Algebraic Algorithms

Ioannis Z. Emiris, Victor Y. Pan, Elias Tsigaridas

ALGEBRAIC ALGORITHMS

Ioannis Z. Emiris
Department of Informatics and Telecommunications,
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
Project PolSys, INRIA Paris-Rocquencourt, UPMC & LIP6 CNRS, Paris, France.
elias@polsys.lip6.fr

1 Introduction

Algebraic algorithms deal with numbers, vectors, matrices, polynomials, formal power series, exponential and differential polynomials, rational functions, algebraic sets, curves and surfaces. In this vast area, manipulation with matrices and polynomials is fundamental for modern computations in Sciences, Engineering, and Signal and Image Processing. The list of the respective computational problems includes 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 (which in turn include 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 characteristic and minimal polynomials, determinants, Smith and Frobenius normal forms, ranks, and (generalized) inverses, univariate and multivariate polynomial resultants, Newton's polytopes, greatest common divisors, and least common multiples as well as manipulation with truncated series and algebraic sets.

Such problems can be solved by using the error-free symbolic computations with infinite precision. Computer algebra systems such as Maple and Mathematica compute the solutions based on various nontrivial computational techniques such as modular computations, the Euclidean algorithm and continuous

[1]This material is based on work supported in part by the European Union through Marie-Curie Initial Training Network "SAGA" (ShApes, Geometry, Algebra), with FP7-PEOPLE contract PITN-GA-2008-214584 (first author), by NSF Grant CCF-1116736 and PSC CUNY Awards 63153–0041 and 64512–0042 (second author), by the Danish Agency for Science, Technology and Innovation (postdoctoral grant), Danish NRF and NSF of China (grant 61061130540), CFEM, and the Danish Strategic Research Council (third author). Sections 3.5, 5, 6 and "Further comments" have been written jointly by all authors, Section 4 by the first author, the other sections by the second author.

fraction approximation, Hensel's and Newton's lifting, Chinese Remainder algorithm, elimination and resultant methods, and Gröbner bases computation. The price to achieve perfect accuracy is the substantial memory space and computer time required to support the computations.

The alternative numerical methods rely on operations with binary or decimal numbers truncated or rounded to a fixed precision. Operating with the IEEE standard floating point numbers represented with single or double precision enables much faster computations using much smaller memory but requires theoretical and/or experimental study of the impact of rounding errors on the output. The study involves forward and backward error analysis, linear and nonlinear operators, and advanced techniques from approximation and perturbation theories. Solution of some problems involves more costly computations with extended precision. The resulting algorithms support high performance libraries and packages of subroutines such as those in Matlab, NAG SMP, LAPACK, ScaLAPACK, ARPACK, PARPACK, MPSolve, and EigenSolve.

In this chapter we cover both approaches, whose combination frequently increases their power and enables more effective computations. We focus on the algebraic algorithms in the large, popular and highly important fields of matrix computations and root-finding for univariate polynomials and systems of multivariate polynomials. We cover part of these huge subjects and include basic bibliography for further study. To meet space limitation we cite books, surveys, and comprehensive articles with pointers to further references, rather than including all the original technical papers. Our expositions in Sections 2 and 3 follow the line of the first surveys in this area in [163, 168, 173, 174, 175].

We state the complexity bounds under the random access machine (RAM) model of computation [1, 96]. 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 single or double) precision, which fits the size of a computer word, and then the arithmetic model turns into the *word* model [96]. In other cases we allow working with extended precision and assume the *Boolean* or *bit* model, assigning a unit cost to every Boolean or bitwise operation. This accounts for both arithmetic operations and the length (precision) of the operands. We denote the bounds for this complexity by $\mathcal{O}_B(\cdot)$. We explicitly specify whether we use the arithmetic, word, or Boolean model unless this is clear from the context.

We write **ops** for "arithmetic operations", "log" for "$\log_2$" unless specified otherwise, and $\tilde{\mathcal{O}}_B(\cdot)$ to show that we are ignoring the logarithmic factors.

2 Matrix Computations

Matrix computations is the most popular and a highly important area of scientific and engineering computing. Most frequently they are performed nu-

merically, with values represented using the IEEE standard single or double precision. In the chapter of this size we must omit or just barely touch on many important subjects of this field. The reader can find further material and bibliography in the surveys [163, 168] and the books [6, 9, 22, 57, 63, 65, 103, 110, 178, 223, 228, 240] and for more specific subject areas in [6, 103, 223, 228, 238, 240] on eigendecomposition and SVD, [9, 57, 63, 103, 110, 223, 228] on other numerical matrix factorizations, [24, 130] on the over- and under-determined linear systems, their least-squares solution, and various other numerical computations with singular matrices, [106] on randomized matrix computations, [114, 178] on structured matrix computations, [22, 103, 172, 211] on parallel matrix algorithms, and [44, 48, 67, 68, 96, 97, 115, 116, 169, 172, 202, 183, 227, 239] on "Error-free Rational Matrix Computations", including computations over finite fields, rings, and semirings that produce solutions to linear systems of equations, matrix inverses, ranks, determinants, characteristic and minimal 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 $A = [\ a_{i,j}\ ,\ i = 1, \dots, m;\ j = 1, \dots, n\ ]$ is also denoted $[a_{i,j}]_{i,j=1}^{m,n}$ and $[\mathbf{A}_1 \mid \dots \mid \mathbf{A}_m]$; it 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. Matrix A is a column vector if $n = 1$ and a row vector if $m = 1$. Vector $\mathbf{v} = [v_i]_{i=1}^{n}$ is an nth dimensional column vector. The straightforward algorithm computes the product $A\mathbf{v}$ by performing $(2n-1)m$ ops; this is optimal for general (dense unstructured) $m \times n$ matrices, represented with their entries, but numerous applications involve structured matrices represented with much fewer than mn scalar values. A matrix is singular if its product by some vectors vanish; they form its *null space*.

An $m \times n$ matrix is *sparse* if it is filled mostly with zeros, having only $\phi = o(mn)$ nonzero entries. An important class is the matrices associated with graphs that have *families of small separators* [102, 134]. This includes *banded* matrices $[b_{i,j}]_{i,j}$ with small *bandwidth* $2w+1$ such that $b_{i,j} = 0$ unless $|i-j| \le w$. A sparse matrix can be stored economically by using appropriate data structures and can be multiplied by a vector fast, in $2\phi - m$ ops. Sparse matrices arise in many important applications, e.g., to solving ordinary and partial differential equations (ODEs and PDEs) and graph computations.

Dense structured $n \times n$ matrices are usually defined by $O(n)$ parameters, and one can apply FFT to multiply such matrix by a vector by using $O(n \log n)$ or $O(n \log^2 n)$ ops [178]. Such matrices are omnipresent in applications in signal and image processing, coding, ODEs, PDEs, particle simulation, and Markov chains. Most popular among them are the *Toeplitz matrices* $T = [t_{i,j}]_{i,j=1}^{m,n}$ and the *Hankel matrices* $H = [h_{i,j}]_{i,j=1}^{m,n}$ where $t_{i,j} = t_{i+1,j+1}$ and $h_{i,j} = h_{i+1,j-1}$ for all i and j in the range of their definition. Each such matrix is defined by $m+n-1$ entries of its first row and first (or last) column. Products $T\mathbf{v}$ and $H\mathbf{v}$

can be equivalently written as polynomial products or vector convolutions; their FFT-based computation takes $O((m+n)\log(m+n))$ ops per product [1, 22, 178]. Many other fundamental computations with Toeplitz and other structured matrices can be linked to polynomial computations enabling acceleration in both areas of computing [18, 19, 20, 21, 22, 25, 81, 85, 150, 151, 152, 168, 172, 178, 187, 208, 209]. Similar properties hold for Vandermonde matrices $V = [v_i^j]_{i,j=0}^{m-1,n-1}$ and Cauchy matrices $C = [\frac{1}{s_i - t_j}{}_{i,j}]_{i,j=1}^{m,n}$ where s_i and t_j denote $m+n$ distinct scalars.

One can extend the structures of *Hankel*, *Bézout, Sylvester, Frobenius (companion)*, *Vandermonde,* and *Cauchy* matrices to more general classes of matrices by associating linear displacement operators. (See [22, 178] for the details and the bibliography.) The important classes of *semiseparable*, *quasiseparable* and other *rank structured* $m \times n$ matrices generalize banded matrices and their inverses; they are expressed by $O(m+n)$ parameters and can be multiplied by vectors by performing $O(m+n)$ ops [69, 232].

2.2 Matrix Multiplication, Factorization, Randomization

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 $m = n = p$. This upper bound is not sharp. Strassen decreased it to $O(n^{2.81})$ ops in 1969. His result was first improved in [162] and 10 times afterward, most recently by Coppersmith and Winograd in [49], Stothers in [225], and Vasilevska Williams in [234], who use Cn^ω ops for $\omega < 2.376$, $\omega < 2.374$ and $\omega < 2.3727$, respectively. Due to the huge overhead constants C, however, we have that $Cn^\omega < 2n^3$ only for enormous values n. The well recognized group-theoretic techniques [45] enable a distinct description of the matrix multiplication algorithms, but so far have only supported the same upper bounds on the complexity as the preceding works. References [225] and [234] extend the algorithms given in Reference [49], which in turn combines arithmetic progression technique with the previous advanced techniques. Each technique, however, contributes to a dramatic increase of the overhead constant that makes the resulting algorithms practically noncompetitive.

The only exception is the *trilinear aggregating* technique of [161] (cf. [163]), which alone supports the exponent 2.7753 [128] and together with the Any Precision Approximation (APA) techniques of [163] was an indispensable ingredient of all algorithms that have beaten Strassen's exponent 2.81 of 1969. The triple product property (TPP), which is the basis of [45], may very well have a natural link to trilinear aggregating, although the descriptions available for the two approaches are distinct. For matrices of realistic sizes the numerical algorithms in [118], relying on trilinear aggregating, use about as many ops as the algorithms of Strassen 1969 and Winograd 1971 but need substantially less memory space and are more stable numerically.

The exponent ω of matrix multiplication is fundamental for the theory of computing because $O(n^\omega)$ or $O(n^\omega \log n)$ bounds the complexity of many impor-

tant matrix computations such as the computation of det A, the **determinant** of an $n \times n$ matrix A; its *inverse* A^{-1} (where det $A \neq 0$); 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; a submatrix of A having the maximal rank, the solution vector $\mathbf{x} = A^{-1}\ \mathbf{v}$ to a nonsingular *linear system of equations* $A\ \mathbf{x} = \mathbf{v}$, and various *orthogonal* and *triangular factorizations* of the matrix A, as well as various *computations with singular matrices* and seemingly unrelated combinatorial and graph computations, e.g., pattern recognition or computing all pair shortest distances in a graph [22, p. 222] or its transitive closure [1]. Consequently, all these operations use $O(n^\omega)$ ops where theoretically $\omega < 2.3727$ [1, chap.6], [22, chap. 2]. In practice, however, the solution of all these problems takes order of n^3 ops, because of the huge overhead constant C of all known algorithms that multiply $n \times n$ matrices in Cn^ω ops for $\omega < 2.775$, the overhead of the reduction to a matrix multiplication problem, the memory space requirements, and numerical stability problems [103].

Moreover, the straightforward algorithm for matrix multiplication remains the users' choice because it is highly effective on parallel and pipeline architectures [103, 211]; on many computers it supersedes even the so called "superfast" algorithms, which 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 with their short generator matrices having $O(n)$ entries [178].

Numerous important practical problems have been reduced to matrix multiplication because it is so effective. This has also motivated the development of block matrix algorithms (called *level-three BLAS*, which is the acronym for Basic Linear Algebra Subprograms).

Devising asymptotically fast matrix multipliers, however, had independent technical interest. E.g., trilinear aggregating was a nontrivial decomposition of the 3-dimensional tensor associated with matrix multiplication, and [161] was the first of now numerous examples where nontrivial tensor decompositions enable dramatic acceleration of important matrix computations [124, 137, 160].

The two basic techniques below extend matrix multiplication. Hereafter O denotes matrices filled with zeros; I is the square identity matrices, with ones on the diagonal and zeros elsewhere.

Suppose we seek 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 $\mathbf{v}$ [103, 104, 239]; in block Krylov computations the vector $\mathbf{v}$ is replaced by a matrix. The straightforward algorithm uses $(2n-1)n(k-1)$ ops, that is about $2n^3$ for $k = n$. An alternative algorithm first computes the matrix powers

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

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

$$B \quad \mathbf{v}\ ,$$

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

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 [170]. According to these techniques, Krylov sequence is recovered from the solution to the associated linear system $(I - A)\ \mathbf{x} = \mathbf{v}$, which is solved fast in the case of a special matrix A.

Another basic idea of matrix algorithms is to represent the input matrix A as a block matrix and to operate with its blocks rather than with its entries. E.g., one can compute $\det A$ and A^{-1} by first factorizing A as a 2×2 block matrix,

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

where $S = A_{1,1} - A_{1,0}A_{0,0}^{-1}A_{0,1}$. The 2×2 block triangular factors are readily invertible, $\det A = (\det A_{0,0}) \det S$ and $(BCD)^{-1} = D^{-1}C^{-1}B^{-1}$, and so the cited tasks for the input A are reduced to the same tasks 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 highly important class of *symmetric positive definite* matrices $A = C^TC$, $\det C \neq 0$, and can be also achieved by means of symmetrization, pivoting, or randomization [1, chap. 6], [22, chap. 2], [178, sects. 5.5 and 5.6]). Recursive application of (1) should produce the LDU factorization $A = LDU$ where the matrices L and U^T are lower triangular and D diagonal. Having this factorization computed , we can readily solve linear systems $A\mathbf{x}_i = \mathbf{b}_i$ for various vectors $\mathbf{b}_i$, by using about $2n^2$ ops for each i, rather than $\frac{2}{3}n^3 + O(n^2)$ in Gaussian elimination.

Factorizations (including PLU, QR, QRP, QLP, CS, LR, Cholesky factorizations and eigenvalue and singular value decompositions) are the most basic tool of matrix computations (see, e.g., [223]), recently made even more powerful with **randomization** (see [106, 186, 188, 193, 194, 195, 196, 198], and the bibliography therein). It is well known that random matrices tend to be nonsingular and well conditioned (see, e.g., [218]), that is they lie far from singular matrices and therefore [103, 110, 223] are not sensitive to rounding errors and are suitable for numerical computations. The solution $\mathbf{x} = A^{-1}\mathbf{b}$ of a nonsingular linear system $A\mathbf{x} = \mathbf{b}$ of n equations can be obtained with a precision p_{out} in $O_{\tilde{B}}(n^3p + n^2p_{\text{out}})$ Boolean time for a fixed low precision p provided the matrix A is well conditioned; that accelerates Gaussian elimination by order of magnitude for large $n + p_{\text{out}}$. Recent randomization techniques in [106, 186, 188, 193, 194, 195, 196, 198] extend this property to much larger class of linear systems and enhance the power of various other matrix computations with singular or ill conditioned matrices, e.g., their approximation by low-rank

matrices, computing a basis for the null space of a singular matrix, and approximating such bases for nearly singular matrices. Similar results have been proved for rectangular and Toeplitz matrices.

We refer the reader to [106, 218, 99] on impressive progress achieved in many other areas of matrix computations by means of randomization techniques.

2.3 Solution of linear systems of equations

The solution of a linear system of n equations, $A\mathbf{x} = \mathbf{b}$ is the most frequent operation in scientific and engineering computations and is highly important theoretically. Gaussian elimination solves such a system by applying $(2/3)n^3 + O(n^2)$ ops.

Both Gaussian elimination and *(Block) Cyclic Reduction* use $O(nw^2)$ ops for banded linear systems with bandwidth $O(w)$. One can solve rank structured linear systems in $O(n)$ ops [69, 232]; generalized nested dissection uses $O(n^{1.5})$ flops for the inputs associated with small separator families [134, 169, 202].

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 as well as matrices with similar structures. The MBA divide-and-conquer "superfast" algorithm (due to Morf 1974/1980 and Bitmead and Anderson 1980) solves nonsingular structured linear systems of n equations in $O(n \log^2 n)$ ops by applying the recursive 2×2 block factorization (1) and preserving matrix structure [22, 178, 191, 205]. In the presence of rounding errors, however, Gaussian elimination, the MBA and Cyclic Reduction algorithms easily fail unless one applies pivoting, that is interchanges the equations (and sometimes unknowns) to avoid divisions by absolutely small numbers. A by-product is the factorization $A = PLU$ or $A = PLUP'$, for lower triangular matrices L and U^T and permutation matrices P and P'.

Pivoting, however, takes its toll. It "usually degrades the performance" [103, page 119] by interrupting the string of arithmetic computations with the foreign operations of comparisons, is not friendly to block matrix algorithms and updating input matrices, hinders parallel processing and pipelining, and tends to destroy structure and sparseness, except for the inputs that have Cauchy-like and Vandermonde-like structure. The latter exceptional classes have been extended to the inputs with structures of Toeplitz/Hankel type by means of *displacement transformation* [167, 178]. The users welcome this numerical stabilization, even though it slows down the MBA algorithm by a factor of $n/\log^2 n$, that is from "superfast" to "fast", which is still by a factor of n faster than the solution for general unstructured inputs, which takes order n^3 ops.

Can we avoid pivoting in numerical algorithms with rounding for general, sparse and structured linear systems to achieve both numerical stability and superfast performance? Yes, for the important classes where the input matrices $A = (a_{ij})_{i,j}$ are diagonally dominant, that is $|a_{ii}| > \sum_{i \neq j} |a_{ij}|$ or $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for all i, or symmetric positive definite, that is $A = C^T C$ for a nonsingular matrix C. To these input classes Gaussian elimination, Cyclic Reduction,

and the MBA algorithm can be safely applied with rounding and with no pivoting. For some other classes of sparse and positive definite linear systems, pivoting has been modified into nested dissection, Markowitz heuristic rule, and other techniques that preserve sparseness during the elimination yielding faster solution without causing numerical problems [63, 101, 134, 169, 202]. Can we extend these benefits to other input matrix classes?

Every nonsingular linear system $A\ \mathbf{x} = \mathbf{b}$ is equivalent to the symmetric positive definite ones $A^TA\ \mathbf{x} = A^T\mathbf{b}$ and $A\ A^T\mathbf{y} = \mathbf{b}$ where $\mathbf{x} = A\mathbf{y}$, but great caution is recommended in such symmetrizations because the condition number $\kappa(A) = ||A||_2||A^{-1}||_2 \geq 1$ is squared in the transition to the matrices A^TA and AA^T, which means growing propagation and magnification of rounding errors.

There are two superior directions. The algorithms of [195, 196, 199] avoid pivoting for general and structured linear systems by applying randomization. These techniques are recent, proposed in [186, Section 12.2], but their effectiveness has formal and experimental support.

A popular classical alternative to Gaussian elimination is the iterative solution, e.g., by means of the Conjugate Gradient and GMRES algorithms [11, 103, 104, 233]. They compute sufficiently long Krylov sequences (defined in the previous section) and then approximate the solution with linear combinations $\sum_i c_iA^i\mathbf{b}$ or $\sum_i c_i(A^TA)^iA^T\mathbf{b}$ for proper coefficients c_i. The cost of computing the product of the matrix A or A^TA by a vector is dominant, but it is small for structured and sparse matrices A. One can even call a matrix sparse or structured if and only if it can be multiplied by a vector fast.

Fast convergence to the solution is critical. It is not generally guaranteed but proved for some important classes of input matrices. The major challenge are the extension of these classes and the design of powerful methods for special input classes, notably *multilevel methods* (based on the *algebraic multigrid*) [149, 140, 201] and tensor decompositions [160, 124], highly effective for many linear systems arising in discretization of ODEs, PDEs, and integral equations.

Preconditioning of the input matrices at a low computational cost accelerates convergence of iterations for many important classes of sparse and structured linear systems [11, 104], and more recently, based on randomized preconditioning, for quite general as well as structured linear systems [186, 188, 193, 194, 195, 196, 198].

One can iteratively approximate the inverse or pseudo-inverse of a matrix [103, Section 5.5.4] by means of Newton's iteration $X_{i+1} = 2X_i - X_iMX_i$, $i = 0, 1, \ldots$. We have $I - MX_{i+1} = (I - MX_i)^2 = (I - MX_0)^{2^{i+1}}$; therefore, the residual norm $||I - MX_i||$ is squared in every iteration step, $||I - MX_i|| \leq ||I - MX_0||^{2^i}$ for $i = 1, 2, \ldots$, and so convergence is very fast unless $||I - MX_0|| \geq 1$ or is near 1. The cost of two matrix multiplications is dominant per an iteration step; this makes the computation fast on multiprocessors as well as in the case of structured matrices M and X_i. See more on Newton's iteration, including the study of its initialization, convergence, and preserving displacement matrix structure, in [178, chapters 4 and 6], [203, 185, 200, 204, 189, 182].

2.4 Symbolic Matrix Computations

Rational matrix computations for a rational or integer input (such as the solution of a linear system and computing the determinant of a matrix) can be performed with no errors. To decrease the computational cost, one should control the growth of the precision of computing. Some special techniques achieve this in rational Gaussian elimination [8, 97]. As a more fundamental tool one can reduce the computations modulo a sufficiently large integer m to obtain the rational or integer output values $z = p/q$ (e.g., the solution vector for a linear system) modulo m. Then we can recover z from two integers m and $z \bmod m$ by applying the continued fraction approximation algorithm, in other contexts called Euclidean algorithm [96, 237]. Instead we can readily obtain $z = z \bmod m$ if $z \bmod m < r$ or $z = -m + z \bmod m$ if $z \bmod m < r$ otherwise, provided we know that the integer z lies in the range $[-r, r]$ and if $m > 2r$.

Computing the determinant of an integer matrix, we can choose the modulus m based on Hadamard's bound. A nonsingular linear system $A\mathbf{x} = \mathbf{v}$ can become singular after the reduction modulo a prime p but only with a low probability for a random choice of a prime p in a fixed sufficiently large interval as well as for a reasonably large power of two and a random integer matrix [205].

One can choose $m = m_1 m_2 \cdots m_k$ for pairwise relatively prime integers $m_1, m_2, \ldots, m_k$ (we call them *coprimes*), then compute z modulo all these coprimes, and finally recover z by applying the Chinese Remainder algorithm [1, 96]. The error-free computations modulo m_i require the precision of $\log m_i$ bits; the cost of computing the values $z \bmod m_i$ for $i = 1, \ldots, k$ dominates the cost of the subsequent recovery of the value $z \bmod m$.

Alternatively one can apply *p-adic* (*Newton–Hensel*) *lifting* [96]. For solving linear systems of equations and matrix inversion they can be viewed as the symbolic counterparts to iterative refinement and Newton's iteration of the previous section, both well known in numerical linear algebra [183].

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, e.g., Toeplitz, Hankel, Vandermonde, Cauchy. Hensel's lifting enables the solution in nearly optimal time under both Boolean and word models [183]. We can choose p being a power of two and use computations in the binary mode. Reference [70] discusses lifting for sparse linear systems.

2.5 Computing the Sign and the Value of a Determinant

The value or 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. Faster numerical methods are preferred as long as the correctness of the output can be certified. In the customary *arithmetic filtering* approach, one applies fast numerical methods as long as they work and, in the rare cases when they fail, shifts to the slower symbolic methods. For fast numerical computation of $\det A$ one can employ factorizations $A = PLUP'$ (see Section 2.2) or $A = QR$ [46, 103], precondition the matrix A [186], and then certify the output sign [206].

If A is a rational or integer matrix, then the Chinese Remainder algorithm of the previous subsection is highly effective, particularly using heuristics for working modulo m for m much smaller than Hadamard's bound on $|\det A|$ [27].

Alternatively [165, 166, 71], one can solve linear systems $A\mathbf{y}(i) = \mathbf{b}(i)$ for random vectors $\mathbf{b}(i)$ and then apply Hensel's lifting to recover $\det A$ as a least common denominator of the rational components of all $\mathbf{y}(i)$.

Storjohann in [224] advanced randomized Newton's lifting to yield $\det A$ more directly in the optimal asymptotic Boolean time $\mathcal{O}_B(n^{\omega+1})$ for $\omega < 2.3727$. Wiedemann in 1986, Coppersmith in 1994, and a number of their successors compute $\det A$ by extending the Lanczos and block Lanczos classical algorithms. This is particularly effective for sparse or structured matrices A and in further extension to multivariate determinants and resultants (cf. [117, 85, 86, 180]).

3 Polynomial Root-Finding and Factorization

3.1 Computational Complexity Issues

Approximate solution of an nth degree polynomial equation,

$$p(x) = \sum_{i=0}^{n} p_i\, x^i = p_n \prod_{j=1}^{n} (x - z_j) = 0\ , \qquad p_n \neq 0, \tag{2}$$

that is the approximation of the roots $z_1, \ldots, z_n$ for given coefficients $p_0, \ldots, p_n$, is a classical problem that has greatly influenced the development of mathematics and computational mathematics throughout four millennia, since the Sumerian times [173, 174]. The problem remains highly important for the theory and practice of the present day algebraic and algebraic/geometric computation, and new root-finding algorithms appear every year [141, 142, 143, 144].

To approximate even a single root of a monic polynomial $p(x)$ within error bound 2^{-b} we must process at least $(n+1)nb/2$ bits of the input coefficients $p_0, \ldots, p_{n-1}$. Indeed perturb the x-free coefficient of the polynomial $(x - 6/7)^n$ by 2^{-bn}. Then the root $x = 6/7$ jumps by 2^{-b}, and similarly if we perturb the coefficients p_i by $2^{(i-n)b}$ for $i = 1, \ldots, n-1$. Thus 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, $i = 0, 1, \ldots, n-1$. We need at least $\lceil (n+1)nb/4 \rceil$ bitwise operations to process these bits, each operation having at most two input bits.

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 for b of order $n \log n$ or higher, that is we can approximate all roots about as fast as we write down the input. We achieve this by applying the *divide-and-conquer algorithms* in [171, 173, 179] (see [123, 157, 219] 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. Then the polynomials $F(x)$ and $G(x)$ are recursively factorized 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. For approximation of a single root see the competitive algorithms of [177].

It is interesting that, up to polylog factors, both lower and upper bounds on the Boolean time decrease to bn [179] if we only seek the factorization of $p(x)$, that is, if instead of the roots z_j, we compute scalars a_j and b_j such that

$$||p(x) - \prod_{j=1}^{n} (a_j x - c_j)|| < 2^{-b} ||p(x)|| \tag{3}$$

for the polynomial norm $||\sum_i q_i x^i|| = \sum_i |q_i|$.

The *isolation of the zeros* of a polynomial $p(x)$ of (2) having integer coefficients and simple zeros is the computation of n disjoint discs, each containing exactly one root of $p(x)$. This can be a bottleneck stage of root approximation because one can contract such discs by performing a few subdivisions and then apply numerical iterations (such as Newton's) that would very rapidly approximate the isolated zeros within a required tolerance. Reference [184] yields even faster refinement by extending the techniques of [171, 173, 179].

Based on the gap theorem of Mahler (1964) (recently advanced in [84]), Schönhage in [219, Section 20] has reduced the isolation problem to computing factorization (3) for $b = \lceil (2n+1)(l+1+\log(n+1)) \rceil$ where l is the maximal coefficient length, that is the minimum integer such that $|\Re(p_j)| < 2^l$ and $|\Im(p_j)| < 2^l$ for $j = 0, 1, \ldots, n$. Combining the cited algorithms of [171, 173, 179] with this reduction yields

Theorem 3.1 *Let polynomial $p(x)$ of (2) have n distinct simple zeros and integer coefficients in the range $[-2^\tau, 2^\tau]$. Then one can isolate the n zeros of $p(x)$ from each other at the Boolean cost $\tilde{\mathcal{O}}_B(n^2\tau)$.*

The algorithms of [171, 173, 179] incorporate the techniques of [157, 219], but advance them and support substantially smaller upper bounds on the computational complexity. In particular these algorithms decrease by a factor of n the estimates of [219, Theorems 2.1, 19.2 and 20.1] on the Boolean complexity of polynomial factorization, root approximation and root isolation.

3.2 Root-Finding via Functional Iterations

About the same record complexity estimates for root-finding would be also supported by some *functional iteration* algorithms if one assumes their convergence rate defined by ample empirical evidence, although never proved formally. The users accept such an evidence instead of the proof and prefer the latter algorithms because they are easy to program and have been carefully implemented; like the algorithms of [171, 173, 177, 179] they allow tuning the precision of computing to the precision required for every output root, which is higher for clustered and multiple roots than for 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 [136], *Laguerre's method* [94, 107], and the *Jenkins–Traub algorithm* [112]. One can deflate the input polynomial via its numerical division by $x-z$ to extend these algorithms to approximating a small number of other roots. If one deflates many roots, the coefficients of the remaining factor can grow large as, e.g., in the divisor of the polynomial $p(x) = x^{1000} + 1$ that has degree 498 and shares with $p(x)$ all its roots having positive real parts.

For the approximation of all roots, a good option is the Weierstrass–Durand–Kerner's (hereafter *WDK*) algorithm, defined by the recurrence

$$z_j(l+1) = z_j(l) - \frac{p\left(z_j(l)\right)}{p_n \prod_{i \neq j} \left(z_j(l) - z_i(l)\right)} \ , \quad j = 1, \dots, n, \quad l = 0, 1, \dots \ . \tag{4}$$

It has excellent empirical global convergence. Reference [209] links it to polynomial factorization and adjusts it to approximating a single root in $O(n)$ ops per step.

A customary choice of n initial approximations $z_j(0)$ to the n roots of the polynomial $p(x)$ (see [17] for a heuristic alternative) is given by $z_j(0) = r\ t\exp(2\pi\sqrt{-1}/n) \ , \quad j = 1, \dots, n$. Here $t > 1$ is a fixed scalar and r is an upper bound on the root radius, such that all roots z_j lie in the disc $\{x : \ |x| = r\}$ on the complex plane. This holds, e.g., for

$$r = 2 \max_{i<n} |p_i/p_n|^{\frac{1}{n-i}} \ . \tag{5}$$

For a fixed l and for all j the computation in (4) uses $O(n^2)$ ops. We can use just $O(n \log^2 n)$ ops if we apply fast multipoint polynomial evaluation algorithms based of fast FFT based polynomial division [1, 22, 26, 178, 190], but then we would face numerical stability problems.

As with Newton's, Laguerre's, Jenkins–Traub's algorithms and the Inverse Power iteration in [18, 208], one can employ this variant of the WDK to approximate many or all roots of $p(x)$ without deflation. Toward this goal, one can concurrently apply the algorithm at sufficiently many distinct initial points $z_j(0) = r\ t\exp(2\pi\sqrt{-1}/N) \ , \quad j = 1, \dots, N \geq n$ (on a large circle for large t) or according to [17]. The work can be distributed among processors that do not need to interact with each other until they compute the roots.

See [141, 142, 143, 144, 173] and references therein on this and other effective functional iteration algorithms. Reference [17] covers MPSolve, the most effective current root-finding subroutines, based on Ehrlich–Aberth's algorithm.

3.3 Matrix Methods for Polynomial Root-Finding

By cautiously avoiding numerical problems [103, Sec.7.4.6], one can approximate the roots of $p(x)$ as the eigenvalues of the associated (generalized) companion matrices, that is matrices having characteristic polynomial $p(x)$. Then one can employ numerically stable methods and the excellent software available for matrix computations, such as the QR celebrated algorithm. E.g., Matlab's subroutine *roots* applies it to the companion matrix of a polynomial. Malek and Vaillancourt (1995), and Fortune [93] and in his root-finding package EigenSolve, apply it to other generalized companion matrices and update them when the approximations to the roots are improved.

The algorithms of [18, 19, 20, 181, 16, 231, 208, 197] exploit the structure of (generalized) companion matrices, e.g., where they are diagonal plus rank-one (hereafter *DPR1*) matrices, to accelerate the eigenvalue computations. The papers [18, 208] apply and extend the Inverse Power method [103, Section 7.6.1]; they exploit matrix structure, simplify the customary use of Rayleigh quotients for updating approximate eigenvalues, and apply special preprocessing techniques. For both companion and DPR1 inputs the resulting algorithms use linear space and linear arithmetic time per iteration step, enable dramatic parallel acceleration, and deflate the input in $O(n)$ ops; for DPR1 matrices repeated deflation can produce all n roots with no numerical problems.

The algorithms of [19, 20, 16, 231] employ the QR algorithm, but decrease the arithmetic time per iteration step from quadratic to linear by exploiting the rank matrix structure of companion matrices. Substantial further refinement of these techniques is required to make them competitive with MPSolve. See [244] on recent progress.

The papers [181, 197] advance Cardinal's polynomial root-finders of 1996, based on repeated squaring. Each squaring is reduced to performing a small number of FFTs and thus uses order $n \log n$ ops. One can weigh potential advantage of convergence to nonlinear factors of $p(x)$, representing multiple roots or root clusters, at the price of increasing the time per step by a factor of $\log n$ versus the Inverse Power method, advanced for root-finding in [18, 208].

3.4 Extension to Approximate Polynomial GCDs

Reference [176] combines polynomial root-finders with algorithms for bipartite matching to compute 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-neighborhood of the input polynomials for a fixed positive ϵ. Approximate GCDs are required in computer vision, algebraic geometry, computer modeling, and control. For a single example, GCD defines the intersection of two algebraic curves defined by

the two input polynomials, and approximate GCD does this under input perturbations of small norms. See [15] on the bibliography on approximate GCDs, but see [167, 178, 195] on the structured matrix algorithms involved.

3.5 Univariate Real Root Isolation and Approximation

In some algebraic and geometric computations, the input polynomial $p(x)$ has real coefficients, and only its real roots must be approximated. One of the fastest real root-finder in the current practice is still MPSolve, which uses almost the same running time for real roots as for all complex roots. This can be quite vexing, because very frequently the real roots make up only a small fraction of all roots [78]. Recently, however, the challenge was taken in the papers [208, 197], whose numerical iterations are directed to converge to real and nearly real roots. This promises acceleration by a factor of d/r where the input polynomial has d roots, of which r roots are real or nearly real. In the rest of this section we cover an alternative direction, that is real root-finding by means of isolation of the real roots of a polynomial.

We write $p(x) = a_d\, x^d + \cdots + a_1\, x + a_0$, assume integral coefficients with the maximum bit size $\tau = 1 + \max_{i \le d}\{\lg |a_i|\}$, and seek isolation of real roots, that is seek real line intervals with rational endpoints, each containing exactly one real root. We may seek also the root's multiplicity. We assume rational algorithms, that is, error-free algorithms that operate with rational numbers.

If all roots of $p(x)$ are simple, then the minimal distance between them, the *separation bound*, is at most $b = d^{-(d+2)/2}(d+1)^{(1-d)/2}2^{\tau(1-d)}$, or roughly $2^{-\widetilde{\mathcal{O}}(d\tau)}$ (e.g., [147]), and we isolate real roots as soon as we approximate them within less than $b/2$. Effective solution algorithms rely on Continued Fractions (see below), having highly competitive implementation in SYNAPS [153, 109] and its descendant REALROOT, a package of MATHEMAGIX, on the Descartes' rule of signs, and the Sturm or Sturm–Habicht sequences.

Theorem 3.2 *The rational algorithms discussed in the sequel isolate all r real roots of $p(x)$ in $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ bitwise ops. Under certain probability distributions for the coefficients, they are expected to use $\widetilde{\mathcal{O}}_B(d^3\tau)$ or $\widetilde{\mathcal{O}}_B(rd^2\tau)$.*

The bounds exceed those of Theorem 3.1, but rational solvers are heavily in use, have long and respected history, and are of independent technical interest. Most popular are the subdivision algorithms, such as STURM, DESCARTES and BERNSTEIN. By mimicking binary search, they repeatedly subdivide an initial interval that contains all real roots until every tested interval contains at most one real root. They differ in the way of counting the real roots in an interval.

The algorithm STURM (due to Sturm 1835) is the closest to binary search; it produces isolating intervals and root multiplicities at the cost $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ [64, 83]; see [78] on the decrease of the expected cost to $\widetilde{\mathcal{O}}_B(rd^2\tau)$.

The complexity of both algorithms DESCARTES and BERNSTEIN is $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ [73, 83]. Both rely on Descartes' rule of sign, but the BERNSTEIN algorithm also employs the Bernstein basis polynomial representation. See [235, 2] on the

theory and history of DESCARTES, [47, 216, 72, 146, 217] on its modern versions, and [83, 156] and the references therein on the BERNSTEIN algorithm.

The Continued Fraction algorithm, CF, computes the continued fraction expansions of the real roots of the polynomial. The first formulation of the algorithm is due to Vincent. By Vincent's theorem repeated transforms $x \mapsto c + \frac{1}{x}$ eventually yield a polynomial with zero or one sign variation and thus (by Descartes' rule) with zero or resp. one real root in $(0, \infty)$. In the latter case the inverse transformation computes an isolating interval. Moreover, the c's in the transform correspond to the partial quotients of the continued fraction expansion of the real root. Variants differ in the way they compute the partial quotients.

Recent algorithms control the growth of coefficient bit-size and decrease the bit-complexity from exponential (of Vincent) to $\widetilde{\mathcal{O}}_B(d^3\tau)$ expected and $\widetilde{\mathcal{O}}_B(d^4\tau^2)$ worst-case bit complexity. See [230], [220], [145, 229], and the references therein on these results, history and variants of CP algorithms.

4 Systems of Nonlinear Equations

Given a system $\{p_1(x_1, \ldots, x_n), \ldots, p_r(x_1, \ldots, x_n)\}$ of nonlinear polynomials with rational coefficients, the n-tuple of complex numbers $(a_1, \ldots, a_n)$ is a solution of the system if $p_i(a_1, \ldots, a_n) = 0$, $1 \leq i \leq r$. 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. In this section, we explore the problem of solving a well-constrained system of nonlinear equations, namely when $r = n$, which is the typical case in applications. 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. [23, 82, 125]. 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.

Resultants, as explained below, formally express the solvability of algebraic systems with $r = n+1$; solving a well-constrained system reduces to a resultant computation as illustrated in the sequel. The *Sylvester resultant method* is the technique most frequently utilized for determining a common root of two polynomial equations in one variable. However, using the Sylvester method successively to solve a system of multivariate polynomials proves to be inefficient.

It is more efficient to eliminate n variables together from $n+1$ polynomials, thus, leading to the notion of the *multivariate resultant.* The three most commonly used multivariate resultant matrix formulations are those named after *Sylvester or Macaulay* [37, 39, 135], those named after *Bézout or Dixon* [34, 61, 121], or the *hybrid formulation* [58, 113, 122]. Extending the Sylvester-Macaulay type, we shall emphasize also *sparse resultant* formulations [38, 98, 226]. For a unified treatment, see [81].

The theory of Gröbner bases provides powerful tools for performing compu-

tations 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 solving polynomial systems. Although introduced for studying algebraic differential equations [214], the method was converted to ordinary polynomial rings when developing an effective method for automatic theorem proving [242]. Given a polynomial system P, the characteristic set algorithm computes a new system in triangular form, such that the set of common roots of P is equivalent to the set of roots of the triangular system [120]. *Triangular systems* have k_1 polynomials in a specific variable, k_2 polynomials in this and one more variable, k_3 polynomials in these two and one more variable, and so on, for a total number of $k_1 + \cdots + k_n$ polynomials.

4.1 Resultant of Univariate Systems

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

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

have a common root leads to a condition that has to be satisfied by the coefficients of f, g. Using a derivation of this condition due to Euler, the *Sylvester matrix* of f, g (which is of dimension $m+n$) 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, g to have common roots in the algebraic closure of the coefficient ring.

As a running example let us consider the following system in two variables provided in [131]:

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

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, g. More specifically, the Sylvester resultant with respect to y is given by the following determinant:

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

An alternative matrix of order $\max\{m, n\}$, named after Bézout, yields the same determinant.

The roots of the Sylvester determinant are $\{-3, 0, 1\}$. For each x value, one can substitute the x value back into the original polynomials yielding the solutions $(-3, 1), (0, 1), (1, -1)$. More practically, one can use the Sylvester matrix to reduce system solving to the computation of eigenvalues and eigenvectors as explained in "Polynomial System Solving by Using Resultants".

The Sylvester formulations has led to a *subresultant theory*, which produced an efficient algorithm for computing the GCD of univariate polynomials and their resultant, while controlling intermediate expression swell [213, 133]; see also section "Subdivision algorithms". Subresultant theory has been generalized to several variables, e.g. [33, 54].

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 resultant of two univariate polynomials. We examine two generalizations, namely, the classical and the sparse resultants. Both of them generalize the determinant of $n+1$ *linear* polynomials in n variables.

The *classical resultant* of a system of $n+1$ polynomials with symbolic coefficients in n variables vanishes exactly when there exists a common solution in the *projective* space over the algebraic closure of the coefficient ring [51]. The *sparse (or toric) resultant* characterizes solvability of the same overconstrained system over a smaller space, which coincides with affine space under certain genericity conditions [52, 98, 226]. The main algorithmic question is to construct a matrix whose determinant is the resultant or a nontrivial multiple of it.

Cayley, and later Dixon, generalized Bézout's method to a set

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

of $n+1$ polynomials in n variables. 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 [34, 61, 81, 121]. A nontrivial resultant multiple, known as the *projection operator,* can be extracted via a method discussed in [42, thm. 3.3.4]. This article, along with [74], explain the correlation between residue theory and the Bézout–Dixon matrix; the former leads to an alternative approach for studying and approximating all common solutions.

Macaulay [135] constructed a matrix whose determinant is a multiple of the classical resultant; he stated his approach for a well-constrained system of n homogeneous polynomials in n variables. The Macaulay matrix simultaneously generalizes the Sylvester matrix and the coefficient matrix of a system of linear equations. Like 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 to yield the exact resultant in the case of generic coefficients. To address arbitrary coefficients, Canny [37] proposed a general method that perturbs any polynomial system and extracts a nontrivial projection operator from Macaulay's construction.

By exploiting the structure of polynomial systems by means of sparse elimination theory, a matrix formula for computing the sparse resultant of $n+1$ polynomials in n variables was given in [38] and consequently improved in [41, 77]. Like the Macaulay and Dixon matrices, the determinant of the sparse resultant matrix, also known as Newton matrix, only yields a projection operation. However, in certain cases of bivariate and multihomogeneous systems, determinantal formulae for the sparse resultant have been derived [58, 80, 122]. To address degeneracy issues, Canny's perturbation has been extended in the sparse context [55]. D'Andrea [53] extended Macaulay's rational formula for the resultant to the sparse setting, thus defining the sparse resultant as the quotient of two determinants; see [79] for a simplified algorithm in certain cases.

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 an integer-valued function that bounds the number of affine common roots of these polynomials [14]. This remarkable bound is the cornerstone of sparse elimination theory. The mixed volume bound is significantly smaller than the classical Bézout bound for polynomials with small Newton polytopes but they coincide for polynomials whose Newton polytope is the unit simplex multiplied by the polynomial's total degree. 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 can still yield the desired information about their common roots [52].

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 $\{p_1(x_1, \ldots, x_n), \ldots, p_n(x_1, \ldots, x_n)\}$. By augmenting the polynomial set by a generic linear polynomial [37, 52], 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 method relies on the properties of the multivariate resultant, and hence, can be constructed using either Macaulay's, Dixon's, or sparse formulations. An alternative approach is to *hide* a variable in the coefficient field [75, 81, 138].

Consider the previous example augmented by a generic linear form:

$$\begin{aligned} p_1 &= x^2 + xy + 2x \quad + y - 1 = 0 \ , \\ p_2 &= x^2 \quad + 3x - y^2 + 2y - 1 = 0 \ , \\ p_l &= \qquad ux \qquad + vy + w = 0 \ . \end{aligned}$$

As described in [39], the following (transposed) Macaulay matrix M corresponds to the u-resultant of the above system of polynomials:

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

It should be noted that

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

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

Resultant matrices can also 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 numerically approximate all solutions [4, 40, 42, 75, 81]. 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 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:

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

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 division [75]. As for the eigenvalues, they correspond to the values of w at the solutions; see [76] on numerical issues, and an implementation.

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 [36, 215]. This allows us to express each root coordinate

as the value of a univariate polynomial, evaluated over an algebraic number, which is specified as a solution of a single polynomial equation. All 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 [25, 85, 86, 150, 151, 152].

A comparison of the resultant formulations can be found, e.g., in [81, 120, 138]. The multivariate resultant formulations have been used for diverse applications such as *algebraic and geometric reasoning* [42, 60, 138], including separation bounds for the isolated roots of arbitrary polynomial systems [84], *robot kinematics* [56, 212, 138], and *nonlinear computational geometry, computer-aided geometric design and, in particular, implicitization* [33, 43, 87, 111].

4.4 Gröbner Bases

Solving systems of nonlinear equations can be formulated in terms of polynomial ideals [51, 105, 127]. The *ideal* generated by a system of polynomials $p_1, \ldots, p_r$ over $\mathbf{Q}[x_1, \ldots, x_n]$ is the set of all linear combinations

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

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

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

A version of the *Hilbert Nullstellensatz* states that

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

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

A term $t = x_1^{e_1} x_2^{e_2} \ldots x_n^{e_n}$ of a polynomial is a product of powers with $\deg(t) = e_1 + \cdots + e_n$. 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 [51, 97]. 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.

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 [51, 105, 127]. 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. Buchberger [28, 29, 30] 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 [139]. In the case of zero-dimensional ideals, those whose varieties have only isolated points, a change of basis algorithm was outlined in [90], which can be utilized for solving: 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. Significant progress has been achieved in the algorithmic realm by Faugère [88, 89].

Another way to finding all common real roots is by means of RUR; see the previous section. 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 be numerically more stable [127, 154, 222, 155].

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

$$\begin{array}{lcccccccc} p_1 & = & x^2 & +xy & +2x & & +y & -1 \ , \\ p_2 & = & x^2 & & +3x & -y^2 & +2y & -1 \ , \end{array}$$

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

$$\begin{array}{lcccccccc} {p_1}^* & = & x^2 & & +3x & & +2y & -2 \ , \\ {p_2}^* & = & & xy & -x & & -y & +1 \ , \\ {p_3}^* & = & & & & y^2 & & -1 \ . \end{array}$$

The final univariate polynomial has minimal degree, whereas the polynomials used in the back substitution have total degree no larger than the number of roots. As an example of polynomial reduction the polynomial x^2y^2 is reduced with respect to the previously computed Gröbner basis $\{p_1^*, p_2^*, p_3^*\} = GB(p_1, p_2)$ along two distinct reduction paths, both yielding $-3x - 2y + 2$ as the normal form.

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.

Gröbner bases can be used for many polynomial ideal theoretic operations [30, 50]. Other applications include computer-aided geometric design [111], polynomial interpolation [129], coding and cryptography [92], and robotics [91].

5 Research Issues and Summary

Algebraic algorithms deal with numbers, vectors, matrices, polynomials, formal power series, exponential and differential polynomials, rational functions, algebraic sets, curves and surfaces. In this vast area, manipulations with matrices and polynomials, in particular the solution of a polynomial equation and linear and polynomial systems of equations, are most fundamental in modern computations in Sciences, Engineering, and Signal and Image Processing. We reviewed the state of the art for the solution of these three tasks 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 their applications to polynomial root-finding, currently of growing interest, and new techniques for randomized preprocessing for matrix computations, evaluation of resultants 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. This includes practical applications to small and medium-size systems. Efficient implementations that handle the nongeneric cases, including multiple roots and nonisolated solutions, is probably the most crucial issue today in relation to resultants. The latter are also studied in relation to a more general object, namely the discriminant of a well-constrained system, which characterizes the existence of multiple roots. Another interesting current direction is algorithmic improvement by exploiting the structure of the polynomial systems, including sparsity, or the structure of the encountered matrices, for both resultants and Gröbner bases.

6 Defining Terms

Characteristic polynomial: Shift an input matrix A by subtracting the identity matrix xI scaled by variable x. The determinant of the resulting matrix is the characteristic polynomial of the matrix A. Its roots coincide with the eigenvalues of the shifted matrix $A - xI$.

Condition number of a matrix is a scalar κ which grows large as the matrix approaches a singular matrix; then numeric inversion becomes an ill-conditioned problem. κ `OUTPUT ERROR NORM` $\approx$ `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 whose value is invariant in adding to a row (resp. column) any linear combination of other rows (resp. columns). $\det(AB) = \det A \cdot \det B$ for a pair of square matrices A and B, $\det B = -\det A$ if the matrix B is obtained by interchanging a pair of adjacent rows or columns of a matrix A, $\det A \neq 0$ if and only if a matrix A is invertible. Determinant of a block diagonal or block triangular matrix is the product of the diagonal blocks, and so $\det A = (\det A_{0.0}) \det S$ under (1). One can compute a determinant by using these properties and matrix factorizations, e.g., recursive factorization (1).

Gröbner basis: Given a term ordering, the Gröbner basis of a polynomial ideal is a generating set of this ideal, such that the (multivariate) division of any polynomial by the basis 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 $\mathbf{v}$ such that $A\mathbf{v} = \lambda\mathbf{v}$, for a square matrix A and the associated eigenvalue λ. A generalized eigenvector $\mathbf{v}$ satisfies the equation $A\mathbf{v} = \lambda B\mathbf{v}$ for two square matrices A and B and the associated eigenvalue λ. Both definitions extend to row vectors that premultiply the associated matrices.

Mixed volume: An integer-valued function of n convex polytopes in n-dimensional Euclidean space. Under proper scaling, this function bounds the number of toric complex roots of a well-constrained polynomial system, where the convex polytopes are defined to be the Newton polytopes of the given polynomials.

Newton polytope: The convex hull of the exponent vectors corresponding to terms with nonzero coefficients in a given multivariate polynomial.

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 $n+1$ variables, whose vanishing is the minimal 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 whose zero entries are much more numerous than its nonzero entries.

Structured matrix: A matrix whose every entry can be derived by a formula depending on a smaller number of parameters, typically on $O(m+n)$ 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 and is defined by $m+n$ parameters s_i and t_j, $i = 1, \ldots, m$; $j = 1, \ldots, n$. Typically a structured matrix can be multiplied by a vector in nearly linear arithmetic time.

References

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

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

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

[4] Auzinger, W., 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.

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

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

[7] Ballard, G., Demmel, J., Holtz, O., Schwartz, O., Minimizing Communication in Numerical Linear Algebra. Tech Report *UCB/EECS-2011-15*, February 2011.

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

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

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

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

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

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

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

[15] Bini, D.A., Boito, P., A fast algorithm for approximate polynomial GCD based on structured matrix computations. *Operator Theory: Advances and Applications*, 199, 155–173, Birkhäuser, 2010.

[16] Bini, D.A., Boito, P., Eidelman, Y., Gemignani, L., Gohberg, I., A fast implicit QR algorithm for companion matrices. *Linear Algebra and Applications*, 432, 2006–2031, 2010.

[17] Bini, D.A., Fiorentino, G., Design, analysis, and implementation of a Mmultiprecision polynomial rootfinder. *Numer. Algs.*, 23, 127–173, 2000.

[18] Bini, D.A., Gemignani, L., Pan, V.Y., Inverse power and Durand/Kerner iteration for univariate polynomial root-finding. *Computers and Mathematics (with Applications)*, 47 (2/3), 447–459, 2004.

[19] Bini, D.A., Gemignani, L., Pan, V.Y., Fast and stable QR eigenvalue algorithms for generalized companion matrices and secular equation. *Numer. Math.*, 3, 373–408, 2005.

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

[21] Bini, D., Pan, V.Y., Polynomial division and its computational complexity, *J. Complexity*, 2, 179–203, 1986.

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

[23] Bini, D.A., Pan, V.Y., Verschelde, J., eds., Special Issue on Symbolic–Numerical Algorithms. *Theoretical Comp. Sci.*, 409, 2, 255–268, 2008.

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

[25] Bondyfalat, D., Mourrain, B., Pan, V.Y., Computation of a specified root of a polynomial system of equations using eigenvectors. *Linear Alg. Appls.*, 319, 193–209, 2000. Also *Proc. ISSAC*, 252–259, ACM Press, 1998.

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

[27] Brönnimann, H., Emiris, I.Z., Pan, V.Y., Pion, S., Sign determination in residue number systems. *Theor. Comp, Science*, 210 (1), 173–197, 1999.

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

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

[30] 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.

[31] Buchberger, B., Collins, G.E., Loos, R., Albrecht, R., eds. *Computer Algebra: Symbolic Algebraic Computation.* Springer, 2nd edition, 1983.

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

[33] Busé, L., D'Andrea, C. Inversion of parameterized hypersurfaces by means of subresultants. In *Proc. ISSAC,* 65–71, ACM Press, 2004.

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

[35] Busé, L., Elkadi, M., Mourrain, B., eds. Special Issue on Algebraic–Geometric Computations, *Theor. Comp. Science*, 392 (1-3), 1-178, 2008.

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

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

[38] Canny, J., Emiris, I.Z., A Subdivision-Based Algorithm for the Sparse Resultant, *J. ACM*, 47(3), 417–451, 2000.

[39] Canny, J., Kaltofen, E., Lakshman, Y., Solving systems of non-linear polynomial equations faster. In *Proc. ISSAC,* 121–128, ACM, 1989.

[40] Canny, J., Manocha, D., Efficient techniques for multipolynomial resultant algorithms. In *Proc. ISSAC,* 85–95, ACM Press, 1991.

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

[42] Cardinal, J.-P., Mourrain, B., Algebraic approach of residues and applications. In *The Math. of Numerical Analysis, Lects. in Applied Math.,* 32, 189–210, AMS, Providence, RI, 1996.

[43] Chen, F., Cox, D.A., Liu, Y., The mu-basis and implicitization of a rational parametric surface. *J. Symbolic Computation*, 39(6), 689–706, 2005.

[44] Chen, Z., Storjohann, A., A BLAS based C library for exact linear algebra on integer matrices. In *Proc. ISSAC,* 92–99, ACM Press, 2005.

[45] Cohn, H., Kleinberg, R., Szegedy, B., Umans, C., Group-theoretic algorithms for matrix multiplication. *Proc. IEEE FOCS*, 379–388, 2005.

[46] Clarkson, K.L., Safe and effective determinant evaluation. *Proc. IEEE FOCS*, 387–395, IEEE Computer Society Press, 1992.

[47] Collins, G.E., Akritas, A., Polynomial real root isolation using Descartes' rule of signs. In *SYMSAC '76*, 272–275, ACM Press, NY, 1976.

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

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

[50] Cox, D.A., Gröbner bases: a sampler of recent developments. In *Proc. ISSAC*, 387–388, ACM, 2007.

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

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

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

[54] D'Andrea, C., Krick, T., Szanto, A., Multivariate subresultants in roots. *J. Algebra*, 302(1), 16–36, 2006.

[55] D'Andrea, C., Emiris, I.Z., Computing sparse projection operators. In *Symbolic Computation: Solving Equations in Algebra, Geometry, and Engineering*, 121–139, AMS, Providence, RI, 2001.

[56] Daney, D., Emiris, I.Z., Robust parallel robot calibration with partial information. In *Proc. IEEE Intern. Conf. Robotics Automation*, Seoul, pp. 3262–3267, 2001.

[57] Demmel, J.J.W., *Applied Numerical Linear Algebra.* SIAM, 1997.

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

[59] Dickenstein, A., Emiris, I.Z., editors, Solving polynomial equations: foundations, algorithms and applications. In *Algorithms and Computation in Mathematics*, 14, Springer, Berlin, 2005.

[60] Dickenstein, A., Sturmfels, B., Elimination theory in codimension 2. *J. Symb. Comp.*, 34(2):119–135, 2002.

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

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

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

[64] Du, Z., Sharma, V., Yap, C.K., Amortized bound for root isolation via Sturm sequences. [236], pp. 113–129.

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

[66] 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., Takayama, N., eds., *Proc. ICMS 200*, 40-50, Beijing, China, 2002.

[67] Dumas, J-G., Gautier, T., Pernet, C., Finite field linear algebra subroutines. In *Proc. ISSAC,* 63–74, ACM Press, 2002.

[68] Dumas, J-G., Giorgi, P., Pernet, C., Finite field linear algebra package. In *Proc. ISSAC,* 118–126, ACM Press, 2004.

[69] Eidelman, Y., Gohberg, I., On a New Class of Structured Matrices, *Integral Equations and Operator Theory*, **34**, 293–324, Birkhäuser, Basel, 1999.

[70] Eberly, W., Giesbrecht, M., Giorgi, P., Storjohann, A., Villard, G., Faster inversion and other black box matrix computations using efficient block projections. In *Proc. ISSAC*, 143–150, ACM Press, 2007.

[71] Eberly, W., Giesbrecht, M., Villard, G., On computing the determinant and Smith form of an integer matrix. *Proc. IEEE FOCS*, 675–685, 2000.

[72] Eigenwillig, A., Kettner, L., Krandick, W., Mehlhorn, K., Schmitt, S., Wolpert, N., A Descartes Algorithm for Polynomials with Bit-Stream Coefficients. In *CASC'2005*, *LNCS*, 3718, 38–149. Springer, 2005.

[73] Eigenwillig, A., Sharma, V., Yap, C.K., Almost tight recursion tree bounds for the Descartes method. *Proc. ISSAC*, 71–78, ACM, 2006.

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

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

[76] Emiris, I.Z., Matrix Methods for Solving Algebraic Systems, In *Symbolic Algebraic Methods and Verification Methods*, Springer, Wien, pp. 69–78, 2001. Also `arxiv.org/abs/1201.5810`, 2011.

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

[78] Emiris, I.Z., Galligo, A., Tsigaridas, E.P., Random polynomials and expected complexity of bisection methods for real solving. *Proc. ISSAC*, 235–242, ACM Prss, 2010.

[79] Emiris, I.Z., Konaxis, C., Single-lifting Macaulay-type formulae of generalized unmixed sparse resultants, *J. Symb. Comp.*, 46(8), 919–942, 2011.

[80] Emiris, I.Z., Mantzaflaris, A., Multihomogeneous resultant matrices for systems with scaled support. In *Proc. ISSAC*, 143–150, ACM, 2009.

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

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

[83] Emiris, I.Z., Mourrain, B., Tsigaridas, E.P., Real Algebraic Numbers: Complexity Analysis and Experimentation. In *Reliable Implementations of Real Number Algorithms: Theory and Practice*, LNCS, Springer, 2007.

[84] Emiris, I.Z., Mourrain, B., Tsigaridas, E.P., The DMM Bound: Multivariate (Aggregate) Separation Bound. In *Proc. ISSAC*, 242–250, ACM Press, 2010.

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

[86] Emiris I.Z., Pan, V.Y., Improved algorithms for computing determinants and resultants. *J. Complexity*, 21 (1), 43–71, 2005. Also *Proc. CASC'03*, (E. W. Mayr, V. G. Ganzha, E. V. Vorozhtzov, eds.) 81–94, 2003.

[87] Emiris, I.Z., Tzoumas, G.M., Exact and efficient evaluation of the InCircle predicate for parametric ellipses and smooth convex objects, *Computer-Aided Design*, 40(6), 691–700, 2008.

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

[89] Faugère, J.-C., A new efficient algorithm for computing Gröbner bases w/o Reduction to Zero (F5). In *Proc. ISSAC,* 75–83, ACM, 2002.

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

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

[92] Faugère, J.-C., Levy-dit-Vehel, F., Perret, L., Cryptanalysis of MinRank. In *Proc. CRYPTO*, pp. 280–296, 2008.

[93] Fortune, S., An Iterated Eigenvalue Algorithm for Approximating Roots of Univariate Polynomials. *J. Symbolic Comp.*, 33 (5), 627–646, 2002.

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

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

[96] von zur Gathen, J., Gerhard, J., *Modern Computer Algebra.* Cambridge U. Press, Cambridge, 2003 (2nd edition).

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

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

[99] Gilbert, A., Indyk, P., Sparse Recovery Using Sparse Matrices. *Proceedings of IEEE*, 2010.

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

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

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

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

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

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

[106] Halko, N., Martinsson, P.G., Tropp, J.A., Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions, *SIAM Review*, 53, 2, 217–288, 2011.

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

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

[109] Hemmer, M., Tsigaridas, E.P., Zafeirakopoulos, Z., Emiris, I.Z., Karavelas, M., Mourrain, B., Experimental evaluation and cross-benchmarking of univariate real solvers. In Proc. *SNC'09*, Kyoto, Japan, 2009.

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

[111] Hoffmann, C.M., Sendra, J.R., Winkler, F. Special Issue on Parametric Algebraic Curves and Applications, *J. Symbolic Comp.*, 23, 1997.

[112] Jenkins, M.A., 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.

[113] Jouanolou, J.-P., Formes d'Inertie et Résultant : Un Formulaire. *Adv. in Math.*, 126:119–250, 1997. Also TR 499/P-288, IRMA, Strasbourg, 1992.

[114] Kailath, T., Sayed, A., eds. *SIAM volume on Fast Reliable Algorithms for Matrices with Structure*, SIAM Publications, Philadelphia, 1999.

[115] Kaltofen, E., Pan, V.Y., Processor efficient parallel solution of linear systems over an abstract field. In *Proc. SPAA'91,* 180–191, ACM, NY, 1991.

[116] Kaltofen, E., Pan, V.Y., Processor-efficient parallel solution of linear systems II: the positive characteristic and singular cases. In *Proc. FOCS'92,* 714–723, IEEE Computer Society, Los Alamitos, CA, 1992.

[117] Kaltofen, E., 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.

[118] Kaporin, I., The aggregation and cancellation techniques as a practical tool for faster matrix multiplication. [82], pp. 469–510.

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

[120] Kapur, D., 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.

[121] Kapur, D., Saxena, T., Comparison of various multivariate resultant formulations. In in *Proc. ISSAC*, 187–195, ACM, 1995.

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

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

[124] Kolda, T. G., and Bader, B. W., Tensor Decompositions and Applications. *SIAM Review*, 51(3), 455–500, 2009.

[125] Kotsireas, I., Mourrain, B., Pan, V. Y., eds., Special Issue on Algebraic and Numerical Algorithms. *Theor. Comp. Sci.*, 412, 16, 1443–1543, 2011.

[126] Kotsireas, I., Mourrain, B., Pan, V. Y., Zhi, L., eds., Special Issue on Symbolic-Numerical Algorithms. *Theor. Comp. Sci.*, in print.

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

[128] Laderman, J., Pan, V.Y., Sha, H.X., On practical algorithms for accelerated matrix multiplication. *Lin. Alg. Appls.*, 162–164, 557–588, 1992.

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

[130] Lawson, C.L., Hanson, R.J., *Solving Least Squares Problems.* Prentice-Hall, NJ, 1974, and (with a survey of recent developments) SIAM, 1995.

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

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

[133] Lickteig, T., Roy, M.-F., Sylvester–Habicht sequences and fast Cauchy index computation. *J. Symbolic Comp.*, 31(3), 315–341, 2001.

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

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

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

[137] Mahoney, M. W., Maggioni, M., Drineas, P., Tensor-CUR Decompositions for Tensor-based Data. *SIAM J. Matrix Anal. Appls.*, 30, 2, 957–987, 2008.

[138] Manocha, D., *Algebraic and Numeric Techniques for Modeling and Robotics*. Ph.D. Thesis, CSD, DEECS, UC, Berkeley, CA, 1992.

[139] Mayr, E.W., Meyer, A.R., The Complexity of the Finite Containment Problem for Petri Nets. *J. ACM*, 28(3), 561–576, 1981.

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

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

[142] McNamee, J.M., *Numerical Methods for Roots of Polynomials (Part 1)*, Elsevier, Amsterdam, 2007.

[143] McNamee, J.M., Pan, V.Y., Efficient polynomial root-refiners: a survey and new record estimates. *Computers Math. (Appls.)*, 63, 239–254, 2012.

[144] McNamee, J.M., Pan, V.Y., *Numerical Methods for Roots of Polynomials, Part 2*, 780+XIX pages, submitted to Elsevier publishers.

[145] Mehlhorn, K., Ray, S., Faster algorithms for computing Hong's bound on absolute positiveness. *J. Symbolic Computation*, 45(6), 677 – 683, 2010.

[146] Mehlhorn, K., Sagraloff, M., A deterministic algorithm for isolating real roots of a real polynomial. *J. Symb. Computation*, 46(1), 70–90, 2011.

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

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

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

[150] Mourrain, B., Pan, V.Y., Asymptotic acceleration of solving polynomial systems. *Proc. STOC'98*, 488–496, ACM Press, New York, 1998.

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

[152] Mourrain, B., Pan, V.Y., Ruatta, O., Accelerated solution of multivariate polynomial systems of equations. *SIAM J. Comp.*, 32, 2, 435–454, 2003.

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

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

[155] Mourrain, B., Trébuchet, P., Stable normal forms for polynomial system solving *Theor. Comp. Science,* 409(2), 229–240, 2008.

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

[157] Neff, C.A., Reif, J.H., An $O(n^{l+\epsilon})$ algorithm for the complex root problem. Proc. *IEEE FOCS*, 540–547, 1994.

[158] Nguyen, P.Q., Valle, B. (Eds.) *The LLL Algorithm, Survey and Applications. Series: Information Security and Cryptography*, XIV, 496 pp., Springer, 2010, ISBN 978-3-642-02294-4.

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

[160] Oseledets, I. V., Tyrtyshnikov, E. E., Breaking the curse of dimensionality, or how to use SVD in many dimensions. *SISC*, 31, 5, 3744–3759, 2009.

[161] Pan, V. Y., On schemes for the evaluation of products and inverses of matrices (in Russian), *Uspekhi Mat. Nauk*, 27, 5 (167), 249–250, 1972.

[162] Pan, V. Y., Strassen's algorithm is not optimal. Trilinear technique of aggregating. *Proc. IEEE FOCS*, 166–176, 1978.

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

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

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

[166] 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.

[167] Pan, V. Y., On computations with dense structured matrices. *Math. of Comp.*, 55, 191, 179–190, 1990. Also *Proc. ISSAC*, 34–42, ACM, 1989.

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

[169] 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.

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

[171] Pan, V.Y., Optimal and nearly optimal algorithms for approximating polynomial zeros. *Computers in Mathematics (with Applications),* 31(12), 97–138, 1996. Also STOC'95, 741–750, ACM, Press, New York, 1995.

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

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

[174] Pan, V.Y., Solving polynomials with computers. *American Scientist*, 86, 62–69, January–February 1998.

[175] Pan, V.Y., Some recent algebraic/numerical algorithms. *Electronic Proc. IMACS/ACA'98*, 1998.
www-troja.fjfi.cvut.cz/aca98/sessions/approximate.

[176] Pan, V.Y., Numerical computation of a polynomial GCD and extensions. *Information and Computation*, 167(2), 71–85, 2001. Also *Proc. SODA'98*, 68–77, ACM Press, and SIAM Publications, 1998.

[177] 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.

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

[179] Pan, V.Y., Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. *J. Symbolic Computations*, 33 (5), 701–733, 2002.

[180] 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/

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

[182] Pan, V.Y., Newton's iteration for matrix inversion, advances and extensions. *Matrix Methods: Theory, Algorithms and Applications* (eds. V. Olshevsky, E. Tyrtyshnikov), pp. 364–381, World Scientific, 2010.

[183] Pan, V.Y., Nearly optimal solution of rational linear systems of equations with symbolic lifting and numerical initialization. *Computers & Math. with Applications*, 62, 1685–1706, 2011.

[184] Pan, V.Y., Root-refining for a Polynomial Equation. *Proceedings of CASC 2012, Lecture Notes in Computer Science*, Springer, 2012.

[185] Pan, V.Y., Branham, S., Rosholt, R., Zheng, A., Newton's iteration for structured matrices and linear systems of equations. [114], Ch. 7, 189–210.

[186] Pan, V.Y., Grady, D., Murphy, B., Qian, G., Rosholt, R.E., Schur aggregation for linear systems and determinants. Pp. 255–268 in [23].

[187] Pan, V.Y., Ivolgin, D., Murphy, B., Rosholt, R.E., Tang, Y., Wang, X., Root-finding with eigen-solving. [236], pp. 185–210.

[188] Pan, V.Y., Ivolgin, D., Murphy, B., Rosholt, R.E., Tang, Y., Yan, X., Additive preconditioning for matrix computations. *Linear Algebra and Applications*, 432, 1070–1089, 2010.

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

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

[191] Pan, V.Y., Murphy, B., Rosholt, R.E., Unified nearly optimal algorithms for structured integer matrices. *Operator Theory: Advances and Applications*, 199, 359–375, Birkhaüser, Basel, 2010.

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

[193] Pan, V.Y., Qian, G., Randomized preprocessing of homogeneous linear systems of equations. *Linear Algebra and Applications*, 432, 3272–3318, 2010.

[194] Pan, V.Y., Qian, G., Randomization, augmentation and aggregation in matrix computations. *Linear Algebra and Its Applications*, in press.

[195] Pan, V.Y., Qian, G., Zheng, A., Randomized preconditioning of the MBA algorithm. *Proc. ISSAC*, 281–288, ACM, 2011.

[196] Pan, V.Y., Qian, G., Zheng, A., Randomized preprocessing versus pivoting. *Linear Algebra and Its Applications*, in print.

[197] Pan, V.Y., Qian, G., Zheng, A., Real and complex polynomial root-finding via eigen-solving and randomization. *Proc. CASC 2012, LNCS*, 2012.

[198] Pan, V.Y., Qian, G., Zheng, A., Randomized matrix computations, Tech. Report TR 2012005, *Ph.D. Program in Computer Science*, *Graduate Center, the City University of New York*, 2012.

Available at http://www.cs.gc.cuny.edu/tr/techreport.php?id=432 .

[199] Pan, V. Y., Qian, G., Zheng, A., Chen, Z., Matrix computations and polynomial root-finding with preprocessing. *Linear Algebra and Its Applications*, 434, 854–879, 2011.

[200] Pan, V.Y., Rami, Y., Wang, X., Structured matrices and Newton's iteration: unified approach. *Linear Algebra Applics.*, 343/344, 233–265, 2002.

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

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

[203] Pan, V.Y., 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.

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

[205] Pan, V.Y., Wang, X., Degeneration of integer matrices modulo an integer. *Linear Algebra and Its Applications*, 429, 2113–2130, 2008.

[206] Pan, V.Y., Yu, Y., Certification of numerical computation of the sign of the determinant of a matrix. *Algorithmica*, 30, 708–724, 2001. Also *Proc. SODA '99*, 715–724, ACM/SIAM, 1999.

[207] Pan, V.Y., Yan, X., Additive preconditioning, eigenspaces, and the inverse iteration. *Linear Algebra & Its Applications*, 430, 186–203, 2009.

[208] Pan, V. Y., Zheng, A., New progress in real and complex polynomial root-finding. *Computers and Math. (with Applications)* **61**, 1305–1334. Also *Proc. ISSAC*, 219–226, ACM Press, 2010.

[209] Pan, V. Y., Zheng, A., Root-finding by expansion with independent constraints. *Computers and Math. (with Applications)*, 62, 3164-3182, 2011.

[210] Parlett, B., *Symmetric Eigenvalue Problem.* Prentice Hall, 1980.

[211] Quinn, M.J., *Parallel Computing: Theory and Practice.* McGraw-Hill, NY, 1994.

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

[213] Reischert, D., Asymptotically fast computation of subresultants. In *Proc. ISSAC*, 233–240, ACM, 1997.

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

[215] Rouillier, F., Solving zero-dimensional systems through the rational univariate representation. *AAECC Journal*, 9, 433–461, 1999.

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

[217] Sagraloff, M., When Newton meets Descartes: A simple and fast algorithm to isolate the real roots of a polynomial. *CoRR*, abs/1109.6279, 2011.

[218] Sankar, A., Spielman, D., Teng, S.-H., Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM J. Matrix Analysis*, 28(2), 446–476, 2006.

[219] Schönhage, A., The fundamental theorem of algebra in terms of computational complexity. *Math. Dept., Univ. Tübingen*, Germany, 1982.

[220] Sharma, V., Complexity of real root isolation using continued fractions. *Theor. Comput. Sci.*, 409(2):292–310, 2008.

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

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

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

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

[225] Stothers, A. J., On the Complexity of Matrix Multiplication. Ph.D. Thesis, *University of Edinburgh*, 2010.

[226] Sturmfels, B., Sparse elimination theory. *Proc. Comp.. Alg. Geom. Commut. Algebra,* Eisenbud, D., Robbiano, L., Eds., Cortona, Italy, 1991.

[227] Tarjan, R.E., A unified approach to path problems. *J. of ACM*, 28(3), 577–593 and 594–614, 1981.

[228] Trefethen, L.N., Bau III, D., *Numerical Linear Algebra.* SIAM, 1997.

[229] Tsigaridas, E.P., Improved complexity bounds for real root isolation using Continued Fractions. In S. Ratschan, ed., *Proc. 4th Int'l Conf. on Math. Aspects Comp. Inf. Sci. (MACIS)*, 226–237, Beijing, China, 2011.

[230] Tsigaridas, E.P., Emiris, I.Z., On the complexity of real root isolation using Continued Fractions. *Theor. Computer Sci.*, 392, 158–173, 2008.

[231] Van Barel, M., Vandebril, R., Van Dooren, P., Frederix, K., Implicit double shift QR-algorithm for companion matrices. *Numer. Math.* **116**, **2**, 177–212, 2010.

[232] Vandebril, R., Van Barel, M., Mastronardi, N., *Matrix Computations and Semiseparable Matrices: Linear Systems* (Volume 1). The Johns Hopkins University Press, Baltimore, Maryland, 2007.

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

[234] Vassilevska Williams, V., Multiplying matrices faster than Coppersmith–Winograd. *STOC 2012*, 887–898 and Breaking the Coppersmith–Winograd barrier. *UC Berkeley and Stanford University*, preprint, November 2011.

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

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

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

[238] Watkins, D.S., *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods.* SIAM, Philadelphia, PA, 2007.

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

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

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

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

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

[244] Zhlobich, P., Differential qd Algorithm with Shifts for Rank-Structured Matrices. *SIAM J. on Matrix Analysis and Applications*, in press.

[245] Zippel, R., *Effective Polynomial Computations.* Kluwer Academic, 1993.

Further Information

The books and special issues of journals [1, 5, 22, 26, 32, 59, 82, 97, 178, 222, 245] provide a 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, [62] for solving linear systems of equations, [95], [221], ARPACK, and PARPACK for approximating matrix eigenvalues, and [3] for both of the two latter computational problems. Comprehensive treatment of numerical matrix computations and extensive bibliography can be found in [103, 223], and there are many more specialized books on them [6, 9, 63, 100, 104, 110, 210, 228, 240] as well as many survey articles [108, 159, 168] 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 [169].

On parallel matrix computations see [101, 103, 115, 116, 192, 7] assuming general input matrices, [101, 108, 169, 202] assuming sparse inputs, [63] assuming banded inputs, and [22, 172, 178] assuming dense structured inputs. On Symbolic-Numeric algorithms, see the books [22, 178, 236], surveys [168, 173, 175], special issues [82, 23, 125, 126], and the bibliography therein. For the general area of exact computation and the theory behind algebraic algorithms and computer algebra, see [10, 31, 51, 52, 59, 96, 97, 148, 241, 243, 147, 245].

There is a lot of generic software packages for exact computation, SYNAPS (www-sop.inria.fr/galaad/software/synaps/, [153]), a `C++` open source library devoted to symbolic and numeric computations with polynomials, algebraic numbers and polynomial systems, which has been evolving into the REALROOT package of the open source computer algebra system MATHEMAGIX (www.mathemagix.org); NTL (www.shoup.net/ntl/) a high-performance `C++` library providing data structures and algorithms for vectors, matrices, and polynomials over the integers and finite fields; and EXACUS (www.mpi-inf.mpg.de/projects/Exacus, [12]), a `C++` library for curves and surfaces that provides exact methods for solving polynomial equations. A highly efficient tool is FGB(http://fgbrs.lip6.fr/salsa/Software) for Gröbner basis, and RS for the rational univariate representation, and real solutions of systems of polynomial equations and inequalities. Finaly, LINBOX (www.linalg.org and [66]) is a `C++` library that provides exact high-performance implementations of linear algebra algorithms.

This chapter does not cover the area of polynomial factorization. We refer the interested reader to [96, 132, 158], and the bibliography therein.

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

The *Journal of Symbolic Computation* and the *Foundations of Computational Mathematics* specialize on topics in Computer Algebra, which are also covered in the *Journal of Computational Complexity*, the *Journal of Pure and Applied Algebra* and, less regularly, in the *Journal of Complexity. Mathematics for Computer Science* and *Applicable Algebra in Engineering, Communication*

and Computing are currently dedicated to the subject of the chapter as well. *Theoretical Computer Science* has become more open to algebraic–numerical and algebraic–geometric subjects [23, 35, 82, 125].

The annual *International Symposium on Symbolic and Algebraic Computation (ISSAC)* is the main conference in computer algebra; these topics are also presented at the bi-annual Conference *MEGA* and the newly founded SIAM conference on Applications of Algebraic Geometry. They also appear, in the annual *ACM Conference on Computational Geometry*, as well as at various Computer Science conferences, including SODA, FOCS, and STOC.

Among many conferences on numerical computing, most comprehensive ones are organized under the auspices of SIAM and ICIAM. The International Workshop on Symbolic-Numeric Algorithms can be traced back to 1997 (SNAP in INRIA, Sophia Antipolis) and a special session in IMACS/ACA98 Conference in Prague, Czech Republic, in 1998 [175]. It restarted in Xi'an, China, 2005; Timishiora, Romania, 2006 (supported by IEEE), and London, Ontario, Canada, 2007 (supported by ACM). The topics of Symbolic-Numerical Computation are also represented at the conferences on the *Foundations of Computational Mathematics (FoCM)* (meets every 3 years) and quite often at ISSAC.