Chapter 16

Algebraic algorithms

Angel Díaz[1,5]
IBM T. J. Watson Research Center
Yorktown Heights, New York 10598
Internet: aldiaz@us.ibm.com

Ioannis Z. Emiris[2]
INRIA Sophia-Antipolis
B.P. 93, Sophia-Antipolis 06902, France
Internet: Ioannis.Emiris@sophia.inria.fr

Erich Kaltofen[3,5]
Department of Mathematics, North Carolina State University
Raleigh, NC 27695-8205
Internet: kaltofen@eos.ncsu.edu

Victor Y. Pan[4]
Mathematics and Computer Science Department, Lehman College
City University of New York, Bronx, NY 10468
Internet: vpan@lcvax.lehman.cuny.edu

[1]Supported by NSF grant No. CCR-9319776 and by GTE under a Graduate Computer Science Fellowship.
[2]Supported by the European Union under ESPRIT FRISCO project LTR 21.024.
[3]Supported by NSF grant No. CCR-9319776 and CCR-9712267.
[4]Supported by NSF grants Nos. CCR-9020690 and CCR-9625344 and by PSC CUNY Awards 666327 and 667340.
[5]Part of this work was done at Rensselaer Polytechnic Institute in Troy, New York.

Contents

1 Introduction **3**

2 Matrix Computations and Approximation of Polynomial Zeros **3**

2.1 Products of Vectors and Matrices, Convolution of Vectors 3
2.2 Some Computations Related to Matrix Multiplication 6
2.3 Gaussian Elimination Algorithm 7
2.4 Sparse Linear Systems. Direct and Iterative Solution Algorithms 8
2.5 Dense and Structured Matrices and Linear Systems 8
2.6 Parallel Matrix Computations 9
2.7 Rational Matrix Computations, Computations in Finite Fields and Semirings . 9
2.8 Matrix Eigenvalues and Singular Values Problems 10
2.9 Approximating Polynomial Zeros 12

3 Systems of Nonlinear Equations **14**

3.1 The Sylvester Resultant 15
3.2 Resultants of Multivariate Systems 16
3.3 Polynomial System Solving by Using Resultants 17
3.4 Gröbner Bases 19

4 Polynomial Factorization **22**

4.1 Polynomials in a single variable over a finite field 23
4.2 Polynomials in a single variable over fields of characteristic zero 24
4.3 Polynomials in two variables 26
4.4 Polynomials in many variables 26

5 Research Issues and Summary **28**

6 Further Information **39**

1 Introduction

The title's subject is the algorithmic approach to algebra: arithmetic with numbers, polynomials, matrices, differential polynomials, such as $y''+(1/2+x^4/4)y$, truncated series, and algebraic sets, i.e., quantified expressions such as $\exists x \in \mathbb{R}: x^4+p\cdot x+q=0$, which describes a subset of the two-dimensional space with coordinates p and q for which the given quartic equation has a real root. Algorithms that manipulate such objects are the backbone of modern symbolic mathematics software such as the Maple and Mathematica systems, to name but two among many useful systems. This chapter restricts itself to algorithms in four areas: linear algebra, root finding for univariate polynomials, solution of systems of nonlinear algebraic equations, and polynomial factorization (see section 5 on some pointers to the vast further material on algebraic algorithms and section 2.2 and [Pan 1993] on further applications to the graph and combinatorial computations).

2 Matrix Computations and Approximation of Polynomial Zeros

This section covers several major algebraic and numerical problems of scientific and engineering computing that are usually solved numerically, with rounding-off or chopping the input and computed values to a fixed number of bits that fit the computer precision (sections 3 and 4 are devoted to some fundamental infinite precision symbolic computations, and in section 2.7 we comment on the infinite precision techniques for some matrix computations). We also study approximation of polynomial zeros, which is an important, fundamental, and very popular subject, too. In our presentation, we will very briefly list the major subtopics of our huge subject and will give some pointers to the bibliography. We will include brief coverage of the topics of the algorithm design and analysis, regarding the complexity of matrix computation and of approximating polynomial zeros. The reader may find further material on these subjects and further references in the survey articles [Pan 1984a, 1991, 1992a, 1997b] and in the books [Wilkinson 1965, Golub and Van Loan 1996, Bini and Pan 1994, Higham 1996, Demmel 1997, Greenbaum 1997, Trefethen and Bau III 1997, Bini and Pan 1998].

2.1 Products of Vectors and Matrices, Convolution of Vectors

An $m \times n$ matrix $A = [\ a_{i,j}\ ,\ i=0,1,\ldots,m-1;\ j=0,1,\ldots,n-1\ ]$ is a 2-dimensional array, whose (i,j)-entry is $[A]_{i,j}=a_{i,j}$. A is a column vector of dimension m if $n=1$ and is a row vector of dimension n if $m=1$. Transposition, hereafter, indicated by the superscript T, transforms a row vector $\vec{v}^{\,T}=[\ v_0,\ldots,v_{n-1}\ ]$ into a column vector $\vec{v}=[\ v_0,\ldots,v_{n-1}\ ]^T$.

For two vectors, $\vec{u}^{\,T}=[\ u_0,\ldots,u_{m-1}\ ]$ and $\vec{v}^{\,T}=[\ v_0,\ldots,v_{n-1}\ ]^T$, their *outer product* is an $m\times n$ matrix,

$$W=\vec{u}\,\vec{v}^{\,T}=[\ w_{i,j}\ ,\ i=0,\ldots,m-1;\ j=0,\ldots,n-1\ ]\ ,$$

where $w_{i,j} = u_i \, v_j$, for all i and j, and their *convolution* vector is said to equal

$$\vec{w} = \vec{u} \circ \vec{v} = [\ w_0, \ldots, w_{m+n-2}\ ]^T, \quad w_k = \sum_{i=0}^{k} u_i v_{k-i} \ ,$$

where $u_i = v_j = 0$, for $i \geq m$, $j \geq n$; in fact, $\vec{w}$ is the coefficient vector of the product of 2 polynomials,

$$u(x) = \sum_{i=0}^{m-1} u_i x^i \quad \text{and} \quad v(x) = \sum_{i=0}^{n-1} v_i x^i \ ,$$

having coefficient vectors $\vec{u}$ and $\vec{v}$, respectively.

If $m = n$, then the scalar value

$$\vec{v}^{\ T}\ \vec{u} = \vec{u}^{\ T}\ \vec{v} = u_0 v_0 + u_1 v_1 + \cdots + u_{n-1} v_{n-1} = \sum_{i=0}^{n-1} u_i v_i$$

is called the *inner* (*dot*, or *scalar*) *product* of $\vec{u}$ and $\vec{v}$.

The straightforward algorithms compute the inner and outer products of $\vec{u}$ and $\vec{v}$ and their convolution vector by using $2n-1$, mn, and $mn + (m-1)(n-1) = 2mn - m - n + 1$ arithmetic operations (hereafter, referred to as *ops*), respectively. By counting the ops involved, we estimate the arithmetic cost of the computations. This is a realistic measure for the computational complexity if the precision of computing is within the range of the computer precision. In practical numerical computations with rounding-off, the latter requirement is usually stated in terms of *conditioning of the computational problem* and *numerical stability of algorithms* ([Wilkinson 1965, Golub and Van Loan 1996, Higham 1996], [Bini and Pan 1994, ch. 3]). A more universal and computer independent measure is given by the number of Boolean (bit) operations involved into the computations, which grows with the growth of both ops number and precision of computing.

The above upper bounds on the numbers of ops for computing the inner and outer products are sharp, that is, cannot be decreased, for the general pair of the input vectors $\vec{u}$ and $\vec{v}$, whereas (see, e.g. [Bini and Pan 1994]) one may apply the *Fast Fourier Transform* (FFT) (see the next chapter) in order to compute the convolution vector $\vec{u} \circ \vec{v}$ much faster, for larger m and n; namely, it suffices to use $4.5K \log K + 2K$ ops, for $K = 2^k$, $k = \lceil \log(m+n+1) \rceil$. (Here and hereafter, all logarithms are binary unless specified otherwise.)

If $A = [a_{i,j}]$ and $B = [b_{j,k}]$ are $m \times n$ and $n \times p$ matrices, respectively, and $\vec{v} = [v_k]$ is a p-dimensional vector, then the straightforward algorithms compute the vector

$$\vec{w} = B\vec{v} = [\ w_0, \ldots, w_{n-1}\ ]^T, \qquad w_i = \sum_{j=0}^{p-1} b_{i,j} v_j, \quad i = 0, \ldots, n-1 \ ,$$

by using $(2p-1)n$ ops (sharp bound), and compute the *matrix product*

$$AB = [\ w_{i,k}\ ,\ i = 0, \ldots, m-1;\ k = 0, \ldots, p-1\ ]$$

by using $2mnp - mp$ ops, which is $2n^3 - n^2$ if $m = n = p$. The latter upper bound is not sharp: the subroutines for $n \times n$ matrix multiplication on some modern computers, such as CRAY and Connection Machines, rely on algorithms using $O(n^{2.81})$ ops [Golub and Van Loan 1996]. Such algorithms rely on representing an $n \times n$ matrix A, for $n = 2^s$, as a 2×2 block matrix,

$$A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix},$$

where the blocks $A_{i,j}$, $i, j = 0, 1$, are $(n/2) \times (n/2)$ matrices. Multiplication of a pair of such 2×2 block matrices is reduced to 7 multiplications of $(n/2) \times (n/2)$ matrices represented as linear combinations of the input blocks and to 15 or 18 matrix additions/subtractions. For block multiplications, one applies the same algorithm recursively, until arriving at matrices of a small size. More involved technically advanced but nonpractical algorithms decrease the exponent from 2.81 below 2.376 [Coppersmith and Winograd 1990, Bini and Pan 1994, 1998].

If all the input entries and components are bounded integers having short binary representation, each of the above operations with vectors and matrices can be reduced to a single multiplication of 2 longer integers, by means of the techniques of *binary segmentation* (cf. [Pan 1984b, sect. 40]; [Pan 1991, 1992b], or [Bini and Pan 1994, Examples 3.9.1–3.9.3]). The Boolean cost of the computations does not decrease much or even at all, but the techniques may be practically useful if the 2 longer integers still fit the computer precision.

For an $n \times n$ matrix B and an n-dimensional vector $\vec{v}$, one frequently needs to compute the vectors $B^i \, \vec{v}$, $\quad i = 1, 2, \ldots, k-1$, which define *Krylov sequence* or *Krylov matrix*

$$[\ B^i \ \vec{v},\ i = 0, 1, \ldots, k-1\ ]\ ,$$

[Golub and Van Loan 1996, Greenbaum 1997]. The straightforward sequential algorithm takes on $(2n-1)nk$ ops, which is order n^3 if k is of order n. An alternative algorithm (most effective for parallel computation) first computes the matrix powers

$$B^2, B^4, B^8, \ldots, B^{2^s}\ , \qquad s = \lceil\ \log k\ \rceil - 1\ ,$$

and then, the products of $n \times n$ matrices B^{2^i} by $n \times 2^i$ matrices, for $i = 0, 1, \ldots, s$:

$$\begin{array}{ll} B & v\ , \\ B^2 & [\ v,\ Bv\ ] = [\ B^2 v,\ B^3 v\ ]\ , \\ B^4 & [\ v,\ Bv,\ B^2 v,\ B^3 v\ ] = [\ B^4 v,\ B^5 v,\ B^6 v,\ B^7 v\ ]\ , \\ \vdots & \end{array}$$

The last step completes the evaluation of the Krylov sequence, which amounts to $2s$ matrix multiplications, for $k = n$, and, therefore, can be performed (in theory) in $O(n^{2.376} \log k)$ ops [Coppersmith and Winograd 1990].

2.2 Some Computations Related to Matrix Multiplication

Several fundamental matrix computations can be ultimately reduced to relatively few (that is, to a constant number, or, say, to $O(\log n)$) $n \times n$ matrix multiplications. The list of such computations includes the evaluation of the *determinant*, $det\ A$, of an $n \times n$ matrix A; its *inverse* A^{-1} (where A is nonsingular, that is, where $det\ A \neq 0$); the coefficients of its *characteristic polynomial*, $c_A(x) = det(\ xI - A\ )$, x denoting a scalar variable and I being the $n \times n$ identity matrix, which has ones on its diagonal and zeros elsewhere; its *minimal polynomial*, $m_A(x)$; its *rank*, rank A; the solution vector $\vec{x} = A^{-1}\ \vec{v}$ to a nonsingular *linear system of equations*, $A\ \vec{x} = \vec{v}$; various *orthogonal* and *triangular factorizations* of A, and a submatrix of A having the maximal rank, as well as some fundamental *computations with singular matrices.* Furthermore, similar reductions to matrix multiplication have been obtained for some apparently quite distant computational problems, including some major problems of combinatorial and graph computations such as Boolean matrix multiplication and computing the transitive closure of a graph [Aho et al. 1974], computing all pair shortest distances in graphs, [Bini and Pan 1994], page 222, and pattern recognition. Consequently, all these operations can be performed by using (theoretically) $O(n^{2.376})$ ops (cf. [Bini and Pan 1994, chap. 2]). One of the basic ideas is to represent the input matrix A as a block matrix and, operating with its blocks (rather than with its entries), to apply fast matrix multiplication algorithms. In particular, suppose that all the square northwestern blocks of A (called leading principal submatrices) are nonsingular. (This assumption holds for a large and practically important class of matrices, it also can be achieved by means of symmetrization of A or by randomization). Then one may compute det A and A^{-1} by factoring A as a 2×2 block matrix,

$$A = \begin{bmatrix} I & O \\ A_{1,0}A_{0,0}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{0,0} & O \\ O & S \end{bmatrix} \begin{bmatrix} I & A_{0,0}^{-1}A_{0,1} \\ O & I \end{bmatrix},$$

$S = A_{1,1} - A_{1,0}A_{0,0}^{-1}A_{0,1}$, I and O denoting the identity and null matrices, respectively, and then recursively factorizing $A_{0,0}$ and S (see e.g. [Bini and Pan 1994, sect. 2.2]). This is very close to recursive 2×2 block version of Gaussian elimination (cf. section 2.3). Many of such block matrix algorithms are practically important for parallel computations (see section 2.6). On the other hand, however, due to various other considerations (accounting, in particular, for the overhead constants hidden in the $''O''$ notation, for the memory space requirements, and particularly, for numerical stability problems), these computations are, in practice, based either on the straightforward algorithm for matrix multiplication or on other methods allowing order n^3 arithmetic operations (cf. [Golub and Van Loan 1996]).

In the next 3 sections, we will more closely consider the solution of a linear system of equations, $A\,\vec{v} = \vec{b}$, which is the most frequent operation in practice of scientific and engineering computing and is highly important theoretically. We will partition the known solution methods depending on whether the coefficient matrix A is *dense and unstructured*, *sparse*, or *dense and structured.* We will omit the study of singular (in particular, over- and under-determined) linear systems, their least-squares solution, and various computations with singular matrices, referring the reader to [Pan 1992b, Bini and Pan 1994, Golub and Van Loan 1996]. Also, we refer the reader to [Eberly and Kaltofen 1997] (and references therein) on the major subject of solving sparse linear systems of equations in finite fields.

2.3 Gaussian Elimination Algorithm

The solution of a nonsingular (lower or upper) triangular linear system $A\,\vec{x} = \vec{v}$ only involves about n^2 ops. For example [Pan 1992b], let $n = 3$,

$$\begin{aligned} x_1 + 2x_2 - \ x_3 &= \quad 3\ , \\ -2x_2 - 2x_3 &= -10\ , \\ -6x_3 &= -18\ . \end{aligned}$$

Compute $x_3 = 3$ from the last equation, substitute into the previous ones, and arrive at a triangular system of $n - 1 = 2$ equations. In $n - 1$ (in our case, in 2) such recursive substitution steps, we compute the solution.

The triangular case is itself important; furthermore, every nonsingular linear system is reduced to 2 triangular ones by means of *forward elimination* of the variables, which essentially amounts to computing the PLU-factorization of the input matrix A, that is, to computing 2 lower triangular matrices L and U^T (where L has unit values on its diagonal) and a permutation matrix P such that $A = PLU$. (A permutation matrix P is filled with zeros and ones and has exactly one nonzero entry in each row and in each column; in particular, this implies that $P^T = P^{-1}$. $P\,\vec{u}$ has the same components as $\vec{u}$ but written in a distinct (fixed) order, for any vector $\vec{u}$.) As soon as the latter factorization is available, we may compute $\vec{x} = A^{-1}\,\vec{v}$ by solving 2 triangular systems, that is, at first, $L\,\vec{y} = P^T\,\vec{v}$, in $\vec{y}$, and then $U\,\vec{x} = \vec{y}$, in $\vec{x}$. Computing the factorization (elimination stage) is more costly than the subsequent *back substitution stage*, the latter involving about $2n^2$ ops. Gaussian elimination algorithm involves about $2n^3/3$ ops and some comparisons, required in order to ensure appropriate *pivoting*, also called *elimination ordering*. Pivoting enables us to avoid divisions by small values. Otherwise, we would have needed a higher precision of computing, thus facing numerical stability problems. Theoretically, one may employ fast matrix multiplication and compute the matrices P, L, and U in $O(n^{2.376})$ ops [Aho et al. 1974] (and then compute the vectors $\vec{y}$ and $\vec{x}$ in $O(n^2)$ ops). Pivoting can be dropped for some important classes of linear systems, notably, for *positive definite* and for *diagonally dominant* systems ([Golub and Van Loan 1996, Pan 1991, 1992b], or [Bini and Pan 1994]).

We refer the reader to [Golub and Van Loan 1996, pp. 82–83] or [Pan 1992b, p. 794] on sensitivity of the solution to the input and round-off errors in numerical computing. The output errors grow with the *condition number* of A, represented by $||A||\ ||A^{-1}||$ for a fixed matrix norm or by the ratio of maximum and minimum singular values of A. Except for ill-conditioned linear systems $A\,\vec{x} = \vec{v}$, having very large condition numbers, a rough initial approximation to the solution can be rapidly refined (cf. [Golub and Van Loan 1996]) via the *iterative improvement* *algorithm*, as soon as we know P and rough approximations to the matrices L and U of the PLU factorization of A. Then, b correct bits of each output value can be computed in $(b+n)n^2$ ops as $b \to \infty$.

2.4 Sparse Linear Systems. Direct and Iterative Solution Algorithms

A matrix is sparse if it is filled mostly with zeros, say, if its all nonzero entries lie on 3 or 5 of its diagonals; it can be stored economically if the disposition of its nonzero entries has a certain structure. Such matrices arise in many important applications, in particular, to solving partial and ordinary differential equations (PDEs and ODEs). Then, memory space and computation time can be dramatically decreased (say, from order n^2 to order $n \log n$ words of memory and from n^3 to $n^{3/2}$ or $n \log n$ ops) by using some special data structures and special solution methods. The methods are either direct, that is, are modifications of Gaussian elimination with some special policies of elimination ordering that preserve sparsity during the computation (notably, *Markowitz rule* and *nested dissection*, [George and Liu 1981, Gilbert and Tarjan 1987, Lipton et al. 1979, Pan 1993]), or various iterative algorithms. The latter ones usually rely either on computing Krylov sequences [Greenbaum 1997] or on multilevel or multigrid techniques [McCormick 1987, Pan and Reif 1992, Fiorentino and Serra 1996], specialized for solving linear systems that arise from discretization of PDEs. *Banded linear systems* have $n \times n$ coefficient matrices $A = [a_{i,j}]$ where $a_{i,j} = 0$ if $i - j > g$ or $j - i > h$, for $g + h$ being much less than n. For such systems, the nested dissection is known under the name of *block cyclic reduction* and is highly effective, but [Pan et al. 1995] gives some alternative algorithms too. Some special techniques for parallel computation of Krylov sequences for sparse and other special matrices A can be found in [Pan 1995]; according to these techniques, Krylov sequence is recovered from the solution of the associated linear system $(I - A)\ \vec{x} = \vec{v}$, which is solved fast in the case of a special matrix A.

2.5 Dense and Structured Matrices and Linear Systems

Many dense $n \times n$ matrices are defined by $O(n)$, say, by less than $2n$, parameters and can be multiplied by a vector by using $O(n \log n)$ or $O(n \log^2 n)$ ops. Such matrices arise in numerous applications (to signal and image processing, coding, algebraic computation, PDEs, integral equations, particle simulation, Markov chains, and many others). An important example is given by $n \times n$ *Toeplitz matrices* $T = [t_{i,j}]$, $t_{i,j} = t_{i+1,j+1}$ for $i,\ j = 0, 1, \dots, n-2$. Such a matrix can be represented by $2n - 1$ entries of its first row and first column or by $2n - 1$ entries of its first and last columns. The product $T\ \vec{v}$ is defined by vector convolution, and its computation uses $O(n \log n)$ ops. Other major examples are given by *Hankel matrices* (obtained by reflecting the row or column sets of Toeplitz matrices), *circulant* (which are a subclass of Toeplitz matrices), *Bézout*, *Sylvester*, *Vandermonde*, and *Cauchy* matrices. The known solution algorithms for linear systems with such dense structured coefficient matrices use from order $n \log n$ to order $n \log^2 n$ ops. These properties and algorithms are extended via associating some linear operators of displacement and scaling to some more general classes of matrices and linear systems. (See chapter 13 in this volume for details and further bibliography.)

2.6 Parallel Matrix Computations

Algorithms for matrix multiplication are particularly suitable for parallel implementation; one may exploit natural association of processors to rows and/ or columns of matrices or to their blocks, particularly, in the implementation of matrix multiplication on loosely coupled multiprocessors (cf. [Golub and Van Loan 1996, Quinn 1994]). In particular, the straightforward algorithm for $n \times n$ matrix multiplication uses $2n^2$ fetches of input data and n^2 writings into the memory, for $2n^3 - n^2$ ops, that is, in block matrix algorithms, the slow data manipulation operations are relatively few (versus the more numerous but faster ops). This motivates special attention to and rapid progress in devising effective practical parallel algorithms for block matrix computations (see also section 5). The theoretical complexity of parallel computations is usually measured by the computational and communication time and the number of processors involved; decreasing all these parameters, we face a trade-off; the product of time and processor bounds (called potential work of parallel algorithms) cannot usually be made substantially smaller than the sequential time bound for the solution. This follows because, according to a variant of *Brent's scheduling principle*, a single processor can simulate the work of s processors in time $O(s)$. The usual goal of designing a parallel algorithm is in decreasing its parallel time bound (ideally, to a constant, logarithmic or polylogarithmic level, relative to n) and keeping its work bound at the level of the record sequential time bound for the same computational problem (within constant, logarithmic, or at worst polylog factors). This goal has been easily achieved for matrix and vector multiplications, but turned out to be nontrivial for linear system solving, inversion, and some other related computational problems. The recent solution for general matrices [Kaltofen and Pan 1991, 1992] relies on computation of a Krylov sequence and the coefficients of the minimum polynomial of a matrix, by using randomization and auxiliary computations with structured matrices (see the details in [Bini and Pan 1994, Pan 1996c]).

2.7 Rational Matrix Computations, Computations in Finite Fields and Semirings

Rational algebraic computations with matrices are performed for a rational input given with no errors, and the computations are also performed with no errors. The precision of computing can be bounded by reducing the computations modulo one or several fixed primes or prime powers. At the end, the exact output values $z = p/q$ are recovered from $z \ mod \ M$ (if M is sufficiently large relative to p and q) by using the continued fraction approximation algorithm, which is the Euclidean algorithm applied to integers (cf. [Pan 1991], [Pan 1992a], and [Bini and Pan 1994, sect. 3 of ch. 3]). If the output z is known to be an integer lying between $-m$

and m and if $M > 2m$, then z is recovered from z mod M as follows:

$$z = \begin{cases} z \bmod M & \text{if} \quad z \bmod M < m \\ -M + z \bmod M & \text{otherwise} . \end{cases}$$

The reduction modulo a prime p may turn a nonsingular matrix A and a nonsingular linear system $A\vec{x} = \vec{v}$ into singular ones, but this is proved to occur only with a low probability for a random choice of the prime p in a fixed sufficiently large interval (see [Bini and Pan 1994, sect. 9 of ch. 4]). To compute the output values z modulo M for a large M, one may first compute them modulo several relatively prime integers $m_1, m_2, \ldots, m_k$ having no common divisors and such that $m_1 m_2 \ldots m_k > M$ and then easily recover z mod M by means of the Chinese remainder algorithm. For matrix and polynomial computations, there is an effective alternative technique of *p-adic* (*Newton-Hensel*) *lifting* (cf. [Bini and Pan 1994, sect. 3 of ch. 3]), which is particularly powerful for computations with dense structured matrices, since it preserves the structure of a matrix. We refer the reader to [Bareiss 1968] and [Geddes et al. 1992] on some special techniques, which enable one to control the growth of all intermediate values computed in the process of performing rational Gaussian elimination, with no round-off and no reduction modulo an integer. The highly important topic of randomized solution of linear systems of equations in finite fields is covered in [Eberly and Kaltofen 1997], which also contains further bibliography. [Gondran and Minoux 1984] and [Pan 1993] describe some applications of matrix computations on semirings (with no divisions and subtractions allowed) to graph and combinatorial computations.

2.8 Matrix Eigenvalues and Singular Values Problems

The matrix eigenvalue problem is one of the major problems of matrix computation: given an $n \times n$ matrix A, one seeks the maximum k and a $k \times k$ diagonal matrix Λ and an $n \times k$ matrix V of full rank k such that

$$A\,V = V\Lambda. \tag{1}$$

The diagonal entries of Λ are called the *eigenvalues* of A; the entry (i, i) of Λ is associated with the i-th column of V, called an *eigenvector* of A. The eigenvalues of an $n \times n$ matrix A coincide with the zeros of the characteristic polynomial

$$c_A(x) = \det(\ xI - A\ )\ .$$

If this polynomial has n distinct zeros, then $k = n$, and V of (1) is a nonsingular $n \times n$ matrix. The Toeplitz matrix $A = I + Z$, $Z = (z_{i,j})$, $z_{i,j} = 0$ unless $j = i + 1$, $z_{i,i+1} = 1$, is an example of a matrix for which $k = 1$, so that the matrix V degenerates to a vector.

In principle, one may compute the coefficients of $c_A(x)$, the characteristic polynomial of A, and then approximate its zeros (see the next section) in order to approximate the eigenvalues of A. Given the eigenvalues, the corresponding eigenvectors can be recovered by means of the inverse power iteration [Golub and Van Loan 1996], [Wilkinson 1965]. Practically, the computation of the eigenvalues via the computation of the coefficients of $c_A(x)$ is not recommended, due to arising numerical stability problems [Wilkinson 1965], and most frequently, the eigenvalues and eigenvectors of a general (unsymmetric) matrix are approximated by means of the *QR algorithm* [Wilkinson 1965, Golub and Van Loan 1996]. Before application of this algorithm, the matrix A is simplified by transforming it into the more special (*lower Hessenberg*) *form*, $H = [h_{i,j}]$, $h_{i,j} = 0$ if $i - j > 1$, by a *similarity transformation*,

$$H = UAU^H \tag{2}$$

where $U = [u_{i,j}]$ is a unitary matrix, $U^H U = I$, $U^H = [\overline{u}_{j,i}]$ is the Hermitian transpose of U, $\overline{z}$ denoting the complex conjugate of z; $U^H = U^T$ if U is a real matrix [Golub and Van Loan 1996]. Similarity transformation into Hessenberg form is a *rational transformation* of a matrix into a special *canonical form*, *Smith* and *Hermite forms* are two other most important representatives of canonical forms [Kaltofen et al. 1990, Geddes et al. 1992, Giesbrecht 1995].

The eigenvalue problem is *symmetric*, if the matrix A is real symmetric,

$$A^T = [a_{j,i}] = A = [a_{i,j}] \ ,$$

or complex Hermitian,

$$A^H = [\overline{a}_{j,i}] = A = [a_{i,j}] \ .$$

The symmetric eigenvalue problem is simpler: the matrix V of (1) is a nonsingular $n \times n$ matrix, and all the eigenvalues are real and little sensitive to small input perturbations of A [Parlett 1980], [Golub and Van Loan 1996]. Furthermore, the Hessenberg matrix H of (2) becomes symmetric tridiagonal (cf. [Golub and Van Loan 1996] or [Bini and Pan 1994, sect. 2.3]). For such a matrix H, application of the QR algorithm is dramatically simplified; moreover, two competitive algorithms are also widely used, that is, the *bisection* [Parlett 1980] (a slightly slower but very robust algorithm) and the *divide-and-conquer* method [Cuppen 1981], [Golub and Van Loan 1996]. The latter method has a modification [Bini and Pan 1991] that only uses $O(n \log^2 n(\log n + \log^2 b))$ arithmetic operations in order to compute all the eigenvalues of an $n \times n$ symmetric tridiagonal matrix A within the output error bound $2^{-b}||A||$, where $||A|| \leq n \max |a_{i,j}|$.

A natural generalization of the eigenproblem (1) is to the *generalized eigenproblem*. Given a pair A, B of matrices, the generalized eigenvalue Λ and the generalized eigenvector V satisfy

$$A\ V = B\ V\Lambda.$$

The solution algorithm should not compute matrix $B^{-1}A$ explicitly, so as to avoid problems of numerical stability.

Another important extension of the symmetric eigenvalue problem is *singular value decomposition* (SVD) of a (generally unsymmetric and, possibly, rectangular) matrix A: $A = U\Sigma V^T$, where U and V are unitary matrices, $U^H U = V^H V = I$, and Σ is a diagonal (generally rectangular) matrix, filled with zeros, except for its diagonal, filled with (positive) singular values of A and, possibly, with zeros. The SVD is widely used in the study of numerical stability of matrix computations and in numerical treatment of singular and ill-conditioned (close to singular) matrices. It is a basic tool, for instance, in the study of approximate polynomial GCD [Corless et al. 1995, Emiris et al. 1997].

2.9 Approximating Polynomial Zeros

Solution of an n-th degree polynomial equation,

$$p(x) = \sum_{i=0}^{n} p_i \ x^i = 0 \ , \qquad p_n \neq 0$$

(where one may assume that $p_{n-1} = 0$; this can be achieved via shifting the variable x), is a classical problem that has greatly influenced the development of mathematics throughout the centuries [Pan 1997a]. The problem remains highly important for the theory and practice of present day computing, and dozens of new algorithms for its approximate solution appear every year. Among the existent implementations of such algorithms, the practical heuristic champions in efficiency (in terms of computer time and memory space used, according to the results of many experiments) are various modifications of *Newton's iteration*, $z(i+1) = z(i) - a(i)p(z(i))/p'(z(i))$, $a(i)$ being the step-size parameter [Madsen 1973], *Laguerre's method* [Hansen et al. 1977, Foster 1981], and the randomized *Jenkins-Traub algorithm* [Jenkins and Traub 1970] (all three for approximating a single zero z of $p(x)$), which can be extended to approximating other zeros by means of deflation of the input polynomial via its numerical division by $x - z$. For simultaneous approximation of all the zeros of $p(x)$, one may apply the Durand-Kerner algorithm, which is defined by the following recurrence:

$$z_j(l+1) = z_j(l) - \frac{p(z_j(l))}{\prod_{i \neq j}(z_j(l) - z_i(l))} \ , \quad j = 1, \ldots, n, \quad l = 1, 2, \ldots . \tag{3}$$

Here, the most customary choice (see [Bini and Pan 1998] for some effective alternatives) for the n initial approximations $z_j(0)$ to the n zeros of

$$p(x) = p_n \prod_{j=1}^{n} (x - z_j)$$

is given by $z_j(0) = Z \ exp(2\pi\sqrt{-1}/n)$, $\quad j = 1, \ldots, n$, Z exceeding (by some fixed factor $t > 1$) $\max_j |z_j|$; for instance, one may set

$$Z = 2 \ t \ \max_{i<n}(p_i/p_n). \tag{4}$$

For a fixed l and for all j, the computation according to (3) is simple, only involving order n^2 ops, (or even $O(n \log^2 n)$ ops with deteriorated numerical stability [Bini and Pan 1998]).

Furthermore, according to the results of many experiments, the iteration (3) rapidly converges to the solution, though no theory confirms or explains these results. Similar is the situation with various modifications of this algorithm, which are now even more popular than the original algorithms. The reader is referred to [Bini and Pan 1998, McNamee 1993, Pan 1992a], and [Pan 1997a] on many of these algorithms, some implementation issues, and further extensive bibliography.

Neither of the algorithms (cited above) supports any reasonable good estimates for the computational complexity of approximating polynomial zeros for the worst case inputs, but such estimates have been achieved based on algorithms of two other groups. One such group is given by the modern modifications and improvements (due to [Pan 1987, Renegar 1987, Pan 1996a]) of *Weyl's quadtree construction* of 1924. In this approach, an initial square S, containing all the zeros of $p(x)$ is recursively partitioned into 4 congruent subsquares; say, $S = \{ \ x \ , \ | \ Im \ x \ | < Z, \ | \ Re \ x \ | < Z \ \}$ for Z of (4). In the center of each of them, a proximity test is applied that estimates the distance from this center to the closest zero of $p(x)$. If such a distance exceeds one half of the diagonal length, then the subsquare contains no zeros of $p(x)$ and is discarded. When this process ensures a strong isolation from each other for the components formed by the remaining squares, then certain extensions of Newton's iteration [Renegar 1987, Pan 1996a] or some iterative techniques based on numerical integration [Pan 1987] are applied and very rapidly converge to the desired approximations to the zeros of $p(x)$, within the error bound $2^{-b}Z$ for Z of (4). As a result, the algorithms of [Pan 1987, 1996a] solve the entire problem of approximating (within $2^{-b}Z$) all the zeros of $p(x)$ at the overall cost of performing $O(\ (n^2 \log n) \ \log(bn) \ )$ ops (cf. [Bini and Pan 1998]).

The second group is given by the *divide-and-conquer algorithms*. They first compute a sufficiently wide annulus A, which is free of the zeros of $p(x)$ and contains comparable numbers of such zeros (that is, the same numbers up to a fixed constant factor) in its exterior and its interior. Then the 2 factors of $p(x)$ are numerically computed, that is, $F(x)$, having all its zeros in the interior of the annulus, and $G(x) = p(x)/F(x)$, having no zeros there. The same process is recursively repeated for $F(x)$ and $G(x)$ until factorization of $p(x)$ into the product of linear factors is computed numerically. From this factorization, approximations to all the zeros of $p(x)$ are obtained. The algorithms of [Pan 1995, 1996b] based on this approach only require $O(\ n \log(bn) \ (\log n)^2 \ )$ ops in order to approximate all the n zeros of $p(x)$ within $2^{-b}Z$ for Z of (4). (Note that this is a quite sharp bound: at least n ops are necessary in order to output n distinct values.)

The computations for the polynomial zero problem are ill-conditioned, that is, they generally require a high precision for the worst case input polynomials in order to ensure a required output precision, no matter which algorithm is applied for the solution. Consider, for instance, the polynomial $(x - \frac{6}{7})^n$ and perturb its x-free coefficient by 2^{-bn}. Observe the resulting jumps of the zero $x = 6/7$ by 2^{-b}, and observe similar jumps if the coefficients p_i are perturbed by $2^{(i-n)b}$ for $i = 1, 2, \ldots, n-1$. Therefore, to ensure the output precision of b bits, we need an input precision of at least $(n-i)b$ bits for each coefficient p_i, $i = 0, 1, \ldots, n-1$. Consequently, for the worst case input polynomial $p(x)$, any solution algorithm needs at least about by factor n increase of the precision of the input and of computing, versus the output precision.

Numerically unstable algorithms may require even a higher input and computation precision, but inspection shows that this is not the case for the algorithms of [Pan 1987, Renegar 1987, Pan 1996a,b, Bini and Pan 1998].

3 Systems of Nonlinear Equations

Given a system $P = \{p_1(x_1, \ldots, x_n), p_2(x_1, \ldots, x_n), \ldots, p_r(x_1, \ldots, x_n)\}$ of nonlinear polynomials with rational coefficients (each $p_i(x_1, \ldots, x_n)$ is said to be an element of $\mathbb{Q}[x_1, \ldots, x_n]$, the ring of polynomials in $x_1, \ldots, x_n$ over the field of rational numbers), the n-tuple of complex numbers $(a_1, \ldots, a_n)$ is a solution of the system if $f_i(a_1, \ldots, a_n) = 0$ for each i with $1 \leq i \leq r$. In this section, we explore the problem of exactly solving a system of nonlinear equations over the field $\mathbb{Q}$. We also indicate how an initial phase of exact algebraic computation leads to certain numerical methods that approximate the solutions; the interaction of symbolic and numeric computation is currently an active domain of research [Emiris 1998]. We provide an overview and cite references to different symbolic techniques used for solving systems of algebraic (polynomial) equations. In particular, we describe methods involving *resultant* and *Gröbner basis* computations.

The *Sylvester resultant method* is the technique most frequently utilized for determining a common zero of two polynomial equations in one variable [Knuth 1981]. However, using the Sylvester method successively to solve a system of multivariate polynomials proves to be inefficient. Successive resultant techniques, in general, lack efficiency as a result of their sensitivity to the ordering of the variables [Kapur and Lakshman Y. N. 1992]. It is more efficient to eliminate all variables together from a set of polynomials, thus leading to the notion of the *multivariate resultant.* The three most commonly used multivariate resultant formulations are the *Dixon* [Dixon 1908, Kapur and Saxena 1995] *Macaulay* [Macaulay 1916, Canny 1990, Canny et al. 1989], and *sparse resultant formulations* [Canny and Emiris 1993, Sturmfels 1991].

The theory of Gröbner bases provides powerful tools for performing computations in multivariate polynomial rings. Formulating the problem of solving systems of polynomial equations in terms of polynomial ideals, we will see that a Gröbner basis can be computed from the input polynomial set, thus allowing for a form of back substitution (cf. section 1.3) in order to compute the common roots.

Although not discussed, it should be noted that the *characteristic set algorithm* can be utilized for polynomial system solving. Ritt [1950] introduced the concept of a characteristic set as a tool for studying solutions of algebraic differential equations. In 1984, Wu [1984] in search of an effective method for automatic theorem proving, converted Ritt's method to ordinary polynomial rings. Given the before mentioned system P, the characteristic set algorithm transforms P into a triangular form, such that the set of common zeros of P is equivalent to the set of roots of the triangular system [Kapur and Lakshman Y. N. 1992].

Throughout this exposition we will also see that these techniques used to solve nonlinear equations can be applied to other problems as well, such as computer aided design and automatic geometric theorem proving.

3.1 The Sylvester Resultant

The question of whether two polynomials $f(x),\ g(x) \in \mathbb{Q}[x]$,

$$\begin{aligned} f(x) &= f_n x^n + f_{n-1}x^{n-1} + \ldots + f_1 x + f_0, \\ g(x) &= g_m x^m + g_{m-1}x^{m-1} + \ldots + g_1 x + g_0, \end{aligned}$$

have a common root leads to a condition that has to be satisfied by the coefficients of both f and g. Using a derivation of this condition due to Euler, the *Sylvester matrix* of f and g (which is of order $m+n$) can be formulated. The vanishing of the determinant of the Sylvester matrix, known as the *Sylvester resultant*, is a necessary and sufficient condition for f and g to have common roots [Knuth 1981].

As a running example let us consider the following system in two variables provided by Lazard [1981]:

$$\begin{aligned} f &= x^2 + xy + 2x \qquad + \ y - 1 = 0, \\ g &= x^2 \qquad\quad + 3x - y^2 + 2y - 1 = 0. \end{aligned}$$

The Sylvester resultant can be used as a tool for eliminating several variables from a set of equations [Kapur and Lakshman Y. N. 1992]. Without loss of generality, the roots of the Sylvester resultant of f and g treated as polynomials in y, whose coefficients are polynomials in x, are the x-coordinates of the common zeros of f and g. More specifically, the Sylvester resultant of the Lazard system with respect to y is given by the following determinant:

$$\det(\begin{bmatrix} x+1 & x^2+2x-1 & 0 \\ 0 & x+1 & x^2+2x-1 \\ -1 & 2 & x^2+3x-1 \end{bmatrix}) = -x^3 - 2x^2 + 3x.$$

An alternative matrix formulation named after Bézout yields the same determinant. This formulation is discussed below in the context of multivariate polynomials, in section 3.2.

The roots of the Sylvester resultant of f and g are $\{-3, 0, 1\}$. For each x value, one can substitute the x value back into the original polynomials yielding the solutions $(-3,1), (0,1), (1,-1)$.

The method just outlined can be extended recursively, using *polynomial GCD computations*, to a larger set of multivariate polynomials in $\mathbb{Q}[x_1, \ldots, x_n]$. This technique, however, is impractical for eliminating many variables, due to an explosive growth of the degrees of the polynomials generated in each elimination step.

The Sylvester formulations has led to a *subresultant theory*, developed simultaneously by G.E. Collins and W.S. Brown and J. Traub. The subresultant theory produced an efficient algorithm for computing polynomial GCDs and their resultants, while controlling intermediate expression swell [Brown and Traub 1971, Collins 1967, Knuth 1981].

It should be noted that by adopting an implicit representation for symbolic objects, the intermediate expression swell introduced in many symbolic computations can be palliated. Recently, polynomial GCD algorithms have been developed that use implicit representations and thus avoid the computationally costly content and primitive part computations needed in those GCD algorithms for polynomials in explicit representation [Kaltofen 1988, Kaltofen and Trager 1990, Díaz and Kaltofen 1995].

3.2 Resultants of Multivariate Systems

The solvability of a set of nonlinear multivariate polynomials can be determined by the vanishing of a generalization of the Sylvester resultant of two polynomials in a single variable. We examine two generalizations, namely, the classical and the sparse resultants. The *classical resultant* of a system of n homogeneous polynomials in n variables vanishes exactly when there exists a common solution in *projective* space [van der Waerden 1950, Kapur and Lakshman Y. N. 1992]. The *sparse resultant* characterizes solvability over a smaller space which coincides with affine space under certain genericity conditions [Gelfand et al. 1994, Sturmfels 1991]. The main algorithmic question, then, is to construct a matrix whose determinant is the resultant or a non-trivial multiple of it.

Due to the special structure of the Sylvester matrix, Bézout developed a method for computing the resultant as a determinant of order $\mathrm{Max}(m, n)$ during the eighteenth century. Cayley [Cayley 1865] reformulated Bézout's method leading to Dixon's [Dixon 1908] extension to the bivariate case. Dixon's method can be generalized to a set

$$\{p_1(x_1, \ldots, x_n), p_2(x_1, \ldots, x_n), \ldots, p_{n+1}(x_1, \ldots, x_n)\}$$

of $n+1$ generic n-degree polynomials in n variables [Kapur et al. 1994]. The vanishing of the determinant of the Dixon matrix is a necessary and sufficient condition for the polynomials to have a non-trivial projective common zero, and also a necessary condition for the existence of an affine common zero. The Dixon formulation gives the resultant up to a multiple, and hence in the affine case it may happen that the vanishing of the Dixon determinant does not necessarily indicate that the equations in question have a common root. A non-trivial multiple, known as the *projection operator*, can be extracted via a method based on so-called *rank subdeterminant computation* (RSC) [Kapur et al. 1994]. It should be noted that the RSC method can also be applied to the Macaulay and sparse resultant formulations as is detailed below. A more general and simpler method for extracting a projection operator from the Dixon matrix is discussed in [Cardinal and Mourrain 1996], thm. 3.3.4. This article, along with [Elkadi and Mourrain 1996], explain the correlation between residue theory and the Dixon matrix, which yields an alternative method for studying and approximating all common solutions.

In 1916, Macaulay [1916] constructed a matrix whose determinant is a multiple of the classical resultant for n homogeneous polynomials in n variables. The Macaulay matrix simultaneously generalizes the Sylvester matrix and the coefficient matrix of a system of linear equations [Kapur and Lakshman Y. N. 1992]. As the Dixon formulation, the Macaulay determinant is a multiple of the resultant. Macaulay, however, proved that a certain minor of his matrix divides the matrix determinant so as to yield the exact resultant in the case of generic homogeneous polynomials. Canny [1990] has invented a general method that perturbs any polynomial system and extracts a non-trivial projection operator.

Using recent results pertaining to sparse polynomial systems [Gelfand et al. 1994, Sturmfels 1991], a matrix formula for computing the sparse resultant of $n+1$ polynomials in n variables was given by Canny and Emiris [1993] and consequently improved in [Canny and Pedersen 1993, Emiris and Canny 1995]. The determinant of the sparse resultant matrix, like the Macaulay and Dixon matrices, only yields a projection operation, not the exact resultant.

Here, sparsity means that only certain monomials in each of the $n+1$ polynomials have non-zero coefficients. Sparsity is measured in geometric terms, namely, by the *Newton polytope* of the polynomial, which is the convex hull of the exponent vectors corresponding to non-zero coefficients. The *mixed volume* of the Newton polytopes of n polynomials in n variables is defined as a certain integer-valued function that bounds the number of affine common roots of these polynomials, according to a theorem of [Bernstein 1975]. This remarkable theorem is the cornerstone of sparse elimination. The mixed volume bound is significantly smaller than the classical Bézout bound for polynomials with small Newton polytopes. Since these bounds also determine the degree of the sparse and classical resultants, respectively, the latter has larger degree for sparse polynomials.

3.3 Polynomial System Solving by Using Resultants

Suppose we are asked to find the common zeros of a set of n polynomials in n variables $\{p_1(x_1,\ldots,x_n),\ p_2(x_1,\ldots,x_n),\ \ldots,\ p_n(x_1,\ldots,x_n)\}$. By augmenting the polynomial set by a generic linear polynomial [Canny 1990, Canny and Manocha 1991, Kapur and Lakshman Y. N. 1992], one can construct the *u-resultant* of a given system of polynomials. The u-resultant is named after the vector of indeterminates u, traditionally used to represent the generic coefficients of the additional linear polynomial. The u-resultant factors into linear factors over the complex numbers, providing the common zeros of the given polynomials equations. The u-resultant method takes advantage of the properties of the multivariate resultant, and hence can be constructed using either Dixon's, Macaulay's, or sparse formulations. An alternative approach, where we *hide* a variable in the coefficient field, instead of adding a polynomial, is discussed in [Manocha 1992, Emiris 1996].

Consider the previous example augmented by a generic linear form:

$$\begin{aligned}
f_1 &= x^2 + xy + 2x \qquad + \ \ y - \ 1 = 0,\\
f_2 &= x^2 \qquad\qquad + 3x - y^2 + 2y - \ 1 = 0,\\
f_l &= \qquad\qquad\quad ux \qquad + vy + w = 0.
\end{aligned}$$

As described in Canny, Kaltofen and Lakshman [1989], the following matrix M corresponds to the Macaulay u-resultant of the above system of polynomials, with z being the homogenizing variable:

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & u & 0 & 0 & 0 \\ 2 & 0 & 1 & 3 & 0 & 1 & 0 & u & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & v & 0 & 0 & 0 \\ 1 & 2 & 1 & 2 & 3 & 0 & w & v & u & 0 \\ -1 & 0 & 2 & -1 & 0 & 3 & 0 & w & 0 & u \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 2 & -1 & 0 & 0 & v & 0 \\ 0 & -1 & 1 & 0 & -1 & 2 & 0 & 0 & w & v \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & w \end{bmatrix}.$$

It should be noted that

$$\det(M) = (u - v + w)(-3u + v + w)(v + w)(u - v)$$

corresponds to the affine solutions $(1, -1)$, $(-3, 1)$, $(0, 1)$, and one solution at infinity.

Resultants may also be applied to reduce polynomial system solving to a regular or generalized eigenproblem (cf. sect. 2.8), thus transforming the nonlinear question to a problem in linear algebra. This is a classical technique that enables us to approximate all solutions (cf. [Auzinger and Stetter 1988, Canny and Manocha 1991, Cardinal and Mourrain 1996, Emiris 1996]). For demonstration, consider the previous system and its resultant matrix M. The matrix rows are indexed by the following row vector of monomials in the eliminated variables:

$$\vec{v} = [x^3, x^2y, x^2, xy^2, xy, x, y^3, y^2, y, 1].$$

Vector $\vec{v}M$ expresses the polynomials indexing the columns of M, which are multiples of the three input polynomials by various monomials. Let us specialize variables u and v to random values. Then the matrix M contains a single variable w and is denoted $M(w)$. Solving the linear system $\vec{v}M(w) = \vec{0}$ in vector $\vec{v}$ and in scalar w is a generalized eigenproblem, since $M(w)$ can be represented as $M_0 + wM_1$, where M_0 and M_1 have numeric entries. If, moreover, M_1 is invertible, we arrive at the following eigenproblem:

$$\vec{v}\left(M_0 + wM_1\right) = \vec{0} \Longleftrightarrow \vec{v}\left(-M_1^{-1}M_0 - wI\right) = \vec{0} \Longleftrightarrow \vec{v}\left(-M_1^{-1}M_0\right) = w\vec{v}.$$

For every solution (a, b) of the original system, there is a vector $\vec{v}$ among the computed eigenvectors, which we evaluate at $x = a,\ y = b$ and from which the solution can be recovered by means of division (cf. [Emiris 1996]). As for the eigenvalues, they correspond to the values of w at the solutions.

Recently, the structure of various resultant matrices has been studied, continuing work in [Canny et al. 1989]. By exploiting the quasi-Toeplitz or quasi-Hankel structure of such matrices, it is possible to accelerate their construction, the computation of the resultant itself, and the approximation of the system's solutions [Mourrain and Pan 1997, Emiris and Pan 1997].

An empirical comparison of the detailed resultant formulations can be found in [Kapur and Saxena 1995, Manocha 1992]. The multivariate resultant formulations have been used for diverse applications such as *algebraic and geometric reasoning* [Manocha 1992, Kapur et al. 1994, Cardinal and Mourrain 1996], *computer-aided design* [Stederberg and Goldman 1986, Krishnan and Manocha 1995], *robot kinematics* [Manocha 1992, Emiris 1994], computing *molecular conformations* [Emiris 1994, Manocha et al. 1995] and for *implicitization and finding base points* [Chionh 1990, Manocha 1992].

3.4 Gröbner Bases

Solving systems of nonlinear equations can be formulated in terms of polynomial ideals [Becker and Weispfenning 1993, Geddes et al. 1992, Winkler 1996]. Let us first establish some terminology.

The ideal generated by a system of polynomial equations $p_1, \ldots, p_r$ over $\mathbb{Q}[x_1, \ldots, x_n]$ is the set of all linear combinations

$$(p_1, \ldots, p_r) = \{h_1 p_1 + \ldots + h_r p_r \mid h_1, \ldots, h_r \in \mathbb{Q}[x_1, \ldots, x_n]\}.$$

The algebraic variety of $p_1, \ldots, p_r \in \mathbb{Q}[x_1, \ldots, x_n]$ is the set of their common zeros,

$$V(p_1, \ldots, p_r) = \{(a_1, \ldots, a_n) \in \mathbb{C}^n \mid f_1(a_1, \ldots, a_n) = \ldots = f_r(a_1, \ldots, a_n) = 0\}.$$

A version of the *Hilbert Nullstellensatz* states that

$$V(p_1, \ldots, p_r) = \text{the empty set } \emptyset \Longleftrightarrow 1 \in (p_1, \ldots, p_r) \text{ over } \mathbb{Q}[x_1, \ldots, x_n],$$

which relates the solvability of polynomial systems to the ideal membership problem.

A term $t = x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}$ of a polynomial is a product of powers with $\deg(t) = e_1 + e_2 + \ldots + e_n$. In order to add needed structure to the polynomial ring we will require that the terms in a polynomial be ordered in an admissible fashion [Geddes et al. 1992, Kapur and Lakshman Y. N. 1992]. Two of the most common admissible orderings are the *lexicographic order* ($\prec_l$), where terms are ordered as in a dictionary, and the *degree order* ($\prec_d$), where terms are first compared by their degrees with equal degree terms compared lexicographically. A variation to the lexicographic order is the *reverse lexicographic order*, where the lexicographic order is reversed [Davenport et al. 1988], page 96.

It is this above mentioned structure that permits a type of simplification known as polynomial reduction. Much like a polynomial remainder process, the process of polynomial reduction involves subtracting a multiple of one polynomial from another to obtain a smaller degree result [Becker and Weispfenning 1993, Geddes et al. 1992, Kapur and Lakshman Y. N. 1992, Winkler 1996].

A polynomial g is said to be reducible with respect to a set $P = \{p_1, \ldots, p_r\}$ of polynomials if it can be reduced by one or more polynomials in P. When g is no longer reducible by the polynomials in P, we say that g is *reduced* or is *a normal form* with respect to P.

For an arbitrary set of basis polynomials, it is possible that different reduction sequences applied to a given polynomial g could reduce to different normal forms. A basis $G \subseteq \mathbb{Q}[x_1, \ldots, x_n]$ is a *Gröbner basis* if and only if every polynomial in $\mathbb{Q}[x_1, \ldots, x_n]$ has a unique normal form with respect to G. Bruno Buchberger [Buchberger Fall 1965, 1976, 1983, 1985] showed that every basis for an ideal $(p_1, \ldots, p_r)$ in $\mathbb{Q}[x_1, \ldots, x_n]$ can be converted into a Gröbner basis $\{p_1^*, \ldots, p_s^*\} = GB(p_1, \ldots, p_r)$, concomitantly designing an algorithm that transforms an arbitrary ideal basis into a Gröbner basis. Another characteristic of Gröbner bases is that by using the above mentioned reduction process we have

$$g \in (p_1 \ldots, p_r) \iff (g \bmod p_1^*, \ldots, p_s^*) = 0.$$

Further, by using the Nullstellensatz it can be shown that $p_1 \ldots, p_r$ viewed as a system of algebraic equations is solvable if and only if $1 \notin GB(p_1, \ldots, p_r)$.

Depending on which admissible term ordering is used in the Gröbner bases construction, an ideal can have different Gröbner bases. However, an ideal cannot have different (reduced) Gröbner bases for the same term ordering.

Any system of polynomial equations can be solved using a lexicographic Gröbner basis for the ideal generated by the given polynomials. It has been observed, however, that Gröbner bases, more specifically lexicographic Gröbner bases, are hard to compute [Becker and Weispfenning 1993, Geddes et al. 1992, Lakshman 1990, Winkler 1996]. In the case of zero-dimensional ideals, those whose varieties have only isolated points, Faugère, Gianni, Lazard and Mora [1993] outlined a change of basis algorithm which can be utilized for solving zero-dimensional systems of equations. In the zero-dimensional case, one computes a Gröbner basis for the ideal generated by a system of polynomials under a degree ordering. The so-called *change of basis algorithm* can then be applied to the degree ordered Gröbner basis to obtain a Gröbner basis under a lexicographic ordering.

Turning to Lazard's example in form of a polynomial basis,

$$\begin{aligned} f_1 &= x^2 + xy + 2x \phantom{{}-y^2 + 2y} + y - 1, \\ f_2 &= x^2 \phantom{{}+ xy} + 3x - y^2 + 2y - 1, \end{aligned}$$

one obtains (under lexicographical ordering with $x \prec_l y$) a Gröbner basis in which the variables are triangularized such that the finitely many solutions can be computed via back substitution:

$$
\begin{aligned}
{f_1}^* &= x^2 \quad + 3x \quad + 2y - 2, \\
{f_2}^* &= \quad xy - \ x \quad - \ y + 1, \\
{f_3}^* &= \qquad\qquad y^2 \qquad - 1.
\end{aligned}
$$

It should be be noted that the final univariate polynomial is of minimal degree and the polynomials used in the back substitution will have degree no larger than the number of roots.

As an example of the process of polynomial reduction with respect to a Gröbner basis, the following demonstrates two possible reduction sequences to the same normal form. The polynomial x^2y^2 is reduced with respect to the previously computed Gröbner basis $\{f_1^*, f_2^*, f_3^*\} = GB(f_1, f_2)$ along the following two distinct reduction paths, both yielding $-3x - 2y + 2$ as the normal form.

$$
\begin{tikzcd}
 & x^2y^2 \arrow[d, "f_1^*"] & \\
 & -3xy^2 - 2y^3 + 3y^2 \arrow[dl, "f_2^*"'] \arrow[dr, "f_3^*"] & \\
-3xy - 2y^3 - y^2 + 3y \arrow[d, "f_2^*"'] & & -3x - 2y^3 + 2y^2 \arrow[d, "f_3^*"] \\
-3x - 2y^3 - y^2 + 3 \arrow[d, "f_3^*"'] & & -3x - 2y^3 + 2y^2 \arrow[ddl, "f_3^*"] \\
-3x - y^2 - 2y + 3 \arrow[dr, "f_3^*"'] & & \\
 & -3x - 2y + 2 &
\end{tikzcd}
$$

There is a strong connection between lexicographic Gröbner bases and the previously mentioned resultant techniques. For some types of input polynomials, the computation of a reduced system via resultants might be much faster than the computation of a lexicographic Gröbner basis. A good comparison between the Gröbner computations and the different resultant formulations can be found in [Kapur and Saxena 1995, Manocha 1992].

In a survey article, Buchberger [1985] detailed how Gröbner bases can be used as a tool for many polynomial ideal theoretic operations. Other applications of Gröbner basis computations include automatic geometric theorem proving [Kapur 1986, Wu 1984], multivariate polynomial factorization and GCD computations [Gianni and Trager 1985] , and polynomial interpolation [Lakshman and Saunders 1995, 1994].

4 Polynomial Factorization

The problem of factoring polynomials is a fundamental task in symbolic algebra. An example in one's early mathematical education is the factorization $x^2 - y^2 = (x+y)\cdot(x-y)$, which in algebraic terms is a factorization of a polynomial in two variables with integer coefficients. Technology has advanced to a state where most polynomial factorization problems are doable on a computer, in particular, with any of the popular mathematical software, such as the Mathematica or Maple systems. For instance, the factorization of the determinant of a 6×6 symmetric Toeplitz matrix over the integers is computed in Maple as

```
>readlib(showtime):
>showtime():
O1 := T := linalg[toeplitz]([a,b,c,d,e,f]);
```

$$T := \begin{bmatrix} a & b & c & d & e & f \\ b & a & b & c & d & e \\ c & b & a & b & c & d \\ d & c & b & a & b & c \\ e & d & c & b & a & b \\ f & e & d & c & b & a \end{bmatrix}$$

```
time 0.03 words 7701
O2 := factor(linalg[det](T));
```

$$\begin{aligned} &-(2\,d\,c\,a - 2\,b\,c\,e + 2\,c^2\,a - a^3 - d\,a^2 + 2\,d^2\,c + d^2\,a + b^3 + 2\,a\,b\,c - 2\,c^2\,b \\ &\quad + d^3 + 2\,a\,b^2 - 2\,d\,c\,b - 2\,c\,b^2 - 2\,e\,c^2 + 2\,e\,b^2 + 2\,f\,c\,b + 2\,b\,a\,e \\ &\quad + b^2\,f + c^2\,f + b\,e^2 - b\,a^2 - f\,d\,b - f\,d\,a - f\,a^2 - f\,b\,a + e^2\,a - 2\,d\,b^2 \\ &\quad + d\,c^2 - 2\,d\,e\,b - 2\,d\,e\,c - d\,b\,a)(2\,d\,c\,a - 2\,b\,c\,e - 2\,c^2\,a + a^3 \\ &\quad - d\,a^2 - 2\,d^2\,c - d^2\,a + b^3 + 2\,a\,b\,c - 2\,c^2\,b + d^3 - 2\,a\,b^2 + 2\,d\,c\,b \\ &\quad + 2\,c\,b^2 + 2\,e\,c^2 - 2\,e\,b^2 - 2\,f\,c\,b + 2\,b\,a\,e + b^2\,f + c^2\,f + b\,e^2 - b\,a^2 \\ &\quad - f\,d\,b + f\,d\,a - f\,a^2 + f\,b\,a - e^2\,a - 2\,d\,b^2 + d\,c^2 + 2\,d\,e\,b - 2\,d\,e\,c \\ &\quad + d\,b\,a) \end{aligned}$$

```
time 27.30 words 857700
```

Clearly, the Toeplitz determinant factorization requires more than tricks from high school algebra. Indeed, the development of modern algorithms for the polynomial factorization problem is one of the great successes of the discipline of symbolic mathematical computation. Kaltofen has surveyed the algorithms until 1992 in [Kaltofen 1982, 1990, 1992], mostly from a computer science perspective. In this article we shall focus on the applications of the known fast methods to problems in science and engineering. For a more extensive set of references, please refer to Kaltofen's survey articles.

4.1 Polynomials in a single variable over a finite field

At the first glance, the problem of factoring an integer polynomial modulo a prime number appears to be very similar to the problem of factoring an integer represented in a prime radix. That is simply not so. The factorization of the polynomial $x^{511} - 1$ can be done modulo 2 on a computer in a matter of milliseconds, while the factorization of the integer $2^{511} - 1$ into its integer factors is a computational challenge. For those interested: the largest prime factors of $2^{511} - 1$ have 57 and 67 decimals digits, respectively, which makes a tough but not undoable 123 digit product for the number field sieve factorizer [Leyland November 1995]. Irreducible factors of polynomials modulo 2 are needed to construct finite fields. For example, the factor $x^9 + x^4 + 1$ of $x^{511} - 1$ leads to a model of the finite field with 2^9 elements, $\mathrm{GF}(2^9)$, by simply computing with the polynomial remainders modulo $x^9 + x^4 + 1$ as the elements. Such irreducible polynomials are used for setting up error-correcting codes, for instance, the BCH codes [MacWilliams and Sloan 1977]. Berlekamp's [1967], [1970] pioneering work on factoring polynomials over a finite field by linear algebra is done with this motivation. The linear algebra tools that Berlekamp used seem to have been introduced to the subject as early as in 1937 by Petr (cf. [Schwarz 1956]).

Today, factoring algorithms for univariate polynomials over finite fields form the innermost subalgorithm to lifting-based algorithms for factoring polynomials in one [Zassenhaus 1969] and many [Musser 1975] variables over the integers. When Maple computed the factorization of the above Toeplitz determinant, it began with factoring a univariate polynomial modulo a prime integer. The case when the prime integer is very large has lead to a significant development in computer science itself. As it turns out, by selecting random residues the expected performance of the algorithms can be speeded up exponentially [Berlekamp 1970, Rabin 1980]. Randomization is now an important tool for designing efficient algorithms and has proliferated to many fields of computer science. Paradoxically, the random elements are produced by a congruential random number generator, and the actual computer implementations are quite deterministic, which leads some computer scientists to believe that random bits can be eliminated in general at no exponential slow-down. Nonetheless, for the polynomial factoring problem modulo a large prime, no fast methods are known to-date that would work without this "probabilistic" approach.

One can measure the computing time of selected algorithms in terms of n, the degree of the input polynomial, and p, the cardinality of the field. When counting arithmetic operations modulo p (including reciprocals), the best known algorithms are quite recent. Berlekamp's 1970 method performs $O(n^\omega + n^{1+o(1)} \log p)$ residue operations. Here and below, ω denotes the exponent implied by the used linear system solver, i.e., $\omega = 3$ when classical methods are used, and $\omega = 2.376$ when asymptotically fast (though impractical) matrix multiplication is assumed. The correction term $o(1)$ accounts for the $\log n$ factors derived from the FFT-based fast polynomial multiplication and remaindering algorithms. An approach in the spirit of Berlekamp's but possibly more practical for $p = 2$ has recently been discovered by Niederreiter [1994]. A very different technique by Cantor and Zassenhaus [1981] first separates factors of different degrees and then splits the resulting polynomials of equal degree factors. It has $O(n^{2+o(1)} \log p)$ complexity and is the basis for the following two methods. Algorithms by von zur Gathen and Shoup [1992] have running time $O(n^{2+o(1)} + n^{1+o(1)} \log p)$ and those by Kaltofen and Shoup [1995] have running time $O(n^{1.815} \log p)$, the latter with fast matrix multiplication. The techniques of von zur Gathen and Shoup have recently been applied to speed the case $p = 2^k$ for large k in terms of fixed precision (bit) operations [Kaltofen and Shoup 1997].

For n and p simultaneously large, a variant of the method by Kaltofen and Shoup [1995] that uses classical linear algebra and runs in $O(n^{2.5} + n^{1+o(1)} \log p)$ residue operations is the current champion among the practical algorithms. With it Shoup, using his own fast polynomial arithmetic package [Shoup 1995], has factored a random-like polynomial of degree 2048 modulo a 2048-bit prime number in about 12 days on a Sparc-10 computer using 68 Mbyte of main memory. For even larger n, but smaller p, parallelization helps, and Kaltofen and Lobo [1994] could factor a polynomial of degree $n = 15001$ modulo $p = 127$ in about 6 days on 8 computers that are rated at 86.1 MIPS. To-date, the largest polynomial factored modulo 2 is a random polynomial of degree 262143; this was accomplished by von zur Gathen and Gerhard [1996]. They employ Cantor's fast polynomial multiplication algorithm based on additive transforms [Cantor 1989].

4.2 Polynomials in a single variable over fields of characteristic zero

As mentioned before, generally usable methods for factoring univariate polynomials over the rational numbers begin with the Hensel lifting techniques introduced by Zassenhaus [1969]. The input polynomial is first factored modulo a suitable prime integer p, and then the factorization is lifted to one modulo p^k for an exponent k of sufficient size to accommodate all possible integer coefficients that any factors of the polynomial might have. The lifting approach is fast in practice, but there are hard-to-factor polynomials on which it runs an exponential time in the degree of the input. This slowdown is due to so-called parasitic modular factors. The polynomial $x^4 + 1$, for example, factors modulo all prime integers but is irreducible over the integers: it is the cyclotomic equation for 8-th roots of unity. The products of all subsets of modular factors are candidates for integer factors, and irreducible integer polynomials with exponentially many such subsets exist [Kaltofen et al. 1983].

The elimination of the exponential bottleneck by giving a polynomial-time solution to the integer polynomial factoring problem, due to A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász [1982], is considered a major result in computer science algorithm design. The key ingredient to their solution is the construction of integer relations to real or complex numbers.

For the simple demonstration of this idea, consider the polynomial

$$x^4 + 2x^3 - 6x^2 - 4x + 8.$$

A root of this polynomial is $\alpha \approx 1.236067977$, and $\alpha^2 \approx 1.527864045$. We note that $2\alpha + \alpha^2 \approx 4.000000000$, hence x^2+2x-4 is a factor. The main difficulty is to efficiently compute the integer linear relation with relatively small coefficients for the high precision big-float approximations of the powers of a root. Lenstra *et al.* solve this diophantine optimization problem by means of their now famous lattice reduction procedure, which is somewhat reminiscent of the ellipsoid method for linear programming.

The determination of linear integer relations among a set of real or complex numbers is a useful task in science in general. Very recently, some stunning identities could be produced by this method, including the following formula for π [Bailey et al. 1995]:

$$\pi = \sum_{n=0}^{\infty} \frac{1}{16^n} \left(\frac{4}{8n+1} - \frac{2}{8n+4} - \frac{1}{8n+5} - \frac{1}{8n+6} \right).$$

Even more surprising, the lattice reduction algorithm can prove that no linear integer relation with integers smaller than a chosen parameter exists among the real or complex numbers. There is an efficient alternative to the lattice reduction algorithm, originally due to H. Ferguson and R. W. Forcade [1982] and recently improved by Ferguson and D. Bailey [1996].

The complexity of factoring an integer polynomial of degree n with coefficients of no more than l bits is thus a polynomial in n and l. From a theoretical point of view, an algorithm with a low estimate is by V. Miller [1992] and has a running time of $O(n^{5+o(1)} l^{1+o(1)} + n^{4+o(1)} l^{2+o(1)})$ bit-operations. It is expected that the relation-finding methods will become usable in practice on hard-to-factor polynomials in the near future. If the hard-to-factor input polynomial is irreducible, an alternate approach can be used to prove its irreducibility. One finds an integer evaluation point at which the integral value of the polynomial has a large prime factor, and the irreducibility follows by mathematical theorems. M. Monagan [1992] has proven large hard-to-factor polynomials irreducible in this way, which would be hopeless by the lifting algorithm.

Coefficient fields other than finite fields and the rational numbers are of interest. Computing the factorizations of univariate polynomials over the complex numbers is the root finding problem described in the section *Approximating Polynomial Zeros* before. When the coefficient field has an extra variable, such as the field of fractions of polynomials ("rational functions") the problem reduces, by an old theorem of C. F. Gauss, to factoring multivariate polynomials, which we discuss below. When the coefficient field is the field of Laurent series in t with a finite segment of negative powers,

$$\frac{c_{-k}}{t^k} + \frac{c_{-k+1}}{t^{k-1}} + \cdots + \frac{c_{-1}}{t} + c_0 + c_1 t + c_2 t^2 + \cdots, \quad \text{where } k \geq 0,$$

fast methods appeal to the theory of Puiseux series, which constitute the domain of algebraic functions [Walsh 1993].

4.3 Polynomials in two variables

Factoring bivariate polynomials by reduction to univariate factorization via homomorphic projection and subsequent lifting can be done similarly to the univariate algorithm [Musser 1975]. The second variable y takes the role of the prime integer p and $f(x,y) \bmod y = f(x,0)$. Lifting is possible only if $f(x,0)$ had no multiple root. Provided that $f(x,y)$ has no multiple factor, which can be insured by a simple GCD computation, the squarefreeness of $f(x,0)$ can be obtained by variable translation $\hat{y} = y + a$, where a is an easy-to-find constant in the coefficient field. For certain domains, such as the rational numbers, any irreducible multivariate polynomial $h(x,y)$ can be mapped to an irreducible univariate polynomial $h(x,b)$ for some constant b. This is the important *Hilbert irreducibility theorem*, whose consequence is that the combinatorial explosion observed in the univariate lifting algorithm is, in practice, unlikely. However, the magnitude and probabilistic distribution of good points b is not completely analyzed.

For so-called non-Hilbertian coefficient fields good reduction is not possible. An important such field are the complex numbers. Clearly, all $f(x,b)$ completely split into linear factors, while $f(x,y)$ may be irreducible over the complex numbers. An example for an irreducible polynomial is $f(x,y) = x^2 - y^3$. Polynomials that remain irreducible over the complex numbers are called absolutely irreducible. An additional problem is the determination of the algebraic extension of the ground field in which the absolutely irreducible factors can be expressed. In the example

$$x^6 - 2x^3y^2 + y^4 - 2x^3 = (x^3 - \sqrt{2}x - y^2) \cdot (x^3 + \sqrt{2}x - y^2),$$

the needed extension field is $\mathbb{Q}(\sqrt{2})$. The relation-finding approach proves successful for this problem. The root is computed as a Taylor series in y, and the integrality of the linear relation for the powers of the series means that the multipliers are polynomials in y of bounded degree. Several algorithms of polynomial-time complexity and pointers to the literature are found in [Kaltofen 1995].

Bivariate polynomials constitute implicit representations of algebraic curves. It is an important operation in geometric modeling to convert from implicit to parametric representation. For example, the circle

$$x^2 + y^2 - 1 = 0$$

has the rational parameterization

$$x = \frac{2t}{1+t^2}, \quad y = \frac{1-t^2}{1+t^2}, \quad \text{where } -\infty \leq t \leq \infty.$$

Algorithms are known that can find such rational parameterizations provided that they exist [Sendra and Winkler 1991]. It is crucial that the inputs to these algorithms are absolutely irreducible polynomials.

4.4 Polynomials in many variables

Polynomials in many variables, such as the symmetric Toeplitz determinant exhibited above, are rarely given explicitly, due to the fact that the number of possible terms grows exponentially in the number of variables: there can be as many as $\binom{n+v}{n} \geq 2^{\min\{n,v\}}$ terms in a polynomial of degree n with v variables. Even the factors may be dense in canonical representation, but could be sparse in another basis: for instance, the polynomial

$$(x_1 - 1)(x_2 - 2) \cdots (x_v - v) + 1$$

has only 2 terms in the "shifted basis," while it has 2^v terms in the power basis, i.e., in expanded format.

Randomized algorithms are available that can efficiently compute a factor of an implicitly given polynomial, say, a matrix determinant, and even can find a shifted basis with respect to which a factor would be sparse, provided, of course, that such a shift exists. The approach is by manipulating polynomials in so-called black box representations [Kaltofen and Trager 1990]: a black box is an object that takes as input a value for each variable, and then produces the value of the polynomial it represents at the specified point. In the Toeplitz example the representation of the determinant could be the Gaussian elimination program which computes it. We note that the size of the polynomial in this case would be nearly constant, only the variable names and the dimension need to be stored. The factorization algorithm then outputs procedures which will evaluate all irreducible factors at an arbitrary point (supplied as the input). These procedures make calls to the black box given as input to the factorization algorithm in order to evaluate them at certain points, which are derived from the point at which the procedures computing the values of the factors are probed. It is, of course, assumed that subsequent calls evaluate one and the same factor and not associates that are scalar multiples of one another. The algorithm by Kaltofen and Trager [1990] finds procedures that with a controllably high probability evaluate the factors correctly. Randomization is needed to avoid parasitic factorizations of homomorphic images which provide some static data for the factor boxes and cannot be avoided without mathematical conjecture. The procedures that evaluate the individual factors are deterministic.

Factors constructed as black box programs are much more space efficient than those represented in other formats, for example, the straight-line program format [Kaltofen 1989]. More importantly, once the black box representation for the factors is found, sparse representations can be rapidly computed by any of the new sparse interpolation algorithms. See [Grigoriev and Lakshman 1995] for the latest method allowing shifted bases and pointers to the literature of other methods, including ones for the standard power bases.

The black box representation of polynomials is normally not supported by commercial computer algebra systems such as Axiom, Maple, or Mathematica. Díaz and Kaltofen are currently developing the FoxBox system in C++ that makes black box methodology available to users of such systems [Díaz and Kaltofen 1998, Díaz 1997]. They have computed a factor over the rationals of the determinant of a generic 13 by 13 symmetric Toepitz matrix, which has 4982 non-zero terms, in 15 hours and 25 minutes on a single processor of a Sun Ultra 2 computer.

5 Research Issues and Summary

Section 3 of this article has briefly reviewed polynomial system solving based on resultant matrices as well as Gröbner bases. Both approaches are currently active, especially in applications dealing with small and medium-size systems. Handling the non-generic cases, including multiple roots and non-isolated solutiosn, is probably the most crucial issue today in relation to resultants. An interesting practical question is to delimit those cases where each of the two methods and their different variants is preferable.

Defining Terms

characteristic polynomial: a polynomial associated with a square matrix, the determinant of the matrix when a single variable is subtracted to its diagonal entries. The roots of the characteristic polynomial are the eigenvalues of the matrix.

condition number: a scalar derived from a matrix that measures its relative nearness to a singular matrix. Very close to singular means a large condition number, in which case numeric inversion becomes an ill-conditioned problem.

degree order: an order on the terms in a multivariate polynomial; for two variables x and y with $x \prec y$ the ascending chain of terms is $1 \prec x \prec y \prec x^2 \prec xy \prec y^2 \cdots$.

determinant: a polynomial in the entries of a square matrix with the property that its value is non-zero if and only if the matrix is invertible

matrix eigenvector: a column vector v such that, given square matrix A, $Av = \lambda v$, where λ is the matrix eigenvalue corresponding to v. A generalized eigenvector v is such that, given square matrices A, B, it satisfies $Av = \lambda Bv$. Both definitions extend to a row vector which premultiplies the corresponding matrix.

lexicographic order: an order on the terms in a multivariate polynomial; for two variables x and y with $x \prec y$ the ascending chain of terms is $1 \prec x \prec x^2 \prec \cdots \prec y \prec xy \prec x^2y \cdots \prec y^2 \prec xy^2 \cdots$.

ops: arithmetic operations, i.e., additions, subtractions, multiplications, or divisions; as in **flops**, i.e., floating point operations.

singularity: a square matrix is singular if there is a non-zero second matrix such the the product of the two is the zero matrix. Singular matrices do not have inverses.

sparse matrix: a matrix where many of the entries are zero.

structured matrix: a matrix where each entry can be derived by a formula depending on few parameters. For instance, the Hilbert matrix has $1/(i + j - 1)$ as the entry in row i and column j.

References

Note: many of Erich Kaltofen's publications are accessible through links in the online BibTeX bibliography database at `www.math.ncsu.edu/~kaltofen/bibliography/`.

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

E. Anderson et al. *LAPACK Users' Guide.* SIAM Publications, Philadelphia, 1992.

W. Auzinger and H.J. Stetter. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Proc. Intern. Conf. on Numerical Math., Intern. Series of Numerical Math., 86*, pages 12–30. Birkhäuser, Basel, 1988.

E. Bach and J. Shallit. *Algorithmic Number Theory Volume 1: Efficient Algorithms.* The MIT Press, Cambridge, Massachusetts, USA, 1996.

D. Bailey, P. Borwein, and S. Plouffe. On the rapid computation of various polylogarithmic constants. Preprint 95:056, Centre for Experimental and Constructive Mathematics, Simon Fraser University, `http://mosaic.cecm.sfu.ca/preprints/1995pp.html`, 1995.

E. H. Bareiss. Sylvester's identity and multistep integers preserving Gaussian elimination. *Math. Comp.*, 22:565–578, 1968.

T. Becker and V. Weispfenning. *Gröbner bases A Computational Approach to Commutative Algebra.* Springer Verlag, New York, N.Y., 1993.

E. R. Berlekamp. Factoring polynomials over finite fields. *Bell Systems Tech. J.*, 46:1853–1859, 1967. Republished in revised form in: E. R. Berlekamp, *Algebraic Coding Theory*, Chapter 6, McGraw-Hill Publ., New York, 1968.

E. R. Berlekamp. Factoring polynomials over large finite fields. *Math. Comp.*, 24:713–735, 1970.

D.N. Bernstein. The number of roots of a system of equations. *Funct. Anal. and Appl.*, 9(2): 183–185, 1975.

D. Bini and V. Y. Pan. Parallel complexity of tridiagonal symmetric eigenvalue problem. In *Proc. 2nd Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 384–393. ACM Press, New York, and SIAM Publications, Philadelphia, 1991.

D. Bini and V. Y. Pan. *Polynomial and Matrix Computations, Volume 1, Fundamental Algorithms.* Birkhäuser, Boston, 1994.

D. Bini and V. Y. Pan. *Polynomial and Matrix Computations, Volume 2.* Birkhäuser, Boston, 1998.

A. Borodin and I. Munro. *Computational Complexity of Algebraic and Numeric Problems.* American Elsevier, New York, N.Y., 1975.

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

B. Buchberger. A theoretical basis for the reduction of polynomials to canonical form. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976.

B. Buchberger. A note on the complexity of constructing Gröbner-bases. In J. A. van Hulzen, editor, *Proc. EUROCAL '83*, Springer Lec. Notes Comp. Sci., pages 137–145, 1983.

B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Recent Trends in Multidimensional Systems Theory*, pages 184–232. D. Reidel Publ. Comp., Dordrecht (Holland), 1985.

B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* Dissertation, Univ. Innsbruck, Austria, Fall 1965.

P. Bürgisser, M. Clausen, and M.A. Shokrollahi. *Algebraic Complexity Theory.* Springer, Berlin, 1997.

J. Canny. Generalized characteristic polynomials. *J. Symbolic Comput.*, 9(3):241–250, 1990.

J. Canny and I. Emiris. An efficient algorithm for the sparse mixed resultant. In G. Cohen, T. Mora, and O. Moreno, editors, *Proc. AAECC-10*, volume 673 of *Springer Lect. Notes Comput. Sci.*, pages 89–104, 1993.

J. Canny, E. Kaltofen, and Lakshman Yagati. Solving systems of non-linear polynomial equations faster. In *Proc. ACM-SIGSAM 1989 Internat. Symp. Symbolic Algebraic Comput. ISSAC '89*, pages 121–128. ACM, 1989.

J. Canny and D. Manocha. Efficient techniques for multipolynomial resultant algorithms. In S. M. Watt, editor, *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '91*, pages 85–95, New York, N.Y., 1991. ACM Press.

J. Canny and P. Pedersen. An algorithm for the Newton resultant. Technical Report 1394, Comp. Science Dept., Cornell University, 1993.

D. G. Cantor. On arithmetical algorithms over finite fields. *J. Combinatorial Theory, Series A*, 50:285–300, 1989.

D. G. Cantor and H. Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Math. Comp.*, 36:587–592, 1981.

J.-P. Cardinal and B. Mourrain. Algebraic approach of residues and applications. In J. Renegar, M. Shub, and S. Smale, editors, *The Mathematics of Numerical Analysis*, volume 32 of *Lectures in Applied Math.*, pages 189–210. AMS, Providence, R.I., U.S.A., 1996.

A. Cayley. On the theory of eliminaton. *Cambridge and Dublin Mathematical Journal*, 3: 210–270, 1865.

E. Chionh. *Base Points, Resultants and Implicit Representation of Rational Surfaces.* PhD thesis, Dept. Comput. Sci., Univ. Waterloo, 1990.

G. E. Collins. Subresultants and reduced polynomial remainder sequences. *J. ACM*, 14:128–142, 1967.

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

R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The singular value decomposition for polynomial systems. In Levelt [1995], pages 96–103.

J. J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

J. H. Davenport, E. Tournier, and Y. Siret. *Computer Algebra Systems and Algorithms for Algebraic Computation.* Academic Press, London, 1988.

J. J. W. Demmel. *Applied Numerical Linear Algebra.* SIAM Publications, Philadelphia, 1997.

A. Díaz. FoxBox *a System for Manipulating Symbolic Objects in Black Box Representation.* PhD thesis, Rensselaer Polytechnic Instit., Troy, New York, 1997.

A. Díaz and E. Kaltofen. On computing greatest common divisors with polynomials given by black boxes for their evaluation. In Levelt [1995], pages 232–239.

A. Díaz and E. Kaltofen. FoxBox a system for manipulating symbolic objects in black box representation. Manuscript, North Carolina State Univ., Dept. Math., 1998.

A. L. Dixon. The elimination of three quantics in two independent variables. In *Proc. London Mathematical Society*, volume 6, pages 468–478, 1908.

J. Dongarra et al. *LAPACK Users' Guide.* SIAM Publications, Philadelphia, 1978.

W. Eberly and E. Kaltofen. On randomized Lanczos algorithms. In Küchlin [1997], pages 176–183.

M. Elkadi and B. Mourrain. Approche effective des résidus algébriques. Technical Report 2884, INRIA, Sophia-Antipolis, France, 1996.

I. Z. Emiris and V. Y. Pan. The structure of sparse resultant matrices. In *Proc. ACM Intern. Symp. on Symbolic and Algebraic Computation*, pages 189–196, Maui, Hawaii, 1997.

I.Z. Emiris. *Sparse Elimination and Applications in Kinematics.* PhD thesis, Computer Science Division, Univ. of California at Berkeley, 1994.

I.Z. Emiris. On the complexity of sparse elimination. *J. Complexity*, 12:134–166, 1996.

I.Z. Emiris. Symbolic-numeric algebra for polynomials. In *Encyclopedia of Computer Science and Technology.* Marcel Dekker, New York, 1998. To appear.

I.Z. Emiris and J.F. Canny. Efficient incremental algorithms for the sparse resultant and the mixed volume. *J. Symbolic Computation*, 20(2):117–149, 1995.

I.Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure Applied Algebra, Special Issue on Algorithms for Algebra*, 117 & 118:229–251, 1997.

J. C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symbolic Comput.*, 16(4):329–344, 1993.

H. R. P. Ferguson and D. H. Bailey. Analysis of PSLQ, an integer relation finding algorithm. Technical Report NAS-96-005, NASA Ames Research Center, 1996.

H. R. P. Ferguson and R. W. Forcade. Multidimensional Euclidean algorithms. *J. reine angew. Math.*, 334:171–181, 1982.

G. Fiorentino and S. Serra. Multigrid methods for symmetric positive definite block toeplitz matrices with nonnegative generating functions. *SIAM J. Sci. Comput.*, 17:1068–1081, 1996.

L. V. Foster. Generalizations of Laguerre's method: higher order methods. *SIAM J. Numer. Anal.*, 18:1004–1018, 1981.

B. S. Garbow et al. *Matrix Eigensystem Routines: EISPACK Guide Extension.* Springer, New York, 1972.

J. V. Z. Gathen and V. Shoup. Computing Frobenius maps and factoring polynomials. *Comput. Complexity*, 2:187–224, 1992.

J. von zur Gathen and J. Gerhard. Arithmetic and factorization over $\mathbb{F}_2$. In Lakshman Y. N., editor, *ISSAC 96 Proc. 1996 Internat. Symp. Symbolic Algebraic Comput.*, pages 1–9, New York, N. Y., 1996. ACM Press.

K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra.* Kluwer Academic Publ., 1992.

I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, Resultants and Multidimensional Determinants.* Birkhäuser Verlag, Boston, 1994.

A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Linear Systems.* Prentice Hall, Englewood Cliffs, N.J., 1981.

P. Gianni and B. Trager. GCD's and factoring polynomials using Gröbner bases. *Proc. EUROCAL '85, Vol. 2, Springer Lec. Notes Comp. Sci.*, 204:409–410, 1985.

M. Giesbrecht. Nearly optimal algorithms for canonical matrix forms. *SIAM J. Comput.*, 24 (5):948–969, 1995.

J. R. Gilbert and R. E. Tarjan. The analysis of a nested dissection algorithm. *Numer. Math.*, 50:377–404, 1987.

G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, Maryland, third edition, 1996.

M. Gondran and M. Minoux. *Graphs and Algorithms.* Wiley–Interscience, New York, 1984.

A. Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM Publications, Philadelphia, 1997.

D. Yu. Grigoriev and Y. N. Lakshman. Algorithms for computing sparse shifts for multivariate polynomials. In A. H. M. Levelt, editor, *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC '95*, pages 96–103, New York, N. Y., 1995. ACM Press.

E. Hansen, M. Patrick, and J. Rusnak. Some modifications of Laguerre's method. *BIT*, 17: 409–417, 1977.

M. T. Heath, E. Ng, and B. W. Peyton. Parallel algorithms for sparse linear systems. *SIAM Review*, 33:420–460, 1991.

N.J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, 1996.

M. A. Jenkins and J. F. Traub. A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. *Numer. Math.*, 14:252–263, 1970.

E. Kaltofen. Polynomial factorization. In B. Buchberger, G. Collins, and R. Loos, editors, *Computer Algebra, 2nd ed.*, pages 95–113. Springer Verlag, Vienna, 1982.

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

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

E. Kaltofen. Polynomial factorization 1982-1986. In D. V. Chudnovsky and R. D. Jenks, editors, *Computers in Mathematics*, volume 125 of *Lecture Notes in Pure and Applied Mathematics*, pages 285–309. Marcel Dekker, Inc., New York, N. Y., 1990.

E. Kaltofen. Polynomial factorization 1987-1991. In I. Simon, editor, *Proc. LATIN '92*, volume 583 of *Springer Lect. Notes Comput. Sci.*, pages 294–313, 1992.

E. Kaltofen. Effective Noether irreducibility forms and applications. *J. Comput. System Sci.*, 50(2):274–295, 1995.

E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra and Applications*, 136:189–208, 1990.

E. Kaltofen and A. Lobo. Factoring high-degree polynomials by the black box Berlekamp algorithm. In J. von zur Gathen and M. Giesbrecht, editors, *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, pages 90–98, New York, N. Y., 1994. ACM Press.

E. Kaltofen, D. R. Musser, and B. D. Saunders. A generalized class of polynomials that are hard to factor. *SIAM J. Comp.*, 12(3):473–485, 1983.

E. Kaltofen and V. Pan. Processor efficient parallel solution of linear systems over an abstract field. In *Proc. 3rd Ann. ACM Symp. Parallel Algor. Architecture*, pages 180–191, New York, N.Y., 1991. ACM Press.

E. Kaltofen and V. Pan. Processor-efficient parallel solution of linear systems II: the positive characteristic and singular cases. In *Proc. 33rd Annual Symp. Foundations of Comp. Sci.*, pages 714–723, Los Alamitos, California, 1992. IEEE Computer Society Press.

E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. In *Proc. 27th Annual ACM Symp. Theory Comp.*, pages 398–406, New York, N.Y., 1995. ACM Press. *Math. Comput.*, in press.

E. Kaltofen and V. Shoup. Fast polynomial factorization over high algebraic extensions of finite fields. In Küchlin [1997], pages 184–188.

E. Kaltofen and B. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symbolic Computation*, 9(3):301–320, 1990.

D. Kapur. Geometry theorem proving using Hilbert's Nullstellensatz. *J. Symbolic Comp.*, 2: 399–408, 1986.

D. Kapur and Lakshman Y. N. Elimination methods an introduction. In B. Donald, D. Kapur, and J. Mundy, editors, *Symbolic and Numerical Computation for Artificial Intelligence.* Academic Press, 1992.

D. Kapur and T. Saxena. Comparison of various multivariate resultant formulations. In A. H. M. Levelt, editor, *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '95*, pages 187–195, New York, N.Y., 1995. ACM Press.

D. Kapur, T. Saxena, and L. Yang. Algebraic and geometric reasoning using Dixon resultants. In J. von zur Gathen and M. Giesbrecht, editors, *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, pages 99–107, New York, N. Y., 1994. ACM Press.

D. E. Knuth. *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms.* Addison Wesley, Reading, MA, second edition, 1981.

S. Krishnan and D. Manocha. Numeric-symbolic algorithms for evaluating one-dimensional algebraic sets. In *Proc. ACM Intern. Symp. on Symbolic and Algebraic Computation*, pages 59–67, 1995.

W. Küchlin, editor. *ISSAC 97 Proc. 1997 Internat. Symp. Symbolic Algebraic Comput.*, New York, N. Y., 1997. ACM Press.

Y. N. Lakshman and B. D. Saunders. On computing sparse shifts for univariate polynomials. In J. von zur Gathen and M. Giesbrecht, editors, *Proc. Internat. Symp. Symbolic Algebraic Comput. ISSAC '94*, pages 108–113, New York, N. Y., 1994. ACM Press.

Y. N. Lakshman and B. D. Saunders. Sparse polynomial interpolation in non-standard bases. *SIAM J. Comput.*, 24(2):387–397, 1995.

Y.N. Lakshman. *On the complexity of computing Gröbner bases for zero-dimensional polynomial ideals.* PhD thesis, Comp. Science Dept., Rensselaer Polytechnic Institute, Troy, New York, 1990.

D. Lazard. Resolution des systemes d'equation algebriques. *Theoretical Comput. Sci.*, 15: 77–110, 1981. In French.

A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:515–534, 1982.

A. H. M. Levelt, editor. *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95*, New York, N. Y., 1995. ACM Press.

P. Leyland. Cunningham project data. Internet document, Oxford Univ., `ftp://sable.ox.ac.uk/pub/math/cunningham/`, November 1995.

R. J. Lipton, D. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numer. Analysis*, 16(2):346–358, 1979.

F. S. Macaulay. Algebraic theory of modular systems. Cambridge Tracts 19, Cambridge, 1916.

F. J. MacWilliams and N. J. A. Sloan. *The Theory of Error-Correcting Codes.* North-Holland Publ. Comp., New York, 1977.

K. Madsen. A root-finding algorithm based on Newton's method. *BIT*, 13:71–75, 1973.

D. Manocha. *Algebraic and Numeric Techniques for Modeling and Robotics.* PhD thesis, Comp. Science Div., Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley, 1992.

D. Manocha, Y. Zhu, and W. Wright. Conformational analysis of molecular chains using nano-kinematics. *Computer Applications of Biological Sciences*, 11(1):71–86, 1995.

S. McCormick, editor. *Multigrid Methods.* SIAM Publications, Phiadelphia, 1987.

J. M. McNamee. A bibliography on roots of polynomials. *J. Comput. Applied Math.*, 47(3): 391–394, 1993.

V. Miller. Factoring polynomials via relation-finding. In D. Dolev, Z. Galil, and M. Rodeh, editors, *Proc. ISTCS '92*, volume 601 of *Springer Lect. Notes Comput. Sci.*, pages 115–121, 1992.

M. B. Monagan. A heuristic irreducibility test for univariate polynomials. *J. Symbolic Comput.*, 13(1):47–57, 1992.

B. Mourrain and V. Y. Pan. Solving special polynomial systems by using structured matrices and algebraic residues. In F. Cucker and M. Shub, editors, *Proc. Workshop on Foundations of Computational Mathematics*, pages 287–304, Berlin, 1997. Springer.

D. R. Musser. Multivariate polynomial factorization. *J. ACM*, 22:291–308, 1975.

H. Niederreiter. New deterministic factorization algorithms for polynomials over finite fields. In L. Mullen and P. J.-S. Shiue, editors, *Finite Fields: Theory, Applications and Algorithms*, volume 168 of *Contemporary Mathematics*, pages 251–268, Providence, Rhode Island, 1994. Amer. Math. Soc.

J. M. Ortega and R. G. Voight. Solution of partial differential equations on vector and parallel computers. *SIAM Review*, 27(2):149–240, 1985.

V. Y. Pan. How can we speed up matrix multiplication? *SIAM Rev.*, 26(3):393–415, 1984a.

V. Y. Pan. *How to Multiply Matrices Faster*, volume 179 of *Lecture Notes in Computer Science.* Springer Verlag, Berlin, 1984b.

V. Y. Pan. Sequential and parallel complexity of approximate evaluation of polynomial zeros. *Computers in Mathematics (with Applications)*, 14(8):591–622, 1987.

V. Y. Pan. Complexity of algorithms for linear systems of equations,. In E. Spedicato, editor, *Computer Algorithms for Solving Linear Algebraic Equations (State of the Art)*, volume 77 of *NATO ASI Series,* Series F: Computer and Systems Sciences, pages 27–56, Berlin, 1991. Springer.

V. Y. Pan. Complexity of computations with matrices and polynomials. *SIAM Review*, 34(2): 225–262, 1992a.

V. Y. Pan. Linear systems of algebraic equations. In Marvin Yelles, editor, *Encyclopedia of Physical Sciences and Technology*, volume 8, pages 779–804. 2 edition, 1992b. Volume 7, pages 304–329, 1987 (first edition).

V. Y. Pan. Parallel solution of sparse linear and path systems. In J. H. Reif, editor, *Synthesis of Parallel Algorithms*, chapter 14, pages 621–678. Morgan Kaufmann Publ., San Mateo, CA, 1993.

V. Y. Pan. Parallel computation of a Krylov matrix for a sparse and structured input. *Mathematical and Computer Modelling*, 21(11):97–99, 1995. Proceedings version: 27th Ann. ACM STOC, pages 741–750, ACM Press, New York, N.Y., U.S.A., 1995.

V. Y. Pan. On approximating complex polynomial zeros: Modified quadtree (Weyl's) construction and improved Newton's iteration. Technical Report 2894, INRIA, Sophia-Antipolis, France, 1996a. Revised 1997.

V. Y. Pan. Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers in Mathematics (with Applications)*, 31(12):97–138, 1996b.

V. Y. Pan. Parallel computation of polynomial GCD and some related parallel computations over abstract fields. *Theor. Comp. Science*, 162(2):173–223, 1996c.

V. Y. Pan. Faster solution of the key equation for decoding the BCH error-correcting codes. In *Proc. ACM Symp. Theory of Comp.*, pages 168–175. ACM Press, 1997a.

V. Y. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Review*, 39(2):187–220, 1997b.

V. Y. Pan and F. P. Preparata. Work-preserving speed-up of parallel matrix computations. *SIAM J. Comput.*, 24(4):811–821, 1995.

V. Y. Pan and J. H. Reif. Compact multigrid. *SIAM J. on Scientific and Statistical Computing*, 13(1):119–127, 1992.

V. Y. Pan and J. H. Reif. Fast and efficient parallel solution of sparse linear systems. *SIAM J. Comp.*, 22(6):1227–1250, 1993.

V. Y. Pan, I. Sobze, and A. Atinkpahoun. On parallel computations with band matrices. *Information and Computation*, 120(2):227–250, 1995.

B. Parlett. *Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, N.J., 1980.

M. J. Quinn. *Parallel Computing: Theory and Practice.* McGraw-Hill, Inc., New York, N.Y., 1994.

M. O. Rabin. Probabilistic algorithms in finite fields. *SIAM J. Comp.*, 9:273–280, 1980.

J. Renegar. On the worst case arithmetic complexity of approximating zeros of polynomials. *J. Complexity*, 3(2):90–113, 1987.

J. F. Ritt. *Differential Algebra.* AMS, New York, 1950.

Št. Schwarz. On the reducibility of polynomials over a finite field. *Quart. J. Math. Oxford Ser. (2)*, 7:110–124, 1956.

J. R. Sendra and F. Winkler. Symbolic parameterization of curves. *J. Symbolic Comput.*, 12 (6):607–631, 1991.

V. Shoup. A new polynomial factorization algorithm and its implementation. *J. Symbolic Comput.*, 20(4):363–397, 1995.

B. T. Smith et al. *Matrix Eigensystem Routines: EISPACK Guide, 2nd ed.* Springer, New York, 1970.

T. Stederberg and R. Goldman. Algebraic geometry for computer-aided design. *IEEE Computer Graphics and Applications*, 6(6):52–59, 1986.

B. Sturmfels. Sparse elimination theory. In D. Eisenbud and L. Robbiano, editors, *Proc. Computat. Algebraic Geom. and Commut. Algebra*, Cortona, Italy, 1991.

L. N. Trefethen and D. Bau III. *Numerical Linear Algebra.* SIAM Publications, Philadelphia, 1997.

B.L. van der Waerden. *Modern Algebra.* F. Ungar Publishing Co., New York, 3rd edition, 1950.

P. G. Walsh. The computation of puiseux expansions and a quantitative version of runge's theorem on diophantine equations. Ph.d. thesis, Univ. Waterloo, Waterloo, Candad, 1993.

J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Clarendon Press, Oxford, England, 1965.

F. Winkler. *Polynomial Algorithms in Computer Algebra.* Springer, Wien, 1996.

W. Wu. Basis principles of mechanical theorem proving in elementary geometries. *J. Syst. Sci. and Math Sci.*, 4(3):207–235, 1984.

H. Zassenhaus. On Hensel factorization I. *J. Number Theory*, 1:291–311, 1969.

R. Zippel. *Effective Polynomial Computations*, page 384. Kluwer Academic Publ., Boston, Massachusetts, 1993.

6 Further Information

The books [Aho et al. 1974, Borodin and Munro 1975, Knuth 1981, Davenport et al. 1988, Geddes et al. 1992, Zippel 1993, Bini and Pan 1994, Winkler 1996, Bach and Shallit 1996, Bürgisser et al. 1997, Bini and Pan 1998] provide a much broader introduction to the general subject and further bibliography. There are well known libraries and packages of subroutines for the most popular numerical matrix computations, in particular, [Dongarra et al. 1978] for solving linear systems of equations, [Smith et al. 1970] and [Garbow et al. 1972] for approximating matrix eigenvalues, and [Anderson et al. 1992] for both of the two latter computational problems. There is a comprehensive treatment of numerical matrix computations [Golub and Van Loan 1996], with extensive bibliography, and there are several more specialized books on them [George and Liu 1981, Wilkinson 1965, Parlett 1980, Higham 1996, Trefethen and Bau III 1997] as well as many survey articles [Heath et al. 1991, Ortega and Voight 1985, Pan 1992b] and thousands of research articles. Two journals are currently dedicated to the subject, the *Journal of Symbolic Computation* by Academic Press, London and *Applicable Algebra in Engineering, Communication and Computing* by Springer Verlag. The annual *International Symposium on Symbolic and Algebraic Computation* is what we consider the flagship conference of the research community.

Special (more efficient) parallel algorithms have been devised for special classes of matrices, such as sparse [Pan and Reif 1993, Pan 1993], banded [Pan et al. 1995], and dense structured [Bini and Pan 1994, Pan 1996c]. We also refer to [Pan and Preparata 1995] on simple but surprisingly effective extension of Brent's principle for improving the processor and work efficiency of parallel matrix algorithms (with applications to path computations in graphs) and to [Golub and Van Loan 1996, Ortega and Voight 1985, Heath et al. 1991] on practical parallel algorithms for matrix computations.