Symbolic-numeric algorithms for univariate polynomials

著者	TERUI Akira
内容記述	Thesis (Ph. D. in Science)--University of Tsukuba, (B), no. 2485, 2010.3.25 Includes bibliographical references Note to the re-typeset version: This is re-typeset version of the original dissertation. While I have maintained the original contents without changing any words and/or formulas in the main body, I have added the following information: 1. Copyright notice of corresponding articles in each chapter; 2. Digital Object Identifiers (DOI) or URLs of references as many as possible. Please note that the number of pages is slightly increased in the present edition from that of the original edition, possibly by changes of page style parameters.
year	2010
学位授与年度	2009
URL	http://doi.org/10.15068/00151496

Symbolic-Numeric Algorithms for Univariate Polynomials

Akira Terui

February 2010

Symbolic-Numeric Algorithms for Univariate Polynomials

Akira Terui

Submitted to the Graduate School of
Pure and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Science

at the
University of Tsukuba

ABSTRACT

In computer algebra, "symbolic-numeric computation" is attracting broad range of attentions for developing new aspect of scientific computation in recent decades. Symbolic-numeric computation includes the following two approaches such as 1) employing algebraic methods in numerical computation, and 2) executing polynomial and/or rational function computations over the floating-point arithmetic. In this dissertation, we present numerous algorithms in symbolic-numeric computation for univariate polynomials: 1) "the Durand-Kerner method for the real roots," as the former approach; and 2) " 'approximate zero-points' of real univariate polynomial with large error terms," 3) "recursive polynomial remainder sequence (PRS) and its subresultants," and 4) "GPGCD: an iterative method for calculating approximate greatest common divisor (GCD) of univariate polynomials;" as the latter approach.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

First of all, I express my sincere gratitude to Professor Tateaki Sasaki for inviting me to this exciting subject of research and for outstanding guidance for many years, including some parts in the work. I am also grateful to his collaborators in approximate algebraic computation: Professor Matu-Tarow Noda, Professor Fujio Kako, Professor Tetsuo Fukui, and Professor Hiroshi Kai, for their advices and encouragement for many years.

I would like to thank to the member of my dissertation committee for valuable comments and suggestions; Professor Ko Sakai, Professor Akito Tsuboi, Professor Makoto Aoshima, for advices in daily collaboration in Mathematics of Information Group in Department of Mathematics, University of Tsukuba; and Professor Kiyoshi Shirayanagi, for advices and encouragement in the research of approximate algebraic computation.

I am greatly indebted to the member of former computer algebra research group (so-called Sasaki Lab.) in Department of Mathematics, University of Tsukuba. My apologies for not listing everyone's name because there are too many, yet their support and discussions have been and will be invaluable for me. I would also like to express my heartfelt appreciation to Professor Hiroyuki Tachikawa for his discussions, advices and continuous encouragement in collaboration with myself and/or the lab members.

Many thanks go to the faculty of Department of Mathematics, Graduate School of Pure and Applied Sciences, University of Tsukuba, for cooperation and support in daily activities in research and teaching. In particular, I am very grateful to Professor Masahiko Miyamoto, Chair of the department; and Professor Hisao Kato, the person in charge of graduate program in the department; for their arrangements for my dissertation committee.

The author acknowledges that some parts in this work have been supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan, under Grant-in-Aid for Scientific Research (KAKENHI); and University of Tsukuba, under grant-in-aid for research projects.

I would also like to thank Mr. Jiro Miyamoto, the instructor of mathematics when I was in the high school, for guiding me to the journey into the world of mathematics, which I am still continuing, and search for *toretate,* or "just harvested" theorems, for which I am still looking now.

Finally, but not least, I express my deepest appreciation to my wife, Mayumi, for understanding and patience; my daughter, Chihiro, for everyday cheer and hope; and my parents, Yoshio and Tomiko Terui, for countless support and encouragement. This work has only become possible with their unflagging love and dedication.

1 INTRODUCTION

1.1 WHAT IS "SYMBOLIC-NUMERIC ALGORITHMS" ?

The theory of computer algebra (symbolic computation) has been constructed based on the arithmetic including polynomials and/or rational functions over algebra such as groups, rings and fields (finite fields, the fields of real or complex numbers, algebraic extension fields, etc.) with exact arithmetic. Exact arithmetic on the coefficients was a demand for consistency of such algebraic computations, which made significant contributions for solving certain problems in scientific computing.

However, the most of scientific computations has been carried out using numeric computation with the floating-point arithmetic, by the following reasons:

- Sources of data are often observed values thus they frequently contain numerical errors;
- The floating-point arithmetic is far more efficient than the exact arithmetic;
- Numerous numerical methods have been developed for solving various problems accurately, stably and efficiently, with comprehensive analysis on computational accuracy, stability, complexity, and so on.

Nevertheless, there exist some situations in numercal methods which algebraic computations with polynomials and/or rational functions are effective, which have lead to try the following approaches:

(1) Executing algebraic methods in numerical computation;

(2) Executing polynomial and/or rational function computations over the floating-point arithmetic. This approach is called "approximate algebraic computation," which has been initiated in the field of computer algebra [42] and has been attracting attention for new research results.

"Symbolic-numeric computation" includes above approaches in scientific computations, which are getting more popular in recent decades. In the field of symbolic and algebraic computation, an international workshop on symbolic-numeric algebra for polynomials (SNAP) was held in 1996 [16], followed by a series of international workshops on symbolic-numeric computation (SNC) (SNC '05 [62], SNC '07 [63], and SNC '09 [22]).

1.2 TOPICS DISCUSSED IN THIS WORK

Now the reader should understand that "symbolic-numeric algorithms" are algorithms designed to execute symbolic-numeric computations. In the dissertation, we discuss symbolic-numeric algorithms based on above approaches. First, we discuss on the following topic based on approach (1).

THE DURAND-KERNER METHOD FOR THE REAL ROOTS (Chapter 2 [59]). Given a univariate real polynomial, we consider calculating all the real zero-points of the polynomial simultaneously. Among several numerical methods for calculating zero-points of a univariate polynomial, the Durand-Kerner method is quite useful because it is most stable to converge to the zero-points. On the basis of the Durand-Kerner method, we propose two methods for calculating the real zero-points of a univariate polynomial simultaneously. Convergence and error analysis of our methods are discussed. We compared our methods, the original Durand-Kerner method and the Newton's method and found that 1) our methods are more stable than the Newton's method but less than the original Durand-Kerner method, and 2) they are more efficient than the original Durand-Kerner method but less than the Newton's method. We conclude that our methods are useful when good initial values of the zero-points are known.

Then, we discuss on the following topics based on approach (2).

''APPROXIMATE ZERO-POINTS'' OF REAL UNIVARIATE POLYNOMIAL WITH LARGE ERROR TERMS (Chapter 3 [60]). Let $P(x)$ be a real univariate polynomial and let $\tilde{P}(x) = P(x) + \Delta(x)$, where $\Delta(x)$ is the sum of error terms, that is, a polynomial with small real unknown but bounded coefficients. We first consider specifying the "existence domain" of the values of $\tilde{P}(x)$, or the domain in which the value of $\tilde{P}(x)$ exists for any real number x, by the coefficient bounds for $\Delta(x)$, and then introduce a concept of an "approximate real zero-point" of $\tilde{P}(x)$. We present a practical method for estimating the existence domain of zero-points of $\tilde{P}(x)$ by applying the Smith's theorem. We next consider counting the number of real zero-points of $\tilde{P}(x)$. If all the zero-points are sufficiently far apart from each other, the number of real zero-points of $\tilde{P}(x)$ is the same as that of $P(x)$, and we derive a condition for which we can assert that $P(x)$ and $\tilde{P}(x)$ have the same number of real zero-points. We calculate the actual number of real zero-points by the Sturm's method, which encounters the so-called small leading coefficient problem. For this problem, we show that, under some conditions, small leading terms can be discarded. Furthermore, we investigate four methods for evaluating the effect of error terms on the elements of the Sturm sequence.

RECURSIVE POLYNOMIAL REMAINDER SEQUENCE AND ITS SUBRESULTANTS (Chapter 4 ([54], [55], [56])). We introduce concepts of "recursive polynomial remainder sequence (PRS)" and "recursive subresultant," along with investigation of their properties. A *recursive PRS* is defined as, if there exists the greatest common divisor (GCD) of initial polynomials, a sequence of PRSs calculated "recursively" for the GCD and its derivative until a constant is derived, and *recursive subresultants* are defined by determinants representing the coefficients in recursive PRS as functions of coefficients of initial polynomials. We give three different constructions of subresultant matrices for recursive subresultants; while the first one is built-up just with previously defined matrices thus the size of the matrix increases fast as the recursion deepens, the last one reduces the size of the matrix drastically by the Gaussian elimination on the second one which has a "nested" expression, *i.e.* a Sylvester matrix whose elements are themselves determinants.

GPGCD: AN ITERATIVE METHOD FOR CALCULATING APPROXIMATE GCD OF UNIVARIATE POLYNOMIALS (Chapter 5 ([57], [58])). We present an iterative algorithm for calculating approximate greatest common divisor (GCD) of univariate polynomials with the real or the complex coefficients. For a given pair of polynomials and a degree, our algorithm finds a pair of polynomials which has a GCD of the given degree and whose coefficients are perturbed from those in the original inputs, making the perturbations as small as possible, along with the GCD. The problem of approximate GCD is transfered to a constrained minimization problem, then solved with the so-called modified Newton method, which is a generalization of the gradient-projection method, by searching the solution iteratively. We demonstrate that our algorithm calculates approximate GCD with perturbations as small as those calculated by a method based on the structured total least norm (STLN) method, while our method runs significantly faster than theirs by approximately up to 30 times, compared with their implementation. We also show that our algorithm properly handles some ill-conditioned problems with GCD containing small or large leading coefficient.

2 THE DURAND-KERNER METHOD FOR THE REAL ROOTS

Given a univariate real polynomial $P(x)$, we consider calculating all the real zero-points of $P(x)$ simultaneously.

The Durand-Kerner method ([15], [27]) is widely used for calculating both the real and the complex zero-points of a univariate polynomial simultaneously. However, there are many cases in which we need only the real zero-points of a polynomial. For example, consider drawing an algebraic function on the real plane, where the algebraic function is defined by the root of $F(x, y) = 0$ w.r.t. x, with $F(x, y)$ a real bivariate polynomial. We draw a graph of the function by calculating a finite number of real zero-points of $F(x, \bar{y})$ numerically for $\bar{y} = y_j$ $(j = 1, \ldots, k)$, with $y_1, \ldots, y_k$ suitable values of y-coordinate, then connecting these zero-points properly to approximate the graph of the function. In such a case, except for the neighborhood of the singular points, we have only to calculate the real zero-points of univariate polynomial $F(x, \bar{y})$.

As an iterative method for calculating real zero-points of a univariate polynomial, the Newton's method is quite popular. However, for multiple or very close zero-points, it is not easy to obtain all of these zero-points properly. On the other hand, the Durand-Kerner method is more stable thus useful than the Newton's method for multiple or very close zero-points.

In this chapter, on the basis of the Durand-Kerner method, we propose two ways of calculating the real zero-points of a real univariate polynomial. In Section 2.1, we propose a new setting of initial values for the Durand-Kerner method, which allows us to calculate the real and the complex zero-points distinguishably. In Section 2.2, we modify the iteration formula of the Durand-Kerner method to calculate only the real zero-points of a polynomial. In Section 2.3, we compare computing time of methods proposed with that of the original Durand-Kerner method. In Section 5.4, several numerical examples are shown to manifest efficiency, usefulness and weakness of the methods proposed.

2.1 THE DK METHOD AND SMITH'S THEOREM

In order to make the thesis self-contained, we give a brief survey on the Durand-Kerner method, or the DK method in short, and the Smith's theorem for the error estimation. (For details, see the literature ([15], [21], [27], [33], [38], [64]).)

2.1.1 The DK Iteration Formula

Let $P(x)$ be a univariate polynomial with complex coefficients, given as

$$P(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n, \quad a_0 \neq 0.$$

In the quadratic DK method, we give numbers $x_1^{(0)}$, $x_2^{(0)}$, ..., $x_n^{(0)}$ initially and calculate numbers $x_1^{(\nu)}$, $x_2^{(\nu)}$, ..., $x_n^{(\nu)}$ iteratively by

$$x_j^{(\nu+1)} = x_j^{(\nu)} - \frac{P(x_j^{(\nu)})}{a_0 \prod_{k=1,\neq j}^{n} (x_j^{(\nu)} - x_k^{(\nu)})}, \tag{2.1}$$

for $j = 1, \ldots, n$. Then, after some number of iterations, the numbers $x_1^{(\nu)}$, ..., $x_n^{(\nu)}$ approximate the zero-points of $P(x)$ quite well. The initial values $x_1^{(0)}$, ..., $x_n^{(0)}$ are usually determined by the so-called Aberth's method [1].

2.1.2 The Smith's Theorem

Accurate estimation of errors of numerical results is usually quite difficult. In the case of algebraic equation solving, however, we have the following celebrated Smith's theorem [51] which allows us to determine reasonable upper bounds of the approximate zero-points computed numerically.

Theorem 2.1 (Smith [51]). *Let x_1, ..., x_n be n distinct complex numbers and define*

$$r_j = \left| \frac{nP(x_j)}{a_0 \prod_{k=1,\neq j}^{n} (x_j - x_k)} \right|,$$

for $j = 1, \ldots, n$. Let D_j $(1 \leqslant j \leqslant n)$ be a disc of radius r_j and the center at x_j. Then, the union $D_1 \cup \cdots \cup D_n$ contains all the zero-points of $P(x)$. Furthermore, if a union $D_1 \cup \cdots \cup D_m$ $(m \leqslant n)$ is connected and does not intersect with $D_{m+1}, \ldots, D_n$, then this union contains exactly m zero-points. $\square$

Corollary 2.2. *Let α_1, ..., α_n and $x_1^{(\nu)}$, ..., $x_n^{(\nu)}$ be the zero-points of $P(x)$ and their approximations, respectively. If every α_j is a single zero-point and all the discs defined above do not intersect with each other, then, for $j = 1, \ldots, n$, the errors of approximations $x_j^{(\nu)}$ is bounded as*

$$|x_j^{(\nu)} - \alpha_j| \leqslant n \left| \frac{P(x_j^{(\nu)})}{a_0 \prod_{k=1,\neq j}^{n} (x_j^{(\nu)} - x_k^{(\nu)})} \right|. \tag{2.2}$$

$\square$

The above bound is called the Smith's error bound. Note that the right-hand-side of (2.2) is proportional to the "correction term" in (2.1), although the above formula is derived independently from the DK method.

2.1.3 Initial Values for a Real Polynomial

All the complex zero-points of a real polynomial $P(x)$ are pairwise conjugate. For a real polynomial $P(x)$, we can determine the initial values by modifying the Aberth's method as follows.

1. Let the zero-points be distributed within a circle of radius r and the center at β. Calculate β and r_0, an upper bound for r, from the coefficients of $P(x)$.
2. Let m be the number of real zero-points of $P(x)$, with μ multiple zero-points counted as μ. Determine the value of m by the Sturm's method [52].

3. For $j = 1, \dots, m$, determine the initial values for real zero-points as

$$x_j{}^{(0)} = \beta + r_0 \cdot \left[1 - \frac{2(j-d)}{m+1}\right], \tag{2.3}$$

   where $d \approx 0.3$.

4. For $j = m+1,\ m+3,\ \dots,\ n-1$, determine the initial values for imaginary zero-points as

$$\begin{aligned} x_j{}^{(0)} &= \beta + r_0 \cdot \exp\left[\frac{j-m-d}{n-m}\pi i\right], \\ x_{j+1}{}^{(0)} &= \overline{x_j{}^{(0)}}, \end{aligned} \tag{2.4}$$

   where $d \approx 0.3$.

The number d is set to prevent the initial values to be placed symmetrically with respect to the imaginary axis. With the above-defined initial values for the DK method, we expect that $x_j{}^{(\nu)}$ is real if $x_j{}^{(0)}$ is so and $x_{j+1}{}^{(\nu)} = \overline{x_j{}^{(\nu)}}$ if $x_{j+1}{}^{(0)} = \overline{x_j{}^{(0)}}$. This expectation is in fact true by the following proposition.

Proposition 2.3. *Let* $P(x)$, n, m *and* $x_j{}^{(\nu)}$ $(j = 1, \dots, n; \nu = 0, 1, \dots)$ *be defined as above. Then, for* $\nu = 0, 1, 2, \dots$, *we have the followings.*

1. *For* $j = 1, \dots, m$, *if* $x_j{}^{(0)}$ *is real then* $x_j{}^{(\nu)}$ *is also real.*

2. *For* $j = m+1, m+3, \dots, n-1$, *if* $x_{j+1}{}^{(0)} = \overline{x_j{}^{(0)}}$ *then* $x_{j+1}{}^{(\nu)} = \overline{x_j{}^{(\nu)}}$.

Proof. By the induction on ν. The proposition is obviously valid for $\nu = 0$. By the induction hypothesis, assume that the proposition is valid for $0, \dots, \nu$. Put $\tilde{P}_\nu(x) = a_0 \prod_{k=1}^{n} (x - x_k{}^{(\nu)})$ then formula (2.1) can be expressed as

$$x_j{}^{(\nu+1)} = x_j{}^{(\nu)} - \frac{P(x_j{}^{(\nu)})}{\tilde{P}'_\nu(x_j{}^{(\nu)})},$$

where $\tilde{P}'_\nu(x)$ is the derivative of $\tilde{P}_\nu(x)$. Note that $\tilde{P}_\nu(x)$ and $\tilde{P}'_\nu(x)$ are real polynomials. Therefore, if $x_j{}^{(\nu)}$ is a real number then $x_j{}^{(\nu+1)}$ is also a real number because $P(x_j{}^{(\nu)})$ and $\tilde{P}'_\nu(x_j{}^{(\nu)})$ are real, and if $x_{j+1}{}^{(\nu)} = \overline{x_j{}^{(\nu)}}$ then $x_{j+1}{}^{(\nu+1)} = \overline{x_j{}^{(\nu+1)}}$ because $P(x_{j+1}{}^{(\nu)})$ and $\tilde{P}'_\nu(x_{j+1}{}^{(\nu)})$ are complex conjugate of $P(x_j{}^{(\nu)})$ and $\tilde{P}'_\nu(x_j{}^{(\nu)})$, respectively. Thus, the proposition is valid for $\nu + 1$, which proves the proposition. □

Remark 2.1. The above proposition is also valid for the cubic DK method.

Remark 2.2. We have only to calculate one of mutually conjugate roots, which makes the computation considerably efficient.

Remark 2.3. In the above method, the Sturm's method is applied only for determining the number of real zero-points, not for determining the position of real zero-points. When we calculate the position of real zero-points by the Sturm's method, we first calculate the Sturm's sequence, then calculate the values of elements of the sequence accurately for quite a few times. However, for calculating the number of real zero-points, we have only to find the sign changes of leading coefficients of elements of the Sturm's sequence, which is quite fast.

Remark 2.4. If we set $d = 0$ in formulas (2.3) and (2.4), the initial values may not converge to the true zero-points. For example, consider $P(x) = (x^2+4)(x^2+9)$ which has zero-points at $x = \pm 2\boldsymbol{i}$ and $x = \pm 3\boldsymbol{i}$, and define the initial values as $x_1{}^{(0)} = a + b\boldsymbol{i}$, $x_2{}^{(0)} = \overline{x_1{}^{(0)}}$, $x_3{}^{(0)} = -x_1{}^{(0)}$ and $x_4{}^{(0)} = -\overline{x_1{}^{(0)}}$, where $a, b \in \mathbf{R}$. Then, formula (2.1) gives us $x_2{}^{(\nu)} = \overline{x_1{}^{(\nu)}}$, $x_3{}^{(\nu)} = -x_1{}^{(\nu)}$ and $x_4{}^{(\nu)} = -\overline{x_1{}^{(\nu)}}$ for $\nu = 1, 2, \ldots$, thus some of $x_j{}^{(\nu)}$ will never converge to the corresponding zero-points. Note that, even if we set as $d \approx 0.3$, the DK method may not give a sequence of numbers which converges to the zero-points. Therefore, we must be careful to define the initial values for the DK method so that they converge correctly.

Throughout this paper, we call the DK method with the above-determined initial values "**DKAreal**," while we call the DK method with Aberth's initial values "**DKA**."

2.2 THE DK METHOD CALCULATING ONLY REAL ROOTS

In this section, assuming that $P(x)$ is monic and that all the coefficients of $P(x)$ are real, we make a modification of the DK method so that we will calculate only the real zero-points of $P(x)$.

Let $F(x)$ and $G(x)$ be polynomials and assume that the degree of $F(x)$ is greater than or equal to that of $G(x)$. Throughout this paper, we denote polynomial quotient of $F(x)$ divided by $G(x)$ by $\mathrm{quotient}(F(x), G(x))$.

2.2.1 Modification of the DK Formula

We first state how we modify the DK method.

1. Let the roots be distributed within a circle of radius r and the center at β. Calculate β and r_0, an upper bound for r, from the coefficients of $P(x)$. Note that β is real.

2. Let m be the number of real zero-points of $P(x)$, with μ multiple zero-points counted as μ. Determine the value of m by the Sturm's method.

3. For $j = 1, \ldots, m$, determine the initial value for real zero-points as

$$x_j{}^{(0)} = \beta + r_0 \left[1 - \frac{2(j-d)}{m+1} \right],$$

   where $d \approx 0.3$.

4. For $j = 1, \ldots, m$, modify the DK iteration formula as

$$x_j{}^{(\nu+1)} = x_j{}^{(\nu)} - \frac{P(x_j{}^{(\nu)})}{\prod_{k=1,\neq j}^{m} (x_j{}^{(\nu)} - x_k{}^{(\nu)}) \cdot Q_\nu(x_j{}^{(\nu)})}, \tag{2.5}$$

   where

$$Q_\nu(x) = \mathrm{quotient}(P(x), \textstyle\prod_{k=1}^{m} (x - x_k{}^{(\nu)})). \tag{2.6}$$

Obviously the calculated values $x_1{}^{(\nu)}$, $x_2{}^{(\nu)}$, $\ldots$, $x_n{}^{(\nu)}$ are real numbers for all ν. Throughout this paper, we call the above method "**DKreal**."

2.2.2 Convergence Property

Let $\alpha_1, \dots, \alpha_n$ be the zero-points of $P(x)$, where we assume that the real zero-points are $\alpha_1, \dots, \alpha_m$ $(m \leqslant n)$ and $P(x)$ is monic, for simplicity. Assume that $\alpha_1 = \cdots = \alpha_{m'}$ $(m' \leqslant m)$ and $\alpha_k \neq \alpha_1$ for $k \geqslant m'+1$. Let ${x_1}^{(\nu)}, \dots, {x_m}^{(\nu)}$ be the approximations of $\alpha_1, \dots, \alpha_m$, respectively. Put ${\varepsilon_j}^{(\nu)} = {x_j}^{(\nu)} - \alpha_j$ and assume that $|{\varepsilon_j}^{(\nu)}| \ll 1$ for $j = 1, \dots, m$. Then, formula (2.5) allows us to calculate ${\varepsilon_j}^{(\nu+1)}$ $(1 \leqslant j \leqslant m')$ as

$$\begin{aligned}
{\varepsilon_j}^{(\nu+1)} =& {x_j}^{(\nu+1)} - \alpha_j \\
=& {x_j}^{(\nu)} - \frac{P({x_j}^{(\nu)})}{\prod_{k=1,\neq j}^{m}({x_j}^{(\nu)} - {x_k}^{(\nu)}) \cdot Q_\nu({x_j}^{(\nu)})} - \alpha_j \\
=& {\varepsilon_j}^{(\nu)} - \left[\frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({x_j}^{(\nu)} - {x_k}^{(\nu)})}\right]\left[\prod_{k=m'+1}^{m}\frac{{x_j}^{(\nu)} - \alpha_k}{{x_j}^{(\nu)} - {x_k}^{(\nu)}}\right] \\
&\times \left[\frac{\prod_{k=m+1}^{n}({x_j}^{(\nu)} - \alpha_k)}{Q_\nu({x_j}^{(\nu)})}\right].
\end{aligned} \tag{2.7}$$

Let ${\varepsilon_0}^{(\nu)} = \max\{|{\varepsilon_1}^{(\nu)}|, \dots, |{\varepsilon_m}^{(\nu)}|\}$, then we see that

$$\begin{aligned}
\prod_{k=m'+1}^{m}\frac{{x_j}^{(\nu)} - \alpha_k}{{x_j}^{(\nu)} - {x_k}^{(\nu)}} &= \prod_{k=m'+1}^{m}\left(1 + \frac{{\varepsilon_k}^{(\nu)}}{{x_j}^{(\nu)} - {x_k}^{(\nu)}}\right) \\
&= \prod_{k=m'+1}^{m}\left(1 + \frac{{\varepsilon_k}^{(\nu)}}{\alpha_1 + {\varepsilon_j}^{(\nu)} - {x_k}^{(\nu)}}\right) \\
&= \prod_{k=m'+1}^{m}\left(1 + \frac{{\varepsilon_k}^{(\nu)}}{\alpha_1 - {x_k}^{(\nu)}} + O({\varepsilon_j}^{(\nu)}{\varepsilon_k}^{(\nu)})\right) \\
&= \ 1 + \sum_{k=m'+1}^{m}\frac{{\varepsilon_k}^{(\nu)}}{\alpha_1 - {x_k}^{(\nu)}} + O(({\varepsilon_0}^{(\nu)})^2).
\end{aligned}$$

We next consider the quotient $Q_\nu(x)$. The dividend in the right-hand-side of (2.6) is decomposed as

$$\begin{aligned}
P(x) =& \prod_{k=1}^{m}(x - {x_k}^{(\nu)} + {\varepsilon_k}^{(\nu)}) \prod_{k=m+1}^{n}(x - \alpha_k) \\
=& \prod_{k=1}^{m}(x - {x_k}^{(\nu)}) \prod_{k=m+1}^{n}(x - \alpha_k) \\
&+ \sum_{j=1}^{m}\left[{\varepsilon_j}^{(\nu)} \prod_{k=1,\neq j}^{m}(x - x_k^{(\nu)}) \prod_{k=m+1}^{n}(x - \alpha_k)\right] + O(\varepsilon^2(x)) \\
=& \prod_{k=1}^{m}(x - {x_k}^{(\nu)}) \prod_{k=m+1}^{n}(x - \alpha_k) \\
&+ \sum_{j=1}^{m}\left[\varepsilon_j^{(\nu)}(x - x_j^{(\nu)} + x_j^{(\nu)} - \alpha_n) \prod_{k=1,\neq j}^{m}(x - x_k^{(\nu)}) \prod_{k=m+1}^{n-1}(x - \alpha_k)\right] + O(\varepsilon^2(x))
\end{aligned}$$

$$
\begin{aligned}
&= \prod_{k=1}^{m} (x - x_k^{(\nu)}) \prod_{k=m+1}^{n} (x - \alpha_k) \\
&\quad + \sum_{j=1}^{m} \Bigg[\varepsilon_j^{(\nu)} \Bigg\{ \prod_{k=1}^{m} (x - x_k^{(\nu)}) \prod_{k=m+1}^{n-1} (x - \alpha_k) \\
&\quad + (x_j^{(\nu)} - \alpha_n) \prod_{k=1,\neq j}^{m} (x - x_k^{(\nu)}) \prod_{k=m+1}^{n-1} (x - \alpha_k) \Bigg\} \Bigg] + O(\varepsilon^2(x)),
\end{aligned}
$$

where $O(\varepsilon^2(x))$ denotes a polynomial of terms whose coefficients are of magnitudes $O((\varepsilon_0^{(\nu)})^2)$. Continuing this rewriting, we see that

$$
Q_\nu(x) = \textstyle\prod_{k=m+1}^{n} (x - \alpha_k) + \delta Q_\nu(x), \tag{2.8}
$$

where

$$
\begin{aligned}
&\delta Q_\nu(x) \\
&= \sum_{j=1}^{m} \varepsilon_j^{(\nu)} \Bigg\{ \prod_{k=m+1}^{n-1} (x - \alpha_k) + (x_j^{(\nu)} - \alpha_n) \prod_{k=m+1}^{n-2} (x - \alpha_k) + \cdots \Bigg\} + O(\varepsilon^2(x)).
\end{aligned}
$$

Remembering $x_j^{(\nu)} = \alpha_1 + O(\varepsilon_0^{(\nu)})$ for $j = 1, \ldots, m$, we find that

$$
\delta Q_\nu({x_j}^{(\nu)}) = \delta Q_\nu(\alpha_1) + O(({\varepsilon_0}^{(\nu)})^2), \quad \delta Q_\nu(\alpha_1) = O({\varepsilon_0}^{(\nu)}),
$$

and we obtain

$$
\begin{aligned}
\frac{\prod_{k=m+1}^{n} (x_j^{(\nu)} - \alpha_k)}{Q_\nu(x_j^{(\nu)})} &= \frac{\prod_{k=m+1}^{n} (x_j^{(\nu)} - \alpha_k)}{\prod_{k=m+1}^{n} (x_j^{(\nu)} - \alpha_k) + \delta Q_\nu(x_j^{(\nu)})} \\
&= 1 - \frac{\delta Q_\nu({x_j}^{(\nu)})}{\prod_{k=m+1}^{n} ({x_j}^{(\nu)} - \alpha_k)} + O(({\varepsilon_0}^{(\nu)})^2) \\
&= 1 - \frac{\delta Q_\nu(\alpha_1)}{\prod_{k=m+1}^{n} (\alpha_1 - \alpha_k)} + O(({\varepsilon_0}^{(\nu)})^2).
\end{aligned}
$$

In the case that α_1 is a single zero-point, *i.e.* $m' = 1$, the above arguments lead us to that

$$
\begin{aligned}
{\varepsilon_1}^{(\nu+1)} &= {\varepsilon_1}^{(\nu)} - {\varepsilon_1}^{(\nu)} \cdot (1 + O({\varepsilon_0}^{(\nu)}))(1 + O({\varepsilon_0}^{(\nu)})) \\
&= O({\varepsilon_1}^{(\nu)} {\varepsilon_0}^{(\nu)}).
\end{aligned}
$$

As a consequence, we have the following proposition.

Proposition 2.4. *Let $\alpha_1, \ldots, \alpha_m$ be the real zero-points of $P(x)$ and assume that $\alpha_i \neq \alpha_j$ for any $i \neq j$. Let ${x_1}^{(\nu)}, \ldots, {x_m}^{(\nu)}$ be the approximations of $\alpha_1, \ldots, \alpha_m$, respectively, and assume that every ${x_j}^{(\nu)}$ is sufficiently close to α_j. Then, iteration formula (2.5) converges quadratically.* □

If $m' \geqslant 2$, the approximations ${x_1}^{(\nu)}, \ldots, {x_{m'}}^{(\nu)}$ corresponding to m' multiple roots do not converge quadratically. However, we find that the "center" of these approximations, or $({x_1}^{(\nu)} + \cdots + {x_{m'}}^{(\nu)})/m'$, converges quadratically, as we show below.

Proposition 2.5. *Let* $\alpha_1, \ldots, \alpha_m$ *be the real zero-points of* $P(x)$ *and assume that* $\alpha_1 = \cdots = \alpha_{m'}$ $(m' \leqslant m)$ and $\alpha_1 \neq \alpha_i \neq \alpha_j$ for $m' < i < j \leqslant m$. Let ${x_1}^{(\nu)}, \ldots, {x_{m'}}^{(\nu)}$ be the approximations of $\alpha_1, \ldots, \alpha_{m'}$, respectively, and assume that every ${x_j}^{(\nu)}$ is sufficiently close to α_j. Then, $({x_1}^{(\nu)} + \cdots + {x_{m'}}^{(\nu)})/m'$ converges to α_1 quadratically.

Proof. Using (2.7), we can calculate the error of the "center" as

$$\begin{aligned}
\frac{1}{m'}\left|\sum_{j=1}^{m'} {\varepsilon_j}^{(\nu+1)}\right| =& \frac{1}{m'}\left|\sum_{j=1}^{m'}\left\{{\varepsilon_j}^{(\nu)} - \left[\frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({x_j}^{(\nu)} - {x_k}^{(\nu)})}\right]\right.\right. \\
&\left.\left. \times \left[\prod_{k=m'+1}^{m} \frac{{x_j}^{(\nu)} - \alpha_k}{{x_j}^{(\nu)} - {x_k}^{(\nu)}}\right]\left[\frac{\prod_{k=m+1}^{n}({x_j}^{(\nu)} - \alpha_k)}{Q_\nu({x_j}^{(\nu)})}\right]\right\}\right| \\
=& \frac{1}{m'}\left|\sum_{j=1}^{m'} {\varepsilon_j}^{(\nu)} - \sum_{j=1}^{m'}\left\{\left[\frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({\varepsilon_j}^{(\nu)} - {\varepsilon_k}^{(\nu)})}\right]\right.\right. \\
&\times \left[1 + \sum_{k=m'+1}^{m} \frac{{\varepsilon_k}^{(\nu)}}{\alpha_1 - {x_k}^{(\nu)}} + O(({\varepsilon_0}^{(\nu)})^2)\right] \\
&\left.\left.\times \left[1 - \frac{\delta Q_\nu(\alpha_1)}{\prod_{k=m+1}^{n}(\alpha_1 - \alpha_k)} + O(({\varepsilon_0}^{(\nu)})^2)\right]\right\}\right| \\
=& \frac{1}{m'}\left|\sum_{j=1}^{m'} {\varepsilon_j}^{(\nu)} - \sum_{j=1}^{m'} \frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({\varepsilon_j}^{(\nu)} - {\varepsilon_k}^{(\nu)})}\right. \\
&- \sum_{j=1}^{m'}\left\{\left[\frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({\varepsilon_j}^{(\nu)} - {\varepsilon_k}^{(\nu)})}\right]\right. \\
&\times \left[\sum_{k=m'+1}^{m} \frac{{\varepsilon_k}^{(\nu)}}{\alpha_1 - {x_k}^{(\nu)}} - \frac{\delta Q_\nu(\alpha_1)}{\prod_{k=m+1}^{n}(\alpha_1 - \alpha_k)}\right. \\
&\left.\left.\left. + O(({\varepsilon_0}^{(\nu)})^2)\right]\right\}\right|.
\end{aligned} \tag{2.9}$$

In the right-hand-side of (2.9), we have

$$\sum_{j=1}^{m'} \frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({\varepsilon_j}^{(\nu)} - {\varepsilon_k}^{(\nu)})} = \sum_{j=1}^{m'} {\varepsilon_j}^{(\nu)}, \tag{2.10}$$

which can be seen as follows. Putting

$$F = \sum_{j=1}^{m'} ({\varepsilon_j}^{(\nu)})^{m'} \frac{\prod_{j'=1}^{m'-1}\prod_{k=j'+1}^{m'}({\varepsilon_{j'}}^{(\nu)} - {\varepsilon_k}^{(\nu)})}{\prod_{k=1,\neq j}^{m'}({\varepsilon_j}^{(\nu)} - {\varepsilon_k}^{(\nu)})},$$

$$G = \prod_{j'=1}^{m'-1}\prod_{k=j'+1}^{m'}({\varepsilon_{j'}}^{(\nu)} - {\varepsilon_k}^{(\nu)}),$$

we see that

$$\sum_{j=1}^{m'} \frac{({\varepsilon_j}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}({\varepsilon_j}^{(\nu)} - {\varepsilon_k}^{(\nu)})} = \frac{F}{G}.$$

We note that F and G are polynomials in $\varepsilon_j{}^{(\nu)}$ $(j = 1, \dots, m')$. If we put $\varepsilon_1{}^{(\nu)} = \varepsilon_2{}^{(\nu)}$, then we see that $G = 0$ directly and find that $F = 0$ because the first and the second terms in the summation over j cancel each other and other terms become zero. Similarly, for any $i \neq j$, if we put $\varepsilon_i{}^{(\nu)} = \varepsilon_j{}^{(\nu)}$ then we obtain $F = G = 0$. This means that every factor in G is contained in F, therefore we find $G|F$. Putting $H = F/G$, we see from F and G that H is a symmetric polynomial. With respect to any "variable" $\varepsilon_j^{(\nu)}$, $1 \leqslant j \leqslant m'$, the degree of H is 1, H contains no constant term and the leading coefficient of H is 1 because those of F and G are 1. Therefore, H is an elementary symmetric polynomial of degree 1, which is equal to $\varepsilon_1{}^{(\nu)} + \cdots + \varepsilon_{m'}{}^{(\nu)}$.

Eq. (2.10) allows us to rewrite (2.9) as

$$\begin{aligned}
&\frac{1}{m'}\left|\sum_{j=1}^{m'} \varepsilon_j{}^{(\nu+1)}\right| \\
=&\frac{1}{m'}\left|\sum_{j=1}^{m'}\left\{\left[\frac{(\varepsilon_j{}^{(\nu)})^{m'}}{\prod_{k=1,\neq j}^{m'}(\varepsilon_j{}^{(\nu)} - \varepsilon_k{}^{(\nu)})}\right]\right.\right. \\
&\times\left.\left.\left[\sum_{k=m'+1}^{m} \frac{\varepsilon_k{}^{(\nu)}}{\alpha_1 - x_k{}^{(\nu)}} - \frac{\delta Q_\nu(\alpha_1)}{\prod_{k=m+1}^{n}(\alpha_1 - \alpha_k)} + O((\varepsilon_0{}^{(\nu)})^2)\right]\right\}\right| \\
=&O((\varepsilon_0{}^{(\nu)})^2).
\end{aligned}$$

Therefore, the "center" of $x_1{}^{(\nu)}$, ..., $x_m{}^{(\nu)}$ converges to α_1 quadratically, which proves the proposition. □

A property described in Proposition 2.5 is called "quadratic-like convergence of the mean," discussed in Fraigniaud [18] and Iri [21] for the DK method and in Pasquini and Trigiante [37] for its variations.

2.2.3 Error Bound

We can estimate an upper bound of the errors of the approximations $x_1{}^{(\nu)}, \dots, x_m{}^{(\nu)}$, by the following proposition.

Proposition 2.6. *Let* $P(x)$, α_k $(k = 1, \dots, n)$ *and* $x_j{}^{(\nu)}$ $(j = 1, \dots, m)$ *be defined as above. For* $j = 1, \dots, m$, *let* r'_j *be*

$$r'_j = n\left|\frac{P(x_j{}^{(\nu)})}{a_0\prod_{k=1,\neq j}^{m}(x_j{}^{(\nu)} - x_k{}^{(\nu)}) \cdot Q_\nu(x_j{}^{(\nu)})}\right|,$$

and D_j $(1 \leqslant j \leqslant m)$ *be a disc of radius* r'_j *and center at* $x_j{}^{(\nu)}$. *Then, for* $j = 1, \dots, m$, *the error of approximation* $x_j{}^{(\nu)}$ *is bounded as*

$$|x_j{}^{(\nu)} - \alpha_j| \leqslant r'_j\left|1 - \frac{\delta Q_\nu(x_j{}^{(\nu)})}{Q_\nu(x_j{}^{(\nu)})}\right|^{-1},$$

if all the discs D_j, $j = 1, \dots, m$, *do not intersect with each other and each* $x_j{}^{(\nu)}$ *is sufficiently close to* α_j.

Proof. Applying Theorem 2.1 to the case $x_j = x_j{}^{(\nu)}$ for $j = 1, \ldots, m$ and $x_k = \alpha_k$ for $k = m+1, \ldots, n$, we have

$$r_j = n \left| \frac{P(x_j{}^{(\nu)})}{a_0 \prod_{k=1,\neq j}^{m} (x_j{}^{(\nu)} - x_k{}^{(\nu)}) \cdot \prod_{k=m+1}^{n} (x_j{}^{(\nu)} - \alpha_k)} \right| .$$

Using (2.8), we rewrite the above expression as

$$r_j = n \left| \frac{P(x_j{}^{(\nu)})}{a_0 \prod_{k=1,\neq j}^{m} (x_j{}^{(\nu)} - x_k{}^{(\nu)}) \cdot \{Q_\nu(x_j{}^{(\nu)}) - \delta Q_\nu(x_j{}^{(\nu)})\}} \right| .$$

Since $x_j{}^{(\nu)}$ is sufficiently close to α_j, we have

$$|Q_\nu(x_j{}^{(\nu)})| \gg |\delta Q_\nu(x_j{}^{(\nu)})| = O(\varepsilon_0{}^{(\nu)}).$$

Therefore, we have

$$\begin{aligned} r_j &= n \left| \frac{P(x_j{}^{(\nu)})}{a_0 \prod_{k=1,\neq j}^{m} (x_j{}^{(\nu)} - x_k{}^{(\nu)}) \cdot Q_\nu(x_j{}^{(\nu)})} \right| \cdot \left| 1 - \frac{\delta Q_\nu(x_j{}^{(\nu)})}{Q_\nu(x_j{}^{(\nu)})} \right|^{-1} \\ &= r_j' \left| 1 - \frac{\delta Q_\nu(x_j{}^{(\nu)})}{Q_\nu(x_j{}^{(\nu)})} \right|^{-1} , \end{aligned} \tag{2.11}$$

which proves the proposition. □

Remark 2.5. We can estimate the number $|1 - \delta Q_\nu(x_j{}^{(\nu)})/Q_\nu(x_j{}^{(\nu)})|^{-1}$ in (2.11) as

$$\left| 1 - \frac{\delta Q_\nu(x_j{}^{(\nu)})}{Q_\nu(x_j{}^{(\nu)})} \right|^{-1} \simeq 1 + \frac{|\delta Q_\nu(x_j{}^{(\nu)})|}{|Q_\nu(x_j{}^{(\nu)})|}.$$

Therefore, for $i = 1, \ldots, m$, the error of $x_j{}^{(\nu)}$ is approximately bounded as

$$|x_j{}^{(\nu)} - \alpha_j| \lesssim r_j' \left(1 + \frac{|\delta Q_\nu(x_j{}^{(\nu)})|}{|Q_\nu(x_j{}^{(\nu)})|} \right).$$

Remark 2.6. We can estimate the number $|\delta Q_\nu(x_j{}^{(\nu)})/Q_\nu(x_j{}^{(\nu)})|$ in (2.11) quite precisely as follows. Formula (2.8) tells us that

$$\begin{aligned} Q_\nu(x_j{}^{(\nu)}) - Q_{\nu+1}(x_j{}^{(\nu)}) &= \delta Q_\nu(x_j{}^{(\nu)}) - \delta Q_{\nu+1}(x_j{}^{(\nu)}) \\ &\simeq \delta Q_\nu(x_j{}^{(\nu)}), \end{aligned}$$

because $|\delta Q_{\nu+1}(x_j{}^{(\nu)})| \ll |\delta Q_\nu(x_j{}^{(\nu)})|$. Therefore, we have

$$\left| \frac{\delta Q_\nu(x_j{}^{(\nu)})}{Q_\nu(x_j{}^{(\nu)})} \right| \simeq \frac{|Q_\nu(x_j{}^{(\nu)}) - Q_{\nu+1}(x_j{}^{(\nu)})|}{|Q_\nu(x_j{}^{(\nu)})|}.$$

2.3 COMPUTING TIME ANALYSIS

Let $P(x)$ be a monic polynomial of degree n and let m be the number of real zero-points of $P(x)$, as above. In this section, we analyze the computing time for one

Real arithmetic	Complex arithmetic		
	Addition	Multiplication	Division
Addition	2	2	3
Multiplication	—	4	6
Division	—	—	2
Total	2	6	11

Table 2.1: The number of real arithmetic operations required for each complex arithmetic operation.

iteration in algorithms **DKA**, **DKAreal** and **DKreal**. Here, by "computing time," we mean the number of arithmetic operations (addition, multiplication and division) on real numbers, because the ordinary CPU executes only real arithmetic operations. For operations on polynomials, we consider only the number of coefficient operations and discard the operations on structuring polynomials such as sorting and collecting terms.

As Table 2.1 shows, complex arithmetic operations can be done by combinations of real arithmetic operations. For example, let $z_j = a_j + b_j \boldsymbol{i}$ ($j = 1, 2$) with $a_j, b_j \in$ **R** and $z_2 \neq 0$, then the division of z_1 by z_2 is executed as $z_1/z_2 = \{(a_1a_2 + b_1b_2) + (-a_1b_2 + b_1a_2)\boldsymbol{i}\}/({a_2}^2 - {b_2}^2)$, or three additions, six multiplications and two divisions. In this paper, by "real" addition, multiplication and division and by "complex" addition, multiplication and division, we mean the arithmetic operations on real numbers and complex numbers, respectively.

2.3.1 DKA Method

In iteration formula (2.1), computation of the denominator $\prod_{k=1,\neq j}^{n}({x_j}^{(\nu)} - {x_k}^{(\nu)})$ requires $n-1$ complex additions and $n-2$ complex multiplications, thus it requires $4n-6$ real additions and $4n-8$ real multiplications. Computation of $P({x_j}^{(\nu)})$ can be done by the Horner's rule. Since every coefficient in $P(x)$ is real, it requires n real additions and and n complex multiplications, or $3n$ real additions and $4n$ real multiplications. Consequently, ${x_j}^{(\nu+1)}$ can be calculated by $15n$ arithmetic operations for each j. Therefore, the computing time for one iteration of **DKA** is $15n^2$.

2.3.2 DKAreal Method

We apply iteration formula (2.1) to m real zero-points ${x_j}^{(\nu)}$ for $j = 1, \ldots, m$ and to $(n-m)/2$ complex zero-points ${x_j}^{(\nu)}$ for $j = m+1, m+3, \ldots, n-1$. We consider the real zero-points and the complex zero-points separately.

For real ${x_j}^{(\nu)}$: Calculation of the product $\prod_{k=1,\neq j}^{m}({x_j}^{(\nu)} - {x_k}^{(\nu)})$ requires $m-1$ real additions and $m-2$ real multiplications, because all the numbers concerned are real. We calculate the product $\prod_{k=m+1}^{n}({x_j}^{(\nu)} - {x_k}^{(\nu)})$ as

$$\prod_{l=1}^{(n-m)/2} \left({x_j}^{(\nu)} - {x_{m+2l-1}}^{(\nu)}\right)\left({x_j}^{(\nu)} - {x_{m+2l}}^{(\nu)}\right)$$
$$= \prod_{l=1}^{(n-m)/2} \left[\{{x_j}^{(\nu)} - \mathrm{Re}({x_{m+2l-1}}^{(\nu)})\}^2 + \mathrm{Im}({x_{m+2l-1}}^{(\nu)})^2\right].$$

Thus, we can calculate it by $n-m$ real additions and $\frac{3}{2}(n-m)-1$ real multiplications. Therefore, calculation of $\prod_{k=1,\neq j}^{n}({x_j}^{(\nu)} - {x_k}^{(\nu)})$ requires $n-1$ real additions and $\frac{3}{2}n - \frac{1}{2}m - 2$ real multiplications. Computation of $P({x_j}^{(\nu)})$ by the Horner's rule requires n real additions and n real multiplications. Consequently, ${x_j}^{(\nu+1)}$ is calculated by $\frac{9}{2}n - \frac{1}{2}m$ real arithmetic operations.

For complex ${x_j}^{(\nu)}$: Since $\prod_{k=1}^{m}({x_j}^{(\nu)} - {x_k}^{(\nu)}) = \prod_{k=1}^{m}\{\mathrm{Re}({x_j}^{(\nu)}) - {x_k}^{(\nu)} + \mathrm{Im}({x_j}^{(\nu)})\boldsymbol{i}\}$, we can calculate it by m real additions and $m-1$ complex multiplications, or by $3m-2$ real additions and $4(m-1)$ real multiplications. We calculate the product $\prod_{k=m+1,\neq j}^{n}({x_j}^{(\nu)} - {x_k}^{(\nu)})$ by $n-m-1$ complex additions and $n-m-2$ complex multiplications, or $4(n-m)-6$ real additions and $4(n-m-2)$ real multiplications. Computation of $P({x_j}^{(\nu)})$ is executed by the Horner's rule with the same computing time as in **DKA**. Therefore, ${x_j}^{(\nu+1)}$ is calculated by $15n-m-1$ real arithmetic operations.

Consequently, the computing time for one iteration in **DKAreal** is

$$m\left(\frac{9}{2}n - \frac{1}{2}m\right) + \left(\frac{n-m}{2}\right)(15n-m) = \frac{15}{2}n^2 - \frac{7}{2}mn.$$

We see that, if the number of real zero-points is small, the computing time of **DKAreal** is approximately equal to the half of that of **DKA**, and it decreases as m increases.

2.3.3 **DKreal** Method

In **DKreal**, all the computations are done with real arithmetic. In each iteration step of (2.5), we calculate $Q_\nu(x)$ in (2.6). Since the multiplication of two monic polynomials of degrees k and 1 requires $k-1$ additions and $k-1$ multiplications on their coefficients, calculation of a term $\prod_{k=1}^{m}(x - {x_k}^{(\nu)})$ requires $m^2 - m$ arithmetic operations. Next, we consider the quotient$(P(x), \prod_{k=1}^{m}(x - {x_k}^{(\nu)}))$. Let $F(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ and $G(x) = \prod_{k=1}^{m}(x - {x_k}^{(\nu)}) = x^m + b_1x^{m-1} + \cdots + b_{m-1}x + b_m$. The elimination of the leading term of $F(x)$ by $G(x)$ is executed as $F(x) - a_0x^{n-m}G(x)$ and it takes $2m$ arithmetic operations (m multiplications for calculating the coefficients of $a_0x^{n-m}G(x)$ and m additions for calculating the coefficients of $F(x) - a_0x^{n-m}G(x)$). Since the polynomial division of $P(x)$ by $G(x)$ requires $n-m+1$ eliminations of the leading term, calculation of the quotient$(P(x), \prod_{k=1}^{m}(x - {x_k}^{(\nu)}))$ requires $2m(n-m+1)$ arithmetic operations. Therefore, the computing time for $Q_\nu(x)$ is $2mn - m^2 + m$.

In iteration formula (2.5) for **DKreal**, the denominator $\prod_{k=1,\neq j}^{m}({x_j}^{(\nu)} - {x_k}^{(\nu)})$ requires $m-1$ additions and $m-2$ multiplications. Evaluations of $P({x_j}^{(\nu)})$ and $Q_\nu({x_j}^{(\nu)})$ require $2n$ and $2(n-m)$ arithmetic operations, respectively. Consequently, ${x_j}^{(\nu+1)}$ can be calculated by $4n-1$ arithmetic operations for each j. Therefore, the computing time for one iteration in **DKreal** is $6mn - m^2$.

We see that, if the number of real zero-points is small, the computing time of **DKreal** is approximately equal to $\frac{2}{5}m/n$ of that of **DKA**. However, as the number of real zero-points increases, the computing time of **DKreal** increases in proportion to m.

2.4 EXPERIMENTS

We have implemented the algorithms **DKA**, **DKAreal** and **DKreal** on a computer algebra system GAL [43] (General Algebraic Language/Laboratory, a LISP-based general purpose computer algebra system), using numeric computation functions in LISP and GAL's facility of computing polynomial quotient. All the following experiments were carried out on a SPARC Station 5 (microSPARC II of 85 MHz) with 32MB RAM. The computing time is shown in milli-seconds and the time for garbage collection is discarded.

2.4.1 Comparison in the General Case

We first consider the case that we have no information on the zero-points of target polynomial. Let $P_{1,k}(x)$ $(k = 0, 1, \ldots, 10)$ and $P_{2,k}(x)$ $(k = 0, 1, \ldots, 20)$ be polynomials of degree 20 and 40, respectively, containing $2k$ real zero-points, as

$$\begin{aligned}
P_{1,k}(x) &= \prod_{j=1}^{k}\left\{x^2 - (1 - \frac{2j}{2k+1})^2\right\} \\
&\quad \times \prod_{j=k+1}^{10-k}\left\{[x - \cos(\delta + \frac{j}{11-k})\pi]^2 + [\sin(\delta + \frac{j}{11-k})\pi]^2\right\}, \\
P_{2,k}(x) &= \prod_{j=1}^{k}\left\{x^2 - (1 - 2\frac{j}{2k+1})^2\right\} \\
&\quad \times \prod_{j=k+1}^{20-k}\left\{[x - \cos(\delta + \frac{j}{21-k})\pi]^2 + [\sin(\delta + \frac{j}{21-k})\pi]^2\right\},
\end{aligned}$$

where δ is a small number $(0 < \delta \ll 1)$ such that complex zero-points are not placed symmetrically to the imaginary axis. In our experiment we set $\delta = 0.03$.

Figures 2.1 and 2.2 show the result of computations of the real zero-points of $P_{1,k}(x)$ and $P_{2,k}(x)$, respectively. We measured the computing time by repeating each computation 10 times and calculating the average for one iteration.

We see that computing time of one iteration of **DKAreal** is approximately equal to the one-half of that of **DKA** for $k = 0$, and it becomes smaller as k increases. We also see that computing time of one iteration of **DKreal** increases as k increases, and it becomes greater than that of **DKAreal** for relatively large value of k.

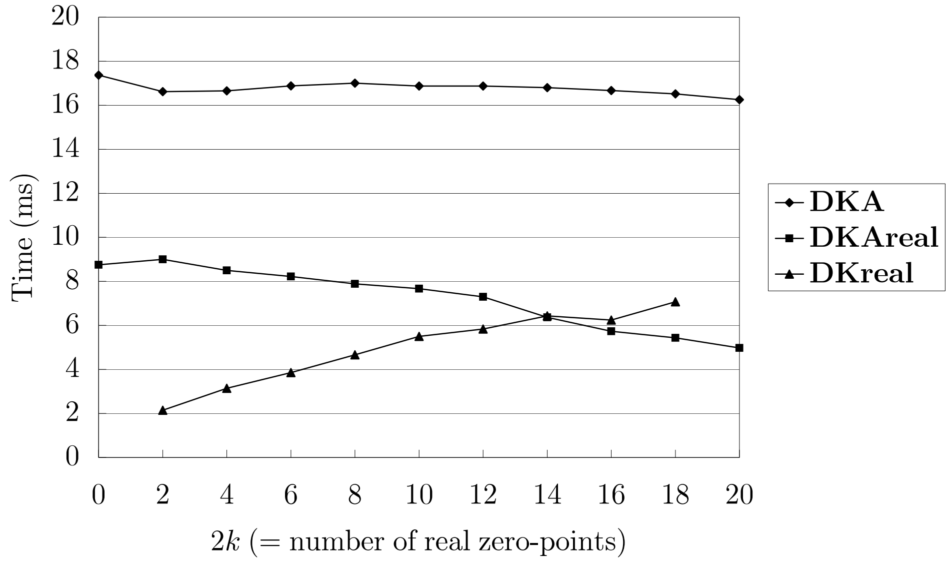

Figure 2.1: Average computing time of one iteration for $P_{1,k}(x)$.

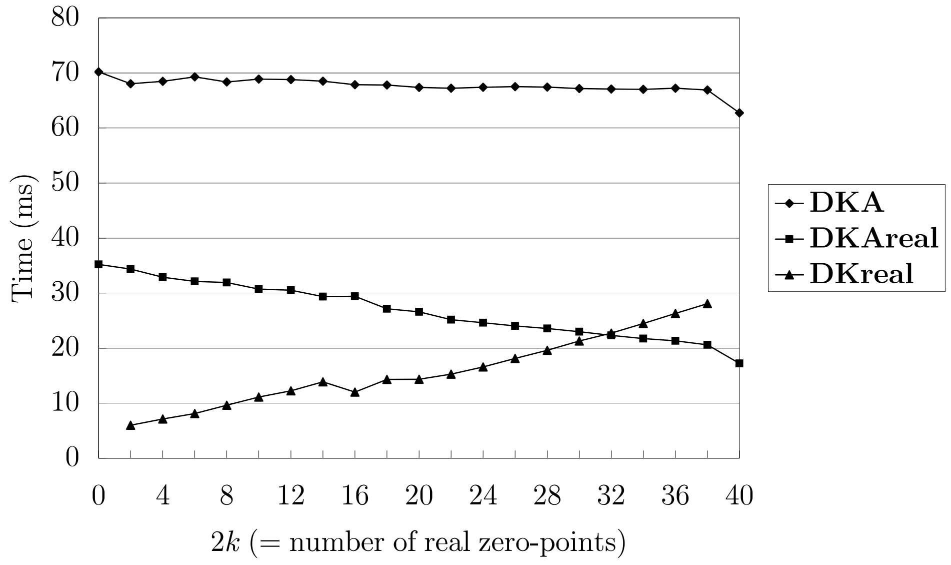

Figure 2.2: Average computing time of one iteration for $P_{2,k}(x)$.

2.4.2 Usefulness in a Special Case

We next consider the case that we know rather good approximations of real zero-points of target polynomial. Such a case happens quite often practically. For example, let us consider drawing an algebraic function on the real plane, determined as the solution of a bivariate polynomial equation $P_3(x, y) = 0$. Let $P_3(x, y)$ be

$$\begin{aligned} P_3(x,y) =& \frac{93392896}{15625}x^6 + \left(\frac{94359552}{625}y^2 + \frac{91521024}{625}y - \frac{249088}{125}\right)x^4 \\ &+ \left(\frac{1032192}{25}y^4 - 36864y^3 - \frac{7732224}{25}y^2 - 207360y + \frac{770048}{25}\right)x^2 \\ &+ (65536y^6 + 49152y^5 - 135168y^4 - 72704y^3 + 101376y^2 \\ &+ 27648y - 27648). \end{aligned}$$

Throughout this paper, by $\deg_x(P_3(x, y))$ we denote the degree of $P_3(x, y)$ with respect to x. Function $P_3(x, y) = 0$ has eight multiple points of multiplicity 2 in the real plane; two are on lines which are tangent with horizontal line $y = \sqrt{2}$, two are "cusps" at $(0, -1)$ and $(0, 3/4)$, and the other four are isolated points at

$$y = \begin{cases} -\frac{380}{351} \ (\simeq -1.0997150997151), \\ -\frac{41}{76} \ (\simeq -0.53947368421053). \end{cases}$$

Suppose that we draw the graph in the interval $-1 \leqslant y \leqslant \sqrt{2}$. We first divide the y-axis of drawing area into the following three intervals as

$$\begin{cases} I_1 = [-1.0, -0.53947368421053], \\ I_2 = [-0.53947368421053, 0.75], \\ I_3 = [0.75, 1.414213562]. \end{cases}$$

In each interval I_j $(1 \leqslant j \leqslant 3)$, we calculate the graph of $P_3(x, y) = 0$ on the real plane as follows.

1. Let $y_{j,0} = (\max I_j + \min I_j)/2$ (the middle point of I_j) and $\delta_j = (\max I_j - \min I_j)/L$, where L is an even integer such as $L \simeq 10$, and divide I_j into L subintervals of width δ_j.
2. Calculate the zero-points of univariate polynomial $P_3(x, y_{j,0})$.
3. Let $y_{j,\pm k} = y_{j,0} \pm k\delta_j$ $(k = 1, \ldots, L/2 - 1)$ and calculate the zero-points of polynomials $P_3(x, y_{j,+k})$ and $P_3(x, y_{j,-k})$, respectively. In order to calculate the zero-points of $P_3(x, y_{j,\pm(k+1)})$, use the zero-points of $P_3(x, y_{j,\pm k})$ as the initial values.
4. Connect the real zero-points computed smoothly.

Note that, in Step 3 (the main step), we calculate the real zero-points with initial values which are good approximations. We measured the computing time by repeating each computation for 100 times. Tables 2.2, 2.3 and 2.4 show the result of computations of zero-points in intervals I_1, I_2 and I_3, respectively.

Except for the computation at $y = y_{j,0}$, the initial values are apart from true zero-points by about 0.01 to 0.1. In intervals I_1 and I_2, the number of the real zero-points is two which is much smaller than $\deg_x(P_3(x, y))$. The number of iteration is not much different among the three algorithms. However, the computing time of

Step	Value	The number of iteration			Computing time (100 times, ms.)		
	of y	**DKreal**	**DKAreal**	**DKA**	**DKreal**	**DKAreal**	**DKA**
$y_{1,-1}$	-0.815789	5	6	6	210	450	1040
$y_{1,-2}$	-0.861842	5	6	6	230	460	1030
$y_{1,-3}$	-0.907895	6	6	6	240	470	1060
$y_{1,-4}$	-0.953947	6	7	7	240	520	1180
$y_{1,1}$	-0.723684	5	5	5	210	400	860
$y_{1,2}$	-0.677632	5	5	5	210	400	870
$y_{1,3}$	-0.631579	5	6	6	210	470	1090
$y_{1,4}$	-0.585526	4	6	6	180	470	1020

Table 2.2: Result of computation in interval I_1, centered at $y_{1,0} = -0.769737$.

Step	Value	The number of iteration			Computing time (100 times, ms.)		
	of y	**DKreal**	**DKAreal**	**DKA**	**DKreal**	**DKAreal**	**DKA**
$y_{2,-1}$	-0.023684	5	8	8	270	630	1380
$y_{2,-2}$	-0.152632	4	6	6	190	490	1040
$y_{2,-3}$	-0.281579	5	6	6	210	520	1050
$y_{2,-4}$	-0.410526	4	6	6	220	530	1050
$y_{2,1}$	0.234211	5	110	80	210	8400	10230
$y_{2,2}$	0.363158	5	7	7	260	550	1220
$y_{2,3}$	0.492105	5	6	6	210	460	1060
$y_{2,4}$	0.621053	5	6	6	260	500	1050

Table 2.3: Result of computation in interval I_2, centered at $y = 0.105263$.

Step	Value	The number of iteration			Computing time (100 times, ms.)		
	of y	**DKreal**	**DKAreal**	**DKA**	**DKreal**	**DKAreal**	**DKA**
$y_{3,-1}$	1.032225	5	6	6	430	450	1060
$y_{3,-2}$	0.961669	6	6	6	510	450	1070
$y_{3,-3}$	0.891113	6	6	6	530	450	1050
$y_{3,-4}$	0.820556	7	7	7	600	520	1200
$y_{3,1}$	1.173338	5	6	10	420	450	1700
$y_{3,2}$	1.243895	5	5	5	410	390	1970
$y_{3,3}$	1.314451	5	5	5	420	390	870
$y_{3,4}$	1.385007	5	5	5	420	390	890

Table 2.4: Result of computation in interval I_3, centered at $y = 1.102782$.

DKreal is approximately equal to one-fourth of that of **DKA**, and the computing time of **DKAreal** is approximately equal to one-half of that of **DKA**. We see that this result is almost the same as the result in Section 2.4.1.

At $y_{2,1} = 0.234211$, **DKA** and **DKAreal** spent much times with many iterations. This is due to the existence of complex singular point in the neighborhood of $y_{2,1}$. In this case **DKreal** is much faster and more stable than **DKA** and **DKAreal**.

In interval I_3, the number of the real zero-points is increased to four and the computing time of **DKreal** also increases so that it becomes a little greater than that of **DKAreal**. This is due to that the number of real zero-points of polynomial is close to $\deg_x(P_3(x,y))$, as our computing time analysis shows. In this case **DKAreal** shows a better performance than **DKreal**.

We see that, in the case that we know good approximations of real zero-points of target polynomial, **DKreal** is the most efficient method among three if the number of the real zero-points is relatively small, and **DKAreal** is efficient if the number of the real zero-points is relatively large.

2.4.3 Comparison with the Newton's Method and Weakness

We have also implemented the Newton's method on GAL. The initial value for the Newton's method is determined by the maximum of the absolute values calculated by formula (2.3). We compared the performance of the Newton's method, **DKreal**, **DKAreal** and **DKA** by the following polynomials.

$$\begin{aligned}
P_4(x) &= (x^2-3.2)(x^2+0.0001),\\
P_5(x) &= (x^2-3.2)\{(x-3.5)^2+0.01\},\\
P_6(x) &= (x^2-3.2)\{(x-9.57)^2+0.01\},\\
P_7(x) &= (x-5.1378890651692)(x-5.138)(x-0.057396173363021)(x-0.05738),\\
P_8(x) &= (x-5.1378890651692)(x-5.1378890630871)\{(x+3.457)^2+17.64567\},\\
P_9(x) &= (x-12.2)(x-19.666666666666)^3.
\end{aligned}$$

Note that polynomials $P_4(x)$, $P_5(x)$ and $P_6(x)$ have complex zero-points which are close to the real axis, $P_7(x)$ and $P_8(x)$ have two real close zero-points, and $P_9(x)$ has real multiple zero-points of multiplicity 3.

Table 2.5 shows the result of computations, where blanks in the table mean that the computation did not converge within 100 iterations. For $P_4(x)$, the Newton's method converges while **DKreal** does not, and we get the opposite result for $P_6(x)$. Furthermore, for $P_5(x)$, neither the Newton's method nor **DKreal** converge within 100 iterations. As these examples show, if we compute only real zero-points, the computation often will not converge often when the given polynomial has complex zero-points with small imaginary parts.

Tables 2.6 and 2.7 show accuracies of calculated zero-points of $P_7(x)$ and $P_8(x)$, respectively, where inaccurate digits are underlined and we show only the real parts for **DKA**. For each zero-point computed, its accuracy is not much different among the algorithms.

For $P_9(x)$, the Sturm's method fails to calculate correct number of the real zero-points, thus we gave correct number of the real zero-points to methods except for **DKA** to calculate real zero-points. Table 2.8 shows accuracies of calculated zero-points of $P_9(x)$. The Newton's method fails to give all the real zero-points, while **DKreal**, **DKAreal** and **DKA** succeed, by the following reason. In the Newton's method, we perform *deflation* of polynomial after calculating each zero-point.

Poly-	Computing time (100 times, ms.)			
nomial	Newton	**DKreal**	**DKAreal**	**DKA**
P_4	370	—	690	1170
P_5	—	—	1730	1000
P_6	—	2100	—	1090
P_7	670	1280	840	2000
P_8	570	1290	1080	2130
P_9	—	1330	870	1760

Table 2.5: Computing time of four methods; 100 times of computations for each method. "—" means that the method failed to converge within 100 iterations.

Method	Real zero-points			
Newton	$5.1378890651\underline{343}$	$5.1380000000\underline{349}$	$0.05739617336\underline{2996}$	$0.057380000000\underline{25}$
DKreal	$5.1378890651\underline{439}$	$5.1380000000\underline{376}$	$0.05739617336\underline{2978}$	$0.057380000000\underline{71}$
DKAreal	$5.1378890651\underline{439}$	$5.1380000000\underline{376}$	$0.05739617336\underline{2978}$	$0.057380000000\underline{71}$
DKA	$5.1378890651\underline{302}$	$5.1380000000\underline{311}$	$0.05739617336\underline{2958}$	$0.057380000000\underline{88}$

Table 2.6: Accuracies of computed real zero-points of $P_7(x)$.

Method	Real zero-points	
Newton	$5.1378891\underline{215841}$	$5.1378890\underline{066722}$
DKreal	$5.1378891\underline{318238}$	$5.137888\underline{9991241}$
DKAreal	$5.1378891\underline{189296}$	$5.1378890\underline{066883}$
DKA	$5.13788908\underline{57556}$	$5.1378890\underline{413680}$

Table 2.7: Accuracies of computed real zero-points of $P_8(x)$.

Method	Real zero-points			
Newton	—	—	—	$19.666\underline{985842244}$
DKreal	12.200000000000	$19.666\underline{327629088}$	$19.666\underline{713528554}$	$19.666\underline{999258452}$
DKAreal	12.200000000000	$19.666\underline{327629088}$	$19.666\underline{713528554}$	$19.666\underline{999258452}$
DKA	12.200000000000	$19.666\underline{346248713}$	$19.666\underline{732891919}$	$19.666\underline{921927080}$

Table 2.8: Accuracies of computed real zero-points of $P_9(x)$. Newton's method failed to calculate all the real zero-points.

Hence, if one of multiple zero-points is calculated with a low accuracy, then coefficients in the deflated polynomial become inaccurate and the Newton's method does not converge. On the other hand, the other methods calculate all the real zero-points including multiple zero-points, although accuracies of approximations for the multiple zero-points become low.

We see that, so long as correct number of the real zero-points is given, **DKreal** and **DKAreal** give approximations for all the real zero-points even if there exist multiple zero-points.

2.5 SUMMARY

In this chapter, we have proposed two methods for calculating the real zero-points of a real univariate polynomial, on the basis of the Durand-Kerner method. In the first method we calculate the real and the complex zero-points separately by a new setting of initial values for the Durand-Kerner method, and in the second method we calculate only the real zero-points of the polynomial. Furthermore, on the basis of the Smith's theorem, we have given a method to estimate the errors of the approximations in the second method.

As we have seen in the experiments, algorithm **DKreal** has an advantage over the Newton's method in that it calculates upper bounds of the errors of approximations simultaneously. However, the experiments also show that, in the case that there exist complex zero-points with very small imaginary parts, both the Newton's method and **DKreal** may not converge correctly. Therefore, in application, we had better use **DKA** if we have no information on the zero-points. In the case we already know good approximations of real zero-points and there does not exist complex zero-points with very small imaginary parts, we had better choose **DKreal** if the number of real zero-points is relatively small, or **DKAreal** if the number of real zero-points is relatively large.

In this chapter, we have assumed that the number of the real zero-points can be calculated by the Sturm's method. However, this assumption falls down in general if the given polynomial has multiple or close zero-points. How to calculate the number of real zero-points correctly for polynomial with large error terms will be discussed in the next chapter.

3 "APPROXIMATE ZERO-POINTS" OF REAL UNIVARIATE POLYNOMIAL WITH LARGE ERROR TERMS

In traditional computer algebra on polynomials, we usually assume that the coefficients of polynomials are given rigorously by integers, rational numbers, or algebraic numbers, and that manipulation on the polynomials is also exact. However, in many practical applications or real-world problems, the coefficients contain errors; that is, polynomials have "error terms." In such cases, many of the traditional algorithms in computer algebra break down.

In this chapter, we consider the real zero-points of a real univariate polynomial with error terms, or "approximate polynomial," where the coefficients of error terms can be much larger than the machine epsilon ε_M of the floating-point arithmetic. In fact, even if the initial errors in coefficients are as small as ε_M, the errors can become much larger than ε_M after the calculation. Furthermore, in approximate algebraic calculation, we handle polynomials with perturbed terms that are much larger than ε_M in general.

If a polynomial $P(x)$ has error terms, we cannot draw the graph of function $y = P(x)$ exactly; all we can draw is the "existence domain" of $P(x)$, or the domain in which values of $P(x)$ can exist. Similarly, in such a case, the positions of its zero-points cannot be determined exactly; all we can handle is the domains in which zero-points can exist. Therefore, we introduce a concept of an "approximate real zero-point" by defining a minimal interval outside of which no real zero-points can exist. Although the existence domains of real zero-points can be calculated rigorously, we propose methods for calculating them approximately and efficiently by using the Smith's theorem on the error bounds of zero-points of a polynomial [51].

Next, we consider calculation of the number of real zero-points of an approximate polynomial by the Sturm's method. If all the zero-points are single and well separated, the number of real zero-points is definite unless some error term is quite large, although the positions of zero-points are changed by the error terms. However, in the calculation of the Sturm sequence, the leading coefficient of some element may become too small to determine whether it is equal to zero or not. Since the sign of the leading coefficient in the Sturm sequence is essential in determining the number of real zero-points, this is a serious problem. Our answer to it is that, under some conditions, we may discard the small leading term and continue further calculation of the Sturm sequence. Shirayanagi and Sekigawa also attacked this problem [50], and proposed an interval arithmetic method with zero rewriting. We will investigate the Sturm sequence with interval coefficients in Section 3.4.

The rest of this chapter is organized as follows. In Section 3.1, we investigate the existence domains of the values of a real approximate polynomial, then define an approximate real zero-point. In Section 3.2, we propose a practical method for calculating the existence domains of the zero-points of an approximate polynomial. In Section 3.3, on the assumption that the polynomial does not have multiple or close zero-points, we derive a sufficient condition for the number of real zero-points not to be changed by error terms. In Section 3.4, we propose and investigate several methods for checking the effect of the error terms of a given polynomial on the Sturm sequence.

3.1 APPROXIMATE POLYNOMIALS AND APPROXIMATE REAL ZERO-POINTS

Let $P(x)$ be a univariate polynomial with real coefficients, given as

$$P(x) = c_n x^n + \cdots + c_0 x^0, \tag{3.1}$$

and let $\tilde{P}(x)$ be a real univariate polynomial such that

$$\tilde{P}(x) = P(x) + \Delta(x), \tag{3.2}$$

where $\Delta(x)$ represents the sum of real "error terms," that is, a polynomial with unknown small real coefficients. Thus, we know neither $\tilde{P}(x)$ nor $\Delta(x)$; what we know usually is an upper bound for each coefficient in $\Delta(x)$. Representing $\Delta(x)$ as

$$\Delta(x) = \delta_{n-1} x^{n-1} + \cdots + \delta_0 x^0,$$

we assume that we know upper bounds $\varepsilon_{n-1}, \dots, \varepsilon_0$, given as

$$|\delta_i| \leqslant \varepsilon_i, \tag{3.3}$$

for $i = n-1, \dots, 0$. Throughout this chapter, we write

$$\tilde{P}(x \mid \delta_i = \varepsilon_i' \, (i = n-1, \dots, 0))$$

to denote that the values of $\delta_{n-1}, \dots, \delta_0$ in $\tilde{P}(x)$ are specified as $\delta_i = \varepsilon_i'$ for $i = n-1, \dots, 0$.

3.1.1 The Existence Domain of Values of $\tilde{P}(x)$

Assume that the variable x is fixed to x_0 and that $\delta_{n-1}, \dots, \delta_0$ are changed continuously under the restrictions in Formula (3.3); the value of $\tilde{P}(x_0)$ moves continuously inside an interval. By changing x_0 in $\mathbf{R}$, we will have the minimal domain outside of which there is no possibility of the existence of the value of $\tilde{P}(x)$.

Definition 3.1 (The existence domain). Let x_0 be a real number and δ_i move continuously in the whole interval $[-\varepsilon_i, \varepsilon_i]$ for $i = 0, \dots, n-1$. Define $P_U(x_0)$ and $P_L(x_0)$ as

$$P_U(x_0) = \max_{\substack{\delta_i \in [-\varepsilon_i, \varepsilon_i] \\ i=0,\dots,n-1}} \tilde{P}(x_0), \quad P_L(x_0) = \min_{\substack{\delta_i \in [-\varepsilon_i, \varepsilon_i] \\ i=0,\dots,n-1}} \tilde{P}(x_0).$$

By changing the value x_0 in $\mathbf{R}$, we obtain a domain

$$\{[P_L(x), P_U(x)] \mid x \in \mathbf{R}\}. \tag{3.4}$$

We call this domain the "existence domain of $\tilde{P}(x)$." □

The existence domain of $\tilde{P}(x)$ can be specified rigorously by using $P(x)$.

Lemma 3.1. *Let the value of δ_i in $\tilde{P}(x)$ be changed continuously within the range $[-\varepsilon_i, \varepsilon_i]$, while the values of δ_j's ($j \neq i$) are fixed, and, for each real value of x, define $P_{U_i}(x)$ and $P_{L_i}(x)$ as*

$$P_{U_i}(x) = \max_{\delta_i \in [-\varepsilon_i, \varepsilon_i]} \tilde{P}(x), \quad P_{L_i}(x) = \min_{\delta_i \in [-\varepsilon_i, \varepsilon_i]} \tilde{P}(x).$$

Then, we have

$$P_{U_i}(x) = \begin{cases} \tilde{P}(x \mid \delta_i = \varepsilon_i) & \text{if } x \geqslant 0 \text{ or } i \text{ is even,} \\ \tilde{P}(x \mid \delta_i = -\varepsilon_i) & \text{if } x \leqslant 0 \text{ and } i \text{ is odd,} \end{cases}$$

$$P_{L_i}(x) = \begin{cases} \tilde{P}(x \mid \delta_i = -\varepsilon_i) & \text{if } x \geqslant 0 \text{ or } i \text{ is even,} \\ \tilde{P}(x \mid \delta_i = \varepsilon_i) & \text{if } x \leqslant 0 \text{ and } i \text{ is odd.} \end{cases}$$

Furthermore, for any real value x_0, $\tilde{P}(x_0)$ moves all the points inside $[P_{L_i}(x_0), P_{U_i}(x_0)]$.

Proof. Let x_0 be any real number. We see that $-\varepsilon_i|x_0|^i \leqslant \delta_i|x_0|^i \leqslant \varepsilon_i|x_0|^i$, and since $\delta_i|x_0|^i$ moves all the points inside $[-\varepsilon_i|x_0|^i, \varepsilon_i|x_0|^i]$, we obtain the lemma. □

This lemma directly leads us to the following theorem.

Theorem 3.2. *Let the polynomials $P(x)$ and $\tilde{P}(x)$ be as above. Then, the functions $P_U(x)$ and $P_L(x)$ in Formula (3.4) are given as follows.*

$$P_U(x) = \begin{cases} \tilde{P}(x \mid \delta_i = \varepsilon_i \ (i = n-1, \ldots, 0)) & \text{for } x \geqslant 0, \\ \tilde{P}(x \mid \delta_i = (-1)^i \varepsilon_i \ (i = n-1, \ldots, 0)) & \text{for } x < 0, \end{cases}$$

$$P_L(x) = \begin{cases} \tilde{P}(x \mid \delta_i = -\varepsilon_i \ (i = n-1, \ldots, 0)) & \text{for } x \geqslant 0, \\ \tilde{P}(x \mid \delta_i = (-1)^{i+1} \varepsilon_i \ (i = n-1, \ldots, 0)) & \text{for } x < 0. \end{cases}$$

Furthermore, for any real number x_0, the values of $\tilde{P}(x_0)$ move all the points inside $[P_L(x_0), P_U(x_0)]$. □

3.1.2 Approximate Real Zero-points and their Existence Domains

We first define a concept of "approximate real zero-points" and their existence domains.

Definition 3.2 (Approximate real zero-point)**.** A real number ζ is an "approximate real zero-point of $\tilde{P}(x)$" if there exist numbers $\varepsilon_i' \in [-\varepsilon_i, \varepsilon_i]$ for $i = n-1, \ldots, 0$, such that $\tilde{P}(\zeta \mid \delta_i = \varepsilon_i' \ (i = n-1, \ldots, 0)) = 0$. Let $[\zeta_{1,1}, \zeta_{1,2}], \ldots, [\zeta_{r,1}, \zeta_{r,2}]$, with $\zeta_{1,1} \leqslant \zeta_{1,2} < \cdots < \zeta_{r,1} \leqslant \zeta_{r,2}$, be the set of all the approximate real zero-points of $\tilde{P}(x)$. Then, for $i = 1, \ldots, r$, we call each interval $[\zeta_{i,1}, \zeta_{i,2}]$ an "existence domain" of the approximate real zero-point of $\tilde{P}(x)$. □

Theorem 3.2 tells us that the existence domains of all the approximate real zero-points can be specified rigorously by drawing graphs of $P_L(x)$ and $P_U(x)$. Suppose $[\zeta_1, \zeta_2]$ is an existence domain of an approximate real zero-point. Since ζ_1 and ζ_2 are real zero-points of $P_U(x)$ and/or $P_L(x)$, and since $P_L(x_0) < P_U(x_0)$ for any real number x_0, the graphs of $P_L(x)$ and $P_U(x)$ around this interval can be classified into one of the following four cases:

(a) $P_L(\zeta_1) = P_U(\zeta_2) = 0$, $P_L(x) < 0$ for $\zeta_1 < x \leqslant \zeta_2$, $P_U(x) > 0$ for $\zeta_1 \leqslant x < \zeta_2$, and there exists $\delta > 0$ such that $P_L(\zeta_1 - x) > 0$ and $P_U(\zeta_2 + x) < 0$ for any $x \in [0, \delta]$,

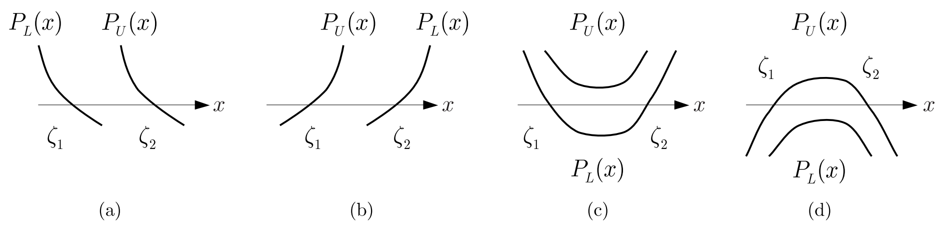

Figure 3.1: Existence domain of an approximate real zero-point.

(b) $P_U(\zeta_1) = P_L(\zeta_2) = 0$, $P_U(x) > 0$ for $\zeta_1 < x \leqslant \zeta_2$, $P_L(x) < 0$ for $\zeta_1 \leqslant x < \zeta_2$, and there exists $\delta > 0$ such that $P_U(\zeta_1 - x) < 0$ and $P_L(\zeta_2 + x) > 0$ for any $x \in [0, \delta]$,

(c) $P_L(\zeta_1) = P_L(\zeta_2) = 0$, $P_U(x) > 0$ for $\zeta_1 \leqslant x \leqslant \zeta_2$, $P_L(x) < 0$ for $\zeta_1 < x < \zeta_2$, and there exists $\delta > 0$ such that $P_L(\zeta_1 - x) > 0$ and $P_L(\zeta_2 + x) > 0$ for any $x \in [0, \delta]$,

(d) $P_U(\zeta_1) = P_U(\zeta_2) = 0$, $P_L(x) < 0$ for $\zeta_1 \leqslant x \leqslant \zeta_2$, $P_U(x) > 0$ for $\zeta_1 < x < \zeta_2$, and there exists $\delta > 0$ such that $P_U(\zeta_1 - x) < 0$ and $P_U(\zeta_2 + x) < 0$ for any $x \in [0, \delta]$.

Fig. 3.1 illustrates these four cases conceptually. Cases (a) and (b) usually correspond to a single zero-point, while cases (c) and (d) correspond to multiple zero-points.

We now give a simple example of approximate real zero-points and their existence domains. We will see that one of the existence domains is fairly wide, which indicates that the concept of approximate zero-point is indispensable in handling polynomials with error terms.

Example 3.1. Let $F(x, y)$ be

$$F(x, y) = x^3 - x^2 + y^2.$$

We calculate a singular point of $F(x, y)$ with approximate arithmetic of precision $\varepsilon_M = 1.0 \times 10^{-6}$. First, let us calculate the discriminant $R(y)$ of $F(x, y)$ with respect to x as

$$R(y) = \mathrm{res}(F, dF/dx) = 27y^4 - 4y^2.$$

Note that $R(y)$ has zero-points at $y = 0$ and $\pm 2\sqrt{3}/9$. Assume that we have calculated the value of $y = 2\sqrt{3}/9$ approximately as 0.384900. (Note that if $\deg(R) \geqslant 5$ then use of approximate arithmetic is necessary in general to solve $R(y) = 0$.) Let $P(x)$ and $\tilde{P}(x)$ be

$$P(x) = x^3 - x^2 + (0.384900)^2, \quad \tilde{P}(x) = P(x) + \delta_0,$$

where $|\delta_0| \leqslant 1.0 \times 10^{-6}$, and let us calculate the approximate real zero-points of $\tilde{P}(x)$. From Theorem 3.2, we have

$$P_U(x) = x^3 - x^2 + 0.148149, \quad P_L(x) = x^3 - x^2 + 0.148147.$$

$P_U(x)$ has a real zero-point at $x \simeq -0.333334$, and $P_L(x)$ has real zero-points at $x \simeq -0.333332$, 0.665595, and 0.667738. From Definition 3.2, the existence domains of approximate real zero-points of $\tilde{P}(x)$ are intervals $[-0.333334, -0.333332]$, and $[0.665595, 0.667738]$. Therefore, with an approximate arithmetic of precision $\varepsilon_M = 1.0 \times 10^{-6}$, the singular point (x_0, y_0) of $F(x, y)$ can be specified only vaguely as $y_0 \in [0.384899, 0.384901]$ and $x_0 \in [0.665595, 0.667738]$.

3.2 BOUNDING EXISTENCE DOMAINS BY USING THE SMITH'S THEOREM

Although we have defined rigorously the existence domain of only real zero-points, we present in this section a method for bounding the existence domains of both real and complex zero-points by means of discs in the complex plane, because the method is common to both of them.

A key to bounding existence domains is the Smith's theorem: see Theorem 2.1 (page 6).

3.2.1 Single Zero-points

Without loss of generality, we assume that P and $\tilde{P}$ are monic. Let $\zeta_1, \dots, \zeta_n$ and $\tilde{\zeta}_1, \dots, \tilde{\zeta}_n$ be the zero-points of $P(x)$ and $\tilde{P}(x)$, respectively, as

$$P(x) = (x - \zeta_1)(x - \zeta_2) \cdots (x - \zeta_n),$$
$$\tilde{P}(x) = (x - \tilde{\zeta}_1)(x - \tilde{\zeta}_2) \cdots (x - \tilde{\zeta}_n).$$

First, we consider the case in which ζ_1 is a single zero-point such that $|\zeta_1 - \zeta_j| \gg \varepsilon_M$ for $j = 2, \dots, n$. Let $z_1, \dots, z_n$ be approximate values for $\zeta_1, \dots, \zeta_n$, respectively. (Actually, we may determine $z_1, \dots, z_n$ by solving equation $P(x) = 0$ numerically, and thus approximately, with accuracy ε_M.) Using Theorem 2.1, we can formally calculate the domain that contains $\tilde{\zeta}_1$ in $\mathbf{C}$, as follows. Let R_1 be

$$R_1 = n \cdot \frac{|\tilde{P}(z_1)|}{\left|\prod_{j=2}^{n}(z_1 - z_j)\right|}, \tag{3.5}$$

then $\tilde{\zeta}_1$ is contained in the disc of radius R_1 with its center at z_1. Although we cannot calculate $\tilde{P}(z_1)$ explicitly, we have

$$|\tilde{P}(z_1)| \leqslant |P(z_1)| + |\Delta(z_1)| \leqslant |P(z_1)| + \sum_{j=0}^{n-1} \varepsilon_j |z_1|^j.$$

Therefore, R_1 is bounded as

$$R_1 \leqslant n \cdot \frac{|P(z_1)| + \sum_{j=0}^{n-1} \varepsilon_j |z_1|^j}{\left|\prod_{j=2}^{n}(z_1 - z_j)\right|}. \tag{3.6}$$

In ordinary numerical computation, we calculate an error bound by the above formula with $\varepsilon_j = 0$, which gives a good estimate such that the magnitude of the error bound is only several times larger than the true error. Therefore, we expect that the above formula gives a good bound.

3.2.2 Multiple or Close Zero-points

Next, we consider the case of multiple or close zero-points. Without loss of generality, let $\zeta_1 \simeq \cdots \simeq \zeta_m$ ($m \leqslant n$) and assume that $\zeta_{m+1}, \dots, \zeta_n$ satisfy $|\zeta_j - \zeta_1| \gg \sqrt[m]{\varepsilon_M}$ for $j = m+1, \dots, n$. In this case, we cannot apply Formula (3.6) directly, for the following reason. Let $z_1, \dots, z_n$ be the same as above and assume that we have calculated them by a numerical method. Then, $z_1, \dots, z_m$ usually satisfy $|z_j - z_1| \simeq \sqrt[m]{\varepsilon_M}$ for $j = 2, \dots, m$; thus, in Formula (3.6), we have

$$\left|\prod_{j=2}^{n}(z_1 - z_j)\right| \simeq \varepsilon_M \left|\prod_{j=m+1}^{n}(z_1 - z_j)\right|.$$

Therefore, if $|\Delta(z_i)| \gg \varepsilon_M$, an upper bound calculated by Formula (3.6) will be an overestimate.

We determine $z_1, \dots, z_m$ so that the radius R_1 in Formula (3.5) becomes as small as possible. (The determination method is the same as that described in the literature; for example, see Iri [21]; the only difference is that our setting of error terms is different from the conventional ones.) We express $P(x)$ as

$$P(x) = (x - \zeta_1) \cdots (x - \zeta_m) \cdot Q(x).$$

From our assumption, we have

$$Q(z_1) = \prod_{j=m+1}^{n}(z_1 - \zeta_j) \simeq \prod_{j=m+1}^{n}(z_1 - z_j);$$

thus R_1 defined by Formula (3.5) can be approximated as

$$R_1 = n \cdot \frac{\left|\prod_{j=1}^{n}(z_1 - \zeta_j) + \Delta(z_1)\right|}{\left|\prod_{j=2}^{n}(z_1 - z_j)\right|} \simeq n \cdot \frac{\left|\prod_{j=1}^{m}(z_1 - \zeta_j) + \dfrac{\Delta(z_1)}{Q(z_1)}\right|}{\left|\prod_{j=2}^{m}(z_1 - z_j)\right|}. \tag{3.7}$$

If $z_1, \dots, z_m$ are distributed equally on a disc of radius r with its center at $(\zeta_1 + \cdots + \zeta_m)/m$, we have

$$\left|\prod_{j=1}^{m}(z_1 - \zeta_j)\right| \approx r^m, \quad \left|\prod_{j=2}^{m}(z_1 - z_j)\right| = mr^{m-1},$$

and Formula (3.7) can be evaluated as

$$R_1 \simeq n \cdot \frac{r^m + C}{mr^{m-1}},$$

where $C = |\Delta(z_1)/Q(z_1)|$. We can almost minimize the magnitude of R_1 by setting r as

$$r = \sqrt[m]{(m-1)C}. \tag{3.8}$$

With the above consideration, we calculate an upper bound for R_1 as follows:

1. Calculate r from Formula (3.8).
2. Let $\beta = (\zeta_1 + \cdots + \zeta_m)/m$ and

$$z_j = \beta + r\exp(2\pi j \boldsymbol{i}/m)$$

for $j = 1, \dots, m$. The approximate values $z_1, \dots, z_m$ are distributed equally on a disc of radius r with its center at β.

3. Substitute $z_1, \dots, z_m$ into Formula (3.6) to obtain a rigorous bound of R_1.

Remark 3.1. While research for calculating existence domains of zero-points has already been carried out, especially in control theory, and some results including Kharitonov's Theorem [28] and the Edge Theorem [3] have been obtained, our method is relatively simple and efficient for practical computation. Recent progress of the research includes calculating existence domains of real and complex zero-points of an interval polynomial which is a generalization of our definition of approximate polynomial ([48], [49]).

3.3 CALCULATING THE NUMBER OF REAL ZERO-POINTS OF A REAL APPROXIMATE POLYNOMIAL

If a real approximate polynomial has multiple or close zero-points, they may change significantly, or some real zero-points may become complex, when the coefficients are changed slightly. Therefore, it is not adequate to count the number of real zero-points of a real approximate polynomial that may have multiple or close zero-points. On the other hand, if a polynomial has only single zero-points, the number of its real zero-points rarely changes, although their positions may change considerably, when the coefficients are changed slightly. In this section, we focus on calculating the number of real zero-points of a real approximate polynomial containing only single zero-points.

3.3.1 Sufficient Condition for Fixing the Number of Real Zero-points

We first derive a sufficient condition for asserting that $P(x)$ and $\tilde{P}(x)$ have the same number of real zero-points.

Theorem 3.3. *Let $P(x)$ and $\tilde{P}(x)$ be as in Formulas (3.1) and (3.2), respectively. Then, the number of real zero-points of $\tilde{P}(x)$ is the same as that of $P(x)$ if the discriminant of $\tilde{P}$, or $\mathrm{res}(\tilde{P}, d\tilde{P}/dx)$ does not become zero for any values $\delta_{n-1}, \dots, \delta_0$ satisfying Formula (3.3).*

Proof. As the coefficients of $\tilde{P}(x)$ change continuously, the number of real zero-points of $\tilde{P}(x)$ changes only if there exist $\delta_i \in [-\varepsilon_i, \varepsilon_i]$ for $i = 0, \dots, n-1$ such that $\tilde{P}(x)$ has real multiple zero-points. Its contraposition shows the validity of the theorem. □

Theorem 3.3 tells us that we can calculate the number of real zero-points of an unknown polynomial $\tilde{P}(x)$ by calculating the number of the real zero-points of $P(x)$, so long as the discriminant $\mathrm{res}(\tilde{P}, d\tilde{P}/dx)$ does not become zero for any values $\delta_{n-1}, \dots, \delta_0$ satisfying Formula (3.3). Therefore, we can check the definiteness of the number of real zero-points by checking whether or not $\mathrm{res}(\tilde{P}, d\tilde{P}/dx)$ becomes zero because of the error terms.

3.3.2 Problem of Small Leading Coefficient in the Sturm Sequence

Below, the leading coefficient and the degree of $P(x)$ are denoted as $\mathrm{lc}(P)$ and $\deg(P)$, respectively. Let $\zeta_{\max}$ be the maximum of the absolute values of real zero-points of $P(x)$.

The p-norm of $P(x)$, with $P(x)$ given in Formula (3.1), is defined as

$$\|P\|_p = \left(\sum_{i=1}^{n} |c_i|^p \right)^{1/p}, \tag{3.9}$$

where $p = 1, 2, \ldots, \infty$. In this chapter, we use the 2-norm for polynomials.

Assuming that $P(x)$ and $\tilde{P}(x)$ satisfy the condition in Theorem 3.3, $\|P\|_2 \simeq 1$, and $\|\tilde{P}\|_2 \simeq 1$, let us consider calculating the number of real zero-points of $\tilde{P}(x)$ by the Sturm's method on $P(x)$. The Sturm's theorem is as follows (for the proof, see Cohen [8], for example):

Theorem 3.4 (Sturm). *Let $P(x)$ be a real square-free polynomial of degree n, and define a polynomial sequence (the Sturm sequence)*

$$(P_0(x), P_1(x), \ldots, P_n(x)) \tag{3.10}$$

as

$$\begin{cases} P_0 = P(x), \quad P_1 = \dfrac{d}{dx}P(x), \\ P_i = -\mathrm{rem}(P_{i-2}, P_{i-1}) \quad \text{for } i = 2, \ldots, n, \end{cases}$$

where $\mathrm{rem}(P_{i-2}, P_{i-1})$ denotes the remainder of P_{i-2} divided by P_{i-1}. For a real number x, let $N(x)$ be the number of sign changes, counting from the left to the right without counting zeros, in the sequence (3.10), and let s and t be real numbers satisfying $s < t$. Then, the number of the real zero-points of P in the interval $[s, t]$ is equal to $N(s) - N(t)$. $\square$

Note that we can calculate the number of all the real zero-points of P by putting $s = -\infty$ and $t = \infty$ in Theorem 3.4. In the following, the zeros of the Sturm sequence and its modifications are not counted as sign changes.

Consider calculation of the Sturm sequence of $P(x)$ by means of floating-point arithmetic. During the calculation, we may encounter the so-called leading coefficient problem as follows.

1. It is hard for us to decide whether or not a very small leading coefficient is equal to zero.

2. The division by a polynomial by a small leading coefficient will cause large cancellation errors in the coefficients of the remainder polynomial.

Let P, s, and t be the same as in Theorem 3.4. A Sturm sequence of P with $P_n \equiv$ (constant) $\neq 0$ has the following properties (for example, see Cohen [8]):

(1) For any real number x, consecutive elements $P_{i-1}(x)$ and $P_i(x)$ do not simultaneously become zero.

(2) If $P_j(x) = 0$ for some j $(1 \leqslant j < n)$ and $x \in \mathbf{R}$, then we have $P_{j-1}(x)P_{j+1}(x) < 0$.

(3) P_n has no real zero-point.

With Property (1), we can calculate the number of sign changes by investigating each P_i separately. Let $P_j(x_j) = 0$ for some $x_j \in \mathbf{R}$; then Property (2) means that P_{j-1} and P_{j+1} have no zero-point in the neighborhood of $x = x_j$. Property (3) is trivial in our case, because $P_n =$ (constant), but it is not trivial for the general Sturm sequence. The above three properties are sufficient for determining the number of real zero-points, and a sequence that has those properties is called a *general* Sturm sequence.

We note that the sign change of $P_j(x)$ at $x = x_j$, $j \geqslant 1$, does not affect the number of sign changes in sequence (3.10); the value of $N(x)$ changes only when the evaluation point x passes a real zero-point of $P_0(x)$ $(= P(x))$. Furthermore, we have the following property of the Sturm sequence.

Lemma 3.5. *Let $P(x)$ and $P_0, \ldots, P_n$ be the same as in Theorem 3.4, and assume that $P_k(x) = 0$ $(1 < k < n)$ at $x = x_{k,1}, \ldots, x_{k,l_k}$, where $l_k < \deg(P_k)$ and $|x_{k,j}| > \zeta_{\max}$ for $j = 1, \ldots, l_k$. Define $P''_k(x)$ as*

$$P''_k(x) = \frac{P_k(x)}{(x - x_{k,1}) \cdots (x - x_{k,l_k})},$$

and let s and t be real numbers satisfying $s < t$. For real number x, let $N(x)$ be the same as in Theorem 3.4, and let $N''_k(x)$ be the numbers of sign changes in the sequence

$$(P_0(x), \ldots, P_{k-1}(x), P''_k(x), P_{k+1}(x), \ldots, P_n(x)).$$

Then we have

$$N''_k(s) - N''_k(t) = N(s) - N(t).$$

That is, $N''_k(s) - N''_k(t)$ is equal to the number of real zero-points of $P(x)$ in the interval $[s, t]$.

Proof. Property (1) assures us that there exists a small positive number δ such that $[x_{k,j_1} - \delta, x_{k,j_1} + \delta] \cap [x_{k,j_2} - \delta, x_{k,j_2} + \delta] = \emptyset$ for $1 \leqslant j_1 < j_2 \leqslant l_k$ and $P_{k\pm1}(x) \neq 0$ for any $x \in [x_{k,j} - \delta, x_{k,j} + \delta]$. We show $N''_k(x) = N(x)$ for any $x \in [x_{k,j} - \delta, x_{k,j} + \delta]$. Consider a case in which $dP_k/dx < 0$ at $x = x_{k,1}$, $P_{k-1}(x_{k,1}) > 0$, and $P_{k+1}(x_{k,1}) < 0$. Property (2) says that the sequence of signs of polynomials $(P_{k-1}(x), P_k(x), P_{k+1}(x))$ at $x = x_{k,1} - \delta$, $x = x_{k,1}$ and $x = x_{k,1} + \delta$ are $(+, +, -)$, $(+, 0, -)$ and $(+, -, -)$, respectively; thus the number of sign changes of the sequence $(P_{k-1}(x), P_k(x), P_{k+1}(x))$ is equal to 1 for any $x \in [x_{k,1} - \delta, x_{k,1} + \delta]$. Now, assume that $P''_k(x) > 0$ for $x \in [x_{k,1} - \delta, x_{k,1} + \delta]$; then the sequence of signs of polynomials $(P_{k-1}(x), P''_k(x), P_{k+1}(x))$ is $(+, +, -)$ for any $x \in [x_{k,1} - \delta, x_{k,1} + \delta]$. Therefore, we have $N''_k(x) = N(x)$ for any $x \in [x_{k,1} - \delta, x_{k,1} + \delta]$. The other cases can be proved similarly. □

Theorem 3.6. *Assume the same hypotheses as in Lemma 3.5, and define a polynomial sequence*

$$(P_0(x), \ldots, P_{k-1}(x), P''_k(x), \ldots, P''_{n''}(x)) \tag{3.11}$$

as

$$\begin{cases} P''_k = \dfrac{P_k(x)}{(x - x_{k,1}) \cdots (x - x_{k,l_k})}, & P''_{k+1} = -\mathrm{rem}(P_{k-1}, P''_k), \\ P''_i = -\mathrm{rem}(P''_{i-2}, P''_{i-1}) & \text{for } i = k+2, \ldots, n'', \end{cases} \tag{3.12}$$

where $\deg(P''_{n''}) = 0$. For a real number x, let $N''(x)$ be the number of sign changes in sequence (3.11), and let s and t be real numbers satisfying $s < t$. Then, the number of real zero-points of $P(x)$ in the interval $[s, t]$ is equal to $N''(s) - N''(t)$.

Proof. From Lemma 3.5, we need not consider $x_{k,1}, \ldots, x_{k,l_k}$ for calculating the number of real zero-points of $P(x)$. Let x_k be any zero-point of P''_k; thus $P_{k-1}(x_k) \neq 0$. Then, we have $P_{k-1}(x_k) \cdot P''_{k+1}(x_k) < 0$ because $-P''_{k+1}(x) = P_{k-1}(x) - Q''_k(x)P''_k(x)$. Repeating this argument for P''_{k+1}, P''_{k+2}, and so on, we see that the new polynomial sequence (3.11) satisfies Properties (1), (2), and (3) described above, and that sequence (3.11) is a general Sturm sequence of $P(x)$. Thus, we can count all the real zero-points of $P(x)$ by using sequence (3.11). □

Remark 3.2. Properties (1), (2), and (3) are sufficient to prove Theorem 3.6, and Lemma 3.5 is unnecessary. We introduced Lemma 3.5 to help the reader to understand what happens when large real zero-points of P_k are removed.

In Theorem 3.6, calculating the general Sturm sequence by using P_k'' in Formula (3.12) is theoretically simple but not practical, because we have to calculate the real zero-points of P_k rigorously. We next show that, if a polynomial has small leading terms, these terms correspond to zero-points of large magnitudes.

Lemma 3.7. *Let $\varepsilon_n, \dots, \varepsilon_{n-s+1}$ be real numbers such that $0 < |\varepsilon_j| \ll 1$, and, without loss of generality, let $Q(x)$ be*

$$Q(x) = \varepsilon_n x^n + \cdots + \varepsilon_{n-s+1} x^{n-s+1} + b_{n-s} x^{n-s} + \cdots + b_0 x^0,$$

where $|b_i| \geqslant 1$ $(i = n-s, \dots, 0)$ for $b_i \neq 0$. Let $x_1, \dots, x_n$ be the zero-points of $Q(x)$ such that $|x_1| < \cdots < |x_n|$. Then, for $j = n-s+1, \dots, n$, we have

$$\lim_{(\varepsilon_n, \dots, \varepsilon_{n-s+1}) \to (0, \dots, 0)} |x_j| = \infty.$$

Proof. Define $Q_I(x)$ as

$$Q_I(x) = x^n \cdot Q(1/x) = \bar{b}_n x^n + \cdots + \bar{b}_0 x^0,$$

and let $\bar{x}_1, \dots, \bar{x}_n$ be the zero-points of $Q_I(x)$ with $|\bar{x}_1| < \cdots < |\bar{x}_n|$. Then we have $\bar{b}_{n-j} = \varepsilon_j$ for $j = n, \dots, n-s+1$ and $\bar{x}_{n-i+1} = 1/x_i$ for $i = 1, \dots, n$. We have $|\bar{x}_i| \to 0$ $(i = n, \dots, n-s+1)$ for $|\bar{b}_{n-j}| \to 0$ $(j = n, \dots, n-s+1)$; thus $|x_i| \to \infty$ for $\varepsilon_j \to 0$. □

Remark 3.3. Although Lemma 3.7 is a limiting case of $(\varepsilon_n, \dots, \varepsilon_{n-s+1}) \to (0, \dots, 0)$ and is sufficient to prove Theorem 3.8, we investigate the location of zero-points of $Q_I(x)$ in our supplementary work ([46], [60, Appendix]).

Theorem 3.6 and Lemma 3.7 lead us to an idea of discarding the small leading terms to calculate a general Sturm sequence in practice. Since the zero-points of $P_k(x)$ are moved slightly by discarding the small leading terms, we must be more careful than in Theorem 3.6.

Theorem 3.8. *Define $P(x)$ and $\tilde{P}(x)$ as in Formulas (3.1) and (3.2), respectively. Let $(P_0 = P(x), P_1 = dP/dx, P_2, \dots, P_i, \dots)$ be the Sturm sequence of $P(x)$ and assume that $P_k(x)$ has small leading terms as*

$$P_k(x) = \varepsilon_{k,n_k} x^{n_k} + \cdots + \varepsilon_{k,n_k-s+1} x^{n_k-s+1} + b_{k,n_k-s} x^{n_k-s} + \cdots + b_{k,0} x^0,$$

where

$$\max\{|\varepsilon_{k,n_k}|, \dots, |\varepsilon_{k,n_k-s+1}|\} \ll \min_{b_{k,j} \neq 0} \{|b_{k,n_k-s}|, \dots, |b_{k,0}|\}.$$

Define a polynomial sequence

$$(P_0(x), \dots, P_{k-1}(x), P_k'(x), \dots, P_{n'}'(x)) \tag{3.13}$$

as

$$\begin{cases} P_k' = b_{k,n_k-s} x^{n_k-s} + \cdots + b_{k,0} x^0, & P_{k+1}' = -\mathrm{rem}(P_{k-1}, P_k'), \\ P_i' = -\mathrm{rem}(P_{i-2}', P_{i-1}') & \text{for } i = k+2, \dots, n', \end{cases}$$

where $\deg(P_{n'}') = 0$. For a real number x, let $N'(x)$ be the number of sign changes in sequence (3.13), and let s and t be real numbers such that $s < -\zeta_{\max}$ and $\zeta_{\max} < t$. Then, if $\tilde{P}(x)$, $P_{k-1}(x)$, and $P_k(x)$ satisfy the following two conditions, the number of real zero-points of $\tilde{P}(x)$ is equal to $N'(s) - N'(t)$:

(1) *The resultant* $\operatorname{res}(\tilde{P}, P_k)$ *does not become zero for any values* $\delta_{n-1}, \ldots, \delta_0$ *satisfying* (3.3) *or when the values of* $\varepsilon_{k,n_k}, \ldots, \varepsilon_{k,n_{k-s+1}}$ *are changed to zero.*

(2) *The resultant* $\operatorname{res}(P_{k-1}, P_k)$ *does not become zero when the values of* $\varepsilon_{k,n_k}, \ldots,$ $\varepsilon_{k,n_{k-s+1}}$ *are changed to zero.*

Proof. Even if $P_k(x)$ has real zero-points whose magnitudes are larger than that of any zero-point of $\tilde{P}(x)$, Lemma 3.7 and condition (1) assure us that these real zero-points will be "safely removed" from $P_k(x)$ by changing the values of $\varepsilon_{k,n}, \ldots,$ $\varepsilon_{k,n-s+1}$ to 0. We also see that the removed zero-points do not affect the calculation of the number of real zero-points, as Theorem 3.6 shows. Next, changing the values of $\varepsilon_{k,j}$'s to 0 will change the values of the other zero-points of $P_k(x)$ slightly. However, condition (2) assures us that none of the real zero-points of $P_k(x)$ passes through the real zero-points of $P_{k-1}(x)$; thus sequence (3.13) is a general Sturm sequence. Therefore, as in Theorem 3.6, we can calculate the number of real zero-points of $\tilde{P}(x)$ by using sequence (3.13). □

Theorem 3.8 tells us that the problem of small leading coefficients reduces to that of checking whether or not any resultants become zero. We will propose several methods for this in Section 3.4.

We explain Theorem 3.8 by means of an example with exact arithmetic.

Example 3.2. Let $P(x)$ and $\tilde{P}(x)$ be

$$P(x) = x^5 + 4x^4 + \frac{6401}{1000}x^3 - 20x^2 + 5x + 1,$$
$$\tilde{P}(x) = P(x) + \delta_{0,4}x^4 + \delta_{0,3}x^3 + \cdots + \delta_{0,0}x^0,$$

where numbers $\delta_{0,4}, \ldots, \delta_{0,0}$ are unknown but bounded as

$$|\delta_{0,j}| \leqslant \varepsilon = 1/10000.$$

We obtain $(P_0, \ldots, P_5)$, the Sturm sequence of $P(x)$, as

$$\begin{aligned}
P_0 =& P(x),\\
P_1 =& \frac{d}{dx}P(x) = 5x^4 + 16x^3 + \frac{19203}{1000}x^2 - 40x + 5,\\
P_2 =& -\frac{1}{2500}x^3 + \frac{94203}{6250}x^2 - \frac{52}{5}x - \frac{1}{5},\\
P_3 =& -\frac{7099837085603}{1000}x^2 + 4898974540x + 94210995,\\
P_4 =& -\frac{1838986143841703970}{504076866421037007498736 09}x + \frac{581470528239934409}{5040768664210370074987360 9},\\
P_5 =& -\frac{31566508567667286525829957694414724087925197 08557}{33818700372417803243846409931137609 00000}.
\end{aligned} \tag{3.14}$$

Therefore, we have $N(-\infty) - N(\infty) = 3$. In Formula (3.14), P_2 has a small leading coefficient. (Correspondingly, $P_2(x)$ has a real zero-point at $x \simeq 37680.5$.) The conditions in Theorem 3.8 are satisfied as follows. First, the existence domains of approximate zero-points of $\tilde{P}(x)$ in the neighborhood of $x = 0$ are the intervals $[-0.12992, -0.12989]$, $[0.44536, 0.44541]$, and $[0.19803, 0.19810]$, while the existence domains of approximate zero-points of $P_2(x)$ when we change the value of the leading coefficient continuously from $-1/2500$ to 0 are the intervals $[-0.01877227\cdots,$ $-0.01877227\cdots]$, $[0.708722,\ 0.708735]$, and $[37680.5, \infty)$. Therefore, the existence domains of the real zero-points of $\tilde{P}(x)$ and $P_2(x)$ do not overlap; thus we have

$\mathrm{res}(\tilde{P}, P_2) \neq 0$. Second, the existence domains of approximate zero-points of $P_1(x)$ are the intervals $[0.134731, 0.134738]$, and $[0.910227, 0.910260]$. Therefore, the existence domains of the real zero-points of $P_1(x)$ and $P_2(x)$ do not overlap; thus we have $\mathrm{res}(P_1, P_2) \neq 0$. Since $P(x)$, $\tilde{P}(x)$, P_1, and P_2 satisfy the conditions in Theorem 3.8, we can calculate $P'_2, \dots, P'_4$ as

$$\begin{aligned} P'_2 &= \frac{94203}{6250}x^2 - \frac{52}{5}x - \frac{1}{5}, \\ P'_3 &= \frac{14367059719609325}{835976753303427}x - \frac{18170016322960675}{3343907013213708}, \\ P'_4 &= \frac{65440159831618155883480530106785213}{330259847978913242060689003129000000}. \end{aligned}$$

We have $N'(-\infty) - N'(\infty) = 3 = N(-\infty) - N(\infty)$.

3.4 EVALUATING THE EFFECTS OF ERROR TERMS

Theorems 3.3 and 3.8 show that some important problems in counting the number of approximate real zero-points can be reduced to checking whether or not certain resultants become zero owing to the error terms. In this section, we consider how to evaluate errors in the resultant of an approximate univariate polynomial. We investigate the following four methods:

1. Evaluating the "subresultant determinant" by using the Hadamard's inequality (Section 3.4.1),
2. Calculating the Sturm sequence with the coefficients of interval numbers (Section 3.4.2),
3. Solving a linear system on polynomial coefficients and evaluating errors in the solution by backward error analysis, which is a standard method in numerical analysis (Section 3.4.3),
4. Calculating the Sturm sequence with parametric error terms (Section 3.4.4).

The experiments were performed with a computer algebra system GAL [43] (General Algebraic Language/Laboratory, a LISP-based computer algebra system) running on a SPARC Station 5 (CPU: microSPARC II, 70MHz) and SunOS 4.1.4.

3.4.1 Evaluation of the Subresultant Determinant

Except for the overall signs of polynomials, the Sturm sequence is the same as the polynomial remainder sequence (PRS) for which the subresultant theory has been developed. (For the subresultant theory, see Mishra [32], for example.) With this theory, we can express the elements in the Sturm sequence by the determinants of the coefficients of two consecutive elements. Let $(P_0 = P, P_1 = dP/dx, P_2, \dots, P_{k-1}, P_k, \dots)$ be a Sturm sequence, and assume that

$$\begin{aligned} P_{k-1}(x) &= a_l x^l + \cdots + a_0 x^0, \\ P_k(x) &= \varepsilon_m x^m + \cdots + \varepsilon_{m-s+1} x^{m-s+1} + b_{m-s} x^{m-s} + \cdots + b_0 x^0, \end{aligned}$$

where

$$\max\{|\varepsilon_{k,n_k}|, \dots, |\varepsilon_{k,n_k-s+1}|\} \ll \min_{b_{k,j} \neq 0}\{|b_{k,n_k-s}|, \dots, |b_{k,0}|\},$$

as before.

Let $S_i(P_{k-1}, P_k)$ be the following determinant:

$$S_i(P_{k-1}, P_k) = \\ \begin{vmatrix} a_l & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{l-2i+1} & x^{i-1}P_{k-1} \\ & \ddots & & & & & & \vdots & \vdots \\ & & a_l & \cdots & \cdots & \cdots & \cdots & a_{l-i} & x^0P_{k-1} \\ \varepsilon_m & \cdots & \varepsilon_{m-s+1} & b_{m-s} & \cdots & \cdots & \cdots & b_{m-2i+1} & x^iP_k \\ & \ddots & & \ddots & \ddots & & & \vdots & \vdots \\ & & \varepsilon_m & \cdots & \varepsilon_{m-s+1} & b_{m-s} & \cdots & b_{m-i+1} & x^0P_k \end{vmatrix}.$$

$S_i(P_{k-1}, P_k)$ is called the i-th subresultant of $P_{k-1}(x)$ and $P_k(x)$, and we have $P_{k+i}(x) = \gamma_i S_i(P_{k-1}, P_k)$, with γ_i a constant. For example, if $\deg(P_{k-1}) = \deg(P_k) + 1$, we have

$$P_{k+1}(x) = S_1(P_{k-1}, P_k) = \begin{vmatrix} a_l & a_{l-1} & P_{k-1}(x) \\ \varepsilon_m & \varepsilon_{m-1} & xP_k(x) \\ & \varepsilon_m & P_k(x) \end{vmatrix}.$$

Below, we consider only the leading coefficients of P_{k+1}, P_{k+2}, and so on. Applying the Hadamard's inequality to the subresultant, we can bound the effect of $\varepsilon_m, \ldots, \varepsilon_{m-s+1}$ on $\mathrm{lc}(P_{k+i})$, as follows.

Proposition 3.9. *Define* P'_k *and* L *as*

$$\begin{aligned} P'_k =& P_k - (\varepsilon_m x^m + \cdots + \varepsilon_{m-s+1}x^{m-s+1}) = b_{m-s}x^{m-s} + \cdots + b_0, \\ L =& \|P_k\|_2^{(i-1)} \left\{ (i-s)\,|a_l|^s\, \|P_{k-1}\|_2^{(i-s)} + \sum_{j=1}^{s} |a_l|^{(j-1)} \|P_{k-1}\|_2^{(i-j+1)} \right\}. \end{aligned} \tag{3.15}$$

If $\mathrm{lc}(S_i(P_{k-1}, P_k)) \neq 0$ *and we have*

$$\{|\varepsilon_m| + \cdots + |\varepsilon_{m-s+1}|\} \cdot L < |a_l{}^s \cdot \mathrm{lc}(S_i(P_{k-1}, P'_k))|, \tag{3.16}$$

for $i = s, \ldots, m$, *then we have*

$$\mathrm{lc}(S_i(P_{k-1}, P_k)) \cdot a_l{}^s \cdot \mathrm{lc}(S_i(P_{k-1}, P'_k)) > 0. \tag{3.17}$$

Proof. Note that

$$\begin{aligned} \mathrm{lc}(S_i(P_{k-1}, P_k)) =& \begin{vmatrix} a_l & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{l-2i} \\ & \ddots & & & & & & \vdots \\ & & a_l & \cdots & \cdots & \cdots & \cdots & a_{l-i-1} \\ \varepsilon_n & \cdots & \varepsilon_{m-s+1} & b_{m-s} & \cdots & \cdots & \cdots & b_{m-2i} \\ & \ddots & & \ddots & \ddots & & & \vdots \\ & & \varepsilon_m & \cdots & \varepsilon_{m-s+1} & b_{m-s} & \cdots & b_{m-i} \end{vmatrix} \\ =& \begin{vmatrix} A_i \\ B_i \end{vmatrix}, \end{aligned} \tag{3.18}$$

where A_i and B_i are the row blocks consisting of the coefficients in P_{k-1} and P_k, respectively, and

$$\mathrm{lc}(S_i(P_{k-1}, P_k')) = \begin{vmatrix} a_l & \cdots & \cdots & \cdots & a_{l-2i+s} \\ & \ddots & & & \vdots \\ & & a_l & \cdots & a_{l-i-1} \\ b_{m-s} & \cdots & \cdots & \cdots & b_{m-2i} \\ & \ddots & & & \vdots \\ & & b_{m-s} & \cdots & b_{m-i} \end{vmatrix} = \begin{vmatrix} A_i' \\ \hline B_i' \end{vmatrix},$$

where A_i' and B_i' are the row blocks consisting of the coefficients in P_{k-1} and P_k', respectively, Note that, in the determinants above, we assume $a_j = b_j = 0$ for $j < 0$. Furthermore, let $B_{i,j}'$ be the top $j-1$ rows of B_i', $\mathbf{b}_{i,j}$ be the j-th row of B_i with replacing the coefficients in P_k with zeros, and $B_{i,j}$ be the bottom $i+1-j$ rows of B_i.

By expanding the determinant in Eq. (3.18) with respect to the $(i+1)$-th row as

$$\begin{aligned} &\begin{vmatrix} & \cdots & & & \cdots & \\ \varepsilon_m & \cdots & \varepsilon_{m-s+1} & b_{m-s} & \cdots & b_{m-2i} \\ & \cdots & & & \cdots & \end{vmatrix} \\ &= \begin{vmatrix} & \cdots & & & \cdots & \\ \varepsilon_m & \cdots & \varepsilon_{m-s+1} & 0 & \cdots & 0 \\ & \cdots & & & \cdots & \end{vmatrix} + \begin{vmatrix} & \cdots & & & \cdots & \\ 0 & \cdots & 0 & b_{m-s} & \cdots & b_{m-2i} \\ & \cdots & & & \cdots & \end{vmatrix}, \end{aligned}$$

and expanding the last determinant similarly, we finally obtain

$$\mathrm{lc}(S_i(P_{k-1}, P_k)) = {a_l}^s \cdot \mathrm{lc}(S_i(P_{k-1}, P_k')) + \sum_{j=1}^{i+1} \det(R_{i,j}), \tag{3.19}$$

where

$$R_{i,j} = \begin{pmatrix} A_i \\ \hline B_{i,j}' \\ \hline \mathbf{b}_{i,j} \\ \hline B_{i,j} \end{pmatrix}.$$

Expanding $\det(R_{i,j})$ with respect to the $(i+j)$-th row, or the row

$$\mathbf{b}_{i,j} = (0 \cdots 0\, \varepsilon_m \cdots \varepsilon_{m-s+1}\, 0 \cdots 0),$$

we have

$$\det(R_{i,j}) \quad = \quad (-1)^{i+2j} \varepsilon_m \det(\tilde{R}_{i,j,j}) + \cdots + (-1)^{i+2j+s} \varepsilon_{m-s+1} \det(\tilde{R}_{i,j,j+s}),$$

where $\tilde{R}_{i,j,q}$ is a $2i \times 2i$ submatrix obtained by removing the $(i+j)$-th row and the q-th column from $R_{i,j}$. After removing several top-left diagonal elements a_l's of $\tilde{R}_{i,j,q}$, and applying the Hadamard's inequality to $\det(\tilde{R}_{i,j,q})$, with inequalities $|a_l|^2 + |a_{l-1}|^2 + \cdots + |a_{l-2i}|^2 \leqslant \|P_{k-1}\|_2^2$ and $|\varepsilon_m|^2 + \cdots + |\varepsilon_{m-s+1}|^2 + |b_{m-s}|^2 + \cdots + |b_{m-2i}|^2 \leqslant \|P_k\|_2^2$, we finally obtain the following inequality:

$$\begin{aligned} |\det(R_{i,j})| \leqslant & M_i \left\{ |a_l|^{(j-1)} \, \|P_{k-1}\|_2^{(i-j+1)} \right\} \quad \text{for } j = 1, \ldots, s, \\ |\det(R_{i,j})| \leqslant & M_i \left\{ |a_l|^{s} \, \|P_{k-1}\|_2^{(i-s)} \right\} \quad \text{for } j = s+1, \ldots, i+1, \end{aligned}$$

where

$$M_i = \{|\varepsilon_m| + \cdots + |\varepsilon_{m-s+1}|\} \cdot \|P_k\|_2^i.$$

From assumption (3.16), we have

$$\begin{aligned}\left|\sum_{j=1}^{i+1} \det(R_{i,j})\right| &\leqslant \sum_{j=1}^{i+1} \left|\det(R_{i,j})\right| \leqslant \{|\varepsilon_m| + \cdots |\varepsilon_{m-s+1}|\} \cdot L \\ &< |a_l{}^s \cdot \mathrm{lc}(S_i(P_{k-1}, P_k'))|.\end{aligned} \tag{3.20}$$

Therefore, from (3.19) and (3.20), we obtain Formula (3.17). □

From the fundamental theorem of subresultants (see Theorem 4.1 on page 47 [6]), we have

$$\begin{aligned}S_i(P_{k-1}, P_k') &= P_{k+h}' \,\mathrm{lc}(P_{k+h}')^{d_{k+h-1}-1} \\ &\times \prod_{l=1}^{h} \left\{ \mathrm{lc}(P_{k+l-1}')^{(d_{k+l-2}+d_{k+l-1})} (-1)^{(n_{k+l-2}-n_{k+h})(n_{k+l-1}-n_{k+h})} \right\},\end{aligned}$$

where $h = i - s$, $n_{k+j} = \deg(P_{k+j}')$ and $d_j = n_j - n_{j+1}$. Therefore, we can calculate $\mathrm{lc}(S_i(P_{k-1}, P_k'))$ easily from $\mathrm{lc}(P_{k+h}')$.

Proposition 3.9 shows that, so long as $\varepsilon_m, \ldots, \varepsilon_{m-s+1}$ satisfy condition (3.16), discarding terms $\varepsilon_m x^m, \ldots, \varepsilon_{m-s+1} x^{m-s+1}$ in P_k does not change the signs of leading coefficients of the subresultants $S_i(P_{k-1}, P_k')$ for $i = 0, \ldots, m-s-1$. However, in actual calculation of the Sturm sequence, the number L in (3.15) seems to become too large, thus condition (3.16) is not useful in practice.

3.4.2 Utilization of Interval Arithmetic

In this method, we transform the coefficients of the given polynomial into interval numbers each of which includes the corresponding error, and calculate the Sturm sequence by using interval arithmetic.

By observing how the widths of intervals increased during the calculation, we found that the increase of the width of each interval was about one decimal-digit for each remainder computation. In fact, the division of polynomials of degree difference 1 requires two "polynomial × number" multiplications and two polynomial subtractions. The width of an interval is increased to about twice that of the original interval by one arithmetic operation if the operands are of almost the same widths; thus the width increases by about $2^4 = 16$ times after the polynomial division. As a consequence, for a polynomial of degree 10, for example, the width of an interval in the last element of the Sturm sequence may become about 10^{10} times larger than the initial widths, which shows that this method is not useful in practice.

3.4.3 Backward Error Analysis for a Linear System

In numerical analysis, we have a good method of error estimation for the solution of a system of linear equations called "backward error analysis." Calculation of the resultant can be reduced to solving a linear system.

Usually, the norm of vectors and matrices are defined as follows. Let $\mathbf{x} = (x_1, \dots, x_m)^T$ be a vector in $\mathbf{R}^m$. Then, for $p = 1, 2, \infty$, the p-norm of $\mathbf{x}$ is defined as

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{m} |x_i|^p \right)^{1/p}.$$

Let $A = (a_{ij})$ be a real (m, m)-matrix. Then, by using the norm of a vector, we define the p-norm of A as

$$\|A\|_p = \max_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}.$$

In this chapter we use only $\|A\|_1$ and $\|A\|_\infty$.

Let $F(x)$ and $G(x)$ be

$$\begin{aligned} F(x) &= f_m x^m + \cdots + f_0 x^0, \quad f_m \neq 0, \\ G(x) &= g_n x^n + \cdots + g_0 x^0, \quad g_n \neq 0, \end{aligned}$$

where $m \geqslant n$. Calculation of the PRS is equivalent to eliminating the terms of higher degrees of F and G to derive R_s, a polynomial of degree s, for $0 \leqslant s \leqslant n-1$. For each R_s, there exist polynomials U_s and V_s such that

$$U_s F + V_s G = R_s, \quad \deg(U_s) \leqslant n - s - 1, \quad \deg(V_s) \leqslant m - s - 1.$$

We consider calculating $R_0 = \mathrm{res}(F, G)$. Let U_0 and V_0 be expressed as

$$U_0 = u_{n-1} x^{n-1} + \cdots + u_0 x^0, \quad V_0 = v_{m-1} x^{m-1} + \cdots + v_0 x^0.$$

From the relation $U_0 F + V_0 G = R_0$, we obtain a system of linear equations on the coefficients in U_0 and V_0, as

$$\begin{pmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ f_0 & & f_m & g_0 & & g_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & f_0 & & & g_0 \end{pmatrix} \begin{pmatrix} u_{n-1} \\ \vdots \\ u_0 \\ v_{m-1} \\ \vdots \\ v_0 \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ R_0 \end{pmatrix}. \tag{3.21}$$

U_0 and V_0 can be normalized in any way so long as U_0 and V_0 satisfy the above relation. Therefore, we normalize U_0 and V_0 as $u_{n-1} = g_n$ and $v_{m-1} = -f_m$. With this normalization, we can rewrite relation (3.21) as

$$\begin{pmatrix} f_m & & & & g_n & & & \\ \vdots & \ddots & & & \vdots & \ddots & & \\ \vdots & & f_m & & \vdots & & g_n & \\ f_1 & & \vdots & f_m & g_1 & & \vdots & g_n \\ f_0 & \ddots & \vdots & \vdots & g_0 & \ddots & \vdots & \vdots \\ & \ddots & f_1 & \vdots & & \ddots & g_1 & \vdots \\ & & f_0 & f_1 & & & g_0 & g_1 \end{pmatrix} \begin{pmatrix} u_{n-2} \\ \vdots \\ u_0 \\ v_{m-2} \\ \vdots \\ v_0 \end{pmatrix} = \begin{pmatrix} g_{n-1} f_m - f_{m-1} g_n \\ \vdots \\ g_{n-m} f_m - f_0 g_n \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \tag{3.22}$$

where $g_j = 0$ for $j < 0$, and

$$R_0 = f_0 u_0 + g_0 v_0. \tag{3.23}$$

The linear system (3.22) is of the form

$$A\mathbf{x} = \mathbf{b}, \tag{3.24}$$

where A is a "coefficient matrix," and $\mathbf{x}$ and $\mathbf{b}$ are vectors of unknowns and given numbers, respectively. We briefly describe a perturbation theory for linear system. (The theory can be found in various literature on numerical analysis; see Higham [20], for example.) Assume that $\mathbf{b}$ has an error $\Delta\mathbf{b}$ that causes an error $\Delta\mathbf{x}_1$ in the solution $\mathbf{x}$. Then we have

$$A(\mathbf{x} + \Delta\mathbf{x}_1) = \mathbf{b} + \Delta\mathbf{b}. \tag{3.25}$$

Using Formula (3.24), we can easily evaluate the magnitude of $\Delta\mathbf{x}_1$ as

$$\frac{\|\Delta\mathbf{x}_1\|}{\|\mathbf{x}\|} \leqslant \|A\|\,\|A^{-1}\|\,\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}. \tag{3.26}$$

Furthermore, assume that A has an error ΔA and that the error of $\mathbf{x}$ becomes $\Delta\mathbf{x}_1 + \Delta\mathbf{x}_2$, as

$$(A + \Delta A)(\mathbf{x} + \Delta\mathbf{x}_1 + \Delta\mathbf{x}_2) = \mathbf{b} + \Delta\mathbf{b}.$$

Using (3.25), we derive the evaluation of $\Delta\mathbf{x}_2$ as

$$\frac{\|\Delta\mathbf{x}_2\|}{\|\mathbf{x} + \Delta\mathbf{x}_1 + \Delta\mathbf{x}_2\|} \leqslant \|A\|\,\|A^{-1}\|\,\frac{\|\Delta A\|}{\|A\|}. \tag{3.27}$$

Formulas (3.26) and (3.27) lead us to the following evaluation as

$$\frac{\|\Delta\mathbf{x}_1\| + \|\Delta\mathbf{x}_2\|}{\|\mathbf{x}\| + \|\Delta\mathbf{x}_1\| + \|\Delta\mathbf{x}_2\|} \leqslant \|A\|\,\|A^{-1}\| \left\{ \frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|} \right\}, \tag{3.28}$$

where the number $\|A\|\,\|A^{-1}\|$ is called the "condition number" which specifies how the initial errors are magnified in the solution.

Although we did not consider rounding errors in floating-point arithmetic in the above evaluation, the evaluation of rounding errors can easily be included by adding ΔR, a term representing rounding errors, into A. It is known that, if we solve (3.24) by the Gaussian elimination with pivoting, for example, the errors $\Delta\mathbf{x}_1 + \Delta\mathbf{x}_2$ in the solution $\mathbf{x}$ are well bounded by the formula (3.28) (see Higham [20]).

Applying formula (3.28) to linear system (3.22), we can bound the errors $|\delta_{u_0}|$ and $|\delta_{v_0}|$ of the solutions u_0 and v_0, due to the perturbations δ_{f_i} of f_i $(i = 0, \ldots, m)$ and δ_{g_j} of g_j $(j = 0, \ldots, n)$. Formula (3.23) tells us that if $|f_0 u_0 + g_0 v_0| \gg |f_0 \cdot \delta_{u_0}|, |g_0 \cdot \delta_{v_0}|$ then we can say definitely that $R_0 \neq 0$ for the perturbations of the coefficients of F and G. If $|f_0 u_0 + g_0 v_0| \ll |f_0 u_0|, |g_0 v_0|$ then this case corresponds to F and G having mutually close zero-points, and the above method cannot be applied to such cases. If $|f_0 u_0 + g_0 v_0|$ is not small, then we can apply the above method so long as $|\delta_{u_0}|$ and $|\delta_{v_0}|$ are not large. Formula (3.28) shows that the measure of largeness of $|\delta_{u_0}|$ and $|\delta_{v_0}|$ is the condition number. Therefore, in order to check whether or not the above method is useful, we check the largeness of the condition number for polynomials of degrees from 10 to 50. We generate a real univariate polynomial $P(x)$ with random coefficients, and construct the matrix in the left-hand-side of Formula (3.22) by putting $F = P$ and $G = dP/dx$. We generate each coefficient c of $P(x)$ to satisfy $|c| \leqslant 10$. We set $\deg(P) = 10, 20, 30, 40, 50$, and generate 10 polynomials for each degree. We used the LAPACK library [2] linked to GAL to estimate the condition number (for estimating the condition number, see Natori [33], for example).

Degree of $P(x)$	Condition number					
	1-norm			∞-norm		
	Maximum	Minimum	Average	Maximum	Minimum	Average
10	8.73×10^3	1.69×10^2	2.55×10^3	7.96×10^3	2.92×10^2	2.99×10^3
20	2.57×10^6	4.44×10^3	2.95×10^5	8.51×10^5	1.83×10^3	1.08×10^5
30	1.16×10^7	4.97×10^4	2.46×10^6	5.97×10^7	2.45×10^4	1.18×10^6
40	5.37×10^7	1.48×10^5	7.44×10^6	4.76×10^7	6.01×10^4	6.00×10^6
50	1.47×10^8	1.56×10^5	2.25×10^7	6.42×10^7	7.38×10^4	8.09×10^6

Table 3.1: Condition number of the matrix in Formula (3.22) computed for 10 polynomials with random-number coefficients.

Table 3.1 shows the result of computations. For each degree of polynomial, we show the maximum, minimum, and average values of our estimates of 10 condition numbers. We see from this result that, for a polynomial of degree 10, for example, the error in $\mathrm{res}(P, dP/dx)$ may become 10^3 or 10^4 times larger than the error in the initial polynomial. Although these numbers are rather large, they are much smaller than the increase of the interval width explained above.

3.4.4 Calculating Error Terms Parametrically

The method described in this subsection gives good estimates of errors in the Sturm sequence, but the calculated value does not give the rigorous error bound.

For simplicity, we assume that $P(x)$ is monic in (3.1), and express $\tilde{P}(x)$ in Formula (3.2) as

$$\tilde{P}(x, \delta_{n-1}, \dots, \delta_0) = x^n + (c_{n-1} + \delta_{n-1})x^{n-1} + \cdots + (c_0 + \delta_0)x^0,$$

where $\delta_{n-1}, \dots, \delta_0$ are parameters representing errors in the coefficients. Exact calculation of the Sturm sequence of a parametric polynomial $\tilde{P}$ exactly is extremely time-consuming, because $\tilde{P}$ is $(n+1)$-variate. However, if we neglect all the quadratic and higher-order terms with respect to $\delta_{n-1}, \dots, \delta_0$, then the computation cost is only $O(n)$ times larger than that of a numerical Sturm sequence. Therefore, we calculate the i-th element $\tilde{P}_i$ of the Sturm sequence as

$$\tilde{P}_i(x, \delta_{n-1}, \dots, \delta_0) \simeq P_i(x) + \tilde{P}_{i,n-1}(x, 0, \dots, 0)\delta_{n-1} + \cdots + \tilde{P}_{i,0}(x, 0, \dots, 0)\delta_0,$$

where $\tilde{P}_{i,j} = \partial\tilde{P}_i/\partial\delta_j$ ($j = n-1$, ..., 0). Then, by neglecting the terms of order $O(\delta^2)$, we can approximately bound the effect of error terms fairly well, as

$$|\tilde{P}_i - P_i| \lesssim |\tilde{P}_{i,n-1}(x, 0, \dots, 0)|\, \varepsilon_{n-1} + \cdots + |\tilde{P}_{i,0}(x, 0, \dots, 0)|\, \varepsilon_0,$$

where |(polynomial)| denotes a polynomial with the coefficients replaced by their absolute values.

Actually, the calculation is performed by introducing the total-degree variable t for $\delta_{n-1}, \dots, \delta_0$ as $\delta_i \to \delta_i t$ ($i = 0, \dots, n-1$). We calculate the Sturm sequence only up to the total-degree 1, and substitute 1 for t after the calculation.

We calculated the Sturm sequences with and without parameterized error terms. For this experiment, we used the same polynomials as in Section 3.4.3.

Table 3.2 shows the value $\|\tilde{P}_n(x, \delta_{n-1}, \dots, \delta_0)\|/\|\tilde{P}_n(x, 0, \dots, 0)\|$, where $\tilde{P}_n(x, \delta_{n-1}, \dots, \delta_0)$ is the last element of the Sturm sequence, and Table 3.3 shows

Degree of $\tilde{P}(x)$	Polynomial norm					
	1-norm			∞-norm		
	Maximum	Minimum	Average	Maximum	Minimum	Average
10	2.57×10^4	1.02×10^2	5.97×10^3	1.52×10^4	1.68×10^2	5.00×10^3
20	2.83×10^5	1.34×10^5	1.81×10^5	3.65×10^5	3.44×10^5	2.62×10^5
30	2.68×10^9	7.01×10^8	1.13×10^9	4.75×10^9	1.78×10^9	1.76×10^9
40	3.50×10^{13}	2.99×10^{11}	5.39×10^{12}	1.60×10^{14}	1.42×10^{11}	3.65×10^{12}
50	2.80×10^{17}	7.81×10^{15}	9.19×10^{16}	2.47×10^{17}	1.32×10^{16}	1.36×10^{17}

Table 3.2: $\|\tilde{P}_n(x, \delta_{n-1}, \ldots, \delta_0)\| / \|\tilde{P}_n(x, 0, \ldots, 0)\|$ for 10 polynomials, where $\tilde{P}_n$ is the last element of the Sturm sequence.

Degree of $\tilde{P}(x)$	Computing time (msec.)					
	With error terms			Without error terms		
	Maximum	Minimum	Average	Maximum	Minimum	Average
10	70	50	55	10	< 10	< 10
20	420	400	403	10	< 10	< 10
30	1420	1330	1357	20	10	11
40	3280	3210	3244	50	10	31
50	6080	6030	6050	50	30	37

Table 3.3: Computing times for calculating the Sturm sequences with and without parameterized error terms.

the computing times of the Sturm sequences with and without parametric errors. In Table 3.2, for each degree of polynomial, we show the maximum, the minimum, and the average of 10 ratios. Note that the values in Table 3.2 show how the initial errors are magnified by the computation of the Sturm sequence, just as the values in Table 3.1 show. Comparing with Table 3.1, we see that the numbers are too large for polynomials of higher degrees. Table 3.3 shows the maximum, minimum, and average values of the computation times for ten examples. We see that, very roughly speaking, the computation time for a parameterized sequence is about $\deg(P)$ times larger than that for a numerical sequence. These results indicate that we can use this method only for polynomials of low or medium degrees.

3.5 SUMMARY

In this chapter we have considered the real zero-points of a real univariate polynomial with error terms whose coefficients may be much larger than the machine epsilon ε_M of the floating-point arithmetic. For such an approximate polynomial, we introduced the concept of an "approximate real zero-point" and proposed a method for calculating the existence domains of zero-points fairly accurately and simply.

Next, we considered how to calculate the number of real zero-points of an approximate polynomial by the Sturm's method. We gave a sufficient condition for the number of real zero-points to be definite. We also derived a sufficient condition for the small leading coefficients in the Sturm sequence to be discarded, and showed that these problems can be reduced to a problem to that of estimating the errors in the resultants of univariate polynomials.

Finally, in order to estimate the errors in the Sturm sequence, we investigated four methods:

(1) Evaluating the "subresultant determinant" by using the Hadamard's inequality,

(2) Calculating the Sturm sequence with coefficients of interval numbers,

(3) Solving a linear system on polynomial coefficients and evaluating errors in the solution by backward error analysis,

(4) Calculating the Sturm sequence with parametric error terms.

Method (1) is theoretically correct, but the calculated upper bound is too large, and with method (2) the width of each interval number grows too rapidly during the calculation of the Sturm sequence; thus methods (1) and (2) do not seem to be useful in practice. Method (3) gives a rather practical estimation, and thus seems to be useful in practice. Method (4) gives the errors rather accurately, and we have seen that calculating the resultant by PRS gives much larger errors than method (3). This means that the errors contained in the resultant depend on which method we have used to calculate the resultant, and method (3) seems to be the best for evaluating the errors.

We still have a problem in cases where $P(x)$ has multiple or close zero-points. Let us briefly mention what happens if $P(x)$ has close zero-points. Let $\|.\|$ be an appropriate norm of a polynomial defined by Formula (3.9), and assume that $\|P\| = 1$ and P_k contains m close zero-points of closeness δ, $0 < \delta \ll 1$, around the origin. Then, Sasaki and Sasaki [45] tell us that $\|P_k\| = O(\delta^0)$ and $\|P_{k+1}\| = O(\delta^2)$, $\|P_{k+2}\| = O(\delta^3), \ldots, \|P_{k+m}\| = O(\delta^{m+1})$. Therefore, if these close zero-points can be separated and counted as m single zero-points, we must have $\|P_{k+m}\| \gg \varepsilon_M$ or $\delta \gg \sqrt[m+1]{\varepsilon_M}$. On the other hand, if we change coefficients of $P(x)$ slightly, the positions of these close zero-points are changed considerably. Therefore, the treatment of close zero-points is not easy, but an approach toward possible treatment will be discussed in the next chapter.

4 RECURSIVE POLYNOMIAL REMAINDER SEQUENCE AND ITS SUBRESULTANTS

The polynomial remainder sequence (PRS) is one of the most fundamental tools in computer algebra. Although the Euclidean algorithm (see Knuth [29]) for calculating PRS is simple, coefficient growth in PRS makes the Euclidean algorithm often very inefficient. To overcome this problem, the mechanism of coefficient growth has been extensively studied through the theory of subresultants; see Collins [9], Brown and Traub [6], Loos [31], etc. By the theory of subresultant, we can remove extraneous factors of the elements of PRS systematically.

In this chapter, we consider a variation of the subresultant. When we calculate PRS for polynomials which have a nontrivial greatest common divisor (GCD), we usually stop the calculation with the GCD. However, it is sometimes useful to continue the calculation by calculating the PRS for the GCD and its derivative; this is necessary for calculating the number of real zeros including their multiplicities. We call such a PRS a "recursive PRS."

Although the theory of subresultants has been developed widely, the corresponding theory for recursive PRS is still unknown within the author's knowledge; this is the main problem which we investigate here. By "recursive subresultants," we denote determinants which represent elements of recursive PRS as functions of the coefficients of initial polynomials.

We give three different constructions of subresultant matrices to express recursive subresultants. The first matrix construction recursively builds the matrix by shifting previously defined matrices, similarly as the Sylvester matrix shifts coefficients of the initial polynomials, thus the size of the matrices increases fast as the recursion deepens. The second matrix construction uses "nested" matrices, or a Sylvester matrix whose entries are themselves determinants. Finally, by the Gaussian elimination with the Sylvester's identity on the second construction, we succeed to give the reduced matrix construction which expresses the coefficients of the polynomials in the recursive PRS as determinants of very small matrices, whose size actually decreases as the recursion deepens.

This chapter is organized as follows. In Section 4.1, we introduce the concept of recursive PRS. In Section 4.2, we define recursive subresultant and show its relationship to recursive PRS. In Section 4.3, we define the "nested subresultant," which is derived from the second construction of subresultant matrix, and show its equivalence to the recursive subresultant. In Section 4.4, we define the "reduced nested subresultant," whose matrix is derived from the nested subresultant, and show that it is a reduced expression of the recursive subresultant. In Section 4.5, we briefly discuss usage of the reduced nested subresultant in approximate algebraic computation.

4.1 RECURSIVE POLYNOMIAL REMAINDER SEQUENCE (PRS)

First, we review the PRS, then define the recursive PRS. In the end of this section, we show a recursive Sturm sequence as an example of recursive PRS. We follow definitions and notations by von zur Gathen and Lücking [61]. Throughout this chapter, let R be an integral domain and K be its quotient field. We define a polynomial remainder sequence as follows.

Definition 4.1 (Polynomial Remainder Sequence (PRS))**.** Let F and G be polynomials in $R[x]$ of degree m and n ($m > n$), respectively. A sequence

$$(P_1, \dots, P_l)$$

of nonzero polynomials is called a *polynomial remainder sequence (PRS)* for F and G, abbreviated to $\mathrm{prs}(F, G)$, if it satisfies

$$P_1 = F, \quad P_2 = G, \quad \alpha_i P_{i-2} = q_{i-1} P_{i-1} + \beta_i P_i,$$

for $i = 3, \dots, l$, where $\alpha_3, \dots, \alpha_l$, $\beta_3, \dots, \beta_l$ are elements of R and $\deg(P_{i-1}) > \deg(P_i)$. A sequence $((\alpha_3, \beta_3), \dots, (\alpha_l, \beta_l))$ is called a *division rule* for $\mathrm{prs}(F, G)$. If P_l is a constant, then the PRS is called *complete*.

If F and G are coprime, the last element in the complete PRS for F and G is a constant. Otherwise, it equals the GCD of F and G up to a constant: we have $\mathrm{prs}(F, G) = (P_1 = F, P_2 = G, \dots, P_l = \gamma \cdot \gcd(F, G))$ for some $\gamma \in R$. Then, we can calculate new PRS, $\mathrm{prs}(P_l, \frac{d}{dx} P_l)$, and if this PRS ends with a non-constant polynomial, then calculate another PRS for the last element, and so on. By repeating this calculation, we can calculate several PRSs "recursively" such that the last polynomial in the last sequence is a constant. Thus, we define "recursive PRS" as follows.

Definition 4.2 (Recursive PRS)**.** Let F and G be the same as in Definition 4.1. Then, a sequence

$$(P_1^{(1)}, \dots, P_{l_1}^{(1)}, P_1^{(2)}, \dots, P_{l_2}^{(2)}, \dots, P_1^{(t)}, \dots, P_{l_t}^{(t)})$$

of nonzero polynomials is called a *recursive polynomial remainder sequence* (recursive PRS) for F and G, abbreviated to $\mathrm{rprs}(F, G)$, if it satisfies

$$\begin{aligned}
&P_1^{(1)} = F, \quad P_2^{(1)} = G, \quad P_{l_1}^{(1)} = \gamma_1 \cdot \gcd(P_1^{(1)}, P_2^{(1)}) \text{ with } \gamma_1 \in R, \\
&(P_1^{(1)}, P_2^{(1)}, \dots, P_{l_1}^{(1)}) = \mathrm{prs}(P_1^{(1)}, P_2^{(1)}), \\
&P_1^{(k)} = P_{l_{k-1}}^{(k-1)}, \quad P_2^{(k)} = \frac{d}{dx} P_{l_{k-1}}^{(k-1)}, \quad P_{l_k}^{(k)} = \gamma_k \cdot \gcd(P_1^{(k)}, P_2^{(k)}) \text{ with } \gamma_k \in R, \\
&(P_1^{(k)}, P_2^{(k)}, \dots, P_{l_k}^{(k)}) = \mathrm{prs}(P_1^{(k)}, P_2^{(k)}),
\end{aligned}$$

for $k = 2, \dots, t$. If $\alpha_i^{(k)}, \beta_i^{(k)} \in R$ satisfy

$$\alpha_i^{(k)} P_{i-2}^{(k)} = q_{i-1}^{(k)} P_{i-1}^{(k)} + \beta_i^{(k)} P_i^{(k)}$$

for $k = 1, \dots, t$ and $i = 3, \dots, l_k$, then a sequence $((\alpha_3^{(1)}, \beta_3^{(1)}), \dots, (\alpha_{l_t}^{(t)}, \beta_{l_t}^{(t)}))$ is called a *division rule* for $\mathrm{rprs}(F, G)$. Furthermore, if $P_{l_t}^{(t)}$ is a constant, then the recursive PRS is called complete.

Remark 4.1. In this chapter, we use the following notations unless otherwise defined: for $k = 1, \dots, t$ and $i = 1, \dots, l_k$, let $c_i^{(k)} = \mathrm{lc}(P_i^{(k)})$, $n_i^{(k)} = \deg(P_i^{(k)})$ (letters lc and deg denote the leading coefficient and the degree of the polynomial, respectively), $j_0 = m$ and $j_k = n_{l_k}^{(k)}$, and let $d_i^{(k)} = n_i^{(k)} - n_{i+1}^{(k)}$ for $k = 1, \dots, t$ and $i = 1, \dots, l_k - 1$. Furthermore, we represent $P_i^{(k)}(x)$ as

$$P_i^{(k)}(x) = a_{i,n_i^{(k)}}^{(k)} x^{n_i^{(k)}} + \cdots + a_{i,0}^{(k)} x^0,$$

and its "coefficient vector" as

$$\mathbf{p}_i^{(k)} = {}^t(a_{i,n_i^{(k)}}^{(k)}, \dots, a_{i,0}^{(k)}).$$

As an example of recursive PRS, we calculate Sturm sequences recursively for calculating the number of real zeros of univariate polynomial including multiplicities (see Bochnak, Coste and Roy [5]), as follows.

Example 4.1 (Recursive Sturm Sequence). Let $P(x) = (x+2)^2\{(x-3)(x+1)\}^3$, and calculate the recursive Sturm sequence of $P(x)$ as

$$\text{(complete) rprs}\left(P(x), \frac{d}{dx}P(x)\right),$$

with division rule given by

$$(\alpha_i^{(k)}, \beta_i^{(k)}) = (1, -1),$$

for $k = 1, \dots, t$ and $i = 3, \dots, l_k$.

The first sequence $L_1 = (P_1^{(1)}, \dots, P_4^{(1)})$ has the following elements:

$$\begin{aligned}
P_1^{(1)} &= P(x) = (x+2)^2\{(x-3)(x+1)\}^3,\\
P_2^{(1)} &= \frac{d}{dx}P(x) = 8x^7 - 14x^6 - 102x^5 + 80x^4 + 460x^3 + 66x^2 - 558x - 324,\\
P_3^{(1)} &= \frac{75}{16}x^6 - \frac{45}{16}x^5 - 60x^4 - \frac{225}{8}x^3 + \frac{3315}{16}x^2 + \frac{4815}{16}x + \frac{945}{8},\\
P_4^{(1)} &= \frac{128}{25}x^5 - \frac{256}{25}x^4 - \frac{256}{5}x^3 + \frac{1024}{25}x^2 + \frac{4224}{25}x + \frac{2304}{25}.
\end{aligned}$$

The second sequence $L_2 = (P_1^{(2)}, \dots, P_4^{(2)})$ has the following elements:

$$\begin{aligned}
P_1^{(2)} &= P_4^{(1)} = \frac{128}{25}x^5 - \frac{256}{25}x^4 - \frac{256}{5}x^3 + \frac{1024}{25}x^2 + \frac{4224}{25}x + \frac{2304}{25},\\
P_2^{(2)} &= \frac{d}{dx}P_4^{(1)} = \frac{128}{5}x^4 - \frac{1024}{25}x^3 - \frac{768}{5}x^2 + \frac{2048}{25}x + \frac{4224}{25},\\
P_3^{(2)} &= \frac{14848}{625}x^3 - \frac{1536}{125}x^2 - \frac{88576}{625}x - \frac{66048}{625},\\
P_4^{(2)} &= \frac{12800}{841}x^2 - \frac{25600}{841}x - \frac{38400}{841}.
\end{aligned}$$

The last sequence $L_3 = (P_1^{(3)}, \dots, P_3^{(3)})$ has the following elements:

$$\begin{aligned}
P_1^{(3)} &= P_4^{(2)} = \frac{12800}{841}x^2 - \frac{25600}{841}x - \frac{38400}{841},\\
P_2^{(3)} &= \frac{d}{dx}P_4^{(2)} = \frac{25600}{841}x - \frac{25600}{841},\\
P_3^{(3)} &= \frac{51200}{841}.
\end{aligned}$$

For PRS L_k, $k = 1,2,3$, define sequences of nonzero real numbers $\lambda(L_k, -\infty)$ and $\lambda(L_k, +\infty)$ as

$$\lambda(L_k, -\infty) = \left((-1)^{n_1^{(k)}} \operatorname{lc}(P_1^{(k)}), \dots, (-1)^{n_{l_k}^{(k)}} \operatorname{lc}(P_{l_k}^{(k)})\right),$$
$$\lambda(L_k, +\infty) = \left(\operatorname{lc}(P_1^{(k)}), \dots, \operatorname{lc}(P_{l_k}^{(k)})\right),$$

where $n_i^{(k)} = \deg(P_i^{(k)})$ denotes the degree of $P_i^{(k)}$ and $\operatorname{lc}(P_i^{(k)})$ denotes the leading coefficients of $P_i^{(k)}$. Then, $\lambda(L_k, -\infty)$ and $\lambda(L_k, +\infty)$ for $k = 1,2,3$ are

$$\lambda(L_1, \pm\infty) = \left(1, \pm 8, \frac{75}{16}, \pm\frac{128}{25}\right),$$
$$\lambda(L_2, \pm\infty) = \left(\pm\frac{128}{25}, \frac{128}{5}, \pm\frac{18848}{625}, \frac{12800}{841}\right),$$
$$\lambda(L_3, \pm\infty) = \left(\frac{12800}{841}, \pm\frac{25600}{841}, \frac{51200}{841}\right).$$

For a sequence of nonzero real numbers $L = (a_1, \dots, a_m)$, let $V(L)$ be the number of sign variations of the elements of L. Then, we calculate the number of the real zeros of $P(x)$, including multiplicity, as

$$\sum_{k=1}^{3} \{V(\lambda(L_k, -\infty)) - V(\lambda(L_k, +\infty))\} = 3 + 3 + 2 = 8.$$

4.2 SUBRESULTANTS FOR RECURSIVE PRS

To make this chapter self-contained and to use notations in our definitions, we first review the fundamental theorem of subresultants, then discuss subresultants for recursive PRS.

Although the theory of subresultants is established for polynomials over an integral domain, in what follows, we handle polynomials over a field for the sake of simplicity. Let F and G be polynomials in $K[x]$ such that

$$F(x) = f_m x^m + \cdots + f_0 x^0, \quad G(x) = g_n x^n + \cdots + g_0 x^0, \tag{4.1}$$

with $m \geqslant n > 0$. For a square matrix M, we denote its determinant by $|M|$.

4.2.1 Fundamental Theorem of Subresultants

Definition 4.3 (Sylvester Matrix)**.** Let F and G be as in (4.1). The *Sylvester matrix* of F and G, denoted by $N(F,G)$, is an $(m+n) \times (m+n)$ matrix constructed from the coefficients of F and G, such that

$$N(F,G) = \underbrace{\begin{pmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ f_0 & & f_m & g_0 & & g_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & f_0 & & & g_0 \end{pmatrix}}_{n \qquad\qquad m}.$$

Definition 4.4 (Subresultant Matrix)**.** Let F and G be defined as in (4.1). For $j < n$, the j-*th subresultant matrix* of F and G, denoted by $N^{(j)}(F,G)$, is an $(m+n-j) \times (m+n-2j)$ sub-matrix of $N(F,G)$ obtained by taking the left $n-j$ columns of coefficients of F and the left $m-j$ columns of coefficients of G, such that

$$N^{(j)}(F,G) = \begin{pmatrix} f_m & & & g_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ f_0 & & f_m & g_0 & & g_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & f_0 & & & g_0 \end{pmatrix}.$$
$$\qquad\qquad \underbrace{\qquad\qquad}_{n-j} \quad \underbrace{\qquad\qquad}_{m-j}$$

Furthermore, define $N_U^{(j)}(F,G)$ as a sub-matrix of $N^{(j)}(F,G)$ by deleting the bottom $j+1$ rows.

Definition 4.5 (Subresultant)**.** Let F and G be defined as in (4.1). For $j < n$ and $k = 0,\dots,j$, let $N_k^{(j)} = N_k^{(j)}(F,G)$ (distinguish it from $N_U^{(j)}(F,G)$ in the above) be a sub-matrix of $N^{(j)}(F,G)$ obtained by taking the top $m+n-2j-1$ rows and the $(m+n-j-k)$-th row (note that $N_k^{(j)}(F,G)$ is a square matrix). Then, the polynomial

$$S_j(F,G) = |N_j^{(j)}|x^j + \cdots + |N_0^{(j)}|x^0$$

is called the j-*th subresultant* of F and G.

Theorem 4.1 (Fundamental Theorem of Subresultants [6])**.** *Let F and G be defined as in (4.1), $(P_1,\dots,P_k) = \mathrm{prs}(F,G)$ be complete PRS, and $((\alpha_3,\beta_3),\dots,(\alpha_k,\beta_k))$ be its division rule. Let $n_i = \deg(P_i)$ and $c_i = \mathrm{lc}(P_i)$ for $i = 1,\dots,k$, and $d_i = n_i - n_{i+1}$ for $i = 1,\dots,k-1$. Then, we have*

$$S_j(F,G) = 0 \quad \textit{for } 0 \leqslant j < n_k, \tag{4.2}$$

$$S_{n_i}(F,G) = P_i c_i^{d_{i-1}-1} \prod_{l=3}^{i} \left\{ \left(\frac{\beta_l}{\alpha_l}\right)^{n_{l-1}-n_i} c_{l-1}^{d_{l-2}+d_{l-1}} \times (-1)^{(n_{l-2}-n_i)(n_{l-1}-n_i)} \right\}, \tag{4.3}$$

$$S_j(F,G) = 0 \quad \textit{for } n_i < j < n_{i-1}-1, \tag{4.4}$$

$$S_{n_{i-1}-1}(F,G) = P_i c_{i-1}^{1-d_{i-1}} \prod_{l=3}^{i} \left\{ \left(\frac{\beta_l}{\alpha_l}\right)^{n_{l-1}-n_{i-1}+1} c_{l-1}^{d_{l-2}+d_{l-1}} \times (-1)^{(n_{l-2}-n_{i-1}+1)(n_{l-1}-n_{i-1}+1)} \right\}, \tag{4.5}$$

for $i = 3,\dots,k$. □

By the Fundamental Theorem of subresultants, we can express coefficients of PRS by determinants of matrices whose elements are the coefficients of initial polynomials.

4.2.2 Recursive Subresultants

We construct "recursive subresultant matrix" whose determinants represent elements of recursive PRS by the coefficients of initial polynomials. To help the readers, we first show an example of recursive subresultant matrix for the recursive Sturm sequence in Example 4.1.

Example 4.2 (Recursive Subresultant Matrix). We express $P(x)$ and $\frac{d}{dx}P(x)$ in Example 4.1 by

$$P(x) = f_8x^8 + \cdots + f_0x^0, \quad \frac{d}{dx}P(x) = g_7x^7 + \cdots + g_0x^0.$$

Let $\bar{N}^{(1,5)}(F,G) = N^{(5)}(F,G)$, then the matrices $\bar{N}_U^{(1,5)}(F,G)$, $\bar{N}_L^{(1,5)}(F,G)$ and $\bar{N}_L'^{(1,5)}(F,G)$ are given as

$$\bar{N}^{(1,5)}(F,G) = \left(\begin{array}{c} \bar{N}_U^{(1,5)} \\ \hline \bar{N}_L^{(1,5)} \end{array}\right) = \left(\begin{array}{ccccc} f_8 & & g_7 & & \\ f_7 & f_8 & g_6 & g_7 & \\ f_6 & f_7 & g_5 & g_6 & g_7 \\ f_5 & f_6 & g_4 & g_5 & g_6 \\ \hline f_4 & f_5 & g_3 & g_4 & g_5 \\ f_3 & f_4 & g_2 & g_3 & g_4 \\ f_2 & f_3 & g_1 & g_2 & g_3 \\ f_1 & f_2 & g_0 & g_1 & g_2 \\ f_0 & f_1 & & g_0 & g_1 \\ & f_0 & & & g_0 \end{array}\right),$$

$$\bar{N}_L'^{(1,5)}(F,G) = \begin{pmatrix} 5f_4 & 5f_5 & 5g_3 & 5g_4 & 5g_5 \\ 4f_3 & 4f_4 & 4g_2 & 4g_3 & 4g_4 \\ 3f_2 & 3f_3 & 3g_1 & 3g_2 & 3g_3 \\ 2f_1 & 2f_2 & 2g_0 & 2g_1 & 2g_2 \\ f_0 & f_1 & & g_0 & g_1 \end{pmatrix},$$

where horizontal lines in matrices divide them into the upper and the lower components. Note that the matrix $\bar{N}'^{(1,5)}(F,G)$ is derived from $\bar{N}_L^{(1,5)}(F,G)$ by multiplying the l-th row by $6-l$ for $l = 1,\dots,5$ and deleting the bottom row. Then, the $(2,3)$-th recursive subresultant matrix $\bar{N}^{(2,3)}(F,G)$ is constructed as

$$\bar{N}^{(2,3)}(F,G) = \left(\begin{array}{c|c|c} \bar{N}_U^{(1,5)} & & \\ \hline & \bar{N}_U^{(1,5)} & \\ \hline & & \bar{N}_U^{(1,5)} \\ \hline & & 0\cdots0 \\ \hline \bar{N}_L^{(1,5)} & \bar{N}_L'^{(1,5)} & \\ & & \bar{N}_L'^{(1,5)} \\ & 0\cdots0 & \end{array}\right). \tag{4.6}$$

Definition 4.6 (Recursive Subresultant Matrix)**.** Let F and G be defined as in (4.1), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. Then, for each pair of numbers (k,j) with $k = 1,\dots,t$ and $j = j_{k-1}-2,\dots,0$, define matrix $\bar{N}^{(k,j)} = \bar{N}^{(k,j)}(F,G)$ recursively as follows.

1. For $k = 1$, let $\bar{N}^{(1,j)}(F,G) = N^{(j)}(F,G)$.

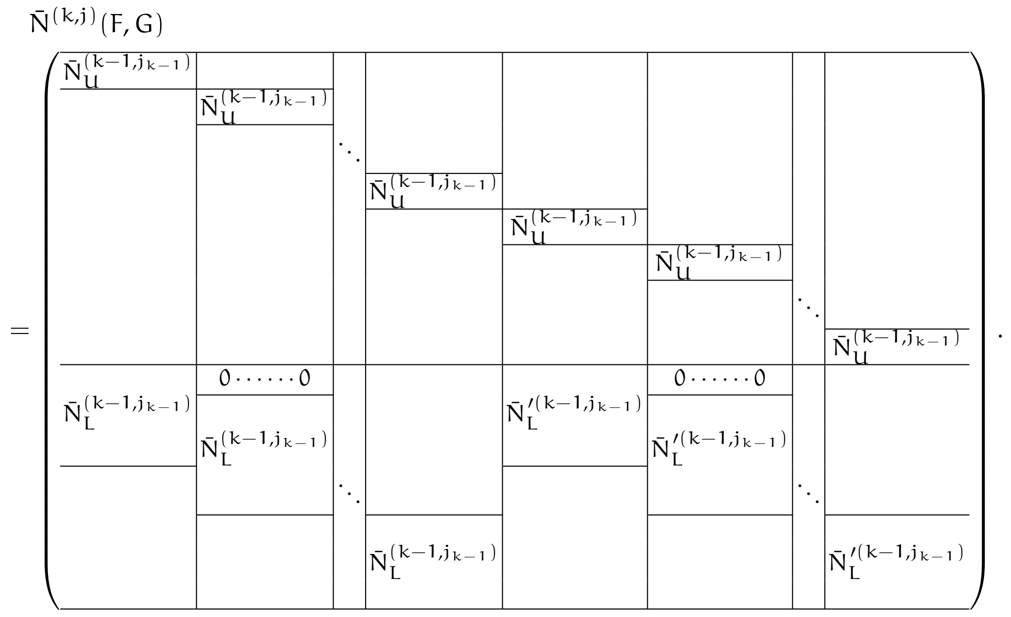
$\bar{N}^{(k,j)}(F,G)$
$=$
$\bar{N}_U^{(k-1,j_{k-1})}$
$\bar{N}_U^{(k-1,j_{k-1})}$
$\bar{N}_U^{(k-1,j_{k-1})}$
$\bar{N}_U^{(k-1,j_{k-1})}$
$\bar{N}_U^{(k-1,j_{k-1})}$
$\bar{N}_U^{(k-1,j_{k-1})}$
$0 \cdots\cdots 0$
$0 \cdots\cdots 0$
$\bar{N}_L^{(k-1,j_{k-1})}$
$\bar{N}_L^{(k-1,j_{k-1})}$
$\bar{N}_L^{(k-1,j_{k-1})}$
$\bar{N}_L'^{(k-1,j_{k-1})}$
$\bar{N}_L'^{(k-1,j_{k-1})}$
$\bar{N}_L'^{(k-1,j_{k-1})}$

Figure 4.1: Illustration of $\bar{N}^{(k,j)}(F,G)$. Note that the number of blocks $\bar{N}_L^{(k-1,j_{k-1})}$ is $j_{k-1} - j - 1$, whereas that of $\bar{N}_L'^{(k-1,j_{k-1})}$ is $j_{k-1} - j$; see Definition 4.6 for details.

2. For $k > 1$, let $\bar{N}^{(k,j)}(F,G)$ consist of the upper block and the lower block, defined as follows:
   a) The upper block is partitioned into $(2j_{k-1} - 2j - 1) \times (2j_{k-1} - 2j - 1)$ blocks with the diagonal blocks filled with $\bar{N}_U^{(k-1,j_{k-1})}$, where $\bar{N}_U^{(k-1,j_{k-1})}$ is a sub-matrix of $\bar{N}^{(k-1,j_{k-1})}(F,G)$ obtained by deleting the bottom $j_{k-1} + 1$ rows.
   b) Let $\bar{N}_L^{(k-1,j_{k-1})}$ be a sub-matrix of $\bar{N}^{(k-1,j_{k-1})}$ obtained by taking the bottom $j_{k-1} + 1$ rows, and let $\bar{N}_L'^{(k-1,j_{k-1})}$ be a sub-matrix of $\bar{N}_L^{(k-1,j_{k-1})}$ by multiplying the $(j_{k-1} + 1 - \tau)$-th rows by τ for $\tau = j_{k-1}, \dots, 1$, then by deleting the bottom row. Then, the lower block consists of $j_{k-1} - j - 1$ blocks of $\bar{N}_L^{(k-1,j_{k-1})}$ such that the leftmost block is placed at the top row of the container block and the right-side block is placed down by 1 row from the left-side block, then followed by $j_{k-1} - j$ blocks of $\bar{N}_L'^{(k-1,j_{k-1})}$ placed by the same manner as $\bar{N}_L^{(k-1,j_{k-1})}$.

As a result, $\bar{N}^{(k,j)}(F,G)$ becomes as shown in Fig. 4.1. Then, $\bar{N}^{(k,j)}(F,G)$ is called the (k,j)*-th recursive subresultant matrix* of F and G.

Proposition 4.2. *The numbers of rows and columns of* $\bar{N}^{(k,j)}(F,G)$*, the* (k,j)*-th recursive subresultant matrix of* F *and* G*, are as follows: for* $k = 1$ *and* $j < n$*, they are equal to*

$$m + n - j \quad \textit{and} \quad m + n - 2j, \tag{4.7}$$

respectively, and, for $(k,j)=(1,j_1)$ *and* $k=2,\dots,t$ *and* $j<j_{k-1}-1$*, they are equal to*

$$(m+n-2j_1)\left\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\right\}(2j_{k-1}-2j-1)+j \tag{4.8}$$

and

$$(m+n-2j_1)\left\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\right\}(2j_{k-1}-2j-1), \tag{4.9}$$

respectively, with $j_0=j_1+1$ *for* $(k,j)=(1,j_1)$.

Proof. By induction on k. For $k=1$, (4.7) immediately follows from Case 1 of Definition 4.6, and we also have (4.8) and (4.9) for $(k,j)=(1,j_1)$. Let us assume that we have (4.8) and (4.9) for $1,\dots,k-1$. Then, we calculate the numbers of the rows and columns of $\bar{N}^{(k,j)}(F,G)$ as follows.

1. The numbers of rows of $\bar{N}_L^{(k-1,j_{k-1})}$ and $\bar{N}_L^{\prime(k-1,j_{k-1})}$ are equal to $j_{k-1}+1$ and j_{k-1}, respectively, thus the number of rows a block which consists of $\bar{N}_L^{(k-1,j_{k-1})}$ and $\bar{N}_L^{\prime(k-1,j_{k-1})}$ in $\bar{N}^{(k,j)}(F,G)$ equals
$$2j_{k-1}-j-1. \tag{4.10}$$
On the other hand, the number of rows of $\bar{N}_U^{(k-1,j_{k-1})}$ is equal to $(m+n-2j_1)\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\}-1$, thus the number of rows of diagonal blocks in $\bar{N}^{(k,j)}(F,G)$ is equal to
$$\left\{(m+n-2j_1)\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)-1\right\}(2j_{k-1}-2j-1). \tag{4.11}$$
By adding (4.10) and (4.11), we obtain (4.8).
2. The number of columns of $\bar{N}^{(k-1,j_{k-1})}(F,G)$ is equal to $(m+n-2j_1)\times\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\}$, thus the number of columns of $\bar{N}^{(k,j)}(F,G)$ is equal to (4.9).

This proves the proposition. □

Now, we define the recursive subresultant.

Definition 4.7 (Recursive Subresultant)**.** Let F and G be defined as in (4.1), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. For $j=j_{k-1}-2,\dots,0$ and $\tau=j,\dots,0$, let $\bar{N}_\tau^{(k,j)}=\bar{N}_\tau^{(k,j)}(F,G)$ be a sub-matrix of the (k,j)-th recursive subresultant matrix $\bar{N}^{(k,j)}(F,G)$ obtained by taking the top $(m+n-2j_1)\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\}(2j_{k-1}-2j-1)-1$ rows and the $\left((m+n-2j_1)\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\}(2j_{k-1}-2j-1)+j-\tau\right)$-th row (note that $\bar{N}_\tau^{(k,j)}$ is a square matrix). Then, the polynomial

$$\bar{S}_{k,j}(F,G)=|\bar{N}_j^{(k,j)}|x^j+\cdots+|\bar{N}_0^{(k,j)}|x^0$$

is called the (k,j)-*th recursive subresultant* of F and G.

We show the relationship between recursive subresultants and coefficients in the recursive PRS.

Lemma 4.3. *Let* F *and* G *be defined as in* (4.1)*, and let* $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ *be complete recursive PRS for* F *and* G *as in Definition* 4.2*. For* $k=1,\dots,t-1$*, define*

$$\begin{aligned}B_k =&(c_{l_k}^{(k)})^{d_{l_k-1}^{(k)}-1}\prod_{l=3}^{l_k}\Bigg\{\left(\frac{\beta_l^{(k)}}{\alpha_l^{(k)}}\right)^{n_{l-1}^{(k)}-n_{l_k}^{(k)}}\\ &\times(c_{l-1}^{(k)})^{(d_{l-2}^{(k)}+d_{l-1}^{(k)})}(-1)^{(n_{l-2}^{(k)}-n_{l_k}^{(k)})(n_{l-1}^{(k)}-n_{l_k}^{(k)})}\Bigg\}.\end{aligned}$$

For $k=2,\dots,t$ *and* $j=j_{k-1}-2,\dots,0$*, define*

$$\begin{aligned}u_{k,j} =&(m+n-2j_1)\left\{\prod_{l=2}^{k-1}(2j_{l-1}-2j_l-1)\right\}(2j_{k-1}-2j-1)\\ &\textit{with } u_k=u_{k,j_k} \textit{ and } u_1=m+n-2j_1,\\ b_{k,j} =&2j_{k-1}-2j-1 \textit{ with } b_k=b_{k,j_k} \textit{ and } b_{1,j}=1 \textit{ for } j<n,\\ r_{k,j} =&(-1)^{(u_{k-1}-1)(1+2+\cdots+(b_{k,j}-1))} \textit{ with } r_k=r_{k,j_k} \textit{ and } r_{1,j}=1 \textit{ for } j<n,\\ \bar{R}_k =&(\bar{R}_{k-1})^{b_k}r_kB_k \textit{ with } \bar{R}_0=1.\end{aligned}$$

Then, for the (k,j)*-th recursive subresultant of* F *and* G *with* $k=1,\dots,t$ *and* $j=j_{k-1}-2,\dots,0$*, we have*

$$\bar{S}_{k,j}(F,G)=(\bar{R}_{k-1})^{b_{k,j}}r_{k,j}\cdot S_j(P_1^{(k)},P_2^{(k)}). \tag{4.12}$$

To prove Lemma 4.3, we prove the following lemma.

Lemma 4.4. *For* $k=1,\dots,t$*,* $j=j_{k-1}-2,\dots,0$ *and* $\tau=j,\dots,0$*, we have*

$$|\bar{N}_\tau^{(k,j)}(F,G)|=(\bar{R}_{k-1})^{b_{k,j}}r_{k,j}\,|N_\tau^{(j)}(P_1^{(k)},P_2^{(k)})|.$$

Proof. By induction on k. For $k=1$, it is obvious from Case 1 in Definition 4.6. Let us assume that the lemma is valid for $1,\dots,k-1$, then we prove the claim for k by the following steps.

Lemma 4.5. *Assume that we have Lemma* 4.4 *for* $1,\dots,k-1$*. Then, for* k*,* $j=j_{k-1}-2,\dots,0$ *and* $\tau=j,\dots,0$*,* $\bar{N}^{(k,j)}(F,G)$ *can be transformed by certain eliminations and permutations on its columns into* $M^{(k,j)}(F,G)$ *as shown in Fig.* 4.2*, satisfying*

$$|\bar{N}_\tau^{(k,j)}(F,G)|=((\bar{R}_{k-2})^{b_{k-1}}r_{k-1})^{b_{k,j}}|M_\tau^{(k,j)}(F,G)|, \tag{4.13}$$

where $M_\tau^{(k,j)}(F,G)$ *is a sub-matrix of* $M^{(k,j)}(F,G)$ *obtained by the same manner as we have obtained* $\bar{N}_\tau^{(k,j)}(F,G)$ *from* $\bar{N}^{(k,j)}(F,G)$ *in Definition* 4.7*.*

Proof. By the induction hypothesis, for $\tau'=j_{k-1},\dots,0$, we have

$$|\bar{N}_{\tau'}^{(k-1,j_{k-1})}(F,G)|=(\bar{R}_{k-2})^{b_{k-1}}r_{k-1}\,|N_{\tau'}^{(j_{k-1})}(P_1^{(k-1)},P_2^{(k-1)})|.$$

Let $\bar{N}'^{(k,j)}(F,G)$ be a matrix defined as

$$\bar{N}'^{(k,j)}(F,G)=\left(\frac{\bar{N}_U^{(k,j)}}{\bar{N}_L'^{(k,j)}}\right),$$

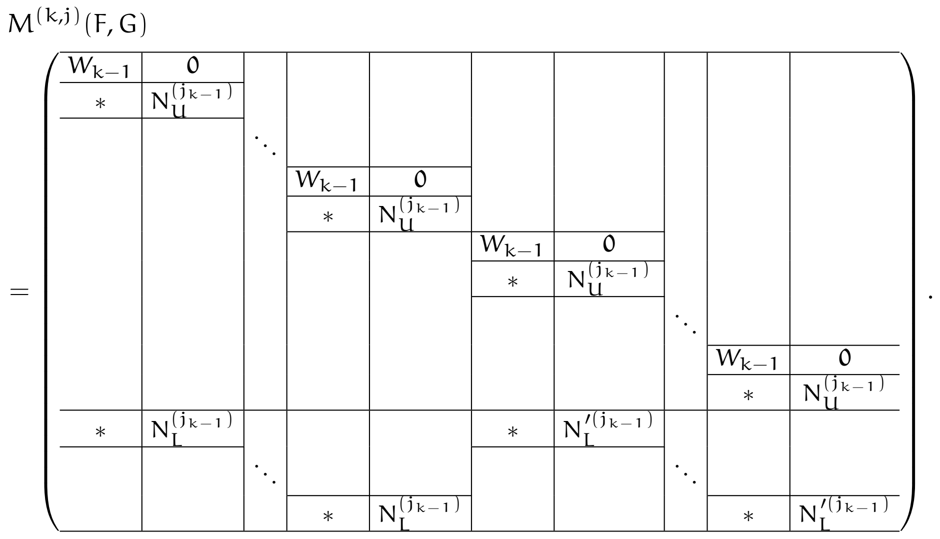

Figure 4.2: Illustration of $M^{(k,j)}(F,G)$. Note that the number of column blocks is equal to $b_{k,j} = 2j_{k-1} - 2j - 1$; see Lemma 4.5 for details.

where $\bar{N}_U^{(k,j)}$ and $\bar{N}_L'^{(k,j)}$ are defined as in Definition 4.6. Furthermore, let $N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ be defined as $N^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ with the $(j_{k-2} + 1 - \tau)$-th row multiplied by τ for $\tau = j_{k-1}, \dots, 1$, then by deleting the bottom row, $N_U^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ be a sub-matrix of $N^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ and $N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ by taking the top $j_{k-2} - j_{k-1}$ rows, and $N_L^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ and $N_L'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ be sub-matrices of $N^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ and $N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$, respectively, by eliminating the top $j_{k-2} - j_{k-1}$ rows.

Then, by certain eliminations and exchanges on columns, we can transform $\bar{N}^{(k-1,j_{k-1})}(F,G)$ and $\bar{N}'^{(k-1,j_{k-1})}(F,G)$ to

$$\begin{aligned}
D^{(k-1,j_{k-1})}(F,G) &= \left(\begin{array}{c|c} W_{k-1} & 0 \\ \hline * & N^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)}) \end{array}\right) \\
&= \left(\begin{array}{c|c} W_{k-1} & 0 \\ \hline * & N_U{}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)}) \\ \hline * & N_L{}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)}) \end{array}\right), \\
D'^{(k-1,j_{k-1})}(F,G) &= \left(\begin{array}{c|c} W_{k-1} & 0 \\ \hline * & N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)}) \end{array}\right) \\
&= \left(\begin{array}{c|c} W_{k-1} & 0 \\ \hline * & N_U{}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)}) \\ \hline * & N_L{}'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)}) \end{array}\right),
\end{aligned} \tag{4.14}$$

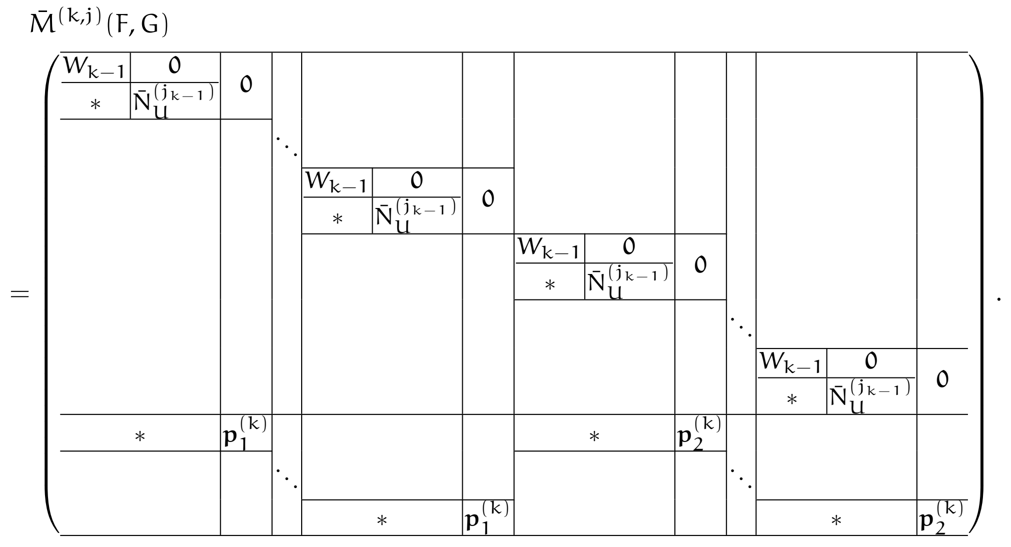

Figure 4.3: Illustration of $\bar{M}^{(k,j)}(F,G)$; see Lemma 4.6 for details.

respectively, satisfying

$$\begin{aligned}|W_{k-1}| &= 1,\\ |D_{\tau'}^{(k-1,j_{k-1})}(F,G)| &= (\bar{R}_{k-2})^{b_{k-1}} r_{k-1} |N_{\tau'}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})|,\\ |D'^{(k-1,j_{k-1})}_{\tau'}(F,G)| &= (\bar{R}_{k-2})^{b_{k-1}} r_{k-1} |N'^{(j_{k-1})}_{\tau'}(P_1^{(k-1)}, P_2^{(k-1)})|,\end{aligned}$$

where $D_{\tau'}^{(k-1,j_{k-1})}(F,G)$ and $D'^{(k-1,j_{k-1})}_{\tau'}(F,G)$ are sub-matrices of $D^{(k-1,j_{k-1})}(F,G)$ and $D'^{(k-1,j_{k-1})}(F,G)$, respectively, obtained by the same manner as we have obtained $\bar{N}_{\tau'}^{(k-1,j_{k-1})}(F,G)$ from $\bar{N}^{(k-1,j_{k-1})}(F,G)$ (see Definition 4.7).

Therefore, by the above transformations on the columns in each column blocks in $\bar{N}^{(k,j)}(F,G)$ as shown in Fig. 4.1, we obtain $M^{(k,j)}(F,G)$ as shown in Fig. 4.2, where $N_U{}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$, $N_L{}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ and $N_L'{}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ are abbreviated to $N_U{}^{(j_{k-1})}$, $N_L{}^{(j_{k-1})}$ and $N_L'{}^{(j_{k-1})}$, respectively, satisfying (4.13) because $M^{(k,j)}(F,G)$ has $b_{k,j} = 2j_{k-1} - 2j - 1$ column blocks that have $D^{(k-1,j_{k-1})}(F,G)$ or $D'^{(k-1,j_{k-1})}(F,G)$. This proves the lemma. □

Lemma 4.6. *For k, $j = j_{k-1} - 2, \dots, 0$ and $\tau = j, \dots, 0$, $M^{(k,j)}(F,G)$ can be transformed by certain eliminations and permutations on its columns into $\bar{M}^{(k,j)}(F,G)$ as shown in Fig. 4.3, satisfying*

$$|M_\tau^{(k,j)}(F,G)| = (B_{k-1})^{b_{k,j}} |\bar{M}_\tau^{(k,j)}(F,G)|, \tag{4.15}$$

where $\bar{M}_\tau^{(k,j)}$ is a sub-matrix of $\bar{M}^{(k,j)}$ obtained by the same manner as we have obtained $\bar{N}_{\tau'}^{(k-1,j_{k-1})}$ from $\bar{N}^{(k-1,j_{k-1})}$ in Definition 4.7.

Proof. For $N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ (defined as in the proof of Lemma 4.5) and $\tau'' = j_{k-1} - 1, \dots, 0$, let $N'^{(j_{k-1})}_{\tau''}(P_1^{(k-1)}, P_2^{(k-1)})$ be a sub-matrix of

$N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ obtained by taking the top $2(n_1^{(k-1)} - j_{k-1} - 1)$ rows and the $(2n_1^{(k-1)} - 2 - j_{k-1} - \tau'')$-th row. Then, by the Fundamental Theorem of subresultants (Theorem 4.1), we have

$$\begin{aligned}
&|N_{j_{k-1}}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})|x^{j_{k-1}} + \cdots + |N_0^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})|x^0 \\
=&S_{j_{k-1}}(P_1^{(k-1)}, P_2^{(k-1)}) = B_{k-1}P_{l_{k-1}}^{(k-1)} = B_{k-1}P_1^{(k)}, \\
&|N'^{(j_{k-1})-1}_{j_{k-1}}(P_1^{(k-1)}, P_2^{(k-1)})|x^{j_{k-1}-1} + \cdots + |N'^{(j_{k-1})}_0(P_1^{(k-1)}, P_2^{(k-1)})|x^0 \\
=&\frac{d}{dx}S_{j_{k-1}}(P_1^{(k-1)}, P_2^{(k-1)}) = B_{k-1}\frac{d}{dx}P_{l_{k-1}}^{(k-1)} = B_{k-1}P_2^{(k)},
\end{aligned}$$

thus, for $\tau' = j_{k-1}, \dots, 0$ and $\tau'' = j_{k-1} - 1, \dots, 0$, we have

$$|N_{\tau'}^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})| = B_{k-1}a_{1,\tau'}^{(k)}, \quad |N'^{(j_{k-1})}_{\tau''}(P_1^{(k-1)}, P_2^{(k-1)})| = B_{k-1}a_{2,\tau''}^{(k)},$$

where $a_{i,j}^{(k)}$ represents the coefficient of degree j of $P_i^{(k)}$ (see Remark 4.1). Therefore, by certain eliminations and exchanges on columns, we can transform $N^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ and $N'^{(j_{k-1})}(P_1^{(k-1)}, P_2^{(k-1)})$ into

$$\left(\begin{array}{c|c} \bar{N}_U^{(j_{k-1})} & 0 \\ \hline * & \mathbf{p}_1^{(k)} \end{array}\right) \quad \text{and} \quad \left(\begin{array}{c|c} \bar{N}_U^{(j_{k-1})} & 0 \\ \hline * & \mathbf{p}_2^{(k)} \end{array}\right),$$

respectively, satisfying $|\bar{N}_U^{(j_{k-1})}| = 1$ (see Remark 4.1 for the notation of "coefficient vectors"). By these transformations, we can transform $D^{(k-1,j_{k-1})}(F,G)$ and $D'^{(k-1,j_{k-1})}(F,G)$ in (4.14) to

$$\bar{D}^{(k-1,j_{k-1})}(F,G) = \left(\begin{array}{c|cc} W_{k-1} & \multicolumn{2}{c}{0} \\ \hline & \bar{N}_U^{(j_{k-1})} & \\ \cline{2-3} * & & \mathbf{p}_1^{(k)} \end{array}\right),$$

$$\bar{D}'^{(k-1,j_{k-1})}(F,G) = \left(\begin{array}{c|cc} W_{k-1} & \multicolumn{2}{c}{0} \\ \hline & \bar{N}_U^{(j_{k-1})} & \\ \cline{2-3} * & & \mathbf{p}_2^{(k)} \end{array}\right),$$

respectively, satisfying

$$\begin{aligned}
|\bar{N}_U^{(j_{k-1})}| &= 1, \\
|\bar{D}_{\tau'}^{(k-1,j_{k-1})}(F,G)| &= B_{k-1}|D_{\tau'}^{(k-1,j_{k-1})}(F,G)|, \\
|\bar{D}'^{(k-1,j_{k-1})}_{\tau'}(F,G)| &= B_{k-1}|D'^{(k-1,j_{k-1})}_{\tau'}(F,G)|,
\end{aligned}$$

where $D_{\tau'}^{(k-1,j_{k-1})}$, $D'^{(k-1,j_{k-1})}_{\tau'}$, $\bar{D}_{\tau'}^{(k-1,j_{k-1})}$ and $\bar{D}'^{(k-1,j_{k-1})}_{\tau'}$ are sub-matrices of $D^{(k-1,j_{k-1})}$, $D'^{(k-1,j_{k-1})}$, $\bar{D}^{(k-1,j_{k-1})}$ and $\bar{D}'^{(k-1,j_{k-1})}$, respectively, obtained by the same manner as we have obtained $\bar{N}_{\tau'}^{(k-1,j_{k-1})}$ from $\bar{N}^{(k-1,j_{k-1})}$. Therefore, by the above eliminations on the columns in each column blocks, we can transform $M^{(k,j)}(F,G)$ to $\bar{M}^{(k,j)}(F,G)$ as shown in Fig. 4.3 satisfying (4.15) because $M^{(k,j)}(F,G)$ has $b_{k,j} = 2j_{k-1} - 2j - 1$ column blocks that have $D^{(k-1,j_{k-1})}(F,G)$ or $D'^{(k-1,j_{k-1})}(F,G)$. This proves the lemma. □

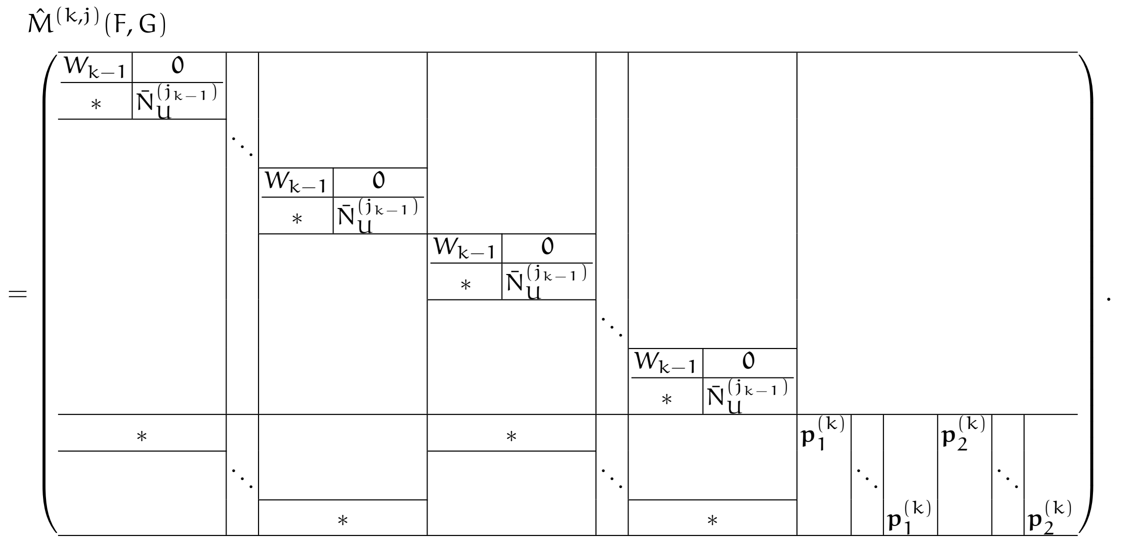

Figure 4.4: Illustration of $\hat{M}^{(k,j)}(F,G)$. Note that the lower-right block which consists of $\mathbf{p}_1^{(k)}$ and $\mathbf{p}_2^{(k)}$ is equal to $N^{(j)}(P_1^{(k)}, P_2^{(k)})$ and $|W_{k-1}| = |\bar{N}_U^{(j_{k-1})}| = 1$; see Lemma 4.4 for details.

Proof of Lemma 4.4 (continued). By exchanges on column blocks, we can transform $\bar{M}^{(k,j)}(F,G)$ to $\hat{M}^{(k,j)}(F,G)$ as shown in Fig. 4.4, with

$$|\bar{M}_\tau^{(k,j)}(F,G)| = r_{k,j}\, |\hat{M}_\tau^{(k,j)}(F,G)|, \tag{4.16}$$

where $\hat{M}_\tau^{(k,j)}$ is a sub-matrix of $\hat{M}^{(k,j)}$ obtained by the same manner as we have obtained $\bar{N}_\tau^{(k,j)}$ from $\bar{N}^{(k,j)}$, because the $(u_{k,j} - (l-1)u_{k-1})$-th column in $\bar{M}^{(k,j)}(F,G)$ was moved to the $(u_{k,j} - (l-1))$-th column in $\hat{M}^{(k,j)}(F,G)$ for $l = 1, \ldots, b_{k,j}$. (Note that $\hat{M}^{(k,j)}(F,G)$ is a block lower triangular matrix.) Then, we have

$$|\hat{M}_\tau^{(k,j)}(F,G)| = |N_\tau^{(j)}(P_1^{(k)}, P_2^{(k)})|, \tag{4.17}$$

because we have $|W_{k-1}| = |\bar{N}_U^{(j_{k-1})}| = 1$ and the lower-right block of $\mathbf{p}_1^{(k)}$s and $\mathbf{p}_2^{(k)}$s in $\hat{M}^{(k,j)}(F,G)$ is equal to $N^{(j)}(P_1^{(k)}, P_2^{(k)})$.

Finally, from (4.13), (4.15), (4.16) and (4.17), we have

$$\begin{aligned}|\bar{N}_\tau^{(k,j)}(F,G)| &= ((\bar{R}_{k-2})^{b_{k-1}} r_{k-1})^{b_{k,j}} (B_{k-1})^{b_{k,j}} r_{k,j} |N_\tau^{(j)}(P_1^{(k)}, P_2^{(k)})| \\ &= (\bar{R}_{k-1})^{b_{k,j}} r_{k,j} |N_\tau^{(j)}(P_1^{(k)}, P_2^{(k)})|,\end{aligned}$$

which proves the lemma. □

Theorem 4.7. *With the same conditions as in Lemma 4.3, and for* $k = 1, \dots, t$ *and* $i = 3, 4, \dots, l_k$*, we have*

$$\bar{S}_{k,j}(F,G) = 0 \quad \textit{for } 0 \leqslant j < n_{l_k}^{(k)}, \tag{4.18}$$

$$\begin{aligned}\bar{S}_{k,n_i^{(k)}}(F,G) &= P_i^{(k)} (c_i^{(k)})^{d_{i-1}^{(k)}-1} (R_{k-1})^{b_{k,n_i^{(k)}}} r_{k,n_i^{(k)}} \\ &\times \prod_{l=3}^{i} \left\{ \left(\frac{\beta_l^{(k)}}{\alpha_l^{(k)}} \right)^{n_{l-1}^{(k)}-n_i^{(k)}} (c_{l-1}^{(k)})^{(d_{l-2}^{(k)}+d_{l-1}^{(k)})} (-1)^{(n_{l-2}^{(k)}-n_i^{(k)})(n_{l-1}^{(k)}-n_i^{(k)})} \right\},\end{aligned} \tag{4.19}$$

$$\bar{S}_{k,j}(F,G) = 0 \quad \textit{for } n_i^{(k)} < j < n_{i-1}^{(k)} - 1, \tag{4.20}$$

$$\begin{aligned}\bar{S}_{k,n_{i-1}^{(k)}-1}(F,G) &= P_i^{(k)} (c_{i-1}^{(k)})^{1-d_{i-1}^{(k)}} (R_{k-1})^{b_{k,n_{i-1}^{(k)}-1}} r_{k,n_{i-1}^{(k)}-1} \\ &\times \prod_{l=3}^{i} \left\{ \left(\frac{\beta_l^{(k)}}{\alpha_l^{(k)}} \right)^{n_{l-1}^{(k)}-n_{i-1}^{(k)}+1} (c_{l-1}^{(k)})^{(d_{l-2}^{(k)}+d_{l-1}^{(k)})} \right. \\ &\left. \times (-1)^{(n_{l-2}^{(k)}-n_{i-1}^{(k)}+1)(n_{l-1}^{(k)}-n_{i-1}^{(k)}+1)} \right\}.\end{aligned} \tag{4.21}$$

Proof. By substituting $S_j(P_1^{(k)}, P_2^{(k)})$ in (4.12) by (4.2)–(4.5), we obtain (4.18)–(4.21), respectively. □

We show an example of the proof of Lemma 4.3 for the recursive subresultant matrix in Example 4.2.

Example 4.3. (Continued from Example 4.2.) Since we have $\bar{N}^{(1,5)}(F,G) = N^{(5)}(F,G)$, we can regard $\bar{N}^{(1,5)}(F,G)$ and $\bar{N}'^{(1,5)}(F,G)$ as $D^{(1,5)}(F,G)$ and $D'^{(1,5)}(F,G)$ in (4.14), respectively. Then, by eliminations and exchanges of columns as shown in Lemma 4.6, we can transform $\bar{N}^{(1,5)}(F,G) = \begin{pmatrix} \bar{N}_U^{(1,5)} \\ \bar{N}_L^{(1,5)} \end{pmatrix}$ and $\bar{N}'^{(1,5)}(F,G) = \begin{pmatrix} \bar{N}_U^{(1,5)} \\ \bar{N}_L'^{(1,5)} \end{pmatrix}$ in (4.6) to $\bar{D}^{(1,5)}(F,G)$ and $\bar{D}'^{(1,5)}(F,G)$, respectively, as

$$\bar{D}^{(1,5)}(F,G) = \left(\begin{array}{c|c} \bar{N}_U^{(5)} & 0 \\ \hline * & \mathbf{p}_1^{(2)} \end{array} \right), \quad \bar{D}'^{(1,5)}(F,G) = \left(\begin{array}{c|c} \bar{N}_U^{(5)} & 0 \\ \hline * & \mathbf{p}_2^{(2)} \end{array} \right),$$

with $|\bar{N}_U^{(5)}| = 1$ and $B_1 = (a_{2,7}^{(1)})^2 (a_{3,6}^{(1)})^2$. Therefore, by the above transformations of columns in each column blocks in $\bar{N}^{(2,1)}(F,G)$, we have

$$\bar{M}^{(2,3)}(F,G) = \left(\begin{array}{c|c|c|c|c|c} \bar{N}_U^{(5)} & 0 & & & & \\ \cline{1-2} & & \bar{N}_U^{(5)} & 0 & & \\ \cline{3-4} & & & & \bar{N}_U^{(5)} & 0 \\ \cline{1-6} & & & & \multicolumn{2}{c}{0 \cdots 0} \\ \cline{5-6} * & \mathbf{p}_1^{(2)} & * & \mathbf{p}_2^{(2)} & & \\ & & & & * & \mathbf{p}_2^{(2)} \\ \cline{3-4} & & \multicolumn{2}{c|}{0 \cdots 0} & & \end{array} \right),$$

satisfying $|\bar{N}_\tau^{(2,3)}(F,G)| = (B_1)^3\ |\bar{M}_\tau^{(2,3)}(F,G)|$ for $\tau = 3,\dots,0$. Furthermore, by exchanges of columns, we can transform $\bar{M}^{(2,3)}(F,G)$ to $\hat{M}^{(2,3)}(F,G)$ as

$$\hat{M}^{(2,3)}(F,G) = \left(\begin{array}{c|c|c|c|c|c}
\bar{N}_u^{(5)} & & & & & \\ \cline{1-2}
 & \bar{N}_u^{(5)} & & & & \\ \cline{2-3}
 & & \bar{N}_u^{(5)} & & & \\ \hline
 & & 0\cdots0 & & & 0 \\ \cline{3-3}\cline{6-6}
* & * & * & \mathbf{p}_1^{(2)} & \mathbf{p}_2^{(2)} & \\
 & & & & & \mathbf{p}_2^{(2)} \\ \cline{2-2}\cline{5-5}
 & 0\cdots0 & & & 0 &
\end{array}\right)$$

$$= \left(\begin{array}{c|c|c|c}
\bar{N}_u^{(5)} & & & \\ \cline{1-2}
 & \bar{N}_u^{(5)} & & \\ \cline{2-3}
 & & \bar{N}_u^{(5)} & \\ \hline
\multicolumn{3}{c|}{*} & N^{(3)}(P_1^{(2)},P_2^{(2)})
\end{array}\right),$$

satisfying $|\bar{M}_\tau^{(2,3)}(F,G)| = r_{2,3}\ |\hat{M}_\tau^{(2,3)}(F,G)| = r_{2,3}\ |N_\tau^{(3)}(P_1^{(2)},P_2^{(2)})|$. Therefore, we have

$$|\bar{N}_\tau^{(2,3)}(F,G)| = (B_1)^3 r_{2,3}\ |N_\tau^{(3)}(P_1^{(2)},P_2^{(2)})| = (R_1)^3 r_{2,3}\ |N_\tau^{(3)}(P_1^{(2)},P_2^{(2)})|,$$

for $\tau = 3,\dots,0$, and we have

$$\bar{S}_{2,3}(F,G) = (R_1)^3 r_{2,3}\cdot S_3(P_1^{(2)},P_2^{(2)}) = \{(a_{2,7}^{(1)})^2(a_{3,6}^{(1)})^2\}^3(a_{2,4}^{(2)})^2\ P_3^{(2)}.$$

4.3 NESTED SUBRESULTANTS

As we have seen in the above, the recursive subresultant can represent the coefficients of the elements in the recursive PRS. However, the size of the recursive subresultant matrix increases rapidly as the recursion of the recursive PRS deepens, thus making use of the recursive subresultant matrix become inefficient.

To overcome this problem, we should introduce other representations for the subresultant that are equivalent to the recursive subresultant, and more suitable for efficient computations. The nested subresultant matrix is a subresultant matrix whose elements are again determinants of certain subresultant matrices (or even the nested subresultant matrices), and the nested subresultant is a subresultant whose coefficients are determinants of the nested subresultant matrices.

Note that the nested subresultant is mainly used to show the relationship between the recursive subresultant and the reduced nested subresultant that will be defined in the next section.

We show an example of a nested subresultant matrix.

Example 4.4. Let $F(x)$ and $G(x)$ be defined as

$$\begin{aligned} F(x) &= a_6x^6 + a_5x^5 + \cdots + a_0, \quad a_6 \neq 0,\\ G(x) &= b_5x^5 + b_4x^4 + \cdots + b_0, \quad b_5 \neq 0. \end{aligned}$$

Let $\mathrm{prs}(F,G) = (P_1^{(1)} = F,\ P_2^{(1)} = G,\ P_3^{(1)} = \gcd(F,G))$ with $\deg(P_3^{(1)}) = 4$, and let us consider recursive PRS for F and G.

Let $P_1^{(2)} = P_3^{(1)}$, $P_2^{(2)} = \frac{d}{dx}P_3^{(1)}$, and calculate a subresultant of degree 1, which corresponds to $P_4^{(2)}$. By the Fundamental Theorem of subresultants (Theorem 4.1), we have

$$S_4(F,G) = A_4x^4 + A_3x^3 + A_2x^2 + A_1x + A_0,$$
$$\frac{d}{dx}S_4(F,G) = 4A_4x^3 + 3A_3x^2 + 2A_2x + A_1,$$

where

$$A_j = |N_j^{(4)}(F,G)| \tag{4.22}$$

for $j = 0, \dots, 4$ with $N_k^{(j)}(F,G)$ as in Definition 4.4.

Then, we can express the subresultant matrix $N^{(2)}(S_4(F,G), \frac{d}{dx}S_4(F,G))$ as

$$N^{(2)}\left(S_4(F,G), \frac{d}{dx}S_4(F,G)\right) = \begin{pmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_2 & 2A_2 & 3A_3 \\ A_1 & A_1 & 2A_2 \\ A_0 & & A_1 \end{pmatrix}, \tag{4.23}$$

and the subresultant $S_2(S_4(F,G), \frac{d}{dx}S_4(F,G))$ as

$$S_2\left(S_4(F,G), \frac{d}{dx}S_4(F,G)\right) = \begin{vmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_2 & 2A_2 & 3A_3 \end{vmatrix} x^2 + \begin{vmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_1 & A_1 & 2A_2 \end{vmatrix} x + \begin{vmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_0 & & A_1 \end{vmatrix}, \tag{4.24}$$

respectively, with A_j as in (4.22). We see that the elements in (4.23) are minors of subresultant matrix, hence the coefficients in (4.24) is "nested" expression of determinants.

Definition 4.8 (Nested Subresultant Matrix)**.** Let F and G be defined as in (4.1), and let $(P_1^{(1)}, \dots, P_{l_1}^{(1)}, \dots, P_1^{(t)}, \dots, P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. Then, for each pair of numbers (k,j) with $k = 1, \dots, t$ and $j = j_{k-1} - 2, \dots, 0$, define matrix $\tilde{N}^{(k,j)}(F,G)$ recursively as follows.

1. For $k = 1$, let $\tilde{N}^{(1,j)}(F,G) = N^{(j)}(F,G)$.
2. For $k > 1$, let
$$\tilde{N}^{(k,j)}(F,G) = N^{(j)}\left(\tilde{S}_{k-1,j_{k-1}}(F,G), \frac{d}{dx}\tilde{S}_{k-1,j_{k-1}}(F,G)\right),$$
where $\tilde{S}_{k-1,j_{k-1}}(F,G)$ is defined by Definition 4.9. Then, $\tilde{N}^{(k,j)}(F,G)$ is called the (k,j)*-th nested subresultant matrix of* F *and* G.

Definition 4.9 (Nested Subresultant)**.** Let F and G be defined as in (4.1), and let $(P_1^{(1)}, \dots, P_{l_1}^{(1)}, \dots, P_1^{(t)}, \dots, P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. For $j = j_{k-1} - 2, \dots, 0$ and $\tau = j, \dots, 0$, let $\tilde{N}_\tau^{(k,j)} = \tilde{N}_\tau^{(k,j)}(F,G)$ be a sub-matrix of the (k,j)-th nested subresultant matrix $\tilde{N}^{(k,j)}(F,G)$ obtained by taking the top $n_1^{(k)} + n_2^{(k)} - 2j - 1$ rows and the $(n_1^{(k)} + n_2^{(k)} - j - \tau)$-th row (note that $\tilde{N}_\tau^{(k,j)}$ is a square matrix). Then, the polynomial

$$\tilde{S}_{k,j}(F,G) = |\tilde{N}_j^{(k,j)}|x^j + \cdots + |\tilde{N}_0^{(k,j)}|x^0$$

is called the (k,j)-*th nested subresultant* of F and G.

We show the relationship between the nested subresultant and the recursive subresultant.

Lemma 4.8. *Let F and G be defined as in (4.1), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. For $k=1,\dots,t-1$, define B_k, and for $k=2,\dots,t$ and $j=j_{k-1}-2,\dots,0$, define $b_{k,j}$ as in Lemma 4.3. Furthermore, for $k=2,\dots,t-1$, define $\tilde{R}_k=(\tilde{R}_{k-1})^{b_k}B_k$ with $\tilde{R}_0=\tilde{R}_1=1$. Then, we have*

$$\tilde{S}_{k,j}(F,G)=(\tilde{R}_{k-1})^{b_{k,j}}\cdot S_j(P_1^{(k)},P_2^{(k)}). \tag{4.25}$$

Proof. By induction on k. For $k=1$, it is obvious by the definition of the nested subresultant. Assume that (4.25) is valid for $1,\dots,k-1$. Then, by the Fundamental Theorem of subresultants (Theorem 4.1), we have

$$\begin{aligned}\tilde{S}_{k-1,j_{k-1}}(F,G)&=(\tilde{R}_{k-2})^{b_{k-1}}B_{k-1}P_{l_{k-1}}^{(k-1)}=(\tilde{R}_{k-1})P_1^{(k)},\\ \frac{d}{dx}\left(\tilde{S}_{k-1,j_{k-1}}(F,G)\right)&=(\tilde{R}_{k-1})\frac{d}{dx}\left(P_1^{(k)}\right)=(\tilde{R}_{k-1})P_2^{(k)}.\end{aligned}$$

Then, we have

$$\begin{aligned}\tilde{N}^{(k,j)}(F,G)&=(\tilde{R}_{k-1})\,N^{(j)}(P_1^{(k)},P_2^{(k)}),\\ |\tilde{N}_\tau^{(k,j)}(F,G)|&=(\tilde{R}_{k-1})^{b_{k,j}}\,|N_\tau^{(j)}(P_1^{(k)},P_2^{(k)})|,\end{aligned}$$

for $\tau=j,\dots,0$. Therefore, we have (4.25), which proves the lemma. □

Theorem 4.9. *Let F and G be defined as in (4.1), and let $(P_1^{(1)},\dots,P_{l_1}^{(1)},\dots,P_1^{(t)},\dots,P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. For $k=2,\dots,t$ and $j=j_{k-1}-2,\dots,0$, define $u_{k,j}$, $b_{k,j}$, $r_{k,j}$ as in Lemma 4.3 and $R'_k=(R'_{k-1})^{b_k}r_k$ with $R'_0=R'_1=1$. Then, we have*

$$\bar{S}_{k,j}(F,G)=(R'_{k-1})^{b_{k,j}}r_{k,j}\cdot\tilde{S}_{k,j}(F,G).$$

Proof. By induction on k. For $k=1$, it is obvious by the definitions of the recursive and the nested subresultants. We first show that $\bar{R}_k=\tilde{R}_k\cdot R'_k$ for $k=0,\dots,t-1$. It is obvious for $k=0$ and 1. Let us assume $\bar{R}_{k-1}=\tilde{R}_{k-1}\cdot R'_{k-1}$. Then, we have

$$\begin{aligned}\bar{R}_k&=(\bar{R}_{k-1})^{b_k}r_kB_k=(\tilde{R}_{k-1}\cdot R'_{k-1})^{b_k}r_kB_k\\ &=(\tilde{R}_{k-1})^{b_k}B_k\cdot(R'_{k-1})^{b_k}r_k=\tilde{R}_k\cdot R'_k.\end{aligned}$$

Now, by Lemma 4.3, we have $\bar{S}_{k,j}(F,G)=(\bar{R}_{k-1})^{b_{k,j}}r_{k,j}\cdot S_j(P_1^{(k)},P_2^{(k)})$, then, by Lemma 4.8, we have

$$\begin{aligned}\bar{S}_{k,j}(F,G)&=(\tilde{R}_{k-1}\cdot R'_{k-1})^{b_{k,j}}r_{k,j}\cdot S_j(P_1^{(k)},P_2^{(k)})\\ &=(R'_{k-1})^{b_{k,j}}r_{k,j}\cdot(\tilde{R}_{k-1})^{b_{k,j}}\cdot S_j(P_1^{(k)},P_2^{(k)})\\ &=(R'_{k-1})^{b_{k,j}}r_{k,j}\cdot\tilde{S}_{k,j}(F,G),\end{aligned}$$

which proves the theorem. □

Remark 4.2. Since $r_{k,j}=\pm1$, we see that $R'_k=\pm1$, thus the nested subresultant is equal to the recursive subresultant up to a sign.

4.4 REDUCED NESTED SUBRESULTANTS

The nested subresultant matrix has "nested" representation of subresultant matrices, which makes practical use difficult. However, in some cases, we can reduce the representation of the nested subresultant matrix to a "flat" representation, or a representation without nested determinants by the Gaussian elimination; this is the reduced nested subresultant (matrix). As we will see, the size of the reduced nested subresultant matrix becomes much smaller than that of the recursive subresultant matrix, with reasonable computing time.

First, we illustrate the idea of reduction of the nested subresultant matrix with an example.

Example 4.5. Let $F(x)$ and $G(x)$ be defined as

$$\begin{aligned} F(x) &= a_6x^6 + a_5x^5 + \cdots + a_0, \quad a_6 \neq 0, \\ G(x) &= b_5x^5 + b_4x^4 + \cdots + b_0, \quad b_5 \neq 0, \end{aligned}$$

with vectors of coefficients (a_6, a_5) and (b_5, b_4) are linearly independent as vectors over K. Assume that $\mathrm{prs}(F, G) = (P_1^{(1)} = F,\ P_2^{(1)} = G,\ P_3^{(1)} = \gcd(F, G))$ with $\deg(P_3^{(1)}) = 4$. Consider the $(2, 2)$-th nested subresultant; its matrix $\tilde{N}^{(2,2)}(F, G)$ is defined as

$$\tilde{N}^{(2,2)}(F,G) = \begin{pmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_2 & 2A_2 & 3A_3 \\ A_1 & A_1 & 2A_2 \\ A_0 & & A_1 \end{pmatrix}, \quad A_j = \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_j & b_{j-1} & b_j \end{vmatrix},$$

for $j \leqslant 4$ with $b_j = 0$ for $j < 0$. Now, let us calculate the leading coefficient of $\tilde{S}_{2,2}(F, G)$ as

$$\begin{aligned} |\tilde{N}_2^{(2,2)}| &= \begin{vmatrix} A_4 & 4A_4 & \\ A_3 & 3A_3 & 4A_4 \\ A_2 & 2A_2 & 3A_3 \end{vmatrix} \\ &= \begin{vmatrix} \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_4 & b_3 & b_4 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 4a_4 & 4b_3 & 4b_4 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 0a_4 & 0b_3 & 0b_4 \end{vmatrix} \\ \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_3 & b_2 & b_3 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 3a_3 & 3b_2 & 3b_3 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 4a_4 & 4b_3 & 4b_4 \end{vmatrix} \\ \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ a_2 & b_1 & b_2 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 2a_2 & 2b_1 & 2b_2 \end{vmatrix} & \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 3a_3 & 3b_2 & 3b_3 \end{vmatrix} \end{vmatrix} \\ &= |H| = \left|\left(H_{p,q}\right)\right|. \end{aligned} \tag{4.26}$$

Then, we make the $(3, 1)$ and the $(3, 2)$ elements in $H_{p,q}$ $(p, q = 1, 2, 3)$ equal to 0 by adding the first and the second rows, multiplied by certain numbers, to the third row. For example, in $H_{1,1}$, calculate x_{11} and y_{11} by solving a system of linear equations

$$\begin{cases} a_6x_{11} + a_5y_{11} = -a_4 \\ b_5x_{11} + b_4y_{11} = -b_3 \end{cases}, \tag{4.27}$$

(Note that (4.27) has a solution in K by the assumption), followed by adding the first row multiplied by x_{11} and the second row multiplied by y_{11}, respectively, to the third row. Then, we have

$$H_{1,1} = \begin{vmatrix} a_6 & b_5 & \\ a_5 & b_4 & b_5 \\ 0 & 0 & h_{11} \end{vmatrix} = \begin{vmatrix} a_6 & b_5 \\ a_5 & b_4 \end{vmatrix} h_{11}, \quad \text{with} \quad h_{11} = b_4 + y_{11} b_5. \tag{4.28}$$

Doing similar calculations for the other $H_{p,q}$, we calculate $h_{p,q}$ $(p,q = 1,2,3)$ for $H_{p,q}$ similarly as in (4.28). Finally, by putting such new representations of $H_{p,q}$ into (4.26), we have

$$|\tilde{N}_2^{(2,2)}| = \begin{vmatrix} a_6 & b_5 \\ a_5 & b_4 \end{vmatrix}^3 \begin{vmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{vmatrix} = \begin{vmatrix} a_6 & b_5 \\ a_5 & b_4 \end{vmatrix}^3 |\hat{N}_2^{(2,2)}|, \tag{4.29}$$

note that we have derived $\hat{N}_2^{(2,2)}$ as a reduced form of $\tilde{N}_2^{(2,2)}$.

As (4.29) shows, we derive a "reduced" form of the nested subresultant matrix by the Gaussian elimination for solving certain systems of linear equations. We define the reduced nested subresultant (matrix), as follows.

Definition 4.10 (Reduced Nested Subresultant Matrix)**.** Let F and G be defined as in (4.1), and let $(P_1^{(1)}, \dots, P_{l_1}^{(1)}, \dots, P_1^{(t)}, \dots, P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. Then, for each pair of numbers (k,j) with $k = 1,\dots,t$ and $j = j_{k-1} - 2, \dots, 0$, define matrix $\hat{N}^{(k,j)}(F,G)$ recursively as follows.

1. For $k = 1$, let $\hat{N}^{(1,j)}(F,G) = N^{(j)}(F,G)$.

2. For $k > 1$, let $\hat{N}_U^{(k-1,j_{k-1})}(F,G)$ be a sub-matrix of $\hat{N}^{(k-1,j_{k-1})}(F,G)$ by deleting the bottom $j_{k-1}+1$ rows, and $\hat{N}_L^{(k-1,j_{k-1})}(F,G)$ be a sub-matrix of $\hat{N}^{(k-1,j_{k-1})}(F,G)$ by taking the bottom $j_{k-1}+1$ rows, respectively. For $\tau = j_{k-1}, \dots, 0$ let $\hat{N}_\tau^{(k-1,j_{k-1})}(F,G)$ be a sub-matrix of $\hat{N}^{(k-1,j_{k-1})}(F,G)$ by putting $\hat{N}_U^{(k-1,j_{k-1})}(F,G)$ on the top and the $(j_{k-1} - \tau + 1)$-th row of $\hat{N}_L^{(k-1,j_{k-1})}(F,G)$ in the bottom row. Let $\hat{A}_\tau^{(k-1)} = |\hat{N}_\tau^{(k-1,j_{k-1})}|$ and construct a matrix $H^{(k,j)}$ as
$$H^{(k,j)} = \left(H_{p,q}^{(k,j)}\right) = N^{(j)}\left(\hat{A}^{(k-1)}(x), \frac{d}{dx}\hat{A}^{(k-1)}(x)\right), \tag{4.30}$$
where
$$\hat{A}^{(k-1)}(x) = \hat{A}_{j_{k-1}}^{(k-1)} x^{j_{k-1}} + \cdots + \hat{A}_0^{(k-1)} x^0.$$
Since $\hat{N}_\tau^{(k-1,j_{k-1})}$ consists of $\hat{N}_U^{(k-1,j_{k-1})}$ and a row vector in the bottom, we express $\hat{N}_U^{(k-1,j_{k-1})} = (U^{(k)} | \boldsymbol{v}^{(k)})$, where $U^{(k)}$ is a square matrix and $\boldsymbol{v}^{(k)}$ is a column vector, and the row vector in the bottom by $\left(\mathbf{b}_{p,q}^{(k,j)} \;\middle|\; g_{p,q}^{(k,j)}\right)$, where $\mathbf{b}_{p,q}^{(k,j)}$ is a row vector and $g_{p,q}^{(k,j)}$ is a number, respectively, such that
$$H_{p,q}^{(k,j)} = \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \mathbf{b}_{p,q}^{(k,j)} & g_{p,q}^{(k,j)} \end{array}\right|, \tag{4.31}$$

with $\mathbf{b}_{p,q}^{(k,j)} = 0$ and $g_{p,q}^{(k,j)} = 0$ for $H_{p,q}^{(k,j)} = 0$. Furthermore, we assume that $U^{(k)}$ is not singular. Then, for $p = 1, \dots, n_1^{(k)} + n_2^{(k)} - j$ and $q = 2, \dots, n_1^{(k)} + n_2^{(k)} - j$, calculate a row vector $\mathbf{x}_{p,q}^{(k,j)}$ by solving a system of linear equations

$$\mathbf{x}_{p,q}^{(k,j)} U^{(k)} = -\mathbf{b}_{p,q}^{(k,j)}, \tag{4.32}$$

and define $h_{p,q}^{(k,j)}$ as

$$h_{p,q}^{(k,j)} = \mathbf{x}_{p,q}^{(k,j)} \mathbf{v}^{(k,j)}.$$

Note that we have

$$H_{p,q}^{(k,j)} = \left| \begin{array}{c|c} U^{(k)} & \mathbf{v}^{(k)} \\ \hline 0 & h_{p,q}^{(k,j)} \end{array} \right| = \left| U^{(k)} \right| h_{p,q}^{(k,j)}.$$

Finally, define $\hat{N}^{(k,j)}(F,G)$ as

$$\hat{N}^{(k,j)}(F,G) = \begin{pmatrix} h_{1,1}^{(k,j)} & h_{1,2}^{(k,j)} & \cdots & h_{1,J_{k,j}}^{(k,j)} \\ h_{2,1}^{(k,j)} & h_{2,2}^{(k,j)} & \cdots & h_{2,J_{k,j}}^{(k,j)} \\ \vdots & \vdots & & \vdots \\ h_{I_{k,j},1}^{(k,j)} & h_{I_{k,j},2}^{(k,j)} & \cdots & h_{I_{k,j},J_{k,j}}^{(k,j)} \end{pmatrix}, \tag{4.33}$$

where

$$\begin{aligned} I_{k,j} &= n_1^{(k)} + n_2^{(k)} - j = (2j_{k-1} - 2j - 1) + j, \\ J_{k,j} &= n_1^{(k)} + n_2^{(k)} - 2j = 2j_{k-1} - 2j - 1. \end{aligned} \tag{4.34}$$

Then, $\hat{N}^{(k,j)}(F,G)$ is called the (k,j)-*th reduced nested subresultant matrix of* F *and* G.

Remark 4.3. Definition 4.10 shows that, For $k = 1, \dots, t$ and $j < j_{k-1} - 1$, the numbers of rows and columns of the (k,j)-th reduced nested subresultant matrix $\hat{N}^{(k,j)}(F,G)$ are $I_{k,j}$ and $J_{k,j}$ in (4.34), respectively, which are much smaller than those of the recursive subresultant matrix of the corresponding degree (see Proposition 4.2).

Definition 4.11 (Reduced Nested Subresultant)**.** Let F and G be defined as in (4.1), and let $(P_1^{(1)}, \dots, P_{l_1}^{(1)}, \dots, P_1^{(t)}, \dots, P_{l_t}^{(t)})$ be complete recursive PRS for F and G as in Definition 4.2. For $j = j_{k-1} - 2, \dots, 0$ and $\tau = j, \dots, 0$, let $\hat{N}_\tau^{(k,j)} = \hat{N}_\tau^{(k,j)}(F,G)$ be a sub-matrix of the (k,j)-th reduced nested subresultant matrix $\hat{N}^{(k,j)}(F,G)$ obtained by the top $n_1^{(k)} + n_2^{(k)} - 2j - 1$ rows and the $(n_1^{(k)} + n_2^{(k)} - j - \tau)$-th row (note that $\hat{N}_\tau^{(k,j)}(F,G)$ is a square matrix). Then, the polynomial

$$\hat{S}_{k,j}(F,G) = |\hat{N}_j^{(k,j)}(F,G)| x^j + \cdots + |\hat{N}_0^{(k,j)}(F,G)| x^0$$

is called the (k,j)-th reduced nested subresultant of F and G.

Now, we derive the relationship between the nested and the reduced nested subresultants.

Theorem 4.10. *Let* F *and* G *be defined as in* (4.1), *and let* $(P_1^{(1)}, \dots, P_{l_1}^{(1)}, \dots, P_1^{(t)}, \dots, P_{l_t}^{(t)})$ *be complete recursive PRS for* F *and* G *as in Definition* 4.2. *For* $k = 2, \dots, t$, $j = j_{k-1} - 2, \dots, 0$ *with* $J_{k,j}$ *as in* (4.33), *define* $\hat{B}_{k,j}$ *and* $\hat{R}_k$ *as*

$$\hat{B}_{k,j} = |U^{(k)}|^{J_{k,j}}$$

with $\hat{B}_k = \hat{B}_{k,j_k}$ *and* $\hat{B}_1 = \hat{B}_2 = 1$, *and*

$$\hat{R}_k = (\hat{R}_{k-1} \cdot \hat{B}_{k-1})^{J_{k,j_k}}$$

with $\hat{R}_1 = \hat{R}_2 = 1$, *respectively. Then, we have*

$$\tilde{S}_{k,j}(F,G) = (\hat{R}_{k-1} \cdot \hat{B}_{k-1})^{J_{k,j}} \hat{B}_{k,j} \cdot \hat{S}_{k,j}(F,G).$$

To prove Theorem 4.10, we prove the following lemma.

Lemma 4.11. *For* $k = 1, \dots, t$, $j = j_{k-1} - 2, \dots, 0$ *and* $\tau = j, \dots, 0$, *we have*

$$|\tilde{N}_\tau^{(k,j)}(F,G)| = (\hat{R}_{k-1} \cdot \hat{B}_{k-1})^{J_{k,j}} \hat{B}_{k,j} |\hat{N}_\tau^{(k,j)}(F,G)|. \tag{4.35}$$

Proof. By induction on k. For $k = 1$, it is obvious from the definitions of the nested and the reduced nested subresultants. Assume that (4.35) is valid for $1, \dots, k-1$. Then, for $\tau = j_{k-1}, \dots, 0$, we have

$$\begin{aligned} |\tilde{N}_\tau^{(k-1,j_{k-1})}(F,G)| &= (\hat{R}_{k-2} \cdot \hat{B}_{k-2})^{J_{k-1,j_{k-1}}} \hat{B}_{k-1,j_{k-1}} |\hat{N}_\tau^{(k-1,j_{k-1})}(F,G)| \\ &= (\hat{R}_{k-1} \cdot \hat{B}_{k-1}) |\hat{N}_\tau^{(k-1,j_{k-1})}(F,G)|. \end{aligned}$$

Let

$$\tilde{A}_\tau^{(k-1)} = |\tilde{N}_\tau^{(k-1,j_{k-1})}(F,G)|, \quad \hat{A}_\tau^{(k-1)} = |\hat{N}_\tau^{(k-1,j_{k-1})}(F,G)|,$$

and $H_\tau^{(k,j)} = \left(H_{\tau_{p,q}}^{(k,j)}\right)$ be a sub-matrix of $H^{(k,j)}$ in (4.30) by taking the top $J_{k,j}$ rows and the $(I_{k,j} - \tau)$-th row, where $I_{k,j}$ and $J_{k,j}$ are defined as in (4.34), respectively. Then, we have

$$|H_\tau^{(k,j)}|$$
$$= \left| \begin{array}{ccccc} \hat{A}_{j_{k-1}}^{(k-1)} & & j_{k-1}\hat{A}_{j_{k-1}}^{(k-1)} & & \\ \vdots & \ddots & \vdots & \ddots & \\ \vdots & \hat{A}_{j_{k-1}}^{(k-1)} & \vdots & & j_{k-1}\hat{A}_{j_{k-1}}^{(k-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \hat{A}_{2j-j_{k-1}+3}^{(k-1)} & \cdots \hat{A}_{j+1}^{(k-1)} & (2j - j_{k-1} + 3)\hat{A}_{2j-j_{k-1}+3}^{(k-1)} & \cdots & (j+2)\hat{A}_{j+2}^{(k-1)} \\ \hline \hat{A}_{j-j_{k-1}+\tau+2}^{(k-1)} & \cdots \hat{A}_\tau^{(k-1)} & (j - j_{k-1} + \tau + 2)\hat{A}_{j-j_{k-1}+\tau+2}^{(k-1)} & \cdots & (\tau+1)\hat{A}_{\tau+1}^{(k-1)} \end{array} \right|,$$

where $\hat{A}_l^{(k-1)} = 0$ for $l < 0$. On the other hand, by the definition of the (k,j)-th nested subresultant, we have

$$
\begin{aligned}
&|\tilde{N}_\tau^{(k,j)}(F,G)| \\
&= \left|\begin{array}{cccccc}
\tilde{A}_{j_{k-1}}^{(k-1)} & & & j_{k-1}\tilde{A}_{j_{k-1}}^{(k-1)} & & \\
\vdots & \ddots & & \vdots & \ddots & \\
\vdots & & \tilde{A}_{j_{k-1}}^{(k-1)} & \vdots & & j_{k-1}\tilde{A}_{j_{k-1}}^{(k-1)} \\
\vdots & & \vdots & \vdots & & \vdots \\
\tilde{A}_{2j-j_{k-1}+3}^{(k-1)} & \cdots & \tilde{A}_{j+1}^{(k-1)} & (2j-j_{k-1}+3)\tilde{A}_{2j-j_{k-1}+3}^{(k-1)} & \cdots & (j+2)\tilde{A}_{j+2}^{(k-1)} \\
\hline
\tilde{A}_{j-j_{k-1}+\tau+2}^{(k-1)} & \cdots & \tilde{A}_{\tau}^{(k-1)} & (j-j_{k-1}+\tau+2)\tilde{A}_{j-j_{k-1}+\tau+2}^{(k-1)} & \cdots & (\tau+1)\tilde{A}_{\tau+1}^{(k-1)}
\end{array}\right| \\
&= (\hat{R}_{k-1}\cdot\hat{B}_{k-1})^{J_{k,j}}|H_\tau^{(k,j)}|,
\end{aligned}
\tag{4.36}
$$

where $\tilde{A}_l^{(k-1)} = 0$ for $l < 0$. (Note that $\tilde{N}_\tau^{(k,j)}$ and $H_\tau^{(k,j)}$ are square matrices of order $J_{k,j}$.) By Definition 4.10, we can express $H_{\tau_{p,q}}^{(k,j)}$ as

$$
H_{\tau_{p,q}}^{(k,j)} = \left|\begin{array}{c|c} U^{(k)} & \boldsymbol{v}^{(k)} \\ \hline \mathbf{b}_{p,q}^{\prime(k,j)} & g_{p,q}^{\prime(k,j)} \end{array}\right|,
$$

with $\mathbf{b}_{p,q}^{\prime(k,j)} = \mathbf{0}$ and $g_{p,q}^{\prime(k,j)} = 0$ for $H_{\tau_{p,q}}^{(k,j)} = 0$. Note that, for $q = 1,\dots,J_{k,j}$, we have $\mathbf{b}_{p,q}^{\prime(k,j)} = \mathbf{b}_{p,q}^{(k,j)}$ and $g_{p,q}^{\prime(k,j)} = g_{p,q}^{(k,j)}$ for $p = 1,\dots,J_{k,j}-1$, and $\mathbf{b}_{J_{k,j},q}^{\prime(k,j)} = \mathbf{b}_{I_{k,j}-\tau,q}^{(k,j)}$ and $g_{J_{k,j},q}^{\prime(k,j)} = g_{I_{k,j}-\tau,q}^{(k,j)}$, where $\mathbf{b}_{p,q}^{(k,j)}$ and $g_{p,q}^{(k,j)}$ are defined as in (4.31), respectively. Thus, by (4.32)–(4.33), we have

$$
|H_\tau^{(k,j)}| = |U^{(k)}|^{J_{k,j}}|\hat{N}_\tau^{(k,j)}(F,G)| = \hat{B}_{k,j}|\hat{N}_\tau^{(k,j)}(F,G)|, \tag{4.37}
$$

and, by putting (4.37) into (4.36), we prove the lemma. □

Remark 4.4. We can calculate the (k,j)-th reduced nested subresultant matrix as a sub-matrix of the $(k,0)$-th reduced nested subresultant matrix. In (4.30), we see that the matrix $H^{(k,j)}$ is a sub-matrix of $N\left(\hat{A}^{(k-1)}(x), \frac{d}{dx}\hat{A}^{(k-1)}(x)\right)$, and, by the construction of the reduced nested resultant matrix (4.33), we see that $\hat{N}^{(k,j)}(F,G)$ is a sub-matrix of $\hat{N}^{(k,0)}(F,G)$ by taking the left $n_2^{(k)} - j$ columns from those corresponding to the coefficients of $\hat{A}^{(k-1)}(x)$ and the left $n_1^{(k)} - j$ columns from those corresponding to the coefficients of $\frac{d}{dx}\hat{A}^{(k-1)}(x)$, then taking the top $n_1^{(k)} + n_2^{(k)} - j$ rows.

Remark 4.5. We can estimate arithmetic computing time for the (k,j)-th reduced nested resultant matrix $\hat{N}^{(k,j)}$ in (4.33), as follows. The computing time for the elements $h_{p,q}$ is dominated by the time for the Gaussian elimination of $U^{(k)}$. Since the order of $U^{(k)}$ $(k = 2,\dots,t)$ is equal to $2(j_{k-2} - j_{k-1} - 1)$ (see Remark 4.3), it is bounded by $O((j_{k-2} - j_{k-1})^3)$ (see Golub and van Loan [19]). As Remark 4.4 shows, we can calculate $\hat{N}^{(k,j)}(F,G)$ for $j < j_{k-1} - 2$ by $\hat{N}^{(k,0)}(F,G)$. Therefore, total computing time for $\hat{N}^{(k,j)}$ for entire recursive PRS $(k = 1,\dots,t)$ is bounded by

$$
\begin{aligned}
\sum_{k=2}^{t} O\left((j_{k-2} - j_{k-1})^3\right) &= O\left(\sum_{k=2}^{t}(j_{k-2} - j_{k-1})^3\right) \\
&= O\left((j_0 - j_{t-1})^3\right) = O(m^3),
\end{aligned}
$$

note that $j_0 = m$ (see Remark 4.1). See also for remarks below.

4.5 SUMMARY

In this chapter, we have introduced concepts of *recursive PRS* and *recursive subresultants*, and investigated constructions of their subresultant matrices to compute the recursive subresultants. Among three different constructions of recursive subresultant matrices, we have shown that the reduced nested subresultant matrix reduces the size of the matrix drastically to at most the order of the degree of initial polynomials in each PRSs, compared with the naive recursive subresultant matrix. We have also shown that we can calculate the reduced nested subresultant matrix by the Gaussian elimination of order at most the sum of the degree of initial polynomials in each PRSs.

From a point of view of computational complexity, the algorithm for the reduced nested subresultant matrix has a cubic complexity bound in terms of the degree of the input polynomials (see Remark 4.5). However, subresultant algorithms which have a quadratic complexity bound in terms of the degree of the input polynomials have been proposed ([14], [30]); those algorithms exploit the structure of the Sylvester matrix to increase their efficiency with controlling the size of coefficients well. Although, in this chapter, we have primarily focused our attention into reducing the structure of the nested subresultant matrix to "flat" representation, development of more efficient algorithms such as exploiting the structure of the Sylvester matrix would be the next problem. Furthermore, the reduced nested subresultant may involve fractions which may be unusual for subresultants, thus more detailed analysis of computational efficiency including comparison with (ordinary and recursive) subresultants would also be necessary.

We expect that the reduced nested subresultants can be used for approximate algebraic computation such as the square-free decomposition of approximate univariate polynomials with approximate GCD computations based on Singular Value Decomposition (SVD) of subresultant matrices ([10], [17]), which motivates the present work. For the approximate square-free decomposition of the given polynomial $P(x)$, we have to calculate the approximate GCDs of $P(x), \dots, P^{(n)}(x)$ (by $P^{(n)}(x)$ we denote the n-th derivative of $P(x)$) or those of the recursive PRS for $P(x)$ and $P'(x)$; we have to find the representation of the subresultant matrices for $P(x), \dots, P^{(n)}(x)$, or that for the recursive PRS for $P(x)$ and $P'(x)$, respectively. As for the former approach, several algorithms based on different representations of subresultant matrices have been proposed ([13], [40]); our reduced nested subresultant matrix can be used as for the latter approach. To make use of the reduced nested subresultant matrix, we need to reveal the relationship between the structure of the subresultant matrices and their singular values; this is among the next problems in the future. Approximate GCD computation using (plain) subresultant matrices will be discussed in the next chapter.

5 GPGCD: AN ITERATIVE METHOD FOR CALCULATING APPROXIMATE GCD OF UNIVARIATE POLYNOMIALS

For algebraic computations on polynomials and matrices, approximate algebraic algorithms are attracting broad range of attentions recently. These algorithms take inputs with some "noise" such as polynomials with floating-point number coefficients with rounding errors, or more practical errors such as measurement errors, then, with minimal changes on the inputs, seek a meaningful answer that reflect desired property of the input, such as a common factor of a given degree. By this characteristic, approximate algebraic algorithms are expected to be applicable to more wide range of problems, especially those to which exact algebraic algorithms were not applicable.

As an approximate algebraic algorithm, we consider calculating the approximate greatest common divisor (GCD) of univariate polynomials with the real or the complex coefficients, such that, for a given pair of polynomials and a degree d, finding a pair of polynomials which has a GCD of degree d and whose coefficients are perturbations from those in the original inputs, with making the perturbations as small as possible, along with the GCD. This problem has been extensively studied with various approaches including the Euclidean method on the polynomial remainder sequence (PRS) ([4], [44], [47]), the singular value decomposition (SVD) of the Sylvester matrix ([10], [17]), the QR factorization of the Sylvester matrix or its displacements ([11], [65], [67]), Padé approximation [36], optimization strategies ([7], [24], [25], [26], [66]). Furthermore, stable methods for ill-conditioned problems have been discussed ([11], [35], [41]).

Among methods in the above, we focus our attention on optimization strategy in this paper, especially iterative method for approaching an optimal solution, after transferring the approximate GCD problem into a constrained minimization problem. Already proposed algorithms utilize iterative methods including the Levenberg-Marquardt method [7], the Gauss-Newton method [66] and the structured total least norm (STLN) method ([24], [25]). Among them, STLN-based methods have shown good performance calculating approximate GCD with sufficiently small perturbations efficiently.

Here, we utilize the so-called modified Newton method [53], which is a generalization of the gradient-projection method [39], for solving the constrained minimization problem. This method has interesting features such that it combines the *projection* and the *restoration* steps in the original gradient-projection method, which reduces the number of solving a linear system. We demonstrate that our algorithm calculates approximate GCD with perturbations as small as those calculated by the STLN-based methods, while our method show significantly better performance over them in its speed compared with their implementation, by approximately up to 30 times. Furthermore, we also show that our algorithm can properly handle some ill-conditioned problems such as those with GCD containing small or large leading coefficient.

The rest part of this chapter is organized as follows. In Section 5.1, we transform the approximate GCD problem into a constrained minimization problem. In

Section 5.2, we review the framework of the gradient-projection method and a modified Newton method. In Section 5.3, we show an algorithm for calculating the approximate GCD, with discussing issues in the application of the gradient-projection method or a modified Newton method. In Section 5.4, we demonstrate performance of our algorithm with experiments.

5.1 FORMULATION OF THE APPROXIMATE GCD PROBLEM

Let $F(x)$ and $G(x)$ be univariate polynomials with the real or the complex coefficients, given as

$$F(x) = f_m x^m + f_{m-1} x^{m-1} + \cdots + f_0,$$
$$G(x) = g_n x^n + g_{n-1} x^{n-1} + \cdots + g_0,$$

with $m \geqslant n > 0$. We permit F and G to be relatively prime in general. For a given integer d satisfying $n \geqslant d > 0$, let us calculate a deformation of $F(x)$ and $G(x)$ in the form of

$$\begin{aligned} \tilde{F}(x) &= F(x) + \Delta F(x) = H(x) \cdot \bar{F}(x), \\ \tilde{G}(x) &= G(x) + \Delta G(x) = H(x) \cdot \bar{G}(x), \end{aligned} \tag{5.1}$$

where $\Delta F(x)$, $\Delta G(x)$ are polynomials whose degrees do not exceed those of $F(x)$ and $G(x)$, respectively, $H(x)$ is a polynomial of degree d, and $\bar{F}(x)$ and $\bar{G}(x)$ are pairwise relatively prime. If we find $\tilde{F}$, $\tilde{G}$, $\bar{F}$, $\bar{G}$ and H satisfying (5.1), then we call H *an approximate GCD of* F *and* G. For a given degree d, we tackle the problem of finding an approximate GCD H with minimizing the norm of the deformations $\|\Delta F(x)\|_2^2 + \|\Delta G(x)\|_2^2$.

In the case $\tilde{F}(x)$ and $\tilde{G}(x)$ have a GCD of degree d, then the theory of subresultants tells us that the $(d-1)$-th subresultant of $\tilde{F}$ and $\tilde{G}$ becomes zero, namely we have

$$S_{d-1}(\tilde{F}, \tilde{G}) = 0,$$

where $S_k(\tilde{F}, \tilde{G})$ denotes the subresultant of $\tilde{F}$ and $\tilde{G}$ of degree k. Then, the $(d-1)$-th subresultant matrix

$$N_{d-1}(\tilde{F}, \tilde{G}) = \begin{pmatrix} \tilde{f}_m & & & \tilde{g}_n & & \\ \vdots & \ddots & & \vdots & \ddots & \\ \tilde{f}_0 & & \tilde{f}_m & \tilde{g}_0 & & \tilde{g}_n \\ & \ddots & \vdots & & \ddots & \vdots \\ & & \tilde{f}_0 & & & \tilde{g}_0 \end{pmatrix}, \tag{5.2}$$
$$\underbrace{\qquad\qquad}_{n-d+1} \quad \underbrace{\qquad\qquad}_{m-d+1}$$

where the k-th subresultant matrix $N_k(\tilde{F}, \tilde{G})$ is a submatrix of the Sylvester matrix $N(\tilde{F}, \tilde{G})$ by taking the left $n-k$ columns of coefficients of $\tilde{F}$ and the left $m-k$ columns of coefficients of $\tilde{G}$, has a kernel of dimension equal to 1. Thus, there exist polynomials $A(x), B(x) \in \mathbf{R}[x]$ or $\mathbf{C}[x]$ satisfying

$$A\tilde{F} + B\tilde{G} = 0, \tag{5.3}$$

with $\deg(A) < n - d$ and $\deg(B) < m - d$ and $A(x)$ and $B(x)$ are relatively prime. Therefore, for the given $F(x)$, $G(x)$ and d, our problem is to find $\Delta F(x)$, $\Delta G(x)$, $A(x)$ and $B(x)$ satisfying Eq. (5.3) with making $\|\Delta F\|_2^2 + \|\Delta G\|_2^2$ as small as possible.

5.1.1 The Real Coefficient Case

Assuming that we have $F(x)$ and $G(x)$ as polynomials with the real coefficients and find an approximate GCD with the real coefficients as well, we represent $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$ with the real coefficients as

$$\begin{aligned} \tilde{F}(x) &= \tilde{f}_m x^m + \cdots + \tilde{f}_0 x^0, \quad & \tilde{G}(x) &= \tilde{g}_n x^n + \cdots + \tilde{g}_0 x^0, \\ A(x) &= a_{n-d} x^{n-d} + \cdots + a_0 x^0, \quad & B(x) &= b_{m-d} x^{m-d} + \cdots + b_0 x^0, \end{aligned} \tag{5.4}$$

respectively, thus $\|\Delta F\|_2^2 + \|\Delta G\|_2^2$ and Eq. (5.3) become as

$$\begin{aligned} &\|\Delta F\|_2^2 + \|\Delta G\|_2^2 \\ &\qquad = (\tilde{f}_m - f_m)^2 + \cdots + (\tilde{f}_0 - f_0)^2 + (\tilde{g}_n - g_n)^2 + \cdots + (\tilde{g}_0 - g_0)^2, \end{aligned} \tag{5.5}$$

$$N_{d-1}(\tilde{F}, \tilde{G}) \cdot \boldsymbol{v} = \mathbf{0}, \tag{5.6}$$

respectively, with $N_{d-1}(\tilde{F}, \tilde{G})$ as in (5.2) and

$$\boldsymbol{v} = {}^t(a_{n-d}, \cdots, a_0, b_{m-d}, \cdots, b_0). \tag{5.7}$$

Then, Eq. (5.6) is regarded as a system of $m+n-d+1$ equations in $\tilde{f}_m, \ldots, \tilde{f}_0$, $\tilde{g}_n, \ldots, \tilde{g}_0$, $a_{n-d}, \ldots, a_0$, $b_{m-d}, \ldots, b_0$, as

$$q_1 = \tilde{f}_m a_{n-d} + \tilde{g}_n b_{m-d} = 0, \cdots, q_{m+n-d+1} = \tilde{f}_0 a_0 + \tilde{g}_0 b_0 = 0, \tag{5.8}$$

by putting q_j as the j-th row. Furthermore, for solving the problem below stably, we add another constraint enforcing the coefficients of $A(x)$ and $B(x)$ such that $\|A(x)\|_2^2 + \|B(x)\|_2^2 = 1$; thus we add

$$q_0 = a_{n-d}^2 + \cdots + a_0^2 + b_{m-d}^2 + \cdots + b_0^2 - 1 = 0 \tag{5.9}$$

into Eq. (5.8).

Now, we substitute the variables

$$(\tilde{f}_m, \ldots, \tilde{f}_0, \tilde{g}_n, \ldots, \tilde{g}_0, a_{n-d}, \ldots, a_0, b_{m-d}, \ldots, b_0) \tag{5.10}$$

as $\boldsymbol{x} = (x_1, \ldots, x_{2(m+n-d+2)})$, thus Eq. (5.5) and (5.8) with (5.9) become

$$\begin{aligned} f(\boldsymbol{x}) = (x_1 - f_m)^2 + \cdots + (x_{m+1} - f_0)^2 \\ + (x_{m+2} - g_n)^2 + \cdots + (x_{m+n+2} - g_0)^2, \end{aligned} \tag{5.11}$$

$$\boldsymbol{q}(\boldsymbol{x}) = {}^t(q_0(\boldsymbol{x}), q_1(\boldsymbol{x}), \ldots, q_{m+n-d+1}(\boldsymbol{x})) = \mathbf{0}, \tag{5.12}$$

respectively. Therefore, the problem of finding an approximate GCD can be formulated as a constrained minimization problem of finding a minimizer of the objective function $f(\boldsymbol{x})$ in (5.11), subject to $\boldsymbol{q}(\boldsymbol{x}) = \mathbf{0}$ in Eq. (5.12).

5.1.2 The Complex Coefficient Case

Now let us assume that we have $F(x)$ and $G(x)$ with the complex coefficients in general, represented as

$$\begin{aligned} F(x) &= (f_{m,1} + f_{m,2}\boldsymbol{i})x^m + \cdots + (f_{0,1} + f_{0,2}\boldsymbol{i}), \\ G(x) &= (g_{n,1} + g_{n,2}\boldsymbol{i})x^n + + \cdots + (g_{0,1} + g_{0,2}\boldsymbol{i}), \end{aligned}$$

where $f_{j,1}$, $g_{j,1}$, $f_{j,2}$, $g_{j,2}$ are real numbers; $f_{j,1}$, and $g_{j,1}$ represent the real parts; $f_{j,2}$, $g_{j,2}$ represent the imaginary parts, with $\boldsymbol{i}$ as the imaginary unit, and find an approximate GCD with the complex coefficients. Then, we represent $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$ with the complex coefficients as

$$\begin{aligned}
\tilde{F}(x) &= (\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i})x^m + \cdots + (\tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i})x^0,\\
\tilde{G}(x) &= (\tilde{g}_{n,1} + \tilde{g}_{n,2}\boldsymbol{i})x^n + \cdots + (\tilde{g}_0 x^0 + \tilde{g}_{0,2}\boldsymbol{i})x^0,\\
A(x) &= (a_{n-d,1} + a_{n-d,2}\boldsymbol{i})x^{n-d} + \cdots + (a_{0,1} + a_{0,2}\boldsymbol{i})x^0,\\
B(x) &= (b_{m-d,1} + b_{m-d,2}\boldsymbol{i})x^{m-d} + \cdots + (b_{0,1} + b_{0,2}\boldsymbol{i})x^0,
\end{aligned} \tag{5.13}$$

respectively, where $\tilde{f}_{j,1}$, $\tilde{f}_{j,2}$, $\tilde{g}_{j,1}$, $\tilde{g}_{j,2}$, $a_{j,1}$, $a_{j,2}$, $b_{j,1}$, $b_{j,2}$ are real numbers.

For the objective function, $\|\Delta F\|_2^2 + \|\Delta G\|_2^2$ becomes as

$$\sum_{j=0}^{m}[(\tilde{f}_{j,1} - f_{j,1})^2 + (\tilde{f}_{j,2} - f_{j,2})^2] + \sum_{j=0}^{n}[(\tilde{g}_{j,1} - g_{j,1})^2 + (\tilde{g}_{j,2} - g_{j,2})^2]. \tag{5.14}$$

For the constraint, Eq. (5.3) becomes as

$$\begin{pmatrix}
\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i} & & & \tilde{g}_{n,1} + \tilde{g}_{n,2}\boldsymbol{i} & & \\
\vdots & \ddots & & \vdots & \ddots & \\
\tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i} & & \tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i} & \tilde{g}_{0,1} + \tilde{g}_{0,2}\boldsymbol{i} & & \tilde{g}_{n,1} + \tilde{g}_{n,2}\boldsymbol{i} \\
& \ddots & \vdots & & \ddots & \vdots \\
& & \tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i} & & & \tilde{g}_{0,1} + \tilde{g}_{0,2}\boldsymbol{i}
\end{pmatrix}
\times
\begin{pmatrix}
a_{n-d,1} + a_{n-d,2}\boldsymbol{i} \\ \vdots \\ a_{0,1} + a_{0,2}\boldsymbol{i} \\ b_{m-d,1} + b_{m-d,2}\boldsymbol{i} \\ \vdots \\ b_{0,1} + b_{0,2}\boldsymbol{i}
\end{pmatrix} = 0. \tag{5.15}$$

By expressing the subresultant matrix and the column vector in (5.15) separated into the real and the complex parts, respectively, we express (5.15) as

$$(N_1 + N_2\boldsymbol{i})(\boldsymbol{v}_1 + \boldsymbol{v}_2\boldsymbol{i}) = 0, \tag{5.16}$$

with

$$N_1 = \begin{pmatrix}
\tilde{f}_{m,1} & & & \tilde{g}_{n,1} & & \\
\vdots & \ddots & & \vdots & \ddots & \\
\tilde{f}_{0,1} & & \tilde{f}_{m,1} & \tilde{g}_{0,1} & & \tilde{g}_{n,1} \\
& \ddots & \vdots & & \ddots & \vdots \\
& & \tilde{f}_{0,1} & & & \tilde{g}_{0,1}
\end{pmatrix}, \quad
N_2 = \begin{pmatrix}
\tilde{f}_{m,2} & & & \tilde{g}_{n,2} & & \\
\vdots & \ddots & & \vdots & \ddots & \\
\tilde{f}_{0,2} & & \tilde{f}_{m,2} & \tilde{g}_{0,2} & & \tilde{g}_{n,2} \\
& \ddots & \vdots & & \ddots & \vdots \\
& & \tilde{f}_{0,2} & & & \tilde{g}_{0,2}
\end{pmatrix},
\tag{5.17}$$
$$\begin{aligned}
\boldsymbol{v}_1 &= {}^t(a_{n-d,1}, \dots, a_{0,1}, b_{m-d,1}, \dots, b_{0,1}),\\
\boldsymbol{v}_2 &= {}^t(a_{n-d,2}, \dots, a_{0,2}, b_{m-d,2}, \dots, b_{0,2}).
\end{aligned}$$

We can expand the left-hand-side of Eq. (5.16) as

$$(N_1 + N_2\boldsymbol{i})(\boldsymbol{v}_1 + \boldsymbol{v}_2\boldsymbol{i}) = (N_1\boldsymbol{v}_1 - N_2\boldsymbol{v}_2) + \boldsymbol{i}(N_1\boldsymbol{v}_2 + N_2\boldsymbol{v}_1),$$

thus, Eq. (5.16) is equivalent to a system of equations

$$N_1\boldsymbol{v}_1 - N_2\boldsymbol{v}_2 = 0, \quad N_1\boldsymbol{v}_2 + N_2\boldsymbol{v}_1 = 0,$$

which is expressed as

$$\begin{pmatrix} N_1 & -N_2 \\ N_2 & N_1 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} = 0. \tag{5.18}$$

Furthermore, as well as in the real coefficients case, we add another constraint for the coefficient of $A(x)$ and $B(x)$ as

$$\begin{aligned} \|A(x)\|_2^2 + \|B(x)\|_2^2 = (a_{n-d,1}^2 + \cdots + a_{0,1}^2) + (b_{m-d,1}^2 + \cdots + b_{0,1}^2) \\ + (a_{n-d,2}^2 + \cdots + a_{0,2}^2) + (b_{m-d,2}^2 + \cdots + b_{0,2}^2) - 1 = 0, \end{aligned} \tag{5.19}$$

which can be expressed together with (5.18) as

$$\begin{pmatrix} {}^t\boldsymbol{v}_1 & {}^t\boldsymbol{v}_2 & -1 \\ N_1 & -N_2 & 0 \\ N_2 & N_1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ 1 \end{pmatrix} = 0, \tag{5.20}$$

where Eq. (5.19) has been put on the top of Eq. (5.18). Note that, in Eq. (5.20), we have total of $2(m+n-d+1)+1$ equations in the coefficients of polynomials in (5.13) as a constraint, with the j-th row of which is expressed as $q_j = 0$, as similarly as in the real case (5.8) with (5.9).

Now, as in the real case, we substitute the variables

$$\begin{aligned} (\tilde{f}_{m,1}, \ldots, \tilde{f}_{0,1}, \tilde{g}_{n,1}, \ldots, \tilde{g}_{0,1}, \tilde{f}_{m,2}, \ldots, \tilde{f}_{0,2}, \tilde{g}_{n,2}, \ldots, \tilde{g}_{0,2}, \\ a_{n-d,1}, \ldots, a_{0,1}, b_{m-d,1}, \ldots, b_{0,1}, a_{n-d,2}, \ldots, a_{0,2}, b_{m-d,2}, \ldots, b_{0,2}) \end{aligned} \tag{5.21}$$

as $\boldsymbol{x} = (x_1, \ldots, x_{4(m+n-d+2)})$, thus Eq. (5.14) and (5.20) become as

$$\begin{aligned} f(\boldsymbol{x}) = {} & (x_1 - f_{m,1})^2 + \cdots + (x_{m+1} - f_{0,1})^2 \\ & + (x_{m+2} - g_{n,1})^2 + \cdots + (x_{m+n+2} - g_{0,1})^2 \\ & + (x_{m+n+3} - f_{m,2})^2 + \cdots + (x_{2m+n+3} - f_{0,2})^2 \\ & + (x_{2m+n+4} - g_{n,2})^2 + \cdots + (x_{2(m+n+2)} - g_{0,2})^2, \end{aligned} \tag{5.22}$$

$$\boldsymbol{q}(\boldsymbol{x}) = {}^t(q_1(\boldsymbol{x}), \ldots, q_{2(m+n-d+1)+1}(\boldsymbol{x})) = 0, \tag{5.23}$$

respectively. Therefore, the problem of finding an approximate GCD can be formulated as a constrained minimization problem of finding a minimizer of the objective function $f(\boldsymbol{x})$ in Eq. (5.22), subject to $\boldsymbol{q}(\boldsymbol{x}) = 0$ in Eq. (5.23).

5.2 THE GRADIENT-PROJECTION METHOD AND A MODIFIED NEWTON METHOD

In this section, we consider the problem of minimizing an objective function $f(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}$, subject to the constraints $\boldsymbol{q}(\boldsymbol{x}) = 0$ for $\boldsymbol{q}(\boldsymbol{x}) = {}^t(q_1(\boldsymbol{x}), q_2(\boldsymbol{x}), \ldots, q_m(\boldsymbol{x}))$, with $m \leqslant n$, where $q_j(\boldsymbol{x})$ is a function of $\mathbf{R}^n \to \mathbf{R}$, and $f(\boldsymbol{x})$ and $q_j(\boldsymbol{x})$ are twice continuously differentiable (here, we refer presentations of the problem to Tanabe [53] and the references therein).

If we assume that the Jacobian matrix

$$J_{\mathbf{q}}(\mathbf{x}) = \left(\frac{\partial q_i}{\partial x_j}\right)$$

is of full rank, or

$$\operatorname{rank}(J_{\mathbf{q}}(\mathbf{x})) = m, \tag{5.24}$$

on the feasible region $V_{\mathbf{q}}$ defined by

$$V_{\mathbf{q}} = \{\mathbf{x} \in \mathbf{R}^n \mid \mathbf{q}(\mathbf{x}) = 0\},$$

then the feasible region $V_{\mathbf{q}}$ is an $(n-m)$-dimensional differential manifold in $\mathbf{R}^n$ and f is differentiable function on the manifold $V_{\mathbf{q}}$. Thus, our problem is to find a point in $V_{\mathbf{q}}$, which will be a candidate of a local minimizer, satisfying the well-known "first-order necessary conditions" (for the proof, refer to the literature on optimization [34]).

Theorem 5.1 (First-order necessary conditions)**.** *Suppose that $\mathbf{x}^* \in V_{\mathbf{q}}$ is a local solution of the problem in the above, that the functions $f(\mathbf{x})$ and $\mathbf{q}(\mathbf{x})$ are continuously differentiable at $\mathbf{x}^*$, and that we have (5.24) at $\mathbf{x}^*$. Then, there exist a* Lagrange multiplier vector *$\boldsymbol{\lambda}^* \in \mathbf{R}^m$ satisfying*

$$\nabla f(\mathbf{x}^*) - {}^t(J_{\mathbf{q}}(\mathbf{x}^*))\boldsymbol{\lambda}^* = 0, \quad \mathbf{q}(\mathbf{x}^*) = 0. \quad \square$$

5.2.1 The Gradient-Projection Method

Let $\mathbf{x}_k \in \mathbf{R}^n$ be a feasible point, or a point satisfying $\mathbf{x}_k \in V_{\mathbf{q}}$. Rosen's gradient projection method [39] is based on projecting the steepest descent direction onto the tangent space of the manifold $V_{\mathbf{q}}$ at $\mathbf{x}_k$, which is denoted to $T_{\mathbf{x}_k}$ and represented by the kernel of the Jacobian matrix $J_{\mathbf{q}}(\mathbf{x}_k)$ as

$$T_{\mathbf{x}_k} = \ker(J_{\mathbf{q}}(\mathbf{x}_k)) = \{\mathbf{z} \in \mathbf{R}^n \mid J_{\mathbf{q}}(\mathbf{x}_k)\mathbf{z} = 0 \in \mathbf{R}^m\}. \tag{5.25}$$

We have steepest descent direction of the objective function f at $\mathbf{x}_k$ as

$$-\nabla f(\mathbf{x}_k) = -{}^t\left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right). \tag{5.26}$$

Then, the search direction $\mathbf{d}_k$ is defined by the projection of the steepest descent direction of f in (5.26) onto $T_{\mathbf{x}_k}$ in (5.25) as

$$\mathbf{d}_k = -P(\mathbf{x}_k)\nabla f(\mathbf{x}_k). \tag{5.27}$$

Here, $P(\mathbf{x}_k)$ is the orthogonal projection operator on $T_{\mathbf{x}_k}$ defined as

$$P(\mathbf{x}_k) = I - (J_{\mathbf{q}}(\mathbf{x}_k))^+(J_{\mathbf{q}}(\mathbf{x}_k)),$$

where I is the identity matrix and $(J_{\mathbf{q}}(\mathbf{x}_k))^+$ is the Moore-Penrose inverse of $(J_{\mathbf{q}}(\mathbf{x}_k))$. Under the assumption (5.24), we have

$$(J_{\mathbf{q}}(\mathbf{x}_k))^+ = {}^t(J_{\mathbf{q}}(\mathbf{x}_k)) \cdot ({}^t(J_{\mathbf{q}}(\mathbf{x}_k)))^{-1}.$$

With an appropriate step width α_k (in this paper, we omit how to calculate α_k in detail) satisfying $0 < \alpha_k \leqslant 1$, let

$$\mathbf{y}_k = \mathbf{x}_k + \alpha_k \cdot \mathbf{d}_k.$$

Since $V_{\mathbf{q}}$ is nonlinear in general, $\mathbf{y}_k$ may not in $V_{\mathbf{q}}$: in such a case, we take a *restoration* move to bring $\mathbf{y}_k$ back to $V_{\mathbf{q}}$, as follows. Let $\mathbf{x} \in \mathbf{R}^n$ be an arbitrary point. Then, at $\mathbf{y}_k$, the constraint $\mathbf{q}(\mathbf{x})$ can be linearly approximated as

$$\mathbf{q}(\mathbf{y}_k + \mathbf{x}) \simeq \mathbf{q}(\mathbf{y}_k) + J_{\mathbf{q}}(\mathbf{y}_k)\mathbf{x}.$$

Assuming $\mathbf{y}_k + \mathbf{x} \in V_{\mathbf{q}}$, we have $\mathbf{q}(\mathbf{y}_k + \mathbf{x}) = 0$ thus the approximation of $\mathbf{x}$ can be calculated as

$$\mathbf{x} = -(J_{\mathbf{q}}(\mathbf{y}_k))^{+}\mathbf{q}(\mathbf{y}_k). \tag{5.28}$$

If $\mathbf{y}_k$ is sufficiently close to $V_{\mathbf{q}}$, then we can restore $\mathbf{y}_k$ back onto $V_{\mathbf{q}}$ by applying (5.28) iteratively for several times. Note that the restoration move can also be used in the case the initial point of the minimization process is away from the feasible region $V_{\mathbf{q}}$.

Summarizing the above, we obtain an algorithm for the gradient projection as follows.

Algorithm 1 (The gradient-projection method [39]).

STEP 1 **[Restoration]** If the given point $\mathbf{x}_0$ does not satisfy $\mathbf{x}_0 \in V_{\mathbf{q}}$, first move $\mathbf{x}_0$ onto $V_{\mathbf{q}}$ by the iteration of Eq. (5.28), then let $\mathbf{x}_0$ be the restored point on $V_{\mathbf{q}}$. Let $k = 0$.

STEP 2 **[Projection]** For $\mathbf{x}_k$, calculate $\mathbf{d}_k = -P(\mathbf{x}_k)\nabla f(\mathbf{x}_k)$ by (5.27). If $\|\mathbf{d}_k\|$ is sufficiently small for an appropriate norm, go to Step 4. Otherwise, calculate the step width α_k by an appropriate line search method (we omit its detail here) then let $\mathbf{y}_{k,0} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$.

STEP 3 **[Restoration]** If $\mathbf{q}(\mathbf{y}_{k,0}) \neq 0$, move $\mathbf{y}_{k,0}$ back onto $V_{\mathbf{q}}$ iteratively by (5.28). Let $\mathbf{y}_{k,l+1} = -(J_{\mathbf{q}}(\mathbf{y}_{k,l}))^{+}\mathbf{q}(\mathbf{y}_{k,l})$ for $l = 0, 1, 2, \ldots$. When $\mathbf{y}_{k,l}$ satisfies $\mathbf{q}(\mathbf{y}_{k,l})$, then let $\mathbf{x}_{k+1} = \mathbf{y}_{k,l}$ and go to Step 2.

STEP 4 **[Checking the first-order necessary conditions]** If $\mathbf{x}_k$ satisfies Theorem 5.1, then return $\mathbf{x}_k$.

5.2.2 The Modified Newton Method

The modified Newton method by Tanabe [53] is a generalization of the Newton's method, which derives several different methods, by modifying the Hessian of the Lagrange function. A generalization of the gradient-projection method combines the *restoration step* and the *projection step* in Algorithm 1. For $\mathbf{x}_k \in V_{\mathbf{q}}$, we calculate the search direction $\mathbf{d}_k$, along with the associated Lagrange multipliers $\boldsymbol{\lambda}_{k+1}$, by solving a linear system

$$\begin{pmatrix} I & -{}^t(J_{\mathbf{q}}(\mathbf{x}_k)) \\ J_{\mathbf{q}}(\mathbf{x}_k) & \mathbf{O} \end{pmatrix} \begin{pmatrix} \mathbf{d}_k \\ \boldsymbol{\lambda}_{k+1} \end{pmatrix} = - \begin{pmatrix} \nabla f(\mathbf{x}_k) \\ \mathbf{q}(\mathbf{x}_k) \end{pmatrix}, \tag{5.29}$$

then put $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \cdot \mathbf{d}_k$ with an appropriate step width α_k. Solving Eq. (5.29) under assumption (5.24), we have

$$\begin{aligned} \mathbf{d}_k &= -P(\mathbf{x}_k)\nabla f(\mathbf{x}_k) - (J_{\mathbf{q}}(\mathbf{x}_k))^{+}\mathbf{q}(\mathbf{x}_k), \\ \boldsymbol{\lambda}_{k+1} &= {}^t((J_{\mathbf{q}}(\mathbf{x}_k))^{+})\nabla f(\mathbf{x}_k) - (J_{\mathbf{q}}(\mathbf{x}_k) \cdot {}^t(J_{\mathbf{q}}(\mathbf{x}_k)))^{-1}\mathbf{q}(\mathbf{x}_k). \end{aligned} \tag{5.30}$$

Note that, in $\mathbf{d}_k$ in (5.30), the term $-P(\mathbf{x}_k)\nabla f(\mathbf{x}_k)$ comes from the projection (5.27), while another term $-(J_{\mathbf{q}}(\mathbf{x}_k))^{+}\mathbf{q}(\mathbf{x}_k)$ comes from the restoration (5.28). If we have

$\boldsymbol{x}_k \in V_{\boldsymbol{q}}$, the iteration formula (5.29) is equivalent to the projection (5.27). After an iteration, the new estimate $\boldsymbol{x}_{k+1}$ may not satisfy $\boldsymbol{x}_{k+1} \in V_{\boldsymbol{q}}$: in such a case, in the next iteration, the point will be pulled back onto $V_{\boldsymbol{q}}$ by the $-(J_{\boldsymbol{q}}(\boldsymbol{x}_k))^{+}\boldsymbol{q}(\boldsymbol{x}_k)$ term. Therefore, by solving Eq. (5.29) iteratively, we expect that the approximations $\boldsymbol{x}_k$ moves toward descending direction of f along with tracing the feasible set $V_{\boldsymbol{q}}$.

Summarizing the above, we obtain an algorithm as follows.

Algorithm 2 (The modified Newton method [53]).

STEP 1 **[Finding a search direction]** For $\boldsymbol{x}_k$, calculate $\boldsymbol{d}_k$ by solving the linear system (5.29). If $\|\boldsymbol{d}_k\|$ is sufficiently small, go to Step 2. Otherwise, calculate the step width α_k by an appropriate line search method (we omit its detail here), let $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, then go to Step 1.

STEP 2 **[Checking the first-order necessary conditions]** If $\boldsymbol{x}_k$ satisfies Theorem 5.1, then return $\boldsymbol{x}_k$.

5.3 THE ALGORITHM FOR APPROXIMATE GCD

In applying the gradient-projection method or a modified Newton method to the approximate GCD problem, we discuss issues in the construction of the algorithm in detail, such as

- Representation of the Jacobian matrix $J_{\boldsymbol{q}}(\boldsymbol{x})$ (Section 5.3.1),
- Certifying that $J_{\boldsymbol{q}}(\boldsymbol{x})$ has full rank (Section 5.3.2),
- Setting the initial values (Section 5.3.3),
- Regarding the minimization problem as the minimum distance problem (Section 5.3.4),
- Calculating the actual GCD and correcting the coefficients of $\tilde{F}$ and $\tilde{G}$ (Section 5.3.5),

as follows. After presenting the algorithm, we give a modification for preserving monicity for the real coefficient case, and end this section with examples.

5.3.1 Representation of the Jacobian Matrix

For a polynomial $P(x) \in \mathbf{R}[x]$ or $\mathbf{C}[x]$ represented as

$$P(x) = p_n x^n + \cdots + p_0 x^0,$$

let $C_k(P)$ be a complex $(n+k, k+1)$ matrix defined as

$$C_k(P) = \underbrace{\begin{pmatrix} p_n & & \\ \vdots & \ddots & \\ p_0 & & p_n \\ & \ddots & \vdots \\ & & p_0 \end{pmatrix}}_{k+1}.$$

We show the Jacobian matrix in the real and the complex coefficient cases, both of which can easily be constructed in every iteration in Algorithms 1 and 2.

The Real Coefficient Case

For co-factors $A(x)$ and $B(x)$ as in (5.4), consider matrices $C_m(A)$ and $C_n(B)$. Then, by the definition of the constraint (5.12), we have the Jacobian matrix $J_q(\boldsymbol{x})$ (with the original notation of variables for $\boldsymbol{x}$ as in (5.10)) as

$$J_q(\boldsymbol{x}) = \begin{pmatrix} 0 & 0 & 2\cdot {}^t\boldsymbol{v} \\ C_m(A) & C_n(B) & N_{d-1}(\tilde{F},\tilde{G}) \end{pmatrix}, \tag{5.31}$$

with $N_{d-1}(\tilde{F},\tilde{G})$ as in (5.2) and $\boldsymbol{v}$ as in (5.7), respectively.

The Complex Coefficient Case

For co-factors $A(x)$ and $B(x)$ as in (5.13), consider matrices $C_m(A)$ and $C_n(B)$ and express them as the sum of matrices consisting of the real and the imaginary parts of whose elements, respectively, as

$$\begin{aligned} C_m(A) &= \begin{pmatrix} a_{n-d,1} & & \\ \vdots & \ddots & \\ a_{0,1} & & a_{n-d,1} \\ & \ddots & \vdots \\ & & a_{0,1} \end{pmatrix} + \boldsymbol{i} \begin{pmatrix} a_{n-d,2} & & \\ \vdots & \ddots & \\ a_{0,2} & & a_{n-d,2} \\ & \ddots & \vdots \\ & & a_{0,2} \end{pmatrix} \\ &= C_m(A)_1 + \boldsymbol{i}C_m(A)_2, \\ C_n(B) &= \begin{pmatrix} b_{m-d,1} & & \\ \vdots & \ddots & \\ b_{0,1} & & b_{m-d,1} \\ & \ddots & \vdots \\ & & b_{0,1} \end{pmatrix} + \boldsymbol{i} \begin{pmatrix} b_{m-d,2} & & \\ \vdots & \ddots & \\ b_{0,2} & & b_{m-d,2} \\ & \ddots & \vdots \\ & & b_{0,2} \end{pmatrix} \\ &= C_n(B)_1 + \boldsymbol{i}C_n(B)_2, \end{aligned}$$

respectively, and define

$$\begin{aligned} A_1 = [C_m(A)_1\ C_n(B)_1] &= \begin{pmatrix} a_{n-d,1} & & b_{m-d,1} & \\ \vdots & \ddots & \vdots & \ddots \\ a_{0,1} & a_{n-d,1} & b_{0,1} & b_{m-d,1} \\ & \ddots\ \vdots & & \ddots\ \vdots \\ & a_{0,1} & & b_{0,1} \end{pmatrix}, \\ A_2 = [C_m(A)_2\ C_n(B)_2] &= \begin{pmatrix} a_{n-d,2} & & b_{m-d,2} & \\ \vdots & \ddots & \vdots & \ddots \\ a_{0,2} & a_{n-d,2} & b_{0,2} & b_{m-d,2} \\ & \ddots\ \vdots & & \ddots\ \vdots \\ & a_{0,2} & & b_{0,2} \end{pmatrix}. \end{aligned} \tag{5.32}$$

(Note that A_1 and A_2 are matrices of the real numbers of $m+n-d+1$ rows and $m+n+2$ columns.) Then, by the definition of the constraint (5.23), we have the Jacobian matrix $J_q(\boldsymbol{x})$ (with the original notation of variables for $\boldsymbol{x}$ as in (5.21)) as

$$J_q(\boldsymbol{x}) = \begin{pmatrix} 0 & 0 & 2\cdot {}^t\boldsymbol{v}_1 & 2\cdot {}^t\boldsymbol{v}_2 \\ A_1 & -A_2 & N_1 & -N_2 \\ A_2 & A_1 & N_2 & N_1 \end{pmatrix}, \tag{5.33}$$

with A_1 and A_2 as in (5.32) and N_1, N_2, $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ as in (5.17), respectively.

5.3.2 Certifying the Rank of the Jacobian Matrix

In executing Algorithm 1 or 2, we need to keep that $J_{\boldsymbol{q}}(\boldsymbol{x})$ has full rank: otherwise, we cannot correctly calculate $(J_{\boldsymbol{q}}(\boldsymbol{x}))^+$ (in Algorithm 1) or the matrix in (5.29) becomes singular (in Algorithm 2) thus we are unable to decide proper search direction. For this requirement, we have the following observations.

Proposition 5.2. *Let $\boldsymbol{x}^* \in V_{\boldsymbol{q}}$ be any feasible point satisfying Eq. (5.12). Then, if the corresponding polynomials do not have a GCD whose degree exceeds d, then $J_{\boldsymbol{q}}(\boldsymbol{x}^*)$ has full rank.*

Proof. We prove the proposition in the real and the complex coefficient cases separately.

The Real Coefficient Case

Let $\boldsymbol{x}^* = (\tilde{f}_m, \dots, \tilde{f}_0, \tilde{g}_n, \dots, \tilde{g}_0, a_{n-d} \dots, a_0, b_{m-d}, \dots, b_0)$ with its polynomial representation expressed as in (5.4) (note that this assumption permits the polynomials $\tilde{F}(x)$ and $\tilde{G}(x)$ to be relatively prime in general). To verify our claim, we show that we have $\operatorname{rank}(J_{\boldsymbol{q}}(\boldsymbol{x}^*)) = m+n-d+2$ as in (5.24), with $J_{\boldsymbol{q}}(\boldsymbol{x}^*)$ as in (5.31). Let us express $J_{\boldsymbol{q}}(\boldsymbol{x}^*) = (J_L \mid J_R)$, where J_L and J_R are column blocks expressed as

$$J_L = \begin{pmatrix} 0 & 0 \\ C_m(A) & C_n(B) \end{pmatrix}, \quad J_R = \begin{pmatrix} 2 \cdot \boldsymbol{v} \\ N_{d-1}(\tilde{F}, \tilde{G}) \end{pmatrix},$$

respectively. Then, we have the following lemma.

Lemma 5.3. *We have* $\operatorname{rank}(J_L) = m+n-d+1$.

Proof. Let us express $J_L = (J_{LL} \mid J_{LR})$, where

$$J_{LL} = \begin{pmatrix} 0 \\ C_m(A) \end{pmatrix}, \quad J_{LR} = \begin{pmatrix} 0 \\ C_n(B) \end{pmatrix},$$

and let $\bar{J}_L$ be a submatrix of J_L by taking the right $m-d$ columns of J_{LL} and the right $n-d$ columns of J_{LR}. Then, we see that the bottom $m+n-2d$ rows of $\bar{J}_L$ is equal to $N(A, B)$, the Sylvester matrix of $A(x)$ and $B(x)$. By the assumption, polynomials $A(x)$ and $B(x)$ are relatively prime, and there exist no nonzero elements in $\bar{J}_L$ except for the bottom $m+n-2d$ rows, we have $\operatorname{rank}(\bar{J}_L) = m+n-2d$.

By the above structure of $\bar{J}_L$ and the lower triangular structure of J_{LL} and J_{LR}, we can take the left $d+1$ columns of J_{LL} or J_{LR} satisfying linear independence along with the $m+n-2d$ columns in $\bar{J}_L$. Therefore, these $m+n-d+1$ columns generate a $(m+n-d+1)$-dimensional subspace in $\mathbf{R}^{m+n-d+2}$ satisfying

$$\{{}^t(x_1, \dots, x_{m+n-d+2}) \in \mathbf{R}^{m+n-d+2} \mid x_1 = 0\}, \tag{5.34}$$

and we see that none of the columns in J_L have nonzero element in the top coordinate. This proves the lemma. □

Proof of Proposition 5.2 (in the real coefficient case, continued). By the assumptions, we have at least one column vector in J_R with nonzero coordinate on the top row. By adding such a column vector to the basis of the subspace (5.34) that are generated as in Lemma 5.3, we have a basis of $\mathbf{R}^{m+n-d+2}$. This implies $\operatorname{rank}(J_{\boldsymbol{q}}(\boldsymbol{x})) = m+n-d+2$, which proves the proposition in the real coefficient case.

The Complex Coefficient Case

Let $\mathbf{x}^* = (\tilde{f}_{m,1}, \ldots, \tilde{f}_{0,1}, \tilde{g}_{n,1}, \ldots, \tilde{g}_{0,1}, \tilde{f}_{m,2}, \ldots, \tilde{f}_{0,2}, \tilde{g}_{n,2}, \ldots, \tilde{g}_{0,2}, a_{n-d,1}, \ldots, a_{0,1}, b_{m-d,1}, \ldots, b_{0,1}, a_{n-d,2}, \ldots, a_{0,2}, b_{m-d,2}, \ldots, b_{0,2})$ with its polynomial representation expressed as in (5.13) (note that this assumption permits the polynomials $\tilde{F}(x)$ and $\tilde{G}(x)$ to be relatively prime in general). To verify our claim, we show that we have $\mathrm{rank}(J_{\mathbf{q}}(\mathbf{x}^*)) = 2(m+n-d+1)+1$ as in (5.24), with $J_{\mathbf{q}}(\mathbf{x}^*)$ as in (5.33). Let us express $J_{\mathbf{q}}(\mathbf{x}^*) = (J_L \mid J_R)$, where J_L and J_R are column blocks expressed as

$$J_L = \begin{pmatrix} 0 & 0 \\ A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix}, \quad J_R = \begin{pmatrix} 2 \cdot {}^t\mathbf{v}_1 & 2 \cdot {}^t\mathbf{v}_2 \\ N_1 & -N_2 \\ N_2 & N_1 \end{pmatrix},$$

respectively. Then, we have the following lemma.

Lemma 5.4. *We have* $\mathrm{rank}(J_L) = 2(m+n-d+1)$.

Proof. For $A_1 = [C_m(A)_1 \ \ C_n(B)_1]$, let $\overline{C_m(A)_1}$ be the right $m-d$ columns of $C_m(A)_1$ and $\overline{C_n(B)_1}$ be the right $n-d$ columns of $C_n(B)_1$. Then, we see that the bottom $m+n-2d$ rows of the matrix $\bar{C} = [\overline{C_m(A)_1} \ \overline{C_n(B)_1}]$ is equal to the matrix consisting of the real part of the elements of $N(A,B)$, the Sylvester matrix of $A(x)$ and $B(x)$. By the assumption, polynomials $A(x)$ and $B(x)$ are relatively prime, and there exist no nonzero elements in $\bar{C}$ except for the bottom $m+n-2d$ rows, thus we have $\mathrm{rank}(\bar{C}) = m+n-2d$.

By the structure of $\bar{C}$ and the lower triangular structure of $C_m(A)_1$ and $C_n(B)_1$, we can take the left $d+1$ columns of $C_m(A)_1$ or $C_n(B)_1$ satisfying linear independence along with $\bar{C}$, which implies that there exist a nonsingular square matrix T of order $m+n+2$ satisfying

$$A_1 T = R, \tag{5.35}$$

where R is a lower triangular matrix, thus we have $\mathrm{rank}(A_1) = \mathrm{rank}(R) = m+n-d+1$.

Furthermore, by using T and R in (5.35), we have

$$\begin{pmatrix} 0 & 0 \\ A_1 & -A_2 \\ A_2 & A_1 \end{pmatrix} \begin{pmatrix} T & 0 \\ 0 & T \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ R & -A_2 T \\ A_2 T & R \end{pmatrix}, \tag{5.36}$$

followed by a suitable transformation on columns on the matrix in the right-hand-side of (5.36), we can make $A_2 T$ to zero matrix, which implies that

$$\mathrm{rank}(J_L) = \mathrm{rank}\left(\begin{pmatrix} 0 & 0 \\ R & -A_2 T \\ A_2 T & R \end{pmatrix} \right) = 2 \cdot \mathrm{rank}(R) = 2(m+n-d+1).$$

This proves the lemma. □

Proof of Proposition 5.2 (in the complex coefficient case, continued). By the assumptions, we have at least one nonzero coordinate in the top row in J_R, while we have no nonzero coordinate in the top row in J_L, thus we have $\mathrm{rank}(J_{\mathbf{q}}(\mathbf{x})) = 2(m+n-d+1)+1$, which proves the proposition in the complex coefficient case. □

Proposition 5.2 says that, so long as the search direction in the minimization problem satisfies that corresponding polynomials have a GCD of degree not exceeding d, then $J_{\mathbf{q}}(\mathbf{x})$ has full rank, thus we can safely calculate the next search direction for approximate GCD.

5.3.3 Setting the Initial Values

At the beginning of iterations, we give the initial value $\boldsymbol{x}_0$ by using the singular value decomposition (SVD) [12], as follows.

The Real Coefficient Case

In the case of the real coefficients, we calculate the SVD of the $(d-1)$-th subresultant matrix $N_{d-1}(F,G) : \mathbf{R}^{m+n-2d} \to \mathbf{R}^{m+n-d}$ in (5.2). Let $N_{d-1}(F,G) = U\Sigma\,{}^tV$ be the SVD of $N_{d-1}(F,G)$, where

$$\begin{aligned} N_{d-1}(F,G) &= U\Sigma\,{}^tV, \quad U = (\boldsymbol{u}_1,\dots,\boldsymbol{u}_{m+n-2d}),\\ \Sigma &= \operatorname{diag}(\sigma_1,\dots,\sigma_{m+n-2d}), \quad V = (\boldsymbol{v}_1,\dots,\boldsymbol{v}_{m+n-2d}), \end{aligned} \tag{5.37}$$

with $\boldsymbol{u}_j \in \mathbf{R}^{m+n-d}$, $\boldsymbol{v}_j \in \mathbf{R}^{m+n-2d}$, and $\Sigma = \operatorname{diag}(\sigma_1,\dots,\sigma_{m+n-2d})$ denotes the diagonal matrix whose the j-th diagonal element is σ_j. Note that U and V are orthogonal matrices. Then, by a property of the SVD [12, Theorem 3.3], the smallest singular value σ_{m+n-2d} gives the minimum distance of the image of the unit sphere $S^{m+n-2d-1}$, given as

$$S^{m+n-2d-1} = \{\boldsymbol{x} \in \mathbf{R}^{m+n-2d} \mid \|\boldsymbol{x}\|_2 = 1\},$$

by N_{d-1}, represented as

$$N_{d-1}\cdot S^{m+n-d-1} = \{N_{d-1}\boldsymbol{x} \mid \boldsymbol{x} \in \mathbf{R}^{m+n-2d}, \|\boldsymbol{x}\|_2 = 1\},$$

from the origin, along with $\sigma_{m+n-2d}\boldsymbol{u}_{m+n-2d}$ as its coordinates. By (5.37), we have

$$N_{d-1}\cdot \boldsymbol{v}_{m+n-2d} = \sigma_{m+n-2d}\boldsymbol{u}_{m+n-2d},$$

thus $\boldsymbol{v}_{m+n-2d}$ represents the coefficients of $A(x)$ and $B(x)$: let

$$\begin{aligned} \boldsymbol{v}_{m+n-2d} &= {}^t(\bar{a}_{n-d},\dots,\bar{a}_0,\bar{b}_{n-d},\dots,\bar{b}_0),\\ \bar{A}(x) &= \bar{a}_{n-d}x^{n-d} + \cdots + \bar{a}_0x^0,\\ \bar{B}(x) &= \bar{b}_{m-d}x^{m-d} + \cdots + \bar{b}_0x^0. \end{aligned}$$

Then, $\bar{A}(x)$ and $\bar{B}(x)$ give the least norm of $AF+BG$ satisfying $\|A\|_2^2 + \|B\|_2^2 = 1$ by putting $A(x) = \bar{A}(x)$ and $B(x) = \bar{B}(x)$.

Therefore, we admit the coefficients of F, G, $\bar{A}$ and $\bar{B}$ as the initial values of the iterations as

$$\boldsymbol{x}_0 = (f_m,\dots,f_0,g_n,\dots,g_0,\bar{a}_{n-d},\dots,\bar{a}_0,\bar{b}_{m-d},\dots,\bar{b}_0). \tag{5.38}$$

The Complex Coefficient Case

In the complex case, we calculate the SVD of $N = \begin{pmatrix} N_1 & -N_2 \\ N_2 & N_1 \end{pmatrix}$ in (5.18) as

$$\begin{aligned} N &= U\Sigma\,{}^tV, \quad U = (\boldsymbol{u}_1,\dots,\boldsymbol{u}_{2(m+n-2d+2)}),\\ \Sigma &= \operatorname{diag}(\sigma_1,\dots,\sigma_{2(m+n-2d+2)}), \quad V = (\boldsymbol{v}_1,\dots,\boldsymbol{v}_{2(m+n-2d+2)}), \end{aligned} \tag{5.39}$$

where $\boldsymbol{u}_j \in \mathbf{R}^{2(m+n-d+1)}$, $\boldsymbol{v}_j \in \mathbf{R}^{2(m+n-2d+2)}$, and U and V are orthogonal matrices. Then, as in the case of the real coefficients, the smallest singular value

$\sigma_{2(m+n-2d+2)}$ gives the minimum distance of the image of the unit sphere $S^{2(m+n-2d+2)-1}$, given as

$$S^{2(m+n-2d+2)-1} = \{\boldsymbol{x} \in \mathbf{R}^{2(m+n-2d+2)} \mid \|\boldsymbol{x}\|_2 = 1\},$$

by N, represented as

$$N \cdot S^{2(m+n-2d+1)-1} = \{N\boldsymbol{x} \mid \boldsymbol{x} \in \mathbf{R}^{2(m+n-2d+2)}, \|\boldsymbol{x}\|_2 = 1\},$$

from the origin, along with $\sigma_{2(m+n-2d+2)}\boldsymbol{u}_{2(m+n-2d+2)}$ as its coordinates. By (5.39), we have

$$N \cdot \boldsymbol{v}_{2(m+n-2d+2)} = \sigma_{2(m+n-2d+2)}\boldsymbol{u}_{2(m+n-2d+2)},$$

thus $\boldsymbol{v}_{2(m+n-2d+2)}$ represents the coefficients of $A(x)$ and $B(x)$: let

$$\begin{aligned}\boldsymbol{v}_{2(m+n-2d+2)} = {}^t(&\bar{a}_{n-d,1}, \dots, \bar{a}_{0,1}, \bar{b}_{n-d,1}, \dots, \bar{b}_{0,1},\\ &\bar{a}_{n-d,2}, \dots, \bar{a}_{0,2}, \bar{b}_{n-d,2}, \dots, \bar{b}_{0,2}),\\ \bar{A}(x) &= (\bar{a}_{n-d,1} + \bar{a}_{n-d,2}\boldsymbol{i})x^{n-d} + \cdots + (\bar{a}_{0,1} + \bar{a}_{0,2}\boldsymbol{i})x^0,\\ \bar{B}(x) &= (\bar{b}_{m-d,1} + \bar{b}_{m-d,2}\boldsymbol{i})x^{m-d} + \cdots + (\bar{b}_{0,1} + \bar{b}_{0,2}\boldsymbol{i})x^0.\end{aligned}$$

Then, $\bar{A}(x)$ and $\bar{B}(x)$ give the least norm of $AF + BG$ satisfying $\|A\|_2^2 + \|B\|_2^2 = 1$ by putting $A(x) = \bar{A}(x)$ and $B(x) = \bar{B}(x)$ in (5.13).

Therefore, we admit the coefficients of F, G, $\bar{A}$ and $\bar{B}$ as the initial values of the iterations as

$$\begin{aligned}\boldsymbol{x}_0 = (&f_{m,1}, \dots, f_{0,1}, g_{n,1}, \dots, g_{0,1}, f_{m,2}, \dots, f_{0,2}, g_{n,2}, \dots, g_{0,2},\\ &\bar{a}_{n-d,1}, \dots, \bar{a}_{0,1}, \bar{b}_{n-d,1}, \dots, \bar{b}_{0,1}, \bar{a}_{n-d,2}, \dots, \bar{a}_{0,2}, \bar{b}_{n-d,2}, \dots, \bar{b}_{0,2}).\end{aligned} \tag{5.40}$$

5.3.4 Regarding the Minimization Problem as the Minimum Distance (Least Squares) Problem

Since we have the object function f as in (5.11) or (5.22) in the case of the real or the complex coefficients, respectively, we have $\nabla f(\boldsymbol{x})$ as

$$2 \cdot {}^t(x_1 - f_m, \dots, x_{m+1} - f_0, x_{m+2} - g_n, \dots, x_{m+n+2} - g_0, 0, \dots, 0),$$

in the case of the real coefficients, or

$$\begin{aligned}2 \cdot {}^t(x_1 - f_{m,1}, \dots, x_{m+1} - f_{0,1}, x_{m+2} - g_{n,1}, \dots, x_{m+n+2} - g_{0,1},\\ x_{m+n+3} - f_{m,2}, \dots, x_{2m+n+3} - f_{0,2},\\ x_{2m+n+4} - g_{n,2}, \dots, x_{2(m+n+2)} - g_{0,2}, 0, \dots, 0),\end{aligned}$$

in the case of the complex coefficients, respectively. However, we can regard our problem as finding a point $\boldsymbol{x} \in V_{\boldsymbol{q}}$ which has the minimum distance to the initial point $\boldsymbol{x}_0$ with respect to the $(x_1, \dots, x_{m+n+2})$-coordinates in the case of the real coefficients or the $(x_1, \dots, x_{2(m+n+2)})$-coordinates in the case of the complex coefficients, respectively, which correspond to the coefficients in $F(x)$ and $G(x)$. Therefore, in the gradient projection method at $\boldsymbol{x} \in V_{\boldsymbol{q}}$, the projection of $-\nabla f(\boldsymbol{x})$ in (5.27) should be the projection of

$${}^t(x_1 - f_m, \dots, x_{m+1} - f_0, x_{m+2} - g_n, \dots, x_{m+n+2} - g_0, 0, \dots, 0),$$

in the case of the real coefficients, or

$$\begin{aligned}{}^t(x_1 - f_{m,1}, \dots, x_{m+1} - f_{0,1}, x_{m+2} - g_{n,1}, \dots, x_{m+n+2} - g_{0,1},\\ x_{m+n+3} - f_{m,2}, \dots, x_{2m+n+3} - f_{0,2},\\ x_{2m+n+4} - g_{n,2}, \dots, x_{2(m+n+2)} - g_{0,2}, 0, \dots, 0),\end{aligned}$$

in the case of the complex coefficients, respectively, onto $T_{\mathbf{x}}$. These changes are equivalent to changing the objective function as $\bar{f}(\mathbf{x}) = \frac{1}{2} f(\mathbf{x})$ then solving the minimization problem of $\bar{f}(\mathbf{x})$, subject to $\mathbf{q}(\mathbf{x}) = \mathbf{0}$.

5.3.5 Calculating the Actual GCD and Correcting the Deformed Polynomials

After successful end of the iterations in Algorithms 1 or 2, we obtain the coefficients of $\tilde{F}(x)$, $\tilde{G}(x)$, $A(x)$ and $B(x)$ satisfying (5.3) with $A(x)$ and $B(x)$ are relatively prime. Then, we need to compute the actual GCD $H(x)$ of $\tilde{F}(x)$ and $\tilde{G}(x)$. Although H can be calculated as the quotient of $\tilde{F}$ divided by B or $\tilde{G}$ divided by A, naive polynomial division may cause numerical errors in the coefficient. Thus, we calculate the coefficients of H by the so-called least squares division [66], followed by correcting the coefficients in $\tilde{F}$ and $\tilde{G}$ by using the calculated H, as follows.

Calculating Candidates for the GCD in the Real Coefficient Case

For polynomials $\tilde{F}$, $\tilde{G}$, A and B represented as in (5.4) and H represented as

$$H(x) = h_d x^d + \cdots + h_0 x^0,$$

solve the equations $HB = \tilde{F}$ and $HA = \tilde{G}$ with respect to H as solving the least squares problems of linear systems

$$C_d(A)\,{}^t(h_d, \dots, h_0) = {}^t(\tilde{g}_n, \dots, \tilde{g}_0), \tag{5.41}$$

$$C_d(B)\,{}^t(h_d, \dots, h_0) = {}^t(\tilde{f}_m, \dots, \tilde{f}_0), \tag{5.42}$$

respectively. Let $H_1(x), H_2(x) \in \mathbf{R}[x]$ be the candidates for the GCD whose coefficients are calculated as the least squares solutions of (5.41) and (5.42), respectively.

Calculating Candidates for the GCD in the Complex Coefficient Case

For polynomials $\tilde{F}$, $\tilde{G}$, A and B represented as in (5.13) and H represented as

$$H(x) = (h_{d,1} + h_{d,2}\boldsymbol{i})x^d + \cdots + (h_{0,1} + h_{0,2}\boldsymbol{i})x^0,$$

solve the equations $HB = \tilde{F}$ and $HA = \tilde{G}$ with respect to H as solving the least squares problems of linear systems

$$C_d(A)\,{}^t(h_{d,1} + h_{d,2}\boldsymbol{i}, \dots, h_{0,1} + h_{0,2}\boldsymbol{i}) = {}^t(\tilde{g}_{n,1} + \tilde{g}_{n,2}\boldsymbol{i}, \dots, \tilde{g}_{0,1} + \tilde{g}_{0,2}\boldsymbol{i}), \tag{5.43}$$

$$C_d(B)\,{}^t(h_{d,1} + h_{d,2}\boldsymbol{i}, \dots, h_{0,1} + h_{0,2}\boldsymbol{i}) = {}^t(\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i}, \dots, \tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i}), \tag{5.44}$$

respectively. Then, we transfer the linear systems (5.43) and (5.44), as follows. For (5.44), let us express the matrices and vectors as the sum of the real and the imaginary part of which, respectively, as

$$\begin{aligned} C_d(B) &= B_1 + \boldsymbol{i}B_2,\\ {}^t(h_{d,1} + h_{d,2}\boldsymbol{i}, \dots, h_{0,1} + h_{0,2}\boldsymbol{i}) &= \mathbf{h}_1 + \boldsymbol{i}\mathbf{h}_2,\\ {}^t(\tilde{f}_{m,1} + \tilde{f}_{m,2}\boldsymbol{i}, \dots, \tilde{f}_{0,1} + \tilde{f}_{0,2}\boldsymbol{i}) &= \mathbf{f}_1 + \boldsymbol{i}\mathbf{f}_2. \end{aligned}$$

Then, (5.42) is expressed as

$$(B_1 + iB_2)(\mathbf{h}_1 + i\mathbf{h}_2) = (\mathbf{f}_1 + i\mathbf{f}_2). \tag{5.45}$$

By equating the real and the imaginary parts in Eq. (5.45), respectively, we have

$$(B_1\mathbf{h}_1 - B_2\mathbf{h}_2) = \mathbf{f}_1, \quad (B_1\mathbf{h}_2 + B_2\mathbf{h}_1) = \mathbf{f}_2,$$

or

$$\begin{pmatrix} B_1 & -B_2 \\ B_2 & B_1 \end{pmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{pmatrix} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{pmatrix}. \tag{5.46}$$

Thus, we can calculate the coefficients of $H(x)$ by solving the real least squares problem (5.46). We can solve (5.43) similarly. Let $H_1(x), H_2(x) \in \mathbf{C}[x]$ be the candidates for the GCD whose coefficients are calculated as the least squares solutions of (5.43) and (5.44), respectively.

Choosing the GCD and Calculating the Deformed Polynomials

Let $H_1(x), H_2(x) \in \mathbf{C}[x]$ be the candidates for the GCD calculated as in the above. Then, for $i = 1, 2$, calculate the norms of the residues as

$$r_i = \|\tilde{F} - H_i B\|_2^2 + \|\tilde{G} - H_i A\|_2^2,$$

respectively, and set the GCD $H(x)$ be $H_i(x)$ giving the minimum value of r_i.

Finally, for the chosen $H(x)$, correct the coefficients of $\tilde{F}(x)$ and $\tilde{G}(x)$ as

$$\tilde{F}(x) = H(x) \cdot B(x), \quad \tilde{G}(x) = H(x) \cdot A(x),$$

respectively.

5.3.6 The Algorithm

Summarizing the above, the algorithm for calculating approximate GCD becomes as follows.

Algorithm 3 (GPGCD: Approximate GCD by the Gradient-Projection Method)**.**

- Inputs:
  - $F(x), G(x) \in \mathbf{R}[x]$ or $\mathbf{C}[x]$ with $\deg(F) \geqslant \deg(G) > 0$,
  - $d \in \mathbf{N}$: the degree of approximate GCD with $d \leqslant \deg(G)$,
  - $\varepsilon > 0$: a threshold for terminating iteration in the gradient-projection method,
  - $u \in \mathbf{N}$: an upper bound for the number of iterations permitted in the gradient-projection method.
- Outputs: $\tilde{F}(x), \tilde{G}(x), H(x) \in \mathbf{R}[x]$ or $\mathbf{C}[x]$ such that $\tilde{F}$ and $\tilde{G}$ are deformations of F and G, respectively, whose GCD is equal to H with $\deg(H) = d$.

STEP 1 **[Setting the initial values]** As the discussions in Section 5.3.3, set the initial values $\mathbf{x}_0$ as in (5.38) in the case of the real coefficients, or (5.40) in the case of the complex coefficients, respectively.

STEP 2 [Iteration] As the discussions in Section 5.3.4, solve the minimization problem of $\bar{f}(\mathbf{x}) = \frac{1}{2}f(\mathbf{x})$, subject to $\mathbf{q}(\mathbf{x}) = \mathbf{0}$, with $f(\mathbf{x})$ and $\mathbf{q}(\mathbf{x})$ as in (5.11) and (5.12) in the case of the real coefficients, or in (5.22) and (5.23) in the case of the complex coefficients, respectively. Apply Algorithm 1 or 2 for the minimization: repeat iterations until the search direction $\mathbf{d}_k$ (as in (5.27) in the gradient-projection method or in (5.30) in a modified Newton method, respectively) satisfies $\|\mathbf{d}_k\|_2 < \varepsilon$, or the number of iteration reaches its upper bound u.

STEP 3 [Construction of $\tilde{F}$, $\tilde{G}$ and H] As the discussions in Section 5.3.5, construct the GCD $H(x)$ and correct the coefficients of $\tilde{F}(x)$ and $\tilde{G}(x)$. Then, return $\tilde{F}(x)$, $\tilde{G}(x)$ and $H(x)$. If Step 2 did not end with the number of iterations less than u, report it to the user.

5.3.7 Preserving Monicity

While Algorithm 3 permits changing the leading coefficients for calculating $\tilde{F}(x)$ and $\tilde{G}(x)$, we can also give an algorithm restricting inputs $F(x)$ and $G(x)$ and outputs $\tilde{F}(x)$ and $\tilde{G}(x)$ to be monic as follows, in the case of the real coefficients.

Let $\tilde{F}(x)$ and $\tilde{G}(x)$ be represented as in (5.4) with $\tilde{f}_m = \tilde{g}_n = 1$, then, by Eq. (5.6), we have $b_{m-d} = -a_{n-d}$. Thus, we eliminate the variables $\tilde{f}_m$, $\tilde{g}_n$ and b_{m-d}, which cause the following changes.

Changes on the Subresultant Matrix

By eliminating the variables as in the above, we see that Eq. (5.6) is equivalent to

$$N'_{d-1}(\tilde{F},\tilde{G}) \cdot {}^t(a_{m-d},\dots,a_0,b_{n-d-1},\dots,b_0) = \mathbf{0},$$

where $N'_{d-1}(\tilde{F},\tilde{G})$ is defined as

$$N'_{d-1}(\tilde{F},\tilde{G}) = \begin{pmatrix} \tilde{f}_{m-1} - \tilde{g}_{n-1} & 1 & & & 1 & & \\ \vdots & \tilde{f}_{m-1} & \ddots & & \tilde{g}_{n-1} & \ddots & \\ \tilde{f}_0 - \tilde{g}_{n-m} & \vdots & \ddots & 1 & \vdots & \ddots & 1 \\ & \tilde{f}_0 & & \tilde{f}_{m-1} & \tilde{g}_0 & & \tilde{g}_{n-1} \\ & & \ddots & \vdots & & \ddots & \vdots \\ & & & \tilde{f}_0 & & & \tilde{g}_0 \end{pmatrix},$$

with (in the first column) $\tilde{g}_j = 0$ for $j < 0$, by subtracting the first column by the $(n-d+1)$-th column, then deleting the first row and the $(n-d+1)$-th column (corresponding to the b_{m-d} term) in $N_{d-1}(\tilde{F},\tilde{G})$.

Changes on the Settings in the Minimization Problem

In solving the minimization problem, we substitute the variables

$$(\tilde{f}_{m-1},\dots,\tilde{f}_0,\tilde{g}_{n-1},\dots,\tilde{g}_0,a_{n-d},\dots,a_0,b_{m-d-1},\dots,b_0)$$

as $\mathbf{x} = (x_1,\dots,x_{2(m+n-d)+1})$, instead of (5.10).

As a consequence, in contrast to (5.11), the objective function $f(\mathbf{x})$ becomes as

$$f(\mathbf{x}) = (x_1 - f_{m-1})^2 + \cdots + (x_m - f_0)^2 \\ + (x_{m+1} - g_{n-1})^2 + \cdots + (x_{m+n} - g_0)^2. \quad (5.47)$$

Also, in contrast to (5.8) and (5.9), the constraints $\mathbf{q}(\mathbf{x})$ become as

$$\begin{aligned} q_0 &= 2a_{n-d}^2 + a_{n-d-1}^2 \cdots + a_0^2 + b_{m-d-1}^2 + \cdots + b_0^2 - 1 = 0, \\ q_1 &= (\tilde{f}_{m-1} - \tilde{g}_{n-1})a_{n-d} + a_{n-d-1} + b_{m-d-1} = 0, \\ &\vdots \\ q_{m+n-d+1} &= \tilde{f}_0 a_0 + \tilde{g}_0 b_0 = 0. \end{aligned} \tag{5.48}$$

Changes on the Initial Values

Let $N'_{d-1} = U\,\Sigma\,{}^tV$ be the SVD of $N'_{d-1}(F,G)$, with

$$\begin{aligned} V &= {}^t(\boldsymbol{v}_1,\dots,\boldsymbol{v}_{m+n-2d-1}), \\ \boldsymbol{v}_{m+n-2d-1} &= {}^t(\bar{a}_{n-d},\dots,\bar{a}_0,\bar{b}_{m-d-1},\dots,\bar{b}_0), \end{aligned}$$

Then, in contrast to (5.38), the initial values become as

$$\mathbf{x}_0 = (f_{m-1},\dots,f_0,g_{n-1},\dots,g_0,\bar{a}_{n-d},\dots,\bar{a}_0,\bar{b}_{m-d-1},\dots,\bar{b}_0). \tag{5.49}$$

The Algorithm

Summarizing discussions in the above, for preserving $\tilde{F}(x)$ and $\tilde{G}(x)$ to be monic, we modify Algorithm 3 as follows.

Algorithm 4 (GPGCD with preserving monicity)**.** Change Steps 1 and 2 in Algorithm 3 as follows.

STEP 1 **[Setting the initial values]** Set the initial values $\mathbf{x}_0$ as in (5.49).

STEP 2 **[Iteration]** Solve the minimization problem of $\bar{f}(\mathbf{x}) = \frac{1}{2}(\mathbf{x})$, subject to $\mathbf{q}(\mathbf{x}) = 0$, with $f(\mathbf{x})$ and $\mathbf{q}(\mathbf{x})$ defined as in (5.47) and (5.48), respectively, as Step 2 in Algorithm 3.

5.3.8 Examples

Now we show examples of Algorithm 3 in the case of the real coefficients (more comprehensive experiments are presented in the next section).

Note that, for the minimization method, we have employed a modified Newton method (Algorithm 2). Computations in Example 5.1 have been executed on a computer algebra system Mathematica 6 with hardware floating-point arithmetic, while those in Examples 5.2 and 5.3 have been executed on another computer algebra system Maple 12 with `Digits=10`.

Example 5.1*.* This example is given by Karmarker and Lakshman [26], followed by Kaltofen *et al* [25]. Let $F(x), G(x) \in \mathbf{R}[x]$ be

$$\begin{aligned} F(x) &= x^2 - 6x + 5 = (x-1)(x-5), \\ G(x) &= x^2 - 6.3x + 5.72 = (x-1.1)(x-5.2), \end{aligned}$$

and find $\tilde{F}(x), \tilde{G}(x) \in \mathbf{R}[x]$ which have the GCD of degree 1, namely $\tilde{F}(x)$ and $\tilde{G}(x)$ have one common zero.

Case 1: The leading coefficient can be perturbed. Applying Algorithm 3 to F and G, with $d = 1$ and $\varepsilon = 1.0 \times 10^{-8}$, after 7 iterations, we obtain the polynomials $\tilde{F}$ and $\tilde{G}$ as

$$\begin{aligned}\tilde{F}(x) &= 0.985006x^2 - 6.00294x + 4.99942,\\ \tilde{G}(x) &= 1.01495x^2 - 6.29707x + 5.72058,\end{aligned}$$

with perturbations as $\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2 = 0.0004663065027$ and the common zero of $\tilde{F}(x)$ and $\tilde{G}(x)$ as $x = 5.09890419203$. In Kaltofen *et al* [25], the calculated perturbations obtained is 0.0004663 with the common zero as $x = 5.09890429$. Karmarker and Lakshman [26] only give an example without perturbations on the leading coefficients.

Case 2: The leading coefficient cannot be perturbed. Applying Algorithm 3 (preserving monicity) with the same arguments as in Case 1, after 7 iterations, we obtain the polynomials $\tilde{F}$ and $\tilde{G}$ as

$$\begin{aligned}\tilde{F}(x) &= x^2 - 6.07504x + 4.98528,\\ \tilde{G}(x) &= x^2 - 6.22218x + 5.73527,\end{aligned}$$

with perturbations as $\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2 = 0.01213604416$ and the common zero of $\tilde{F}(x)$ and $\tilde{G}(x)$ as $x = 5.0969464650$. In Kaltofen *et al* [25], the calculated perturbations obtained is 0.01213604583 with the common zero as $x = 5.0969478$. In Karmarker and Lakshman [26], the calculated perturbations obtained is 0.01213605293 with the common zero as $x = 5.096939087$.

The next examples, originally by Sanuki and Sasaki [41], are ill-conditioned ones with the small or large leading coefficient GCD.

Example 5.2 (A small leading coefficient problem [41, Example 4]). Let $F(x)$ and $G(x)$ be

$$\begin{aligned}F(x) &= (x^4 + x^2 + x + 1)(0.001x^2 + x + 1),\\ G(x) &= (x^3 + x^2 + x + 1)(0.001x^2 + x + 1).\end{aligned}$$

Applying Algorithm 3 to F and G, with $d = 2$ and $\varepsilon = 1.0 \times 10^{-8}$, after 1 iteration, we obtain the polynomials $\tilde{F}$, $\tilde{G}$ and H as

$$\begin{aligned}&\tilde{F}(x) \simeq F(x), \quad \tilde{G}(x) \simeq G(x),\\ &H(x) = 0.001x^2 + 0.9999999936x + 0.9999999936,\end{aligned}$$

with $\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2 = 7.2 \times 10^{-23}$.

Example 5.3 (A big leading coefficient problem [41, Example 5]). Let $F(x)$ and $G(x)$ be

$$\begin{aligned}F(x) &= (x^6 - 0.00001(0.8x^5 + 3x^4 - 4x^3 - 4x^2 - 5x + 1)) \cdot C(x),\\ G(x) &= (x^5 + x^4 + x^3 - 0.1x^2 + 1) \cdot C(x),\end{aligned}$$

with $C(x) = x^2 + 0.001$. Applying Algorithm 3 to F and G, with $d = 2$ and $\varepsilon = 1.0 \times 10^{-8}$, after 1 iteration, we obtain the polynomials $\tilde{F}$, $\tilde{G}$ and H as

$$\begin{aligned}&\tilde{F}(x) \simeq F(x), \quad \tilde{G}(x) \simeq G(x),\\ &H(x) = x^2 + 1.548794164 \times 10^{-16}x + 0.001,\end{aligned}$$

with $\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2 = 3.01 \times 10^{-28}$.

5.4 EXPERIMENTS

We have implemented our GPGCD method (Algorithm 3) on computer algebra systems Mathematica and Maple, and carried out the following tests:

- Comparison of performance of the gradient-projection method (Algorithm 1) and the modified Newton method (Algorithm 2) (Section 5.4.1),
- Comparison of performance of the GPGCD method with a method based on the structured total least norm (STLN) method [24] (Section 5.4.2),

on randomly generated polynomials with approximate GCD. Note that, in the former test, we have tested only the case of the real coefficients, while, in the latter case, we have tested both the cases of the real and the complex coefficients.

In the tests, we have generated random polynomials with GCD then added noise, as follows. First, we have generated a pair of monic polynomials $F_0(x)$ and $G_0(x)$ of degrees m and n, respectively, with the GCD of degree d. The GCD and the prime parts of degrees $m-d$ and $n-d$ are generated as monic polynomials and with random coefficients $c \in [-10, 10]$ of floating-point numbers. For noise, we have generated a pair of polynomials $F_N(x)$ and $G_N(x)$ of degrees $m-1$ and $n-1$, respectively, with random coefficients as the same as for $F_0(x)$ and $G_0(x)$. Then, we have defined a pair of test polynomials $F(x)$ and $G(x)$ as

$$F(x) = F_0(x) + \frac{e_F}{\|F_N(x)\|_2} F_N(x), \quad G(x) = G_0(x) + \frac{e_G}{\|G_N(x)\|_2} G_N(x),$$

respectively, scaling the noise such that the 2-norm of the noise for F and G is equal to e_F and e_G, respectively. In the present test, we set $e_F = e_G = 0.1$.

The tests have been carried out on Intel Core2 Duo Mobile Processor T7400 (in Apple MacBook "Mid-2007" model) at 2.16 GHz with RAM 2GB, under MacOS X 10.5.

5.4.1 Comparison of the Gradient-Projection Method and the Modified Newton Method

In this test, we have used an implementation on Mathematica and compared performance of the gradient-projection method (Algorithm 1) and a modified Newton method (Algorithm 2), only in the case of the real coefficients. For every example, we have generated one random test polynomial as in the above, and we have applied Algorithm 3 (preserving monicity) with $u = 100$ and $\varepsilon = 1.0 \times 10^{-8}$.

Table 5.1 shows the result of the test: m and n denotes the degree of a tested pair F and G, respectively, and d denotes the degree of approximate GCD; "Error" is the perturbation

$$\|\tilde{F} - F\|_2^2 + \|\tilde{G} - G\|_2^2, \tag{5.50}$$

where "ae$-b$" denotes $a \times 10^{-b}$; "#Iterations" is the number of iterations; "Time" is computing time in seconds. The columns with "Alg. 1" and "Alg. 2" are the data for Algorithm 1 (the gradient-projection method) and Algorithm 2 (the modified Newton method), respectively. Note that, the "Error" is a single column since both algorithms give almost the same values in each examples.

We see that, in all the test cases, the number of iterations of the gradient-projection method (Algorithm 1) is equal to 3, which is smaller than that of the modified Newton method (Algorithm 2) which is equal to 4. However, an iteration in Algorithm 1 includes solving a linear system at least twice: once in the *projection step* (Step 2)

Ex.	m, n	d	Error	#Iterations		Time (sec.)	
				Alg. 1	Alg. 2	Alg. 1	Alg. 2
1	10, 10	5	7.65e−3	3	4	0.08	0.05
2	30, 30	10	3.10e−3	3	4	2.05	0.80
3	40, 40	20	3.60e−3	3	4	3.37	1.33
4	60, 60	30	7.27e−3	3	4	10.14	4.41
5	80, 80	40	5.24e−3	3	4	22.61	10.39
6	100, 100	50	4.92e−3	3	4	42.88	20.34

Table 5.1: Test results comparing the gradient-projection method and the modified Newton method; see Section 5.4.1 for details.

and at least once in the *restoration step* (Step 3); whereas an iteration in Algorithm 2 includes that only once. Thus, total number of solving a linear system in Algorithm 2 is about a half of that in Algorithm 1. Furthermore, computing time shows that, although both implementations are rather inefficient because of elementary implementations, a modified Newton method runs approximately twice as fast as the gradient projection method. Therefore, we adopt Algorithm 2 as the method of minimization in the GPGCD method (Algorithm 3).

5.4.2 Tests on Large Sets of Randomly-generated Polynomials

In this test, we have used our implementation on Maple and compared its performance with a method based on the structured total least norm (STLN) method [24], using their implementation, in the both cases of the real and the complex coefficients. In our implementation of Algorithm 3, we have chosen the modified Newton method (Algorithm 2) for minimization, while, in the STLN-based method, we have used their procedure `R_con_mulpoly` and `C_con_mulpoly`, which calculates the approximate GCD of several polynomials in $\mathbf{R}[x]$ and $\mathbf{C}[x]$, respectively. The tests have been carried out on Maple 12 with `Digits=15` executing hardware floating-point arithmetic.

For every example, we have generated 100 random test polynomials as in the above. In executing Algorithm 3, we set $u = 200$ and $\varepsilon = 1.0 \times 10^{-8}$; in `R_con_mulpoly` and `C_con_mulpoly`, we set the tolerance $e = 1.0 \times 10^{-8}$.

Tables 5.2 and 5.3 show the results of the test in the case of the real and the complex coefficients, respectively: m and n denotes the degree of a pair F and G, respectively, and d denotes the degree of approximate GCD. The columns with "STLN" are the data for the STLN-based method, while "GPGCD" are the data for the GPGCD method (Algorithm 3). In Table 5.2, "#Fail" is the number of "failed" cases such as: in the STLN-based method, the number of iterations exceeds 50 times (which is the built-in threshold in the program), while, in the GPGCD method, the perturbation (5.50) exceeds 1 (note that, in the GPGCD method, all the iterations have converged within far less than 200 times). Note that, in contrast to the test case with the real coefficients, both the STLN-based and the GPGCD methods have converged in all the test cases with the complex coefficients, within the number of iterations above and sufficiently small amount of perturbations as approximately equal to those shown as in Table 5.3. All the other data are the average over results for the "not failed" cases: "Error", "#Iterations" and "Time" are the same as those in Table 5.1, respectively.

We see that, in the most of tests, both methods calculate approximate GCD with almost the same amount of perturbations, while GPGCD method runs much faster

Ex.	m, n	d	#Fail		Error		#Iterations		Time (sec.)	
			STLN	GPGCD	STLN	GPGCD	STLN	GPGCD	STLN	GPGCD
1	10, 10	5	0	2	3.63e−3	3.67e−3	4.65	4.99	0.43	0.05
2	20, 20	10	0	4	4.37e−3	4.28e−3	4.97	4.78	1.33	0.09
3	30, 30	15	2	1	4.65e−3	4.64e−3	4.34	5.28	2.54	0.16
4	40, 40	20	0	0	4.73e−3	4.73e−3	4.28	4.54	4.41	0.23
5	50, 50	25	0	0	4.79e−3	4.79e−3	4.32	4.51	6.96	0.33
6	60, 60	30	0	0	4.82e−3	4.54e−3	4.27	4.45	10.44	0.45
7	70, 70	35	1	1	4.71e−3	4.71e−3	3.97	4.27	13.28	0.58
8	80, 80	40	0	2	4.77e−3	4.77e−3	4.06	4.34	17.96	0.78
9	90, 90	45	0	1	5.10e−3	4.94e−3	4.18	4.29	23.61	0.97
10	100, 100	50	1	0	4.82e−3	4.81e−3	4.11	4.56	29.87	1.28

Table 5.2: Test results for large sets of polynomials with approximate GCD, in the case of the real coefficients; see Section 5.4.2 for details.

Ex.	m, n	d	Error		#Iterations		Time (sec.)	
			STLN	GPGCD	STLN	GPGCD	STLN	GPGCD
1	10, 10	5	3.72e−3	3.72e−3	4.48	4.43	1.79	0.15
2	20, 20	10	4.16e−3	4.16e−3	4.24	4.22	5.88	0.30
3	30, 30	15	4.33e−3	4.33e−3	4.54	4.48	14.29	0.58
4	40, 40	20	4.48e−3	4.48e−3	4.08	4.08	24.10	0.88
5	50, 50	25	4.63e−3	4.64e−3	4.05	4.12	39.19	1.36
6	60, 60	30	4.61e−3	4.61e−3	4.02	4.06	60.48	1.96
7	70, 70	35	4.82e−3	4.82e−3	3.90	4.02	84.51	2.66
8	80, 80	40	4.84e−3	4.84e−3	3.88	4.04	116.03	3.65
9	90, 90	45	4.79e−3	4.79e−3	3.85	4.01	151.27	4.66
10	100, 100	50	4.77e−3	4.78e−3	3.83	4.06	199.48	6.00

Table 5.3: Test results for large sets of polynomials with approximate GCD, in the case of the complex coefficients; see Section 5.4.2 for details.

than STLN-based method by approximately from 10 to 30 times. On the other hand, in some test cases with the real coefficients, the GPGCD method did not give an answer with sufficiently small amount of perturbations.

Remark 5.1. In this experiment, we compare our implementation designed for problems of two univariate polynomials against theirs designed for multivariate multi-polynomial problems with additional linear coefficient constraints. Kaltofen [23] has reported that they have tested their implementation for real univariate polynomials [25] on an example similar to ours with degree 100 and GCD degree 50, and it took (on a ThinkPad of 1.8 GHz with RAM 1GB) 2 iterations and 9 seconds. This result will give the reader some idea on efficiency of our method.

5.5 SUMMARY

We have proposed an iterative method, based on the modified Newton method which is a generalization of the gradient-projection method, for calculating approximate GCD of univariate polynomials with the real or the complex coefficients.

Our experiments have shown that our algorithm calculates approximate GCD with perturbations as small as those calculated by methods based on the structured total least norm (STLN) method, while our method has shown significantly better performance over the STLN-based methods in its speed, by approximately up to 30 times, which seems to be sufficiently practical for inputs of low or moderate degrees. Furthermore, by other examples, we have shown that our algorithm can properly handle some ill-conditioned problems such as those with GCD containing small or large leading coefficient. However, our experiments have also shown that there are some cases with the real coefficients in which the GPGCD method did not give an answer with sufficiently small amount of perturbations. Factors leading to such phenomena is among our next topics for investigation.

Our result have shown that, in contrast to the STLN-based methods which uses *structure preserving* feature for matrix computations, our simple method can achieve accurate and efficient computation as or more than theirs in calculating approximate GCDs. This suggests that there are some opportunities for improvements of efficiency in calculating approximate GCDs with optimization strategies.

For the future research, the followings are of interest.

- Convergence analysis of the minimizations: from both the theoretical and experimental point of view, it is important and should be investigated thoroughly.
- Improvements on the efficiency: time complexity of our method depends on the minimization, or solving a system of linear equations in each iteration. Thus, analyzing the structure of matrices might improve the efficiency in solving a linear system.
- Comparison with other methods (approaches) for approximate GCD: from various points of view such as accuracy, stability, efficiency, and so on, comparison of our methods with other methods will reveal advantages and drawbacks of our method in more detail.

Other topics, such as generalization of our method to several input polynomials, are also among our next problems.

BIBLIOGRAPHY

[1] O. Aberth. Iteration Methods for Finding all Zeros of a Polynomial Simultaneously. *Math. Comput.*, 27:339–344, 1973. (Cited on page 6.)

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, third edition, 1999. ISBN 0-89871-447-8 (paperback). (Cited on page 39.)

[3] A. C. Bartlett, C. V. Hollot, and Huang Lin. Root locations of an entire polytope of polynomials: it suffices to check the edges. *Math. Control Signals Systems*, 1 (1):61–71, 1988. ISSN 0932-4194. doi: 10.1007/BF02551236. (Cited on page 29.)

[4] B. Beckermann and G. Labahn. A fast and numerically stable Euclidean-like algorithm for detecting relatively prime numerical polynomials. *J. Symbolic Comput.*, 26(6):691–714, 1998. ISSN 0747-7171. Symbolic numeric algebra for polynomials. (Cited on page 67.)

[5] J. Bochnak, M. Coste, and M.-F. Roy. *Real algebraic geometry*, volume 36 of *Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)]*. Springer-Verlag, Berlin, 1998. ISBN 3-540-64663-9. Translated from the 1987 French original, Revised by the authors. (Cited on page 45.)

[6] W. S. Brown and J. F. Traub. On Euclid's Algorithm and the Theory of Subresultants. *J. ACM*, 18(4):505–514, 1971. (Cited on pages 37, 43, and 47.)

[7] P. Chin, R. M. Corless, and G. F. Corliss. Optimization strategies for the approximate GCD problem. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*, pages 228–235 (electronic). ACM, 1998. (Cited on page 67.)

[8] H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1993. (Cited on page 30.)

[9] G. E. Collins. Subresultants and Reduced Polynomial Remainder Sequences. *J. ACM*, 14(1):128–142, 1967. (Cited on page 43.)

[10] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The singular value decomposition for polynomial systems. In *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation*, pages 195–207. ACM, 1995. (Cited on pages 65 and 67.)

[11] R. M. Corless, S. M. Watt, and L. Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Process.*, 52(12):3394–3402, 2004. ISSN 1053-587X. (Cited on page 67.)

[12] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997. ISBN 0-89871-389-7. (Cited on page 78.)

[13] G. M. Diaz-Toca and L. Gonzalez-Vega. Barnett's theorems about the greatest common divisor of several univariate polynomials through Bezout-like matrices. *J. Symbolic Comput.*, 34(1):59–81, 2002. ISSN 0747-7171. (Cited on page 65.)

[14] L. Ducos. Optimizations of the subresultant algorithm. *J. Pure Appl. Algebra*, 145(2):149–163, 2000. (Cited on page 65.)

[15] E. Durand. *Solutions Numériques des Équations Algébriques*, Tome I. Masson, Paris, 1960. (Cited on page 5.)

[16] I. Z. Emiris and B. Mourrain, editors. *Proceedings of the Workshop on Symbolic-Numeric Algebra for Polynomials (SNAP '96)*, INRIA Sophia-Antipolis, France, July 1996. URL `http://www-sop.inria.fr/galaad/conf/1996/snap.html`. (Cited on page 1.)

[17] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate GCDs. *J. Pure Appl. Algebra*, 117/118:229–251, 1997. ISSN 0022-4049. Algorithms for algebra (Eindhoven, 1996). (Cited on pages 65 and 67.)

[18] P. P. Fraigniaud. The Durand-Kerner Polynomials Roots-finding Method in case of Multiple Roots. *BIT*, 31(1):112–123, 1991. ISSN 0006-3835. (Cited on page 12.)

[19] G. H. Golub and C. F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996. (Cited on page 64.)

[20] N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, Philadelphia, Second edition, 2002. ISBN 0-89871-521-0. (Cited on page 39.)

[21] M. Iri. *Numerical Analysis* (in Japanese). Asakura Publishing Co., Tokyo, 1981. (Cited on pages 5, 12, and 28.)

[22] H. Kai and H. Sekigawa, editors. *Proceedings of the 2009 International Workshop on Symbolic-Numeric Computation (SNC '09)*, Kyoto, Japan, August 2009. ACM. ISBN 978-1-60558-664-9. (Cited on page 1.)

[23] E. Kaltofen. Private communication, 2009. (Cited on page 88.)

[24] E. Kaltofen, Z. Yang, and L. Zhi. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In *Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation*, pages 169–176, New York, NY, USA, 2006. ACM. ISBN 1-59593-276-3. doi: 10.1145/1145768.1145799. (Cited on pages 67, 85, and 86.)

[25] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a Sylvester matrix. In D. Wang and L. Zhi, editors, *Symbolic-Numeric Computation*, Trends in Mathematics, pages 69–83. Birkhäuser, 2007. (Cited on pages 67, 83, 84, and 88.)

[26] N. K. Karmarkar and Y. N. Lakshman. On approximate GCDs of univariate polynomials. *J. Symbolic Comput.*, 26(6):653–666, 1998. ISSN 0747-7171. Symbolic numeric algebra for polynomials. (Cited on pages 67, 83, and 84.)

[27] I. O. Kerner. Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen. *Numer. Math.*, 8:290–294, 1966. (Cited on page 5.)

[28] V. L. Kharitonov. The asymptotic stability of the equilibrium state of a family of systems of linear differential equations. *Differentsial′nye Uravneniya*, 14(11): 2086–2088, 2111, 1978. ISSN 0374-0641. (Cited on page 29.)

[29] D. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, third edition, 1998. (Cited on page 43.)

[30] H. Lombardi, M.-F. Roy, and M. S. El Din. New structure theorem for subresultants. *J. Symbolic Comput.*, 29(4–5):663–689, 2000. (Cited on page 65.)

[31] R. Loos. Generalized polynomial remainder sequences. In B. Buchberger, G. E. Collins, and R. Loos, editors, *Computer Algebra: Symbolic and Algebraic Computation*, pages 115–137. Springer-Verlag, Second edition, 1983. (Cited on page 43.)

[32] B. Mishra. *Algorithmic Algebra*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993. (Cited on page 34.)

[33] M. Natori. *Numerical Analysis and its Applications* (in Japanese). Corona Publishing Co., Tokyo, 1990. (Cited on pages 5 and 39.)

[34] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, second edition, 2006. ISBN 978-0387-30303-1; 0-387-30303-0. (Cited on page 72.)

[35] N. Ohsako, H. Sugiura, and T. Torii. A stable extended algorithm for generating polynomial remainder sequence (in Japanese). *Trans. Japan Soc. Indus. Appl. Math*, 7(3):227–255, 1997. (Cited on page 67.)

[36] V. Y. Pan. Computation of approximate polynomial GCDs and an extension. *Inform. and Comput.*, 167(2):71–85, 2001. ISSN 0890-5401. (Cited on page 67.)

[37] L. Pasquini and D. Trigiante. A globally convergent method for simultaneously finding polynomial roots. *Math. Comp.*, 44(169):135–149, 1985. ISSN 0025-5718. (Cited on page 12.)

[38] M. S. Petković, D. D. Herceg, and S. M. Ilić. *Point Estimation Theory and its Applications*. Institute of Mathematics, University of Novi Sad, Novi Sad, 1997. ISBN 86-7031-003-1. (Cited on page 5.)

[39] J. B. Rosen. The gradient projection method for nonlinear programming. II. Nonlinear constraints. *J. Soc. Indust. Appl. Math.*, 9:514–532, 1961. (Cited on pages 67, 72, and 73.)

[40] D. Rupprecht. An algorithm for computing certified approximate GCD of n univariate polynomials. *J. Pure Appl. Algebra*, 139:255–284, 1999. (Cited on page 65.)

[41] M. Sanuki and T. Sasaki. Computing approximate GCDs in ill-conditioned cases. In *SNC '07: Proceedings of the 2007 International Workshop on Symbolic-Numeric Computation*, pages 170–179, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-744-5. doi: 10.1145/1277500.1277525. (Cited on pages 67 and 84.)

[42] T. Sasaki. Approximate algebraic computation (in Japanese). In *Fundamental Theory of Numerical Analysis and Its Vicinity*, number 676 in RIMS Kokyuroku (Collection of Research Reports), pages 307–319. Research Institute for Mathematical Sciences, Kyoto University, Kyoto, Japan, December 1988. (Cited on page 1.)

[43] T. Sasaki. Formula Manipulation System GAL. In Y. Watase and F. Abe, editors, *Computing in High Energy Physics '91* (Tsukuba, 1991), pages 383–389. Universal Academy Press, Tokyo, 1991. ISBN 4-946443-09-6. (Cited on pages 16 and 34.)

[44] T. Sasaki and M-T. Noda. Approximate square-free decomposition and root-finding of ill-conditioned algebraic equations. *J. Inform. Process.*, 12(2):159–168, 1989. (Cited on page 67.)

[45] T. Sasaki and M. Sasaki. Analysis of accuracy decreasing in polynomial remainder sequence with floating-point number coefficients. *J. Inform. Process.*, 12(4):394–403 (1990), 1989. ISSN 0387-6101. (Cited on page 42.)

[46] T. Sasaki and A. Terui. A formula for separating small roots of a polynomial. *ACM SIGSAM Bulletin*, 36(3):19–29, September 2002. (Cited on page 32.)

[47] A. Schönhage. Quasi-gcd computations. *J. Complexity*, 1(1):118–137, 1985. ISSN 0885-064X. (Cited on page 67.)

[48] H. Sekigawa and K. Shirayanagi. On the Location of Zeros of an Interval Polynomial (in Japanese). *The Transactions of the Institute of Electronics, Information and Communication Engineers. A*, J89-A(3):199–216, 2006. (Cited on page 29.)

[49] H. Sekigawa and K. Shirayanagi. On the location of zeros of an interval polynomial. In D. Wang and L. Zhi, editors, *Symbolic-Numeric Computation*, Trends in Mathematics, pages 69–83. Birkhäuser, 2007. (Cited on page 29.)

[50] K. Shirayanagi and H. Sekigawa. An Interval Method Based on Zero Rewriting and Its Application to Sturm's Algorithm (in Japanese). *Transactions of the Institute of Electronics, Information and Communication Engineers A*, J80-A(5):791–802, 1997. (Cited on page 23.)

[51] B. T. Smith. Error bounds for zeros of a polynomial based upon gerschgorin's theorems. *J. ACM*, 17(4):661–674, 1970. (Cited on pages 6 and 23.)

[52] T. Takagi. *Lectures in Algebra* (in Japanese, revised ed.). Kyōritsu Publishing Co., Tokyo, 1965. (Cited on page 6.)

[53] K. Tanabe. A geometric method in nonlinear programming. *J. Optim. Theory Appl.*, 30(2):181–210, 1980. ISSN 0022-3239. (Cited on pages 67, 71, 73, and 74.)

[54] A. Terui. Subresultants in recursive polynomial remainder sequence. In V.G. Ganzha, E.W. Mayr, and E.V. Vorozhtsov, editors, *Proc. The 6th International Workshop on Computer Algebra in Scientific Computing: CASC 2003 (Passau, Germanay, September 20–26, 2003)*, pages 363–375, Garching, Germanay, 2003. Institute für Informatik, Technische Universität München. URL `http://wwwmayr.in.tum.de/cgi-bin/openURL?debug=&genre=article&title=casc&volume=2003&spage=363`. (Cited on page 2.)

[55] A. Terui. Recursive polynomial remainder sequence and the nested subresultants. In V.G. Ganzha, E.W. Mayr, and E.V. Vorozhtsov, editors, *Computer Algebra in Scientific Computing (Proc. CASC 2005)*, volume 3718 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 2005. (Cited on page 2.)

[56] A. Terui. Recursive polynomial remainder sequence and its subresultants. *Journal of Algebra*, 320(2):633–659, July 2008. (Cited on page 2.)

[57] A. Terui. An iterative method for calculating approximate GCD of univariate polynomials. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pages 351–358, New York, NY, USA, 2009. ACM Press. (Cited on page 2.)

[58] A. Terui. GPGCD, an iterative method for calculating approximate GCD of univariate polynomials, with the complex coefficients. In *Proceedings of the Joint Conference of ASCM 2009 and MACIS 2009*, volume 22 of *COE Lecture Note*, pages 212–221. Faculty of Mathematics, Kyushu University, December 2009. (Cited on page 2.)

[59] A. Terui and T. Sasaki. Durand-Kerner method for the real roots. *Japan J. Indus. Appl. Math.*, 19(1):19–38, January 2002. (Cited on page 2.)

[60] A. Terui and T. Sasaki. "Approximate zero-points" of real univariate polynomial with large error terms. *IPSJ J.*, 41(4):974–989, April 2000. (Cited on pages 2 and 32.)

[61] J. von zur Gathen and T. Lücking. Subresultants revisited. *Theoret. Comput. Sci.*, 297(1-3):199–239, 2003. ISSN 0304-3975. Latin American theoretical informatics (Punta del Este, 2000). (Cited on page 44.)

[62] D. Wang and L. Zhi, editors. *Symbolic-Numeric Computation*. Trends in Mathematics. Birkhäuser, 2007. ISBN 978-3-764-37983-4. (Cited on page 1.)

[63] S. M. Watt and J. Verschelde, editors. *Proceedings of the 2007 International Workshop on Symbolic-Numeric Computation (SNC '07)*, London, Ontario, Canada, July 2007. ACM. ISBN 978-1-59593-744-5. (Cited on page 1.)

[64] T. Yamamoto, S. Kanno, and L. Atanassova. Validated Computation of Polynomial Zeros by the Durand-Kerner method. In *Topics in Validated Computations* (Oldenburg, 1993), pages 27–53. North-Holland, Amsterdam, 1994. (Cited on page 5.)

[65] C. J. Zarowski, X. Ma, and F. W. Fairman. QR-factorization method for computing the greatest common divisor of polynomials with inexact coefficients. *IEEE Trans. Signal Process.*, 48(11):3042–3051, 2000. ISSN 1053-587X. (Cited on page 67.)

[66] Z. Zeng. The approximate GCD of inexact polynomials, Part I: a univariate algorithm (extended abstract). preprint, 2004. URL `http://www.neiu.edu/~zzeng/`. 8 pages. (Cited on pages 67 and 80.)

[67] L. Zhi. Displacement structure in computing approximate GCD of univariate polynomials. In *Computer mathematics: Proc. Six Asian Symposium on Computer Mathematics (ASCM 2003)*, volume 10 of *Lecture Notes Ser. Comput.*, pages 288–298. World Sci. Publ., River Edge, NJ, 2003. (Cited on page 67.)