings. In *Proc. LATIN'92*, pp. 191–203, Sao Paulo, Brasil, 1992a.

M. Giesbrecht. Fast algorithms for matrix normal forms. In *Proc. 33rd IEEE Symp. Foundations of Comp. Sci.*, pp. 121–130, Pittsburgh, USA, 1992b.

I. Gil. Computation of the Jordan canonical form of a square matrix (using the Axiom programming language). In *Proc. ISSAC'92*, pp. 138–145, Berkeley, USA, 1992.

I. Gohberg. *Invariant Subspaces of Matrices with Applications.* Wiley (New York), 1986.

G. Golub and C. Van Loan. *Matrix Computations.* Johns Hopkins University Press (Baltimore, USA), 1989.

G. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM review* **18**, pp. 578–619, 1976.

G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers.* Clarendon Press (Oxford), 1962.

K. Hensel. Ueber die Darstellung der Zahlen eines Gattungsbereiches für einen beliebigen Primdivisor. *J. Reine Angew. Math.* **103**, pp. 230–237, 1888.

J. H. Hodges. Scalar polynomial equations for matrices over a finite field. *Duke Math. J.* **25**, pp. 291–296, 1958.

K. Hoffman and R. Kunze. *Linear Algebra.* Prentice-Hall (Englewood Cliffs, N.J.), 1971.

J. Howell. An algorithm for the exact reduction of a matrix to Frobenius form using modular arithmetic. I. *Math. Comp* **27**, pp. 887–904, 1973a.

J. Howell. An algorithm for the exact reduction of a matrix to Frobenius form using modular arithmetic. II. *Math. Comp* **27**, pp. 905–920, 1973b.

N. Jacobson. *The Theory of Rings.* American Math. Soc. (New York), 1943.

B. Kågström and A. Ruhe. An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Soft.* **6**, pp. 398–419, 1980a.

B. Kågström and A. Ruhe. Algorithm 560 jnf: An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Soft.* **6**, pp. 437–443, 1980b.

E. Kaltofen. Single-factor Hensel lifting and its application to straight-line complexity of certain polynomials. In *Proc. 19th Annual ACM Symp. Theory Comp.*, pp. 443–452, 1987.

E. Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *J. Assoc. Comput. Mach.* **35**, pp. 231–264, 1988.

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

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*, pp. 180–191. ACM Press, 1991.

E. Kaltofen and V. Pan. Processor-efficient parallel solution of linear systems II: The general case. In *Proc. 33rd IEEE Symp. Foundations of Comp. Sci.*, pp. 714–723, Pittsburgh, USA, 1992.

E. Kaltofen and B. D. Saunders. On Wiedemann's method of solving sparse linear systems. In *Proc. AAECC-9*, vol. 539 of *Springer Lecture Notes in Comp. Sci.*, 1991. 29-38.

E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Fast parallel algorithms for similarity of matrices. In *Proc. ACM Symp. Symbolic and Algebraic Computation*, pp. 65–70, Waterloo, Ontario, 1986.

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**, pp. 683–690, 1987.

E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and its Applications* **136**, pp. 189–208, 1990.

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

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

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

W. Keller-Gehrig. Fast algorithms for the characteristic polynomial. *Theor. Computer Science* **36**, pp. 309–317, 1985.

D. E. Knuth. *The Art of Computer Programming, Vol.2, Seminumerical Algorithms.* Addison-Wesley (Reading MA), 2 edition, 1981.

D. Kozen and S. Landau. Polynomial decomposition algorithms. *J. Symb. Comp.* **7**, pp. 445–456, 1989.

S. Lang. *Algebra.* Addison-Wesley (Reading MA), 1984.

B. A. Laws and C. K. Rushforth. A cellular-array multiplier for $GF(2^m)$. *IEEE Trans. Comput.* **C-20**, pp. 1573–1578, 1971.

H. W. Lenstra. Finding isomorphisms between finite fields. *Math. Comp.* **56**, pp. 329–247, 1991.

H. W. Lenstra and R. J. Schoof. Primitive normal bases for finite fields. *Math. Comp.* **48**, pp. 217–231, 1987.

R. Lidl and H. Niederreiter. *Finite Fields*, vol. 20 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley (Reading MA), 1983.

J. D. Lipson. *Elements of Algebra and Algebraic Computing.* Addison-Wesley, 1981.

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

M. Mathieu and D. Ford. On $p$-adic computation of the rational form of a matrix. *J. of Symb. Comp* **10**, pp. 453–464, 1990.

B. McDonald. *Finite Rings with Identity.* Marcel Dekker, Inc. (New York), 1974.

G. Michler. Representations of groups over finite fields. In *Trends in Computer Algebra: Proc. International Symposium*, pp. 119–136, Bad Neuenahr, 1987.

M. Mignotte. An inequality about factors of polynomials. *Math. Comp.* **28**, pp. 1153–1157, 1974.

O. Ore. Theory of non-commutative polynomials. *Annals of Mathematics* **34**(22), pp. 480–508, 1933.

O. Ore. Contributions to the theory of finite fields. *Trans. Amer. Math. Soc.* **36**, pp. 243–274, 1934.

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*, pp. 143–152, Providence, Rhode Island, USA, 1985.

R. A. Parker. The computer calculation of modular characters (the meat-axe). In *Computational Group Theory: Proceedings of the London Mathematical Society Symposium on Computational Group Theory*, pp. 267–274, London, 1984. Academic Press.

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

R. Pierce. *Associative Algebras.* Springer-Verlag (Heidelberg), 1982.

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

L. Rónyai. Simple algebras are difficult. In *Proc. 19th ACM Symp. on Theory of Comp.*, pp. 398–408, New York, 1987.

L. Rónyai. Computing the structure of finite algebras. *J. Symb. Comp.* **9**, pp. 355–373, 1990.

J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Ill. J. Math.* **6**, pp. 64–94, 1962.

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

A. Schönhage. Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm. In *Proc. ICALP 84, Springer Lecture Notes in Computer Science*, vol. 172, pp. 436–447, 1984.

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

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

S. S. Schwarz. Construction of normal bases in cyclic extensions of a field. *Czechoslovak Math. Journal* **38**(113), pp. 291–312, 1988.

I. A. Semaev. Construction of polynomials irreducible over a finite field with linearly independent roots. *Math. USSR Sbornik* **63**(2), pp. 507–519, 1989.

V. Shoup. On the deterministic complexity of factoring polynomials over finite fields. *Information Processing Letters* **33**, pp. 261–267, 1990.

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

V. M. Sidel'nikov. On normal bases of a finite field. *Math. USSR Sbornik* **61**(2), pp. 485–494, 1988.

H. J. S. Smith. On systems of linear indeterminate equations and congruences. *Philos. Trans. Royal Soc. London* **151**, pp. 293–326, 1861.

S. A. Stepanov and I. E. Shparlinsky. On structure complexity of normal basis of finite field. In *Fundamentals of Computation Theory, Proc.*, ed. L. Budach, R. G. Bukharajev, and O. B. Lupanov, vol. 278 of *Lecture Notes in Computer Science*, pp. 414–416. Springer (Berlin), 1987.

S.A. Stepanov and I.E. Shparlinsky. On the construction of a primitive normal basis of a finite field. *Mat. Sbornik* **180**, pp. 1067–1072, 1989.

D. R. Stinson. Some observations on parallel algorithms for fast exponentiation in $GF(2^n)$. *SIAM J. Comp.* **19**(4), pp. 711–717, August 1990.

V. Strassen. Vermeidung von Divisionen. *J. reine angew. Math.* **264**, pp. 182–202, 1973.

B. L. van der Waerden. *Algebra, Erster Teil.* Springer-Verlag (Berlin), 7 edition, 1966.

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

C. C. Wang, T. K. Truong, H. M. Shao, L. J. Deutsch, J. K. Omura, and I. S. Reed. VLSI architectures for computing multiplications and inverses in $GF(2^m)$. *IEEE Trans. Comput.* **C-34**, pp. 709–717, 1985.

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

J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford, Clarendon Press, 1965.

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

R. Zippel. Decomposition of rational functions, 1991. Preprint: Extended abstract in Proc. ISSAC'91.