Contents

Chapter 1
Regularization and Matrix Computation in Numerical Polynomial Algebra

Zhonggang Zeng

Abstract Numerical polynomial algebra emerges as a growing field of study in recent years with a broad spectrum of applications and many robust algorithms. Among the challenges in solving polynomial algebra problems with floating-point arithmetic, difficulties frequently arise in regularizing ill-posedness and handling large matrices. We elaborate regularization principles for reformulating the ill-posed algebraic problems, derive matrix computation arising in numerical polynomial algebra, as well as subspace strategies that substantially improve computing efficiency by reducing matrix sizes. Those strategies have been successfully applied to numerical polynomial algebra problems such as GCD, factorization, multiplicity structure and elimination.

Introduction

Accelerated by the advancement of computer algebra systems (CAS), symbolic algebraic computation has enjoyed tremendous success since the advent of the Gröbner basis and continues to be a driving force in computational commutative algebra. The abundance of algorithms along with the depth and breadth of the theories developed over the years for algebraic computation sets a solid foundation for numerical polynomial algebra, which emerges as a rapidly growing field of study in recently years. Pioneered by Li [57], Sommese [82] and others, numerical polynomial system solving based on the homotopy continuation method has set the standard in efficiency as well as robustness, and enters the stage of expansion into numerical algebraic geometry as a new field [81]. Meanwhile, numerical polynomial algebra emanating from fundamental numerical analysis and numerical linear

Zhonggang Zeng
Department of Mathematics, Northeastern Illinois University, Chicago, IL 60625, USA, Research supported in part by NSF under Grants DMS-0412003 and DMS-0715127, e-mail: zzeng@neiu.edu

algebra has also thrived in theory and problem solving, as presented in Stetter's recently published textbook [84] and evidenced by healthy development of software (cf. [86]).

In comparison to symbolic computation, numerical computation and approximate solutions offer substantial advantages in many areas and come with drawbacks in other aspects. Computing numerical solutions in many cases may be inevitable due to lack of alternatives, such as locating roots of a polynomial with a degree five or higher. Even in those cases where exact solutions are available, approximate solutions may make better sense and go deeper in revealing the physical nature. For a simple example, the exact GCD (greatest common divisor) of the polynomial pair

$$\begin{cases} f(x) &= 3x^3 - 9.2426x^2 + 13.071x - 10 \\ g(x) &= 1.7321x^4 + 2.4642x^2 - 2.4495x^3 + 1.4142x - 2 \end{cases} \tag{1.1}$$

is a trivial polynomial $u = 1$ in symbolic computation. However, its approximate GCD $\tilde{u} = 2.000006 - 1.4142x + x^2$ along with the verifiable residual provides more penetrating information: The given polynomial pair is a tiny distance 0.0000056 away from a polynomial pair having $\tilde{u}$ as its exact GCD. The difference here may mean a more desirable deblurred image restored by an approximate GCD [69] or a meaningless blank image represented by the trivial exact GCD. Furthermore, numerical computation tends to be more efficient in both storage and computing time, and more suitable for large engineering problems, as evidenced by the homotopy continuation method for solving polynomial systems applied to kinematics (cf. [82]).

Numerical polynomial algebra differs from symbolic computation in many aspects. The problem data are expected to be inexact and the computation is carried out with floating-point arithmetic. Accuracy and stability, which may not be a concern in symbolic computation, become of paramount importance in developing numerical algorithms for solving polynomial problems. A classical algorithm that is flawless in exact arithmetic, such as the Euclidean algorithm for computing polynomial GCD, may be prohibitively impractical for numerical computation and *vice versa*. As a result, it is often necessary to develop numerical algorithms from scratch and to employ totally different strategies from their symbolic cousins.

Even the meaning of "solution" may be in question and may depend on the objective in problem solving. The exact GCD $u = 1$ in (1.1) is indisputable in symbolic computation, while the meaning of the approximate GCD has evolved in several formulations over the years (see the remark in §1.2.2). The comparison between exact and approximate GCD is typical and common in algebraic problems where ill-posedness is often encountered and the solution is infinitely sensitive to data perturbations. Numerical computation which, by nature, seeks the exact solution of a nearby problem becomes unsuitable unless the problem is regularized as a well-posed one. As it turns out, the collections of ill-posed problems form pejorative manifolds of positive codimensions that entangled together with stratification structures, as elaborated in §1.2, where a manifold is embeded in the closure of manifolds of lower codimensions. Thus a tiny arbitrary perturbation pushes the problem

away from its residing manifold, losing the very structure of the solution. Dubbed as the "three-strikes" principle for formulating an approximate solution and removing ill-posedness, we summarize that the approximate solution should be the exact solution of a *nearby problem* residing in the manifold of the *maximum codimension* and having the *minimum distance* to the given problem. Approximate solutions with such a formulation become viable, attainable, and continuous with respect to data.

Numerical polynomial algebra can also be regarded as having numerical linear algebra as one of its cornerstones. Pioneered by Wilkinson [93], Golub [34], Kahan [43] and many others around the same time of Buchberger's work on Gröbner basis , numerical linear algebra flourished since 1960s with well established theories, algorithms and software libraries for numerical matrix computations. One of the early catalysts that contributed to the recent advances of numerical polynomial algebra is the introduction of the singular value decomposition to polynomial computation by Corless, Gianni, Trager and Watt [11] in 1995. As elaborated in §1.3, polynomial algebra problems in numerical computation lead to matrix computation problems such as least squares, eigenproblem and particularly, numerical rank/kernel identification. Theories, techniques and ideas accumulated in numerical linear algebra are rich resources for numerical polynomial algebra.

This paper surveys basic approaches and techniques in developing algorithms for numerical polynomial algebra, emphasizing on regularizing ill-posed algebraic problems and employing matrix computation. The approaches and techniques elaborated in this survey have been used in many algorithms with successful results and implementations. As a growing field of study, theoretical advance and algorithm development are on-going and gradually evolving. Further advancement will continue to emerge in the future.

1.1 Notation and preliminaries

1.1.1 Notation

The n dimensional complex vector space is denoted by $\mathbb{C}^n$, in which vectors are column arrays denoted by boldface lower case letters such as $\mathbf{a}$, $\mathbf{u}$, $\mathbf{v}_2$, etc, with $\mathbf{0}$ being a zero vector whose dimension can be understood from the context. Matrices are represented by upper case letters like A and J, with $\mathbb{C}^{m\times n}$ denoting the vector space consists of all $m\times n$ complex matrices. Notations $(\cdot)^\top$ and $(\cdot)^{\mathsf{H}}$ stand for the transpose and the Hermitian transpose, respectively, of the matrix or vector $(\cdot)$.

The ring of polynomials with complex coefficients in indeterminates $x_1,\ldots,x_s$ is denoted by $\mathbb{C}[x_1,\ldots,x_s]$, or $\mathbb{C}[\mathbf{x}]$ for $\mathbf{x}=(x_1,\ldots,x_s)$. A polynomial as a function is denoted by a lower case letter, say f, v, or p_1, etc. The collection of all the polynomials with a certain degree bound forms a vector space over $\mathbb{C}$. Throughout this paper, if a letter (say f) represents a polynomial, then either

$[\![f]\!]$ or the same letter in boldface (*i.e.* $\mathbf{f}$) denotes its coefficient vector, where the underlying vector space and its basis are clear from the context.

1.1.2 Numerical rank and kernel

The fundamental difference between exact and numerical computation can be demonstrated in the meaning of the matrix rank. With exact arithmetic used in symbolic computation, a matrix is non-singular if and only if its determinant is nonzero. Such a characterization of being full rank, however, is practically meaningless in numerical computation as shown in a simple example below.

Example 1.1.1. The polynomial division is equivalent to linear system solving: Finding the quotient q and the remainder r of a polynomial f divided by g satisfying $f = g \cdot q + r$ is, in fact, solving a linear system $G \begin{bmatrix} \mathbf{q} \\ \mathbf{r} \end{bmatrix} = \mathbf{f}$, where $\mathbf{f}$, $\mathbf{q}$ and $\mathbf{r}$ are vector representations of f, q and r respectively, along with a corresponding matrix G. For instance, let $g(x) = x + 10$ and use the standard monomial basis for the vector representations of $\mathbf{f}$, $\mathbf{q}$ and $\mathbf{r}$. The matrix G is

$$G = \begin{bmatrix} 1 & & & \\ 10 & 1 & & \\ & 10 & \ddots & \\ & & \ddots & 1 \\ & & & 10 \; 1 \end{bmatrix}_{(n+1)\times(n+1)} \tag{1.2}$$

where n is the degree of f. The matrix G may appear benign with a full rank and $\det(G) = 1$. However, its 2-norm distance to a singular matrix is less than 10^{-n}. Such a matrix with even a modest size, say $n = 15$, behaves the same way as a singular matrix in numerical computation. Consequently, round-off errors in the order of hardware precision during synthetic division can result in substantial errors of magnitude $O(1)$ in the coefficients of q and r (c.f. [98, §4.2.3]). The numerical rank of G should be n, not $n+1$ unless n is small. □

Remark: Example 1.1.1 indicates that the Euclidean algorithm can be highly unreliable in numerical computation of the polynomial GCD since it consists of recursive polynomial division in the form of $f = g \cdot q + r$. Interestingly, polynomial division in the form of $f = g \cdot q$, or equivalently the polynomial division in the form of $f = g \cdot q + r$ combined with an additional constraint $r = 0$, is a stable least squares problem. This can be seen from Example 1.1.1 by deleting the last column from matrix G. The resulting matrix possesses a near perfect condition number (≈ 1) with its smallest singular value larger than 9. □

The condition of a matrix in numerical computation depends on its distance to the nearest rank-deficient matrices. Likewise, the *numerical rank* (or approximate

rank) of a matrix depends on the (exact) ranks of its nearby matrices. If a matrix A can have an error of magnitude θ, then the "worst" (*i.e.* lowest rank) matrix within a distance θ dictates its numerical behavior. The numerical rank of A within θ, denoted by $rank_\theta(A)$, is thus defined as the smallest rank of all matrices within a distance θ of A:

$$rank_\theta(A) = \min_{\|B-A\|_2 \le \theta} rank(B) \tag{1.3}$$

where $rank(\cdot)$ denotes the rank of matrix $(\cdot)$ in exact sense. Naturally, the exact kernel $\mathscr{K}(B)$ of B in (1.3) is taken as the *numerical kernel* $\mathscr{K}_\theta(A)$ of A within θ:

$$\mathscr{K}_\theta(A) = \mathscr{K}(B) \quad \text{where } \|B-A\|_2 = \min_{rank(C)=rank_\theta(A)} \|C-A\|_2. \tag{1.4}$$

The numerical rank and numerical kernel of a matrix $A \in \mathbb{C}^{m\times n}$ can equivalently be defined using the singular value decomposition (SVD) [34]:

$$A = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^{\mathsf{H}} + \cdots + \sigma_n \mathbf{u}_n \mathbf{v}_n^{\mathsf{H}} = U \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ & & & \end{bmatrix} V^{\mathsf{H}} \tag{1.5}$$

where $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n \ge 0$ are the singular values of A, the matrices

$$U = [\mathbf{u}_1, \cdots, \mathbf{u}_m] \in \mathbb{C}^{m\times m} \text{ and } V = [\mathbf{v}_1, \cdots, \mathbf{v}_n] \in \mathbb{C}^{n\times n}$$

are unitary matrices whose columns are left and right singular vectors of A respectively. The numerical rank $rank_\theta(A) = k$ if and only if there are exactly k singular values of A lie above the threshold θ: $\sigma_k > \theta \ge \sigma_{k+1}$, and the numerical kernel $\mathscr{K}_\theta(A) = span\{\mathbf{v}_{k+1}, \ldots, \mathbf{v}_n\}$.

Computing the exact rank $rank(A)$ is an ill-posed problem in the sense that a tiny perturbation can alter the rank completely if A is of rank-deficient. As a result, the exact rank and kernel in general can not be computed in numerical computation since round-off errors are inevitable. By formulating numerical rank and kernel in (1.3) and (1.4) respectively, matrix rank-revealing is *regularized* as a well-posed problem and becomes suitable for numerical computation. In fact, the singular value decomposition is remarkably stable and well established in numerical linear algebra. The sensitivity of the numerical kernel can be measured by the condition number σ_1/σ_k by Wedin's Theorem [92].

On the other hand, the standard singular value decomposition can be unnecessarily costly to compute. The numerical rank/kernel computation in numerical polynomial algebra often involves matrices of low numerical nullities. For those matrices, numerical ranks and kernels can be computed efficiently using a specialized rank-

revealing method [59], which has become an indispensable component of numerical polynomial algebra algorithms that will be discussed later in this survey. The rank-revealing method assumes the input matrix R is in upper-triangular form without loss of generality. Every matrix A has a QR decomposition [34] $A = QR$ and the numerical kernel of A is identical to that of the upper-triangular matrix R. The following null vector finder is at the core of the numerical rank-revealing method in [59]:

$$\left\{\begin{array}{l} \text{set } \mathbf{z}_0 \text{ as a random vector} \\ \text{for } j = 1, 2, \cdots \text{ do} \\ \quad \left\lfloor \begin{array}{l} \text{solve } R^{\mathsf{H}}\mathbf{x} = \mathbf{z}_{j\text{-}1} \text{ with a forward substitution} \\ \text{solve } R\mathbf{y} = \mathbf{x} \text{ with a backward substitution} \\ \text{set } \mathbf{z}_j = \frac{\mathbf{y}}{\|\mathbf{y}\|_2} \text{ and } \varsigma_j = \frac{\|\mathbf{x}\|_2}{\|\mathbf{y}\|_2} \end{array} \right. \end{array}\right. \tag{1.6}$$

The iteration in (1.6) produces sequences $\{\varsigma_j\}$ and $\{\mathbf{z}_j\}$ satisfying

$$\lim_{j\to\infty} \varsigma_j = \sigma_n \quad \text{and} \quad \lim_{j\to\infty} \mathbf{z}_j = \mathbf{v}_n$$

where σ_n and $\mathbf{v}_n$ are the smallest singular value of A and the associated right singular vector respectively. Each iterative step of the algorithm (1.6) requires a forward substitution and a backward substitution on triangular matrices R^{H} and R respectively. After finding a numerical null vector $\mathbf{z}$ of matrix R within θ, insert a multiple of $\mathbf{z}$ on top of R to form

$$\hat{R} = \begin{bmatrix} \|R\|\mathbf{z}^{\mathsf{H}} \\ R \end{bmatrix}.$$

We can continue to calculate a numerical null vector $\hat{\mathbf{z}}$ of $\hat{R}$ within θ. If such a $\hat{\mathbf{z}}$ exists, it is also a numerical null vector of R orthogonal to $\mathbf{z}$ [59]. By updating the QR decomposition of $\hat{R}$, we can apply the algorithm (1.6) again to find $\hat{\mathbf{z}}$. An orthonormal basis for the numerical kernel $\mathscr{K}_\theta(A)$ can be obtained by continuing this process.

Numerical rank can also be considered a generalization of the conventional rank since $rank(A) = rank_\theta(A) = k$ whenever $0 < \theta < \sigma_k$. A matrix A in practical computation is usually known in a perturbed form $\hat{A} = A + E$ where E is the noise that can be expected to be small. Let $\sigma_j(\cdot)$ denote the j-th singular value of the matrix $(\cdot)$. It is known that [85, Chapter 1, Corollary 4.31]

$$|\sigma_j(A+E) - \sigma_j(A)| \leq \|E\|_2, \quad j = 1, \cdots, n.$$

Consequently, the underlying (exact) rank $rank(A)$ can be recovered as $rank_\theta(A)$ as long as

$$\|E\|_2 < \theta < \sigma_k(A) - \|E\|_2.$$

In practical computation, the choice of the threshold θ is problem dependent, and may be difficult to decide in some cases. The general rule is that θ should

be above the expected noise magnitude $\|E\|_2$ and below the smallest positive (unknown) singular value $\sigma_k(A)$.

The formulation of the numerical rank is intended to answer the following question: *How to find the rank of a matrix A that is under a small perturbation?* The answer is conditional: *If the perturbation is sufficiently small, then the rank of A can be recovered as $rank_\theta(A)$ for a threshold slightly larger than the perturbation*. Numerical rank-revealing would become intractable when the window $\sigma_k(A) - \|E\|_2 > \theta > \|E\|_2$ of choosing the threshold θ disappears.

Consider Example 1.1.1 again. For any threshold θ chosen within the interval $10^{-n} < \theta < 9$, the matrix G in (1.2) is of numerical rank n, or numerical nullity 1 within θ.

1.1.3 The linear and nonlinear least squares problems

Conventional solutions do not exist for an overdetermined linear system

$$A\mathbf{x} = \mathbf{b}$$

when the matrix $A \in \mathbb{C}^{m\times n}$ for $m > n$ unless, for a zero probability, the vector $\mathbf{b}$ happens to be in the range of A. Instead, the *least squares solution* $\mathbf{x}_*$ becomes the alternative that satisfies

$$\left\|A\mathbf{x}_* - \mathbf{b}\right\|_2^2 = \min_{\mathbf{y}\in\mathbb{C}^n} \left\|A\mathbf{y} - \mathbf{b}\right\|_2^2, \tag{1.7}$$

The least squares solution is unique: $\mathbf{x}_* = A^+\mathbf{b}$ when A is of full rank, where $A^+ = (A^H A)^{-1}A^H$ is the *pseudo-inverse* of A.

Although solving $A\mathbf{x} = \mathbf{b}$ for its least squares solution is equivalent to solving the normal equation $(A^H A)\mathbf{x} = A^H\mathbf{b}$, there is a fundamental difference between symbolic computation and numerical approach from here. Solving the normal equation directly may be a natural approach using exact arithmetic. Numerical computation, where accuracy is a concern, goes a step further by solving the first n equations of $R\mathbf{x} = Q^H\mathbf{b}$ after obtaining the QR decomposition $A = QR$ (c.f. [34, §5.3]). Both approaches are equivalent in theory but not much so in practical computations. It is neither wise nor necessary to construct or to solve the normal equation as it is in numerical computation. On the other hand, the standard numerical approach of solving $R\mathbf{x} = Q^H\mathbf{b}$ is not attractive for symbolic computation due to the square roots required in the QR decomposition.

Solving linear least squares solutions is essential in one of the most basic operations in numerical polynomial algebra: Polynomial division in floating-point arithmetic. As discussed in Example 1.1.1, dividing a polynomial f by g for the quotient q and the remainder r in the form of $f = g\cdot q + r$ can easily be ill-conditioned. Consequently, the synthetic division is prohibitively unstable in numerical computation. However, if the remainder is known, say $r = 0$, then finding

q in the equation $g \cdot q = f$ is a stable linear least squares problem

$$C(g)\mathbf{q} \;=\; \mathbf{f} \tag{1.8}$$

where $C(g)$ is the convolution matrix [17] representing the linear transformation $\mathscr{L} : u \longrightarrow g \cdot u$ between relevant polynomial vector spaces in which vectors $\mathbf{q}$ and $\mathbf{f}$ represent q and f respectively. Equation (1.8) can be accurately solved in numerical computation for its least squares solution, avoiding the difficulty of synthetic division. This *least squares division* plays an indispensable role in many works such as [30, 98, 102].

The least squares problem also arises in polynomial algebra (c.f. [48, 98, 102] and Example 1.1.3 below) in a nonlinear form of solving an overdetermined system of (nonlinear) equations

$$F(\mathbf{z}) \;=\; \mathbf{b}, \quad \mathbf{z} \in \mathbb{C}^n \tag{1.9}$$

for an analytic mapping $F : \mathbb{C}^n \to \mathbb{C}^m$ with $m > n$. Similar to the linear case, solving (1.9) requires seeking the least squares solution $\mathbf{z}_*$ defined by

$$\big\|F(\mathbf{z}_*) - \mathbf{b}\big\|_2^2 \;=\; \min_{\mathbf{y}\in\mathbb{C}^n} \big\|F(\mathbf{y}) - \mathbf{b}\big\|_2^2.$$

The necessary condition for $\mathbf{z}_*$ to be a (local or global) least squares solution is

$$J(\mathbf{z})^{\mathsf{H}} F(\mathbf{z}) \;=\; \mathbf{0} \tag{1.10}$$

where $J(\mathbf{z})$ is the Jacobian of $F(\mathbf{z})$ (cf. [20, 98]). Although the global minimum would be attractive theoretically, a local minimum is usually sufficient in practical computations, and the local minimum is global if the residual $\|F(\mathbf{z}_*) - \mathbf{b}\|_2$ is tiny. In principle, the least squares solution of (1.9) can be solved by many nonlinear optimization methods. There is a distinct feature of the overdetermined systems arising from numerical polynomial algebra: The residual $\|F(\mathbf{z}_*) - \mathbf{b}\|_2$ is expected to be small (*e.g.* an approximate factorization $p_1^{m_1} \cdots p_k^{m_k}$ of f is expected to satisfy $\big\|p_1^{m_1} \cdots p_k^{m_k} - f\big\| \ll 1$). As a result, a simple optimization method, the Gauss-Newton iteration, is particularly effective.

The Gauss-Newton iteration is given as follows: From an initial iterate $\mathbf{z}_0$,

$$\mathbf{z}_k \;=\; \mathbf{z}_{k\text{-}1} - J(\mathbf{z}_{k\text{-}1})^{+}\big[F(\mathbf{z}_{k\text{-}1}) - \mathbf{b}\big], \quad k = 1, 2, \cdots. \tag{1.11}$$

The Gauss-Newton iteration is a natural generalization of the standard Newton iteration. Detailed studies of the Gauss-Newton iteration can be found in some special topic textbooks and articles, such as [18, 20, 63, 98].

The Gauss-Newton iteration is well defined in a neighborhood of the desired (local or global) minimum point $\mathbf{z}_*$ if the Jacobian $J(\mathbf{z}_*)$ is injective (or, equivalently, of nullity zero). The condition of being injective on $J(\mathbf{z}_*)$ is also essential to ensure the local convergence of the Gauss-Newton iteration, as asserted in the following lemma. Different from Newton's iteration for normally determined systems, however, the locality of the convergence consists *two* requirements: The initial iter-

ate $\mathbf{z}_0$ must be sufficiently near the least squares solution $\mathbf{z}_*$, while the residual $\|F(\mathbf{z}_*)-\mathbf{b}\|_2$ must be sufficiently small.

Lemma 1.1.2. [98, Lemma 2.8] *Let $\Omega \subset \mathbb{C}^m$ be a bounded open convex set and $F : D \subset \mathbb{C}^m \longrightarrow \mathbb{C}^n$ be analytic in an open set $D \supset \overline{\Omega}$. Let $J(\mathbf{z})$ be the Jacobian of $F(\mathbf{z})$. Assume $\mathbf{z}_* \in \Omega$ is a local least squares solution to system* (1.9)*. Let σ be the smallest singular value of $J(\mathbf{z}_*)$. Let $\delta \geq 0$ be a constant such that*

$$\left\| \left[J(\mathbf{z}) - J(\mathbf{z}_*) \right]^{\mathrm{H}} \left[F(\mathbf{z}_*) - \mathbf{b} \right] \right\|_2 \;\leq\; \delta \left\| \mathbf{z} - \mathbf{z}_* \right\|_2 \quad \textit{for all } \mathbf{z} \in \Omega.$$

If $\delta < \sigma^2$, then for any $c \in \left(\frac{1}{\sigma}, \frac{\sigma}{\delta}\right)$, there exists $\varepsilon > 0$ such that for all $\mathbf{z}_0 \in \Omega$ with $\|\mathbf{z}_0 - \mathbf{z}_*\|_2 < \varepsilon$, the sequence $\left\{\mathbf{z}_1, \mathbf{z}_2, \cdots\right\}$ generated by the Gauss-Newton iteration (1.11) *is well defined inside Ω, converges to $\mathbf{z}_*$, and satisfies*

$$\left\| \mathbf{z}_{k+1} - \mathbf{z}_* \right\|_2 \;\leq\; \frac{c\delta}{\sigma} \left\| \mathbf{z}_k - \mathbf{z}_* \right\|_2 + \frac{c\alpha\gamma}{2\sigma} \left\| \mathbf{z}_k - \mathbf{z}_* \right\|_2^2, \tag{1.12}$$

where $\alpha > 0$ is the upper bound of $\|J(\mathbf{z})\|_2$ on $\overline{\Omega}$, and $\gamma > 0$ is the Lipschitz constant of $J(\mathbf{z})$ in Ω, namely, $\|J(\mathbf{z}+\mathbf{h}) - J(\mathbf{z})\|_2 \leq \gamma \|\mathbf{h}\|$ for all $\mathbf{z}, \mathbf{z}+\mathbf{h} \in \Omega$.

Notice that the constant δ in (1.12) is proportional to the residual $\|F(\mathbf{z}_*)-\mathbf{b}\|_2$. Therefore, the smaller is this residual, the faster is the convergence. When the least squares solution $\mathbf{z}_*$ is a conventional solution, the residual will be zero and the convergence is quadratic.

The condition that $J(\mathbf{z}_*)$ is injective also implies that the smallest singular value $\sigma_{min}(J(\mathbf{z}_*))$ is strictly positive and provides an asymptotic sensitivity measurement [98]

$$\tau(F, \mathbf{z}_*) \;=\; \frac{1}{\sigma_{min}(J(\mathbf{z}_*))} \;\equiv\; \left\| J(\mathbf{z}_*)^+ \right\|_2 \tag{1.13}$$

for the least squares solution $\mathbf{z}_*$.

The Gauss-Newton iteration is extensively applied in the algorithms for numerical polynomial algebra in this survey. A generic Gauss-Newton iteration module is available in the software packages `ApaTools/Apalab` [99]. When formulating the overdetermined system $F(\mathbf{z}) = \mathbf{b}$, it is important to have enough equations to ensure $J(\mathbf{z}_*)$ is injective. Auxiliary equations are often needed for this purpose as shown in the following example.

Example 1.1.3. Consider the polynomial factorization problem. For simplicity of the exposition, assume f can be factorized in three factors $f = u^\alpha v^\beta w^\gamma$ where u, v and w are pairwise co-prime polynomials, and the objective is to compute the coefficient vectors $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ of u, v and w respectively. Naturally, the overdetermined system $G(\mathbf{u}, \mathbf{v}, \mathbf{w}) = \mathbf{0}$ with

$$G(\mathbf{u}, \mathbf{v}, \mathbf{w}) \;\equiv\; [\![u^\alpha v^\beta w^\gamma - f]\!] \tag{1.14}$$

needs to be solved, as pointed out in [48], where $[\![\cdot]\!]$ denotes the vector representation of the polynomial $(\cdot)$ in a given vector space. However, the Jacobian of

$G(\mathbf{u},\mathbf{v},\mathbf{w})$ in (1.14) is not injective since $G(\mathbf{u},\mathbf{v},\mathbf{w}) = G(c_1\mathbf{u},c_2\mathbf{v},c_3\mathbf{w})$ as long as $c_1^\alpha c_2^\beta c_3^\gamma = 1$. More constraints are therefore needed. A simple remedy we typically employ is to include extra equations $\mathbf{a}^{\mathsf{H}}\mathbf{u} - 1 = \mathbf{b}^{\mathsf{H}}\mathbf{v} - 1 = 0$ with certain vectors $\mathbf{a}$ and $\mathbf{b}$ of proper dimensions. The only restriction for choosing $\mathbf{a}$ and $\mathbf{b}$ is to avoid $\mathbf{a}^{\mathsf{H}}\mathbf{u} = \mathbf{b}^{\mathsf{H}}\mathbf{v} = \mathbf{0}$ near the solution u and v. They can be random vectors, or more preferably the scalar multiples of the initial approximations to $\mathbf{u}$ and $\mathbf{v}$ respectively. Then the Jacobian $J(\mathbf{u},\mathbf{v},\mathbf{w})$ of the analytic mapping

$$F(\mathbf{u},\mathbf{v},\mathbf{w}) \;=\; \begin{bmatrix} \mathbf{a}^{\mathsf{H}}\mathbf{u} - 1 \\ \mathbf{b}^{\mathsf{H}}\mathbf{v} - 1 \\ [\![u^\alpha v^\beta w^\gamma - f]\!] \end{bmatrix}$$

can be easily proved as injective at the desired solution $(\mathbf{u},\mathbf{v},\mathbf{w})$: Assume

$$J(\mathbf{u},\mathbf{v},\mathbf{w}) \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \\ \mathbf{r} \end{bmatrix} \;=\; \mathbf{0} \tag{1.15}$$

where $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{r}$ are vector representations of polynomials p, q and r respectively. From straightforward differentiations, we have

$$\begin{aligned} \alpha u^{\alpha\text{-}1} v^\beta w^\gamma p + \beta u^\alpha v^{\beta\text{-}1} w^\gamma q + \gamma u^\alpha v^\beta w^{\gamma\text{-}1} r = 0 \\ \mathbf{a}^{\mathsf{H}}\mathbf{p} \;=\; \mathbf{b}^{\mathsf{H}}\mathbf{q} = 0. \end{aligned}$$

Consequently $\alpha pvw + \beta uqw + \gamma uvr = 0$, which leads to $p = g_1 u$, $q = g_2 v$ and $r = g_3 w$ since u, v, and w are pairwise co-prime. We further know that g_1, g_2 and g_3 must be constants from (1.15). Then $\mathbf{a}^{\mathsf{H}}\mathbf{p} = \mathbf{b}^{\mathsf{H}}\mathbf{q} = 0$ and $\mathbf{a}^{\mathsf{H}}\mathbf{u} = \mathbf{b}^{\mathsf{H}}\mathbf{v} = 1$ implies $g_1 = g_2 = 0$ and thus $p = q = 0$. Furthermore $\gamma uvr = 0$ implies $r = 0$. Therefore $J(\mathbf{u},\mathbf{v},\mathbf{w})$ is injective. □

Appending extra equations as shown in Example 1.1.3 is a typical strategy for making the Jacobian injective before applying the Gauss-Newton iteration. The approaches of proving the injectiveness are also similar among different problems.

1.2 Formulation of the approximate solution

1.2.1 The ill-posed problem and the pejorative manifold

Hadamard characterized a problem as *well-posed* if its solution satisfies existence, uniqueness, and continuity with respect to data [36]. A problem whose solution lacks one of the three properties is termed an *ill-posed problem*. In the current literature, the term ill-posed problem mainly refers to those having solutions infinitely sensitive to data perturbations [19]. Contrary to Hadamard's misbelief that ill-posed problems are artificial and unsuitable for modeling physical systems, this type of problems arise quite often in science and engineering applications (cf. *e.g.* [37,

§1.2]). As one of the main challenges for numerical computation, algebraic problems are commonly ill-posed. Such examples are abundant:

- *Univariate polynomial factorization:* Under an arbitrary tiny perturbation, a polynomial $p(x) = (x-x_1)^{m_1} \cdots (x-x_k)^{m_k}$ in expanded form loses those repeated factors corresponding to multiplicities m_j's higher than one, and multiple roots turn into clusters of simple roots.
- *Polynomial GCD:* For a given pair of polynomials (p,q) under arbitrary perturbations, its GCD generically degrades to a trivial constant polynomial even if (p,q) originally possesses a GCD of positive degree.
- *Irreducible factorization of multivariate polynomials:* For a multivariate polynomial $p = f_1^{\alpha_1} f_2^{\alpha_2} \cdots f_k^{\alpha_k}$ with nontrivial factors $f_1, \ldots, f_k$, the factorizability is lost and the polynomial becomes irreducible under an infinitesimal but arbitrary perturbation. The irreducible factorization of polynomial p is thus discontinuous, preventing a meaningful solution using conventional methods at presence of data perturbation.
- *The matrix rank and kernel:* As perhaps the most basic ill-posed problem (c.f. §1.1.2), the rank and kernel of a matrix are discontinuous when the matrix is rank-deficient. A tiny perturbation will generically destroy the kernel entirely and turns the matrix into a full-ranked one. The ill-posedness of matrix rank/kernel may be under-noticed because the singular value decomposition is available and effective. The capability of identifying the *numerical* rank and kernel is one of the very cornerstones of methods for solving ill-posed problems. This ill-posed problem extends to solving a linear system $A\mathbf{x} = \mathbf{b}$ where the matrix A is rank-deficient. A perturbation generically renders the system unsolvable in exact sense even if the original system has infinitely many solutions.
- *The matrix Jordan Canonical Form:* For an $n \times n$ matrix A, there is a Jordan canonical decomposition $A = XJX^{-1}$ where J is the block diagonal Jordan matrix. When any of the Jordan block in J is larger than 1×1, however, the perturbed matrix $A+E$ generically loses all the non-trivial Jordan structure of A, making it extremely difficult to compute the Jordan decomposition and the underlying multiple eigenvalues.

In a seminal technical report [43] that has never been formally published, Kahan studied three ill-posed problems (rank-deficient linear system, multiple roots of polynomials and multiple eigenvalues of matrices) and pointed out that it may be a misconception to consider ill-posed problems as hypersensitive to data perturbations. The collection of those problems having solutions of a common structure forms what Kahan calls a *pejorative manifold*. An artificial perturbation pushes the problem away from the manifold in which it originally resides and destroys its solution structure. Kahan proves, however, the solution may not be sensitive at all if the problem is perturbed with a restriction that it surfs in the pejorative manifold it belongs.

Kahan's observation of pejorative manifolds extends to other ill-posed problems such as those in the following examples. Furthermore, it has become known that these manifolds may have a certain *stratification* structure.

Example 1.2.1. In the vector space $\mathbb{C}^{m\times n}$ for $m \geq n$, the collection of rank-k matrices forms a pejorative manifold

$$\mathscr{M}_k^{m\times n} \;=\; \big\{A \in \mathbb{C}^{m\times n} \;\big|\; rank(A) = k\big\}.$$

Counting the dimension from the basis for the column space $(m\cdot k)$ and the remaining columns as linear combinations of the basis $((n-k)\cdot k)$, it is easy to see *[58]* that the codimension $codim(\mathscr{M}_k^{m\times n}) = (m-k)(n-k)$. Those manifolds form a stratification structure

$$\overline{\mathscr{M}_0^{m\times n}} \;\subset\; \overline{\mathscr{M}_1^{m\times n}} \;\subset\; \cdots \;\subset\; \overline{\mathscr{M}_n^{m\times n}} \equiv \mathbb{C}^{m\times n},$$

where $\overline{(\cdot)}$ denotes the closure of the set $(\cdot)$. □

Example 1.2.2. Associating a monic polynomial

$$p(x) = x^4 + p_1x^3 + p_2x^2 + p_3x + p_4$$

of degree 4 with the coefficient vector $\mathbf{p} = [p_1, p_2, p_3, p_4]^\top \in \mathbb{C}^4$, the set of all polynomials possessing a common factorization structure forms a factorization manifold. For instance

$$\Pi(1,3) \;=\; \big\{(x-z_1)^1(x-z_2)^3 \;\big|\; z_1, z_2 \in \mathbb{C},\; z_1 \neq z_2\big\}$$

with codimension $codim(\Pi(1,3)) = 2$. On the other hand, manifold $\Pi(1,3)$ is in the closure of manifold

$$\Pi(1,1,2) \;=\; \big\{(x-z_1)^1(x-z_2)^1(x-z_3)^2 \;\big|\; z_1, z_2, z_3 \in \mathbb{C},\; z_i \neq z_j \text{ for } i \neq j\big\}$$

with $codim\big(\Pi(1,1,2)\big) = 1$ since

$$\lim_{\varepsilon\to 0}(x-z_1)^1(x-z_2)^1(x-z_2+\varepsilon)^2 \;=\; (x-z_1)^1(x-z_2)^3$$

Likewise $\Pi(1,1,2) \subset \overline{\Pi(1,1,1,1)} \equiv \mathbb{C}^4$, and the five manifolds form a *stratification* as shown in Figure 1.1. Moreover, each manifold can be parametrized in the form of $\mathbf{p} = F(\mathbf{z})$. For instance, $\Pi(1,3)$ is parametrized as

$$\mathbf{p} \;=\; [\![(x-z_1)(x-z_2)^3]\!],$$

namely,

$$\begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \;=\; \begin{bmatrix} -3z_2 - z_1 \\ 3z_2^2 + 3z_1z_2 \\ -z_2^3 - 3z_1z_2^2 \\ z_1z_2^3 \end{bmatrix}$$

The Jacobian of such $F(\mathbf{z})$ is injective in the corresponding manifold, ensuring the local Lipschitz continuity of the root vector $\mathbf{z} = [z_1, z_2]^\top$ with respect to the coefficients $\mathbf{p}$ as well as the capability of applying the Gauss-Newton iteration for refining the multiple roots [98]. □

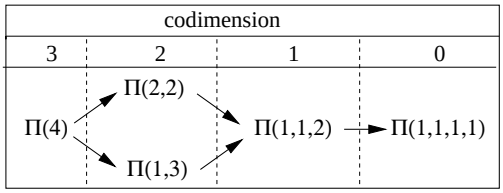

Fig. 1.1 Stratification of manifolds of degree 4 polynomials, with "$\longrightarrow$" denoting "in the closure of"

Example 1.2.3. Consider univariate polynomial pairs (p,q) of degrees m and n respectively with $m \geq n$. Let $\mathscr{P}_k^{m,n}$ be the collection of those pairs having GCDs of a common degree k:

$$\mathscr{P}_k^{m,n} = \big\{ (p,q) \in \mathbb{C}[x] \times \mathbb{C}[x] \;\big|\; deg(p) = m,\ \ deg(q) = n,\ \ deg\big(gcd(p,q)\big) = k \big\} \tag{1.16}$$

where $gcd(p,q)$ is the GCD of the polynomial pair (p,q). Every polynomial pair $(p,q) \in \mathscr{P}_k^{m,n}$ can be written as $p = uv$ and $q = uw$ where u is a monic polynomial of degree k. Thus it is easy to see that $\mathscr{P}_k^{m,n}$ is a manifold of the dimension $k + (m-k+1) + (n-k+1)$, or codimension k exactly, and those GCD manifolds form a stratification structure

$$\overline{\mathscr{P}_n^{m,n}} \subset \overline{\mathscr{P}_{n\text{-}1}^{m,n}} \subset \cdots \subset \overline{\mathscr{P}_0^{m,n}} \equiv \mathbb{C}^{m+n+2}. \tag{1.17}$$

Furthermore, each manifold can again be parametrized in the form of $\mathbf{u} = F(\mathbf{z})$. In fact,

$$\mathbf{u} \;=\; \begin{bmatrix} \mathbf{p} \\ \mathbf{q} \end{bmatrix} \;=\; \begin{bmatrix} [\![uv]\!] \\ [\![uw]\!] \end{bmatrix} \;=\; F(\mathbf{z})$$

with $\deg(u) = k$, where $\mathbf{z}$ is a vector consists of the coefficients of u, v and w except the leading coefficient 1 of u. Also similar to Example 1.2.2, the Jacobian of $F(\mathbf{z})$ is injective, ensuring the local Lipschitz continuity of (u,v,w) with respect to (p,q) on the manifold $\mathscr{P}_k^{m,n}$ as well as the applicability of the Gauss-Newton iteration for refinement. □

In summary, there exist similar geometric structures for many ill-posed problems in polynomial algebra such as multivariate GCD, multivariate square-free factorization, and multivariate irreducible factorization: The collection of the problems sharing a common solution structure forms a pejorative manifold $\mathscr{P}$ that can be

parametrized in the form of $\mathbf{u} = F(\mathbf{z})$ where the Jacobian of $F(\mathbf{z})$ is injective on the manifold $\mathscr{P}$. In addition, the pejorative manifolds form a stratification structure in which a manifold is in the closure of some other manifolds of lower codimension. As a result, a problem may be near many manifolds of lower codimension if it is near (or resides in) one particular pejorative manifold.

When an ill-posed problem with inexact data is encountered and it needs to be solved approximately using floating-point arithmetic, the first and foremost question is the meaning of "solving the problem". Computing the exact solution of an inexact ill-posed problem, such as calculating the trivial constant GCD, does not make much sense in practical applications. On the other hand, approximate solutions need to possess continuity. That is, the approximate solution must converge to the exact solution if the problem noise approaches zero. It appears reasonable to set the objective of solving an ill-posed problem numerically as follows:

Let P be a given problem that is a small perturbation from an ill-posed problem $\hat{P}$ residing in a pejorative manifold Π with exact solution $\hat{S}$. Find an approximate solution $\tilde{S}$ of P in the sense that $\tilde{S}$ is the exact solution of a certain problem $\tilde{P}$ where $\tilde{P}$ belongs to the same manifold Π and $\|\tilde{S}-\hat{S}\| = O(\|P-\hat{P}\|)$.

1.2.2 The three-strikes principle for removing ill-posedness

We shall use the univariate factorization problem in Example 1.2.2 as a case study for a rigorous formulation of the approximate solution that removes the ill-posedness. When the given polynomial p is a small perturbation from $\hat{p} = (x-z_1)(x-z_2)^3$ that belongs to the factorization manifold $\Pi(1,3)$ of codimension 2, the stratification structure as shown in Figure 1.1 indicates that p is also near two other manifolds $\Pi(1,1,2)$ and $\Pi(1,1,1,1)$ of lower codimensions 1 and 0 respectively. Here the distance $\delta(p,\Pi)$ of p from a manifold can be naturally defined as

$$\delta(p,\Pi) \;=\; \inf_{q\in\Pi} \big\|p-q\big\|.$$

Actually, the polynomial p is generically closer to those "other" manifolds than it is to the correct one:

$$\delta(p,\Pi(1,3)) \;\geq\; \delta(p,\Pi(1,1,2)) \;\geq\; \delta(p,\Pi(1,1,1,1)).$$

Let μ be the minimum of the distances between p and manifolds $\Pi(2,2)$ and $\Pi(4)$ and assume the perturbation is sufficiently small such that

$$\|p-\hat{p}\| \;\leq\; \delta(p,\Pi(1,3)) \;\ll\; \mu.$$

Then the desired manifold $\Pi(1,3)$ stands out as the *highest codimension* manifold among all the manifolds that intersect the θ-neighborhood of p for every θ sat-

isfying $\delta(p,\Pi(1,3)) < \theta < \mu$. That is, the key to identifying the correct manifold is to seek the manifold of the *highest codimension* within a proper threshold θ.

Upon identifying the pejorative manifold $\Pi(1,3)$, it is natural to look for $\tilde{p} = (x-\tilde{z}_1)(x-\tilde{z}_2)^3 \in \Pi(1,3)$ that minimizes the distance $\|p-\tilde{p}\|$, and take the exact factorization $(x-\tilde{z}_1)(x-\tilde{z}_2)^3$ of $\tilde{p}$ as the *approximate factorization* of p.

Definition 1.2.4. Let p be a given polynomial of degree n and let $\theta > 0$. Assume $m_1 + \cdots + m_k = n$ and $\Pi(m_1,\ldots,m_k)$ is of the highest codimension among all the factorization manifolds in $\mathbb{C}^n$ that intersect the θ-neighborhood of p. Then the exact factorization of $\tilde{p} \in \overline{\Pi(m_1,\ldots,m_k)}$ is called the approximate factorization of p if

$$\big\|p-\tilde{p}\big\| \;=\; \min_{q\in\overline{\Pi(m_1,\cdots,m_k)}} \big\|p-q\big\|.$$

A theoretical elaboration of the univariate factorization and its regularization is presented in a recent paper [101]. Generally, solving an ill-posed algebraic problem starts with formulating the *approximate solution* following the "three-strikes" principle below to remove the discontinuity:

Backward nearness: The approximate solution to the given (inexact) problem P is the exact solution of a nearby problem $\tilde{P}$ within a distance θ of P.

Maximum codimension: The nearby problem $\tilde{P}$ is in the closure $\overline{\Pi}$ of the pejorative manifold Π of the highest codimension among all the pejorative manifolds intersecting the θ-neighborhood of the given problem P.

Minimum distance: The nearby problem $\tilde{P}$ is the nearest point in the closure $\overline{\Pi}$ of Π to the given problem P

Based on these principles, the *approximate GCD* of a polynomial pair can be defined similarly using the notation in Example 1.2.3.

Definition 1.2.5. [96] Let (p,q) be a given polynomial pair of degree m and n respectively, and let $\theta > 0$ be a given GCD threshold. Assume k is the maximum codimension among all the GCD manifolds $\mathscr{P}_0^{m,n}$, $\mathscr{P}_1^{m,n}$, $\ldots$, $\mathscr{P}_n^{m,n}$ embedded in $\mathbb{C}^{m+n+2}$ that intersect the θ-neighborhood of (p,q):

$$k \;=\; \max_{\delta((p,q),\mathscr{P}_j^{m,n})<\theta} codim\big(\mathscr{P}_j^{m,n}\big)$$

Then the exact GCD of $(\tilde{p},\tilde{q}) \in \overline{\mathscr{P}_k^{m,n}}$ is called the approximate GCD of (p,q) within θ if

$$\big\|(p,q)-(\tilde{p},\tilde{q})\big\| \;=\; \min_{(f,g)\in\overline{\mathscr{P}_k^{m,n}}} \big\|(p,q)-(f,g)\big\|. \tag{1.18}$$

Remark: The univariate GCD is the first ill-posed problem in numerical polynomial algebra that has been going through rigorous formulations. In 1985, Schönhage [77] proposed the *quasi-GCD* for univariate polynomials in a definition

requiring only the backward nearness. Schönhage also assumes the given polynomial pair is inexact but arbitrarily precise. Similar formulations are later used in [14, 30, 39, 67, 68, 72] with some variations. In 1995, Corless, Gianni, Trager and Watt [11] first noticed the importance of the "highest degree" requirement of the approximate GCD in addition to Schönhage's notion. This requirement is also adopted in [24]. In the same paper [11], Corless et al. also suggest seeking the minimum distance. In 1996/1998 Karmarkar and Lakshman [52, 53] formulated two numerical GCD problems: "The nearest GCD problem" and the "highest degree approximate common divisor problem" in detail, with the latter specifying requirements of backward nearness, highest degree, and minimum distance. The formulation in Definition 1.2.5 differs somewhat with Karmarkar-Lakshman's. The polynomial pair $(\tilde{p},\tilde{q})$ is on the *closure* of the manifold of highest degree polynomial pairs and not necessarily monic. There is a different type of numerical GCD formulation: The "nearest GCD problem". Originated by Karmarkar and Lakshman [53, p. 654], this problem seeks the nearest polynomial pairs possessing an exact nontrivial GCD. Kaltofen, Yang and Zhi (cf. *e.g.* [49, 50]) generalized this formulation: Given a polynomial pair (p,q) and an integer $k>0$, find the polynomial pair $(\tilde{p},\tilde{q})$ that is nearest from (p,q) such that the (exact) GCD of $(\tilde{p},\tilde{q})$ is of degree k or higher. This problem is equivalent to the constrained minmization (1.18) as a stand-alone problem with no need to specify a tolerance θ or to maximize the manifold codimension. □

In a straightforward verification, the formulation of the numerical kernel of a matrix in §1.1.2 conforms with the "three-strikes" principle. We can formulate the approximate multivariate GCD, the approximate square-free factorization, the approximate irreducible factorization, the approximate Jordan Canonical Form and other ill-posed problems the same way as and Definition 1.2.4 and Definition 1.2.5 according to the principles of backward nearness, maximum codimension and minimum distance.

Computing the approximate solution as formulated above involves identification of the pejorative manifold of the highest codimension within the given threshold as well as solving a least squares problem to obtain the minimum distance. Algorithms can be developed using a two-staged approach: Finding the manifold with matrix computation, followed by applying the Gauss-Newton iteration to obtain the approximate solution.

Example 1.2.6. The effectiveness of this formulation and the robustness of the corresponding two-staged algorithms can be illustrated by the polynomial factorization problem for

$$\begin{aligned} p(x) &= x^{200} - 400x^{199} + 79500x^{198} + \ldots + 2.04126914035338\cdot 10^{86}x^{100} \\ &\quad -3.55467815448396\cdot 10^{86}x^{99} + \ldots + 1.261349023419937\cdot 10^{53}x^{2} \\ &\quad -1.977831229290266\cdot 10^{51}x + 1.541167191654753\cdot 10^{49} \\ &\approx (x-1)^{80}(x-2)^{60}(x-3)^{40}(x-4)^{20} \end{aligned} \tag{1.19}$$

whose coefficients are rounded to hardware precision. A new algorithm UVFACTOR *[99]* designed for computing the approximate factorization outputs the precise factorization structure and accurate factors within a threshold $\theta = 10^{-10}$, along with error estimates and a sensitivity measurement. This result is a substantial improvement over the previous algorithm MULTROOT [97, 98] which is limited to extracting factors of multiplicity under 30. In contrast, standard methods like Matlab function `roots` output scattered root clusters, as shown in Figure 1.2.2.

```
>> [F,res,cond] = uvfactor(f,1e-10);

THE CONDITION NUMBER:                          2.57705
THE BACKWARD ERROR:                      7.62e-016
THE ESTIMATED FORWARD ROOT ERROR:        3.93e-015

  FACTORS

   ( x -   4.000000000000008 )^20
   ( x -   2.999999999999994 )^40
   ( x -   2.000000000000002 )^60
   ( x -   1.000000000000000 )^80
```

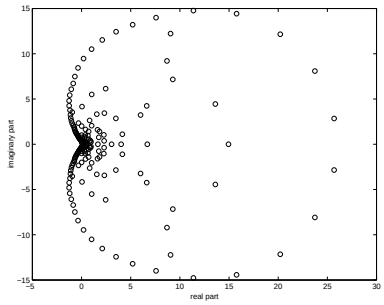

Fig. 1.2 Matlab results for the polynomial (1.19)

□

Pejorative manifolds in numerical polynomial algebra can often be parametrized in the form of $\mathbf{u} = F(\mathbf{z})$ with injective Jacobians, as shown in previous examples. In such cases, it is our conjecture that the approximate solution formulated based on backward nearness, maximum codimension and minimum distance is well-posed in the following sense: If the given problem P is a perturbation from an (exact) problem $\hat{P}$ with sufficiently small error ε, then there is an upper bound μ on the threshold θ. As long as θ is chosen in the interval (ε, μ), there is a unique approximate solution S of P within all such θ, and the solution S is continuous with respect to the problem P. Moreover, the approximate solution S converges to the exact solution $\hat{S}$ of $\hat{P}$ if P approaches to $\hat{P}$ with

$$\left\|S - \hat{S}\right\| \;=\; O\left(\left\|P - \hat{P}\right\|\right).$$

1.3 Matrix computation arising in polynomial algebra

As Stetter points out in [84] and in the title of [83], matrix eigenproblem is at the heart of polynomial system solving. Furthermore, matrix computation problems such as least squares and numerical rank/kernel computation also arise frequently and naturally in polynomial algebra. We shall survey the approximate GCD, factorization, multiplicity structure and elimination problems in this section and derive related matrix computation problems.

1.3.1 Approximate GCD

Given a polynomial pair $(p,q) \in \mathbb{C}[x] \times \mathbb{C}[x]$ of degrees m and n, respectively, with

$$\begin{aligned} p(x) &= p_0 + p_1 x + \cdots + p_m x^m \\ q(x) &= q_0 + q_1 x + \cdots + q_n x^n, \end{aligned}$$

we can write

$$p = uv \text{ and } q = uw$$

where $u = gcd(p,q)$ is the GCD along with cofactors v and w. Then

$$p \cdot w - q \cdot v = [p,\ q] \begin{bmatrix} w \\ -v \end{bmatrix} = 0. \tag{1.20}$$

That is, the polynomial pair (v,w) belongs to the kernel $\mathscr{K}(\mathscr{L}_{p,q})$ of the linear transformation

$$\mathscr{L}_{p,q} : (r,s) \longrightarrow p \cdot s - q \cdot r. \tag{1.21}$$

Moreover, it is clear that the kernel $\mathscr{K}(\mathscr{L}_{p,q})$ is spanned by polynomial pairs in the set $\left\{ (x^j v, x^j w) \;\middle|\; j = 0,1,\cdots \right\}$.

The classical Sylvester matrix

$$S(p,q) = \begin{array}{c} \overbrace{}^{n} \overbrace{}^{m} \\ \begin{pmatrix} p_0 & & q_0 & \\ p_1 & \ddots & q_1 & \ddots \\ \vdots & \ddots & p_0 & \vdots & \ddots & q_0 \\ p_m & & p_1 & q_n & & q_1 \\ & \ddots & \vdots & & \ddots & \vdots \\ & & p_m & & & q_n \end{pmatrix} \end{array} \tag{1.22}$$

is the matrix representation of the linear transformation $\mathscr{L}_{p,q}$ in (1.21) restricted in the domain $\left\{ (r,s) \;\middle|\; deg(r) < n,\ deg(s) < m \right\}$. It becomes clear that $S(p,q)$ has a nullity equals to $k = deg(u)$ since the kernel of the restricted $\mathscr{L}_{p,q}$ is spanned by

$\{(v, w),\ (xv, xw),\ \ldots,\ (x^{k-1}v, x^{k-1}w)\}$. Consequently, the (exact) GCD structure is represented as the nullity of the Sylvester matrix:

$$deg\big(gcd(p,q)\big) \;=\; nullity\,(S(p,q))\,.$$

For the problem of the *approximate* GCD, the polynomial pair (p,q) is considered a perturbation from $(\hat{p},\hat{q})$ residing in the GCD manifold $\mathscr{P}_k^{m,n}$ (cf. Example 1.2.3 in §1.2.1). Then

$$S(p,q) \;=\; S(\hat{p},\hat{q}) + S(p-\hat{p}, q-\hat{q}).$$

Namely, the Sylvester matrix $S(p,q)$ is near $S(\hat{p},\hat{q})$ of nullity k with a distance $\|S(p-\hat{p},q-\hat{q})\|_2$, and identifying the maximum codimension manifold $\mathscr{P}_k^{m,n}$ becomes the *numerical* rank/kernel problem of the Sylvester matrix.

After identifying the degree k of the approximate GCD, one can further restrict the domain of the linear transformation $\mathscr{L}_{p,q}$ as

$$\big\{(r,s)\;\big|\; deg(r) \le m-k,\ deg(s) \le n-k\big\}.$$

The corresponding Sylvester submatrix

$$S_k(p,q) \;=\; \overbrace{\begin{pmatrix} p_0 & & q_0 & \\ p_1 & \ddots & q_1 & \ddots \\ \vdots & \ddots & p_0 & \vdots & \ddots & q_0 \\ p_m & & p_1 & q_n & & q_1 \\ & \ddots & \vdots & & \ddots & \vdots \\ & & p_m & & & q_n \end{pmatrix}}^{n-k+1 \qquad\qquad m-k+1}$$

has a numerical kernel of dimension one and the coefficients of the cofactors v and w can be obtained approximately from the lone basis vector of the numerical kernel:

$$\mathscr{K}\big(S_k(p,q)\big) \;=\; span\left\{\begin{bmatrix} [\![v]\!] \\ -[\![w]\!] \end{bmatrix}\right\}.$$

An *estimate* of the coefficients of the approximate GCD u can then be obtained by solving for the least squares solution u in the linear system $u\cdot v = f$ instead of the unstable synthetic division.

The approximate GCD computed via numerical kernel and linear least squares alone may not satisfy the minimum distance requirement. The accuracy of the computed GCD depends on the numerical condition of the Sylvester submatrix $S_k(p,q)$, while the GCD sensitivity measured from (1.13) can be much healthier. Therefore, the above matrix computation serves as stage I of the approximate GCD finding. The Gauss-Newton iteration (c.f. §1.1.3) is needed to ensure the highest attainable accuracy.

For a simple example [96], consider the polynomial pair

$$f(x) = x^7 - .999999999999x^6 + x^5 - .999999999999x^4 + x^3 + .999999999999x^2 + x - .999999999999$$
$$g(x) = x^6 - 3x^3 - x^5 + 2x^2 + x^4 - 2x + 2$$

possessing an exact GCD as $gcd(f,g) = x^2 + 1$. Matrix computation alone can only produce $u \approx x^2 + 0.99992$ with four-digit accuracy, while the Gauss-Newton iteration attains the solution with a high accuracy near hardware precision. The GCD condition number 3.55 shows the approximate GCD is not sensitive at all, but the kernel sensitivity for the Sylvester matrix is quite high ($\approx 10^{12}$).

The upper-bound of the distance to rank-deficiency for the Sylvester matrix

$$\begin{aligned} \|S(p-\hat{p}, q-\hat{q})\|_2 &\leq \|S(p-\hat{p}, q-\hat{q})\|_F \\ &= \sqrt{n\|p-\hat{p}\|_2^2 + m\|q-\hat{q}\|_2^2} \;=\; \theta, \end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm [34, §2.3] of a matrix. However, rank-deficiency within the above threshold θ is only the necessary condition for the given (p,q) to be near a GCD manifold. The converse is not true. This is another reason why the iterative refinement is needed as Stage II for computing the minimum distance and certifying the approximate GCD.

There is also a different approach for GCD computation [38, 47, 50, 104]: Computing displacement polynomials Δp and Δq such that the Sylvester matrix (or submatrix) $S(p+\Delta p, q+\Delta q)$ has a specified nullity. This approach leads to a total least squares problem with special structures. This approach may have an advantage when there is a large distance from (p,q) to the GCD manifold beyond the convergence domain of the Gauss-Newton iteration, and it theoretically seeks a global minimum in comparison with the local minimum of the Gauss-Newton iteration. An apparent disadvantage of this approach is that it is subject to the sensitivity of the matrix kernel, not the actual GCD condition.

Both approaches can extend to multivariate GCD in a straightforward generalization. However, the matrix sizes may become huge when the number of indeterminates increases, and it may become necessary to reduce those sizes for the consideration of both storage and computing time. A subspace strategy for this purpose shall be discussed later in §1.4.2.

1.3.2 The multiplicity structure

For a polynomial ideal (or system) $I = \langle f_1, f_2 \ldots, f_t \rangle \subset \mathbb{C}[x_1, \ldots, x_s]$ with an isolated zero $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_s)$, the study on the multiplicity of I at $\hat{\mathbf{x}}$ traces back to Newton's time with evolving formulations [26, pp. 127-129][60, 64, 84]. Several computing methods for identifying the multiplicity have been proposed in the literature, such as [54, 62, 65, 88] and more recently in [6, 17, 94]. Here we elaborate the dual basis approach that can be directly adapted to numerical kernel computation. This approach is originated by Macaulay [60] in 1916 and reformulated by

Gröbner in 1939. For that reason it is also (somewhat inappropriately) referred to as the Gröbner duality [64, §31.3].

A univariate polynomial $f(x)$ has an m-fold zero $\hat{x}$ if

$$f(\hat{x}) = f'(\hat{x}) = \cdots = f^{(m\text{-}1)}(\hat{x}) = 0 \quad \text{and} \quad f^{(m)}(\hat{x}) \neq 0.$$

The Macaulay/Gröbner duality can be considered a generalization of this multiplicity to multivariate cases using partial differentiation functionals.

For an index array $\mathbf{j} = [j_1, \cdots, j_s] \in \mathbb{N}^s$ of non-negative integers, denote

$$\mathbf{x}^{\mathbf{j}} = x_1^{j_1} \cdots x_s^{j_s} \quad \text{and} \quad (\mathbf{x} - \mathbf{y})^{\mathbf{j}} = (x_1 - y_1)^{j_1} \cdots (x_s - y_s)^{j_s}.$$

Define a (differential) monomial functional at $\hat{\mathbf{x}} \in \mathbb{C}^s$ as

$$\partial_{\mathbf{j}}[\hat{\mathbf{x}}] : \mathbb{C}[\mathbf{x}] \longrightarrow \mathbb{C}, \quad \text{where } \partial_{\mathbf{j}}[\hat{\mathbf{x}}](p) = (\partial_{\mathbf{j}} p)(\hat{\mathbf{x}}) \quad \text{for any } p \in \mathbb{C}[\mathbf{x}].$$

Here, the differentiation operator

$$\partial_{\mathbf{j}} \equiv \partial_{j_1 \cdots j_s} \equiv \partial_{x_1^{j_1} \cdots x_s^{j_s}} \equiv \frac{1}{j_1! \cdots j_s!} \frac{\partial^{j_1 + \cdots + j_s}}{\partial x_1^{j_1} \cdots \partial x_s^{j_s}}. \tag{1.23}$$

Namely, the monomial functional $\partial_{\mathbf{j}}[\hat{\mathbf{x}}]$ applying to a polynomial p equals the partial derivative $\partial_{\mathbf{j}}$ of p evaluated at $\hat{\mathbf{x}}$. We may use $\partial_{\mathbf{j}}$ for $\partial_{\mathbf{j}}[\hat{\mathbf{x}}]$ when the zero $\hat{\mathbf{x}}$ is clear from context. A (differential) functional at $\hat{\mathbf{x}} \in \mathbb{C}^s$ is a linear combination of those $\partial_{\mathbf{j}}[\hat{\mathbf{x}}]$'s. The collection of all functionals at the zero $\hat{\mathbf{x}}$ that vanish on the entire ideal I forms a vector space $\mathscr{D}_{\hat{\mathbf{x}}}(I)$ called the *dual space* of I at $\hat{\mathbf{x}}$

$$\mathscr{D}_{\hat{\mathbf{x}}}(I) \equiv \left\{ c = \sum_{\mathbf{j} \in \mathbb{N}^s} c_{\mathbf{j}} \partial_{\mathbf{j}}[\hat{\mathbf{x}}] \;\middle|\; c(f) = 0, \text{ for all } f \in I \right\} \tag{1.24}$$

where $c_{\mathbf{j}} \in \mathbb{C}$ for all $\mathbf{j} \in \mathbb{N}^s$. The dimension of the dual space $\mathscr{D}_{\hat{\mathbf{x}}}(I)$ is the *multiplicity* of the zero $\hat{\mathbf{x}}$ to the ideal I. The dual space itself forms the multiplicity structure of I at $\hat{\mathbf{x}}$.

For example, a univariate polynomial p having an m-fold zero $\hat{x}$ if and only if the dual space of the ideal $\langle p \rangle$ at $\hat{x}$ is spanned by functionals ∂_0, ∂_1, …, $\partial_{m\text{-}1}$. The ideal $I = \langle x_1^3, x_1^2 x_2 + x_2^4 \rangle$ has a zero $(0,0)$ of multiplicity 12 with the dual space spanned by [17]

$$\overbrace{\partial_{00}}^{1}, \overbrace{\partial_{10}, \partial_{01}}^{2}, \overbrace{\partial_{20}, \partial_{11}, \partial_{02}}^{3}, \overbrace{\partial_{12}, \partial_{03}}^{2}, \overbrace{\partial_{13}, \partial_{04} - \partial_{21}}^{2}, \overbrace{\partial_{05} - \partial_{22}}^{1}, \overbrace{\partial_{06} - \partial_{23}}^{1} \tag{1.25}$$

and Hilbert function $\{1,2,3,2,2,1,1,0,\cdots\}$ as a partition of the multiplicity 12.

The most crucial requirement for the dual space is the *closedness condition*: For c to be a functional in the dual space $\mathscr{D}_{\hat{\mathbf{x}}}(I)$, not only $c(f_1) = \cdots = c(f_t) = 0$, but also $c(pf_i) = 0$ for all polynomial $p \in \mathbb{C}[\mathbf{x}]$ and for $i = 1, \ldots, t$. Since all differential functionals are linear, the closedness condition can be simplified to the system of linear equations

$$\sum_{\mathbf{k}\in\mathbb{N}^s,\ |\mathbf{k}|\le\alpha} c_{\mathbf{k}}\partial_{\mathbf{k}}[\hat{\mathbf{x}}]\big((\mathbf{x}-\hat{\mathbf{x}})^{\mathbf{j}} f_i\big) \;=\; 0 \qquad (1.26)$$
$$\text{for } \mathbf{j}\in\mathbb{N}^s,\ \mathbf{j}<\mathbf{k},\ \text{and } i\in\{1,\ldots,s\}.$$

in the coefficients $c_{\mathbf{j}}$'s for sufficiently large $\alpha\in\mathbb{N}$.

For each $\alpha\in\mathbb{N}$, the equations in (1.26) can be expressed as a homogeneous linear system in matrix form

$$S_\alpha\,\mathbb{C} \;=\; 0$$

where S_α is the Macaulay matrix of order α, and each functional in the basis for the dual space corresponds to a null vector of S_α and *vice versa*. The Hilbert function

$$\begin{cases} H(0)=1, \\ H(\alpha) = \mathit{nullity}(S_\alpha) - \mathit{nullity}(S_{\alpha\text{-}1}) \quad \text{for } \alpha=1,2,\ldots. \end{cases}$$

The (approximate) dual basis can be obtained by computing the (numerical) kernels of S_0, S_1, ... until reaching the α where the Hilbert function $H(\alpha)=0$ [17].

The algorithm based on the above analysis and matrix kernel computation is implemented as a computational routine MULTIPLICITYSTRUCTURE in the software package APATOOLS [99]. The algorithm is also applied to identifying the local dimension of algebraic sets [3] as part of the software package BERTINI [4].

1.3.3 Numerical elimination

Numerical elimination with approximate data arises in many applications such as kinematics [16, 21, 61, 87, 91, 90], and computational biology/chemistry [23, 25]. Numerical elimination methods have been studied in many reports, such as [1, 2, 21, 22, 42, 61, 70, 87, 90, 91]. The main strategy for the existing elimination approaches is using various resultants [32] whose computation requires calculating determinants of polynomial matrices. There are formidable obstacles for calculating resultants using floating point arithmetic since determinant computation can be inefficient and unstable. There is a new approach [95] that avoids resultant calculation and transforms the elimination to a problem of matrix rank/kernel computation.

Consider the ring $\mathbb{C}[x,y]$ of complex polynomials in variables x and y, where x is a single scalar variable and y may be either a scalar variable or a vector of variables. For polynomials $f,g\in\mathbb{C}[x,y]$, there exist polynomials p and q such that

$$f\,p+g\,q \;=\; h$$

belongs to the first elimination ideal $\langle f,g\rangle\cap\mathbb{C}[y]$ of f and g, unless (f,g) has a GCD with positive degree in x. We wish to calculate p, q and h with floating point arithmetic. Since $f\,p+g\,q=h\in\mathbb{C}[y]$, there is an obvious homogeneous equation

$$\frac{\partial}{\partial x}(f\,p+g\,q) \;=\; \left[\frac{\partial}{\partial x}f+f\cdot\frac{\partial}{\partial x}\right]p+\left[\frac{\partial}{\partial x}g+g\cdot\frac{\partial}{\partial x}\right]q \;=\; 0. \qquad (1.27)$$

which leads to a simple strategy: Finding a polynomial $f\,p+g\,q$ in the elimination ideal $\langle f,g\rangle\cap\mathbb{C}[y]$ is equivalent to computing the kernel $\mathscr{K}(L_n)$ of the linear transformation

$$L_n\,:\,(p,q)\;\longrightarrow\;\left[\frac{\partial}{\partial x}f+f\cdot\frac{\partial}{\partial x},\quad \frac{\partial}{\partial x}g+g\cdot\frac{\partial}{\partial x}\right]\begin{bmatrix}p\\ q\end{bmatrix} \qquad (1.28)$$

in the vector space $\mathbb{P}^n$ of polynomial pairs p and q with degree n or less. With proper choices of bases for the polynomial vector spaces, the linear transformation L_n induces an *elimination matrix* M_n where the rank-revealing method elaborated in §1.1.2 produces the polynomial $h=f\,p+g\,q$ in the elimination ideal. The elimination algorithm based on matrix kernel computation can be outlined below with a test version implemented in ApaTools [99] as computational routine PolynomialEliminate.

```
For  n = 1,2,⋯  do
  Update elimination matrix  M_n
  If  M_n  is rank deficient then
     extract  (p,q)  from its kernel,  break,
  end if
end do
```

The method can be generalized to eliminating more than one variables from several polynomials and, combined with numerical multivariate GCD, can be applied to solving polynomial systems for solution sets of positive dimensions [95].

1.3.4 Approximate irreducible factorization

Computing the approximate irreducible factorization of a multivariate polynomial is the first problem listed in "challenges in symbolic computation" [44] by Kaltofen. The first numerical algorithm with an implementation is developed by Sommese, Verschelde and Wampler [78, 79, 80]. Many authors studied the problem and proposed different approaches [8, 7, 10, 12, 13, 27, 28, 40, 45, 73, 74, 75, 76]. In 2003, Gao [29] proposed a hybrid algorithm and later adapted it as a numerical algorithm [30, 46, 48] along with Kaltofen, May, Yang and Zhi. Here we elaborate the strategies involved in numerical irreducible factorization from the perspective of matrix computation.

The collection of polynomials sharing the structure of irreducible factorization $p_1^{m_1}p_2^{m_2}\cdots p_k^{m_k}$ forms a factorization manifold which we denote as $\Pi^{m_1\cdots m_k}_{n_1\cdots n_k}$, where n_j is the degree of p_j for $j=1,\cdots,k$. Namely

$$\Pi^{m_1\cdots m_k}_{n_1\cdots n_k} \;=\; \left\{p_1^{m_1}p_2^{m_2}\cdots p_k^{m_k}\;\middle|\; deg(p_j)=n_j \;\text{ for}\, j=1,\ldots,k\right\}$$

For simplicity of exposition, consider the manifold $\Pi^{1,1,1}_{n_1 n_2 n_3}$ where polynomials are square-free in two variables x and y with three factors. An approximate square-free factorization algorithm is implemented in APATOOLS [99] as a computational routing SQUAREFREEFACTOR which is a recursive application of approximate GCD computation [102].

The first task of factorization is to identify the reducibility of the polynomial, the number of factors, and the degrees of each factor. The Ruppert approach[71] for reducibility detection is very effective and can be easily explained using the differential identity

$$\frac{\partial}{\partial y}\left(\frac{f_x}{f}\right) \;=\; \frac{\partial}{\partial x}\left(\frac{f_y}{f}\right)$$

for any function f. For a given polynomial $f = p\cdot q\cdot r$ of bidegree $[m,n]$, applying this identity to the first factor p yields $\frac{\partial}{\partial y}\left(\frac{p_x}{p}\cdot\frac{q\cdot r}{q\cdot r}\right) = \frac{\partial}{\partial x}\left(\frac{p_y}{p}\cdot\frac{q\cdot r}{q\cdot r}\right)$, namely

$$\frac{\partial}{\partial y}\left(\frac{p_x\cdot q\cdot r}{f}\right) \;=\; \frac{\partial}{\partial x}\left(\frac{p_y\cdot q\cdot r}{f}\right).$$

The same manipulation on the remaining factors q and r reveals that the differential equation

$$\frac{\partial}{\partial y}\left(\frac{g}{f}\right) \;=\; \frac{\partial}{\partial x}\left(\frac{h}{f}\right)$$

in the unknown polynomial pair (g,h) has three linearly independent solutions

$$(g,h) \;=\; (p_x qr,\; p_y qr),\;\; (pq_x r,\; pq_y r),\quad \text{or}\quad (pqr_x,\; pqr_y).$$

From this observation, it can be seen that the linear transformation

$$\begin{aligned}\mathscr{L}_f \;:\; (g,h) \;\longrightarrow\; & f^2\left[\frac{\partial}{\partial y}\left(\frac{g}{f}\right)-\frac{\partial}{\partial x}\left(\frac{h}{f}\right)\right] \\ =\; & [f\cdot\partial_y-\partial_y f]\,g-[f\cdot\partial_x-\partial_x f]\,h \qquad (1.29)\end{aligned}$$

has a kernel whose dimension is identical to the number of irreducible factors of f if we restrict the domain of $\mathscr{L}_f$ to those g and h of bidegrees $[m-1,n]$ and $[m,n-1]$ respectively. As a result, the reducibility and the number of irreducible factors in f can be identified by computing the nullity of the Ruppert matrix L_f corresponding to the linear transformation $\mathscr{L}_f$ with a restricted domain. When the polynomial f is inexact using floating-point arithmetic, the reducibility identification becomes numerical kernel problem.

Moreover, a vector in the numerical kernel $\mathscr{K}_\theta(L_f)$ selected at random corresponds to polynomials

$$\begin{aligned} g &= \lambda_1 p_x qr+\lambda_2 pq_x r+\lambda_3 pqr_x \\ h &= \lambda_1 p_y qr+\lambda_2 pq_y r+\lambda_3 pqr_y \end{aligned}$$

with undetermined constants λ_1, λ_2 and λ_3. From $f = pqr$ and

$$g - \lambda_1 f_x \;=\; p[(\lambda_2 - \lambda_1) q_x r + (\lambda_3 - \lambda_1) q r_x],$$

we have identities associated with the unknown λ_j's

$$\begin{cases} p &= gcd(f,\ g - \lambda_1 f_x) \\ q &= gcd(f,\ g - \lambda_2 f_x) \\ r &= gcd(f,\ g - \lambda_3 f_x) \end{cases} \tag{1.30}$$

Select complex numbers $\hat{x}$ and $\hat{y}$ at random and consider univariate polynomials $p(x,\hat{y})$ and $p(\hat{x},y)$ in (1.30). Applying the nullity count of the Sylvester matrix (1.22) elaborated in §1.3.1 yields that the Sylvester matrices

$$S\big(f(x,\hat{y}),\ g(x,\hat{y}) - \lambda_1 f_x(x,\hat{y})\big) \quad \text{and} \quad S\big(f(\hat{x},y),\ g(\hat{x},y) - \lambda_1 f_x(\hat{x},y)\big)$$

are of nullities identical to the degrees $deg_x(p)$ and $deg_y(p)$ respectively. The unknown value of λ_1 then becomes the generalized eigenvalue of the matrix pencils in the form of $A - \lambda B$:

$$S\big(f(x,\hat{y}),\ g(x,\hat{y})\big) - \lambda S\big(0,\ f_x(x,\hat{y})\big) \tag{1.31}$$

and

$$S\big(f(\hat{x},y),\ g(\hat{x},y)\big) - \lambda S\big(0,\ f_x(\hat{x},y)\big). \tag{1.32}$$

From the identities in (1.30), both pencils have the same eigenvalues λ_1, λ_2 and λ_3 of geometric multiplicities identical to the degrees $deg_x(p)$, $deg_x(q)$ and $deg_x(r)$ respectively for the pencil (1.31) and to the degrees $deg_y(p)$, $deg_y(q)$ and $deg_y(r)$ respectively for the pencil (1.32). As a result, finding the unknown constants λ_j's in (1.30) and degree structures of the irreducible factors p, q and r becomes generalized eigenvalue problem of matrix pencils in (1.31)–(1.32).

Computing eigenvalues with nontrivial multiplicities has been a challenge in numerical linear algebra. With new techniques developed in [103] on computing the Jordan Canonical Form along with the known eigen-structure of the pencils in (1.31)-(1.32), their generalized eigenvalues and multiplicities can be computed efficiently and accurately in floating-point arithmetic even if the polynomials are inexact.

In summary, Stage I of the numerical irreducible factorization can be accomplished with a sequence of matrix computations:

(a) Finding the numerical kernel of the Ruppert matrix by matrix rank/kernel computation; followed by
(b) solving the generalized eigenproblem of pencils in (1.31)-(1.32) to obtain the degrees of the irreducible factors and values of λ_j's; and concluded with
(c) computing approximate GCDs in (1.30) to obtain approximations to the irreducible factors.

This stage of the computation identifies the maximum codimension factorization manifold.

In Stage II, the approximations of the factors obtained in Stage I are used as the initial iterate for the Gauss-Newton iteration (1.11) applied on the overdetermined system $F(\mathbf{u},\mathbf{v},\mathbf{w})=\mathbf{0}$ where

$$F(\mathbf{u},\mathbf{v},\mathbf{w}) \equiv \begin{bmatrix} \mathbf{a}^{\mathsf{H}}\mathbf{u}-1 \\ \mathbf{b}^{\mathsf{H}}\mathbf{v}-1 \\ [\![uvw-f]\!] \end{bmatrix}.$$

The Jacobian of F is injective (cf. Example 1.1.3 in §1.1.3, not without the auxiliary equations $\mathbf{a}^{\mathsf{H}}\mathbf{u}=\mathbf{b}^{\mathsf{H}}\mathbf{v}=1$) at the solution $(\mathbf{p},\mathbf{q},\mathbf{r})$, ensuring the Gauss-Newton iteration to be locally convergent.

1.4 A subspace strategy for efficient matrix computations

1.4.1 The closedness subspace for multiplicity matrices

The Macaulay matrix S_α for the homogeneous system (1.26) can be undesirably large when α increases. For example, consider the benchmark problem KSS system [17, 54] of $n\times n$

$$f_j(x_1,\cdots,x_n)=x_j^2+\sum_{\nu=1}^{n}x_\nu-2x_j-n+1, \quad \text{for } j=1,\ldots,n \tag{1.33}$$

for multiplicity computation at the zero $\hat{\mathbf{x}}=(1,\ldots,1)$. The largest Macaulay matrix required is 12012×3432 for $n=7$ to obtain the multiplicity 64, and the matrix size reaches 102960×24310 for $n=8$ for identifying the multiplicity 128, exceeding the memory capacity of today's desktop personal computers. Therefore, reducing the size of the multiplicity matrices is of paramount importance.

The key idea, which is originated by Stetter and Thallinger [84, 88], is to employ the closedness condition that can be rephrased in the following lemma.

Lemma 1.4.1. [84, Theorem 8.36] *Let $I\in\mathbb{C}[x_1,\ldots,x_s]$ be a polynomial ideal and let $\mathscr{D}_{\hat{\mathbf{x}}}(I)$ be its dual space at an isolated zero $\hat{\mathbf{x}}$. Then every functional $c\in\mathscr{D}_{\hat{\mathbf{x}}}(I)$ satisfies $s_\sigma(c)\in\mathscr{D}_{\hat{\mathbf{x}}}(I)$ for all $\sigma\in\{1,\ldots,s\}$ where s_σ is the linear anti-differentiation operator defined by*

$$s_\sigma(\partial_{j_1\cdots j_s}[\hat{\mathbf{x}}]) = \begin{cases} 0, & \text{if } j_\sigma=0 \\ \partial_{j'_1\cdots j'_s}[\hat{\mathbf{x}}], & \text{otherwise} \end{cases}$$

with $j'_\sigma=j_\sigma-1$ and $j'_i=j_i$ for $i\in\{1,\cdots,s\}\backslash\{\sigma\}$.

For example, the functional $\partial_{06}-\partial_{23}$ belongs to the dual space $\mathscr{D}_{\mathbf{0}}(I)$ spanned by the functionals in (1.25) implies

$$\begin{aligned} s_1(\partial_{06}-\partial_{23}) &= 0-\partial_{13} \\ s_2(\partial_{06}-\partial_{23}) &= \partial_{05}-\partial_{22} \end{aligned}$$

are both in $\mathscr{D}_{\mathbf{0}}(I)$, as shown in (1.25). This property leads to the closedness subspace strategy: After obtaining the dual subspace

$$\mathscr{D}_{\hat{\mathbf{x}}}^{\alpha\text{-}1}(I) \equiv \mathscr{D}_{\hat{\mathbf{x}}}(I) \cap span\big\{\partial_{\mathbf{j}}[\hat{\mathbf{x}}] \;\big|\; |\mathbf{j}| \le \alpha-1\big\}$$

of order $\alpha-1$, the additional basis functionals in the dual subspace $\mathscr{D}_{\hat{\mathbf{x}}}^{\alpha}(I)$ of order α can be found in the *closedness subspace*

$$\mathscr{C}_{\hat{\mathbf{x}}}^{\alpha}(I) = \left\{ \sum_{\mathbf{j}\in\mathbb{N}^s,|\mathbf{j}|\le\alpha} c_{\mathbf{j}}\partial_{\mathbf{j}}[\hat{\mathbf{x}}] \;\middle|\; s_\sigma(c)\in\mathscr{D}_{\hat{\mathbf{x}}}^{\alpha\text{-}1}(I),\ \sigma=1,\ldots,s \right\}, \tag{1.34}$$

of order α. An algorithm for computing the bases for closedness subspaces have been developed in [100] using a sequence of numerical kernel computation involving matrices of much smaller sizes. Let $\{\phi_1,\ldots,\phi_m\}$ form a basis for the closedness subspace $\mathscr{C}_{\hat{\mathbf{x}}}^{\alpha}(I)$. Then the functionals in the dual subspace $\mathscr{D}_{\hat{\mathbf{x}}}^{\alpha}(I)$ can be expressed in the form of $u_1\phi_1+\cdots+u_m\phi_m$ and can be computed by solving the homogeneous linear system

$$u_1\phi_1(f_i)+\cdots+u_m\phi_m(f_i) = 0, \quad \text{for } i=1,\cdots t \tag{1.35}$$

where $f_1,\ldots,f_t$ are generators for the ideal I. In comparison with the linear system (1.26), the system (1.35) consists of a fixed number (t) of equations. Since the solution space of (1.35) is isomorphic to the dual subspace $\mathscr{D}_{\hat{\mathbf{x}}}^{\alpha}(I)$, the number m of unknowns u_i's is bounded by

$$m \le dim\big(\mathscr{D}_{\hat{\mathbf{x}}}^{\alpha}(I)\big)+t.$$

The process of identifying the closedness subspace $\mathscr{C}_{\hat{\mathbf{x}}}^{\alpha}(I)$ and finding dual basis functionals can be illustrated in the following example.

Example 1.4.2. Use the dual basis in (1.25) as an example. To identify the closedness subspace $\mathscr{C}_{\mathbf{0}}^{4}(I)$ after obtaining the dull subspace and the closedness subspace of order 3

$$\begin{aligned} \mathscr{D}_{\mathbf{0}}^{3}(I) &= span\{\partial_{00},\ \partial_{10},\ \partial_{01},\ \partial_{20},\ \partial_{11},\ \partial_{02},\ \partial_{12},\ \partial_{03}\} \\ \mathscr{C}_{\mathbf{0}}^{3}(I) &= span\{\partial_{00},\ \partial_{10},\ \partial_{01},\ \partial_{20},\ \partial_{11},\ \partial_{02},\ \partial_{30},\ \partial_{21},\ \partial_{12},\ \partial_{03}\}. \end{aligned}$$

The monomial functionals ∂_{22}, ∂_{31}, and ∂_{40} can be excluded since $s_2(\partial_{22}) = s_1(\partial_{31}) = \partial_{21}$ and $s_1(\partial_{40}) = \partial_{30}$ are not in the monomial support of $\mathscr{D}_{\mathbf{0}}^{3}(I)$. As a result, we have

$$\mathscr{C}_{\mathbf{0}}^{4}(I) \subset span\{\partial_{00},\ \partial_{10},\ \partial_{01},\ \partial_{20},\ \partial_{11},\ \partial_{02},\ \partial_{30},\ \partial_{21},\ \partial_{12},\ \partial_{03},\ \partial_{04},\ \partial_{13}\}$$

and every functional in $\mathscr{C}_{\mathbf{0}}^{4}(I)$ can be written as

$$\phi \;=\; \gamma_1\partial_{00}+\gamma_2\partial_{10}+\gamma_3\partial_{01}+\cdots+\gamma_{10}\partial_{04}+\gamma_{11}\partial_{13}.$$

The closedness conditions $s_1(\phi), s_2(\phi) \in \mathscr{D}^3_{\mathbf{0}}(I)$ become

$$\begin{cases} \gamma_1\partial_{00}+\gamma_3\partial_{10}+\gamma_4\partial_{01}+\gamma_6\partial_{20}+\gamma_7\partial_{11}+\gamma_8\partial_{02}+\gamma_{11}\partial_{03} \\ \qquad\qquad = \;\eta_1\partial_{00}+\eta_2\partial_{10}+\cdots+\eta_8\partial_{03} \\ \gamma_2\partial_{00}+\gamma_4\partial_{10}+\gamma_5\partial_{01}+\gamma_7\partial_{20}+\gamma_8\partial_{11}+\gamma_9\partial_{02}+\gamma_{10}\partial_{03}+\gamma_{11}\partial_{12} \\ \qquad\qquad = \;\eta_9\partial_{00}+\eta_{10}\partial_{10}+\cdots+\eta_{16}\partial_{03} \end{cases}.$$

These equations lead to

$$[\gamma_1,\ \gamma_2,\ \ldots,\gamma_{11}]^\top = [\eta_1,\ \eta_9,\ \eta_2,\ \eta_3,\ \eta_{11},\ \eta_4,\ \eta_5,\ \eta_6,\ \eta_{14},\ \eta_{16},\ \eta_8]^\top \qquad (1.36)$$

along with

$$\gamma_4=\eta_3=\eta_{10},\ \ \gamma_7=\eta_5=\eta_{12},\ \ \gamma_8=\eta_6=\eta_{13},\ \ \gamma_{11}=\eta_8=\eta_{15},\ \ \eta_7=0.$$

Consequently, we have a system of homogeneous equations

$$\eta_3-\eta_{10} \;=\; \eta_5-\eta_{12} \;=\; \eta_6-\eta_{13} \;=\; \eta_8-\eta_{15} \;=\; \eta_7 \;=\; 0. \qquad (1.37)$$

Since a basis for $\mathscr{C}^3_{\mathbf{0}}(I)$ is already obtained, we need only additional basis functionals in $\mathscr{C}^4_{\mathbf{0}}(I)$ by requiring

$$\phi \perp \mathscr{C}^3_{\mathbf{0}}(I). \qquad (1.38)$$

In general, systems of equations in (1.36), (1.37) and (1.38) can be written in matrix forms

$$\begin{bmatrix}\gamma_1\\ \vdots\\ \gamma_m\end{bmatrix} \;=\; A\begin{bmatrix}\eta_1\\ \vdots\\ \eta_n\end{bmatrix},\quad B\begin{bmatrix}\eta_1\\ \vdots\\ \eta_n\end{bmatrix} \;=\; \mathbf{0},\quad \text{and}\quad C\begin{bmatrix}\eta_1\\ \vdots\\ \eta_n\end{bmatrix} \;=\; \mathbf{0}$$

respectively. Thus the problem of identifying the closedness subspace becomes the (numerical) kernel problem of the matrix $\begin{bmatrix}B\\ C\end{bmatrix}$, from which we obtain

$$\begin{bmatrix}\eta_1\\ \vdots\\ \eta_n\end{bmatrix} \;=\; N\begin{bmatrix}\mu_1\\ \vdots\\ \mu_k\end{bmatrix},\quad \text{and thus}\quad \begin{bmatrix}\gamma_1\\ \vdots\\ \gamma_m\end{bmatrix} \;=\; AN\begin{bmatrix}\mu_1\\ \vdots\\ \mu_k\end{bmatrix}.$$

In this example, $\gamma_1=\cdots=\gamma_9=0$ and

$$\begin{bmatrix}\gamma_{10}\\ \gamma_{11}\end{bmatrix} \;=\; \begin{bmatrix}0&1\\ 1&0\end{bmatrix}\begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}$$

implying

$$\mathscr{C}^4_{\mathbf{0}}(I) \;=\; \mathscr{C}^3_{\mathbf{0}}(I)\oplus span\{\partial_{13},\ \partial_{04}\}$$

where additional dual basis functionals ∂_{13} and $\partial_{04}-\partial_{21}$ in $\mathscr{D}^4_{\mathbf{0}}(I)$ are solved in the equation (1.35). □

Preliminary experiments show that the closedness subspace strategy tremendously improves the computational efficiency over its predecessor implemented in `ApaTools` as the module `MultiplicityStructure` in both memory and computing time. For instance, `MultiplicityStructure` can handle only $n \le 6$ for the KSS system (1.33) with multiplicity 42 before running out of memory. The new code increase the capacity to $n = 9$ with multiplicity 256. Moreover, the new code is as much as 76 times faster on the KSS system at $n = 6$. The speed up reaches thousands on some large systems with low multiplicities [100].

1.4.2 The fewnomial subspace strategy for multivariate polynomials

A vector space of multivariate polynomials within a degree bound may have huge dimensions. For example, we can construct a simple GCD test problem with

$$\begin{cases} p &= (x_1+\cdots+x_n-1)\big(x_1(x_2^n+\cdots+x_n^n)+2\big), \\ q &= (x_1+\cdots+x_n-1)\big(x_n(x_1^n+\cdots+x_{n\text{-}1}^n)-2\big) \end{cases} \tag{1.39}$$

in expanded form. For $n = 10$, the dimension of the vector space of polynomial pairs within total degree $n+2$ is 1,293,292, while both p and q are "fewnomials" of only 110 terms. To compute the approximate GCD of (p,q), the standard Sylvester matrix has a size as large as $92,561,040 \times 705,432$ requiring nearly 500 terabytes of memory. It is not practical to construct such a large matrix on a desktop computer.

To compute the approximate GCD of such polynomial pairs, we employ a simple *fewnomial subspace strategy* for reducing the sizes of matrices required for the computation by identifying the monomial support of the GCD and the cofactors. The strategy is similar to the sparse interpolation approach in [15, 33, 105] that is applied to other polynomial problems [30, 48, 51].

We shall use the polynomial pair in (1.39) as an example for $n = 3$ to illustrate the fewnomial subspace strategy.

Example 1.4.3. Consider the polynomial pair

$$\begin{aligned} p(x_1,x_2,x_3) &= (x_1+x_2+x_3-1)(x_1x_2^3+x_1x_3^3+2) \\ q(x_1,x_2,x_3) &= (x_1+x_2+x_3-1)(x_1^3x_3+x_2^3x_3-2) \end{aligned}$$

with $u = \gcd(p,q)$ and cofactors v and w. Assign fixed random unit complex values $\hat{x}_2 = .8126623341 - .5827348718\,i$ and $\hat{x}_3 = .8826218380 + .4700837065\,i$ in (p,q) and compute its univariate approximate GCD in x_1, obtaining

$$p(x_1,\hat{x}_2,\hat{x}_3) = (x_1+.6953-.1127i)(-(0.1887-.0381i)x_1+2)$$
$$q(x_1,\hat{x}_2,\hat{x}_3) = (x_1+.6953-.1127i)((.8826+.4701i)x_1^3-1.807-.9813i)$$

*(*showing 4 digits for each number*)*, implying

$$u(x_1,\hat{x}_2,\hat{x}_3) \in span\{1,\ x_1\}, \tag{1.40}$$
$$v(x_1,\hat{x}_2,\hat{x}_3) \in span\{1,\ x_1\}, \quad w(x_1,\hat{x}_2,\hat{x}_3) \in span\{1,\ x_1^3\}. \tag{1.41}$$

Similarly, we can apply univariate GCD in x_2 and

$$u(\hat{x}_1,x_2,\hat{x}_3) \in span\{1,x_2\}, \tag{1.42}$$
$$v(\hat{x}_1,x_2,\hat{x}_3) \in span\{1,x_2^3\},\ w(\hat{x}_1,x_2,\hat{x}_3) \in span\{1,x_2^3\} \tag{1.43}$$
$$u(\hat{x}_1,\hat{x}_2,x_3) \in span\{1,x_3\}, \tag{1.44}$$
$$v(\hat{x}_1,\hat{x}_2,x_3) \in span\{1,x_3^3\},\ w(\hat{x}_1,\hat{x}_2,x_3) \in span\{1,x_3\} \tag{1.45}$$

Consequently, combining the monomial bases in (1.40)-(1.43) yields three fewnomial subspaces

$$u(x_1,x_2,\hat{x}_3) \in span\{1,\ x_1,\ x_2,\ x_1x_2\},$$
$$v(x_1,x_2,\hat{x}_3) \in span\{1,\ x_1,\ x_2^3,\ x_1x_2^3\},$$
$$w(x_1,x_2,\hat{x}_3) \in span\{1,\ x_1^3,\ x_2^3,\ x_1^3x_2^3\}.$$

In these subspaces we compute bivariate GCD of (p,q) in x_1 and x_2:

$$p(x_1,x_2,\hat{x}_3) = (x_1+x_2-.1174+.4701i)(x_1x_2^3+(.1025+.9947i)x_1+2)$$
$$q(x_1,x_2,\hat{x}_3) = (x_1+x_2-.1174+.4701i)((.8826+.4701i)x_1^3+(.8826+.4701i)x_2^3-2).$$

Combining monomial bases in (1.44) and (1.45) yields the fewnomial subspaces

$$u \in span\{1,\ x_1,\ x_2,\ x_3,\ x_1x_3,\ x_2x_3\},$$
$$v \in span\{1,\ x_1,\ x_1x_2^3,\ x_3^3,\ x_1x_3^3,\ x_1x_2^3x_3^3\},$$
$$w \in span\{1,\ x_1^3,\ x_2^3,\ x_3,\ x_1^3x_3,\ x_2^3x_3\}.$$

where u, v and w are computed as approximations to

$$u = x_1+x_2+x_3-1, \quad v = x_1x_2^3+x_1x_3^2+2, \quad w = x_1^3x_3+x_2^3x_3-2$$

□

The process in Example 1.4.3 can be extended to general cases of computing the approximate GCD of multivariate polynomials in $\mathbb{C}[x_1,\cdots,x_s]$: For $k=1,2,\ldots,s$, compute the approximate GCD of

$$p(x_1,\ldots,x_k,\hat{x}_{k+1},\ldots,\hat{x}_s)$$
$$q(x_1,\ldots,x_k,\hat{x}_{k+1},\ldots,\hat{x}_s)$$

in $x_1,\ldots,x_k$ with remaining variables fixed using random unit complex constants $\hat{x}_{k+1},\ldots,\hat{x}_s$. Then we can identify the monomial subspaces for u, v and w in the first $k+1$ variables, and on those subspaces we can compute the approximate GCD of (p,q) in the first $k+1$ variables. Continuing this process we complete the GCD computation at $k=s$. This simple technique is tremendously effective in practical computations for sparse polynomials. For the case, say $n=10$ in (1.39), the largest Sylvester matrix on the fewnomial subspaces has only 40 columns, which is quite a reduction from 705,432.

1.5 Software development

Numerical polynomial algebra is a growing field of study with many algorithms that are still in early stages of development. Nonetheless, many software packages are already available. Most advanced software packages are perhaps the polynomial system solvers based on the homotopy continuation method [57, 82], including PHCPACK [35, 89], BERTINI [4, 5] and MIXEDVOL [31]. There are also specialized implementations of algorithms such as approximate GCD finder QRGCD [14] that is bundled in recent Maple releases, approximate factorization [48], multiplicity structure [6, 88], SNAP package [41] bundled in Maple, univariate factorization and multiple root solver MULTROOT [97], SYNAPS package [66], and COCOA [9, 55, 56], etc. Those packages provide a broad range of versatile tools for applications and academic research.

A comprehensive software toolbox APATOOLS is also in development for approximate polynomial algebra with a preliminary release [99]. APATOOLS is built on two commonly used platforms: Maple and Matlab. The Matlab version is named APALAB. There are two objectives for designing APATOOLS: Assembling robust computational routines as finished product for applications as well as providing a utility library as building blocks for developing more advanced algorithms in numerical polynomial algebra. Currently, APATOOLS includes the following computational routines:

UVGCD: univariate approximate GCD finder (§1.3.1)

MVGCD: multivariate approximate GCD finder (§1.3.1)

UVFACTOR: univariate approximate factorization with multiplicities (§1.2.2)

SQUAREFREEFACTOR: multivariate squarefree factorization (§1.3.4)

MULTIPLICITYSTRUCTURE: dual basis and multiplicity identification (§1.3.2)

POLYNOMIALELIMINATE: numerical and symbolic elimination routine (§1.3.3)

APPROXIRANK: numerical rank/kernel routine (§1.1.2)

NUMJCF: (APALAB only) function for computing the approximate Jordan Canonical Form of inexact matrices (§1.2.2)

Those routines implement algorithms that solve ill-posed algebraic problems for approximate solutions formulated based on the three-strikes principle elaborated in §1.2.2 via a two-staged process: Finding the maximum codimension pejorative manifold by matrix computation, followed by minimizing the distance to the manifold via the Gauss-Newton iteration.

As a growing field of study, numerical polynomial algebra algorithms are in continuing development. APATOOLS consists of a comprehensive collection of utility routines designed to simplify software implementation for algorithms. Those utilities include matrix computation tools such as orthogonal transformations and other manipulations, index utilities, monomial utilities, and so on. The two most notable utilities are matrix builder LINEARTRANSFORMMATRIX that conveniently generate matrices from linear transformations between polynomial vector spaces, and the nonlinear least squares solver GAUSSNEWTON for minimizing the distance to a pejorative manifold. Both routines are designed with a priority on simplicity for users and with options to maximize efficiency, as illustrated in the following examples.

Example 1.5.1. Construction of Ruppert matrices in ApaTools. The Ruppert matrix (cf. §1.3.4) is the matrix represents the linear transformation $\mathscr{L}_f$ in (1.29) from a given polynomial f of bidegree $[m,n]$, say

$$f = 2x^3y^3 - 5x^2y^5 + x^2y + 6x^2y^2 - 15xy^4 + 3x - 4xy^2 + 10y^4 - 2$$

of bidegree $[3,5]$. The linear transformation is a combination of the linear transformations

$$\mathscr{L}_{\{,\|} : \sqcap \longrightarrow [\{\cdot\partial_\| - \partial_\|\{]\sqcap, \quad \| = \infty, \in \tag{1.46}$$

A Maple user needs to write a straightforward three-line Maple procedure for this linear transformation:

```
> RupLT := proc( u, x, f, k)
     return expand( f*diff(u,x[k]) - diff(f,x[k])*u )
  end proc:
```

Then the Ruppert matrix is constructed instantly by calling

```
> R := < LinearTransformMatrix(RupLT,[f,2],[x,y],
                          [m-1,n],[2*m-1,2*n-1])  |
         LinearTransformMatrix(RupLT,[f,1],[x,y],
                          [m,n-1],[2*m-1,2*n-1])  >;
```

$$R := \left[\begin{array}{l} \text{60 x 38 Matrix} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran\_order} \end{array}\right]$$

Here, the input items `[f,1]` and `[f,2]` indicate the linear transformation $\mathscr{L}_{\{,\infty}$ and $\mathscr{L}_{\{,\in}$ respectively, `[x,y]` is the list of indeterminates, `[m-1,n]` and `[m,n-1]` are the bidegree bounds on the domains of the linear transformations, and `[2*m-1,2*n-1]` is the bidegree bound of the range. Applying the numerical kernel routine APPROXIRANK

```
> r, s, N := ApproxiRank(R,[1e-8]);
```

yields the numerical nullity 2 that reveals the number of irreducible factors of f.
□

Example 1.5.2. Applying the Gauss-Newton iteration in factorization. After obtaining an initial approximation

$$\begin{aligned} u_1 &= -1.99999 + 2.99999x + x^2y \\ u_2 &= .99998 + 2xy^2 - 4.99998y^4 \end{aligned}$$

to the irreducible factors of the polynomial f in Example 1.5.1, we seek the least squares solution to the overdetermined system of equations

$$\begin{cases} \phi^{\mathrm{H}} u_1 - 1 &= 0 \\ [\![u_1 \cdot u_2 - f]\!] &= \mathbf{0} \end{cases}$$

(cf. Example 1.1.3 and §1.3.4). A user needs to write a straightforward Maple procedure for this system of equations

```
> FacFunc := proc( w, x, f, phi)
     return [PolynomialDotProduct( phi,w[1],x)-1,
                            expand(w[1]*w[2])-f]
     end proc;
```

prepare input data and make a call on the ApaTools routing GAUSSNEWTON

```
> factors, residual := GaussNewton(FacFunc,[u1,u2],
                    [[x,y],f,phi], [1e-12,9,true]);
```

```
    Gauss-Newton step   0,  residual =   1.71e-04
    Gauss-Newton step   1,  residual =   4.82e-10
    Gauss-Newton step   2,  residual =   3.46e-14
```

$$\begin{aligned} factors, residual := [&-2.00000714288266 + 3.00001071432397x + \\ &1.00000357144133x^2y,\ .999996428571430 + 1.99999285714286xy^2 - \\ &4.99998214285715y^4], .346410161513776\,10^{-13} \end{aligned}$$

The result shows computed irreducible factors with a backward error 3.5×10^{-14}, and the factors with a scaling

$$-2.00000000000000 + 2.99999999999998x + 1.00000000000000x^2y$$

and

$$1.00000000000000 + 2.00000000000001xy^2 - 5.00000000000002y^4$$

are accurate near hardware precision. □

APATOOLS is an on-going project with its functionality and library still expanding. We wish to build a comprehensive software toolbox for applications and algorithm development. The near term plan is to incorporate approximate irreducible factorization, fully implement the closedness/fewnomial subspace strategy for enhancing the efficiency, and collect a library of benchmark test problems for numerical polynomial algebra.

Acknowledgements. The author is supported in part by the National Science Foundation of U.S. under Grants DMS-0412003 and DMS-0715127. The author thanks Hans Stetter for providing his student's thesis [88], and Erich Kaltofen for his comment as well as several references in §1.4.2.

References

1. E. L. ALLGOWER, K. GEORG, AND R. MIRANDA, *The method of resultants for computing real solutions of polynomial systems*, SIAM J. Numer. Anal., 29 (1992), pp. 831–844.
2. W. AUZINGER AND H. J. STETTER, *An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations*. Proc. of International Conference on Numerical Mathematics Singapore, 1988.
3. D. J. BATES AND J. D. HAUENSTEIN AND C. PETERSON AND A. J. SOMMESE, *A local dimension test for numerically approximate points on algebraic sets*, Preprint, 2008.
4. D. BATES, J. D. HAUENSTERN, A. J. SOMMESE, AND C. W. WAMPLER, *Bertini: Software for Numerical Algebraic Geometry*. http://www.nd.edu/∼sommese/bertini, 2006.
5. ———, *Software for numerical algebraic geometry: A paradigm and progress towards its implementation*, in Software for Algebraic Geometry, IMA Volume 148, M. Stillman, N. Takayama, and J. Verschelde, eds., Springer, 2008, pp. 1–14.
6. D. J. BATES, C. PETERSON, AND A. J. SOMMESE, *A numerical-symbolic algorithm for computing the multiplicity of a component of an algebraic set*, J. of Complexity, 22 (2006), pp. 475–489.
7. G. CHÈZE AND A. GALLIGO, *Four lessons on polynomial absolute factorization*, in Solving Polynomial Equations: Foundations, Algorithms, and Applications, A. Dickenstein and I. Emiris, eds., vol. 14 of Algorithms and Computation in Mathematics, Springer-Verlag, 2005, pp. 339–392.
8. ———, *From an approximate to an exact absolute polynomial factorization*, J. of Symbolic Computation, 41 (2006), pp. 862–696.
9. THE COCOA TEAM, CoCoA*: a system for doi ng Computations in Commutative Algebra*. Available at `http://cocoa.dima.unige.it`.
10. R. M. CORLESS, A. GALLIGO, I. KOTSIREAS, AND S. WATT, *A geometric-numeric algorithm for factoring multivariate polynomials*. Proc. ISSAC'02, ACM Press, pages 37-45, 2002.
11. R. M. CORLESS, P. M. GIANNI, B. M. TRAGER, AND S. M. WATT, *The singular value decomposition for polynomial systems*. Proc. ISSAC '95, ACM Press, pp 195-207, 1995.
12. R. M. CORLESS, M. GIESBRECHT, D. JEFFREY, AND S. WATT, *Approximate polynomial decomposition*. Proc. ISSAC'99, ACM Press, pages 213-220, 1999.
13. R. M. CORLESS, M. GIESBRECHT, M. VAN HOEIJ, I. KOTSIREAS, AND S. WATT, *Towards factoring bivariate approximate polynomials*. Proc. ISSAC'01, ACM Press, pages 85-92, 2001.
14. R. M. CORLESS, S. M. WATT, AND L. ZHI, *QR factoring to compute the GCD of univariate approximate polynomials*, IEEE Trans. Signal Processing, 52 (2003), pp. 3394–3402.
15. A. CUYT AND W.-S. LEE, *A new algorithm for sparse interpolation of multivariate polynomials*, Theoretical Computer Science, Vol. 409, pp 180-185, 2008

16. D. DANEY, I. Z. EMIRIS, Y. PAPEGAY, E. TSIGARIDAS, AND J.-P. MERLET, *Calibration of parallel robots: on the elimination of pose-dependent parameters*, in Proc. of the first European Conference on Mechanism Science (EuCoMeS), 2006.
17. B. DAYTON AND Z. ZENG, *Computing the multiplicity structure in solving polynomial systems*. Proceedings of ISSAC '05, ACM Press, pp 116–123, 2005.
18. J.-P. DEDIEU AND M. SHUB, *Newton's method for over-determined system of equations*, Math. Comp., 69 (1999), pp. 1099–1115.
19. J. W. DEMMEL, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
20. J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall Series in Computational Mathematics, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
21. I. Z. EMIRIS, *Sparse elimination and application in kinematics*, PhD thesis, Computer Science Division, Dept. of Elec. Eng. and Comput. Sci., Univ. of California, Berkeley, 1994.
22. ———, *A general solver based on sparse resultants*, in Proc. PoSSo (Polynomial System Solving) Workshop on Software, 1995, pp. 35–54.
23. I. Z. EMIRIS, E. D. FRITZILAS, AND D. MANOCHA, *Algebraic algorithms for determining structure in biological chemistry*, Internatinal J. Quantum Chemistry, 106 (2006), pp. 190–210.
24. I. Z. EMIRIS, A. GALLIGO, AND H. LOMBARDI, *Certified approximate univariate GCDs*, J. Pure Appl. Algebra, 117/118 (1997), pp. 229–251.
25. I. Z. EMIRIS AND B. MOURRAIN, *Computer algebra methods for studying and computing molecular conformations*, Algorithmica, 25 (1999), pp. 372–402.
26. W. FULTON, *Intersection Theory*, Springer Verlag, Berlin, 1984.
27. A. GALLIGO AND D. RUPPRECHT, *Semi-numerical determination of irreducible branches of a reduced space curve*. Proc. ISSAC'01, ACM Press, pages 137-142, 2001.
28. A. GALLIGO AND S. WATT, *A numerical absolute primality test for bivariate polynomials*. Proc. ISSAC'97, ACM Press, pages 217–224, 1997.
29. S. GAO, *Factoring multivariate polynomials via partial differential equations*, Math. Comp., 72 (2003), pp. 801–822.
30. S. GAO, E. KALTOFEN, J. MAY, Z. YANG, AND L. ZHI, *Approximate factorization of multivariate polynomials via differential equations*. Proc. ISSAC '04, ACM Press, pp 167-174, 2004.
31. T. GAO AND T.-Y. LI, *MixedVol: A software package for mixed volume computation*, ACM Trans. Math. Software, 31 (2005), pp. 555–560.
32. I. GELFAND, M. KAPRANOV, AND A. ZELEVINSKY, *Discriminants, Resultants and Multidimensional determinants*, Birkhäuser, Boston, 1994.
33. M. GIESBRECHT, G. LABAHN AND W-S. LEE, *Symbolic-numeric sparse interpolation of multivariate polynomials*, Journal of Symbolic Computation, to appear, 2009.
34. G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The John Hopkins University Press, Baltimore and London, 3rd ed., 1996.
35. Y. GUAN AND J. VERSCHELDE, *PHClab: A MATLAB/Octave interface to PHCpack*, in Software for Algebraic Geometry, IMA Volume 148, M. Stillman, N. Takayama, and J. Verschelde, eds., Springer, 2008, pp. 15–32.
36. J. HADAMARD, *Sur les problèmes aux dèrivèes partielles et leur signification physique*. Princeton University Bulletin, 49–52, 1902.
37. P. C. HANSEN, *Rank-Deficient and Discrete Ill-Posed Problems*, SIAM, 1997.
38. M. HITZ, E. KALTOFEN, AND Y. N. LAKSHMAN, *Efficient algorithms for computing the nearest polynomial with a real root and related problems*. Proc. ISSAC'99, ACM Press, pp 205-212, 1999.
39. V. HRIBERNIG AND H. J. STETTER, *Detection and validation of clusters of polynomial zeros*, J. Symb. Comput., 24 (1997), pp. 667–681.
40. Y. HUANG, W. WU, H. STETTER, AND L. ZHI, *Pseudofactors of multivariate polynomials*. Proc. ISSAC '00, ACM Press, pp 161-168, 2000.

41. C.-P. JEANNEROD AND G. LABAHN, *The SNAP package for arithemetic with numeric polynomials.* In International Congress of Mathematical Software, World Scientific, pages 61-71, 2002.
42. G. F. JÓNSSON AND S. A. VAVASIS, *Accurate solution of polynomial equations using Macaulay resultant matrices*, Math. Comp., 74 (2004), pp. 221–262.
43. W. KAHAN, *Conserving confluence curbs ill-condition*. Technical Report 6, Computer Science, University of California, Berkeley, 1972.
44. E. KALTOFEN, *Challenges of symbolic computation: my favorite open problems*, J. Symb. Comput., 29 (2000), pp. 161–168.
45. E. KALTOFEN, B. LI, Z. YANG, AND L. ZHI, *Exact certification of global optimality of approximate factorization via rationalizing sums-of-squares with floating point scalars*, Proc. ISSAC '08, ACM Press, pp 155-163, 2008.
46. E. KALTOFEN AND J. MAY, *On approximate irreducibility of polynomials in several variables.* Proc. ISSAC '03, ACM Press, pp 161-168, 2003.
47. E. KALTOFEN, J. MAY, Z. YANG, AND L. ZHI, *Structured low rank approximation of Sylvester matrix*, in Symbolic-Numeric Computation, Trends in Mathematics, D. Wang and L. Zhi, editors, Birkhäuser Verlag, Basel, Switzerland, (2007), pp. 69–83.
48. ———, *Approximate factorization of multivariate polynomials using singular value decomposition*, J. of Symbolic Computation, 43 (2008), pp. 359–376.
49. E. KALTOFEN, Z. YANG, AND L. ZHI, *Structured low rank approximation of a Sylvester matrix*, Symbolic-Numeric Computation, D. Wang and L. Zhi, Eds, Trend in Mathematics, Birkhäuser Verlag Basel/Switzerland, pp. 69-83, 2006
50. ———, *Approximate greatest common divisor of several polynomials with linearly constrained coefficients and singular polynomials*. Proc. ISSAC'06, ACM Press, pp 169–176, 2006.
51. ———, *On probabilistic analysis of randomization in hybrid symbolic-numeric algorithms*, SNC'07 Proc. 2007 Internat. Workshop on Symbolic-Numeric Comput. pp. 11-17, 2007.
52. N. K. KARMARKAR AND Y. N. LAKSHMAN, *Approximate polynomial greatest common divisors and nearest singular polynomials*. Proc. ISSAC'96, pp 35-42, ACM Press, 1996.
53. ———, *On approximate polynomial greatest common divisors*, J. Symb. Comput., 26 (1998), pp. 653–666.
54. H. KOBAYASHI, H. SUZUKI, AND Y. SAKAI, *Numerical calculation of the multiplicity of a solution to algebraic equations*, Math. Comp., 67 (1998), pp. 257–270.
55. M. KREUZER AND L. ROBBIANO, *Computational Commutative Algebra 1*, Springer Verlag, Heidelberg, 2000.
56. ———, *Computational Commutative Algebra 2*, Springer Verlag, Heidelberg, 2000.
57. T.-Y. LI, *Solving polynomial systems by the homotopy continuation method*, Handbook of Numerical Analysis, XI, edited by P. G. Ciarlet, North-Holland, Amsterdam (2003), pp. 209–304.
58. T.-Y. LI. Private communication, 2006.
59. T. Y. LI AND Z. ZENG, *A rank-revealing method with updating, downdating and applications*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 918–946.
60. F. S. MACAULAY, *The Algebraic Theory of Modular Systems*, Cambridge Univ. Press, London, 1916.
61. D. MANOCHA AND S. KRISHNAN, *Solving algebraic systems using matrix computations*, Communication in Computer Algebra, 30 (1996), pp. 4–21.
62. M. G. MARINARI, T. MORA, AND H. M. MÖLLER, *On multiplicities in polynomial system solving*, Trans. AMS, 348 (1996), pp. 3283–3321.
63. ÅKE BJÖRCK, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.
64. T. MORA, *Solving Polyonmial Equation Systems II*, Cambridge Univ. Press, London, 2004.
65. B. MOURRAIN, *Isolated points, duality and residues*, J. of Pure and Applied Algebra, 117 & 118 (1996), pp. 469–493. Special issue for the Proc. of the 4th Int. Symp. on Effective Methods in Algebraic Geometry (MEGA).

66. B. MOURRAIN AND J.-P. PAVONE, *SYNAPS, a library for dedicated applications in symbolic numeric computing*, in Software for Algebraic Geometry, IMA Volume 148, M. Stillman, N. Takayama, and J. Verschelde, eds., Springer, 2008, pp. 81–100.
67. M.-T. NADA AND T. SASAKI, *Approximate GCD and its application to ill-conditioned algebraic equations*, J. Comput. Appl. Math., 38 (1991), pp. 335–351.
68. V. Y. PAN, *Numerical computation of a polynomial GCD and extensions*, Information and Computation, 167 (2001), pp. 71–85.
69. S. PILLAI AND B. LIANG, *Blind image deconvolution using GCD approach*, IEEE Trans. Image Processing, 8 (1999), pp. 202–219.
70. G. REID AND L. ZHI, *Solving nonlinear polynomial systems*. Proc. international conference on polynomial system solving, pp. 50-53, 2004.
71. W. RUPPERT, *Reducibility of polynomials* $f(x,y)$ *modulo* p, J. Number Theory, 77 (1999), pp. 62–70.
72. D. RUPPRECHT, *An algorithm for computing certified approximate GCD of n univariate polynomials*, J. Pure and Appl. Alg., 139 (1999), pp. 255–284.
73. T. SASAKI, *Approximate multivariate polynomial factorization based on zero-sum relations*. Proc. ISSAC'01, ACM Press, pp 284-291, 2001.
74. T. SASAKI, T. SAITO, AND T. HILANO, *Analysis of approximate factorization algorithm I*, Japan J. Industrial and Applied Math, 9 (1992), pp. 351–368.
75. ———, *A unified method for multivariate polynomial factorization*, Japan J. Industrial and Applied Math, 10 (1993), pp. 21–39.
76. T. SASAKI, M. SUZUKI, M. KOLAR, AND M. SASAKI, *Approximate factorization of multivariate polynomials and absolute irreducibility testing*, Japan J. Industrial and Applied Math, 8 (1991), pp. 357–375.
77. A. SCHÖNHAGE, *Quasi-GCD computations*, J. Complexity, 1 (1985), pp. 118–137.
78. A. J. SOMMESE, J. VERSCHELDE, AND C. W. WAMPLER, *Numerical irreducible decomposition using PHCpack*. In *Algebra, Geometry and Software Systems*, edited by M. Joswig et al, Springer-Verlag 2003, 109-130.
79. ———, *Numerical irreducible decomposition using projections from points on the components*. In J. Symbolic Computation: Solving Equations in Algebra, Geometry and Engineering, volumn 286 of Comtemporary Mathematics, edited by E.L. Green et al, pages 37-51, AMS 2001.
80. ———, *Numerical factorization of multivariate complex polynomials*, Theoretical Computer Science, 315 (2003), pp. 651–669.
81. ———, *Introduction to numerical algebraic geometry*, in Solving Polynomial Equations, A. Dickenstein and I. Z. Emiris, eds., Springer-Verlag Berlin Heidelberg, 2005, pp. 301–337.
82. A. J. SOMMESE AND C. W. WAMPLER, *The Numerical Solution of Systems of Polynomials*, World Scientific Pub., Hackensack, NJ, 2005.
83. H. J. STETTER, *Matrix eigenproblems are at the heart of polynomial system solving*, ACM SIGSAM Bulletin, 30 (1996), pp. 22–25.
84. ———, *Numerical Polynomial Algebra*, SIAM, 2004.
85. G. W. STEWART, *Matrix Algorithms, Volume I, Basic Decompositions*, SIAM, Philadelphia, 1998.
86. M. STILLMAN, N. TAKAYAMA, AND J. VERSCHELDE, eds., *Software for Algebraic Geometry*, vol. 148 of The IMA Volumes in Mathematics and its Applications, Springer, 2008.
87. H.-J. SU, C. W. WAMPLER, AND J. MCCARTHY, *Geometric design of cylindric PRS serial chains*, ASME J. Mech. Design, 126 (2004), pp. 269–277.
88. G. H. THALLINGER, *Analysis of Zero Clusters in Multivariate Polynomial Systems*. Diploma Thesis, Tech. Univ. Vienna, 1996.
89. J. VERSCHELDE, *Algorithm 795: PHCpack: A general-purpose solver for polynomial systems by homotopy continuation*, ACM Trans. Math. Software, (1999), pp. 251–276.
90. C. W. WAMPLER, *Displacement analysis of spherical mechanisms having three or fewer loops*, ASME J. Mech. Design, 126 (2004), pp. 93–100.
91. ———, *Solving the kinematics of planar mechanisms by Dixon determinant and a complex-plane formulation*, ASME J. Mech. Design, 123 (2004), pp. 382–387.

92. P.-Å. WEDIN, *Perturbation bounds in connection with singular value decomposition*, BIT, 12 (1972), pp. 99–111.
93. J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.
94. X. WU AND L. ZHI, *Computing the multiplicity structure from geometric involutive form*. Proc. ISSAC'08, ACM Press, pages 325–332, 2008.
95. Z. ZENG, *A polynomial elimination method for numerical computation*. Theoretical Computer Science, Vol. 409, pp. 318-331, 2008.
96. Z. ZENG, *The approximate GCD of inexact polynomials. Part I.* to appear.
97. ———, *Algorithm 835: MultRoot – a Matlab package for computing polynomial roots and multiplicities*, ACM Trans. Math. Software, 30 (2004), pp. 218–235.
98. ———, *Computing multiple roots of inexact polynomials*, Math. Comp., 74 (2005), pp. 869–903.
99. Z. ZENG, *ApaTools: A Maple and Matlab toolbox for approximate polynomial algebra*, in Software for Algebraic Geometry, IMA Volume 148, M. Stillman, N. Takayama, and J. Verschelde, eds., Springer, 2008, pp. 149–167.
100. Z. ZENG, *The closedness subspace method for computing the multiplicity structure of a polynomial system*, to appear: Contemporary Mathematics series, American Mathematical Society, 2009.
101. Z. ZENG, *The approximate irreducible factorization of a univariate polynomial. Revisited*, preprint, 2009.
102. Z. ZENG AND B. DAYTON, *The approximate GCD of inexact polynomials. II: A multivariate algorithm*. Proceedings of ISSAC'04, ACM Press, pp. 320-327. (2006).
103. Z. ZENG AND T. Y. LI, *A numerical method for computing the Jordan Canonical Form*. Preprint, 2007.
104. L. ZHI, *Displacement structure in computing approximate GCD of univariate polynomials*. In Proc. Sixth Asian Symposium on Computer Mathematics (ASCM 2003), Z. Li and W. Sit, Eds, vol. 10 of Lecture Notes Series on Computing World Scientific, pp. 288-298, 2003.
105. R.E. ZIPPLE, *Probability algorithms for sparse polynomials*, Proc. EUROSAM '79, Springer Lec. Notes Comp. Sci., 72, pp. 216-226, 1979.