ALGEBRAIC AND NUMERICAL ALGORITHMS[1]

Ioannis Z. Emiris
National Kapodistrian University of Athens, Athens 15784, Greece
emiris@di.uoa.gr

Victor Y. Pan, Mathematics and Computer Science Department, Lehman College, City University of New York, Bronx, NY 10468, USA. vpan@lehman.cuny.edu. http://comet.lehman.cuny.edu/vpan/

Elias P. Tsigaridas
INRIA - LORIA Lorraine
615, rue du Jardin Botanique, B.P. 101, 54602 Villers-des-Nancy cedex, France.
elias.tsigaridas@loria.fr

1 Introduction

Arithmetic manipulation with matrices and polynomials is a common subject for algebraic (or symbolic) and numerical computing. Typical computational problems in these areas include the solution of a polynomial equation and linear and polynomial systems of equations, univariate and multivariate polynomial evaluation, interpolation, factorization and decompositions, rational interpolation, computing matrix factorization and decompositions, including various triangular and orthogonal factorizations such as LU, PLU, QR, QRP, QLP, CS, LR, Cholesky factorizations and eigenvalue and singular value decompositions, computation of the matrix inverses, determinants, Smith and Frobenius normal forms, ranks, characteristic and minimal polynomials, univariate and multivariate polynomial resultants, Newton's polytopes, and greatest common divisors and least common multiples as well as manipulation with truncated series and algebraic sets.

Such problems can be solved based on the error-free algebraic (symbolic) computations with infinite precision. This demanding task is achieved in the present day advanced computer library GMP and computer algebra systems such as Maple and Mathematica by employing various nontrivial computational techniques such as the Euclidean algorithm and continuous fraction approximation, Hensel's and Newton's lifting, Chinese Remainder algorithm, elimination and resultant methods, and Gröbner bases computation. The price for the achieved accuracy is the increase of the memory space and computer time supporting the computations.

[0]This material is based on work supported in part by IST Programme of the European Union as a Shared-cost RTD (FET Open) Project under Contract No IST-006413-2 (ACS - Algorithms for Complex Shapes) (first and third authors) and by NSF Grant CCR 9732206 and PSC CUNY Awards 67297-0036 and 68291–0037 (second author).

An alternative numerical approach relies on operations with binary numbers truncated or rounded to a fixed precision. Operating with the IEEE standard floating point numbers represented with double precision enables much faster computations that use much less memory space but requires theoretical and/or experimental study of the affect of the rounding errors on the output. The study uses various advanced techniques from approximation and perturbation theories, forward and backward error analysis, operator theory and numerical linear algebra. If necessary, more costly computations with the extended precision are used to yield uncorrupted output. The resulting algorithms are combined in the high performance libraries and packages of subroutines such as Matlab, NAG SMP, LAPACK, ScaLAPACK, ARPACK, PARPACK, and MPSolve.

Combining algebraic and numerical methods frequently increases their power and enables more effective computations. In this chapter we cover some algebraic and numerical algorithms in the large, popular and highly important areas of matrix computations and root-finding for univariate polynomials and systems of multivariate polynomials. We give some pointers to the bibliography on these and adjacent subjects and in Section 5 to further references on algebraic and numerical algorithms. The bibliography is huge, and we usually cite books, surveys, and comprehensive articles with pointer to further references, rather than the original technical articles. Our expositions in Sections 1 and 2 largely follow the line of the first surveys in this area in (203; 204; 209; 210).

We state the complexity bounds under the random access machine (RAM) model of computation (2). In most cases we assume the *arithmetic* model, that is, we assign a unit cost to addition, subtraction, multiplication, and division of real numbers, as well as to reading or writing them into a memory location. This model is realistic for computations with a fixed (e.g., the IEEE standard double) precision, which fits the size of a computer word. In this case the arithmetic model turns into the *word* model (115). In other cases we compute with the extended precision and assume the *Boolean* or *bit* model, assigning the unit cost to every Boolean or bit operation. This accounts for both arithmetic operations and the length (precision) of the operands. We denote the bounds on this complexity by $\mathcal{O}_B(\cdot)$. We always specify whether we use the arithmetic, word, or Boolean model unless this is clear from the context.

We write **ops** for "arithmetic operations", "section.name" for "Section section.name", and "log" for "$\log_2$" unless specified otherwise.

2 Matrix Computations

Matrix computations is the most popular and highly important area of scientific and engineering computing. Most frequently they are performed numerically, with rounding-off or chopping the input to the IEEE standard double precision. This is mostly assumed in the present section unless specified otherwise.

In the chapter of this size we must omit or just barely touch many important subjects of matrix computations. The reader can find further material and bibliography in the surveys (199; 203; 204) and the books (11; 14; 27; 29; 73; 79; 82; 126; 128; 135; 213; 240; 250; 258; 259; 266; 277). For more specific subject areas we further refer the reader to (11; 73; 126; 240; 259; 266; 277) on the eigendecompositions and SVDs, (14; 73; 79; 126; 135; 258; 266) on other numerical matrix factorizations, (30; 165) on the over- and under-determined linear systems, their least-squares solution, and various other numerical computations with singular matrices, (27; 126; 241) on parallel matrix algorithms, and to (57; 64; 84; 85; 115; 118; 122; 127; 205; 230; 216; 218; 226; 264; 265; 276) on "Error-free Rational Matrix Computations", including computations in finite fields, rings, and semirings that output the solutions to linear systems of equations, matrix inverses, ranks, determinants, characteristic and minimum polynomials, and Smith and Frobenius normal forms.

2.1 Dense, Sparse and Structured Matrices. Their Storage and Multiplication by Vectors

An $m \times n$ matrix $A = [\ a_{i,j}\ ,\ i = 0, 1, \dots, m-1;\ j = 0, 1, \dots, n-1\ ]$, also denoted $[a_{i,j}]_{i,j=0}^{m-1,n-1}$ and $[\mathbf{A}_0, \dots, \mathbf{A}_{m-1}]$, is a 2-dimensional array, with the (i, j)th entry $[A]_{i,j} = a_{i,j}$ and the jth column $\mathbf{A}_j$. A^T is the transpose of A. A is a column vector $\mathbf{A}_0$ of dimension m if $n = 1$. A is a row vector of dimension n if $m = 1$. We write $\mathbf{v} = [v_i]_{i=0}^{n-1}$ to denote an nth dimensional column vector and $\mathbf{w} = A\mathbf{v} = [w_i]_{i=0}^{m-1}, w_i = \sum_{j=0}^{n-1} a_{i,j} v_j, i = 0, \dots, m-1$, to denote the matrix-by-vector product. The straightforward algorithm computes such a product by using $(2n-1)m$ ops. This is the sharp bound for general (that is, dense unstructured) $m \times n$ matrix, represented with its entries. In actual computations, however, matrices are most frequently special and instead of mn entries can be represented with much fewer parameters.

An $m \times n$ matrix is *sparse* if it is filled mostly with zeros, that is, if it has only $\phi << mn$ nonzero entries. An important example is banded matrices $[b_{i,j}]_{i,j}$, whose all nonzero entries lie near the diagonal, so that $b_{i,j} = 0$ unless $|i - j| \leq w$ for a small *bandwidth* $2w + 1$. This class is generalized to sparse matrices associated with graphs that have *families of small separators* (120; 125; 171). A sparse matrix can be stored economically by using appropriate data structures and can be multiplied by a vector fast, theoretically in $2\phi - m$ ops. Sparse matrices arise in many important applications, in particular, to solving ordinary and partial differential equations (ODEs and PDEs).

Dense structured $n \times n$ matrices can be defined by $O(n)$ parameters and can be multiplied by a vector by using $O(n \log n)$ or $O(n \log^2 n)$ ops. Such matrices are omnipresent in computations in signal and image processing, coding, ODEs, PDEs, integral equations, particle simulation, and Markov chains. Most popular are *Toeplitz matrices* $T = [t_{i,j}]_{i,j=0}^{m,n}$, $t_{i,j} = t_{i+1,j+1}$ for all i and j. Such a matrix is defined by $m + n - 1$ entries of its first row and first column. Toeplitz-by-vector product $T\ \mathbf{v}$ is defined by "Vector Convolution" (see Chapter 17). It can be computed by using $O((m + n) \log(m + n))$ ops. Close ties between the computations with Toeplitz matrices and polynomials enable acceleration in both areas.

Similar properties of the *Hankel*, *Bézout, Sylvester, Frobenius (companion), Vandermonde,* and *Cauchy* matrices can be extended to more general classes of structured matrices via associating linear displacement operators. (See (27; 213) and Chapter 17 for the details and the bibliography.) Finally, dense structured *semiseparable* matrices generalize banded matrices, are expressed via $O(n)$ parameters and multiplied by vectors in $O(n)$ ops (269).

2.2 Matrix Multiplication and Some Extensions

The straightforward algorithm computes the $m \times p$ product AB of $m \times n$ by $n \times p$ matrices 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 by Strassen 1969 and Winograd 1971 using $O(n^{2.81})$ ops (126; 135). The algorithms of Coppersmith and Winograd in (65) use at most Cn^{ω} ops for $\omega < 2.376$ and a huge constant C such that $Cn^{\omega} < 2n^3$ only for extremely large values n. Coppersmith and Winograd in (65) combine their technique of arithmetic progression with various previous advanced techniques. Each of these techniques alone contributes a dramatic increase of the overhead constant that makes the resulting algorithms practically noncompetitive. The only exception is the technique of trilinear aggregating that alone supports the exponent 2.7753 (see (160; 199)). The recent practical numerical algorithms in (149) rely on this technique. For matrices of reasonable sizes they use about as many ops as the Strassen's and Winograd's algorithms but need less memory space and are more stable numerically.

One can multiply a pair of $n \times n$ structured matrices in nearly linear arithmetic time, namely, by using $O(n \log n)$ or $O(n \log^2 n)$ ops, where both input and output matrices are represented via their short generator matrices having $O(n)$ entries (see (27; 213) or "Structured Matrices" in Chapter 17).

If the input values are reasonably bounded integers, then matrix multiplication (as well as vector convolution in Chapter 17) can be reduced to a single multiplication of two longer integers, by means of the techniques of *binary segmentation* (cf. (200, Sect. 40); (203), or (27, Examples 3.9.1–3.9.3)). The Boolean cost of the computations does not decrease, but the techniques can be practically useful where the two longer integers still fit the computer precision.

Many fundamental matrix computations can be reduced to $O(\log n)$ or a constant number of $n \times n$ matrix multiplications (27, Chapter 2). This includes the evaluation of det A, the **determinant** of an $n \times n$ matrix A; its *inverse* A^{-1} (where det $A \neq 0$); the coefficients of its **characteristic polynomial** $c_A(x) = \det(\ xI - A\ )$ and *minimal polynomial* $m_A(x)$, for a scalar variable x; the Smith and Frobenius normal forms; the *rank,* rank A; the solution vector $\mathbf{x} = A^{-1}\ \mathbf{v}$ to a nonsingular *linear system of equations* $A\ \mathbf{x} = \mathbf{v}$; various *orthogonal* and *triangular factorizations* of the matrix 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 are known for some apparently distant combinatorial and graph computations such as computing the transitive closure of a graph (2), computing all pair shortest distances in graphs (27, p. 222), and pattern recognition. Consequently, all these operations use $O(n^\omega)$ ops where theoretically $\omega < 2.376$ (2, chap.6), (27, chap. 2).

In practice, however, due to the overhead constants hidden in the "O" notation for $\omega < 2.775$ for matrix multiplication, additional overhead for its extensions, the memory space requirements, and numerical stability problems, all these extensions of matrix multiplication use the order of n^3 ops (126). Nevertheless, the reduction to matrix multiplication is practically important because it allows to employ *block matrix algorithms*. Although they use the order of n^3 ops, they are performed on multiprocessors much faster than the straightforward algorithms (126; 241).

Let us conclude this subsection by demonstrating two basic techniques for the extension of matrix multiplication. Hereafter we denote by 0 the null matrices (filled with zeros) and by I the identity (square) matrices (which have ones on their diagonals and zeros elsewhere).

One of the basic ideas is to represent the input matrix A as a block matrix and to operate with its blocks (rather than with its entries). For example, compute $\det A$ and A^{-1} by first factorizing A as a 2×2 block matrix,

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

where $S = A_{1,1} - A_{1,0}A_{0,0}^{-1}A_{0,1}$. Note that the 2×2 block factors are readily invertible, $\det A = (\det A_{0,0}) \det S$ and $(BCD)^{-1} = D^{-1}C^{-1}B^{-1}$, so that the original problems for the input A are reduced to the same problems for the half-size matrices $A_{0,0}$ and S. It remains to factorize them recursively. The northwestern blocks (such as $A_{0,0}$), called leading principal submatrices, must be nonsingular throughout the recursive process, but this property holds for the large and highly important class of *positive definite* matrices $A = C^TC$, $\det C \neq 0$, and can be always achieved by means of symmetrization, pivoting, or randomization (2, chap. 6), (27, chap. 2), (213, sects. 5.5 and 5.6)).

Another basic technique is the computation of the *Krylov sequence* or *Krylov matrix* $[B^i\mathbf{v}]_{i=0}^{k-1}$ for an $n \times n$ matrix B and an n-dimensional vector $\mathbf{v}$ (126; 128; 250). The straightforward algorithm uses $(2n-1)n(k-1)$ ops, which is about $2n^3$ for $k = n$. An alternative algorithm first computes the matrix powers

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

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

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

The last step completes the evaluation of the Krylov sequence in $2s + 1$ matrix multiplications, by using $O(n^\omega \log k)$ ops overall.

Special techniques for parallel computation of Krylov sequences for sparse and/or structured matrices A can be found in (206). According to these techniques, Krylov sequence is recovered from the solution of the associated linear system $(I - A)\,\mathbf{x} = \mathbf{v}$, which is solved fast in the case of a special matrix A.

In the next two subsections, we more closely consider the solution of a linear system of equations, $A\,\mathbf{x} = \mathbf{b}$, which is the most frequent operation in practice of scientific and engineering computing and is highly important theoretically.

2.3 Solution of linear systems of equations

General nonsingular linear system of n equations $A\,\mathbf{x} = \mathbf{b}$ can be solved in $(2/3)n^3) + O(n^2)$ ops by means of Gaussian elimination. One can perform it numerically and (in spite of rounding errors) arrive at an uncorrupted output by applying pivoting, that is, appropriate interchange of the equations (and sometimes also unknowns) to avoid divisions by absolutely smaller numbers. A by-product is factorization $A = PLU$ (or $A = PLUP'$), for lower triangular matrices L and U^T and permutation matrices P (and P').

For sparse and positive definite linear systems, pivoting can be modified to preserve sparseness during the elimination and thus to yield faster solution (79; 82; 120; 123; 124; 171; 205; 230). Gaussian elimination with the *(generalized) nested dissection* policy of pivoting requires only $O(s(n)^3)$ ops to solve a sparse positive definite linear system of n equations whose associated graph has a family of separators of diameter $s(n)$. $s(n) = O(\sqrt{n})$ for a large and important class of sparse linear systems arising from discretization of ODEs and PDEs. For general sparse linear systems $s(n)$ can be as large as n, and we also have no formal proof for any better uppers bounds than $O(n^3)$ for Gaussian elimination under any other policy of pivoting. Some heuristic policies (such as *Markowitz rule*), however, substantially accelerate sparse Gaussian elimination according to ample empirical evidence.

Both Gaussian elimination and the *(Block) Cyclic Reduction* algorithm use $O(nw^2)$ ops for banded linear systems with bandwidth $O(w)$. This is $O(n)$ where the bandwidth is constant, and similarly for the (dense) semiseparable (rank structured) matrices (269).

Likewise, we can dramatically accelerate Gaussian elimination for dense structured input matrices represented with their short generators, defined by the associated *displacement operators*. This includes Toeplitz, Hankel, Vandermonde, and Cauchy matrices and matrices with similar structures. By applying the recursive 2×2 block factorization in the previous subsection (with proper care about preserving matrix structure in the recursive process), we arrive at the MBA divide-and-conquer algorithm (due to Morf 1974/1980 and Bitmead and Anderson 1980) that solves nonsingular structured linear systems of n equations in $O(n \log^2 n)$ ops (see (27; 213)), although this computation is prone to numerical stability problems unless the input matrix is positive definite.

For indefinite nonsingular Cauchy-like and Vandermonde-like linear systems of n equations, pivoting preserves matrix structure, and Gaussian elimination can be performed by using $O(n^2)$ ops in numerically stable algorithms. The latter property is also true for linear systems with the Toeplitz/Hankel structures. Pivoting destroys their structure, but their solution can be reduced to Cauchy/Vandermonde-like systems by means of "Displacement Transformation" (see Chapter 17).

A popular alternative to Gaussian elimination is the iterative solution algorithms such as the Conjugate Gradient and GMRES algorithms (17; 56; 126; 128; 250; 271). They compute sufficiently long Krylov sequences (defined in the previous section), approximate the solution with linear combinations $\sum_i c_i B^i \mathbf{b}$ for appropriate coefficients c_i, and stop where the solution is approximated within a desired tolerance to the output errors. Typically, the algorithms perform every iteration step at the cost of multiplying the input matrix and its transpose by two vectors. This cost is small for structured and sparse matrices. (We can even call a matrix sparse and/or structured if and only if it can be multiplied by a vector fast.)

The *multilevel methods* (108; 177; 229) are even more effective for some special classes of linear systems arising in discretization of ODEs and PDEs. In the underlying algebraic process (called the *algebraic multigrid*) one first aggregates an input linear system, then solves the resulting smaller system, and finally disaggregates the solution into the solution of the original system (186). The power of this technique is accentuated in its recursive multilevel application.

Generally, iterative methods are highly effective for a sparse and/or structured linear systems (and become the methods of choice) as long as they converge fast. Special techniques of *preconditioning* of the input matrices at a low computational cost enable faster convergence of iterative algorithms for many important special classes of sparse and structured linear systems (17; 56; 128), and more recently, for quite a general class of linear systems (222).

Even with all known preconditioning techniques we cannot deduce competitive upper bounds on the worst case complexity of iterative solution unless we can readily approximate the inverse M^{-1} of the input matrix M. An approximation X_0 serves well as long as the norm ν of the residual matrix $I - MX_0$ is noticeably less than one. Indeed, in this case we can rapidly refine the initial approximation, e.g., with Newton's iteration, $X_{i+1} = 2X_i - X_iMX_i$, for which we have $I - MX_{i+1} = (I - MX_i)^2 = (I - MX_0)^{2^{i+1}}$ and, therefore, $||I - MX_{i+1}|| \leq \nu^{2^{i+1}}$ for $i = 0, 1, \ldots$. See more on Newton's iteration in (224; 232) and the references therein.

A Newton iteration step uses two matrix multiplications. This is relatively costly for general matrices but takes nearly linear time in n for $n \times n$ structured matrices represented with their short displacement generators (see Chapter 17). The multiplications gradually destroy matrix structure, but some advanced techniques in (213, chapters 4 and 6), (221; 228; 234; 235) counter this problem.

2.4 Error-free Rational Matrix Computations

Rational matrix computations for a rational or integer input (such as the solution of a linear system and computing the determinant) can be performed with no errors. To decrease the computational cost, one should control the growth of the precision of computing. We refer the reader to (13) and (118) on some special techniques that achieve this in rational Gaussian elimination. A more fundamental tool of symbolic (algebraic) computing is the reduction of the computations modulo one or several fixed primes or prime powers. Based on such a reduction, the rational or integer output values $z = p/q$ (e.g., the solution vector for a linear system) can be computed modulo a sufficiently large integer m. Then the desired rational values z are recovered from the values $z \bmod m$ by means of the continued fraction approximation algorithm, which is the Euclidean algorithm applied to integers (115; 275), in our case to the integers m and $z \bmod m$. If the output z is known to be an integer lying between $-r$ and r and if $m > 2r$ then the integer z is readily recovered from $z \bmod m$ as follows:

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

For example, if we compute integer determinant, we can choose the modulus m based on the Hadamard's bound. The reduction modulo a prime p can turn a nonsingular matrix A and a nonsingular linear system $A\mathbf{x} = \mathbf{v}$ into singular ones, but this can occur only with a low probability for a random choice of the prime p in a fixed sufficiently large interval as well as, say, for a reasonably large power of two and a random integer matrix (226).

The precision of $\log m$ bits for computing the integer $z \bmod m$ can be excessively large for a large m, but one can first compute this integer modulo k smaller relatively prime integers $m_1, m_2, \ldots, m_k$ (we call them *coprimes*) such that $m_1 m_2 \cdots m_k = m$, and then one can apply the Chinese Remainder algorithm. The error-free computations modulo m_i require the smaller precision of $\log m_i$ bits, whereas the computational cost of the subsequent recovery of the value $z \bmod m$ is dominated by the cost of computing the values $z \bmod m_i$ for $i = 1, \ldots, k$.

For matrix and polynomial computations, there are effective alternative techniques of *p-adic* (*Newton–Hensel*) *lifting*, (115). Moenck and Carter 1979 and Dixon 1982 have elaborated upon them for solving linear systems of equations and matrix inversion, thus creating symbolic counterparts to well known numerical techniques of Newton's iteration and iterative refinement in linear algebra.

Newton's lifting begins with a prime p, a larger integer k, an integer matrix M, and its inverse $Q = M^{-1} \bmod p$, such that $I - QM \bmod p = 0$. Then one writes $X_0 = Q$, recursively computes the matrices $X_j = 2X_{j-1} - X_{j-1}MX_{j-1} \bmod (p^{2^j})$ observing that $I - X_j M = 0 \bmod (p^{2^j})$ for $j = 1, 2, \ldots, k$, and finally recovers the inverse matrix M^{-1} from $X_k = M^{-1} \bmod p^{2^k}$.

Hensel's lifting begins with the same input complemented with an integer vector $\mathbf{b}$. Then one writes $\mathbf{r}^{(0)} = \mathbf{b}$, recursively computes the vectors

$$\mathbf{u}^{(i)} = Q\mathbf{r}^{(i)} \bmod p, \quad \mathbf{r}^{(i+1)} = (\mathbf{r}^{(i)} - M\mathbf{u}^{(i)})/p, \quad i = 0, 1, \ldots, k-1,$$

and $\mathbf{x}^{(k)} = \sum_{i=0}^{k-1} \mathbf{u}^{(i)} p^i$ such that $M\mathbf{x}^{(k)} = \mathbf{b} \bmod (p^k)$, and finally recovers the solution $\mathbf{x}$ to the linear system $M\mathbf{x} = \mathbf{b}$ from the vector $\mathbf{x}^{(k)} = \mathbf{x} \bmod (p^k)$.

Newton's and Hensel's lifting are particularly powerful where the input matrices M and M^{-1} are sparse and/or structured. Then a lifting step takes $O(n)$ ops up to a polylog factor. This includes, e.g., Toeplitz, Hankel, Vandermonde, Cauchy, banded and semiseparable matrices, and the matrices whose associated graphs have small separators families. Newton's lifting uses fewer steps, but recursively doubles the precision of computing. Hensel's lifting is performed with the precision in $O(\log p)$ and, as proved in (216; 226), enables the solution in nearly optimal time under both Boolean and word models. Moreover, the computations can be performed modulo the powers of two, which allows additional practical benefits of applying binary computations.

2.5 Computing the Signs and the Values of Determinants

The value and frequently just the sign of $\det A$, the determinant of a square matrix A, are required in some fundamental geometric and algebraic/geometric computations such as the computation of convex hulls, Voronoi diagrams, algebraic curves and surfaces, multivariate and univariate resultants and Newton's polytopes. The faster numerical methods are preferred as long as the correctness of the output can be certified, which is usually the case in actual geometric and algebraic computations. In the customary *arithmetic filtering* approach, one applies numerical methods as long as they work and, in the rare cases when they fail, shifts to the slower algebraic methods.

Numerical computation of $\det A$ can rely on the factorizations $A = PLUP'$ (see Section 2.2) or $A = QR$ (59; 126). One can certify the output sign where the matrix A is well conditioned (237). The advanced preconditioning techniques in (222) can be employed to improve the conditioning of this matrix.

One can bound the precision of the error-free computations by performing them modulo sufficiently many reasonably bounded coprime moduli m_i and then recovering the value $\det A \bmod m$, $m = \prod_i m_i$, by applying the Chinese Remainder algorithm. Some relevant techniques are elaborated upon in (34). In particular, to minimize the computational cost, one can select random primes or prime powers m_i recursively until the output value modulo their product stabilizes. This signals that the product m is likely to exceed the unknown value $2|\det A|$, so that $(\det A) \bmod m$ is likely to produce the correct value of $\det A$. Typically for many applications, the value $|\det A|$ tends to be much smaller than the Hadamard's upper estimate for $|\det A|$. Then much fewer moduli m_i and hence much less computations are needed.

In an alternative approach in (201, Appendix), (202; 1; 86) $\det A$ is recovered as a least common denominator of the rational components of the solutions to linear systems $A\mathbf{y}(i) = \mathbf{b}(i)$ for random vectors $\mathbf{b}(i)$. The power of Hensel's lifting is employed for computing the solution vectors $\mathbf{y}(i)$.

Storjohann in (260; 261) advances randomized Newton's lifting to yield $\det A$ directly in the optimal asymptotic Boolean time $O(n^{\omega+1})$ for $\omega < 2.376$.

Wiedemann in 1986 and Coppersmith in 1994, followed by a stream of publications by other researchers, extended the Lanczos and block Lanczos classical algorithms to yield $\det A$. The most costly stages are the computation of the Krylov or block Krylov sequence (for the preconditioned matrix A and random vectors or block vectors) and the solution of a Hankel or block Hankel linear system of equations. To the advantage of this approach versus the other ones, including Storjohann's, the Krylov computations are relatively inexpensive for sparse and/or structured matrices. An important application is the computation of multivariate resultants, which are the determinants of sparse and structured matrices associated with the systems of multivariate polynomial equations. Here the approach becomes particularly attractive because the structure and sparseness enable us to multiply the matrices by vectors fast but hardly allow any other computational benefits. In (100) the extension of the algorithms to the multivariate determinants and resultants have been elaborated upon and analyzed in some detail.

Even for general matrix A, however, the bottleneck was initially at the stage of the solution of the block Hankel linear system (146). The structured Hensel lifting has finally become both theoretical and practical way out, which allowed both to decrease the exponent of the randomized Boolean complexity for computing the determinant of a general matrix from 10/3 in (146) to 16/5 and to keep all computational blocks practically valid (215; 218).

3 Univariate Polynomial Root-Finding, Factorization, and Approximate GCDs

3.1 Complexity of Univariate Polynomial Root-Finding

Solution of an nth degree polynomial equation,

$$p(x) = \sum_{i=0}^{n} p_i\, x^i = p_n \prod_{j=1}^{n} (x - z_j) = 0\ , \qquad p_n \neq 0,$$

that is, the computation of the roots $z_1, \dots, z_n$ for given coefficients $p_0, \dots, p_n$, is a classical problem that has greatly influenced the development of mathematics throughout four millennia, since the Sumerian times (209). The problem remains highly important for the theory and practice of the present day algebraic and algebraic/geometric computation, and dozens of new algorithms for its solution appear every year.

Polynomial root-finding requires an input precision exceeding the output precision by the factor of n, so that we need at least $(n+1)nb/2$ bits (and consequently at least $\lceil (n+1)nb/4 \rceil$ bit operations) to represent the input coefficients $p_0, \dots, p_{n-1}$ to approximate even a single root of a monic polynomial $p(x)$ within error bound 2^{-b}. To see why, 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 root $x = 6/7$ by 2^{-b}, and observe similar jumps where 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, $i = 0, 1, \ldots, n-1$.

It can be surprising, but we can approximate all n roots within 2^{-b} by using bn^2 Boolean (bit) operations up to a polylogarithmic factor, that is, we can approximate all roots almost as soon as we write down the input. We achieve this by means of the *divide-and-conquer algorithms* in (207; 209; 214) (see (154; 196; 251) on the related works). The algorithms first compute a sufficiently wide root-free annulus A on the complex plane, whose exterior and interior contain comparable numbers of the roots (that is, the same numbers up to a fixed constant factor). Then the two factors of $p(x)$ are numerically computed, that is, $F(x)$, having all its roots in the interior of the annulus, and $G(x) = p(x)/F(x)$, having no roots 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 roots of $p(x)$ are obtained.

It is interesting that both lower and upper bounds on the Boolean time decrease to bn (up to polylog factors) (214) if we only seek the factorization, rather than the roots, that is, if instead of all roots z_j, we compute some scalars a_j and b_j such that $||p(x) - \prod_{j=1}^{n}(a_j x - c_j)|| < 2^b$ for the polynomial norm defined by $|| \sum_i q_i x^i || = \sum_i |q_i|$.

Combining these bounds with a simple argument in (251, Section 20) readily supports the record complexity bound of $\tilde{O}_B((\tau + n)n^2)$ on the bit-complexity of the isolation of real roots of a polynomial of degree n with integer coefficients in a range $(-2^\tau, 2^\tau)$.

3.2 Root-Finding via Functional Iterations

The record computational complexity estimates for root-finding can be also obtained based on some *functional iteration* algorithms if one assumes their convergence rate based on the ample empirical evidence, although never proved formally. The users seem to accept such an evidence instead of the proof and prefer the latter algorithms because they are easy to program, have been carefully implemented, and allow to tune the precision of the computation to the precision required for every output root (which must be chosen higher for clustered and multiple roots than for the single isolated roots).

For approximating a single root z, the current practical champions are 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 (174), *Laguerre's method* (110; 132), and the *Jenkins–Traub algorithm* (136). One can deflate the input polynomial via its numerical division by $x - z$ to extend these algorithms to approximating the other roots.

To approximate all roots simultaneously, it is even more effective to apply the Durand–Kerner's (actually Weierstrass') algorithm, which is defined by the following recurrence:

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

Here, a simple customary choice (see (23) for some effective alternatives) for the n initial approximations $z_j(0)$ to the n roots of the polynomial $p(x)$ is given by $z_j(0) = Z\ t \exp(2\pi\sqrt{-1}/n)\ ,\quad j = 1, \ldots, n$, where $t > 1$ is a fixed tolerance and Z is an upper bound on the root radius, such that all roots z_j lie in the circle on the complex plane having radius Z and centered in the origin. This holds, e.g., for

$$Z = 2 \max_{i<n} |p_i/p_n|^{\frac{1}{n-i}}\ . \tag{2}$$

For a fixed l and for all j, the computation according to (1) is simple. We only need the order of n^2 ops for every l or only $O(n \log^2 n)$ ops with deteriorated numerical stability if we use the fast multipoint polynomial evaluation algorithms (2; 27; 33; 213).

We refer the reader to (29; 178; 179; 180; 181; 209) on this and other effective functional iteration algorithms and on further extensive bibliography and to (23) on the one of the most advanced current implementation MPSolve, based on the so called Aberth's algorithm (published first by Börsch–Supan and then Ehrlich).

3.3 Matrix Methods for Polynomial Root-Finding

Some recent highly effective polynomial root-finders rely on matrix methods. The roots are approximated as the eigenvalues of the associated (generalized) companion matrices, that is, the matrices whose characteristic polynomial is exactly the input polynomial. To the advantage of this approach, it employs numerically stable methods and the excellent software of matrix computations. Matlab's polynomial root-finder relies on the QR algorithm applied to the Frobenius companion matrix. This is effective because this algorithm is the present day champion in the eigenvalue computation. Malek and Vaillancourt in 1995 and then Fortune (109) succeeded by applying the same algorithm to other generalized companion matrices. They improve the approximations recursively by alternating the QR algorithm with Durand–Kerner's, which was the basis for the Fortune's competitive root-finding package EigenSolve.

The generalized companion matrices can be chosen highly structured, e.g., one can choose the Frobenius companion matrix or a diagonal plus rank-one (hereafter *DPR1*) matrix. The algorithms in (22; 24; 25; 219) exploit this structure to accelerate the eigenvalue computations. At first this was achieved in (22) based on the inverse (power) Rayleigh-Ritz iteration, which turned out to be also closely related to Cardinal's effective polynomial root-finder (cf. (219)). Then in (24; 25) the same idea was pursued based on the QR algorithm. All papers (22; 24; 25) use linear space and linear arithmetic time per iteration step versus quadratic in the general QR algorithm used by Matlab and Fortune. We refer the reader to (223) on various aspects of this approach and on some related directions for its further study.

3.4 Extension to Eigen-solving

According to ample empirical evidence (126), the cited structured eigen-solvers (aoolied for polynomial root-finding) typically use from 2 to 3 iteration steps to approximate an eigenvalue of a matrix and the associated eigenvectors. For an $n \times n$ DPR1 matrix, this means $O(n^2)$ ops for all eigenvalues and eigenvectors, versus the order of n^3 for general matrix. The paper (220) extends these fast eigen-solvers to generic matrix by defining its relatively inexpensive similarity transform into a DPR1 matrix. (Similarity transforms $A \leftarrow S^{-1}AS$ preserve eigenvalues and allow readily reconstruct eigenvectors.)

3.5 Real Polynomial Root-Finding

In many algebraic and geometric applications, the input polynomial $p(x)$ has real coefficients, and only its real roots must be approximated. When all roots are real, the Laguerre algorithm, its modifications, and some divide-and-conquer matrix methods (28) become highly effective, but all these algorithms do not work that well where the polynomial has also nonreal roots. Frequently, the real roots make up only a small fraction of all roots.

Somewhat surprisingly, the fastest real root-finder in the current practice is still MPSolve, which uses about the same running time for real roots as for all complex roots.

Some alternative algorithms specialized to approximation of only real roots are based on using the Descartes rule of signs or the Sturm or Sturm–Habicht sequences to isolate all real roots from each other. The record complexity of the isolation is again based on the factorization of a polynomial in the complex domain (see the end of Section 3.1).

3.6 Extension to Approximate Polynomial GCDs

Polynomial root-finding has been extended in (211) to the computation of approximate univariate polynomial greatest common divisor (gcd) of two polynomials, that is, the gcd of the maximum degree for two polynomials of the same or smaller degrees lying in the ϵ-neighbourhood of the input polynomials for a fixed positive ϵ. When the roots of both polynomials are approximated closely, it remains to compute a maximal matching in the associated bipartite graphs whose two sets of vertices are given by the roots of the two polynomials and whose edges connect the pair of roots that lie near each other (211).

This computational problem is important in control. The Euclidean algorithm is sensitive to input perturbations and can easily fail (94). Partial remedies rely on other approaches (see the bibliography in (111; 211)), notably via computing the singular values of the associated Sylvester (resultant) matrices. These approaches are more sound numerically than the Euclidean algorithm but still treat the gcds indirectly, via the coefficients, whose perturbation actually affects the gcds only via its affect on the roots.

3.7 Univariate real root isolation

In the current section, we consider only exact algorithms; algorithms that involve computations with rational numbers. Consider the following polynomial of degree d,

$$f(x) = a_d\, x^d + \ldots + a_1\, x + a_0,$$

where the coefficients are known exactly, that is they are rational numbers. Let $\tau = 1 + \max_{i \leq n}\{\lg |a_i|\}$ be the maximum bit size of the coefficients.

Real root isolation consists of computing disjoint intervals with rational endpoints that contains one and only one real root of f and every real root is contained in some interval. In addition we may also need to report the multiplicities of the real roots. The process differs from the one of the previous section, since we are interested only in the real roots and we wish to isolate them instead of approximating them to a desired accuracy.

An important ingredient for the performance and the analysis of the real root isolation algorithms is the minimal distance between the roots of f, the so called *separation bound*. It can be shown, e.g. (183; 182; 280), that this is at most $d^{-(d+2)/2}(d+1)^{(1-d)/2}2^{\tau(1-d)}$, or roughly speaking $2^{-\widetilde{O}(d\tau)}$.

3.7.1 Subdivision algorithms

The most frequently used exact algorithms for real root isolation are the subdivision algorithms, that mimic the binary search algorithm. They consider an initial interval that contains all the real roots and then they repeatedly subdivide it until is, some how, certified that the tested interval contains 0 or 1 real root. The subdivision algorithms differ in the way that they count real roots of a polynomial in an interval. Best known are the algorithms STURM, DESCARTES and BERNSTEIN.

The algorithm that resembles the most to the binary search algorithm is STURM, that was introduced by Sturm in 1835 (262). Initially the algorithm computes the (signed) polynomial remainder sequence of f and its derivative (see also section "The Sylvester resultant"). In order to compute the number of real roots in an interval, say $[a, b]$, where the endpoints are not roots of f, we evaluate the polynomial remainder sequence on the left endpoint of the interval and we count the number of sign variations, resulting a number $V_a \in \mathbf{N}$. We do the same for the right endpoint, getting a number $V_b \in \mathbf{N}$. Now the number of distinct real roots of f in $[a, b]$ is $V_a - V_b$. If the interval contains more that one real root then we subdivide it and we continue the algorithm on each subinterval. The complexity of the algorithm accounts the number of steps that it executes, that is the number of intervals that examines, times the time needed to evaluate the polynomial remainder sequence over a rational number of magnitude, in the worst case, proportional to the separation bound. Notice that the STURM algorithm does not assume a square-free polynomial.

Probably the first complete analysis of STURM is due to Heidel (134), see also (63), that achieved a complexity of $\widetilde{\mathcal{O}}_B(d^7\tau^3)$. The number of steps that the algorithm performs is $\mathcal{O}(d\tau+d\lg d)$ (71; 72; 81; 98). Using the fast algorithms for polynomial remainder sequences evaluation (243; 170) we can prove a $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ bound for the overall algorithm (71; 81; 98) and that in the same time we can also compute the multiplicities of the real roots.

DESCARTES algorithm relies on Descartes' rule of sign, which states that the number of sign variations in the coefficient list exceeds the number of positive real roots of f by an even number. It assumes a square-free polynomial. In general this rule provides an overestimation on the number of positive real roots. However, when the number of sign variations is zero or one, then we get the exact number of positive real roots. Thus, in order to count the number real roots of f in an interval, we transform the polynomial to another polynomial, the real roots of which in $(0, \infty)$ correspond to the real roots of f in the initial interval. If the number of real roots in greater than one, then we subdivide it and the we continue the algorithm on each subinterval. Notice that the polynomial transformations needed can be performed by shifting appropriately the variable x. The complexity of the algorithm is the number of intervals that it considers times the time needed for shifting the polynomial by a number of magnitude, in the worst case, proportional to the separation bound. For fast algorithms for polynomial shifts, the reader may refer to (27; 114; 115). The correctness and the termination of the algorithm depend on Vincent's theorem (270) and/or on the one and two circles theorems (6; 62; 157). The DESCARTES algorithm presented in its modern form by Collins and Akritas (61), see also (137; 156) and (249) for a unified scheme, concerning the way we traverse the subdivision tree and optimal memory management.

The BERNSTEIN algorithm is also based on Descartes' rule of sign and additionally takes advantage of the good properties of the Bernstein basis polynomial representation. The algorithm was presented by Lane and Riesenfeld (164) but its complexity, and its connection to the topological degree computation, was first analysed by Mourrain et al (194), see also (15; 138), (192) for a variant that has optimal memory management, and (88) for a probabilistic variant for polynomials with bit-stream coefficients. The complexity of all the approaches is $\widetilde{\mathcal{O}}_B(d^6\tau^2)$. Quite recently, it was proven (89) that the number of steps that both DESCARTES and BERNSTEIN perform is $\mathcal{O}(d\tau + d\lg d)$ and this made it possible to prove that the complexity of both algorithms is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ (89; 98). This bound holds for non square-free polynomials and in the same time we can also compute the multiplicities of the real roots.

3.7.2 The continued fraction algorithm

The continued fraction algorithm, CF, differs from the subdivision-based algorithms in that instead of bisecting a given initial interval it computes the continued fraction expansions of the real roots of the polynomial. The first formulation of the algorithm is due to Vincent (270), see also (4; 6) for historical references, based on his theorem, where it was stated that repeated transformations of the polynomial, of the form $x \mapsto c + \frac{1}{x}$, will eventually yield a polynomial with zero (or one) sign variation. Thus Descartes' rule implies that the transformed polynomial has zero (resp. one) real root in $(0, \infty)$. If one sign variation is attained then the inverse transformation can be applied in order to compute an isolating interval for the real root that corresponds to the original polynomial. Moreover the c's that appear in the transformation correspond, hopefully, to the partial quotients of the continued fraction expansion of the real root. However, Vincent's algorithm is exponential (61). He computed the partial quotients by repeated shift operations of the form $X \mapsto X + 1$, thus if one of the partial quotients, or even the sum of all, is of magnitude, say, 2^{τ} then an exponential number of steps must be performed.

Uspensky (268) extended Vincent's theorem by computing an upper bound on the number of transformations so as to isolate the real roots, but failed to deal with its exponential behaviour, see also (52; 247). Akritas (5; 3) attempted to tackle the exponential behaviour of the CF algorithm, by computing the partial quotients as positive lower bounds of the positive real roots, via Cauchy's bound, see e.g. (183; 184) and announced a complexity of $\widetilde{\mathcal{O}}_B(d^5\tau^3)$. However, it is not clear how this approach accounts for the increased coefficient size in the transformed polynomial after applying $X \mapsto c+X$. Moreover, the *magnitude* of the partial quotients is unbounded in general (31; 153). Tsigaridas and Emiris (267), using results from the metric theory of the continued fractions proved that the numbers of steps of the CF algorithm is $\widetilde{\mathcal{O}}(d\tau)$ and that its expected complexity is $\widetilde{\mathcal{O}}_B(d^4\tau)$. This bound holds for non square-free polynomials and in the same time we can also compute the multiplicities of the real roots. Quite recently, Sharma (254) proved that the worst case complexity of the algorithm is in $\widetilde{\mathcal{O}}_B(d^7\tau^2)$.

To summarize, STURM, DESCARTES and BERNSTEIN, perform $\mathcal{O}(d\tau + d \lg d)$ steps and they have $\widetilde{\mathcal{O}}(d^2\tau)$ and $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ arithmetic and bit complexity, respectively. The CF algorithm has $\widetilde{\mathcal{O}}_B(d^4\tau^2)$, resp. $\widetilde{\mathcal{O}}_B(d^7\tau^2)$, expected, resp. worst case, bit complexity.

There are several open questions concerning the exact algorithms for root isolation. What is the lower bound for the bit complexity of the problem? What is the expected (arithmetic or bit) complexity of the algorithms? Is the $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ bound tight for the subdivision algorithms? Does the $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ bound hold for complex root isolation?

4 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 $\mathbf{Q}[x_1, \ldots, x_n]$, 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 solving a well-constrained system of nonlinear equations. We also indicate how an initial phase of exact algebraic computation leads to certain numerical methods that can approximate all solutions; the interaction of symbolic and numeric computation is currently an active domain of research, e.g. (97). 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 root of two polynomial equations in one variable (155). 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 (151). 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 *Bézout-Dixon* (78; 95), *Sylvester-Macaulay* (46; 48; 172), and the *hybrid formulations* (76; 152). Concerning the Sylvester-Macaulay type, we shall emphasize also *sparse resultant* formulations (47; 263).

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 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 (246) introduced the concept of a characteristic set as a tool for studying solutions of algebraic differential equations. In 1984, Wu (279) in search of an effective method for automatic theorem proving, converted Ritt's method to ordinary polynomial rings. Given the aforementioned system P, the characteristic set algorithm transforms P into a triangular form, such that the set of common roots of P is equivalent to the set of roots of the triangular system (151).

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, robot kinematics and automatic geometric theorem proving.

4.1 The Sylvester Resultant

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

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

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 (155).

As a running example let us consider the following system in two variables provided by Lazard (166):

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

The Sylvester resultant can be used as a tool for eliminating several variables from a set of equations. 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 roots of f and g. More specifically, the Sylvester resultant of the Lazard system with respect to y is given by the following determinant:

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

An alternative matrix formulation named after Bézout yields the same determinant. This formulation is discussed below in the context of multivariate polynomials, in "Resultants of Multivariate Systems."

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 $\mathbf{Q}[x_1, \ldots, x_n]$. 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 (35; 60; 155).

Polynomial GCD algorithms have been developed that use some kind of implicit representations for symbolic objects and thus, avoid the computationally costly content and primitive part computations needed in those GCD algorithms for polynomials in explicit representation (75; 139; 145).

4.2 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 (67; 272). The *sparse (or toric) resultant* characterizes solvability over a smaller space which coincides with affine space under certain genericity conditions (119; 263). More general resultants are not analyzed here, see (42; 43). In any case, the main algorithmic question is to construct a matrix whose determinant is the resultant or a nontrivial multiple of it. Macaulay-type formulas give the resultant as the exact quotient of a determinant divided by one of its minors.

Due to the special structure of the Sylvester matrix, Bézout developed a method for computing the resultant as a determinant of order $\max\{m, n\}$ during the eighteenth century. Cayley (55) reformulated Bézout's method leading to Dixon's (78) extension to the bivariate case. This 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 polynomials in n variables (95). The vanishing of the determinant of the Bézout-Dixon matrix is a necessary and sufficient condition for the polynomials to have a nontrivial projective common root, and also a necessary condition for the existence of an affine common root. The Bézout-Dixon formulation gives the resultant up to a multiple, and hence, in the affine case it can happen that the vanishing of the determinant does not necessarily indicate that the equations in question have a common root. A nontrivial multiple, known as the *projection operator,* can be extracted via a method discussed in (54, thm. 3.3.4). This article, along with (90), explain the correlation between residue theory and the Bézout-Dixon matrix, which yields an alternative method for studying and approximating all common solutions.

In 1916, Macaulay (172) 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 (67). 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 (46) has proposed a general method that perturbs any polynomial system and extracts a nontrivial projection operator.

Using recent results pertaining to sparse polynomial systems (119; 263), a matrix formula for computing the sparse resultant of $n + 1$ polynomials in n variables was given by Canny and Emiris (47) and consequently improved in (50; 92). The determinant of the sparse resultant matrix, like the Macaulay and Dixon matrices, only yields a projection operation, not the exact resultant. D'Andrea (70) extended Macaulay's rational formula for the resultant to the sparse setting, thus defining the sparse resultant as the quotient of two determinants.

Here, sparsity means that only certain monomials in each of the $n + 1$ polynomials have nonzero 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 nonzero 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 (21). 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. Last, but not least, the classical resultant can identically vanish over sparse systems, whereas the sparse resultant does not and, hence, yields information about their common roots. For an example, see (68).

4.3 Polynomial System Solving by Using Resultants

Suppose we are asked to find the common roots 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 (46; 49; 68), 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 roots of the given polynomials equations. The u-resultant method relies on 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 (91; 95; 175).

Consider the previous example augmented by a generic linear form:

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

As described in Canny, Kaltofen and Lakshman (48), 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 can also be applied to reduce polynomial system solving to a regular or generalized eigenproblem (cf. "Matrix Eigenvalues and Singular Values Problems"), thus, transforming the nonlinear question to a problem in linear algebra. This is a classical technique that enables us to approximate all solutions (cf. (9; 49; 54; 91; 95)). 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:

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

Vector $\mathbf{v}M$ 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 $\mathbf{v}M(w) = \mathbf{0}$ in vector $\mathbf{v}$ 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:

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

For every solution (a, b) of the original system, there is a vector $\mathbf{v}$ 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. (91)). As for the eigenvalues, they correspond to the values of w at the solutions.

An alternative method for approximating or isolating all real roots of the system is to use the so-called Rational Univariate Representation (RUR) of algebraic numbers (45; 248). This allows us to express each root coordinate as the value of a univariate polynomial, evaluated over a different algebraic number. The latter are all solutions of a single polynomial equation, and can thus be approximated or isolated by the algorithms presented in preceding sections. The polynomials involved in this approach are derived from the resultant.

The resultant matrices are sparse and have quasi Toeplitz/Hankel structure (also called multilevel Toeplitz/Hankel structure), which enables their fast multiplication by vectors. By combining the latter property with various advanced nontrivial methods of multivariate polynomial root-finding, substantial acceleration of the construction and computation of the resultant matrices and approximation of the system's solutions was achieved in (32; 99; 188; 189; 190).

An empirical comparison of the detailed resultant formulations can be found in (96; 175). The multivariate resultant formulations have been used for diverse applications such as *algebraic and geometric reasoning* (54; 175), *computer-aided design* (159; 252), *robot kinematics* (245; 175), computing *molecular conformations* (93; 176) and for *implicitization and finding base points* (58; 175).

4.4 Gröbner Bases

Solving systems of nonlinear equations can be formulated in terms of polynomial ideals (16; 67; 129; 158). Let us first establish some terminology.

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

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

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

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

A version of the *Hilbert Nullstellensatz* states that

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

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 $\deg(t) = e_1 + e_2 + \cdots + e_n$. 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 (118; 67). 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 (72, p. 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 (16; 67; 129; 158).

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 $G \subseteq \mathbf{Q}[x_1, \ldots, x_n]$ is a *Gröbner basis* if and only if every polynomial in $\mathbf{Q}[x_1, \ldots, x_n]$ has a unique normal form with respect to G. Bruno Buchberger (36; 37; 38; 39) showed that every basis for an ideal $(p_1, \ldots, p_r)$ in $\mathbf{Q}[x_1, \ldots, x_n]$ 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

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

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 (16; 118; 163; 278). In the case of zero-dimensional ideals, those whose varieties have only isolated points, Faugère et al. (103) 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. More recently, significant progress has been achieved in the algorithmic realm of Gröbner basis computations by the work of J-C. Faugère (101; 102).

Another way to finding all common real roots is by means of RUR; see the previous section or (45; 248). All polynomials involved in this approach can be derived from the Gröbner basis.

A rather recent development concerns the generalization of Gröbner bases to *border bases*, which contain all information required for system solving but can be computed faster and seem to numerically more stable (158; 193; 257).

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

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

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:

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

It should 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.

$$x^2y^2$$

$$f_1^*$$

$$-3xy^2 - 2y^3 + 3y^2$$

$$f_2^* \qquad\qquad f_3^*$$

$$-3xy - 2y^3 - y^2 + 3y \qquad -3x - 2y^3 + 2y^2$$

$$f_2^* \qquad\qquad f_3^*$$

$$-3x - 2y^3 - y^2 + 3 \qquad -3x - 2y^3 + 2y^2$$

$$f_3^* \qquad\qquad f_3^*$$

$$-3x - y^2 - 2y + 3$$

$$f_3^*$$

$$-3x - 2y + 2$$

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.

In a survey article, Buchberger (38) 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 (150; 279), multivariate polynomial factorization and GCD computations (121), polynomial interpolation (161; 162), coding and cryptography (8; 104), and robotics (105).

5 Research Issues and Summary

The present day computations in sciences, engineering and signal and image processing employ both algebraic and numerical approaches. These two approaches rely on distinct techniques and ideas, but in many cases combining the power of both of them enhances the efficiency of the computations. This is frequently the case in matrix computations and root-finding for univariate polynomials and multivariate systems of polynomial equations. We briefly reviewed these three subjects and gave pointers to the extensive bibliography.

Among numerous interesting and important research directions of the topics in Sections 2 and 3, we wish to cite computations with structured matrices, including the subject of computations with semiseparable matrices currently of growing interest, their applications to polynomial root-finding, new techniques for preconditioning with many promising extensions (which include the computation of determinants), and polynomial root-finding.

Section 4 of this chapter has briefly reviewed polynomial system solving based on resultant matrices as well as Gröbner bases. Both approaches are currently active, including in applications dealing with small and medium-size systems. Efficient implementations handling the nongeneric cases, including multiple roots and nonisolated solutions, is probably the most crucial issue today in relation to resultants. Other interesting questions include better algorithms, in particular the ones exploiting matrix structure, for both resultants and Gröbner bases.

6 Defining Terms

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

Condition number: A scalar κ 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 and OUTPUT ERROR NORM $\approx \kappa$ INPUT ERROR NORM.

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 nonzero if and only if the matrix is invertible.

Gröbner basis: A generating set of a set of polynomials, such that the (multivariate) division of a polynomial by this generating set, has a unique remainder.

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 $1 \prec x \prec x^2 \prec \cdots \prec y \prec xy \prec x^2y \cdots \prec y^2 \prec xy^2 \cdots$.

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.

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

Resultant: A polynomial in the coefficients of a system of n polynomials with $n + 1$ unknowns, the vanishing of which is the necessary and sufficient condition for the existence of a solution of the system.

Separation bound: The minimum distance between two (complex) roots of a univariate polynomial.

Singularity: A square matrix is singular if there is a nonzero second matrix such that 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 whose every entry can be derived by a formula depending on a smaller number of parameters, typically on the order of at most $m+n$ parameters for an $m \times n$ matrix, as opposed to its mn entries. For instance, an $m \times n$ Cauchy matrix has $\frac{1}{s_i - t_j}$ as the entry in row i and column j for $m + n$ parameters s_i and t_j, $1, \ldots, m$, $j = 1, \ldots, n$.

References

References

[1] Abbott, J. Bronstein, M. and Mulders, T., Fast deterministic computations of the determinants of dense matrices. *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC'99)*, 197–204, ACM Press, New York, 1999.

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

[3] Akritas, A., An implementation of Vincent's theorem. *Numerische Mathematik*, 36:53–62, 1980.

[4] Akritas, A., There is no "Uspensky's method". Extended Abstract. In *Proc. Symposium on Symbolic and Algebraic Computation*, pages 88–90, Waterloo, Ontario, Canada, 1986.

[5] Akritas, A., *Elements of Computer Algebra with Applications.* J. Wiley & Sons, New York, 1989.

[6] Alesina, A. and Galuzzi, M., A new proof of Vincent's theorem. *L'Enseignement Mathématique*, 44:219–256, 1998.

[7] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. *LAPACK Users' Guide.* 3rd Edition, SIAM Publications, Philadelphia, PA, 1999.

[8] Augot, D., Bardet, M., and Faugère, J-C. Efficient decoding of (binary) cyclic codes above the correction capacity of the code using Gröbner bases. In Proc. IEEE Internat. Symp. Information Theory 2003 (ISIT '03). IEEE Press, 2003.

[9] Auzinger, W. and Stetter, H.J., 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,* 12–30. Birkhäuser, Basel, 1988.

[10] Bach, E. and Shallit, J., *Algorithmic Number Theory, Volume 1: Efficient Algorithms.* The MIT Press, Cambridge, MA, 1996.

[11] Bai, Z., Demmel, J., Dongarra, J., Ruhe, A. and van der Vorst, H., editors, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000.

[12] Bailey, D., Borwein, P., and Plouffe, S., On the rapid computation of various polylogarithmic constants. *Math. Comp.,* 66, 903–913, 1997. `http://mosaic.cecm.sfu.ca/preprints/1995pp.html`, 1995.

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

[14] Barrett, R., Berry, M.W., Chan, T.F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C. and Van Der Vorst, H., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia, 1993.

[15] Basu, S., Pollack, R., and Roy, M.-F., *Algorithms in Real Algebraic Geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2003.

[16] Becker, T. and Weispfenning, V., *Gröbner bases: A Computational Approach to Commutative Algebra.* Springer-Verlag, New York, 1993.

[17] Benzi, M., Preconditioning techniques for large linear systems: a survey. *J. of Computational Physics*, 182, 418–477, 2002.

[18] Berberich, E., Eigenwillig, A., Hemmer, M., Hert, S., Kettner, L., Mehlhorn, K., Reichel, J., Schmitt, S., Schömer, E., and Wolpert, N., EXACUS: Efficient and Exact Algorithms for Curves and Surfaces. In *ESA*, volume 1669 of *LNCS*, pages 155–166. Springer, 2005.

[19] Berlekamp, E.R., 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, New York, 1968.

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

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

[22] Bini, D.A., Gemignani, L. and Pan, V.Y., Inverse power and Durand/Kerner iteration for univariate polynomial root-finding. *Computers and Mathematics (with Applications)*, 47 (2/3), 447–459, January 2004. (Also Technical Reports TR 2002 003 and 2002 020, *CUNY Ph.D. Program in Computer Science, Graduate Center, City University of New York*, 2002.)

[23] Bini, D.A. and Fiorentino, G., Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder. *Numerical Algorithms*, 23, 127–173, 2000.

[24] Bini, D.A., Gemignani, L. and Pan, V.Y., Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equation. *Numerische Math.*, 3, 373–408, 2005. (Also Technical Report 1470, *Department of Math., University of Pisa*, Pisa, Italy, July 2003.)

[25] Bini, D.A., Gemignani, L. and Pan, V.Y., Improved initialization of the accelerated and robust QR-like polynomial root-finding. *Electronic Transactions on Numerical Analysis*, 17, 195–205, 2004.

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

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

[28] Bini, D. and Pan, V.Y., Computing matrix eigenvalues and polynomial zeros where the output is real. *SIAM J. on Computing*, 27 (4), 1099–1115, 1998.

[29] Bini, D. and Pan, V.Y., *Polynomial and Matrix Computations, Volume 2.* Birkhäuser, Boston, to appear.

[30] Björck, Å., *Numerical Methods for Least Squares Problems.* SIAM, Philadelphia, 1996.

[31] Bombieri, E. and van der Poorten, A., Continued fractions of algebraic numbers. In *Computational algebra and number theory (Sydney, 1992)*, pages 137–152. Kluwer Acad. Publ., Dordrecht, 1995.

[32] Bondyfalat, D., Mourrain, B. and Pan, V.Y., Computation of a specified root of a polynomial system of equations using eigenvectors. *Linear Algebra and Its Applications*, 319, 193-209, 2000. Proc. version in *Proc. ACM Annual Intern. Symp. on Symbolic and Algebraic Comp. (ISSAC98)*, 252-259, ACM Press, New York, 1998.

[33] Borodin, A. and Munro, I., *Computational Complexity of Algebraic and Numeric Problems.* American Elsevier, New York, 1975.

[34] Brönnimann, H., Emiris, I.Z., Pan, V.Y., Pion, S., Sign determination in residue number systems. *Theoretical Computer Science*, 210 (1), 173–197, 1999. Proceedings Version in *Proc. 13th Ann. ACM Symp. on Computational Geometry*, 174–182, ACM Press, New York, 1997.

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

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

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

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

[39] Buchberger, B., *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal.* Dissertation, University Innsbruck, Austria, 1965.

[40] Buchberger, B., Collins, G.E., Loos, R., and Albrecht, R., editors. *Computer Algebra: Symbolic and Algebraic Computation.* Springer-Verlag, 2nd edition, 1983.

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

[42] Busé, L., Elkadi, M., and Mourrain, B. Generalized resultants over unirational algebraic varieties. *J. Symbolic Comp.*, 29, 515–526, 2000.

[43] Busé, L., Elkadi, M., and Mourrain, B. Residual resultant of complete intersection. *J. Pure & Applied Algebra*, 164, 35–57, 2001.

[44] Busé, L., Elkadi, M., and Mourrain, B., editors. Special Issue on Algebraic–Geometric Computations, *Theor. Comp. Science* (in press).

[45] Canny, J., Some Algebraic and Geometric Computations in PSPACE. In *Proc. ACM Symp. Theory of Computing*, 460–467, 1988.

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

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

[48] Canny, J., Kaltofen, E., and Lakshman, Y., Solving systems of nonlinear polynomial equations faster. In *Proc. ACM-SIGSAM 1989 Internat. Symp. Symbolic Algebraic Comput. ISSAC '89,* 121–128. ACM, 1989.

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

[50] Canny, J. and Pedersen, P., An algorithm for the Newton resultant. Technical Report 1394, Computer Science Department, Cornell University, 1993.

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

[52] Cantor, D., Galyean, P., and Zimmer, H., A continued fraction algorithm for real algebraic numbers. *Mathematics of Computation*, 26(119):785–791, July 1972.

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

[54] Cardinal, J.-P. and Mourrain, B., Algebraic approach of residues and applications. In *The Mathematics of Numerical Analysis,* Renegar, J., Shub, M., and Smale, S., Eds., volume 32 of *Lectures in Applied Math.,* 189–210. AMS, Providence, RI, 1996.

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

[56] Chen, K., *Matrix Preconditioning Techniques and Applications.* Cambridge University Press, Cambridge, England, 2005.

[57] Chen, Z., and Storjohann, A., A BLAS based C library for exact linear algebra on integer matrices. *Proc. 2005 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'05),* (M. Kauers, editor), 92–99, ACM Press, New York, 2005.

[58] Chionh, E., *Base Points, Resultants and Implicit Representation of Rational Surfaces.* Ph.D. Thesis, Department Computer Science, University Waterloo, 1990.

[59] Clarkson, K.L., Safe and effective determinant evaluation. *Proc. 33rd Ann. IEEE Symp. on Foundations of Computer Science*, 387–395, IEEE Computer Society Press, Los Alamitos, California, 1992.

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

[61] Collins, G.E. and Akritas, A., Polynomial real root isolation using Descartes' rule of signs. In *SYMSAC '76*, pages 272–275, New York, USA, 1976. ACM Press.

[62] Collins, G.E., and Johnson, J., Quantifier elimination and the sign variation method for real root isolation. In *ISSAC*, pages 264–271, 1989.

[63] Collins, G.E., and Loos, R., Real zeros of polynomials. In B. Buchberger, G.E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 83–94. Springer-Verlag, Wien, 2nd edition, 1982.

[64] Coppersmith, D., Solving homogeneous linear equations over GF(2) via block Wiedemann algorithm. *Math. of Computation*, 62(205), 333–350, 1994.

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

[66] Corless, R.M., Gianni, P.M., Trager, B.M., and Watt, S.M., The singular value decomposition for polynomial systems. In Levelt (168), 96–103.

[67] Cox, D., Little, J., and O'Shea, D. *Ideals, Varieties, and Algorithms*, 2nd edition. Undergraduate Texts in Mathematics. Springer, New York, 1997.

[68] Cox, D., Little, J., and O'Shea, D. *Using Algebraic Geometry*, 2nd edition. Graduate Texts in Mathematics, 185. Springer, New York, 2005.

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

[70] D'Andrea, C. Macaulay-style formulas for the sparse resultant. *Trans. of the AMS*, 354, 2595–2629, 2002.

[71] Davenport, J.H., Cylindrical algebraic decomposition. Technical Report 88–10, School of Mathematical Sciences, University of Bath, England, available at: http://www.bath.ac.uk/masjhd/, 1988.

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

[73] Demmel, J.J.W., *Applied Numerical Linear Algebra.* SIAM Publications, Philadelphia, PA, 1997.

[74] Díaz, A., Emiris, I.Z., Kaltofen, E., and Pan, V.Y., Algebraic Algorithms. Chapter 16 in Handbook of Algorithms and Theory of Computation, M.J. Atallah, editor. CRC Press, Boca Raton, Florida, 1999.

[75] Díaz, A. and Kaltofen, E., On computing greatest common divisors with polynomials given by black boxes for their evaluation. In *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC '95,* Levelt, A.H.M., Ed., 232–239, ACM Press, New York, l995.

[76] Dickenstein, A., and Emiris, I.Z., Multihomogeneous resultant formulae by means of complexes. *J. Symbolic Comp.*, 36, 317–342, 2003.

[77] Dickenstein, A., and Emiris, I.Z., editors. Solving Polynomial Equations: Foundations, Algorithms and Applications. Volume 14 in "Algorithms and Computation in Mathematics". Springer-Verlag, Berlin, 2005.

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

[79] Dongarra, J.J., Duff, I.S., Sorensen, D.C. and Van Der Vorst, H.A., *Numerical Linear Algebra for High-Performance Computers*, SIAM, Philadelphia, 1998.

[80] Dongarra, J., Bunch, J., Moler, C., and Stewart, P. *LINPACK Users' Guide.* SIAM Publications, Philadelphia, PA, 1978.

[81] Du, Z., Sharma, V., and Yap, C.K., Amortized bound for root isolation via Sturm sequences. In D. Wang and L. Zhi, editors, *Int. Workshop on Symbolic Numeric Computing*, pages 113–129, School of Science, Beihang University, Beijing, China, 2005. Birkhauser.

[82] Duff, I.S., Erisman, A.M. and Reid, J.K., *Direct Methods for Sparse Matrices.* Clarendon Press, Oxford, England, 1986.

[83] Dumas, J.-G., Gautier, T., Giesbrecht, M., Giorgi, P., Hovinen, B., Kaltofen, E., Saunders, B.D., Turner, W.J., and Villard, G. LinBox: A generic library for exact linear algebra. In Cohen, A.M., Gao, X.-S., and Takayama, N., editors, *Proc. First Internat. Congress Math. Software ICMS 200*, pages 40-50, Beijing, China, Singapore, 2002.

[84] Dumas, J-G., Gautier, T. and Pernet, C., Finite field linear algebra subroutines. *Proc. Internat. Symp. Symbolic Algebraic Comput. (ISSAC'02),* 63–74. ACM Press, New York, 2002.

[85] Dumas, J-G., Giorgi, P. and Pernet, C., Finite field linear algebra package. *Proc. Internat. Symp. Symbolic Algebraic Comput. (ISSAC'04),* 118–126. ACM Press, New York, 2004.

[86] Eberly, W., Giesbrecht, M. and Villard, G., On computing the determinant and Smith form of an integer matrix. *Proc. 41st Annual Symposium on Foundations of Computer Science (FOCS'2000)*, 675–685, IEEE Computer Society Press, Los Alamitos, California, 2000.

[87] Eberly, W. and Kaltofen, E., On randomized Lanczos algorithms. In Küchlin, W., Ed., *Proc. 1997 Internat. Symp. Symbolic Algebraic Comput. (ISSAC'97),* 176–183, ACM Press, New York, 1997.

[88] Eigenwillig, A., Kettner, L., Krandick, W., Mehlhorn, K., Schmitt, S., and Wolpert, N., A Descartes Algorithm for Polynomials with Bit-Stream Coefficients. In V. Ganzha, E. Mayr, and E. Vorozhtsov, editors, *CASC*, volume 3718 of *LNCS*, pages 138–149. Springer, 2005.

[89] Eigenwillig, A., Sharma, V., and Yap, C.K., Almost tight recursion tree bounds for the Descartes method. In *ISSAC '06: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pages 71–78, New York, NY, USA, 2006. ACM Press.

[90] Elkadi, M. and Mourrain, B., Algorithms for residues and Lojasiewicz exponents. *J. Pure & Appl. Algebra*, 153, 27–44, 2000.

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

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

[93] Emiris, I.Z., Fritzilas, E., and Manocha, D. Algebraic algorithms for conformational analysis and docking. *Intern. J. Quantum Chemistry*, 106, 190–210, 2005.

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

[95] Emiris, I.Z. and Mourrain, B. Matrices in elimination theory. *J. Symbolic Comp.*, 28, 3–44, 1999.

[96] Emiris, I.Z. and Mourrain, B. Computer Algebra Methods for Studying and Computing Molecular Conformations. *Algorithmica*, 25, 372–402, 1999.

[97] Emiris, I.Z., Mourrain, B., and Pan, V.Y., editors. Special Issue on Algebraic and Numerical Algorithms, *Theor. Comp. Science*, 315, 307–672, 2004.

[98] Emiris, I.Z., Mourrain, B., and Tsigaridas, E.P., Real Algebraic Numbers: Complexity Analysis and Experimentation. In P. Hertling, C. Hoffmann, W. Luther, and N. Revol, editors, *Reliable Implementations of Real Number Algorithms: Theory and Practice*, LNCS (to appear). Springer Verlag, 2007. also available in www.inria.fr/rrrt/rr-5897.html.

[99] Emiris, I.Z. and Pan, V.Y., Symbolic and numeric methods for exploiting structure in constructing resultant matrices. *J. Symbolic Comp.*, 33, 393–413, 2002.

[100] Emiris I.Z. and Pan, V.Y., Improved algorithms for computing determinants and resultants. *J. of Complexity*, 21 (1), 43–71, 2005. Proceedings version in *Proc. of the 6th International Workshop on Computer Algebra in Scientific Computing (CASC '03)*, edited by E. W. Mayr, V. G. Ganzha, and E. V. Vorozhtzov, 81–94, Technische Univ. München, Germany, 2003.

[101] Faugère, J.-C., A new efficient algorithm for computing Gröbner bases (F4). *J. Pure & Applied Algebra*, 139, 61–88, 1999.

[102] Faugère, J.-C., A new efficient algorithm for computing Gröbner bases without Reduction to Zero (F5). In *Proc. 2002 Internat. Symp. Symbolic Algebraic Comput. (ISSAC '02),* pages 75–83, ACM Press, 2002.

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

[104] Faugère, J.-C., and Joux, A., Algebraic cryptanalysis of hidden field equation (HFE): cryptosystems using Gröbner bases. In *Proc. CRYPTO 2003*, pages 44–60, 2003.

[105] Faugère, J-C., and Lazard, D., The Combinatorial Classes of Parallel Manipulators. *Mechanism and Machine Theory*, 30, 765–776, 1995.

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

[107] Ferguson, H.R.P. and Forcade, R.W., Multidimensional Euclidean algorithms. *J. Reine Angew. Math.,* 334, 171–181, 1982.

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

[109] Fortune, S., An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials. *J. of Symbolic Computation*, 33 (5), 627–646, 2002. Proc. version in *Proc. Intern. Symp. on Symbolic and Algebraic Computation (ISSAC'01)*, 121–128, ACM Press, New York, 2001.

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

[111] Gao, S., Kaltofen, E., May, J., Yang, Z. and Zhi, L., Approximate factorization of multivariate polynomial via differential equations. *Proc. International Symposium on Symbolic and Algebraic Computaion (ISSAC'04)*, 167-174, ACM Press, New York, 2004.

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

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

[114] von zur Gathen, J., and Gerhard, J., Fast Algorithms for Taylor Shifts and Certain Difference Equations. In *ISSAC*, pages 40–47, 1997.

[115] von zur Gathen, J. and Gerhard, J., *Modern Computer Algebra.* Cambridge University Press, Cambridge, UK, 2003 (2nd edition).

[116] von zur Gathen, J., and Lücking, T., Subresultants revisited. *Theor. Comput. Sci.*, 1-3(297):199–239, 2003.

[117] von zur Gathen, J. and Shoup, V., Computing Frobenius maps and factoring polynomials. *Comput. Complexity,* 2, 187–224, 1992.

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

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

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

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

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

[123] Gilbert, J.R. and Hafsteinsson, H., Parallel Symbolic Factorization of Sparse Linear Systems *Parallel Computing*, 14, 151–162, 1990.

[124] Gilbert, J.R. and Schreiber, R., Highly parallel sparse Cholesky factorization. *SIAM J. on Scientific Computing*, 13, 1151–1172, 1992.

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

[126] Golub, G.H. and Van Loan, C.F., *Matrix Computations,* 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

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

[128] Greenbaum, A., *Iterative Methods for Solving Linear Systems.* SIAM Publications, Philadelphia, PA, 1997.

[129] Greuel, G.-M., and Pfister, G. A Singular Introduction to Commutative Algebra (with contributions by O. Bachmann, C. Lossen, and H. Schönemann). Springer-Verlag 2002.

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

[131] Habicht, W., Eine verallgemeinerung des sturmschen wurzelzählverfarens. *Comm. Math. Helvetici*, 21:99–116, 1948.

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

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

[134] Heindel, L.E., Integer arithmetic algorithms for polynomial real zero determination. *Journal of the Association for Computing Machinery*, 18(4):533–548, October 1971.

[135] Higham, N.J., *Accuracy and Stability of Numerical Algorithms.* SIAM, Philadelphia, 2002 (second edition).

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

[137] Johnson, J. Algorithms for polynomial real root isolation. In B. Cavinsess and J. Johnson, editors, *Quantifier elimination and cylindrical algebraic decomposition*, pages 269–299. Springer, 1998.

[138] Johnson, J., Krandick, W., Lynch, K., Richardson, D., and Ruslanov, A., High-performance implementations of the Descartes method. In *ISSAC '06: Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 154–161, New York, NY, USA, 2006. ACM Press.

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

[140] Kaltofen, E., Polynomial factorization 1982-1986. In *Computers in Mathematics,* Chudnovsky, D.V. and Jenks, R.D., Eds., volume 125 of *Lecture Notes in Pure and Applied Mathematics,* 285–309. Marcel Dekker, New York, 1990.

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

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

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

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

[145] Kaltofen, E. and Trager, B., 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.

[146] Kaltofen, E. and Villard, G., On the complexity of computing determinants. *Proc. Fifth Asian Symposium on Computer Mathematics (ASCM 2001)*, (Shirayanagi, Kiyoshi and Yokoyama, Kazuhiro, editors), *Lecture Notes Series on Computing*, 9, 13–27, World Scientific, Singapore, 2001.

[147] Kaltofen, E. and Villard, G., Computing the sign or the value of the determinant of an integer matrix, a complexity survey. *J. Computational Applied Math.*, 162(1), 133–146, 2004.

[148] Kaltofen, E. and Villard, G., Complexity of computing determinants. *J. Computational Complexlty*, 13(3–4), 91–130, 2004.

[149] Kaporin, I., The aggregation and cancellation techniques as a practical tool for faster matrix multiplication. *Theoretical Computer Science*, 315 (2–3), 469–510, 2004.

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

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

[152] Khetan, A., The resultant of an unmixed bivariate system. *J. Symbolic Comput.* 36, 425–442, 2003.

[153] Khintchine, A., *Continued Fractions.* University of Chicago Press, Chicago, 1964.

[154] Kirrinnis, P., Polynomial factorization and partial fraction decomposition by simultaneous Newton's iteration. *J. of Complexity*, 14, 378–444, 1998.

[155] Knuth, D.E., *The Art of Computer Programming, Vol. 2, Seminumerical Algorithms,* 2nd ed., Addison-Wesley, Reading, MA, 1981. 3rd ed., 1997.

[156] Krandick, W., Isolierung reeller nullstellen von polynomen. In J. Herzberger, editor, *Wissenschaftliches Rechnen*, pages 105–154. Akademie-Verlag, Berlin, 1995.

[157] Krandick, W., and Mehlhorn, K., New bounds for the Descartes method. *JSC*, 41(1):49–66, Jan 2006.

[158] Kreuzer, M., and Robbiano, L., Computational Commutative Algebra 1. Springer Verlag, Heidelberg, 2000.

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

[160] Laderman, J., Pan, V.Y. and Sha, H.X., On practical algorithms for accelerated matrix multiplication. *Linear Algebra and Its Applications*, 162–164, 557–588, 1992.

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

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

[163] Lakshman, Y.N., *On the complexity of computing Gröbner bases for zero-dimensional polynomial ideals.* Ph.D. Thesis, Computer Science Department, Rensselaer Polytechnic Institute, Troy, New York, 1990.

[164] Lane, J.M., and Riesenfeld, R.F., Bounds on a polynomial. *BIT*, 21:112–117, 1981.

[165] Lawson, C.L. and Hanson, R.J., *Solving Least Squares Problems.* Prentice-Hall, Englewood Cliffs, New Jersey, 1974. Reissued with a survey of recent debvelopments by SIAM, Philadelphia, 1995.

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

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

[168] Levelt, A.H.M., Ed., *Proc. 1995 Internat. Symp. Symbolic Algebraic Comput. ISSAC'95,* ACM Press, New York, 1995.

[169] Leyland, P., Cunningham project data. Internet document, Oxford University,
`ftp://sable.ox.ac.uk/pub/math/cunningham/`, Nov. 1995.

[170] Lickteig, T., and Roy, M.-F., Sylvester-Habicht Sequences and Fast Cauchy Index Computation. *J. Symb. Comput.*, 31(3):315–341, 2001.

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

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

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

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

[175] Manocha, D., *Algebraic and Numeric Techniques for Modeling and Robotics.* Ph.D. Thesis, Comp. Science Div., Dept. of Electrical Engineering and Computer Science, University of California, Berkeley, 1992.

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

[177] McCormick, S., Ed., *Multigrid Methods.* SIAM Publications, Philadelphia, 1987.

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

[179] McNamee, J.M., A Supplementary Bibliography on Roots of Polynomials. *J. Computational and Applied Mathematics*, 78, 1, 1997.

[180] McNamee, J.M., An Updated Supplementary Bibliography on Roots of Polynomials. *J. Computational and Applied Mathematics*, 110, 305–306, 1999.

[181] McNamee, J.M., A 2000 Updated Supplementary Bibliography on Roots of Polynomials. *J. Computational and Applied Mathematics*, 142, 433–434, 2002.

[182] Mignotte, M., Some useful bounds. In B. Buchberger, G.E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 259–263. Springer-Verlag, Wien, 2nd edition, 1982.

[183] M. Mignotte. *Mathematics for Computer Algebra.* Springer-Verlag, 1992.

[184] Mignotte, M., and Stefanescu, D., *Polynomials: An algorithmic approach.* Springer, 1999.

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

[186] Miranker, W.L. and Pan, V.Y., Methods of Aggregations. *Linear Algebra and Its Applications*, 29, 231–257, 1980.

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

[188] Mourrain, B. and Pan, V.Y., Asymptotic acceleration of solving polynomial systems. *Proc. 30th Annual ACM Symp. on Theory of Computing*, 488-496, ACM Press, New York, 1998.

[189] Mourrain, B. and Pan, V.Y., Multivariate polynomials, duality and structured matrices, *J. of Complexity*, 16 (1), 110–180, 2000.

[190] Mourrain, B., Pan, V.Y. and Ruatta, O., Accelerated solution of multivariate polynomial systems of equations. *SIAM J. on Computing*, 32, 2, 435-454, 2003. Proc. version in Proceedings of the Smalefest 2000, F. Cucker and M. Rojas (Eds.), Foundations of Computational Math. Series, 267-294, World Scientific, New Jersey, 2002.

[191] Mourrain, B., Pavone, J.-P., Trébuchet, P., and Tsigaridas, E.P., SYNAPS: a library for symbolic-numeric computing. In *Proc. 8th Int. Symp. on Effective Methods in Algebraic Geometry (MEGA)*, Italy, May 2005. (software presentation).

[192] Mourrain, B., Rouillier, F., and Roy, M.-F., *Bernstein's basis and real root isolation*, pages 459–478. Mathematical Sciences Research Institute Publications. Cambridge University Press, 2005.

[193] Mourrain, B., and Trébuchet, P., Solving projective complete intersection faster. *J. Symbolic Comput.,* 33(5), 679–699, 2002.

[194] Mourrain, B., Vrahatis, M., and Yakoubsohn, J.C., On the complexity of isolating real roots and computing with certainty the topological degree. *J. Complexity*, 18(2), 2002.

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

[196] Neff, C.A., Reif, J.H., An $O(n^{l+\epsilon})$ algorithm for the complex root problem. *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Scinece (FOCS'94)*, 540–547, IEEE Computer Society Press, Los Alamitos, California, 1994.

[197] Niederreiter, H., New deterministic factorization algorithms for polynomials over finite fields. In *Finite Fields: Theory, Applications and Algorithms,* Mullen, L. and Shiue, P.J.-S., Eds., volume 168 of *Contemporary Mathematics,* 251–268. American Math. Society, Providence, RI, 1994.

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

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

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

[201] Pan, V.Y., Complexity of parallel matrix computations. *Theoretical Computer Science*, 54, 65–85, 1987.

[202] Pan, V.Y., Computing the determinant and the characteristic polynomials of a matrix via solving linear systems of equations. *Information Processing Letters*, 28, 71–75, 1988.

[203] Pan, V.Y., Complexity of algorithms for linear systems of equations. In *Computer Algorithms for Solving Linear Algebraic Equations (State of the Art),* Spedicato, E., Ed., volume 77 of *NATO ASI Series,* Series F: Computer and Systems Sciences, 27–56, Springer, Berlin, 1991, and Academic Press, Dordrecht, the Netherlands (1992).

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

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

[206] Pan, V.Y., Parallel computation of a Krylov matrix for a sparse and structured input. *Mathematical and Computer Modelling,* 21(11), 97–99, 1995.

[207] Pan, V.Y., Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers in Mathematics (with Applications),* 31(12), 97–138, 1996. Proceedings version: 27th Ann. ACM STOC, 741–750, ACM Press, New York, 1995.

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

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

[210] Pan, V.Y., Some recent algebraic/numerical algorithms. *Electronic Proceedings of IMACS/ACA98*, 1998. http:www-troja.fjfi.cvut.cz/aca98/sessions/approximate

[211] Pan, V.Y., Numerical Computation of a Polynomial GCD and Extensions. *Information and Computation*, 167(2), 71-85, 2001. Proc. Version in *Proc. 9th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 98)*, 68-77, ACM Press, New York, and SIAM Publications, Philadelphia, 1998.

[212] Pan, V.Y., On approximating complex polynomial zeros: Modified quadtree (Weyl's) construction and improved Newton's iteration. *J. of Complexity*, 16 (1), 213–264, 2000.

[213] Pan, V.Y., *Structured Matrices and Polynomials: Unified Superfast Algorithms*, Birkhäuser/Springer, Boston/New York, 2001.

[214] Pan, V.Y., Univariate Polynomials: Nearly Optimal Algorithms for Factorization and Rootfinding. *Journal of Symbolic Computations*, 33 (5), 701–733, 2002. Proc. version in *Proc. International Symp. on Symbolic and Algebraic Computation (ISSAC 01),* 253–267, ACM Press, New York, 2001.

[215] Pan, V.Y., Randomized acceleration of fundamental matrix computations. *Proc. Symp. on Theoretical Aspects of Computer Science (STACS)*, *Lect. Notes in Comput. Sci.*, 2285, 215–226, Springer, Heidelberg, Germany, 2002.

[216] Pan, V.Y., Can we optimize Toeplitz/Hankel computations? *Proc. of the 5th International Workshop on Computer Algebra in Scientific Computing (CASC 02)*, (E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov, Editors), 253–264, Technische Univ. München, Germany, 2002.

[217] Pan, V.Y., Nearly optimal Toeplitz/Hankel computations. Technical Reports 2002 001 and 2002 017, *Ph.D. Program in Computer Science, the Graduate Center, CUNY*, New York, 2002.

[218] Pan, V.Y., On theoretical and practical acceleration of randomized computation of the determinant of an integer matrix. *Zapiski Nauchnykh Seminarov POMI (in English)*, 316, 163–187, St. Petersburg, Russia, 2004. Also available at http://comet.lehman.cuny.edu/vpan/

[219] Pan, V.Y., Amended DSeSC power method for polynomial root-finding. *Computers and Mathematics (with Applications)*, 49 (9–10), 1515–1524, 2005.

[220] Pan, V.Y., Eigen-solving via reduction to DPR1 matrices. *Computers and Mathematics (with Applications)*, in press.

[221] Pan, V.Y., Branham, S., Rosholt, R. and Zheng, A., Newton's iteration for structured matrices and linear systems of equations. *SIAM volume on Fast Reliable Algorithms for Matrices with Structure* (T. Kailath and A. Sayed editors), Chapter 7, pp.189–210, SIAM Publications, Philadelphia, 1999.

[222] Pan, V.Y., Ivolgin, D., Murphy, B., Rosholt, R.E., Taj-Eddin, I., Tang, Y. and Yan, X., Additive preconditioning and aggregation in matrix computations. *Computers and Math. with Applications*, in press.

[223] Pan, V.Y., Ivolgin, D., Murphy, B., Rosholt, R.E., Tang, Y., and Wang, X., Root-finding with Eigen-solving. In *Symbolic-Numeric Computation*, (Dongming Wang and Lihong Zhi editors), Birkhäuser, Basel/Boston, 2007.

[224] Pan, V.Y., Kunin, M., Rosholt, R.E. and Kodal, H., Homotopic residual correction processes. *Math. of Computation*, 75, 345–368, 2006.

[225] Pan, V.Y., Landowne, E., and Sadikou, A., Univariate polynomial division with a remainder by means of evaluation and interpolation. *Information Processing Letters,* 44, 149–153, 1992.

[226] Pan, V.Y., Murphy, B., Rosholt, R.E. and Wang, X., Toeplitz and Hankel meet Hensel and Newton: nearly optimal algorithms and their practical acceleration with saturated initialization. *Technical Report 2004 013, Ph.D. Program in Computer Science, The Graduate Center, City University of New York*, 2004.

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

[228] Pan, V.Y., Rami, Y. and Wang, X., Structured matrices and Newtons iteration: unified approach. *Linear Algebra and Its Applications*, 343/344, 233–265, 2002.

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

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

[231] Pan, V.Y., Sadikou, A., Landowne, E., and Tiga, O., A new approach to fast polynomial interpolation and multipoint evaluation.

[232] Pan, V.Y. and Schreiber, R., An improved Newton iteration for the generalized inverse of a matrix, with applications. *SIAM Journal on Scientific and Statistical Computing*, 12(5), 1109–1131, 1991.

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

[234] Pan, V.Y., Van Barel, M., Wang, X. and Codevico, G., Iterative inversion of structured matrices. *Theoretical Computer Science*, 315 (2–3), (Special Issue on Algebraic and Numerical Algorithms, I.Z. Emiris, B. Mourrain and V.Y. Pan, editors), 581–592, 2004.

[235] Pan, V.Y. and Wang, X., Inversion of displacement operators. *SIAM J. on Matrix Analysis and Applications*, 24(3), 660–677, 2003.

[236] Pan, V.Y. and Wang, X., On rational number reconstruction and approximation. *SIAM J. on Computing*, 33(2), 502–503, 2004.

[237] Pan, V.Y. and Yu, Y., Certification of numerical computation of the sign of the determinant of a matrix. *Algorithmica*, 30, 708–724, 2001. Proc. version in *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 99)*, 715–724, ACM Press, New York, and SIAM Publications, Philadelphia, 1999.

[238] Pan, V.Y., Zheng, A.L., Huang, X.H. and Yu, Y.Q., Fast multipoint polynomial evaluation and interpolation via computations with structured matrices. *Annals of Numerical Mathematics,* 4, 483–510, 1997.

[239] Parlett, B., *Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, NJ, 1980.

[240] Press, W., Flannery, B., Teukolsky, S., and Vetterling, W., *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1988, and 2nd ed. 1992.

[241] Quinn, M.J., *Parallel Computing: Theory and Practice.* McGraw-Hill, New York, 1994.

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

[243] Reischert, D., Asymptotically fast computation of subresultants. In *ISSAC*, pages 233–240, 1997.

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

[245] Raghavan M. and Roth., B., Solving polynomial systems for the kinematics analysis and synthesis of mechanisms and robot manipulators. *Trans. ASME, Special Issue*, 117, 71–79, 1995.

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

[247] Rosen, D., and Shallit, J., A continued fraction algorithm for approximating all real polynomial roots. *Math. Mag*, 51:112–116, 1978.

[248] Rouillier, F., Solving Zero-Dimensional Systems through the Rational Univariate Representation, *AAECC Journal*, 9, 433–461, 1999.

[249] Rouillier, F., and Zimmermann, P., Efficient isolation of polynomial's real roots. *J. of Computational and Applied Mathematics*, 162(1):33–50, 2004.

[250] Saad, Y., *Iterative Methods for Sparse Linear Systems.* PWS Publishing Co., Boston, 1996 (first edition) and SIAM Publications, Philadelphia, 2003 (second edition).

[251] Schönhage, A., The fundamental theorem of algebra in terms of computational complexity. *Mathematics Department, University of Tübingen*, Germany, 1982.

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

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

[254] Sharma, V., Complexity of real root isolation using Continued Fractions. In C. W. Brown, editor, *Proc. Annual ACM ISSAC*, Waterloo, Canada, 2007.

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

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

[257] Stetter, H., *Numerical polynomial algebra.* SIAM, 2004.

[258] Stewart, G.W., *Matrix Algorithms, Vol I: Basic Decompositions.* SIAM, Philadelphia, 1998.

[259] Stewart, G.W., *Matrix Algorithms, Vol II: Eigensystems.* SIAM, Philadelphia, 1998.

[260] Storjohann, A., High order lifting and integrality certificaiton. *J. of Symbolic Computation*, 36(3–4), 613–648, 2003.

[261] Storjohann, A., The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4), 609–650, 2005.

[262] Sturm, C., Mémoire sur la résolution des equations numériques. *Mém. Savants Étranger*, 6:271–318, 1835.

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

[264] Tarjan, R.E., A unified approach to path problems. *J. of ACM*, 28(3), 577–593, 1981.

[265] Tarjan, R.E., Fast algorithms for solving path problems. *J. of ACM*, 28(3), 594–614, 1981.

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

[267] Tsigaridas, E.P., and Emiris, I.Z., Univariate polynomial real root isolation: Continued fractions revisited. In Y. Azar and T. Erlebach, editors, *Proc. 14th European Symposium of Algorithms (ESA)*, volume 4168 of *LNCS*, pages 817–828, Zurich, Switzerland, 2006. Springer Verlag.

[268] Uspensky, J.V., *Theory of Equations.* McGraw-Hill, 1948.

[269] Vandebril, R., Van Barel, M., Golub, G. and Mastronardi, N., A bibliography on Semiseparable Matrices. *Calcolo*, 42 (3–4), 249–270, 2005.

[270] Vincent, A.J.H., Sur la résolution des équations numériques. *J. Math. Pures Appl.*, 1:341–372, 1836.

[271] van der Vorst, H.A., *Iterative Krylov Methods for Large Linear Systems.* Cambridge University Press, Cambridge, England, 2003.

[272] van der Waerden, B.L., *Modern Algebra,* 3rd ed., F. Ungar, New York, 1950.

[273] Walsh, P.G., The computation of Puiseux expansions and a quantitative version of Runge's theorem on diophantine equations. Ph.D. Thesis, University Waterloo, Waterloo, Canada, 1993.

[274] *Symbolic-Numeric Computation* (Wang, D., and Zhi, L. editors). Birkhäuser, Basel/Boston, 2007.

[275] Wang, X. and Pan, V.Y., Acceleration of Euclidean Algorithm and Rational Number Reconstruction. *SIAM J. of Computing*, 32(2), 548–556, 2003.

[276] Wiedemann, D., Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory* IT–32, 54–62, 1986.

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

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

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

[280] Yap, C.K., *Fundamental Problems of Algorithmic Algebra.* Oxford University Press, New York, 2000.

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

[282] Zippel, R., *Effective Polynomial Computations,* 384. Kluwer Academic, Boston, MA, 1993.

Further Information

The books and journal special issues (2; 10; 27; 29; 33; 41; 72; 97; 115; 118; 155; 213; 274; 282; 257) 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, (80) for solving linear systems of equations, (256), (112), ARPACK, and PARPACK for approximating matrix eigenvalues, and (7) for both of the two latter computational problems. Comprehensive treatment of numerical matrix computations can be found in (126; 258; 259), with extensive bibliography, and there are several more specialized books on them (14; 11; 73; 79; 120; 128; 135; 239; 258; 259; 266; 277) as well as many survey articles (133; 198; 204) and thousands of research articles. Further applications to the graph and combinatorial computations related to linear algebra are cited in "Some Computations Related to Matrix Multiplication" and (205).

Special (more efficient) parallel algorithms have been devised for special classes of matrices, such as sparse (123; 124; 205; 230), banded (79), and dense structured (27; 269). We also refer the reader to (227) on simple but 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 (124; 126; 133) on practical parallel algorithms for matrix computations.

On Symbolic-Numeric Algorithms, see the books (27; 213; 274), surveys (203; 204; 209; 210), a special issue (97), and the bibliography therein.

For the general area of exact computation and the theory behind algebraic algorithms and computer algebra, we refer the reader to (15; 67; 68; 72; 77; 115; 118; 40; 184; 278; 280; 183; 282).

There are a lot of generic software packages for exact computation. We simply mention SYNAPS[1] (191) a `C++` open source library devoted to symbolic and numeric computations with polynomials, algebraic numbers and polynomial systems, NTL[2] a high-performance, portable `C++` library providing data structures

[1]http://www-sop.inria.fr/galaad/software/synaps/
[2]http://www.shoup.net/ntl/

and algorithms for manipulating vectors, matrices, and polynomials over the integers and over finite fields, CORE[3], another `C++` library that provides an API for computations with different levels of accuracy in order to support the Exact Geometric Computation (EGC) approach for numerically robust algorithms, and EXACUS[4] (18), also a `C++` library with algorithm for curves and surfaces that provides exact methods for solving polynomial equations. A highly efficient software tool is FGB/RS[5], which contains algorithms for Gröbner basis computations, the rational univariate representation, and computing certified real solutions of systems of polynomial equalities and inequalities. Finaly, let us also mention LINBOX[6] (83), which is a `C++` library that provides exact and high-performance implementations of linear algebra algorithms.

This chapter does not cover the area of polynomial factorization. We refer the interested reader to (74, Chap. 16), or (115), and the bibliography therein.

The *SIAM Journal on Matrix Analysis and Applications* and Linear Algebra and Its Applications are specialized on Matrix Computations, *Math. of Computations* and *Numerische Math.* are leading among numerous other good journals on numerical computing.

The *Journal of Symbolic Computation* and *Journal of Computational Complexity* specialize on topics in Computer Algebra, which are also covered in the Journal of Pure and Applied Algebra and less regularly in the *J. of Complexity. Theoretical Computer Science* became more open to algebraic–numerical and algebraic–geometric subjects (see particularly (97) and (44)).

The annual *International Symposium on Symbolic and Algebraic Computation (ISSAC)* is devoted to computer algebra, whose topics are also presented at the annual Conference *MEGA* and the annual *ACM Conference on Computational Geometry*, and also frequently at various Computer Science conferences, including STOC, FOCS, and SODA.

Among many conferences on numerical computing, most comprehensive ones are organized under the auspieces of SIAM and ICIAM. International Conferences on Symbolic-Numeric Algorithms can be traced back to 1997 (SNAP in INRIA, Sophia Antipolis), and resumed in Xi'an, China, in 2005, Timishiora, Romania, in 2006 (supported by IEEE), and London, Ontario, Canada, in 2007 (supported by ACM).

The topics of Symbolic Numerical Computation are also represented at the conferences on the *Foundations of Computational Mathematics (FoCM)* (met every 3 years) and occasionally at the ISSAC.

[3]http://cs.nyu.edu/exact/
[4]http://www.mpi-inf.mpg.de/projects/EXACUS/
[5]http://fgbrs.lip6.fr/salsa/Software/
[6]http://www.linalg.org