Shirayanagi–Sweedler Algebraic Algorithm Stabilization and Polynomial GCD Algorithms

by

Pramook Khungurn

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006
[June 2007]

© Pramook Khungurn, MMVI. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly
paper and electronic copies of this thesis document in whole or in part.

Author . . . .
Department of Electrical Engineering and Computer Science
January 30, 2007

Certified by . . . .
Alan Edelman
Professor of Applied Mathematics
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

MASSACHUSETTS INSTITUTE
OF TECHNOLOGY

OCT 0 3 2007

LIBRARIES

ARCHIVES

Shirayanagi–Sweedler Algebraic Algorithm Stabilization and Polynomial GCD Algorithms

by

Pramook Khungurn

Submitted to the Department of Electrical Engineering and Computer Science
on January 30, 2007, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Shirayanagi and Sweedler [12] proved that a large class of algorithms on the reals can be modified slightly so that they also work correctly on floating-point numbers. Their main theorem states that, for each input, there exists a precision, called the minimum converging precision (MCP), at and beyond which the modified "stabilized" algorithm follows the same sequence of steps as the original "exact" algorithm.

In this thesis, we study the MCP of two algorithms for finding the greatest common divisor of two univariate polynomials with real coefficients: the Euclidean algorithm, and an algorithm based on QR-factorization. We show that, if the coefficients of the input polynomials are allowed to be any computable numbers, then the MCPs of the two algorithms are not computable, implying that there are no "simple" bounding functions for the MCP of all pairs of real polynomials.

For the Euclidean algorithm, we derive upper bounds on the MCP for pairs of polynomials whose coefficients are members of $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}[\xi]$, and $\mathbb{Q}[\xi]$ where ξ is a real algebraic integer. The bounds are quadratic in the degrees of the input polynomials or worse.

For the QR-factorization algorithm, we derive a bound on the minimal precision at and beyond which the stabilized algorithm gives a polynomial with the same degree as that of the exact GCD, and another bound on the the minimal precision at and beyond which the algorithm gives a polynomial with the same support as that of the exact GCD. The bounds are linear in (1) the degree of the polynomial and (2) the sum of the logarithm of diagonal entries of matrix R in the QR factorization of the Sylvester matrix of the input polynomials.

Thesis Supervisor: Alan Edelman
Title: Professor of Applied Mathematics

Acknowledgments

First, I would like to thank my thesis advisor, Prof. Alan Edelman, for supervising the thesis and allowing for me to use his Beowulf cluster to perform some experimental studies prior to obtaining the theoretical results.

The majority of the work in this thesis was done while I was interning at the NTT Communication Science Laboratories in Japan in the summer of 2005 and 2006. I thank Dr. Kiyoshi Shirayanagi of the NTT Communication Laboratories (now a professor at Tokai University) for introducing me to the field of symbolic-algebraic computation and the problem, for supervising me in the summer of 2005, and guiding and helping through the research. I also would like to thank Dr. Hiroshi Sekigawa for supervising me in the summer of 2006, and Yutaka Kamamoto for helping me adjust to life in Japan and for being a very reliable *sempai.* I appreciate the hospitality I received from researchers at the Human and Information Science group at NTT, and I hope that my relationship with them will continue in the future.

I thank Daniela Reichert, Patricia Gercik, and Saro Darien of the MIT-Japan program for providing me with the opportunity to do research in Japan, which eventually leads to this thesis. I truly enjoyed my summers in Japan, and am deeply grateful for their assistance.

Lastly, I would like to thank my family and my Thai friends at MIT for their support during my time at MIT.

Contents

Chapter 1

Introduction

One of the premises of incorporating approximate computation via floating-point numbers into symbolic computation is to improve efficiency. When inputs with exact values are given and outputs with numerical values are acceptable, replacing the exact values with their floating-point approximations and carrying out the computation using floating-point arithmetic may save both time and space. However, because most existing computer algebra algorithms assume that inputs and arithmetic operations are exact, they cannot cope with numerical errors: even though the algorithm is provided with a sequence of approximations of the input with successively higher precisions, the corresponding sequence of outputs might not converge to the correct output at all. We call an algorithm with this problem *unstable*.

For example, consider a simple program that receives an input x, compute x^{1000}, and compares the result to 2. If they are equal, it outputs 1. If not, it outputs 0. Depending on how the program handles floating point computation, it might be the case that it never outputs 1 although it is given very accurate approximation of $\sqrt[1000]{2}$. Table 1.1 lists approximations of $\sqrt[1000]{2}$ with different precisions and their powers of 1000, computed by Maple. The fact that the powers of 1000 of the inputs are not exactly 2 implies that the program never outputs 1, and that the program is unstable.

Shirayanagi and Sweedler [12] proposed a general technique that transforms many algorithms to stable ones, enabling one to use legacy computer algebra algorithms to perform approximate computations. The technique has been applied to various computer algebra algorithms such as Buchburger's

Precision	$\sqrt[1000]{2}$	$(\sqrt[1000]{2})^{1000}$
10	1.000693387	1.999999075
15	1.00069338746258	1.99999999999874
20	1.0006933874625806325	1.9999999999999999249
25	1.000693387462580632537569	2.000000000000000000000721
30	1.00069338746258063253756863930	1.99999999999999999999999999229

Table 1.1: Values of $\sqrt[1000]{2}$ at different precisions, and their powers of 1000 as computed by Maple.

algorithm for Gröbner bases [11], Sturm's algorithm for counting real roots of polynomials [10], and Greville's algorithm for computing Moore-Penrose generalized inverses [9, 13].

The technique also guarantees that, for every input, there exists an integer which we refer to as the *minimum converging precision* (MCP). The MCP has the property that, if the the number of digits in the approximation of the input is greater than or equal to the MCP, then the stabilized version of the algorithm performs almost the same computation as the original algorithm, yielding a good approximation of the exact output. Since the MCP determines the space and time required to perform floating-point arithmetic operations, algorithms with lower MCP are faster and more accurate. Thus, estimating the MCP is crucial to measuring the effectiveness of stabilized algorithms.

However, there has been no systematic methods to determine the MCP. Neither are we aware of any published work that estimates the MCP of any specific algorithms. As a result, although researchers have been successful in applying the stabilization technique to a number of computer algebra algorithms, the estimation of the MCP of any useful and nontrivial algorithm has remained an open problem [13].

This thesis provides a theoretical study on the MCP of two algorithms for finding the greatest common divisor (GCD) of univariate polynomials with real coefficients: the Euclidean algorithm, and an algorithm based on QR-factorization given in [14]. The Euclidean algorithm is a fast algorithm used as a subroutine of many computer algebra algorithms [2]. On the other hand, the QR-factorization algorithm, although much slower, is numerically stable and has been observed experimentally to work well with coefficients that are not known exactly [3, 14].

We prove that the MCPs of the two algorithms are generally incomputable, and derive bounds for the MCP and some related numbers, thereby settling the open problem in [13] for GCD algorithms.

1.1 Results

Our results are summarized below:

1. A *degree-aware* GCD algorithm is an algorithm whose execution path changes if the degree of the GCD changes. We show that, for any degree-aware GCD algorithm, there exists no Turing machines that computes the MCP of pairs of polynomials whose coefficients are computable numbers. The Euclidean algorithm and the QR-factorization algorithm are both degree-aware, so their MCPs are in general incomputable. This fact implies that there exists no "simple" bounding function for the MCP of these algorithms.

2. For the Euclidean algorithm, we derive an upper bound for the MCP of a simple variant of the Euclidean algorithm. The bound is $\widetilde{O}(d(\log M - \log m))$ where d is the degree of the input polynomial with bigger degree, and M and m are the largest and the smallest nonzero quantities that arise during the evolution of the Euclidean algorithm, respectively.

We also found that it is possible to simplify the bound when coefficients of the input polynomials are from domains where exact computation can be carried out. To this end, we derive simpler bounds for the MCP in cases where the coefficients are from the sets $\mathbb{Z}[\xi]$, and $\mathbb{Q}[\xi]$, where ξ is a real algebraic integer not equal to 1. The bounds are $\widetilde{O}(d^2 \log N)$ and $\widetilde{O}(d^3 \log N)$, respectively, with N being the largest integer in the coefficients. (For example, the largest integer in the algebraic number $\frac{1}{5} + \frac{2}{7}\xi + \frac{3}{4}\xi^2$ is 7.)

3. For the QR-factorization algorithm, we could not derive bounds on the MCP. However, we managed to derive bounds on two related numbers: the *minimal correct degree precision* (MCDP) and the *minimal same support precision* (MSSP). The MCDP is the minimal precision at and beyond which the stabilized algorithm gives a polynomial with the same degree as that of the exact GCD. The MSSP is the minimal precision at and beyond which the algorithm gives a polynomial with the same support, i.e. nonzero terms, as that of the exact GCD.

   For the MCDP, we show that it is of order $O\big(d(\log d + \log M) - \sum_{i=1}^{r} \log |R[i,i]|\big)$, where d is the sum of the degrees of the input polynomials, M is the absolute value of the largest coefficient, r is the rank of the Sylvester matrix of the input polynomials, and R is the upper triangular matrix of the QR factorization of the Sylvester matrix. For the MSSP, we show that it is of order $O\big(d(\log d + \log M) - \sum_{i=1}^{r} \log |R[i,i]|\big) - \log |\mu|\big)$, where μ is the smallest coefficient of the (monic) GCD of the input polynomials.

1.2 Thesis Organization

Chapter 2 and Chapter 3 give background material of the thesis. Chapter 2 defines notations and states preliminary facts that we use to derive our main results. Chapter 3 describes our model of computation and discusses the Shirayanagi–Sweedler stabilization technique.

The main results are covered in the next 3 chapters. Chapter 4 proves that the MCP of any degree-aware is not computable. Chapter 5 derives upper bounds on the MCP of the Euclidean algorithm. Chapter 6 derives upper bounds on the MCDP and the MSSP of the QR-factorization algorithm.

Proofs of many results rely heavily on lower bounds on sizes of nonzero algebraic numbers of certain forms, and upper bounds on errors generated by primitive algebraic operations. Due to their technical nature, the proofs for these bounds are given in the appendices. Appendix A deals with the former lower bounds, and Appendix B the latter upper bounds.

Chapter 2

Notations and Preliminaries

In this chapter, we define notations and conventions that will be used throughout the thesis. Section 2.1 defines floating-point numbers. Section 2.2 defines bracket coefficients, the workhorse of the Shirayanagi–Sweedler stabilization technique. Section 2.3 defines vector and matrix notations. Section 2.4 defines notations for sets of algebraic numbers whose "largest integers" are given as parameters. Lastly, Section 2.5 define polynomial notations.

2.1 Floating-point Numbers

A floating-point number a with precision τ is an ordered pair $(\mathfrak{M}(a), \mathfrak{e}(a))$, where $\mathfrak{M}(a)$ is an integer with τ digits in base 2, and $\mathfrak{e}(a)$ is another integer. The real value of a is defined to be

$$a = \mathfrak{M}(a) \times 2^{\mathfrak{e}(a)-\tau}.$$

We require that, if $a \neq 0$, then $|\mathfrak{M}(a)| \times 2^{-\tau} \geq 0.5$. That is, the leftmost digit of $\mathfrak{M}(a)$ is 1 unless $a = 0$. Note that our definition of floating-point numbers is very close to the IEEE floating-point standard, except for the absence of the sign bit, and the requirement that the leftmost bit of the mantissa is not zero.

Unless stated otherwise, τ denotes the precision of every floating-point number we discuss.

Let x be a real number. We define the following functions:

- Let $\mathrm{fl}_\tau(x)$ be the floating-point number with precision τ closest to x.
- Let $\mathrm{up}_\tau(x)$ be x rounded up to precision τ away from zero.
- For $x \neq 0$, let $\mathfrak{e}(x)$ be the integer such that $x = y \times 2^{\mathfrak{e}(x)}$ for some y such that $0.5 \leq |y| < 1$. Note that this definition is consistent with the $\mathfrak{e}(a)$ defined above.
- Let $\epsilon_\tau(x)$ be the value $2^{\mathfrak{e}(x)-1-\tau}$.

Proposition 2.1. *For all $x \neq 0$, the four functions above satisfy the following inequalities:*

$$2^{\mathfrak{e}(x)-1} \leq |x|, \tag{2.1}$$

$$2^{-\tau-1}|x| \leq \epsilon_\tau(x) \leq 2^{-\tau}|x|, \tag{2.2}$$

$$\mathrm{fl}_\tau(x) - \epsilon_\tau(x) \leq x \leq \mathrm{fl}_\tau(x) + \epsilon_\tau(x), \tag{2.3}$$

$$\mathrm{fl}_\tau(x) - \epsilon_\tau(\mathrm{fl}_\tau(x)) \leq x \leq \mathrm{fl}_\tau(x) + \epsilon_\tau(\mathrm{fl}_\tau(x)), \tag{2.4}$$

$$|\mathrm{fl}_\tau(x)| \leq (1 + 2^{-\tau})|x|, \tag{2.5}$$

$$|x| \leq |\mathrm{up}_\tau(x)| \leq (1 + 2^{-\tau+1})|x|. \tag{2.6}$$

Proof. Let us assume first that x is positive. Expanding x in binary, we may write

$$x = 0.1b_2b_3 \cdots b_{\tau-1}b_\tau b_{\tau+1} \cdots \times 2^k,$$

where all the b_i's are members of $\{0, 1\}$, and $k \in \mathbb{Z}$. Clearly, $\mathfrak{e}(x) = k$, and $\epsilon_\tau(x) = 2^{k-\tau-1}$. Thus, $2^{\mathfrak{e}(x)-1} = 0.1 \times 2^k \leq |x|$. Also, since $2^{k-1} \leq |x| \leq 2^k$, we have that

$$2^{-\tau-1}|x| \leq 2^{-\tau-1} \cdot 2^k = \epsilon_\tau(x) = 2^{-\tau} \cdot 2^{k-1} \leq 2^{-\tau}|x|.$$

We have proven (2.1) and (2.2).

Consider $\mathrm{fl}_\tau(x)$, there are two cases. In the first case, $b_{\tau+1} = 0$. We have that x is rounded down and $\mathrm{fl}_\tau(x) = 0.1b_2b_3 \cdots b_{\tau-1}b_\tau \times 2^k$. Here, $\mathrm{fl}_\tau(x)$ is less than or equal to x, and $\epsilon_\tau(x)$ and $\epsilon_\tau(\mathrm{fl}_\tau(x))$ are both equal to $2^{k-\tau-1}$. Hence, $\mathrm{fl}_\tau(x) - \epsilon_\tau(x)$ and $\mathrm{fl}_\tau(x) - \epsilon_\tau(\mathrm{fl}_\tau(x))$ are both less than equal to x, and obviously $|\mathrm{fl}_\tau(x)| \leq |x| < (1 + 2^{-\tau})|x|$. On the other hand,

$$\mathrm{fl}_\tau(x) + \epsilon_\tau(x) = \mathrm{fl}_\tau(x) + \epsilon_\tau(\mathrm{fl}_\tau(x)) = 0.1b_2 \cdots b_\tau 1 \times 2^k > 0.1b_2 \cdots b_\tau 0 b_{\tau+2}b_{\tau+3} \cdots \times 2^k = x,$$

and we have proven (2.3), (2.4), and (2.5) for this case.

For the second case, $b_{\tau+1} = 1$. We have that x is rounded up and

$$\mathrm{fl}_\tau(x) = (0.1b_2b_3 \cdots b_{\tau-1}b_\tau + 2^{-\tau-1}) \times 2^k = (0.1b_2b_3 \cdots b_{\tau-1}b_\tau b_{\tau+1}) \times 2^k + \epsilon_\tau(x).$$

Since $0.1b_2b_3 \cdots b_{\tau-1}b_\tau b_{\tau+1} \times 2^k \leq |x|$, and $\epsilon_\tau(x) \leq 2^{-\tau}|x|$, we have that $|\mathrm{fl}_\tau(x)| \leq (1 + 2^{-\tau})|x|$. So, (2.5) is true for this case as well. For (2.3), we have that

$$\mathrm{fl}_\tau(x) - \epsilon_\tau(x) = 0.1b_2 \cdots b_\tau b_{\tau+1} \times 2^k \leq x < \mathrm{fl}_\tau(x) < \mathrm{fl}_\tau(x) + \epsilon_\tau(x).$$

For (2.4), if $\epsilon_\tau(\mathrm{fl}_\tau(x)) = \epsilon_\tau(x)$, then the situation is the same as that in the proof of (2.3). The only case where $\epsilon_\tau(\mathrm{fl}_\tau(x)) \neq \epsilon_\tau(x)$ is when $b_2 = b_3 = \cdots = b_{\tau+1} = 1$. In this case, $\mathrm{fl}_\tau(x) = 2^k$ and $\epsilon_\tau(x) = 2^{k-\tau}$. Still, we have that

$$\mathrm{fl}_\tau(x) - \epsilon_\tau(\mathrm{fl}_\tau(x)) = 0.\underbrace{11\cdots1}_{\tau}\times 2^k < x \leq \mathrm{fl}_\tau(x) < \mathrm{fl}_\tau(x) + \epsilon_\tau(\mathrm{fl}_\tau(x)),$$

and so (2.4) is true in this case as well.

Consider $\mathrm{up}_\tau(x)$. There are also two cases. In the first case, all of $b_{\tau+1}$, $b_{\tau+2}$, $\ldots$ are zeroes. Thus, we have that $\mathrm{up}_\tau(x) = x$, and (2.6) holds trivially. In the second case, not all of $b_{\tau+1}$, $b_{\tau+2}$, $\ldots$ are zeroes. We have that $\mathrm{up}_\tau(x) = (0.1b_2\cdots b_\tau + 2^{-\tau}) \times 2^k \geq x$. Since $0.1b_2\cdots b_\tau < x$ and $2^{-\tau}\cdot 2^k = 2\epsilon_\tau(x) \leq 2^{-\tau+1}x$, we have that $|x| \leq \mathrm{up}_\tau(x) \leq (1+2^{-\tau+1})|x|$ as required.

The case when x is negative can be argued with the same argument as above. □

2.2 Bracket Coefficients

The Shirayanagi–Sweedler stabilization technique relies on replacing real numbers with intervals whose endpoints are floating-point numbers. Following the convention in [12], we represent intervals by objects called bracket coefficients.

A *bracket coefficient* $[\![x]\!]$ is an ordered pair of floating-point numbers $(\langle x\rangle, \lfloor x\rceil)$. We call $\langle x\rangle$ the *approximate value* of $[\![x]\!]$, and $\lfloor x\rceil$ the *error value* or the *error term* of $[\![x]\!]$. The bracket coefficient $[\![x]\!]$ is said to *approximates* a real number x if $\langle x\rangle - \lfloor x\rceil \leq x \leq \langle x\rangle + \lfloor x\rceil$. We denote that fact that $[\![x]\!]$ approximates x by the symbol $[\![x]\!] \approxeq x$. Also, as a convention, $[\![x]\!]$ denotes a bracket coefficient that approximates the real number x. The *floating-point bracket approximation* to a real number x, denoted by $[\![x]\!]_\tau$, is the bracket coefficient $(\mathrm{fl}_\tau(x), \epsilon_\tau(\mathrm{fl}_\tau(x)))$.

Bracket coefficient $[\![x]\!]$ is said to be *equivalent* to zero if $|\langle x\rangle| \leq \lfloor x\rceil$, or, in other words, if it approximates zero. $[\![x]\!]$ is said to be *greater than* zero if it is not equivalent to zero, and $\langle x\rangle > 0$. Similarly, $[\![x]\!]$ is said to be *less than* zero if it is not equivalent to zero, and $\langle x\rangle < 0$. This trichotomy law of bracket coefficients is based on the zero-rewriting, an idea crucial to the Shirayanagi-Sweedler stabilization technique.

Arithmetic operations on bracket coefficients resemble those of the so-called "interval arithmetic" studied in [1]. The operations are defined as follows:

1. Bracket coefficient $[\![s]\!]$ is said to be the *sum* of $[\![a]\!]$ and $[\![b]\!]$, denoted by $[\![s]\!] = [\![a]\!] + [\![b]\!]$, if

   (a) $\langle s\rangle = \mathrm{fl}_\tau(\langle a\rangle + \langle b\rangle)$, and

   (b) $\lfloor s\rceil = \mathrm{up}_\tau(\lfloor a\rceil + \lfloor b\rceil + \epsilon_\tau(\langle s\rangle))$.

2. Bracket coefficient $[\![d]\!]$ is said to be the *difference* of $[\![a]\!]$ and $[\![b]\!]$, denoted by $[\![d]\!] = [\![a]\!] - [\![b]\!]$, if

   (a) $\langle d\rangle = \mathrm{fl}_\tau(\langle a\rangle - \langle b\rangle)$, and

(b) $\lfloor d \rceil = \mathrm{up}_\tau(\lfloor a \rceil + \lfloor b \rceil + \epsilon_\tau(\langle d \rangle))$.

3. Bracket coefficient $[\![p]\!]$ is said to be the *product* of $[\![a]\!]$ and $[\![b]\!]$, denoted by $[\![p]\!] = [\![a]\!] \times [\![b]\!]$ or by just $[\![p]\!] = [\![a]\!][\![b]\!]$, if

   (a) $\langle p \rangle = \mathrm{fl}_\tau(\langle a \rangle \langle b \rangle)$, and

   (b) $\lfloor p \rceil = \mathrm{up}_\tau(\lfloor a \rceil \lfloor b \rceil + |\langle a \rangle| \lfloor b \rceil + |\langle b \rangle| \lfloor a \rceil + \epsilon_\tau(\langle p \rangle))$.

4. For $[\![a]\!]$ not equivalent to zero, bracket coefficient $[\![i]\!]$ is said to be the *inverse* of $[\![a]\!]$, denoted by $[\![i]\!] = [\![a]\!]^{-1}$ or by $[\![i]\!] = 1/[\![a]\!]$, if

   (a) $\langle i \rangle = \mathrm{fl}_\tau(1/\langle a \rangle)$, and

   (b) $\lfloor i \rceil = \mathrm{up}_\tau\left(\frac{\lfloor a \rceil}{|\langle a \rangle|(|\langle a \rangle| - \lfloor a \rceil)} + \epsilon_\tau(\langle i \rangle)\right)$.

5. For $[\![a]\!]$ not equivalent to zero such that $\langle a \rangle > 0$, bracket coefficient $[\![r]\!]$ is said to be the *positive square root* of $[\![a]\!]$, denoted by $[\![r]\!] = [\![a]\!]^{1/2}$, if

   (a) $\langle r \rangle = \mathrm{fl}_\tau(\langle a \rangle^{1/2})$, and

   (b) $\lfloor r \rceil = \mathrm{up}_\tau\left(\frac{\langle a \rangle^{1/2}}{2}\left(\frac{\lfloor a \rceil}{\langle a \rangle - \lfloor a \rceil}\right) + \epsilon_\tau(\langle r \rangle)\right)$.

The following preposition states that bracket coefficient operations yield bracket coefficients that approximates the exact result. Thus, in a way, the set of bracket coefficients along with the above operations is compatible with the field of real numbers.

Proposition 2.2. *The following statements are true for all $[\![a]\!]$ and $[\![b]\!]$.*

1. $[\![a]\!] + [\![b]\!] \approxeq a + b$.
2. $[\![a]\!] - [\![b]\!] \approxeq a - b$.
3. $[\![a]\!][\![b]\!] \approxeq ab$.
4. *If $[\![a]\!]$ is not equivalent to zero, then $1/[\![a]\!] \approxeq 1/a$.*
5. *If $[\![a]\!]$ is not equivalent to zero, and $\langle a \rangle > 0$, then $[\![a]\!]^{1/2} \approxeq \sqrt{a}$.*

The proof of the proposition is trivial, thus omitted.

2.3 Vectors and Matrices

Vectors are denoted by boldfaced lowercase letters, and matrices by uppercase letters. For example, $\mathbf{a}$ is a vector, and A is a matrix. We let $\mathbf{a}[k]$ denote the kth component of vector $\mathbf{a}$, and $A[i, j]$ denote the (i, j)-element of matrix A. Moreover, for integers i, j such that $i \leq j$, we let $\mathbf{a}[i \cdots j]$ denote the vector $(\mathbf{a}[i], \mathbf{a}[i+1], \ldots, \mathbf{a}[j])$. Also, for any integers i_1, i_2, j_1, j_2 such that $i_1 \leq i_2$ and $j_1 \leq j_2$, we let $A[i_1 \cdots i_2, j_1 \cdots j_2]$ denote the matrix

$$\begin{bmatrix} A[i_1, j_1] & A[i_1+1, j_1] & A[i_1+2, j_1] & \cdots & A[i_2, j_1] \\ A[i_1, j_1+1] & A[i_1+1, j_1+1] & A[i_1+2, j_1+1] & \cdots & A[i_2, j_1+1] \\ A[i_1, j_1+2] & A[i_1+1, j_1+2] & A[i_1+2, j_1+2] & \cdots & A[i_2, j_1+2] \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A[i_1, j_2] & A[i_1+1, j_2] & A[i_1+2, j_2] & \cdots & A[i_2, j_2] \end{bmatrix}.$$

Also, let $A[i_1 \cdots i_2, j] = A[i_1 \cdots i_2, j \cdots j]$ (a row vector), and $A[i, j_1 \cdots j_2] = A[i \cdots i, j_1 \cdots j_2]$, (a column vector). Moreover, let $A[*, j]$ denote the jth column of A, and $A[i, *]$ denote the ith row of A.

Let $\|\mathbf{a}\|_1$ and $\|\mathbf{a}\|$ denote the 1-norm and the 2-norm of $\mathbf{a}$, respectively. Similarly, $\|A\|_1$ and $\|A\|$ denote the 1-norm and the 2-norm,of matrix A, respectively.

If $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, then let $|\mathbf{a}|$ denote the vector $(|a_1|, |a_2|, \ldots, |a_n|)$, the vector whose entries are the absolute values of the corresponding entries of $\mathbf{a}$. We define $|A|$ similarly. This notation should not to be confused with $\|\mathbf{a}\|$ or $\|A\|$.

A vector of bracket coefficients $(\llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket, \ldots, \llbracket b_n \rrbracket)$ is said to *approximate* $\mathbf{a}$ if $\llbracket b_i \rrbracket \approxeq a_i$ for all i such that $1 \leq i \leq n$, and we denote this fact by the symbol $(\llbracket b_1 \rrbracket, \llbracket b_2 \rrbracket, \ldots, \llbracket b_n \rrbracket) \approxeq \mathbf{a}$. As a convention, $\llbracket \mathbf{a} \rrbracket$ denote the vector of bracket coefficients $(\llbracket a_1 \rrbracket, \llbracket a_2 \rrbracket, \ldots, \llbracket a_n \rrbracket)$ that approximates $\mathbf{a}$. Also, for convenience, let $\langle \mathbf{a} \rangle$ denote $(\langle a_1 \rangle, \langle a_2 \rangle, \ldots, \langle a_n \rangle)$, and $\lfloor \mathbf{a} \rceil$ denote $(\lfloor a_1 \rceil, \lfloor a_2 \rceil, \ldots, \lfloor a_n \rceil)$. These notations can be extended to matrices in a straightforward manner.

Moreover, we let $\mathrm{fl}_\tau(\mathbf{a})$ denote the vector $(\mathrm{fl}_\tau(a_1), \mathrm{fl}_\tau(a_2), \ldots, \mathrm{fl}_\tau(a_n))$, and $\epsilon_\tau(\mathbf{a})$ denote the vector $(\epsilon_\tau(a_1), \epsilon_\tau(a_2), \ldots, \epsilon_\tau(a_n))$. The *floating-point bracket approximation* to vector $\mathbf{a}$ is the vector of bracket coefficients $\llbracket \mathbf{a} \rrbracket_\tau = (\llbracket a_1 \rrbracket_\tau, \llbracket a_2 \rrbracket_\tau, \ldots, \llbracket a_n \rrbracket_\tau)$. Again, these notations can be extended to matrices in a straightforward way.

We will make extensive uses of the following propositions in this thesis:

Proposition 2.3 (Hadamard's inequality). *If A is an $n \times n$ complex matrix, then*

$$|\det(A)| \leq \prod_{j=1}^{n} \|A[*, j]\|.$$

Proposition 2.4 (Cayley–Hamilton Theorem). *If A is an $n \times n$ complex matrix and p is the characteristic polynomial of A, then $p(A) = 0$.*

2.4 Some Sets of Algebraic Numbers

Let N be a positive real number, and let ξ be a real algebraic integer of degree n. We define the following sets:

- $\langle\langle N \rangle\rangle = \{n \in \mathbb{Z} : |n| \leq N\}$.
- $\mathbb{Q}\langle\langle N \rangle\rangle = \{p/q : p, q \in \langle\langle N \rangle\rangle,\ q \neq 0\}$.
- $\mathbb{Z}[\xi]\langle\langle N \rangle\rangle = \{a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1} : a_i \in \langle\langle N \rangle\rangle,\ 0 \leq i \leq n-1\}$
- $\mathbb{Q}[\xi]\langle\langle N \rangle\rangle = \{a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1} : a_i \in \mathbb{Q}\langle\langle N \rangle\rangle,\ 0 \leq i \leq n-1\}$

For examples, $\langle\langle 2 \rangle\rangle = \{-2, -1, 0, 1, 2\}$, and $\mathbb{Z}[\sqrt{2}]\langle\langle 1 \rangle\rangle = \{0, \pm 1, \pm\sqrt{2}, \pm 1 \pm \sqrt{2}, \}$. In effect, the parameter N is the largest integer that can appear in the canonical representation of elements of the above sets.

The above notations will be used extensively in Chapter 5 and Appendix A.

2.5 Polynomials

Throughout this thesis, every polynomial is a real polynomial, and x is the indeterminate variable of every polynomial. Polynomials are denoted by boldfaced uppercase letters: for examples, $\mathbf{P}$, $\mathbf{Q}$, and $\mathbf{R}$. The degree of polynomial $\mathbf{P}$ is denoted by $\eth(\mathbf{P})$. As a convention, the zero polynomial has degree $-\infty$.

The coefficient of x^r in $\mathbf{P}$ is denoted by $\mathbf{P}[r]$. When $\eth$ appears inside the bracket, it refers to the degree of the polynomial. So, the leading coefficient of $\mathbf{P}$ is denoted by $\mathbf{P}[\eth]$, and, if $\eth(\mathbf{P}) \geq r$, then $\mathbf{P}[\eth - r]$ is the coefficient of $x^{\eth(\mathbf{P})-r}$. As another convention, the leading coefficient of the zero polynomial is 0. A polynomial is said to be *monic* if its leading coefficient is 1.

The *support* of polynomial $\mathbf{P}$ is the set $\text{supp}(\mathbf{P}) = \{r : \mathbf{P}[r] \neq 0\}$. For example, the support of $x^5 + x + 1$ is the set $\{0, 1, 5\}$, and the support of the zero polynomial is the empty set.

For any integers a and b such that $0 \leq a \leq b$, we let $\mathbf{P}[a : b]$ be the polynomial $\sum_{r=a}^{b} x^r \mathbf{P}[r]$. For examples, if $\mathbf{P} = x^4 + 4x^3 + 6x^2 + 4x + 1$, then $\mathbf{P}[1 : 3] = 4x^3 + 6x^2 + 4x$, and $\mathbf{P}[0 : 1] = 4x + 1$. Our use of $\eth$ above carries over to here as well. For examples, $\mathbf{P}[\eth : \eth]$ is the leading monomial, and $\mathbf{P}[\eth - 2 : \eth]$ is the sum of three terms with the highest degrees. Also, if $a > b$, we define $\mathbf{P}[a : b]$ to be 0.

Two polynomials $\mathbf{A}$ and $\mathbf{B}$ are said to be *similar* if there exists $c \in \mathbb{R}$ such that $c \neq 0$ and $\mathbf{A} = c\mathbf{B}$. We denote the fact that $\mathbf{A}$ and $\mathbf{B}$ by symbols $\mathbf{A} \sim \mathbf{B}$.

The *quotient* of dividing $\mathbf{A}$ with $\mathbf{B}$ is denoted by $\text{quo}(\mathbf{A}, \mathbf{B})$, and the *remainder* denoted by $\text{rem}(\mathbf{A}, \mathbf{B})$. The *greatest common divisor* (GCD) of polynomials $\mathbf{A}$ and $\mathbf{B}$, denoted by $\text{gcd}(\mathbf{A}, \mathbf{B})$.

As a convention, the GCD is always a *monic* polynomial. However, the two algorithms we study will output a polynomial similar to the GCD, not the GCD itself.

Chapter 3

Shirayanagi–Sweedler Stabilization Technique

In this chapter, we give backgrounds on the Shirayanagi–Sweedler stabilization technique. Section 3.1 defines the model of computation, algebraic algorithm, and the notion of execution path. Section 3.2 specifies how polynomials are represented in our model of computation. Section 3.3 defines the stability of algebraic algorithm. Section 3.4 states the main result of Shirayanagi's and Sweedler's paper, formally defines the MCP, MCDP, and MSSP, and gives an overview on how the stabilization technique works.

3.1 Model of Computation and Algebraic Algorithms

We are interested in two types of machines: EXACT, and FLOAT_τ. Both machines have a countably infinite number of registers $x_1, x_2, \ldots$. Each register of EXACT holds a real number, and each register of FLOAT_τ holds a floating-point number with precision τ. Both machines can perform addition, subtraction, multiplication, and division on the contents of any two registers, and then store the result in another register. Moreover, it can test whether the content of a register is equal to zero or whether it is less than or greater than zero. The arithmetic operations in EXACT are the canonical operations on real numbers, and the arithmetic operations in FLOAT_τ result in the nearest floating-point number with precision τ to the exact result. For example, adding x_i and x_j in FLOAT_τ would result in $\mathrm{fl}_\tau(x_i + x_j)$.

An *algebraic algorithm* is a finite sequence of the following instructions:

1. $x_i \leftarrow x_j + x_k$

   Add the content of x_j and x_k and store the result in x_i.

2. $x_i \leftarrow x_j - x_k$

   Compute the difference of x_j and x_k and store the result in x_i.

3. $x_i \leftarrow x_j \times x_k$

   Multiply the content of x_j and x_k and store the result in x_i.

4. $x_i \leftarrow x_j^{-1}$

   Compute the multiplicative inverse of x_j and store the result in x_i. If the content of x_j is zero, the algorithm halts immediately.

5. $x_i \leftarrow \sqrt{x_j}$

   Compute the positive square root of x_j and stores the result in x_i. If the content of x_j is negative, the algorithm halts immediately.

6. $x_i \leftarrow C$

   Store the an integer value C in x_i.

7. **goto line ℓ**

   Jump to line ℓ of the algorithm.

8. **if $x_i = 0$ goto line ℓ**

   Test if the content of x_i is zero. If so, jump to line ℓ. Otherwise, continue on to next line.

9. **if $x_i < 0$ goto line ℓ**

   Test if the content of x_i is less than 0. Jump to the appropriate line based on the result of the comparison.

10. **halt**

    Terminate the algorithm.

For example, the algorithm in Figure 3-1 checks whether the content of x_1 is equal to that of x_2. If so, it sets x_3 to 1. Otherwise, it sets x_3 to 0. We claim that most algorithms in computer algebra can be written with the above instructions, and therefore are algebraic algorithms. Note also that any algebraic algorithm can run on both EXACT and FLOAT_τ although their outputs might not agree.

We can regard the state of the vector $\mathbf{x} = (x_1, x_2, \dots)$ before the algorithm runs as the input of the algorithm, and the state of $\mathbf{x}$ after the algorithm terminates as the output of the algorithm. That is, an algebraic algorithm is basically a function on an infinite dimensional Euclidean space.

An *execution path* of an algebraic algorithm is the sequence of instructions the algorithm follows when running on a particular input. For example, if $x_1 = x_2$, then the execution path of the algorithm in Figure 3-1 is:

```
1  $x_1 \leftarrow x_1 - x_2$
2  **if** $x_1 = 0$ **goto line** 5
3  $x_3 \leftarrow 0$
4  **goto line** 6
5  $x_3 \leftarrow 1$
6  **halt**
```

Figure 3-1: A simple algebraic algorithm that compares the content of register x_1 and x_2, and writes the result of the comparison into x_3 in binary.

```
1  $x_1 \leftarrow x_1 - x_2$
2  **if** $x_1 = 0$ **goto line** 5
5  $x_3 \leftarrow 1$
6  **halt**
```

But if $x_1 \neq x_2$, the execution path is:

```
1  $x_1 \leftarrow x_1 - x_2$
2  **if** $x_1 = 0$ **goto line** 4
3  $x_3 \leftarrow 0$
4  **goto line** 6
6  **halt**
```

3.2 Polynomial Representation and Operations

As in most computer algebra systems, we represent a polynomial of degree at most d by an array of $d + 1$ real numbers: one register is reserved for each coefficient.

The degree of a polynomial represented in this way is determined by looping through the array to find the non-zero term with the highest degree which is not zero. The pseudocode for the procedure to do so is given in Figure 3-2. Note that $-\infty$ in the pseudocode is just a symbol which may be represented by any constant that is not a nonnegative integer such as -1 or 2.5.

Two polynomial operations that are relevant to this thesis are (1) polynomial addition, and (2) multiplication of a polynomial by a monomial. The sum of two polynomials are calculated by adding coefficients of terms with the same degree together, and the product of a polynomial with a monomial is calculated by multiplying all coefficients of the polynomial with the coefficient of the monomial, and then shifting the terms according to the exponent of the monomial. We assume implicitly that there are procedures that handle these operations, but we will express them with

```
DEGREE($\mathbf{A}$)
    ▷ Input: $\mathbf{A}$ such that $\partial(\mathbf{A}) \leq d$.
    ▷ Output: $\partial(\mathbf{A})$
    $\ell \leftarrow d$
    while $\mathbf{A}[\ell] = 0$ and $\ell \geq 0$
         do $\ell \leftarrow \ell - 1$.
    if $\ell < 0$
       then return $-\infty$
       else return $\ell$
```

Figure 3-2: The procedure to determine the degree of a polynomial.

arithmetic operators in subsequent pseudocodes.

3.3 Stability of Algebraic Algorithms

Shirayanagi and Sweedler introduced the notion of stability of algebraic algorithms in [12]. A stable algebraic algorithm is similar to a continuous function.

Let $\mathcal{A}$ be an algebraic algorithm, and let $\mathbf{x}$ be its input. For integer $\tau > 0$, let $\mathcal{A}_\tau(\mathrm{fl}_\tau(\mathbf{x}))$ to be the output of $\mathcal{A}$ running on FLOAT_τ with input $\mathrm{fl}_\tau(\mathbf{x})$, and let $\mathcal{A}(\mathbf{x})$ be the output of $\mathcal{A}$ running on EXACT with input $\mathbf{x}$. We refer to the execution of $\mathcal{A}$ on EXACT as the *exact computation*, $\mathbf{x}$ as the *exact input*, and $\mathcal{A}(\mathbf{x})$ as the *exact output*.

An algebraic algorithm $\mathcal{A}$ is said *stable at point* $\mathbf{x}$ if

$$\lim_{\tau \to \infty} \mathcal{A}_\tau(\mathrm{fl}_\tau(\mathbf{x})) = \mathcal{A}(\mathbf{x}).$$

We say that $\mathcal{A}$ is *stable* if it is stable at all points in its domain. For example, any algebraic algorithms with no conditional statements are stable because it only performs operations that preserve continuity. (All arithmetic operations allowed in our model of computations are continuous functions on real vector spaces.) However, the algorithm in Figure 3-3 is not stable at point $(\sqrt[3]{2}, 0, 0, \ldots)$ because numerical approximations of $\sqrt[3]{2}$ do not always yield 2 exactly when cubed. Thus, it may not output 1 at all no matter how precise the approximation of $\sqrt[3]{2}$ in x_1 is.

3.4 Shirayanagi–Sweedler Stabilization Technique

In [11], Shirayanagi proposed an algorithm to compute a Gröbner basis of polynomials whose coefficients are approximated by floating-point numbers. He proved that, as the precision of the floating-point numbers increases, the basis obtained from the algorithm converges to that obtained from the exact computation. This property gives the algorithm an advantage over algorithms which

```
1 $x_2 \leftarrow x_1 \times x_1$
2 $x_2 \leftarrow x_2 \times x_1$
3 $x_3 \leftarrow 2$
4 $x_2 \leftarrow x_2 - x_3$
5 if $x_2 = 0$ goto line 8
6 $x_4 \leftarrow 0$
7 goto line 9
8 $x_4 \leftarrow 1$
9 halt
```

Figure 3-3: An algebraic algorithm that computes the cube of the value stored in x_1 and checks whether that value is 2.

assume that inputs and arithmetics operations are exact because such algorithms are generally not stable.

Later, Shirayanagi and Sweedler [12] generalized Shirayanagi's approach and proposed a scheme that transforms any algebraic algorithm to a stable one. Their main theorem can be restated as follows:

Theorem 3.1. *Let $\mathcal{A}$ be an algebraic algorithm. Then, there exists an algebraic algorithm $\mathcal{A}'$ such that*

1. *$\mathcal{A}'$ is stable at every point in $\mathcal{A}$'s domain.*
2. $\lim_{\tau \to \infty} \mathcal{A}'_\tau(\mathrm{fl}_\tau(\mathbf{x})) = \mathcal{A}(\mathbf{x})$ *for every point* $\mathbf{x}$ *in $\mathcal{A}$'s domain.*
3. *For every point* $\mathbf{x}$ *in $\mathcal{A}$'s domain, there exists a positive integer $\Gamma_\mathbf{x}$ such that, for all $\tau \geq \Gamma_\mathbf{x}$, the execution path of $\mathcal{A}'$ on* FLOAT_τ *with input* $\mathrm{fl}_\tau(\mathbf{x})$ *is the same as the execution path of $\mathcal{A}$ on* EXACT *with input* $\mathbf{x}$.

We say that $\Gamma_\mathbf{x}$ is a *converging precision* of $\mathcal{A}$ on $\mathbf{x}$. The smallest such $\Gamma_\mathbf{x}$ is called the *minimum converging precision* (MCP). The theorem implies that, if the precision of the input is higher than the MCP, then the "stabilized version" of the algorithm performs almost the same sequence of arithmetic operations as the original algorithm, yielding a good approximation to the exact output.

In the context of algorithm $\mathcal{A}$ that outputs a polynomial, Shirayanagi and Sweedler also shows that, for every input $\mathbf{x}$, there exists an integer $\Gamma^{SS}_\mathbf{x}$ such that, if $\tau \geq \Gamma^{SS}_\mathbf{x}$, then $\mathcal{A}'_\tau(\mathrm{fl}_\tau(\mathbf{x}))$ has the same support as that of $\mathcal{A}(\mathbf{x})$. We call the least $\Gamma^{SS}_\mathbf{x}$ the *minimum same support precision* (MSSP) of $\mathcal{A}$ on $\mathbf{x}$. Also, define the *minimum correct degree precision* (MCDP) of $\mathcal{A}$ on $\mathbf{x}$ to be the least integer Γ such that, if $\tau \geq \Gamma$, then the degree of $\mathcal{A}_\tau(\mathrm{fl}_\tau(\mathbf{x}))$ is the same as that of $\mathcal{A}'(\mathbf{x})$. Clearly, the MCDP does not exceed the MSSP, and therefore exists.

The construction of $\mathcal{A}'$ from $\mathcal{A}$ is very simple, and can be outlined as follows:

- All the instructions in $\mathcal{A}$ are kept the same.

```
  ▷ Change $\mathbf{x}$ to $\llbracket \mathbf{x} \rrbracket_\tau$.
1 $\llbracket x_2 \rrbracket \leftarrow \llbracket x_1 \rrbracket \times \llbracket x_1 \rrbracket$
2 $\llbracket x_2 \rrbracket \leftarrow \llbracket x_2 \rrbracket \times \llbracket x_1 \rrbracket$
3 $\llbracket x_3 \rrbracket \leftarrow \llbracket 2 \rrbracket_\tau$
4 $\llbracket x_2 \rrbracket \leftarrow \llbracket x_2 \rrbracket - \llbracket x_3 \rrbracket$
5 if $\llbracket x_2 \rrbracket \approxeq 0$ goto line 8
6 $\llbracket x_4 \rrbracket \leftarrow \llbracket 0 \rrbracket_\tau$
7 goto line 9
8 $\llbracket x_4 \rrbracket \leftarrow \llbracket 1 \rrbracket_\tau$
9 halt
  ▷ For all $i$, if $\llbracket x_i \rrbracket \approxeq 0$, then $\llbracket x_i \rrbracket \leftarrow (0, 0)$.
  ▷ Take $\langle \mathbf{x} \rangle$ to be the output of the algorithm.
```

Figure 3-4: The stabilized version of the algorithm in Figure 3-3.

- If a is a variable in $\mathcal{A}$, then it is replaced by a bracket coefficient $\llbracket a \rrbracket$ that approximates it.

- The input $\mathbf{x}$ is replaced by $\llbracket \mathbf{x} \rrbracket_\tau$, its floating-point bracket approximation.

- All constants in the algorithm are also replaced by their floating-point bracket approximations.

- All the arithmetic operations become those defined in Section 2.2.

- Checking whether a variable is equal to zero becomes checking whether a bracket coefficient is equivalent to zero, and comparing a bracket coefficient to zero is done as defined in Section 2.2.

- After the algorithm terminates, each register $\llbracket x_i \rrbracket$ is examined. If $\llbracket x_i \rrbracket$ is equivalent to zero, then it is rewritten with the bracket coefficient $(0, 0)$.

- Lastly, $\langle \mathbf{x} \rangle$ is taken to be the output of $\mathcal{A}'$.

An example of a stabilized algebraic algorithm is given in Figure 3-4; it is the stabilized version of the algorithm in Figure 3-3.

Observe that the technique doubles the number of registers used: for each variable, one register is used to hold its approximation, and the other its error term. The number of raw instructions also increases as the stabilized algorithm also has to calculate error terms. Hence, the stabilized algorithm does not eventually follow the execution path of the exact algorithm *per se*. However, if we think of an execution path as a sequence of lines of code that the algorithm follows, Shirayanagi and Sweedler showed that the execution paths eventually match, and that the stabilized version has all other properties stated in Theorem 3.1.

Chapter 4

Incomputability Result

In this chapter, we show that the MCP of a large class of algebraic algorithms is incomputable provided that the coefficients of the input polynomials are allowed to be any computable numbers. Recall that a *computable number* x is a real number such that there exists a Turing machine that produces the nth digit of the binary (or decimal) expansion of x, given any positive integer n. Equivalently, x is a computable number if there is a Turing machine that, when given an integer τ, outputs $\mathrm{fl}_\tau(x)$.

Definition 4.1. *A GCD-finding algorithm $\mathcal{A}$ is said to be* degree-aware *if, for any polynomials* $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, *and* $\mathbf{D}$ *such that* $\deg(\gcd(\mathbf{A},\mathbf{B})) \neq \deg(\gcd(\mathbf{C},\mathbf{D}))$, *it is true that* $\mathcal{A}(\mathbf{A},\mathbf{B})$ *follows different execution path from that of* $\mathcal{A}(\mathbf{C},\mathbf{D})$.

We will argue in Chapter 5 and Chapter 6, respectively, that the Euclidean algorithm and the QR-factorization algorithm are degree-aware. The main result of this chapter is the following theorem:

Theorem 4.2. *Let $\mathcal{A}$ be any degree-aware GCD-finding algebraic algorithm. There exists no Turing machine $\mathcal{D}_\mathcal{A}$ such that, for every pair of polynomials* $\mathbf{A}$ *and* $\mathbf{B}$ *whose coefficients are computable numbers,* $\mathcal{D}_\mathcal{A}(\mathbf{A},\mathbf{B})$ *gives the MCP of $\mathcal{A}$ on* $\mathbf{A}$ *and* $\mathbf{B}$.

Proof. Suppose by way of contradiction that such $\mathcal{D}_\mathcal{A}$ exists. Let Γ be the MCP of $\mathcal{A}$ when the input polynomials are $x+1$ and $x+1$. For any Turing machine $\mathcal{M}$ and any input s to $\mathcal{M}$, we define another Turing machine $\mathcal{X}_{\mathcal{M},s}$ as in Figure 4-1. Note that, if $\mathcal{M}$ on input s does not terminate, then $\mathcal{X}_{\mathcal{M},s}$ computes the number 1. Otherwise, it computes the number $1+2^{-\Gamma-k}$ where k is the number of steps $\mathcal{M}$ runs when fed with input s.

Next, we define Turing machine $\mathcal{E}$ as in Figure 4-2.

We claim that $\mathcal{E}$ rejects if and only if $\mathcal{M}$ does not halt on s. If $\mathcal{M}$ does not halt on s, then $\mathcal{X}_{\mathcal{M},s}$ computes the number 1. Therefore, $\mathcal{D}_\mathcal{A}(x+1, x+\mathcal{X}_{\mathcal{M},s})$ returns Γ, and $\mathcal{E}$ rejects.

```
$\mathcal{X}_{\mathcal{M},s}(\tau)$
If $\tau \le \Gamma$
   then Output $\mathrm{fl}_\tau(1)$.
   else Run $\mathcal{M}$ on $s$ for $\tau - \Gamma$ steps.
        if $\mathcal{M}$ does not terminate by $\tau - \Gamma$ steps
           then Output $\mathrm{fl}_\tau(1)$.
           else Let $k$ be the step at which $\mathcal{M}$ terminates.
                Output $\mathrm{fl}_\tau(1 + 2^{-\Gamma-k})$.
```

Figure 4-1: The Turing machine $\mathcal{X}_{\mathcal{M},s}$.

```
$\mathcal{E}(\mathcal{M}, s)$
Construct $\mathcal{X}_{\mathcal{M},s}$.
if $\mathcal{D}_{\mathcal{A}}(x+1, x+\mathcal{X}_{\mathcal{M},s}) = \Gamma$
   then reject
   else accept
```

Figure 4-2: The Turing machine $\mathcal{E}$.

On the other hand, if $\mathcal{M}$ halts on s, then $\mathcal{X}_{\mathcal{M},s}$ computes a number different from 1, and $\deg(\gcd(x+1, x+\mathcal{X}_{\mathcal{M},s})) = 0 \neq \deg(\gcd(x+1, x+1))$. As a result, the execution path of the exact computation $\mathcal{A}(x+1, x+1)$ must be different from that of $\mathcal{A}(x+1, x+\mathcal{X}_{\mathcal{M},s})$ because $\mathcal{A}$ is degree-aware. Let $\mathcal{A}'$ be $\mathcal{A}$'s stabilized version. Since $\mathrm{fl}_\Gamma(\mathcal{X}_{\mathcal{M},s}) = \mathrm{fl}_\Gamma(1)$, we have that the computation of $\mathcal{A}'_\Gamma(\mathrm{fl}_\Gamma(x+1), \mathrm{fl}_\Gamma(x+\mathcal{X}_{\mathcal{M},s}))$ must follow exactly the execution path of $\mathcal{A}(x+1, x+1)$ by the definition of Γ. Since this execution path is different from that of $\mathcal{A}(x+1, x+\mathcal{X}_{\mathcal{M},s})$, it must be the case that the MCP of $\mathcal{A}$ on $x+1$ and $x+\mathcal{X}_{\mathcal{M},s}$ is greater than Γ. Hence, $\mathcal{E}$ accepts.

Because $\mathcal{E}$ is a Turing machine that solves the halting problem, we arrive at a contradiction. □

Chapter 5

Euclidean Algorithm

In this chapter, we study the MCP of the Euclidean algorithm. In Section 5.1 we discuss how polynomials are represented and give the pseudocode of the algorithm. In Section 5.2, we state some useful facts from the theory of subresultants that we use to prove two of our main results. In Section 5.3, we outline our approach to determine the MCP and the MSSP of the Euclidean algorithm. In Section 5.4, we derive a general upper bound on the MCP and the MSSP of the Euclidean algorithm in terms of the largest coefficient and the smallest nonzero coefficient of intermediate polynomials. In Section 5.5, we restrict the domain of the coefficients of the input polynomials to be one of $\mathbb{Z}[\xi]$, and $\mathbb{Q}[\xi]$, and derive simpler upper bounds on the MCP and the MSSP.

5.1 Algorithm Description

The Euclidean algorithm for finding GCD of two univariate polynomials is based on the following recurrence relation:

$$\gcd(\mathbf{P},\mathbf{Q}) = \begin{cases} \mathbf{P}, & \text{if } \mathbf{Q} = 0 \\ \gcd(\mathbf{Q}, \mathrm{rem}(\mathbf{P},\mathbf{R})), & \text{if } \mathbf{Q} \neq 0 \end{cases}$$

In other words, to find the GCD of $\mathbf{P}$ and $\mathbf{Q}$, the algorithm checks whether $\mathbf{Q}$ is equal to the zero polynomial. If so, it outputs $\mathbf{P}$. Otherwise, it divides $\mathbf{P}$ with $\mathbf{Q}$ to find the remainder $\mathbf{R}$. Then, it recursively calls itself to find the GCD of $\mathbf{Q}$ and $\mathbf{R}$.

The pseudocode of the Euclidean algorithm we use is given in Figure 5-1. The only difference between our algorithm and the textbook one is that our algorithm normalize all intermediate polynomials by making them monic. We include this normalization because it makes the analysis of the MCP easier. Also, the algorithm checks whether a polynomial is zero or not by checking whether its degree is equal to $-\infty$ (a symbolic value) or not, and this is accomplished by feeding the polynomial

```
EUCLIDEAN($\mathbf{A}, \mathbf{B}$)
    ▷ Input: $\mathbf{A}, \mathbf{B} \in \mathbb{R}[x]$ such that both are not zero.
    ▷ Output: $\gcd(\mathbf{A}, \mathbf{B})$
if DEGREE($\mathbf{B}$) $= -\infty$
    then return $\mathbf{A}$
if DEGREE($\mathbf{A}$) $= -\infty$
    then return $\mathbf{B}$
$\mathbf{R}_0 \leftarrow \mathbf{A}/\mathbf{A}[\mathfrak{d}]$
$\mathbf{R}_1 \leftarrow \mathbf{B}/\mathbf{B}[\mathfrak{d}]$
$w \leftarrow 0$
while DEGREE($\mathbf{R}_{w+1}$) $= -\infty$
        do $\mathbf{W}_{w+2} \leftarrow$ REMAINDER($\mathbf{R}_w, \mathbf{R}_{w+1}$)
           $\mathbf{R}_{w+2} \leftarrow \mathbf{W}_{w+2}/\mathbf{W}_{w+2}[\mathfrak{d}]$
           $w \leftarrow w + 1$
return $\mathbf{R}_w$
```

Figure 5-1: Pseudocode of the Euclidean algorithm.

```
REMAINDER($\mathbf{A}, \mathbf{B}$)
    ▷ Input: $\mathbf{A}, \mathbf{B} \in \mathbb{R}[x]$ with $\mathbf{B} \neq 0$ and monic.
    ▷ Output: rem($\mathbf{A}, \mathbf{B}$).
if DEGREE($\mathbf{A}$) $<$ DEGREE($\mathbf{B}$)
    then return $\mathbf{A}$
$r \leftarrow$ DEGREE($\mathbf{A}$) $-$ DEGREE($\mathbf{B}$)
$\mathbf{T}_r \leftarrow \mathbf{A}$
while $r \geq 0$
        do $\mathbf{T}_{r-1} \leftarrow \mathbf{T}_r - \mathbf{T}_r[r]x^r\mathbf{B}$
           $r \leftarrow r - 1$
return $\mathbf{T}_{-1}$
```

Figure 5-2: Pseudocode of REMAINDER, a procedure to calculate remainders.

to the procedure DEGREE given in Section 3.2.

The procedure REMAINDER computes the remainder of the two given polynomials, assuming that the second polynomial is monic. Its pseudocode is given in Figure 5-2. Note that, although polynomial summations and multiplications are denoted by arithmetic operators in the pseudocode, REMAINDER actually calls the appropriate procedures that compute the sum or the product as discussed in Section 3.2.

The way we determine degrees of polynomials implies that EUCLIDEAN is degree-aware. The reason is that, in the last time REMAINDER is called before EUCLIDEAN returns, the second argument to REMAINDER is the GCD, and DEGREE is called inside REMAINDER to determine its degree. Therefore, if the degree of the GCD changes, the execution paths of DEGREE and EUCLIDEAN change as well.

5.2 Subresultant Theory

Our analysis of the MCP of the Euclidean algorithm relies heavily on knowing exactly what each coefficient of every intermediate polynomial generated by EUCLIDEAN is. We obtain this knowledge by the theory of subresultants, which relates the coefficients to determinants of matrices called the *Sylvester–Habicht* of the input polynomials. This section states some important facts from the subresultant theory we will make use of. We refer the reader to [8] and [4] for more details.

The sequence of polynomials $\mathbf{R}_0$, $\mathbf{R}_1$, …, $\mathbf{R}_w$ in the pseudocode of the Euclidean algorithm defines the *polynomial remainder sequence* (PRS) of $\mathbf{A}$ and $\mathbf{B}$. Notice that the PRS always begins with $\mathbf{A}$ followed by $\mathbf{B}$ and ends with $\gcd(\mathbf{A},\mathbf{B})$. Moreover, an element of the PRS is the monic polynomial similar to the remainder of the two polynomials immediately preceding it.

Let $\mathbf{A} = a_d x^d + a_{d-1}x^{d-1} + \cdots + a_0$ with $a_d \neq 0$, and let $\mathbf{B} = b_{d-1}x^{d-1} + b_{d-2}x^{d-2} + \cdots + b_0$. Any coefficient of $\mathbf{B}$ can be zero, but we require that $\mathbf{B}$ is not a zero polynomial. Then, for j such that $0 \leq j < d$, the *jth Sylvester–Habicht matrix* of $\mathbf{A}$ and $\mathbf{B}$, denoted by $\mathrm{Syl}_j(\mathbf{A},\mathbf{B})$ or simply Syl_j, is the matrix:

$$\underbrace{\begin{bmatrix} a_d & a_{d-1} & \cdots & \cdots & \cdots & a_1 & a_0 & & & \\ & a_d & a_{d-1} & \cdots & \cdots & \cdots & a_1 & a_0 & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & a_d & a_{d-1} & \cdots & \cdots & \cdots & a_1 & a_0 \\ b_{d-1} & b_{d-2} & \cdots & \cdots & b_1 & b_0 & & & & \\ & b_{d-1} & b_{d-2} & \cdots & \cdots & b_1 & b_0 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & b_{d-1} & b_{d-2} & \cdots & \cdots & b_1 & b_0 & \\ & & & & b_{d-1} & b_{d-2} & \cdots & \cdots & b_1 & b_0 \end{bmatrix}}_{2d-1-j \text{ columns}} \begin{array}{l} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\end{array}}\right\} d-1-j \quad \text{rows} \\ \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\end{array}}\right\} d-j \quad \text{rows}\end{array}$$

We refer to the 0th Sylvester–Habicht matrix simply by the Sylvester–Habicht matrix, and denotes it by the symbol $\mathrm{Syl}(\mathbf{A},\mathbf{B})$. For example, if $\mathbf{P} = p_4x^4 + \cdots + p_0$ and $\mathbf{Q} = q_3x^3 + \cdots + q_0$, then the Sylvester–Habicht matrices of $\mathbf{P}$ and $\mathbf{Q}$ are given in Figure 5-3

Let $\mathrm{Syl}_{j,\ell}(\mathbf{A},\mathbf{B})$ be the square matrix of dimension $(2d-1-2j) \times (2d-1-2j)$ obtained by excluding all columns of $\mathrm{Syl}_j(\mathbf{A},\mathbf{B})$ except the first $2d-2-2j$ columns and the $(\ell+1)$st column from the right. Examples of these square submatrices of Sylvester–Habicht matrices are given in Figure 5-4. Note also that $\mathrm{Syl}_{0,0}(P,Q)$ is equal to the Sylvester–Habicht matrix itself.

For $0 \leq j < d$, the jth *subresultant* A and B, denoted by $\mathbf{S}_j(\mathbf{A},\mathbf{B})$ or simply $\mathbf{S}_j$, is the polynomial

$$\mathbf{S}_j(\mathbf{A},\mathbf{B}) = \sum_{\ell=0}^{j} \det(\mathrm{Syl}_{j,\ell}(\mathbf{A},\mathbf{B}))\ x^{\ell}. \tag{5.1}$$

$$\mathrm{Syl}(\mathbf{P},\mathbf{Q}) = \mathrm{Syl}_0(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_2 & p_1 & p_0 & & \\ & p_4 & p_3 & p_2 & p_1 & p_0 & \\ & & p_4 & p_3 & p_2 & p_1 & p_0 \\ q_3 & q_2 & q_1 & q_0 & & & \\ & q_3 & q_2 & q_1 & q_0 & & \\ & & q_3 & q_2 & q_1 & q_0 & \\ & & & q_3 & q_2 & q_1 & q_0 \end{bmatrix}$$

$$\mathrm{Syl}_1(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_2 & p_1 & p_0 & \\ & p_4 & p_3 & p_2 & p_1 & p_0 \\ q_3 & q_2 & q_1 & q_0 & & \\ & q_3 & q_2 & q_1 & q_0 & \\ & & q_3 & q_2 & q_1 & q_0 \end{bmatrix}$$

$$\mathrm{Syl}_2(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_2 & p_1 & p_0 \\ q_3 & q_2 & q_1 & q_0 & \\ & q_3 & q_2 & q_1 & q_0 \end{bmatrix}$$

$$\mathrm{Syl}_3(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} q_3 & q_2 & q_1 & q_0 \end{bmatrix}$$

Figure 5-3: Examples of Sylvester–Habicht matrices.

$$\mathrm{Syl}_{1,0}(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_2 & p_1 & \\ & p_4 & p_3 & p_2 & p_0 \\ q_3 & q_2 & q_1 & q_0 & \\ & q_3 & q_2 & q_1 & \\ & & q_3 & q_2 & q_0 \end{bmatrix} \qquad \mathrm{Syl}_{1,1}(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_2 & p_1 & p_0 \\ & p_4 & p_3 & p_2 & p_1 \\ q_3 & q_2 & q_1 & q_0 & \\ & q_3 & q_2 & q_1 & q_0 \\ & & q_3 & q_2 & q_1 \end{bmatrix}$$

$$\mathrm{Syl}_{2,0}(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_0 \\ q_3 & q_2 & \\ & q_3 & q_0 \end{bmatrix} \quad \mathrm{Syl}_{2,1}(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_1 \\ q_3 & q_2 & q_0 \\ & q_3 & q_1 \end{bmatrix} \quad \mathrm{Syl}_{2,2}(\mathbf{P},\mathbf{Q}) = \begin{bmatrix} p_4 & p_3 & p_2 \\ q_3 & q_2 & q_1 \\ & q_3 & q_2 \end{bmatrix}$$

$$\mathrm{Syl}_{3,0}(\mathbf{P},\mathbf{Q}) = [q_0] \qquad \mathrm{Syl}_{3,1}(\mathbf{P},\mathbf{Q}) = [q_1] \qquad \mathrm{Syl}_{3,2}(\mathbf{P},\mathbf{Q}) = [q_2] \qquad \mathrm{Syl}_{3,3}(\mathbf{P},\mathbf{Q}) = [q_3].$$

Figure 5-4: Examples of square submatrices of Sylvester–Habicht matrices.

The following two propositions give some properties of subresultant polynomials. Their proofs can be found in [8].

Proposition 5.1. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two polynomials with* $\partial(\mathbf{A}) > \partial(\mathbf{B})$, *and let* $d = \partial(\mathbf{A})$. *Then, there exists a unique pair of polynomials* $\mathbf{U}_j$ *and* $\mathbf{V}_j$ *such that*

(a) $\partial(\mathbf{U}_j) \leq d - j - 2$, *and*

(b) $\partial(\mathbf{V}_j) \leq d - j - 1$, *and*

(c) $\mathbf{U}_j\mathbf{A} + \mathbf{V}_j\mathbf{B} = \mathbf{S}_j(\mathbf{A},\mathbf{B})$, *and*

(d) each the coefficient of $\mathbf{U}_j$ *and* $\mathbf{V}_j$ *is the determinant of a minor of* $\mathrm{Syl}_{j,0}(\mathbf{A},\mathbf{B})$ *obtained by removing the last column of and one of its row.*

Proposition 5.2. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two real polynomials with* $d = \eth(\mathbf{A}) > \eth(\mathbf{B})$. *The subresultants* $\mathbf{S}_d, \mathbf{S}_{d-1}, \ldots, \mathbf{S}_0$ *of* $\mathbf{A}$ *and* $\mathbf{B}$ *are either zero or similar to a polynomial in the polynomial remainder sequence* $\mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_w$ *of* $\mathbf{A}$ *and* $\mathbf{B}$.

In particular, for any i *such that* $1 \leq i < w$, *let* $j = \eth(\mathbf{R}_i)$, *and* $k = \eth(\mathbf{R}_{i+1})$. *Then,*

$$\mathbf{S}_j \sim \mathbf{R}_i, \quad \textit{and} \quad \mathbf{S}_{j-1} \sim \mathbf{S}_k \sim \mathbf{R}_{i+1}. \tag{5.2}$$

Furthermore, let $s_j = \mathbf{S}_j[\eth]$, $s_{j-1} = \mathbf{S}_{j-1}[\eth]$, *and* $s_k = \mathbf{S}_k[\eth]$. *Then,*

(a) $\mathbf{S}_k$ *is given by*

$$\mathbf{S}_k = \left(\frac{s_{j-1}}{s_j}\right)^{j-k-1} \mathbf{S}_{j-1}, \tag{5.3}$$

(b) $\mathbf{S}_\ell = 0$ *for all* $k < \ell < j-1$, *and*

(c) $\mathbf{S}_{k-1}$ *is given by*

$$\mathbf{S}_{k-1} = \frac{(-1)^{j-k+1} s_k s_{j-1}}{s_j^2} \mathbf{S}_j + (-1)^{d-j} \frac{A[\eth]^2}{s_j^2} (\mathbf{U}_j \mathbf{V}_{k-1} - \mathbf{U}_{k-1} \mathbf{V}_j) \mathbf{S}_{j-1}. \tag{5.4}$$

5.3 Approach

We derive a worst-case upper bound on the MCP of the Euclidean algorithm by the following three-step process:

1. Find an upper bound on how large can the error terms of coefficients of intermediate polynomials can become as the stabilized algorithm evolves, assuming that the stabilized algorithm follows the execution path of the exact algorithm.

2. Find a lower bound on the size of the smallest nonzero numerical value that arises during the evolution of the exact algorithm.

3. Find a precision at and beyond which the upper bound in Step 1 never exceeds the lower bound in Step 2. This precision is a converging precision, and therefore is greater than or equal to the MCP.

The rationale behind this method is as follows:

1. The first place that the execution path of the stabilized algorithm differs from the execution path of the exact algorithm must be at a conditional statement. In the Euclidean algorithm, there is only one conditional statement that involves real numbers: the one in the **while**

loop in Line 2 of Degree. Other conditional statements, such as that in the **while** loop of Euclidean or the **if** statement in Line 2 of Remainder, compare the degrees given out by Degree to another constant. Hence, if the stabilized algorithm always branches correctly at the **while** loop in Degree, then it follows the execution path of the exact algorithm.

2. The conditional statement discussed above involves testing whether a coefficient of a polynomial is equal to zero. So, in the stabilized algorithm, it involves testing whether a bracket coefficient that approximates the coefficient is equivalent to zero. According to Section 2.2, we decide whether a bracket coefficient is equivalent to zero by testing if the error term is greater than or equal to the absolute value of the approximate value.

3. Testing whether a bracket coefficient is equal to zero can give a wrong result only when the bracket coefficient actually approximates a nonzero quantity, but its error term is so large that the bracket coefficient is in effect equivalent to zero [11].

4. Therefore, if the precision is large enough so that no error term is larger than the smallest numerical value that arises during the evolution of the algorithm, then any bracket coefficient that approximates a nonzero value cannot be equivalent to zero. As a result, no equality testing gives a wrong result, and consequently the stabilized algorithm follows the same execution path as the exact algorithm.

5.4 General Bound

In this section, we derive an upper bound on the MCP of the Euclidean algorithm in terms of the largest and the smallest values that can arise during the evolution of the Euclidean algorithm. We assume *a priori* that τ is larger than the MCP, so that the stabilized algorithm follows the execution path of the exact algorithm.

In this section and the next section, we let:

- $\mathbf{A}$ and $\mathbf{B}$ be real polynomials such that $\partial(\mathbf{A}) > \partial(\mathbf{B})$ and $\mathbf{B} \neq 0$.
- M is a positive constant that is larger than the absolute value of any coefficient of any intermediate polynomials in the exact computation.
- m is a positive constant that is (1) less than or equal to one, and (2) less than or equal to the absolute value of any coefficient of any intermediate polynomials in the exact computation.

As a convention, if a is a variable in the exact version of Euclidean, then $\llbracket a \rrbracket$ is the corresponding bracket coefficient in the stabilized version (see Section 3.3). For example, $\llbracket \mathbf{A} \rrbracket$ and $\llbracket \mathbf{B} \rrbracket$ are the inputs of the stabilized version, and $\llbracket \mathbf{R}_2 \rrbracket$ corresponds to the polynomial $\mathbf{R}_2 = \text{Remainder}(\mathbf{R}_0, \mathbf{R}_1)$ in the exact algorithm. According to the convention in Section 2.2, a bracket coefficient approximates the exact variable it corresponds to. This convention is appropriate in our case because we assume that the stabilized algorithm follows the execution path of the exact algorithm. Thus, for example, since $\llbracket \mathbf{A} \rrbracket \approxeq \mathbf{A}$ and $\llbracket \mathbf{B} \rrbracket \approxeq \mathbf{B}$, then $\llbracket \mathbf{R}_2 \rrbracket$ also approximates $\mathbf{R}_2$ because $\llbracket \mathbf{R}_2 \rrbracket$ is computed from $\llbracket \mathbf{A} \rrbracket$ and $\llbracket \mathbf{B} \rrbracket$.

As another convention, if the input to a procedure are bracket coefficients, then we mean running the stabilized version on the bracket coefficient input; otherwise, we mean running the exact version on the exact input. For example, $\textsc{Euclidean}(\llbracket \mathbf{A} \rrbracket, \llbracket \mathbf{B} \rrbracket)$ means running the stabilized version of the Euclidean algorithm on bracket polynomials $\llbracket \mathbf{A} \rrbracket$ and $\llbracket \mathbf{B} \rrbracket$, and $\textsc{Euclidean}(\mathbf{A}, \mathbf{B})$ means running the exact algorithm on exact polynomials $\mathbf{A}$ and $\mathbf{B}$.

We now proceed with Step 1 of the method outlined in the last section. The following lemma gives a bound on the error terms of the output of $\textsc{Remainder}$.

Lemma 5.3. *If*

(a) $\lfloor \mathbf{A}[k] \rceil \geq 2^{1-\tau} |\mathbf{A}[k]|$ *and* $\lfloor \mathbf{B}[k] \rceil \geq 2^{1-\tau} |\mathbf{B}[k]|$ *for all* k*,*

(b) ε *is a constant such that* $\lfloor \mathbf{A}[k] \rceil \leq \varepsilon$ *and* $\lfloor \mathbf{B}[k] \rceil \leq \varepsilon$ *for all* k*,*

(c) $\mathbf{T}_{-1}, \mathbf{T}_0, \ldots, \mathbf{T}_{\partial(\mathbf{A})-\partial(\mathbf{B})}$ *are as in the pseudocode of* $\textsc{Remainder}(\mathbf{A}, \mathbf{B})$*,*

(d) $\tau \geq 5$ *and is greater than the MCP of the Euclidean algorithm,*

then

$$\lfloor \mathbf{T}_r[k] \rceil \leq (56M)^{\partial(\mathbf{A})-\partial(\mathbf{B})-r} \varepsilon$$

for all relevant r *and* k*. In particular, if* $\llbracket \mathbf{C} \rrbracket = \textsc{Remainder}(\llbracket \mathbf{A} \rrbracket, \llbracket \mathbf{B} \rrbracket)$*, then*

$$\lfloor \mathbf{C}[k] \rceil \leq (56M)^{\partial(\mathbf{A})-\partial(\mathbf{B})+1} \varepsilon.$$

Proof. The proof is by induction on r. For the base case, $r = \partial(\mathbf{A}) - \partial(\mathbf{B})$, we have that $\lfloor \mathbf{T}_r[k] \rceil = \lfloor \mathbf{A}[k] \rceil \leq \varepsilon = (56M)^0 \varepsilon$ for all k. Next, we assume that the lemma is true for some $r \leq \partial(\mathbf{A}) - \partial(\mathbf{B})$. For $0 \leq k \leq \partial(\mathbf{T}_r)$, we have that, by Line 6 of $\textsc{Remainder}$,

$$\llbracket \mathbf{T}_{r-1}[k] \rrbracket = \llbracket \mathbf{T}_r[k] \rrbracket - \llbracket \mathbf{T}_r[r] \rrbracket \llbracket \mathbf{B}[k-r] \rrbracket.$$

Using Lemma B.1, we have that

$$\begin{aligned}
\lfloor \mathbf{T}_{r-1}[k] \rceil &\leq 7 \max\left\{ \lfloor \mathbf{T}_r[k] \rceil, \lfloor \mathbf{T}_r[r] \mathbf{B}[k-r] \rceil \right\} \\
&\leq 7 \max\left\{ \lfloor \mathbf{T}_r[k] \rceil, 8 \max\{ |\mathbf{T}_r[r]|, |\mathbf{B}[k-r]| \} \max\{ \lfloor \mathbf{T}_r[r] \rceil, \lfloor \mathbf{B}[k-r] \rceil \} \right\} \\
&\leq 7 \max\left\{ (56M)^{\partial(\mathbf{A})-\partial(\mathbf{B})-r} \varepsilon, 8M (56M)^{\partial(\mathbf{A})-\partial(\mathbf{B})-r} \varepsilon \right\} \\
&\leq 56M \cdot (56M)^{\partial(\mathbf{A})-\partial(\mathbf{B})-r} \varepsilon \leq (56M)^{\partial(\mathbf{A})-\partial(\mathbf{B})-r+1} \varepsilon.
\end{aligned}$$

By induction, the lemma is true for all $-1 \le r \le \partial(\mathbf{A}) - \partial(\mathbf{B})$. The inequality involving the error terms of $\llbracket \mathbf{C} \rrbracket$ follows from the fact that $\mathbf{C} = \mathbf{T}_{-1}$. □

The following lemma gives a bound on the error terms of the output of EUCLIDEAN.

Lemma 5.4. *If*

(a) $\lfloor \mathbf{A}[k] \rceil \ge 2^{1-\tau} |\mathbf{A}[k]|$ *and* $\lfloor \mathbf{B}[k] \rceil \ge 2^{1-\tau} |\mathbf{B}[k]|$ *for all* k,

(b) ε *is a constant such that* $\lfloor \mathbf{A}[k] \rceil \le \varepsilon$ *and* $\lfloor \mathbf{B}[k] \rceil \le \varepsilon$ *for all* k,

(c) $\mathbf{R}_0$, $\mathbf{R}_1$, $\ldots$, *and* $\mathbf{W}_2$, $\mathbf{W}_3$, $\ldots$, *are as in the pseudocode of* EUCLIDEAN$(\mathbf{A}, \mathbf{B})$,

(d) τ *is greater than 5 and the MCP of the Euclidean algorithm, and is large enough so that* $\lfloor \mathbf{W}_{w+2}[\partial] \rceil \le |\mathbf{W}_{w+2}[\partial]|/4$ *for all* $w \ge 0$,

then

$$\lfloor \mathbf{R}_{w+1}[k] \rceil \le \frac{48^w (56M)^{\partial(\mathbf{R}_0) - \partial(\mathbf{R}_w) + 2w}}{m^{3w}} \varepsilon$$

for all $w \ge 0$ *and for all* k. *In particular, if* $\llbracket \mathbf{G} \rrbracket = $ EUCLIDEAN$(\llbracket \mathbf{A} \rrbracket, \llbracket \mathbf{B} \rrbracket)$, *then*

$$\lfloor \mathbf{G}[k] \rceil \le \frac{48^{\partial(\mathbf{A})} (56M)^{3\partial(\mathbf{A})}}{m^{3\partial(\mathbf{A})}} \varepsilon.$$

Proof. The proof is by induction on w. There are two base cases: $w = 0$ and $w = 1$. For $w = 0$, we have that, for all k,

$$\lfloor \mathbf{R}_0[k] \rceil \le \varepsilon = \frac{48^0 (56M)^{\partial(\mathbf{R}_0) - \partial(\mathbf{R}_0) + 0}}{m^0} \varepsilon.$$

For $w = 1$, recall that $\llbracket \mathbf{W}_2 \rrbracket = $ REMAINDER$(\llbracket \mathbf{R}_0 \rrbracket, \llbracket \mathbf{R}_1 \rrbracket)$. Since the error term of every coefficient of both $\llbracket \mathbf{R}_0 \rrbracket$ and $\llbracket \mathbf{R}_1 \rrbracket$ is bounded above by ε, we have that, by Lemma 5.3,

$$\lfloor \mathbf{W}_2[k] \rceil \le (56M)^{\partial(\mathbf{R}_0) - \partial(\mathbf{R}_1) + 1} \varepsilon.$$

for all k. Hence,

$$\begin{aligned} \lfloor \mathbf{R}_2[k] \rceil &= \left\lfloor \frac{\mathbf{W}_2[k]}{\mathbf{W}_2[\partial]} \right\rceil = \left\lfloor \mathbf{W}_2[k] \cdot \frac{1}{\mathbf{W}_2[\partial]} \right\rceil \\ &\le 8 \max\left\{ |\mathbf{W}_2[k]|, \left| \frac{1}{\mathbf{W}_2[\partial]} \right| \right\} \max\left\{ \lfloor \mathbf{W}_2[k] \rceil, \left\lfloor \frac{1}{\mathbf{W}_2[\partial]} \right\rceil \right\} \\ &\le 8 \frac{M}{m} \max\left\{ \lfloor \mathbf{W}_2[k] \rceil, \left\lfloor \frac{1}{\mathbf{W}_2[\partial]} \right\rceil \right\}. \end{aligned}$$

Since we assume that τ is large enough so that $\lfloor \mathbf{W}_2[k] \rceil \le |\mathbf{W}_2[k]|/4$ for all k, we can use Lemma B.1 to deduce that

$$\left\lfloor \frac{1}{\mathbf{W}_2[\partial]} \right\rceil \le 6 \frac{\lfloor \mathbf{W}_2[k] \rceil}{\mathbf{W}_2[k]^2} \le \frac{6 (56M)^{\partial(\mathbf{R}_0) - \partial(\mathbf{R}_1) + 1}}{m^2} \varepsilon$$

Therefore,

$$\begin{aligned}\lfloor \mathbf{R}_2[k] \rceil &\le 8\frac{M}{m}\max\left\{(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_1)+1}\varepsilon,\ \frac{6(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_1)+1}}{m^2}\varepsilon\right\}\\ &= \frac{48(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_1)+2}}{m^3}\varepsilon.\end{aligned}$$

We have proved the case where $w = 1$, and established the two base cases.

As for the induction hypothesis, assume that, for some $w \ge 1$, the statement is true for $w-1$ and w. Namely,

$$\lfloor \mathbf{R}_{w+1}[k] \rceil \le \frac{48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_w)+2w}}{m^{3w}}\varepsilon,$$

and

$$\lfloor \mathbf{R}_w[k] \rceil \le \frac{48^{w-1}(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_{w-1})+2(w-1)}}{m^{3(w-1)}}\varepsilon \le \frac{48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_w)+2w}}{m^{3w}}\varepsilon.$$

Consider the bracket polynomial $\llbracket \mathbf{W}_{w+2} \rrbracket = \textsc{Remainder}(\llbracket \mathbf{R}_w \rrbracket, \llbracket \mathbf{R}_{w+1} \rrbracket)$. By Lemma 5.3 and the previous two inequalities,

$$\begin{aligned}\lfloor \mathbf{W}_{w+2}[k] \rceil &\le (56M)^{\partial(R_w)-\partial(R_{w+1})+1}\cdot\frac{48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_w)+2w}}{m^{3w}}\varepsilon\\ &= \frac{48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_{w+1})+2w+1}}{m^{3w}}\varepsilon\end{aligned}$$

for all k. Hence,

$$\begin{aligned}\lfloor \mathbf{R}_{w+2}[k] \rceil &= \left\lfloor \frac{\mathbf{W}_{w+2}[k]}{\mathbf{W}_{w+2}[\partial]} \right\rceil = \left\lfloor \mathbf{W}_{w+2}[k]\cdot\frac{1}{\mathbf{W}_{w+2}[\partial]} \right\rceil\\ &\le 8\max\left\{|\mathbf{W}_{w+2}[k]|, \left|\frac{1}{\mathbf{W}_{w+2}[\partial]}\right|\right\}\max\left\{\lfloor \mathbf{W}_{w+2}[k] \rceil, \left\lfloor \frac{1}{\mathbf{W}_{w+2}[\partial]} \right\rceil\right\}\\ &\le 8\frac{M}{m}\max\left\{\lfloor \mathbf{W}_{w+2}[k] \rceil, \left\lfloor \frac{1}{\mathbf{W}_{w+2}[\partial]} \right\rceil\right\}\end{aligned}$$

Again, by our assumption that τ is large enough, we have

$$\left\lfloor \frac{1}{\mathbf{W}_{w+2}[\partial]} \right\rceil \le \frac{6}{m^2}\cdot\frac{48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_{w+1})+2w+1}}{m^{3w}}\varepsilon \le \frac{6\cdot 48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_{w+1})+2w+1}}{m^{3w+2}}\varepsilon.$$

Therefore,

$$\begin{aligned}\lfloor \mathbf{R}_{w+2}[k] \rceil &\le 8\frac{M}{m}\max\left\{\frac{48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_{w+1})+2w+1}}{m^{3w}}\varepsilon,\ \frac{6\cdot 48^w(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_{w+1})+2w+1}}{m^{3w+2}}\varepsilon\right\}\\ &= \frac{48^{w+1}(56M)^{\partial(\mathbf{R}_0)-\partial(\mathbf{R}_1)+2(w+1)}}{m^{3(w+1)}}\varepsilon.\end{aligned}$$

Thus, by induction, the statement is true for all $w \geq 0$. The bound on the error terms of coefficients of $\lfloor \mathbf{G} \rceil$ follows from the fact that there can be at most $\eth(\mathbf{A}) + 1$ polynomials in the PRS, so $w \leq \eth(\mathbf{A})$. □

A bound on the MCP and the MSSP of the Euclidean algorithm in the case where coefficients of the input polynomials can be any real number is given in the following theorem:

Theorem 5.5. *Let $d = \eth(\mathbf{A})$. Then, the MCP and the MSSP of* REMAINDER$(\mathbf{A}, \mathbf{B})$ *is*

$$\widetilde{O}(d(\log M - \log m)).$$

Proof. Recall that EUCLIDEAN$([\![\mathbf{A}]\!], [\![\mathbf{B}]\!])$ follows the execution path of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$ if none of the error terms of any intermediate quantity generated during the computation of EUCLIDEAN$([\![\mathbf{A}]\!].[\![\mathbf{B}]\!])$ exceeds the absolute value of the smallest nonzero intermediate quantity generated during the computation of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$.

Lemma 5.4 implies that the error term of any intermediate quantity generated during the computation of EUCLIDEAN$([\![\mathbf{A}]\!]_\tau, [\![\mathbf{B}]\!]_\tau)$ does not exceed

$$\begin{aligned} \frac{48^{\eth(\mathbf{A})}(56M)^{3\eth(\mathbf{A})}}{m^{3\eth(\mathbf{A})}} \cdot \epsilon_\tau\Big(\max_{0 \leq k \leq \eth(\mathbf{A})} \{\mathbf{A}[k], \mathbf{B}[k]\} \Big) &\leq \frac{48^{\eth(\mathbf{A})}(56M)^{3\eth(\mathbf{A})}}{m^{3\eth(\mathbf{A})}} 2^{-\tau} M \\ &= \frac{48^{\eth(\mathbf{A})}(56M)^{3\eth(\mathbf{A})+1}}{m^{3\eth(\mathbf{A})}} 2^{-\tau} \end{aligned}$$

if τ is large enough so that EUCLIDEAN$([\![\mathbf{A}]\!]_\tau, [\![\mathbf{B}]\!]_\tau)$ follows the execution path of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$, and that $\lfloor \mathbf{W}_{w+2}[\eth] \rceil \leq |\mathbf{W}_{w+2}[\eth]|/4$ for all $w \geq 0$.

Setting

$$\tau = \log_2 \frac{4 \cdot 48^{\eth(\mathbf{A})}(56M)^{3\eth(\mathbf{A})+1}}{m^{3\eth(\mathbf{A})+1}} = \widetilde{O}(d(\log M - \log m)),$$

we have that no error term exceeds

$$\frac{48^{\eth(\mathbf{A})}(56M)^{3\eth(\mathbf{A})+1}}{m^{3\eth(\mathbf{A})}} \cdot \frac{m^{3\eth(\mathbf{A})+1}}{4 \cdot 48^{\eth(\mathbf{A})}(56M)^{3\eth(\mathbf{A})+1}} = \frac{m}{4}.$$

Since, by definition, m is less than or equal to the absolute value of any non-zero intermediate quantities generated during the computation of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$, we have that EUCLIDEAN$([\![\mathbf{A}]\!]_\tau, [\![\mathbf{B}]\!]_\tau)$ follows the same execution path as that of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$. Additionally, $\lfloor \mathbf{W}_{w+2}[\eth] \rceil \leq \frac{m}{4} \leq |\mathbf{W}_{w+2}[\eth]|/4$ for all $w \geq 0$. Therefore, the value in the statement of the lemma is greater than or equal to the MCP of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$.

Moreover, since no the error term of the output exceeds the absolute value of the smallest coefficient in the output, we have that EUCLIDEAN$([\![\mathbf{A}]\!]_\tau, [\![\mathbf{B}]\!]_\tau)$ has the same support as that of EUCLIDEAN$(\mathbf{A}, \mathbf{B})$. Thus, the value also gives an upper bound for the MSSP as well. □

5.5 Domain-specific Bounds

The bound in Theorem 5.5 is not very useful because it involves M and m which are quantities that are hard to determine. However, it is possible to derive simpler and more useful bounds if we restrict the domain of coefficients to sets where exact computation can be carried out. To this end, we derive, in this section, simpler bounds on the MCP and the MSSP in cases where coefficients of the input polynomials are from the sets $\mathbb{Z}[\xi]$ and $\mathbb{Q}[\xi]$, where ξ is a real algebraic integer not equal to one.

Consider the computation of $\textsc{Remainder}(\mathbf{R}_i, \mathbf{R}_{i+1})$ for some i. Let $j = \mathfrak{d}(\mathbf{R}_i)$ and $k = \mathfrak{d}(\mathbf{R}_{i+1})$. Then, Proposition 5.2 tells us that $\mathbf{R}_i = \frac{\mathbf{S}_j}{\mathbf{S}_j[\mathfrak{d}]} = \frac{\mathbf{S}_j}{s_j}$ and $\mathbf{R}_{i+1} = \frac{\mathbf{S}_{j-1}}{\mathbf{S}_{j-1}[\mathfrak{d}]} = \frac{\mathbf{S}_{j-1}}{s_{j-1}}$ and

$$\begin{aligned}
\mathbf{S}_{k-1} &= \frac{(-1)^{j-k+1} s_k s_{j-1}}{s_j^2}\mathbf{S}_j + (-1)^{d-j}\frac{A[\mathfrak{d}]^2}{s_j^2}(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)\mathbf{S}_{j-1}\\
\mathbf{S}_{k-1} &= \frac{(-1)^{j-k+1} s_k s_{j-1}}{s_j^2}s_j\mathbf{R}_i + (-1)^{d-j}\frac{A[\mathfrak{d}]^2}{s_j^2}(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)s_{j-1}\mathbf{R}_{i+1}\\
\mathbf{S}_{k-1} &= \frac{(-1)^{j-k+1} s_k s_{j-1}}{s_j}\mathbf{R}_i + (-1)^{d-j}\frac{A[\mathfrak{d}]^2 s_{j-1}}{s_j^2}(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)\mathbf{R}_{i+1}\\
\frac{(-1)^{j-k+1} s_j}{s_k s_{j-1}}\mathbf{S}_{k-1} &= \mathbf{R}_i - (-1)^{d-k}\frac{A[\mathfrak{d}]^2 s_k}{s_j}(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)\mathbf{R}_{i+1}.
\end{aligned}$$

In other words, the LHS of the above equation is the remainder of dividing $\mathbf{R}_i$ by $\mathbf{R}_{i+1}$ (because $\mathbf{S}_{k-1}$ has degree lower than that of $\mathbf{R}_{i+1}$), and the quotient of the division is given by

$$\mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1}) = (-1)^{d-k}\frac{A[\mathfrak{d}]^2 s_k}{s_j}(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j). \tag{5.5}$$

Moreover, the sequences of polynomials $\mathbf{T}_{-1}, \mathbf{T}_0, \ldots, \mathbf{T}_{\mathfrak{d}(\mathbf{R}_i)-\mathfrak{d}(\mathbf{R}_{i+1})}$ that arise during the computation of $\textsc{Remainder}(\mathbf{R}_i, \mathbf{R}_{i+1})$ satisfy the following equation:

$$\begin{aligned}
\mathbf{T}_r &= \mathbf{R}_i - \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1:\mathfrak{d}]\cdot\mathbf{R}_{i+1}\\
&= \frac{\mathbf{S}_j}{s_j} - \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1:\mathfrak{d}]\cdot\frac{\mathbf{S}_{j-1}}{s_{j-1}}\\
&= \frac{1}{s_j s_{j-1}}\Big(s_{j-1}\mathbf{S}_j - s_j\cdot\mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1:\mathfrak{d}]\cdot\mathbf{S}_{j-1}\Big)
\end{aligned} \tag{5.6}$$

for any r such that $-1 \le r \le \mathfrak{d}(\mathbf{R}_i) - \mathfrak{d}(\mathbf{R}_{i+1})$. Using (5.5) and (5.6), we can fine simple bounds for M and m when all coefficients of $\mathbf{A}$ and $\mathbf{B}$ belong to: $\mathbb{Z}[\xi]\langle\langle N\rangle\rangle$, and $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$, where N is a positive integer, and thereby can plug the bounds into the formula in Theorem 5.5 and obtain simpler bounds for both the MCP and the MSSP.

For the rest of this chapter, let $d = \mathfrak{d}(\mathbf{A})$. Let n be ξ's degree, and let $g(x) = x^n + \sum_{i=0}^{n-1} c_i x^i$ be ξ's minimal polynomial. Let C be a positive integer such that $|c_i| \le C$ for all i. Lastly, for real polynomial $\mathbf{P}$, we define the ∞-*norm* of $\mathbf{P}$, denoted by $\|\mathbf{P}\|_\infty$, as $\max_{0\le i\le\mathfrak{d}(\mathbf{P})} |\mathbf{P}[i]|$. That is, $\|\mathbf{P}\|_\infty$ is the absolute value of the coefficient of $\mathbf{P}$ that is farthest away from zero. Obviously, we can and will take M to be the maximum ∞-norm of polynomials intermediate generated by $\textsc{Euclidean}(\mathbf{A}, \mathbf{B})$.

Theorem 5.6. *If all coefficients of* $\mathbf{A}$ *and* $\mathbf{B}$ *belong to* $\mathbb{Z}[\xi]\langle\langle N \rangle\rangle$*, then the MCP and the MSSP of* $\textsc{Euclidean}(\mathbf{A}, \mathbf{B})$ *is* $\widetilde{O}(d^2 \log N)$.

Proof. First, we find an upper bound on M. Consider (5.5). By Proposition 5.1, we have that $\mathbf{U}_j$, $\mathbf{V}_{k-1}$, $\mathbf{U}_{k-1}$, $\mathbf{V}_j$, $\mathbf{S}_j$, $\mathbf{S}_{j-1}$, and $\mathbf{S}_k$ are polynomials whose degrees are at most d and whose coefficients are determinants of submatrices of $\mathrm{Syl}(\mathbf{A}, \mathbf{B})$. Thus, Lemma A.9 implies that any coefficient of the four polynomials are members of $\mathbb{Z}[\xi]\langle\langle (2C)^{O(nd)}(ndN)^{O(d)} \rangle\rangle$. Consequently, the ∞-norms of these polynomials are at most

$$\begin{aligned}
\sum_{k=0}^{n-1} (2C)^{O(nd)}(ndN)^{O(d)}|\xi|^k &\le (2C)^{O(nd)}(ndN)^{O(d)} \sum_{k=0}^{n-1} |\xi|^k \\
&= (2C)^{O(nd)}(ndN)^{O(d)} \left(\frac{1 - |\xi|^n}{1 - |\xi|} \right) \\
&= O\big((2C)^{O(nd)}(ndN)^{O(d)}\big)
\end{aligned}$$

Thus, we have

$$\begin{aligned}
\|s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})\|_\infty &= \big\|(-1)^{d-k}\mathbf{A}[\mathfrak{d}]^2 s_k(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)\big\|_\infty \\
&= |\mathbf{A}[\mathfrak{d}]|^2 \cdot |s_k| \cdot \|\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j\|_\infty \\
&\le |\mathbf{A}[\mathfrak{d}]|^2 \cdot |s_k| \cdot \big(\|\mathbf{U}_j\|_\infty\|\mathbf{V}_{k-1}\|_\infty + \|\mathbf{U}_{k-1}\|_\infty\|\mathbf{V}_j\|_\infty\big) \\
&\le \left(N \sum_{k=0}^{n-1} |\xi|^k\right)^2 \cdot 2\Big(O\big((2C)^{O(nd)}(ndN)^{O(d)}\big)\Big)^3 \\
&= O\big((2C)^{O(nd)}(ndN)^{O(d)}\big),
\end{aligned}$$

and, consequently,

$$\begin{aligned}
\|\mathbf{T}_r\|_\infty &= \left\| \frac{1}{s_j s_{j-1}} \Big(s_{j-1}\mathbf{S}_j - s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1 : \mathfrak{d}] \cdot \mathbf{S}_{j-1} \Big) \right\|_\infty \\
&\le \frac{1}{|s_j||s_{j-1}|} \big\| s_{j-1}\mathbf{S}_j - s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1 : \mathfrak{d}] \cdot \mathbf{S}_{j-1} \big\|_\infty \\
&\le \frac{1}{|s_j||s_{j-1}|} \Big(|s_{j-1}|\|\mathbf{S}_j\|_\infty - \big\|s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1 : \mathfrak{d}]\big\|_\infty \|\mathbf{S}_{j-1}\|_\infty \Big) \\
&\le \frac{1}{|s_j||s_{j-1}|} O\Big((2C)^{O(nd)}(ndN)^{O(d)}\Big).
\end{aligned}$$

Lemma A.10 implies that $|s_j|$ and $|s_{j-1}|$ are greater than or equal to $\frac{1}{(2C)^{O(dn^2)}(dnN)^{O(dn)}} \left(\frac{1-|\xi|}{1-|\xi|^n} \right)^n$.

Hence,

$$\begin{aligned}\|\mathbf{T}_r\|_\infty &\le \left((2C)^{O(dn^2)}(dnN)^{O(dn)}\left(\frac{1-|\xi|^n}{1-|\xi|}\right)^n\right)^2 O((2C)^{O(nd)}(ndN)^{O(d)})\\ &= O\big((2C)^{O(dn^2)}(ndN)^{O(dn)}\big).\end{aligned}$$

Therefore, $M = O\big((2C)^{O(dn^2)}(ndN)^{O(dn)}\big)$.

Next, we find a lower bound on m. We claim that every coefficient of the polynomial

$$\mathbf{P} = s_{j-1}\mathbf{S}_j - s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1:\mathfrak{d}] \cdot \mathbf{S}_{j-1}$$

is a member of the set $\mathbb{Z}[\xi]\langle\langle(2C)^{O(nd)}(ndN)^{O(d)}\rangle\rangle$. The justification of the claim is as follows: let $K = (2C)^{O(nd)}(ndN)^{O(d)}$.

1. As noted earlier, coefficients of $\mathbf{U}_j$, $\mathbf{V}_{k-1}$, $\mathbf{U}_{k-1}$, $\mathbf{V}_j$, $\mathbf{S}_{j-1}$, $\mathbf{S}_j$, and $\mathbf{S}_k$ are members of $\mathbb{Z}[\xi]\langle\langle K\rangle\rangle$. This means that s_j, s_{j-1}, and s_k are members of $\mathbb{Z}[\xi]$ as well.
2. By Lemma A.13, coefficients of $\mathbf{U}_j\mathbf{V}_{k-1}$ and $\mathbf{U}_{k-1}\mathbf{V}_j$ are members of $\mathbb{Z}[\xi]\langle\langle(2C)^{O(n)}(ndK)^{O(1)}\rangle\rangle$, which is still equal to $\mathbb{Z}[\xi]\langle\langle K\rangle\rangle$ because $(2C)^{O(n)}(ndK)^{O(1)} = K$. So, coefficients of the difference $\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j$ are members of $\mathbb{Z}[\xi]\langle\langle 2K\rangle\rangle = \mathbb{Z}[\xi]\langle\langle K\rangle\rangle$ as well. Consequently, since $\mathbf{A}[\mathfrak{d}] \in \mathbb{Z}[\xi]\langle\langle N\rangle\rangle \subseteq \mathbb{Z}[\xi]\langle\langle K\rangle\rangle$, we have that coefficients of $(-1)^{d-k}\mathbf{A}[\mathfrak{d}]^2 s_k(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)$ are members of $\mathbb{Z}[\xi]\langle\langle K^4\rangle\rangle = \mathbb{Z}[\xi]\langle\langle K\rangle\rangle$. Therefore, coefficients of $s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1:\mathfrak{d}]$ are members of $\mathbb{Z}[\xi]\langle\langle K\rangle\rangle$.
3. We can keep evaluating the set that contain all coefficients like in Item 2, and conclude that every coefficient of $\mathbf{P}$ is also a member of $\mathbb{Z}[\xi]\langle\langle K\rangle\rangle$.

By Lemma A.2, every nonzero coefficient of the polynomial has absolute value at least

$$\frac{1}{n^{O(n)}(2C)^{O(n^2)}((2C)^{O(nd)}(ndN)^{O(d)})^n}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n = \frac{1}{(2C)^{O(dn^2)}(dnN)^{O(dn)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.$$

Thus, every nonzero coefficient of $\mathbf{T}_r$ has absolute value at least

$$\begin{aligned}&\frac{1}{|s_j||s_j-1|}\cdot\frac{1}{(2C)^{O(dn^2)}(dnN)^{O(dn)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n\\ &\ge \frac{1}{O\big((2C)^{O(nd)}(ndN)^{O(d)}\big)}\cdot\frac{1}{(2C)^{O(dn^2)}(dnN)^{O(dn)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n\\ &= \frac{1}{O\big((2C)^{O(dn^2)}(ndN)^{O(dn)}\big)}\end{aligned}$$

So, we can take $m = \frac{1}{O\big((2C)^{O(dn^2)}(ndN)^{O(dn)}\big)}$.

Plugging M and m to the formula in Theorem 5.5, we have that the MCP and the MSSP of

EUCLIDEAN($\mathbf{A}, \mathbf{B}$) is

$$\widetilde{O}\bigg(d\bigg(\log O\big((2C)^{O(dn^2)}(ndN)^{O(dn)}\big) - \log \frac{1}{O\big((2C)^{O(dn^2)}(ndN)^{O(dn)}\big)}\bigg)\bigg)$$
$$= O(n^2d^2(\log N + \log d + \log C + \log n)) = \widetilde{O}(d^2 \log N)$$

as claimed. □

The bound for the $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$ case relies on the following lemma:

Lemma 5.7. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be polynomials whose coefficients are members of* $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$. *Let* S *be a square submatrix of* $\mathrm{Syl}(\mathbf{A}, \mathbf{B})$. *Then, there exists an integer* $L \leq N^{2nd}$ *such that* $L^{2d} \det S$ *is a member of* $\mathbb{Z}[\xi]\langle\langle (2CndN)^{O(nd^2)}\rangle\rangle$.

Proof. Let $L_{\mathbf{A}}$ be the least common multiple of all the rational numbers in the coefficients of $\mathbf{A}$, and let $L_{\mathbf{B}}$ be the same for $\mathbf{B}$. Let $L = L_{\mathbf{A}} L_{\mathbf{B}}$. Take $L = L_{\mathbf{A}} L_{\mathbf{B}}$. We have that $L \leq N^{2nd}$, and $L\mathbf{A}$ and $L\mathbf{B}$ are polynomials whose coefficients are members of $\mathbb{Z}[\xi]\langle\langle N^{2nd}\rangle\rangle$.

Let S be of size $\ell \times \ell$, and let S' be the corresponding submatrix of $\mathrm{Syl}(L\mathbf{A}, L\mathbf{B})$. We have that $S' = LS$, so $\det S' = L^{\ell} \det S$. Consequently, $L^{2d-\ell} \det S' = L^{2d} \det S$. Since entries of S' are members of $\mathbb{Z}[\xi]\langle\langle N^{2nd}\rangle\rangle$, we have that $\det S'$ is a member of $\mathbb{Z}[\xi]\langle\langle (2C)^{O(\ell n)}(n\ell)^{O(\ell)}N^{O(d\ell)}\rangle\rangle$ by Lemma A.9, which implies that it is also a member of $\mathbb{Z}[\xi]\langle\langle (2CndN)^{O(nd^2)}\rangle\rangle$ because $\ell \leq 2d$. Lastly, because $L^{2d-\ell} = N^{O(d^2)}$, we have that $L^{2d} \det S$ is a member of $\mathbb{Z}[\xi]\langle\langle (2CndN)^{O(nd^2)}\rangle\rangle$. □

Theorem 5.8. *If all coefficients of* $\mathbf{A}$ *and* $\mathbf{B}$ *belong to* $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$, *then the MCP and the MSSP of* EUCLIDEAN($\mathbf{A}, \mathbf{B}$) *is* $\widetilde{O}(d^3 \log N)$.

Proof. For M, we can argue that $\big\|s_{j-1}\mathbf{S}_j - s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1 : \mathfrak{d}] \cdot \mathbf{S}_{j-1}\big\|_\infty$ is of order $O\big((2C)^{O(nd)}(ndN)^{O(d)}\big)$ using the argument in the proof of Theorem 5.6. Now, Lemma A.11 implies that both $|s_j|$ and $|s_{j-1}|$ are greater than or equal to $\frac{1}{(2CndN)^{O(d^2n^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n$. Therefore, $\frac{1}{|s_j||s_{j-1}|} = O\big((2CndN)^{O(d^2n^2)}\big)$, and consequently

$$\|T_r\|_\infty = \frac{1}{|s_j||s_{j-1}|}\big\|s_{j-1}\mathbf{S}_j - s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1 : \mathfrak{d}] \cdot \mathbf{S}_{j-1}\big\|_\infty$$
$$= O\big((2CndN)^{O(d^2n^2)}\big) \cdot O\big((2C)^{O(nd)}(ndN)^{O(d)}\big)$$
$$= O\big((2CndN)^{O(d^2n^2)}\big).$$

Thus, we can take $M = O\big((2CndN)^{O(d^2n^2)}\big)$.

For m, let L be the integer in Lemma 5.7. We have that

$$\begin{aligned}
\mathbf{T}_r &= \frac{1}{s_j s_{j-1}}\Big(s_{j-1}\mathbf{S}_j - s_j \cdot \mathrm{quo}(\mathbf{R}_i, \mathbf{R}_{i+1})[r+1:\eth] \cdot \mathbf{S}_{j-1}\Big) \\
&= \frac{1}{s_j s_{j-1}}\Big(s_{j-1}\mathbf{S}_j - \big((-1)^{d-k}A[\eth]^2 s_k(\mathbf{U}_j\mathbf{V}_{k-1} - \mathbf{U}_{k-1}\mathbf{V}_j)\big)[r+1:\eth] \cdot \mathbf{S}_{j-1}\Big) \\
&= \frac{1}{s_j s_{j-1} L^{8d}}\Bigg(L^{4d}(L^{2d}s_{j-1})(L^{2d}\mathbf{S}_j) \\
&\quad - \bigg((-1)^{d-k}A[\eth]^2(L^{2d}s_k)\Big((L^{2d}\mathbf{U}_j)(L^{2d}\mathbf{V}_{k-1}) - (L^{2d}\mathbf{U}_{k-1})(L^{2d}\mathbf{V}_j)\Big)\bigg)[r+1:\eth] \cdot (L^{2d}\mathbf{S}_{j-1})\Bigg).
\end{aligned}$$

Because the coefficients of $\mathbf{U}_j$, $\mathbf{V}_{k-1}$, and so on are determinants of submatrices of $\mathrm{Syl}(\mathbf{A}, \mathbf{B})$, we have that $L^{2d}\mathbf{U}_j$, $L^{2d}\mathbf{V}_{k-1}$, and so on are polynomials whose coefficients are members of $\mathbb{Z}[\xi]\langle\langle(2CndN)^{O(nd^2)}\rangle\rangle$. Using an argument similar to that in Theorem 5.6, we can deduce that

$$\mathbf{T}_r = \frac{1}{s_j s_{j-1} L^{8d}} \cdot (\text{a polynomial whose coefficients are members of } \mathbb{Z}[\xi]\langle\langle(2CndN)^{O(nd^2)}\rangle\rangle).$$

By Lemma A.3, every coefficient of the big polynomial on the RHS has absolute value greater than or equal to

$$\frac{1}{n^{O(n)}(2C)^{O(n^2)}((2CndN)^{O(nd^2)})^n}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n = \frac{1}{(2CndN)^{O(n^2d^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.$$

Since $|s_j|$ and $|s_{j-1}|$ are of order $O\big((2C)^{O(nd)}(ndN)^{O(d)}\big)$, and $L^8 d = O(N^{O(nd^2)})$, we can take

$$\begin{aligned}
m &= \frac{1}{O\big((2C)^{O(nd)}(ndN)^{O(d)}\big) \cdot O(N^{O(nd^2)})} \cdot \frac{1}{(2CndN)^{O(n^2d^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n \\
&= \frac{1}{O((2CndN)^{O(n^2d^2)})}
\end{aligned}$$

Plugging M and m to the formula in Theorem 5.5, we have that the MCP and the MSSP of $\textsc{Euclidean}(\mathbf{A}, \mathbf{B})$ is

$$\begin{aligned}
&\tilde{O}\bigg(d\bigg(\log O\big((2CndN)^{O(d^2n^2)}\big) - \log \frac{1}{O\big((2CndN)^{O(n^2d^2)}(ndN)^{O(dn)}\big)}\bigg)\bigg) \\
&= O(n^2d^3(\log N + \log d + \log C + \log n)) \\
&= \tilde{O}(d^3 \log N)
\end{aligned}$$

as claimed. □

Chapter 6

QR-factorization Algorithm

In this chapter, we study the QR-factorization in the context of the Shirayanagi–Sweeedler stabilization technique, and find upper bounds on MCDP and the MSSP of the the QR-factorization algorithm. After we describe the QR-factorization algorithm in Section 6.1, we proceed with the approach outlined in Section 5.3, determining an upper bound on the largest numerical value that can arise during the evolution of the QR-factorization algorithm in Section 6.2 and Section 6.3. Then, we use the results of the two sections to bound the MCDP in Section 6.4, and the MSSP in Section 6.5.

6.1 The Algorithm

The algorithm first forms the *Sylvester matrix* of the input polynomials. If $\mathbf{P} = p_0+p_1x+\cdots+p_kx^k$, and $\mathbf{Q} = q_0 + q_1x + \cdots + q_\ell x^\ell$, then the Sylvester matrix of $\mathbf{P}$ and $\mathbf{Q}$, denoted by Sylvester$(\mathbf{P}, \mathbf{Q})$, is the matrix:

$$\underbrace{\begin{bmatrix}
p_k & p_{k-1} & \cdots & \cdots & p_1 & p_0 & & & \\
 & p_k & p_{k-1} & \cdots & \cdots & p_1 & p_0 & & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & & p_k & p_{k-1} & \cdots & \cdots & p_1 & p_0 \\
q_\ell & q_{\ell-1} & \cdots & \cdots & q_1 & q_0 & & & \\
 & q_\ell & q_{\ell-1} & \cdots & \cdots & q_1 & q_1 & & \\
 & & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\
 & & & q_\ell & q_{\ell-1} & \cdots & \cdots & q_1 & q_0
\end{bmatrix}}_{k+\ell}
\begin{matrix} \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}\ell \\ \left.\vphantom{\begin{matrix}a\\a\\a\\a\end{matrix}}\right\}k \end{matrix}$$

(the Sylvester matrix is very similar to the Sylvester–Habicht matrix, but not exactly the same).

The algorithm then performs QR-factorization on the Sylvester matrix, and the last nonzero row of the upper triangular matrix R gives the coefficients of the GCD. Proofs of correctness of this algorithm can be found in [3] and [7]. The pseudocode of the algorithm for finding GCD is given in Figure 6-1.

```
QR-GCD($\mathbf{A},\mathbf{B}$)
1  $A \leftarrow$ Sylvester($\mathbf{A},\mathbf{B}$)
2  $R \leftarrow$ HOUSEHOLDER-QR($A$)
3  $i \leftarrow \eth(\mathbf{A}) - \eth(\mathbf{B})$
4  while $R[i,i] = 0$
5        do $i \leftarrow i - 1$
6  return $\sum_{j=i}^{\eth(\mathbf{A})+\eth(\mathbf{B})-i+1} R[i,j]x^{\eth(\mathbf{A})+\eth(B)-1-j}$
```

Figure 6-1: Pseudocode of the main algorithm.

Our QR-factorization algorithm is a slight variation of the Householder QR algorithm given in [5]. Unlike the algorithm in [5], our HOUSEHOLDER effectively produces a permutation matrix P that swaps Row j with a row under it, and a Householder reflection Q that zeroes out elements under the main diagonal in Column j of PA. Moreover, elements on the main diagonal of the resulting triangular matrix R can either be positive or negative, but the algorithm in [5] always produces R such that all of its diagonal entries are nonnegative. Clearly, our version of HOUSEHOLDER still gives a correct QR-factorization although the factorization might be from those given by other implementations. The pseudocode of the QR-factorization is given in Figure 6-2.

The QR-factorization algorithm is degree-aware because the degree of the GCD determines the numbers of iterations it has to go through the loop on Line 5 to 7 of QR-GCD. Therefore, its MCP is not computable.

6.2 Error Analysis of Householder

The undefined variables in the following lemma are local variables of HOUSEHOLDER.

Lemma 6.1. *For any j, let $\mathbf{x}$ denote the vector $A[j \cdots m, j]$. If $\lfloor A[i,j] \rceil \leq \varepsilon \|\mathbf{x}\|$ for all $j \leq i \leq m$ and for some constant ε such that $2^{-\tau} < \varepsilon < 1/\Theta(m^2)$, then, after* HOUSEHOLDER *runs to completion, the followings are true:*

(i) $\lfloor \mathbf{v}[i] \rceil = O(m\varepsilon\|\mathbf{v}\|)$ for all $1 \leq i \leq m-j+1$, and
(ii) $\lfloor \beta \rceil = O(m^2\varepsilon\beta)$.

Proof. Since $\sigma_1 = \|\mathbf{x}\|^2 = \mathbf{x}\cdot\mathbf{x}$, we have $\lfloor \sigma_1 \rceil = O(m\varepsilon\|\mathbf{x}\|^2)$ by Lemma B.9.

If $\|\mathbf{x}\| = 0$, then HOUSEHOLDER returns $\beta = 0$, and there is nothing to analyze in this case. If $\|\mathbf{x}\| \neq 0$, then the comparison $[\![\sigma_1]\!] \approxeq 0$ in Line 2 of the stabilized algorithm will always results in branching to Line 4. The reason is that $\varepsilon < 1/\Theta(m^2)$, so $\lfloor \sigma_1 \rceil < \frac{O(m)}{\Theta(m^2)}\|\mathbf{x}\|^2 < \frac{1}{2}\|\mathbf{x}\|^2$ if m is

```
HOUSEHOLDER-QR($A$)
     ▷ $A$ is a matrix of dimension $m \times m$.
 1   $A_1 \leftarrow A$
 2   for $j \leftarrow 1$ to $m$
 3        do $(\mathbf{v}_j, \beta_j) \leftarrow$ HOUSEHOLDER($A_j, j$)
 4           $A_{j+1} \leftarrow A_j$
 5           if $\beta_j = 0$
 6              then continue
 7           for $i \leftarrow j$ to $m$
 8                do $c \leftarrow 0$
 9                   for $k \leftarrow j$ to $m$
10                        do $c \leftarrow c + \mathbf{v}_j[k-j+1]A_j[k,i]$
11                   for $k \leftarrow j$ to $m$
12                        do $A_{j+1}[k,i] \leftarrow A_j[k,i]-$
                                 $2\beta_j c\mathbf{v}_j[k-j+1]$
13   return $A_n$

HOUSEHOLDER($A, j$)
     ▷ $A$ is a matrix of dimension $m \times m$, and $1 \le j \le m$.
 1   $\sigma_1 \leftarrow \sum_{i=j}^{m} A[i,j]^2$
 2   if $\sigma_1 = 0$
 3      then $\beta \leftarrow 0$
 4      else for $k \leftarrow j$ to $m$
 5                do if $A[k,j] \neq 0$
 6                      then break
 7           for $i \leftarrow j$ to $m$
 8                do Swap $A[j,i]$ with $A[k,i]$.
 9           $\mu \leftarrow \sqrt{\sigma_1}$
10           $\mathbf{v} \leftarrow A[j \cdots m, j]$
11           if $A[j,j] < 0$
12              then $\mathbf{v}[1] \leftarrow A[j,j] - \mu$
13              else $\mathbf{v}[1] \leftarrow A[j,j] + \mu$
14           $\sigma_2 \leftarrow \|\mathbf{v}\|^2$
15           $\beta \leftarrow 1/\sigma_2$
16   return $(\mathbf{v}, \beta)$
```

Figure 6-2: Pseudocode of the QR-factorization algorithm.

sufficiently large. Therefore, $\langle\sigma_1\rangle \ge \|\mathbf{x}\|^2 - \lfloor\sigma_1\rceil \ge \|\mathbf{x}\|^2/2 > \lfloor\sigma_1\rceil$, and the faulty branching does not occur.

We next bound the error term of the output in case $\|\mathbf{x}\| \neq 0$. The loop in Line 4 to Line 6 is guaranteed to find one value of k such that $\llbracket A[k,j]\rrbracket \ncong 0$. The reason is that, if $\|\mathbf{x}\| \neq 0$, then not all of $A[k,j]$ can have absolute value less than $\|\mathbf{x}\|/\sqrt{m}$. Let k be an integer such that $k > j$ and $|A[k,j]| \ge \|\mathbf{x}\|/\sqrt{m}$. Then,

$$|\langle A[k,j]\rangle| \ge |A[k,j]| - \lfloor A[k,j]\rceil > \frac{\|\mathbf{x}\|}{\sqrt{m}} - \varepsilon\|\mathbf{x}\| = \frac{\|\mathbf{x}\|}{\sqrt{m}} - \frac{\|\mathbf{x}\|}{\Theta(m)^2} > \varepsilon\|\mathbf{x}\| \ge \lfloor A[k,j]\rceil.$$

Note, though, that the stabilized version of HOUSEHOLDER might not find the same k as the exact algorithm. However, we do not care whether it is the case or not.

By Lemma B.4, we have that $\lfloor \mu \rceil = O(\lfloor \sigma_1 \rceil / \|\mathbf{x}\|) = O(m\varepsilon\|\mathbf{x}\|)$. So, after Line 13, by Lemma B.2,

$$\lfloor \mathbf{v}[1] \rceil = O(\lfloor A[j,j] \rceil + \lfloor \mu \rceil + \varepsilon|\mathbf{v}[1]|) = O(\varepsilon\|\mathbf{v}\| + m\varepsilon\|\mathbf{v}\| + \varepsilon\|\mathbf{v}\|) = O(m\varepsilon\|\mathbf{v}\|).$$

Similarly, for any $2 \leq i \leq n-j+1$, we have $\lfloor \mathbf{v}[i] \rceil \leq \varepsilon\|\mathbf{x}\| = O(m\varepsilon\|\mathbf{v}\|)$. We have proven (i).

Thus, by Lemma B.9, $\lfloor \sigma_2 \rceil = O(m(m\varepsilon)\|\mathbf{v}\|^2) = O(m^2\varepsilon\sigma_2)$. Since $\varepsilon < 1/\Theta(m^2)$, we can choose a constant for the $\Theta(m^2)$ high enough so that $\lfloor \sigma_2 \rceil < \sigma_2/4$. So,

$$\begin{aligned}
\lfloor \beta \rceil &= \mathrm{up}_\tau\left(\frac{\lfloor \sigma_2 \rceil}{\langle \sigma_2 \rangle(\langle \sigma_2 \rangle - \lfloor \sigma_2 \rceil)} + \epsilon_\tau(\langle \beta \rangle)\right) \\
&\leq (1+2^{-\tau+1})\left(\frac{\lfloor \sigma_2 \rceil}{(\sigma_2 - \lfloor \sigma_2 \rceil)(\sigma_2 - 2\lfloor \sigma_2 \rceil)} + 2(1+2^{-\tau})\frac{1}{\sigma_2}\right) \\
&\leq (1+2^{-\tau+1})\left(\frac{O(m^2\varepsilon\sigma_2)}{(\sigma_2/2)(\sigma_2/2)} + O(\varepsilon)\frac{1}{\sigma_2}\right) \\
&= O(m^2\varepsilon(\sigma_2)^{-1}) = O(m^2\varepsilon\beta).
\end{aligned}$$

We have proven (ii). □

6.3 Error Analysis of Householder-QR

We now analyze the growth of the error terms of variables involved in HOUSEHOLDER-QR. The variables in the following lemmas are local variables of HOUSEHOULDER.

Lemma 6.2. *Let $M = \max\{1, \max_{k,\ell}\{|A_1[k,\ell]|\}\}$. Let j be an integer between 1 and m, inclusive. Let $\mathbf{x}_j = A[j \cdots m, j]$. Let ε be a constant such that $2^{-\tau} < \varepsilon < 1$, and such that $\lfloor A_j[k,\ell] \rceil \leq \varepsilon\|\mathbf{x}_j\|$ for all k, ℓ. Then, $\lfloor A_{j+1}[k,\ell] \rceil \leq O(m^3 M\varepsilon)$ for all k, ℓ.*

Proof. Consider Line 3 of HOUSEHOLDER-QR. If $\beta_j = 0$, then no elements of A_{j+1} are different from the corresponding elements of A_j. No error terms increase, and we are done.

If $\beta_j \neq 0$, then the loop from Line 7 to Line 12 reflects each column of A_j by the Householder reflection calculated by HOUSEHOLDER. Consider a particular iteration of this loop. The value c calculated by the loop in Line 9 to 10 is the dot product of $A_j[j \cdots m, j]$ and $\mathbf{v}_j$. Let us denote $A_j[j \cdots m, i]$ by $\mathbf{u}$. Then, using Lemma B.7 and, we have

$$\begin{aligned}
\lfloor c \rceil &= O(\lfloor \mathbf{u} \rceil \cdot \lfloor \mathbf{v}_j \rceil + |\mathbf{u}| \cdot \lfloor \mathbf{v}_j \rceil + |\mathbf{v}_j| \cdot \lfloor \mathbf{u} \rceil + 2^{1-\tau} m|\mathbf{u}| \cdot |\mathbf{v}_j|) \\
&= O(\lfloor \mathbf{u} \rceil \cdot \lfloor \mathbf{v}_j \rceil + |\mathbf{u}|\lfloor \mathbf{v}_j \rceil + |\mathbf{v}_j|\lfloor \mathbf{u} \rceil + \varepsilon m\|\mathbf{u}\|\|\mathbf{v}_j\|).
\end{aligned}$$

Since each element of $\lfloor \mathbf{u} \rceil$ is bounded above by $\varepsilon\|\mathbf{x}_j\| \leq \varepsilon\|\mathbf{v}_j\|$, and Lemma 6.1 implies that each element of $\lfloor \mathbf{v}_j \rceil$ is $O(m\varepsilon\|\mathbf{v}_j\|)$, we have that

$$\lfloor \mathbf{u} \rceil \cdot \lfloor \mathbf{v}_j \rceil = \sum_{i=1}^{m-j+1} \lfloor \mathbf{u}[i] \rceil \lfloor \mathbf{v}[i] \rceil \le m(\varepsilon\|\mathbf{v}_j\|)O(m\varepsilon\|\mathbf{v}_j\|) = O(m^2\varepsilon^2\|\mathbf{v}_j\|^2),$$

and

$$\lfloor c \rceil = O\Big(m^2\varepsilon^2\|\mathbf{v}_j\|^2 + \|\mathbf{u}\|_1(m\varepsilon\|\mathbf{v}_j\|) + \|\mathbf{v}_j\|_1(\varepsilon\|\mathbf{v}_j\|) + \varepsilon m\|\mathbf{u}\|\|\mathbf{v}_j\|\Big).$$

Because $A_j = QA_1$ for some orthogonal matrix Q, the length of $A_j[*,i]$ is equal to the length of $A_1[*,i]$. Therefore, $\|\mathbf{u}\|_1 \le \sqrt{m}\|\mathbf{u}\| \le \sqrt{m}\|A_j[*,i]\| = \sqrt{m}\|A_1[*,i]\| \le mM$. Similarly, $\|\mathbf{v}_j\|_1 \le mM$. Hence,

$$\begin{aligned}\lfloor c \rceil &= O(m^2\varepsilon^2\|\mathbf{v}_j\|^2 + m^2M\varepsilon\|\mathbf{v}_j\| + M\varepsilon\|\mathbf{v}_j\| + m\sqrt{m}M\varepsilon\|\mathbf{v}_j\|)\\ &= O(m^2\varepsilon^2\|\mathbf{v}_j\|^2 + m^2M\varepsilon\|\mathbf{v}_j\|).\end{aligned}$$

Since $\|\mathbf{v}_j\| \le \sqrt{m}M$, and $\varepsilon < 1/\Theta(m^2)$, we have $\lfloor c \rceil = O(m\sqrt{m}M\varepsilon\|\mathbf{v}_j\| + m^2M\varepsilon\|\mathbf{v}_j\|) = O(m^2M\varepsilon\|\mathbf{v}_j\|)$. Consequently, by the fact that $c = \mathbf{u}\cdot\mathbf{v}_j \le \sqrt{m}M\|\mathbf{v}_j\|$ and Lemma B.3,

$$\begin{aligned}\lfloor \beta_j c \rceil &= O(\lfloor \beta_j \rceil \lfloor c \rceil + |\beta_j|\lfloor c \rceil + \lfloor \beta_j \rceil |c| + \varepsilon|\beta_j||c|)\\ &= O((m^2\varepsilon\beta_j)(mM\varepsilon\|\mathbf{v}_j\|) + \beta_j(mM\varepsilon\|\mathbf{v}_j\|) + (m^2\varepsilon\beta_j)(\sqrt{m}M\|\mathbf{v}_j\|) + \varepsilon\beta_j(\sqrt{m}M\|\mathbf{v}_j\|))\\ &= O(m^3M\varepsilon\beta_j\|\mathbf{v}_j\|),\end{aligned}$$

and so $\lfloor 2\beta_j c \rceil = O(m^3M\varepsilon\beta_j\|\mathbf{v}_j\|)$ as well.

Let $\pi = 2\beta_j c$. For k such that $j \le k \le m$, let $\varphi_k = \mathbf{v}_j[k-j+1]$. We have

$$\begin{aligned}\lfloor \pi\varphi_k \rceil &= O(\lfloor \pi \rceil \lfloor \varphi_k \rceil + |\pi|\lfloor \varphi_k \rceil + |\varphi_k|\lfloor \pi \rceil + \varepsilon|\pi||\varphi_k|)\\ &= O((m\varepsilon\|\mathbf{v}_j\|)(m^3M\varepsilon\beta_j\|\mathbf{v}_j\|) + (\beta_j\sqrt{m}M\|\mathbf{v}_j\|)(m\varepsilon\|\mathbf{v}_j\|)\\ &\qquad + \|\mathbf{v}_j\|(m^3M\varepsilon\beta_j\|\mathbf{v}_j\|) + \varepsilon(\beta_j\sqrt{m}M\|\mathbf{v}_j\|)\|\mathbf{v}_j\|)\\ &= O((m^4\varepsilon + m^3)M\varepsilon\beta_j\|\mathbf{v}_j\|^2) = O(m^3M\varepsilon\beta_j\|\mathbf{v}_j\|) = O(m^3M\varepsilon).\end{aligned}$$

Lastly, consider the calculation in Line 12. We have that, for any $j \le k \le m$ and for any $j \le i \le m$,

$$\begin{aligned}\lfloor A_{j+1}[k,i] \rceil &= O(\lfloor A_j[k,i] \rceil + \lfloor \pi\varphi_k \rceil + \varepsilon|A_{j+1}[k,i]|) = O(\varepsilon\|\mathbf{x}_j\| + m^3M\varepsilon + \varepsilon\sqrt{m}M)\\ &= O(m^3M\varepsilon)\end{aligned}$$

as required. □

In other words, the error term of every element in the submatrix A_j increases by at most a factor of $\frac{O(m^3 M)}{\|\mathbf{x}_j\|^2}$.

Corollary 6.3. *Let γ be such that $\lfloor A_1[k,\ell]\rceil \leq \gamma$ for any k,ℓ. Let j be an integer such that $1 \leq j \leq m+1$, and $\|\mathbf{x}_j\| \neq 0$. Let ε be such that $2^{-\tau} < \varepsilon < 1/\Theta(m^2)$. Let $\mathbf{x}_i$ denote $A_i[i \cdots m, i]$. If, for all i such that $1 \leq i < j$ and $\|\mathbf{x}_i\| \neq 0$, it is true that $\lfloor A_i[k,\ell]\rceil \leq \varepsilon\|\mathbf{x}_i\|$, then*

$$\lfloor A_j[k,\ell]\rceil \leq \frac{O(m^3 M)^{j-1}}{\prod_{\substack{1\leq i<j \\ \|\mathbf{x}_i\|\neq 0}} \|\mathbf{x}_i\|} \cdot \gamma$$

for all k,ℓ.

Proof. By induction on j. □

6.4 The MCDP

Consider the procedure QR-GCD. QR-GCD gives a polynomial whose degree is less than the exact GCD if $\llbracket R[i,i]\rrbracket$ is not equivalent to zero for some $i \geq \partial(\mathbf{A}) + \partial(\mathbf{B}) - \partial(\gcd(\mathbf{A},\mathbf{B}))$. The following lemma states that this cannot happen if Householder does not branch off incorrectly at the **if** statement in Line 2.

Lemma 6.4. *Let $\mathbf{A}$ and $\mathbf{B}$ be real polynomials. Let $A = \mathrm{Sylvester}(\mathbf{A},\mathbf{B})$, and let r be the rank of A. If, in the first r times* Householder *is called during the execution of the stabilized version of* Householder-QR *on $\llbracket A\rrbracket$, it is true that* Householder *never decides that $\llbracket\sigma_1\rrbracket \approxeq 0$ at the **if** statement on Line 2, then all elements in all rows below Row r in the output of* Householder-QR *are equivalent to zero.*

Proof. The Sylvester matrix A has the property that, if $A = QR$ is the QR-factorization of A, then $R[1,1], R[2,2], \ldots, R[r,r]$ are not zero, and the remaining $\partial(A) + \partial(B) - r$ rows below are all zero rows [3]. Hence, the exact computation must apply r orthogonal transformations to A in the first r iterations of the main loop of Householder-QR. After that, the exact computation will not apply any other transformations, and the return value is equal to A_{r+1}.

Assume then that the stabilized Householder never decides $\llbracket\sigma_1\rrbracket \approxeq 0$ in the first r times it is called. The assumption implies that the stabilized version also applies a transformation to A in each of the first r iterations of the main loop as well. It follows that any elements in Row $r+1$ to Row $\partial(\mathbf{A})+\partial(\mathbf{B})$ of A_{r+1} must be equivalent to zero because r transformations are applied just like in the exact algorithm. After that, the stabilized algorithm will also not apply any more transformation, and return the matrix $\llbracket A_{r+1}\rrbracket$. □

Note that the above lemma only requires that the stabilized algorithm does not branch incorrectly at the **if** statement in Line 2 of Householder. Whether it branches like the exact computation does at other **if** statements (such as those in Line 5 or Line 11) or not does not matter at all. Thus, in order to bound the MCDP, it is sufficient to find a bound on the precision at and beyond which no faulty branchings at the **if** statement in Line 2 occur.

Theorem 6.5. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two real polynomials whose coefficients have absolute values not larger than* $M \in \mathbb{R}$. *Let* $d = \partial(A) + \partial(B)$, *and let* $A = \mathrm{Sylvester}(\mathbf{A}, \mathbf{B})$ *be the Sylvester matrix of the two polynomials. Let* r *be the rank of* A, *and let* $A = QR$ *be the QR-factorization of* A *computed by* Householder-QR. *If*

$$\tau = \Omega\left(d(\log d + \log M) - \sum_{i=1}^{r} \log |R[i,i]| \right),$$

then the matrix $[\![R]\!]$ *returned by the stabilized* Householder-QR *with input* $[\![A]\!]_\tau$ *has the following properties:*

(a) $[\![R]\!]$ *is upper triangular, and Row* r *is the last nonzero row of* R.
(b) $\lim_{\tau \to \infty} \sum_{i=0}^{d-r} \langle R[r, d-i] \rangle x^i = R[r,r] \gcd(\mathbf{A}, \mathbf{B})$. *(Here,* $\gcd(\mathbf{A}, \mathbf{B})$ *is monic.)*

In other words, the MCDP of the QR-factorization algorithm is $O\Big(d(\log d + \log M) - \sum_{i=1}^{r} \log |R[i,i]|\Big)$.

The proof of the theorem relies on the following lemma:

Lemma 6.6. *There exists a constant* C *such that, for any* j *such that* $1 \le j \le r$, *if*

$$2^{-\tau} \le \lambda \frac{\left(\prod_{i=1}^{r} |R[i,i]|\right)^2}{\Theta(d^2 (C d^5 M^3)^r)},$$

then

$$\lfloor A_j[k,\ell] \rceil \le \lambda \frac{|R[j,j]|}{\Theta(d^2 (C d^3 M)^{r-j+1})}$$

for any constant λ, *and for any* $1 \le k, \ell \le d$.

Proof. We choose C to be the the constant of the function $O(m^3 M)$ in Corollary 6.3. Note that, in Corollary 6.3, $\|\mathbf{x}_i\| = |R[i,i]|$ for all i, and $\prod_{\substack{1 \le i < j \\ \|\mathbf{x}_i\| \ne 0}} \|\mathbf{x}_i\| = \prod_{i=1}^{j-1} R[i,i]$ because the first r diagonal entries are nonzero.

The proof is by strong induction on j. For the base case, $j = 1$, we have that

$$\begin{aligned}
\lfloor A_1[k,\ell] \rceil &= \epsilon_\tau(\langle A_1[k,\ell] \rangle) \le 2^{-\tau}(1 + 2^{-\tau}) A_1[k,\ell] \\
&\le \lambda \frac{(\prod_{i=1}^{r} |R[i,i]|)^2}{\Theta(d^2 (C d^5 M^3)^r)} 2M \\
&= \lambda \frac{R[1,1]}{\Theta(d^2 (C d^3 M)^r)} \frac{\prod_{i=1}^{r} |R[i,i]|}{(dM)^r} \frac{\prod_{i=2}^{r} |R[i,i]|}{(dM)^{r-1}} \frac{2M}{dM} \\
&\le \lambda \frac{R[1,1]}{\Theta(d^2 (C d^3 M)^{r-1+1})}.
\end{aligned}$$

The last line follows from the fact that $|R[i,i]| \le \|A[*,i]\| \le dM$. The base case is established.

Suppose, for some $1 \leq j < r$, the claim is true for all i such that $1 \leq i \leq j$. We have that, for all such i,

$$\lfloor A_i[k,\ell] \rceil \leq \lambda \frac{R[i,i]}{\Theta(d^2(Cd^3M)^{r-i+1})}.$$

Setting $\gamma = 1/\Theta(d^2(Cd^3M)^{r-i+1})$, we have that $\varepsilon < 1/\Theta(d^2)$. Hence, by Corollary 6.3,

$$\begin{aligned}
\lfloor A_{j+1}[k,\ell] \rceil &\leq \lambda \frac{(Cd^3M)^j}{\prod_{i=1}^{j} |R[i,i]|} \cdot \lfloor A_1[k,\ell] \rceil \\
&\leq \lambda \frac{(Cd^3M)^j}{\prod_{i=1}^{j} |R[i,i]|} \cdot \frac{(\prod_{i=1}^{r} |R[i,i]|)^2}{\Theta(d^2(Cd^5M^3)^r)} \\
&\leq \lambda \frac{(Cd^3M)^j}{\Theta(d^2(Cd^3M)^r} \cdot |R[j+1,j+1]| \cdot \frac{\prod_{i \neq j+1} |R[i,i]|}{(dM)^{2r}} \\
&\leq \lambda \frac{|R[j+1,j+1]|}{\Theta(d^2(Cd^3M^1)^{r-j}}.
\end{aligned}$$

Thus, the lemma is true for all $1 \leq j \leq r$. □

Proof. (Theorem 6.5) We set τ so that $2^{-\tau}$ is less than the quantity specified in Lemma 6.6 with $\lambda = |R[r,r]|/(dM)$. With this, $\tau = O\Big(d(\log d + \log M) - \sum_{i=1}^{r} \log |R[i,i]|\Big)$. We have that no faulty branching at the **if** statement in Line 2 of HOUSEHOLDER can take place because $\lfloor A_j[k,\ell] \rceil \leq \frac{|R[j,j]|}{\Theta(d^2)}$ for all $1 \leq j \leq r$. This implies that r transformations are applied, and all elements below $\llbracket R[1,1] \rrbracket$, $\llbracket R[2,2] \rrbracket$, $\ldots$, $\llbracket R[r,r] \rrbracket$ are equivalent to zero. Moreover, Lemma 6.4 implies that all rows of $\llbracket R \rrbracket$ below Row r are zero rows. Therefore, $\llbracket R \rrbracket$ is upper triangular.

Lemma 6.6 and Corollary 6.3 together imply that $\llbracket R[r,r] \rrbracket$ is not zero because $\lfloor R[r,r] \rceil < \lambda \cdot \frac{|R[r,r]|}{\Theta(d^2(Cd^3M))} \cdot \frac{Cd^3M}{|R[r,r]|} < \frac{|R[r,r]|}{\Theta(d^3M)}$. So, Row r is the last nonzero row of R. We have proven (a).

In the exact algorithm, the polynomial $\sum_{i=0}^{d-r} R[r,d-i]x^i$ is similar to the GCD of $\mathbf{A}$ and $\mathbf{B}$. Therefore,

$$\sum_{i=0}^{d-r} A_{d+1}[r,d-i]x^i = R[r,r]\gcd(\mathbf{A},\mathbf{B}).$$

The Shirayanagi–Sweedler stabilization technique then implies that Row r of $\llbracket R \rrbracket$ will converge to Row r of R, and (b) follows as a result. □

The theorem states that, if τ is greater than the value indicated, then the QR-factorization of the Sylvester matrix will yield a polynomial with the same degree as the GCD of the two input polynomials. Moreover, the absolute error of every term is less than $\frac{|R[r,r]|}{\Theta(d^3M)}$. This means that the leading coefficient of the GCD given by the stabilized algorithm is a good approximation of the leading coefficient of the exact GCD.

6.5 The MSSP

The output of QR-GCD($[\![\mathbf{A}]\!]_\tau$, $[\![\mathbf{B}]\!]_\tau$) has the same support as $\gcd(\mathbf{A},\mathbf{B})$ if and only if, for all entries $R[r,i]$ such that $R[r,i] \neq 0$, we have that $[\![R[r,i]]\!]$ is not equivalent to zero. We can use Lemma 6.6 to bound the MSSP in terms of the smallest non-zero coefficient of the GCD as follows:

Theorem 6.7. *Let μ be the non-zero coefficient of* $\gcd(\mathbf{A},\mathbf{B})$ *with the smallest absolute value. Then, the MSSP of the QR-factorization is* $O\big(d(\log d + \log M) - \log|\mu| - \sum_{i=1}^{r} \log|R[i,i]|\big)$.

Proof. We use Lemma 6.6 with $\lambda = |R[r,r]\mu|$. This choice of λ gives $\tau = O\big(d(\log d + \log M) - \log|\mu| - \sum_{i=1}^{r} \log|R[i,i]|\big)$ as claimed. By Lemma 6.6, we have that

$$\lfloor R[r,i] \rceil < |R[r,r]\mu| \cdot \frac{|R[r,r]|}{\Theta(d^2(Cd^3M))} \cdot \frac{Cd^3M}{|R[r,r]|} = \frac{|R[r,r]\mu|}{\Theta(d^3M)},$$

for all i. So, if the coefficient of x^k of $\gcd(\mathbf{A},\mathbf{B}) \neq 0$, then

$$\begin{aligned} \lfloor R[r,d-k] \rceil &\le \frac{|R[r,r]\mu|}{\Theta(d^3M)} \le \frac{|R[r,r]\gcd(\mathbf{A},\mathbf{B})[k]|}{\Theta(d^3M)} \\ &\le \frac{|R[r,d-k]|}{\Theta(d^3M)} < |\langle R[r,d-k] \rangle|. \end{aligned}$$

if d is sufficiently large. Therefore, no non-zero coefficients are rewritten to zero. □

Appendix A

Bounds on Algebraic Numbers

In this appendix, we prove a number of statements giving lower bounds on the absolute values of algebraic numbers and determinants of matrices whose entries are algebraic numbers. These bounds facilitate the study of the MCP of the Euclidean algorithm in Chapter 5.

Throughout this chapter, we let $\xi \neq 1$ be a real algebraic number of degree n, and let N be a positive integer. Additionally, let $g(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1 x + c_0$ be its minimal polynomial, and let C be an integer such that $C \geq |c_i|$ for all $0 \leq i < n$.

A.1 Elements of $\mathbb{Z}[\xi]\langle\langle N\rangle\rangle$ and $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$

Lemma A.1. *If $a_0, a_1, \ldots, a_{n-1} \in \mathbb{Z}$ and $\xi^k = a_0 + a_1\xi + \ldots + a_{n-1}\xi^{n-1}$, then, for all i and for all $k \geq n$, we have $|a_i| \leq 2^{k-n}C^{k-n+1}$ for all i.*

Proof. The proof is by induction on k. For the base case, $k = n$, the claim is trivially true. Assume now that the claim is true for some $k \geq n$. Consider

$$\begin{aligned}\xi^{k+1} &= \xi^k \times \xi = (a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1}) \times \xi = a_0\xi + a_1\xi + \cdots + a_{n-2}\xi^{n-1} + a_{n-1}\xi^n \\ &= a_0\xi + a_1\xi + \cdots + a_{n-2}\xi^{n-1} - a_{n-1}(c_0 + c_1\xi + \cdots + c_{n-1}\xi^{n-1}) \\ &= -a_{n-1}c_0 + (a_0 - a_{n-1}c_1)\xi + \cdots + (a_{n-2} - a_{n-1}c_{n-1})\xi^{n-1}.\end{aligned}$$

We have that $|a_{n-1}c_0| \leq (2^{k-n}C^{n-k+1})C \leq 2^{k-n+1}C^{k-n+2}$. Moreover, for any $1 \leq i \leq n-1$,

$$|a_{i-1} - a_i c_i| \leq |a_{i-1}| + |a_i c_i| \leq 2^{k-n}C^{n-k+1} + 2^{k-n}C^{n-k+1}C \leq 2^{k-n+1}C^{n-k+2}.$$

By induction, the lemma is true for all $k \geq n$. □

Lemma A.2. *Let $a_0, a_1, \ldots, a_{n-1} \in \langle\langle N \rangle\rangle$. If $a_0 - a_1\xi - a_2\xi^2 - \cdots - a_{n-1}\xi^{n-1} \neq 0$, then*

$$|a_0 - a_1\xi - a_2\xi^2 - \cdots - a_{n-1}\xi^{n-1}| \geq \frac{1}{(2C)^{O(n^2)}(nN)^{O(n)}} \left(\frac{1-|\xi|}{1-|\xi|^n} \right)^n.$$

Proof. Our proof follows the one outlined in Page 161 of [6].

Consider $\mathbb{Z}[\xi]$ as a module on $\mathbb{Z}$ with basis $\{1, \xi, \xi^2, \ldots, \xi^{n-1}\}$. That is, we represent the element $b_0 + b_1\xi + b_2\xi^2 + \cdots + b_{n-1}\xi^{n-1}$ as the column vector $(b_0, b_1, \ldots, b_{n-1})^T$. Let X be the matrix that represents multiplication by ξ. Then, X^k is the matrix that represents multiplication by ξ^k, and the jth column of X^k is the column vector that represents ξ^{k+j-1}. So, the last column of X^k is the column vector that represents ξ^{k+n-1}. If $k + j - 1 < n$, then each entry in the jth column of X^k is either 0 or 1. Otherwise, by Lemma A.1, each entry of the jth column is bounded above in absolute value by $2^{(k+j-1)-n}C^{(k+j-1)-n+1} \leq 2^{(k+n-1)-n}C^{(k+n-1)-n+1} = 2^{k-1}C^k$. Hence, we can say that every entry of X^k has absolute value at most $2^{k-1}C^k$.

Let $\zeta = a_1\xi + \cdots + a_{n-1}\xi^{n-1}$. We have that ζ is an algebraic integer of degree at most n. Let Y be the matrix that represents multiplication by ζ. Then, $Y = a_1X + a_2X^2 + \cdots + a_{n-1}X^{n-1}$. Thus, every entry of Y is bounded above in absolute value by $\sum_{k=1}^{n-1} 2^{k-1}C^kN \leq 2^nC^nN$. Let f be the characteristic polynomial of Y. We have that each coefficient of f has absolute value no greater than $2^{n^2}C^{n^2}N^nn! = (2C)^{O(n^2)}(nN)^{O(n)}$. Also, by the Cayley–Hamilton theorem (Proposition 2.4), $f(Y) = f(\zeta) = 0$.

Since f is an integer polynomial, we have that $|f(a_0)| \geq 1$, and, by the mean value theorem,

$$f(a_0) = f(a_0) - 0 = f(a_0) - f(\zeta) = (a_0 - \zeta)f'(x)$$

for some value of x lying between a_0 and ζ. Let M be a constant such that $|f'(x)| \leq M$ for all $\zeta - 1 \leq x \leq \zeta + 1$. Since we would like to estimate how small $|a_0 - \zeta|$ can be, it is safe to assume that a_0 is close to ζ, and that it lies in the interval $(\zeta - 1, \zeta + 1)$. Doing so, we have

$$|a_0 - \zeta| = \frac{|f(a_0)|}{|f'(x)|} \geq \frac{1}{M}.$$

It remains to find an upper bound on M.

Because every coefficient of $f(x)$ is bounded above in absolute value by $(2C)^{O(n^2)}(nN)^{O(n)}$, we have that each coefficient of $f'(x)$ is bounded above by $n \cdot (2C)^{O(n^2)}(nN)^{O(n)}$, which can still be described as $(2C)^{O(n^2)}(nN)^{O(n)}$. Moreover, for all $x \in (\zeta - 1, \zeta + 1)$, we have

$$|x| \leq 1 + |\zeta| \leq 1 + |a_1\xi + a_2\xi^2 + \cdots + a_{n-1}\xi^{n-1}| \leq N\frac{1-|\xi|^n}{1-|\xi|}.$$

Therefore,

$$\begin{aligned}M &\le \sum_{i=0}^{n-1}(2C)^{O(n^2)}(nN)^{O(n)}|\xi|^i = (2C)^{O(n^2)}(nN)^{O(n)}|\xi|^i\sum_{i=0}^{n-1}\cdot N^i\left(\frac{1-|\xi|^n}{1-|\xi|}\right)^i \\ &< (2C)^{O(n^2)}(nN)^{O(n)}\cdot N^n\left(\frac{1-|\xi|^n}{1-|\xi|}\right)^n < (2C)^{O(n^2)}(nN)^{O(n)}\left(\frac{1-|\xi|^n}{1-|\xi|}\right)^n.\end{aligned}$$

It follows that

$$|a_0-\zeta| \ge \frac{1}{(2C)^{O(n^2)}(nN)^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.$$

as claimed. □

As a result, every nonzero element of $\mathbb{Z}[\xi]\langle\langle N\rangle\rangle$ has absolute value greater than the value in Lemma A.2.

Lemma A.3. *Let $a_0, a_1, \ldots, a_{n-1} \in \mathbb{Q}\langle\langle N\rangle\rangle$. If $a_0 - a_1\xi - a_2\xi^2 - \cdots - a_{n-1}\xi^{n-1} \neq 0$, then*

$$|a_0 - a_1\xi - a_2\xi^2 - \cdots - a_{n-1}\xi^{n-1}| \ge \frac{1}{(2CN)^{O(n^2)}n^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.$$

Proof. Let ℓ be the least common multiple of the denominators of a_0, a_1, ..., a_{n-1}. Clearly, $\ell \le N^n$, and we have that the numbers ℓa_0, ℓa_1, $\cdots$, ℓa_{n-1} are integers whose absolute values do not exceed N^n. By Lemma A.2, replacing N by N^n,

$$\begin{aligned}|\ell a_0 - \ell a_1\xi - \ell a_2\xi^2 - \cdots - \ell a_{n-1}x^{n-1}| &\ge \frac{1}{(2C)^{O(n^2)}(nN^n)^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n \\ &= \frac{1}{(2CN)^{O(n^2)}n^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.\end{aligned}$$

Thus,

$$\begin{aligned}|a_0 - a_1\xi - a_2\xi^2 - \cdots - a_{n-1}x^{n-1}| &= \frac{|\ell a_0 - \ell a_1\xi - \ell a_2\xi^2 - \cdots - \ell a_{n-1}x^{n-1}|}{\ell} \\ &\ge \frac{1}{\ell(2CN)^{O(n^2)}n^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n \\ &\ge \frac{1}{N^2(2CN)^{O(n^2)}n^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n \\ &= \frac{1}{(2CN)^{O(n^2)}n^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.\end{aligned}$$

□

Therefore, every nonzero element of $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$ has absolute value not less than the value in the the above lemma.

A.2 Determinants

Definition A.4. *A row vector is said to be* pure *with respect to ξ if (1) it is an integer vector, or (2) it is of the form $\xi^k v$ for some integer k and for some integer vector v. A row vector that is not pure is called* impure.

For example, the vector $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ and $\begin{bmatrix} 2\xi & 5\xi & 6\xi \end{bmatrix}$ are pure, but the vector $\begin{bmatrix} 1+\xi & 6\xi^7 \end{bmatrix}$ and $\begin{bmatrix} \xi & \xi^2 & \xi^8 \end{bmatrix}$ are impure.

Definition A.5. *We say that a matrix A is pure if all of its rows are pure.*

For example, the matrix $\begin{bmatrix} 1 & 1 \\ \sqrt{2} & 5\sqrt{2} \end{bmatrix}$ is pure with respect to $\sqrt{2}$, while the matrix $\begin{bmatrix} 1 & \sqrt{2} \\ 3+2\sqrt{2} & 5 \end{bmatrix}$ is not. Every integer matrix is pure.

Lemma A.6. *If A is an $m \times m$ pure matrix on ξ such that whose coefficients are members of $\mathbb{Z}[\xi]\langle\langle N\rangle\rangle$, then $det(A)$ is of the form $M\xi^k$ for some integer k, and for some integer M satisfying $|M| \le m^{m/2}N^m$.*

Proof. For any row whose entries are of the form $a\xi^\ell$, we can factor out ξ^ℓ. After doing so to every row, we have that $det(A) = \xi^k det(A')$, where k is an integer, and A' is an integer matrix whose entries are members of $\langle\langle N\rangle\rangle$, so $det(A')$ is an integer. Since each column of A' has norm less than or equal to $N\sqrt{m}$, we have that $|det(A')| \le (N\sqrt{m})^m = m^{m/2}N^m$ according to the Hadamard's inequality (Proposition 2.3). □

Definition A.7. *Let A be a matrix with m rows whose entries are in $\mathbb{Z}[\xi]$. We say that A is r-mixed if the first $m - r$ rows are pure.*

The next lemma relates the determinant of any matrix with algebraic number entries to the sum of determinants of pure matrices.

Lemma A.8. *If A is an $m \times m$ r-pure matrix with entries from $\mathbb{Z}[\xi]$, then $det(A)$ can be written as a sum of determinants of m^r pure matrices.*

Proof. The proof is by induction on r. For the base case, $r = 0$, we have that the determinant of a 0-mixed matrix, a pure matrix, is the sum of determinant of $n^0 = 1$ pure matrix.

For the induction case, assume that the lemma is true for some $r \ge 0$. Let A be an $(r+1)$-mixed matrix. Then,

$$\det(A) = \left|\begin{array}{cccc} \multicolumn{4}{c}{\text{row } 1 \text{ (pure)}} \\ \multicolumn{4}{c}{\text{row } 2 \text{ (pure)}} \\ \multicolumn{4}{c}{\vdots} \\ \multicolumn{4}{c}{\text{row } m-r \text{ (pure)}} \\ \sum_{i=0}^{n-1} a_{1,i}\xi^i & \sum_{i=0}^{n-1} a_{2,i}\xi^i & \cdots & \sum_{i=0}^{n-1} a_{m,i}\xi^i \\ \multicolumn{4}{c}{\text{row } m-r+2 \text{ (impure)}} \\ \multicolumn{4}{c}{\vdots} \\ \multicolumn{4}{c}{\text{row } m \text{ (impure)}} \end{array}\right| = \sum_{i=1}^{m} \left|\begin{array}{cccc} \multicolumn{4}{c}{\text{row } 1 \text{ (pure)}} \\ \multicolumn{4}{c}{\text{row } 2 \text{ (pure)}} \\ \multicolumn{4}{c}{\vdots} \\ \multicolumn{4}{c}{\text{row } m-r \text{ (pure)}} \\ a_{1,i}\xi^i & a_{2,i}\xi^i & \cdots & a_{m,i}\xi^i \\ \multicolumn{4}{c}{\text{row m-r+2 (impure)}} \\ \multicolumn{4}{c}{\vdots} \\ \multicolumn{4}{c}{\text{row m (impure)}} \end{array}\right|$$

In other words, $\det(A)$ can be written as a sum of determinants of n r-mixed matrices. By the induction hypothesis, each r-mixed matrix's determinant can be written as a sum of determinants of m^r pure matrices. So, $\det(A)$ can be written as a sum of determinants of m^{r+1} pure matrices. By induction, the lemma is true for all r. □

Lemma A.9. *Let A be an $m \times m$ matrix with entries from $\mathbb{Z}[\xi]\langle\langle N\rangle\rangle$. If $\det(A) = \sum_{i=0}^{n-1} a_i\xi^i$, then*

$$|a_i| \leq (2C)^{O(mn)}(nmN)^{O(m)}$$

for all i. In other words, $\det A \in \mathbb{Z}[\xi]\langle\langle (2C)^{O(mn)}(nmN)^{O(m)}\rangle\rangle$.

Proof. Since A is m-mixed, by Lemma A.8, $\det(A)$ is a sum of determinants of n^m pure matrices. By Lemma A.6, any determinant of those pure matrices are of the form $M\xi^k$ for some integer M such that $|M| \leq m^{m/2}N^m = (mN)^{O(m)}$ and for some non-negative integer $k \leq m(n-1)$. By Lemma A.1, if $\xi^k = d_0 + d_1\xi + \cdots + d_{n-1}\xi^{k-1}$, then $|d_i| \leq 2^{k-n}C^{k-n+1} \leq 2^{mn-m-n}C^{mn-m-n+1} = (2C)^{O(mn)}$. Thus, if $\det(A) = a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1}$, then a_i is a sum of at most n^m integers whose absolute values do not exceed $|\xi^k||M| \leq (2C)^{O(mn)}(mN)^{O(m)}$. Therefore, $|a_i| \leq n^m(2C)^{O(mn)}(mN)^{O(m)} = (2C)^{O(mn)}(nmN)^{O(m)}$. □

Lemma A.10. *Let A be an $m \times m$ matrix with entries from $\mathbb{Z}[\xi]\langle\langle N\rangle\rangle$. If $\det(A) \neq 0$, then*

$$|\det(A)| \geq \frac{1}{(2C)^{O(mn^2)}(nmN)^{O(mn)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.$$

Proof. By Lemma A.9, we have that $\det A \in \mathbb{Z}[\xi]\langle\langle (2C)^{O(mn)}(nmN)^{O(m)}\rangle\rangle$. Applying Lemma A.2

with $N := (2C)^{O(mn)}(nmN)^{O(m)}$ gives

$$\begin{aligned}|\det(A)| &\geq \frac{1}{(2C)^{O(n^2)}(n(2C)^{O(mn)}(nmN)^{O(m)})^{O(n)}}\left(\frac{1-|\xi|}{1-|\xi|^n}{}^{n}\right)\\ &= \frac{1}{(2C)^{O(mn^2)}(nmN)^{O(mn)}}\left(\frac{1-|\xi|}{1-|\xi|^n}{}^{n}\right).\end{aligned}$$

Lemma A.11. *Let A be an $m \times m$ matrix whose entries are members of $\mathbb{Q}[\xi]\langle\langle N\rangle\rangle$. Then,*

$$|\det(A)| \geq \frac{1}{(2C)^{O(mn^2)}(nm)^{O(mn)}N^{O(n^2m^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.$$

Proof. For $1 \leq i \leq m$, let ℓ_i be the least common multiple of denominators of all rational coefficients of powers of ξ of elements in Row i of A. We have that $\ell_i \leq N^{mn}$, and

$$A' = \begin{bmatrix}\ell_1 & & & \\ & \ell_2 & & \\ & & \ddots & \\ & & & \ell_m\end{bmatrix} A$$

is a matrix whose entries are from the set $\mathbb{Z}[\xi]\langle\langle N^{mn}\rangle\rangle$. By Lemma A.10, we have

$$\begin{aligned}|\det(A')| &\geq \frac{1}{(2C)^{O(mn^2)}(nmN^{mn})^{O(mn)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n\\ &= \frac{1}{(2C)^{O(mn^2)}(nm)^{O(mn)}N^{O(n^2m^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.\end{aligned}$$

Thus,

$$\begin{aligned}|\det(A)| &= \frac{1}{\prod_{i=1}^m \ell_i}|\det(A')|\\ &\geq \frac{1}{N^{m^2n}}\cdot\frac{1}{(2C)^{O(mn^2)}(nm)^{O(mn)}N^{O(n^2m^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n\\ &= \frac{1}{(2C)^{O(mn^2)}(nm)^{O(mn)}N^{O(n^2m^2)}}\left(\frac{1-|\xi|}{1-|\xi|^n}\right)^n.\end{aligned}$$

□

A.3 Coefficients of Polynomials

Lemma A.12. *If $a, b \in \mathbb{Z}[\xi]\langle\langle N\rangle\rangle$, then $ab \in \mathbb{Z}[\xi]\langle\langle(2C)^{O(n)}(nN)^{O(1)}\rangle\rangle$.*

Proof. Let $a = a_0 + a_1\xi + \cdots + a_{n-1}\xi^{n-1}$ and $b = b_0 + b_1\xi + \cdots + b_{n-1}\xi^{n-1}$. Let $c = ab$. Then, we have

$$c = \sum_{k=0}^{2n-2} \Big(\sum_{i=0}^{k} a_i b_{k-i} \Big) \xi^k.$$

Consider the term $c'_k = \Big(\sum_{i=0}^{k} a_i b_{k-i}\Big)\xi^k$. We have that

$$\big|\sum_{i=0}^{k} a_i b_{k-i}\big| \leq \sum_{i=0}^{k} |a_i b_{k-i}| \leq (k+1)N^2.$$

So, if $k \leq n-1$, then $c'_k \in \mathbb{Z}[\xi]\langle\langle (k+1)N^2 \rangle\rangle$. Otherwise, $k \geq n$, and Lemma A.1 implies that $\xi^k \in \mathbb{Z}[\xi]\langle\langle 2^{k-n}C^{k-n+1}\rangle\rangle$, and therefore $c'_k \in \mathbb{Z}[\xi]\langle\langle (k+1)N^2 2^{k-n}C^{k-n+1}\rangle\rangle$. Thus, $c = \sum_{k=0}^{2n-2} c'_k$ is a member of $\mathbb{Z}[\xi]\langle\langle M \rangle\rangle$ where

$$\begin{aligned}
M &= \sum_{k=0}^{n-1}(k+1)N^2 + \sum_{k=n}^{2n-2}(k+1)N^2 2^{k-n}C^{k-n+1} = \sum_{k=0}^{n-1}(k+1)N^2 + \sum_{k=0}^{n-2}(n+k)N^2 2^k C^{k+1} \\
&< \sum_{k=0}^{n-1} 2nN^2 + \sum_{k=0}^{n-2} 2nN^2 2^k C^{k+1} = 2nN^2\Big(n + C\sum_{k=0}^{n-2} 2^k C^k\Big) = 2nN^2\Big(n + C\frac{2^{n-1}C^{n-1}-1}{2C-1}\Big) \\
&< 2nN^2(n + 2^{n-1}C^n).
\end{aligned}$$

Since $C \geq 1$ and $2^{n-1} \geq n$, we have that $n + 2^{n-1}C^n \leq 2^n C^n$. It follows that

$$M < 2nN^2 2^n C^n = 2^{n+1} n N^2 C^n = (2C)^{O(n)}(nN)^{O(1)}.$$

Thus, $c = ab \in \mathbb{Z}[\xi]\langle\langle (2C)^{O(n)}(nN)^{O(1)}\rangle\rangle$ as required. □

Lemma A.13. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be polynomials whose coefficients are members of* $\mathbb{Z}[\xi]\langle\langle N \rangle\rangle$. *Let* d *be an integer such that* $d \geq \partial(\mathbf{A})$ *and* $d \geq \partial(\mathbf{B})$. *Then, every coefficient of* $\mathbf{AB}$ *is a member of the set* $\mathbb{Z}[\xi]\langle\langle (2C)^{O(n)}(ndN)^{O(1)}\rangle\rangle$.

Proof. Let $\mathbf{A} = \sum_{k=0}^{d} a_k x^k$ and $\mathbf{B} = \sum_{k=0}^{d} b_k x^k$. Then, $\mathbf{AB} = \sum_{k=0}^{2d}\big(\sum_{i=0}^{k} a_i b_{k-i}\big)x^k$. Consider the coefficient of x^k for a particular k, we have that $a_i b_{k-i} \in \mathbb{Z}[\xi]\langle\langle (2C)^{O(n)}(nN)^{O(1)}\rangle\rangle$ by Lemma A.12. So, $\sum_{i=0}^{2d} a_i b_{k-i} \in \mathbb{Z}[\xi]\langle\langle k(2C)^{O(n)}(nN)^{O(1)}\rangle\rangle \subseteq \mathbb{Z}[\xi]\langle\langle (2C)^{O(n)}(ndN)^{O(1)}\rangle\rangle$ because $k \leq 2d$. □

Appendix B

Error Analysis of Basic Arithmetic and Vector Operations

In this section, we derive upper bounds for the error terms of bracket coefficients which are results of basic arithmetic and vector operations.

B.1 Addition, Subtraction, Multiplication, and Taking Inverse

Lemma B.1. *Let $\llbracket a \rrbracket$ and $\llbracket b \rrbracket$ be bracket coefficients such that $\lfloor a \rceil \geq 2^{-\tau-1}|a|$, and $\lfloor b \rceil \geq 2^{-\tau-1}|b|$ If $\tau \geq 5$, then*

(a) If $\llbracket c \rrbracket = \llbracket a \rrbracket + \llbracket b \rrbracket$, then $\lfloor c \rceil \leq 7 \max\{\lfloor a \rceil, \lfloor b \rceil\}$.

(b) If $\llbracket c \rrbracket = \llbracket a \rrbracket \llbracket b \rrbracket$ and $|a| \geq \lfloor a \rceil$, then $\lfloor c \rceil \leq 8 \max\{\lfloor a \rceil, \lfloor b \rceil\} \max\{|a|, |b|\}$.

(c) If $\lfloor a \rceil < |a|/4$ and $\llbracket c \rrbracket = 1/\llbracket a \rrbracket$, then $\lfloor c \rceil \leq 6\frac{\lfloor a \rceil}{|a|^2}$.

Proof. For addition and subtraction, we have that $\lfloor c \rceil = \mathrm{up}_\tau(\lfloor a \rceil + \lfloor b \rceil + \epsilon_\tau(\langle c \rangle))$. We also have that $\lfloor a \rceil + \lfloor b \rceil = 2 \max\{\lfloor a \rceil, \lfloor b \rceil\}$, and

$$\begin{aligned}
\epsilon_\tau(\langle c \rangle) &\leq 2^{-\tau}|\langle c \rangle| = 2^{-\tau}|\mathrm{fl}_\tau(\langle a \rangle \pm \langle b \rangle)| \leq 2^{-\tau}(1+2^{-\tau})|\langle a \rangle \pm \langle b \rangle| \\
&\leq 2^{-\tau}(1+2^{-\tau})(|a| + \lfloor a \rceil + |b| + \lfloor b \rceil) \\
&\leq (1+2^{-\tau})(2^{-\tau}|a| + 2^{-\tau}|b|) + 2^{-\tau}(1+2^{-\tau})(\lfloor a \rceil + \lfloor b \rceil) \\
&= 2(1+2^{-\tau})(\lfloor a \rceil + \lfloor b \rceil) + 2^{-\tau}(1+2^{-\tau})(\lfloor a \rceil + \lfloor b \rceil) \\
&= (1+2^{-\tau})(2+2^{-\tau})(\lfloor a \rceil + \lfloor b \rceil) \\
&\leq 2(1+2^{-\tau})(2+2^{-\tau}) \max\{\lfloor a \rceil, \lfloor b \rceil\}.
\end{aligned}$$

Thus,

$$\begin{aligned}\lfloor c\rceil &\leq (1+2^{-\tau+1})(2\max\{\lfloor a\rceil, \lfloor b\rceil\} + 2(1+2^{-\tau})(2+2^{-\tau})\max\{\lfloor a\rceil, \lfloor b\rceil\}) \\ &= 2(1+2^{-\tau+1})(1+(1+2^{-\tau})(2+2^{-\tau}))\max\{\lfloor a\rceil, \lfloor b\rceil\} \\ &\leq 7\max\{\lfloor a\rceil, \lfloor b\rceil\}\end{aligned}$$

if $\tau \geq 5$.

For multiplication, we have $\lfloor c\rceil = \mathrm{up}_\tau(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|\langle b\rangle| + \lfloor b\rceil|\langle a\rangle| + \epsilon_\tau(\langle c\rangle))$, and

$$\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|\langle b\rangle| + \lfloor b\rceil|\langle a\rangle| = \lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil(|b| + \lfloor b\rceil) + \lfloor b\rceil(|a| + \lfloor a\rceil) = 3\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a|,$$

and

$$\begin{aligned}\epsilon_\tau(\langle c\rangle) &\leq 2^{-\tau}|\mathrm{fl}_\tau(\langle a\rangle\langle b\rangle)| \leq 2^{-\tau}(1+2^{\tau})|\langle a\rangle||\langle b\rangle| \leq 2^{-\tau}(1+2^{\tau})(|a| + \lfloor a\rceil)(|b| + \lfloor b\rceil) \\ &\leq 2^{-\tau}(1+2^{-\tau})(|a||b| + |a|\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor a\rceil\lfloor b\rceil) \\ &= (1+2^{-\tau})(2^{-\tau}|a|)|b| + 2^{-\tau}(1+2^{-\tau})(|a|\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor a\rceil\lfloor b\rceil) \\ &\leq 2(1+2^{-\tau})\lfloor a\rceil|b| + 2^{-\tau}(1+2^{-\tau})(|a|\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor a\rceil\lfloor b\rceil) \\ &\leq (2+3\cdot 2^{-\tau}+2^{-2\tau})\lfloor a\rceil|b| + (2^{-\tau}+2^{-2\tau})|a|\lfloor b\rceil + (2^{-\tau}+2^{-2\tau})\lfloor a\rceil\lfloor b\rceil\end{aligned}$$

so,

$$\begin{aligned}\lfloor c\rceil &\leq (1+2^{-\tau+1})((3+3\cdot 2^{-\tau}+2^{-2\tau})\lfloor a\rceil|b| + (1+2^{-\tau}+2^{-2\tau})|a|\lfloor b\rceil + (3+2^{-\tau}+2^{-2\tau})\lfloor a\rceil\lfloor b\rceil) \\ &\leq (1+2^{-\tau+1})((3+3\cdot 2^{-\tau}+2^{-2\tau})\lfloor a\rceil|b| + (1+2^{-\tau}+2^{-2\tau})|a|\lfloor b\rceil \\ &\qquad + (3+2^{-\tau}+2^{-2\tau})|a|\lfloor b\rceil) \qquad \text{(because } |a| \geq \lfloor a\rceil\text{)} \\ &= (1+2^{-\tau+1})((3+3\cdot 2^{-\tau}+2^{-2\tau})\lfloor a\rceil|b| + (4+2\cdot 2^{-\tau}+2\cdot 2^{-2\tau})|a|\lfloor b\rceil) \\ &\leq (1+2^{-\tau+1})(7+5\cdot 2^{-\tau}+3\cdot 2^{-2\tau})\max\{|a|, |b|\}\max\{\lfloor a\rceil, \lfloor b\rceil\} \\ &\leq 8\max\{|a|, |b|\}\max\{\lfloor a\rceil, \lfloor b\rceil\}\end{aligned}$$

if $\tau \geq 5$.

For division, we have $\lfloor c\rceil = \mathrm{up}_\tau\left(\frac{\lfloor a\rceil}{|\langle a\rangle|(|\langle a\rangle| - \lfloor a\rceil)} + \epsilon_\tau(\langle c\rangle)\right)$, and

$$\frac{\lfloor a\rceil}{|\langle a\rangle|(|\langle a\rangle| - \lfloor a\rceil)} \geq \frac{\lfloor a\rceil}{(|a| - \lfloor a\rceil)(|a| - 2\lfloor a\rceil)} \geq \frac{\lfloor a\rceil}{(|a| - |a|/4)(|a| - |a|/2)} \geq \frac{8}{3|a|^2}\lfloor a\rceil,$$

and

$$\epsilon_\tau(\langle x\rangle) = \epsilon_\tau\left(\mathrm{fl}_\tau\left(\frac{1}{\langle a\rangle}\right)\right) \le 2^{-\tau}(1+2^{-\tau})\frac{1}{|\langle a\rangle|} \le 2^{-\tau}(1+2^{-\tau})\frac{1}{|a|-\lfloor a\rceil} \le 2^{-\tau}(1+2^{-\tau})\frac{1}{3|a|/4}$$
$$= 2^{-\tau}(1+2^{-\tau})\frac{4}{3|a|} = (1+2^{-\tau})\frac{4}{3|a|}\frac{2^{-\tau}|a|}{|a|} \le (1+2^{-\tau})\frac{8}{3|a|^2}\lfloor a\rceil.$$

Therefore,

$$\lfloor c\rceil = (1+2^{-\tau+1})\left(\frac{8}{3|a|^2}\lfloor a\rceil + (1+2^{-\tau})\frac{8}{3|a|^2}\right) = (1+2^{-\tau+1})(2+2^{-\tau})\frac{8}{3|a|^2}\lfloor a\rceil \le \frac{6}{|a|^2}\lfloor a\rceil$$

if $\tau \ge 5$. □

Lemma B.2. *Let $a, b \in \mathbb{R}$, and let $[\![a]\!]$ and $[\![b]\!]$ be bracket coefficients that approximate them, respectively. Let $[\![c]\!] = [\![a]\!] + [\![b]\!]$, and let ε be such that $2^{-\tau} < \varepsilon < 1$. Then, $\lfloor c\rceil = O(\lfloor a\rceil + \lfloor b\rceil + \varepsilon|c|)$.*

Proof.

$$\begin{aligned}
\lfloor c\rceil &= \mathrm{up}_\tau(\lfloor a\rceil + \lfloor b\rceil + \epsilon_\tau(\langle c\rangle)) \\
&\le (1+2^{1-\tau})(\lfloor a\rceil + \lfloor b\rceil + 2^{-\tau}(1+2^{-\tau})(|\langle a\rangle + \langle b\rangle|)) \\
&\le 2(\lfloor a\rceil + \lfloor b\rceil + 2\varepsilon(|a \pm \lfloor a\rceil + b \pm \lfloor b\rceil|)) \\
&\le 2(\lfloor a\rceil + \lfloor b\rceil + 2\varepsilon(|c| + \lfloor a\rceil + \lfloor b\rceil)) \\
&= 2(1+2\varepsilon)\lfloor a\rceil + 2(1+2\varepsilon)\lfloor b\rceil + 4\varepsilon|c| \\
&= O(\lfloor a\rceil + \lfloor b\rceil + \varepsilon|c|)
\end{aligned}$$

as claimed. □

Lemma B.3. *Let $a, b \in \mathbb{R}$, and let $[\![a]\!]$ and $[\![b]\!]$ be bracket coefficients that approximate them, respectively. Let $[\![c]\!] = [\![a]\!][\![b]\!]$, and let ε be such that $2^{-\tau} < \varepsilon < 1$. Then, $\lfloor c\rceil = O(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a| + \varepsilon|a||b|)$.*

Proof. We have that

$$\begin{aligned}
\lfloor c\rceil &= \mathrm{up}_\tau(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a| + \epsilon_\tau(\langle c\rangle)) \\
&\le (1+2^{1-\tau})(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a| + 2^{-\tau}(1+2^{-\tau})(|a| + \langle a\rangle)(|b| + \langle b\rangle)) \\
&\le 2(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a| + 2\varepsilon(|a||b| + \lfloor a\rceil|b| + \lfloor b\rceil|a| + \lfloor a\rceil\lfloor b\rceil)) \\
&= 2(1+2\varepsilon)(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a|) + 4\varepsilon|a||b| \\
&= O(\lfloor a\rceil\lfloor b\rceil + \lfloor a\rceil|b| + \lfloor b\rceil|a| + \varepsilon|a||b|)
\end{aligned}$$

as claimed. □

B.2 Square Root

Lemma B.4. *If $\tau \geq 4$, and $2^{-\tau} < \frac{\lfloor x \rceil}{x} \leq \frac{1}{8}$, and $[\![y]\!] = [\![x]\!]^{1/2}$, then $\lfloor y \rceil < 2\frac{\lfloor x \rceil}{x}\sqrt{x}$. Namely, the relative error increases by at most twice.*

Proof. First, note that

$$x + \lfloor x \rceil \leq x + \lfloor x \rceil + \frac{\lfloor x \rceil^2}{4x} = \left(\sqrt{x} + \frac{\lfloor x \rceil}{2\sqrt{x}}\right)^2 = \left(\sqrt{x}\left(1 + \frac{\lfloor x \rceil / x}{2}\right)\right)^2.$$

Thus, $\sqrt{x + \lfloor x \rceil} \leq \sqrt{x}(1 + \frac{\lfloor x \rceil / x}{2})$, and

$$\begin{aligned}
\lfloor y \rceil &= \mathrm{up}_\tau\left(\epsilon_\tau(\langle y \rangle) + \frac{\langle x \rangle^{1/2}}{2}\left(\frac{\lfloor x \rceil}{\langle x \rangle - \lfloor x \rceil}\right)\right) \\
&\leq (1 + 2^{-\tau})\left(2^{-\tau}\mathrm{fl}_\tau(\sqrt{x + \lfloor x \rceil}\,) + \frac{(x + \lfloor x \rceil)^{1/2}}{2}\left(\frac{\lfloor x \rceil}{x - 2\lfloor x \rceil}\right)\right) \\
&\leq (1 + 2^{-\tau})\left(2^{-\tau}(1 + 2^{-\tau})\sqrt{x + \lfloor x \rceil} + \frac{\sqrt{x}}{2}\left(1 + \frac{\lfloor x \rceil / x}{2}\right)\left(\frac{\lfloor x \rceil / x}{1 - 2\lfloor x \rceil / x}\right)\right) \\
&\leq (1 + 2^{-\tau})\left(2^{-\tau}(1 + 2^{-\tau})\sqrt{x}\left(1 + \frac{\lfloor x \rceil / x}{2}\right) + \frac{\sqrt{x}}{2}\left(1 + \frac{\lfloor x \rceil / x}{2}\right)\left(\frac{\lfloor x \rceil / x}{1 - 2\lfloor x \rceil / x}\right)\right) \\
&\leq (1 + 2^{-\tau})\left(2^{-\tau}(1 + 2^{-\tau})\left(1 + \frac{\lfloor x \rceil / x}{2}\right)\frac{x}{\lfloor x \rceil} + \frac{1}{2}\left(1 + \frac{\lfloor x \rceil / x}{2}\right)\left(\frac{1}{1 - 2\lfloor x \rceil / x}\right)\right) \cdot \frac{\lfloor x \rceil}{x} \cdot \sqrt{x} \\
&\leq (1 + 2^{-\tau})\left((1 + 2^{-\tau})\left(1 + \frac{\lfloor x \rceil / x}{2}\right) + \frac{1}{2}\left(1 + \frac{\lfloor x \rceil / x}{2}\right)\left(\frac{1}{1 - 2\lfloor x \rceil / x}\right)\right) \cdot \frac{\lfloor x \rceil}{x} \cdot \sqrt{x} \\
&\leq \frac{17}{16}\left(\frac{17}{16} \cdot \frac{17}{16} + \frac{1}{2} \cdot \frac{17}{16} \cdot \frac{4}{3}\right) \cdot \frac{\lfloor x \rceil}{x} \cdot \sqrt{x} \\
&< 2 \cdot \frac{\lfloor x \rceil}{x} \cdot \sqrt{x}.
\end{aligned}$$

□

B.3 Sum

Lemma B.5. *Let x_1, x_2, $\ldots$, x_n be real numbers, and $y = \sum_{i=1}^n x_i$. If $\tau \geq \log_2(32n)$, then*

$$\lfloor y \rceil \leq 2\sum_{i=1}^n \lfloor x_i \rceil + 2^{1-\tau} n \sum_{i=1}^n |x_i|.$$

Proof. Let $y_i = \sum_{k=1}^i x_i$. For $i > 1$, we have

$$|\langle y_i \rangle| = |\mathrm{fl}_\tau(\langle y_{i-1} \rangle + \langle x_i \rangle)| \leq (1 + 2^{-\tau})(|\langle y_{i-1} \rangle| + |\langle x_i \rangle|).$$

By induction, we have $|\langle y_i \rangle| \leq \sum_{j=1}^i (1 + 2^{-\tau})^{i-j+1} |\langle x_j \rangle|$, and we can further simplify the inequality to: $|\langle y_i\rangle| \le (1+2^{-\tau})^n \sum_{j=1}^{n} |\langle x_j\rangle|$. Thus,

$$\begin{aligned}\lfloor y_i \rceil &= \mathrm{up}_\tau(\lfloor y_{i-1}\rceil + \lfloor x_i \rceil + \epsilon_\tau(\langle y_i\rangle)) \le (1+2^{-\tau+1})\Big(\lfloor y_{i-1}\rceil + \lfloor x_i\rceil + 2^{-\tau}|\langle y_i\rangle|\Big)\\ &\le (1+2^{-\tau+1})\Big(\lfloor y_{i-1}\rceil + \lfloor x_i\rceil + 2^{-\tau}(1+2^{-\tau})^n \sum_{j=1}^{n} |\langle x_j\rangle|\Big).\end{aligned}$$

Again, by induction, we have

$$\begin{aligned}\lfloor y_n\rceil &\le \sum_{i=1}^{n}(1+2^{-\tau+1})^{n-i+1}\lfloor x_i\rceil \ + \ 2^{-\tau}(1+2^{-\tau})^n\Big(\sum_{j=1}^{n}|\langle x_j\rangle|\Big)\Big(\sum_{j=1}^{n}(1+2^{-\tau+1})^{n-i+1}\Big)\\ &\le (1+2^{-\tau+1})^n\sum_{i=1}^{n}\lfloor x_i\rceil \ + \ 2^{-\tau}n(1+2^{-\tau+1})^{2n}\Big(\sum_{j=1}^{n}|\langle x_j\rangle|\Big)\\ &\le (1+2^{-\tau+1})^{2n}\sum_{i=1}^{n}\lfloor x_i\rceil \ + \ 2^{-\tau}n(1+2^{-\tau+1})^{2n}\sum_{j=1}^{n}(|x_j| + \lfloor x_j\rceil)\\ &\le ((1+2^{-\tau+1})^{2n} + 2^{-\tau}n(1+2^{-\tau+1})^{2n})\sum_{i=1}^{n}\lfloor x_i\rceil \ + \ 2^{-\tau}n(1+2^{-\tau+1})^{2n}\sum_{j=1}^{n}|x_j|.\end{aligned}$$

If $\tau \ge \log_2(16n)$, then

$$(1+2^{-\tau+1})^{2n} \le \left(1+\frac{1}{4}\left(\frac{1}{2n}\right)\right)^{2n} \le e^{1/4} < 1.29.$$

Therefore,

$$\lfloor y_n\rceil \le (1.29 + 1.29\times 0.25)\sum_{i=1}^{n}\lfloor x_i\rceil + 1.29\times 2^{-\tau}n\sum_{i=1}^{n}|x_i| < 2\sum_{i=1}^{n}\lfloor x_i\rceil + 2^{1-\tau}n\sum_{i=1}^{n}|x_i|.$$

Note that the bound does not depend on the order at which the $\langle\langle x_i\rangle, \lfloor x_i\rceil\rangle$ are added. □

Corollary B.6. *If all the x_i's are positive, $\tau > \log_2(16n)$, and there exist $\varepsilon \ge 2^{-\tau}$ such that $\varepsilon \ge \frac{\lfloor x_i\rceil}{x_i}$ for all i, then*

$$\lfloor y\rceil \le (2n+2)\varepsilon y.$$

In other words, the relative error increases by at most a factor of $2n+2$.

Proof. From Lemma B.5, we have

$$\lfloor y_n\rceil \ \le\ 2\sum_{i=1}^{n}\lfloor x_i\rceil + 2^{1-\tau}n\sum_{j=1}^{n}x_j \ \le\ 2\sum_{i=1}^{n}\varepsilon x_i + 2n\varepsilon\sum_{j=1}^{n}x_j \ =\ (2n+2)\varepsilon\sum_{i=1}^{n}x_i$$ □

B.4 Dot Product

Lemma B.7. *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be two* n*-vectors, and let* $\llbracket \mathbf{a} \rrbracket$ *and* $\llbracket \mathbf{b} \rrbracket$ *be vectors of bracket coefficients that approximate them, respectively. Let* $\llbracket c \rrbracket$ *be the bracket coefficient obtained by multiplying corresponding elements of* $\llbracket \mathbf{a} \rrbracket$ *and* $\llbracket \mathbf{b} \rrbracket$*, and then adding all the products together ("dotting"* $\llbracket \mathbf{a} \rrbracket$ *and* $\llbracket \mathbf{b} \rrbracket$*). If* $\tau > \log_2(16n)$ *and* $n \geq 2$*, then*

$$\lfloor c \rceil \;\leq\; 9\lfloor \mathbf{a} \rceil \cdot \lfloor \mathbf{b} \rceil \;+\; 3|\mathbf{a}| \cdot \lfloor \mathbf{b} \rceil \;+\; 3|\mathbf{b}| \cdot \lfloor \mathbf{a} \rceil \;+\; 2^{1-\tau}(n+3)|\mathbf{a}||\mathbf{b}|.$$

Proof. Let $p_i = a_i b_i$ for $1 \leq i \leq n$. By Lemma B.5, we have

$$\begin{aligned}
\lfloor c \rceil &\leq 2\sum_{i=1}^{n} \lfloor p_i \rceil + 2^{1-\tau} n \sum_{i=1}^{n} |p_i| \\
&= 2\sum_{i=1}^{n} \mathrm{up}_\tau(\lfloor a_i \rceil \lfloor b_i \rceil + |\langle a_i \rangle| \lfloor b_i \rceil + |\langle b_i \rangle| \lfloor a_i \rceil + \epsilon_\tau(\mathrm{fl}_\tau(\langle a_i \rangle \langle b_i \rangle))) + 2^{1-\tau} n \sum_{i=1}^{n} |a_i||b_i| \\
&\leq 2(1+2^{-\tau+1})\Big(\sum_{i=1}^{n} \lfloor a \rceil \lfloor b \rceil + \sum_{i=1}^{n} |\langle a_i \rangle| \lfloor b_i \rceil + \sum_{i=1}^{n} |\langle b_i \rangle| \lfloor a_i \rceil + 2^{-\tau}(1+2^{-\tau}) \sum_{i=1}^{n} |\langle a_i \rangle||\langle b_i \rangle|\Big) \\
&\quad + 2^{1-\tau} n \sum_{i=1}^{n} |a_i||b_i| \\
&\leq 2(1+2^{-\tau+1})\Big(\sum_{i=1}^{n} \lfloor a \rceil \lfloor b \rceil + \sum_{i=1}^{n} (|a_i| + \lfloor a_i \rceil) \lfloor b_i \rceil + \sum_{i=1}^{n} (|b_i| + \lfloor b_i \rceil) \lfloor a_i \rceil \\
&\quad + 2^{-\tau}(1+2^{-\tau}) \sum_{i=1}^{n} (|a_i| + \lfloor a_i \rceil)(|b_i| + \lfloor b_i \rceil)\Big) + 2^{1-\tau} n \sum_{i=1}^{n} |a_i||b_i| \\
&\leq 2(1+2^{-\tau+1})((4+2^{-\tau}(1+2^{-\tau}))\lfloor \mathbf{a} \rceil \cdot \lfloor \mathbf{b} \rceil + (1+2^{-\tau}(1+2^{-\tau}))|\mathbf{a}| \cdot \lfloor \mathbf{b} \rceil \\
&\quad + (1+2^{-\tau}(1+2^{-\tau}))|\mathbf{b}| \cdot \lfloor \mathbf{a} \rceil) + 2^{1-\tau}(2(1+2^{-\tau}) + n)|\mathbf{a}||\mathbf{b}|.
\end{aligned}$$

Since $n \geq 2$ and $\tau > \log_2(16n)$, we have $2^{-\tau} \leq 1/32$, and

$$\begin{aligned}
\lfloor c \rceil &\leq 2\left(1+\frac{1}{16}\right)\left(\left(4+\frac{1}{32}\left(1+\frac{1}{32}\right)\right)\lfloor \mathbf{a} \rceil \cdot \lfloor \mathbf{b} \rceil + \left(1+\frac{1}{32}\left(1+\frac{1}{32}\right)\right)|\mathbf{a}| \cdot \lfloor \mathbf{b} \rceil \right. \\
&\quad \left. + \left(1+\frac{1}{32}\left(1+\frac{1}{32}\right)\right)|\mathbf{b}| \cdot \lfloor \mathbf{a} \rceil\right) + 2^{1-\tau}\left(2\left(1+\frac{1}{32}\right) + n\right)|\mathbf{a}||\mathbf{b}| \\
&< 9\lfloor \mathbf{a} \rceil \cdot \lfloor \mathbf{b} \rceil \;+\; 3|\mathbf{a}| \cdot \lfloor \mathbf{b} \rceil \;+\; 3|\mathbf{b}| \cdot \lfloor \mathbf{a} \rceil \;+\; 2^{1-\tau}(n+3)|\mathbf{a}||\mathbf{b}|.
\end{aligned}$$

□

The following two lemmas involve the 2-norm of vectors of bracket coefficients.

Lemma B.8. *Let* $\mathbf{a}$ *be an* n*-vector and let* $\llbracket \mathbf{a} \rrbracket$ *be a* n*-vector of bracket coefficients that approximates it. Let* $\llbracket c \rrbracket = \|\llbracket \mathbf{a} \rrbracket\|^2 = \llbracket \mathbf{a} \rrbracket \cdot \llbracket \mathbf{a} \rrbracket$*. If there exists a constant* ε *such that* $2^{-\tau} < \varepsilon < 1/9$*, and* $\lfloor a_i \rceil \leq \varepsilon |a_i|$

for all $1 \leq i \leq n$*, and* $\tau > \log_2(16n)$*, then*

$$\lfloor c \rceil \leq (2n+13)\varepsilon\|\mathbf{a}\|^2.$$

Proof. By Lemma B.7,

$$\begin{aligned}
\lfloor c \rceil &\leq 9\lfloor \mathbf{a} \rceil \cdot \lfloor \mathbf{a} \rceil \;+\; 6|\mathbf{a}| \cdot \lfloor \mathbf{a} \rceil \;+\; 2^{1-\tau}(n+3)|\mathbf{a}| \cdot |\mathbf{a}| \\
&\leq 9\varepsilon^2\|\mathbf{a}\|^2 \;+\; 6\varepsilon\|\mathbf{a}^2\| \;+\; (2n+6)\varepsilon\|\mathbf{a}\|^2 \\
&< \varepsilon\|\mathbf{a}\|^2 \;+\; 6\varepsilon\|\mathbf{a}^2\| \;+\; \varepsilon(2n+6)\|\mathbf{a}\|^2 \\
&= (2n+13)\varepsilon\|\mathbf{a}\|^2.
\end{aligned}$$

□

Lemma B.9. *Let* $\mathbf{a}$*,* $\llbracket \mathbf{a} \rrbracket$*, and* $\llbracket c \rrbracket$ *be as in Lemma B.8. If there exists a constant* ε *such that such that* $2^{-\tau} < \epsilon < 1/9$*, and* $\lfloor a_i \rceil \leq \varepsilon\|\mathbf{a}\|$ *for all* $1 \leq i \leq n$*, and* $\tau > \log_2(16n)$*, then*

$$\lfloor c \rceil \leq (3n + 6\sqrt{n} + 6)\varepsilon\|\mathbf{a}\|^2.$$

Proof. Again, by Lemma B.7,

$$\begin{aligned}
\lfloor c \rceil &\leq 9\lfloor \mathbf{a} \rceil \cdot \lfloor \mathbf{a} \rceil \;+\; 6|\mathbf{a}| \cdot \lfloor \mathbf{a} \rceil \;+\; 2^{1-\tau}(n+3)|\mathbf{a}| \cdot |\mathbf{a}| \\
&\leq 9n(\varepsilon\|\mathbf{a}\|)^2 \;+\; 6\varepsilon\|\mathbf{a}\|\sum_{i=1}^{n}|a_i| \;+\; (2n+6)\varepsilon\|\mathbf{a}\|^2 \\
&\leq n(\varepsilon\|\mathbf{a}\|)^2 \;+\; 6\varepsilon\|\mathbf{a}\|\sqrt{n}\|\mathbf{a}\| \;+\; (2n+6)\varepsilon\|\mathbf{a}\|^2 \\
&\leq (3n + 6\sqrt{n} + 6)\varepsilon\|\mathbf{a}\|^2.
\end{aligned}$$

□

Bibliography

[1] G. Alefeld and J. Herzberger. *Introduction to Interval Computations, Computer Science, and Applied Mathematics.* Academic Press, 1983.

[2] M. Bronstein. *Symbolic Integration I: Transcendental Functions.* Springer–Verlag, 1997.

[3] R. Corless, S. Watt, and L. Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Processing*, 52, 2004.

[4] L. Ducos. Optimizations of the subresultant algorithm. *J. Symb. Comput*, 145(2):149–163, 2000.

[5] G. Golub and C. Van Loan. *Matrix Computation.* John Hopkins University Press, 1996.

[6] G. Hardy and E. Wright. *An introduction to the theory of numbers.* Oxford University Press, 1979.

[7] M. Laidacker. Another theorem relating Sylvester's matrix and the greatest common divisor. *Math. Mag.*, 42, 1969.

[8] T. Lickteig and M. Roy. Sylvester-habicht sequences and fast Cauchy index computation. *J. Symb. Comput*, 32:315–341, 2001.

[9] H. Minakuchi, H. Kai, K. Shirayanagi, and M-T. Noda. Algorithm stabilization techniques and their application to symbolic computation of generalized inverses. In *Electronic Proc. of the IMACS Conference on Applications of Computer Algebra (IMACS-ACA'97)*, 1997.

[10] H. Sekigawa and K. Shirayanagi. Zero in interval computation and its applications to Sturm's algorithm. Poster presentation at International Symposium on Symbolic and Algebraic Computation (ISSAC'95), 1995.

[11] K. Shirayanagi. Floating-point Gröbner bases. *Mathematics and Computers in Simulation*, 42, 1996.

[12] K. Shirayanagi and M. Sweedler. A theory of stabilizing algebraic algorihms. Technical Report 95-28, Mathematical Sciences Institute, Cornell University, 1995.

[13] K. Shirayanagi and M. Sweedler. Remarks on automatic algorithm stabilization. *J. Symb. Comput.*, 26, 1998.

[14] C. Zarowski, X. Ma., and F. Fairman. A QR-factorization method for computing the greatest common divisor of polynomials with real-valued coefficients. *IEEE Trans. Signal Processing*, 48, 2000.