VOT 71565

DEVELOPMENT OF SYMBOLIC ALGORITHMS FOR CERTAIN ALGEBRAIC PROCESSES

(PEMBINAAN ALGORITMA SIMBOLIK UNTUK BEBERAPA PROSES ALGEBRA)

ALI BIN ABD RAHMAN

NOR'AINI BINTI ARIS

RESEARCH VOTE NO:

71565

Jabatan Matematik

Fakulti Sains

Universiti Teknologi Malaysia

2007

ACKNOWLEDGEMENTS

In The Name Of ALLAH, The Most Beneficent, The Most Merciful

All praise is due only to ALLAH, the Lord of the Worlds. Ultimately, only Allah has given us the strength and courage to proceed with our entire life. His works are truly splendid and wholesome and His knowledge is truly complete with due perfection.

The financial support from UTM, particularly, from the Research Management Center, for its research grant is greatly appreciated. An acknowledgement is also due to the Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Linz, Austria for granting the permission to a copy of SACLIB. A special gratitude to Professor George Collins who initiated the C-based, open source code software and who has always been responsive to questions regarding the implementation of its programs. A lot of professor Collin's work, in particular, his deterministic approach on the theoretical computing time analysis of algorithms has been referred to in this research work.

ABSTRACT

This study investigates the problem of computing the exact greatest common divisor of two polynomials relative to an orthogonal basis, defined over the rational number field. The main objective of the study is to design and implement an effective and efficient symbolic algorithm for the general class of dense polynomials, given the rational number defining terms of their basis. From a general algorithm using the comrade matrix approach, the nonmodular and modular techniques are prescribed.

If the coefficients of the generalized polynomials are multiprecision integers, multiprecision arithmetic will be required in the construction of the comrade matrix and the corresponding systems coefficient matrix. In addition, the application of the nonmodular elimination technique on this coefficient matrix extensively applies multiprecision rational number operations. The modular technique is employed to minimize the complexity involved in such computations. A divisor test algorithm that enables the detection of an unlucky reduction is a crucial device for an effective implementation of the modular technique. With the bound of the true solution not known a priori, the test is devised and carefully incorporated into the modular algorithm.

The results illustrate that the modular algorithm illustrate its best performance for the class of relatively prime polynomials. The empirical computing time results show that the modular algorithm is markedly superior to the nonmodular algorithms in the case of sufficiently dense Legendre basis polynomials with a small GCD solution. In the case of dense Legendre basis polynomials with a big GCD solution, the modular algorithm is significantly superior to the nonmodular algorithms in higher degree polynomials. For more definitive conclusions, the computing time functions of the algorithms that are presented in this report have been worked out. Further investigations have also been suggested.

Key Researches:
Assoc. Prof. Dr. Ali Abd Rahman
Nor'aini Aris

ABSTRAK

Kajian ini mengkaji masalah pengiraan tepat pembahagi sepunya terbesar dua polinomial relatif terhadap suatu asas ortogon yang tertakrif ke atas medan nombor nisbah. Objektif utama kajian ialah untuk membangunkan dan melaksanakan suatu alkhwarizmi simbolik untuk kelas umum polinomial tumpat, diketahui sebutan pentakrif bagi asasnya. Kaedah penyelesaian modulo dan tak modulo diasaskan kepada suatu alkhwarizmi am. Jika pekali bagi polinomial dalam perwakilan teritlak merangkumi integer kepersisan berganda, aritmetik kepersisan berganda diperlukan dalam pembinaan matriks komrad dan matriks pekali sistem berpadanan. Penggunaan teknik penurunan tak modulo terhadap matriks pekali tersebut banyak melibatkan aritmetik nombor nisbah dan operasi integer kepersisan berganda.

Teknik modulo di selidiki bagi mengurangkan kerumitan penggunaan operasi tersebut. Keberkesanan pendekatan modulo bergantung kepada kejayaan mewujudkan suatu teknik yang dapat menyemak kebenaran penyelesaian pada peringkat-peringkat penyelesaian secara berpatutan. Dalam keadaan di mana batas jawapan sebenar tidak dapat ditentukan, suatu pendekatan modulo telah dihasilkan.

Teknik tersebut mempamerkan pencapaian terbaik dalam perlaksanaannya terhadap kelas polinomial yang perdana relatif. Keputusan ujikaji masa pengiraan menunjukkan kecekapan alkhwarizmi modulo jauh mengatasi alkhwarizmi tak modulo dalam kelas polinomial tumpat terhadap asas Legendre yang mempunyai satu polinomial pembahagi sepunya terbesar bersaiz kecil. Bagi kelas polinomial yang sama tetapi mempunyai satu penyelesaian bersaiz besar, alkhwarizmi modulo adalah jauh lebih cekap untuk polinomial berdarjah besar. Keputusan masa pengiraan untuk polinomial nombor nisbah yang diwakili oleh polinomial terhadap asas Legendre mempamerkan kecekapan alkhwarizmi yang menggabungkan pendekatan modulo dan tak modulo pada beberapa bahagian tertentu daripada keseluruhan kaedah penyelesaian. Untuk keputusan secara teori dan berpenentuan, fungsi masa pengiraan bagi alkhwarizmi yang dibina telah diusahakan. Penyelidikan lanjut juga dicadangkan.

Contents

CHAPTER I

INTRODUCTION

1.1 Preface

Fundamental problems in computational algebra that require symbolic computations or manipulations are the computation of greatest common divisors (GCDs) of integers or polynomials, polynomial factorization and finding the exact solutions of systems of algebraic equations. Other problems include symbolic differentiation and integration, determination of solutions of differential equations and the manipulation of polynomials and rational functions within its broad spectrum of applications. In the last decade, tremendous success has been achieved in this area. At the same time, the invention of powerful SAC systems has been motivated to assisting the scientific community in doing algebraic manipulation as well as providing means of designing and implementing new algorithms. Useful as research tools in many scientific areas, notable symbolic and algebraic computation (SAC) systems presently used worldwide include REDUCE, MACSYMA, MAPLE, MATHEMATICA, SACLIB and most recently MUPAD. Among these systems, SACLIB and MUPAD are open source LINUX or UNIX systems which are meant for academic purposes that contribute to further research in the symbolic algorithms and systems development.

The problem of finding the greatest common divisor (GCD) of the integers is one of the most ancient problems in the history of mathematics [37]. As early as 300 B.C., Euclid discovered an algorithm that still bears his name to solve this problem. Today, the concept of divisibility plays a central role in symbolic computations. In this mode, the computation of GCD is not only applied to reducing rational numbers (quotient of integers) to its lowest terms but also to reducing quotients of elements from any integral domain to its simplest form. For example, the computation of GCD of two polynomials reduces a rational function (quotients of polynomials) to its lowest term. In finding the roots of a polynomial with repeated factors, it is often desirable

to divide the polynomial with the GCD of itself and its derivative to obtain the product of only linear factors.

An application in the integration of algebraic functions reported that 95% of the total time was being spent in GCD computation [42]. As such, GCD operations become the bottleneck of many basic applications of exact computations. In addition, problems involving polynomials, such as polynomial GCD computation and polynomial factorization, play an important role in many areas of contemporary applied mathematics, including linear control systems, electrical networks, signal processing and coding theory. In the theory of linear multivariable control systems, GCD determination arises in the computation of the Smith form of a polynomial matrix, and associated problems, such as minimal realization of a transfer function matrix. There are also applications in network theory and in finding solution of systems of linear diophantine equations and integer linear programming.

1.2 Motivation

The theory on polynomials in forms other than the power form has been the results of several works of Barnett and Maroulas, such as in [7], [8], [9] and [10]. The application of polynomials in the generalized form arises in the computation of the Smith form of a polynomial matrix and problems associated with the minimal realization of a transfer function matrix. The zeros and extrema of the Chebyshev polynomials play an important role in the theory of polynomial interpolation [54]. Barnett [11], describes the theory and applications of such polynomials in linear control systems.

Let $a(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$ be a polynomial in a variable x over the field F. If $a_n \neq 0$ then degree of $a(x)$ is denoted $\deg(a)$. The zero polynomial has $a_i = 0$ for all i and when $a_n = 1$, $a(x)$ is termed monic. In this form, $a(x)$ is univariate in the standard form or power series form; that is, a polynomial expressed as a linear combination of the monomial basis $\{1, x, , x^2, ..., x^{n-1}, x^n\}$.

Let $a(x)$ be monic and F be the field of real numbers R. A companion matrix

associated with $a(x)$ is the $n \times n$ matrix

$$\begin{pmatrix} 0 & I_{n-1} & & \\ -a_0 & -a_1 & \dots & -a_{n-1} \end{pmatrix},$$

where I_n denotes the $n \times n$ unit or identity matrix. Given a set of orthogonal polynomials $\{p_i(x)\}$, Barnett [6] shows that associated with a polynomial $a(x) = \sum a_i p_i(x)$, where $a_i \in R$, there exists a matrix A which possesses several of the properties of the companion matrix. The term comrade matrix is an analogue of the companion matrix. The companion matrix is associated with a polynomial expressed in the standard or power series form. The comrade matrix is a generalization of the companion matrix and is associated with a polynomial expressed as a linear combination of an arbitrary orthogonal basis. The term colleague matrix is associated with a polynomial expressed as a linear combination of the Chebyshev polynomials, a special subclass of the orthogonal basis. Further generalization of the companion matrix corresponds to the confederate matrix. In the latter, a polynomial $a(x)$ is expressed in its so called generalized form, such that $a(x) = \sum_{i=0}^{n} a_i p_i(x)$, in which case, $\{p_i(x)\}_{i=0}^{n}$ corresponds to a general basis. In [11], the term congenial matrices which include the colleagues, comrades and confederate matrices is mentioned. However useful these matrices are, Barnett [6] and [7] does not get around to converting the theories and analytical results that have been discovered into its practical use. The theories and analytical results of [6] and [7] have to be converted into a procedure so that the routines that extend their applications can be implemented effectively and efficiently using the computer.

Apart from that, it is natural to speculate whether theorems for polynomials in the standard form carry over to polynomials in the generalized form. Grant and Rahman [29] consider ways in which an algorithm for finding the zeroes of standard form polynomials can be modified to find the zeroes of polynomials in other basis. According to Barnett [11], few results have been reported in this area. For example, the Routh Tabular Scheme (RTS) [7] can be applied to the problem of locating the roots of generalized polynomial in the left and right halves of the complex plane. A result that requires determination of odd and even parts of the polynomial is obtained. Similar attempts to generalize the theorems of Hurwitz, Hermite and Schur-Cohn (section 3.2 of [11]) have also proved unsuccessful in obtaining a criterion which can be applied directly to the coefficients of the generalized form. This means that the

polynomials have to be converted to the standard form for the calculations to be carried out and the solutions in the standard form have to be converted back to the original generalized form. This extra work can be avoided by extending or utilizing the properties of the polynomials given in their respective generalized form.

The RTS calculates the rows of the upper triangular form of the Sylvester's matrix. The last nonzero row of the upper triangular form of the Sylvester's matrix gives the coefficients of the GCD of the associated polynomials. Alternate rows obtained from the RTS give a polynomial remainder sequence. In [7], the relation between the polynomial remainder sequence obtained from the rows of the Sylvester's matrix and the Euclidean polynomial remainder sequence is given. The extension of the RTS to the generalized polynomials, examining its application in the computation of the GCD when the degrees of the polynomials are equal is discussed. In a way, the method shows that the division of polynomials in the generalized form cannot be carried out directly unless the degrees of the polynomials are equal. Analytically, the Routh Tabular Scheme has only been applied for computing the GCD of the polynomials in the orthogonal basis when the degrees of the input polynomials are equal.

These observations lead us to further investigate polynomials in the generalized form. Rather then converting the polynomials to power series form, we aim at obtaining some criteria so that the coefficients of the generalized polynomials can be applied directly when performing certain operations such as multiplication, division and computing the GCD. Adhering to the given form of polynomials can avoid unnecessary work of basis transformation. Furthermore the properties of polynomials in its original form can be fully utilized and further investigated.

A.A. Rahman [1] presents an investigation into GCD algorithms in a floating point environment and uses the results in numerical experiments to solve polynomial exhibiting repeated roots. Although the method developed shows potential, as expected, the build up of round off errors is so enormous as to render the results for high degree polynomials questionable.

In order to compute the GCD of generalized polynomials, the comrade matrix analogue of the companion matrix requires multiplication and division of the defining

terms of the corresponding basis. The entries of the last row of the comrade matrix involve the operations of the basis defining terms with the coefficients of the input polynomial. Considering the results in [1] and the more complex nature of the comrade matrix, in comparison to the simple form exhibited by the companion matrix, the method of working in the floating point environment to compute the GCD of polynomials in its generalized form using the ideas developed by Barnett [7], may only be able to achieve a varying and limited degree of success depending on the conditions imposed by the input polynomials.

However, the effects of the resulting rounding errors and truncation errors may be over come by working in the symbolic computation mode. In this mode, the computations are confined to the rational numbers or integers. In many problems that involve integer polynomial and matrix computations, the symbolic methods of palliating the exponential growth of integers in intermediate steps (known as intermediate expression swell) have been a great success. The development of these methods and their significant contributions, particularly the methods that apply modular techniques in computing the GCD of polynomials and solving the solution to linear systems are mentioned in the literature review of Chapter 2.

On the other hand, in the floating point environment, a major setback is the critical task to identify a nonzero number from a zero number that arises from the accumulation of errors. For example, there exists some limitations of numerical computation techniques in the computation of rank and solutions to nonsingular systems. As a consequence, the applications of numerical techniques in the floating point environment should be able to overcome this task in order to produce an effective implementation of the techniques.

Furthermore, let x and y be real numbers such that $x < y$. Then there exists a rational number z such that $x < z < y$ (refer to [53], page 21). Thus, by the fact that the rational numbers are order dense, we may use a rational number to represent a real number that is sufficiently close to it, without losing accuracy from an effective implementation of the exact computational method.

In an effort to further proceed on implementing the ideas that have been established in [6] and [7], we will apply the comrade matrix approach to design and

implement an effective and efficient symbolic algorithm for computing the GCD of two polynomials represented relative to a general basis. A detail description of the problem is explained in the following section.

1.3 Preliminary Definitions and Concepts

Let the set of real polynomials $p_0(x), p_1(x), ..., p_n(x)$, each with its degree $\deg(p_i) = i$, constitutes an arbitrary basis. The main thrust of the theoretical developments that have been established in [6] and [7] concentrate on the case where the basis consists of a set $\{p_i(x)\}$, which is mutually orthogonal with respect to some interval and weighting function. We proceed by first stating some definitions and basic concepts.

Definition 1.1. *A sequence $\{p_i(x)\}$ is orthogonal with respect to a certain function $w(x)$ if there exists an interval $[c, d]$ such that:*

$$\int_c^d w(x)d(x) > 0,$$

and

$$\int_c^d p_i(x)p_j(x)w(x)dx = \begin{cases} 0 & \text{if } i \neq j, \\ c \neq 0 & \text{if } i = j. \end{cases} \tag{1.1}$$

Definition 1.2. *A polynomial relative to an orthogonal basis is written in the form:*

$$a(x) = \sum_{i=0}^{n} a_i p_i(x),$$

such that

$$p_i(x) = \sum_{j=0}^{i} p_{ij} x^j. \tag{1.2}$$

with the set $\{p_i(x)\}$ known as an orthogonal basis.

Let $\{p_i(x)\}$ be a set of orthogonal polynomials. Then $\{p_i(x)\}$ satisfies the three-term recurrence relation:

$$\begin{aligned} p_0(x) &= 1, \\ p_1(x) &= \alpha_0 x + \beta_0, \\ p_{i+1}(x) &= (\alpha_i x + \beta_i)p_i(x) - \gamma_i p_{i-1}(x), \end{aligned} \tag{1.3}$$

where $i = 1, 2, .., n-1$ with $\alpha_i > 0$, $\gamma_i, \beta_i \geq 0$. From this three-term recurrence relation, which is also known as the defining relations of the basis $\{p_i(x)\}$, we may easily obtain the defining terms $\alpha_i > 0$, $\gamma_i, \beta_i \geq 0$. In this thesis, we consider polynomials represented in the generalized form, that is, polynomials expressed in terms of a general basis, such that the basis polynomials should be generated by the equation (1.3). This definition is based on [29], which determines the zeros of a linear combination of generalized polynomials. The orthogonal basis is a special case, where the basis polynomials will form a family of polynomials $\{p_i(x)\}$ generated by equation (1.3). With $p_i(x)$ a polynomial of degree i in x, $\{p_i(x)\}$ for $i = 0, ..., n$ form a linearly independent set and hence provide a basis for the representation of any polynomial of degree n. Examples of basis generated in this way are given in [29]. Our implementations concentrate on the case when the defining terms are rational numbers and when $\beta_i = 0$, as in the case of the Legendre polynomial basis.

Any given nth degree univariate polynomial in x can be expressed uniquely as a linear combination of the set of basis $\{p_0(x), p_1(x), ..., p_n(x)\}$. Any two arbitrary polynomials in power series form $\tilde{a}(x) = \tilde{a}_0 + \tilde{a}_1 x + ... + \tilde{a}_{n-1}x^{n-1} + x^n$ and $\tilde{b}(x) = \tilde{b}_0 + \tilde{b}_1 x + ... + \tilde{b}_m x^m$ with coefficients over a field can be written in the form:

$$a(x) = a_0 p_0(x) + a_1 p_1(x) + ... + a_n p_n(x),$$

and

$$b(x) = b_0 p_0(x) + b_1 p_1(x) + ... + b_m p_m(x). \tag{1.4}$$

such that $a(x)$ and $b(x)$ are the generalized form of $\tilde{a}(x)$ and $\tilde{b}(x)$ respectively. Note that, in power series form, $\tilde{a}(x)$ can be written as

$$\begin{aligned} \tilde{a}(x) &= a(x)/\alpha_0\alpha_1...\alpha_{n-1}, \\ &= \tilde{a}_0 + \tilde{a}_1 x + ... + \tilde{a}_{n-1}x^{n-1} + x^n. \end{aligned} \tag{1.5}$$

Example 1.1. *Let* $\tilde{a}(x) = 7072x^7 + 1120x^5 + 221x^3 + 663x^2 + 35x + 105$. *In the generalized form,* $\tilde{a}(x)$ *is represented relative to the Legendre basis as*

$$a(x) = p_7(x) + \frac{308}{51}p_5(x) + \frac{887639}{65280}p_3(x) + \frac{429}{256}p_2(x) + \frac{247907}{21760}p_1(x) + \frac{5379}{4352}p_0(x).$$

In the above example, we apply the method of basis transformation given in [11] (refer to page 371-372). The OpGcd package [47] designed in MAPLE are used to do the transformation.

For any field F, let $a(x)$ and $b(x)$ be polynomials in $F[x]$. Then $b(x)$ is a multiple of $a(x)$ if $b(x) = a(x)r(x)$ for some $r(x)$ in $F[x]$. This also implies that $a(x)$ divides $b(x)$ which is denoted as $a(x)|b(x)$. A polynomial $d(x)$ is called a greatest common divisor (GCD) of $a(x)$ and $b(x)$ if $d(x)$ divides $a(x)$ and $b(x)$ and $d(x)$ is a multiple of any other common divisor of $a(x)$ and $b(x)$.

1.4 Problem Statement

Suppose $a(x)$ and $b(x)$ are in the form (1.4) such that the defining equation (1.3) for its basis are given. The aim of the research is to apply appropriate symbolic computation techniques to the comrade matrix approach proposed by Barnett [6] and [7] in the design and implementation of an effective and efficient algorithm for computing the exact GCD of $a(x)$ and $b(x)$ in the generalized form; that is, without converting to power series form polynomials. Rather than obtaining asymptotically fast algorithms, our main goal is to establish state-of-the-art GCD algorithms for generalized polynomial. Besides producing an efficient algorithm to be useful in practice, we will be exploring the state of the art vis-a-vis the problem at hand by investigations into the theoretical contributions that have been established but not yet implemented, tested or analyzed.

1.5 Research Objectives

This section briefly describes our research objectives as follows:

1. Construct a general algorithm to compute the GCD of polynomials expressed in the general basis, in particular the orthogonal basis, which satisfies equation (1.3).

2. Construct the algorithm for computing the comrade matrix and the algorithm

for computing the corresponding systems coefficient matrix (step 2 of the general procedure). In symbolic computation, the auxiliary algorithms for rational number matrix operations need to be constructed and added to the existing SACLIB's library functions. Incorporate these algorithms into the general algorithm.

3. Construct the exact division or fraction free Gauss elimination algorithms. Construct the modular Gauss elimination algorithm.

4. Incorporate the nonmodular Gauss elimination algorithms into the general algorithm to construct nonmodular OPBGCD algorithms(Orthogonal Polynomial Basis GCD).Incorporate the modular Gauss elimination algorithm to construct a modular OPBGCD algorithm.

5. Determine a termination criterion for the modular OPBGCD algorithm, that is, devise a divisor test algorithm to check on the effectiveness of the modular technique. Determine the set of primes required to produce efficient modular algorithms.

6. Conduct the empirical and theoretical computing time analysis of the algorithms.In the empirical observations the best performing nonmodular algorithm serves as a benchmark to measure the efficiency of the modular algorithms.

1.6 Scope of Work

The procedure developed are based on the theories that have been established in Barnett [6] and[7]. Thus, the comrade matrix approach will be investigated and implemented into the algorithms for computing the exact greatest common divisor of polynomials whose basis are particularly the orthogonal polynomials with rational coefficients.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

In this chapter major historical developments in exact polynomial GCD computations of power series polynomials over the integral domain Z are discussed. The developments discussed in this respect are commonly described in a number of literature including the thesis on *Computing the GCD of Two Polynomials over an Algebraic Number Field* [33] and on *Parallel Integral Multivariate Polynomial Greatest Common Divisor Computation* [42] respectively. We do not intend to omit this report, as a prerequisite knowledge to further developments along the lines of the computation of integral or rational number polynomial greatest common divisors.

The representation of polynomials relative to a general basis is another form of representation of polynomials over the integers or rational numbers with respect to a certain basis, particularly the orthogonal basis. Recent developments on polynomial GCD computations involved the work on polynomials over any field, including the finite field of integer modulo a prime and the number field $Q(\alpha)$. Kaltofen and Monagan [36] investigate Brown's dense modular algorithm to other coefficient rings, such as GF[p] for small enough prime p, and $Z[i]$, by a generic Maple code. Much of these work involve some modifications and generalizations of the modular GCD algorithms such as the small prime, prime power and Hensel lifting methods or other nonmodular methods such as the heuristic GCDHEU, to univariate and bivariate polynomials over the algebraic number field. For a detail study refer to the work of Encarnacion's improved modular algorithm [27], the heuristic method of Smedley [49], Monagan and Margot [43], and most recently Monagan and Wittkopt in [44]. We shall not describe the work in this respect, focussing only on the major developments in the GCD computations of integral polynomials.

2.2 Polynomial GCDs over Z

The first algorithms for polynomial GCD computations are commonly known as the polynomial remainder sequence (PRS) methods which are generalizations and modifications of Euclid's algorithm originally stated for the integers. Polynomials over fields have a unique factorization property. Thus the GCD of two polynomials over a field exists and is unique up to nonzero field elements. The GCD of two polynomials over a field can be computed by this version of Euclid's algorithm adapted for polynomials and was first used by Steven (Refer to [37], page 405).

Let g be a univariate polynomial of degree n with rational coefficients. We can transform g so that $g(x) = \frac{f(x)}{c}$ such that $c \in Z$ and f is an integral polynomial. Gauss's lemma states that the factorization of f over the rational numbers and the factorization of f over the integers are essentially the same. Thus, without lost of generality we may assume that the polynomials are integral and compute the GCDs over the integers. The GCD of two polynomials over Z can then be worked over its quotient field Q.

However, the PRS method is impractical on polynomials over the integers due to the complexity of rational number arithmetic of the quotient field elements or the exponential growth of the integer coefficients at each pseudo division step in the integral domain. To overcome this coefficient growth, the primitive PRS algorithm is developed by taking the content out of pseudo remainder at each step. However, this process has been regarded as being very expensive, especially when dealing with dense polynomials. Brown [14] and Collins [18] independently introduce the reduced PRS algorithm and the subresultant PRS algorithm. These two methods control the growth of coefficient at each step by dividing out a quantity that is a factor in the coefficients of the remainder in the pseudo division step. The improved subresultant PRS algorithm is the best nonmodular algorithm. Applying the theory of PRS, a modular algorithm for polynomials over the integers is introduced by Brown [13]. The algorithm known as dense modular GCD algorithm computes the GCD of dense multivariate polynomials over $Z[x_1, x_2, ..., x_n]$, which is better than the subresultant PRS for dense polynomials. Moses and Yun [55] apply the modular homomorphic image scheme and the Hensel p-adic lifting algorithm in the EZ-GCD algorithm of

computing the GCD of two sparse multivariate integer polynomials. Further improvement have been suggested by Zippel [56], who introduces a probabilistic algorithm that preserves sparseness in Brown's interpolation scheme. In his EEZ-GCD algorithm Wang [51] avoids computing with the zero coefficients, which takes the advantage of polynomial sparseness. At the other end, a heuristic method that is based upon the integer GCD computation known as the GCDHEU algorithm is introduced by Char *et.al* [24], with further improvements suggested by Davenport and Padget [25].

We include only the most outstanding methods which are most often described in the literatures. It is observed that the modular techniques are of pertinence to the general class of polynomials; that is, the set of dense univariate or multivariate polynomials. In this case, a polynomial is said to be dense if most of its coefficients up to maximal degree are nonzero, and is sparse, otherwise. For special subclass of problems, faster algorithms can still be derived. As an example, the methods derived by Yun [55] for computing the GCD of two sparse multivariate polynomials are faster than the modular technique described by Brown [13].

2.3 Generalized Polynomial GCDs

Several applications of matrices to GCD computations of univariate polynomial GCDs over the reals such as the companion matrix approach for two and more polynomials are described in [11]. The relationship between Sylvester's resultant matrix approach to GCD computation and the Routh tabular array scheme which is also a Euclidean type scheme; an iterative scheme involving the computation of the determinant of a 2 by 2 matrix has been established. On the other hand, his investigation on polynomials with respect to a general basis leads to the design of the comrade matrix, when the basis are orthogonal and the colleague matrix, when the basis are Chebyshev. Both types of matrices are analogues to the companion matrices [8]. Barnett [7] shows how previous results from the companion matrix approach for polynomials in power series form can be extended to the case of generalized polynomials associated with its comrade matrix.

2.4 The Exact Elimination Methods

A number of methods for finding the exact solution of systems of linear equations with integer coefficients have been established since the early development of systems for symbolic mathematical computation in 1960's. Developments prior to this period involved the nonmodular methods, which control the growth of intermediate results, but introduce many more computations involving small multiprecision numbers. The fraction free Gauss elimination method, which is also known as the integer preserving Gauss elimination method is essentially that of Gauss elimination except that at each stage of the elimination, a common factor is systematically removed from the transformed matrix by a predetermined exact division. This approach is an analog of the reduced PRS-GCD algorithm. Bareiss [4] describes a variant of the method in which, by eliminating two variables at a time rather than one, improves the efficiency of the elimination considerably. In his thesis, Mazukelli [38] compares the number of arithmetic operations involved in the one and two step algorithms, concentrating on multiplications and divisions which require a higher magnitude in computing time than additions and subtractions and assuming classical arithmetic algorithms. From the results it is observed that if a division is counted as two multiplications then the ratio of the number of multiplications and divisions in the one and two steps method is 8:6 which is about 1.6. In general the two step algorithm is about fifty percent more efficient than the one step algorithm [38].

In his Ph.D. thesis, McClellan [39] presents algorithms for computing the exact solution of systems of linear equations with integral polynomial coefficients. According to McClellan [40], the application of modular techniques to the solutions of linear equations and related problems have produced superior algorithms. McClellan [41] works on an extensive empirical study in comparing three most important algorithms for computing the exact solution of systems of linear equations with integral polynomial coefficients, which are the rational Gauss elimination algorithm, exact division elimination algorithm and modular algorithm. Various theoretical analysis and empirical studies of the computing times of these algorithms, much of which involved dense systems have been done earlier but the empirical results were incomplete and inconclusive. McClellan's work [41] presents a unified theoretical and empirical comparisons. The extensive empirical studies illustrate the superiority of the

modular algorithm for completely dense problems, in agreement with the analytical results. When the polynomials are sparse, however, a faster different technique via minor expansion has been developed.

2.5 Concluding Remarks

From the above discussion the modular method seems to be a very important and useful exact method that is applied to the general class or dense problems. However, according to [43], the design of the modular method is a challenging task since it is difficult to implement the method properly to meet the desired complexity time as well as to minimize storage allocation. In CA systems like MAPLE or AXIOM, the modular arithmetic is done using the hardware integer arithmetic instructions, which is written in the systems implementation language, that is in C or Lisp or assembler. The package *modp1* implemented in MAPLE provides a new data structure for polynomial arithmetic in $Z_n[x]$, including routines for multiplication, quotient and remainder, GCD and resultant, evaluation and interpolation. Using this package, Monagan codes the modular GCD method and find that the modular method is almost always better than the GCDHEU method that it is now the default method replacing the heuristic method in MAPLE.

Likewise, SACLIB library functions have provided efficient modular arithmetic functions for the applications of many successful modular algorithms such as Brown's modular integral polynomial GCD algorithm and Encarnacion's improved modular polynomial GCD algorithm for univariate polynomials over algebraic number fields. With these achievements the difficulty in the implementation of the modular arithmetic has been overcome. However, this challenge may lead to the design of many different techniques that differ in performance, depending on the classes of problems that the techniques apply (refer to [31] and [17]). The main task remains to carefully design and implement a number of different possible modular techniques which is the best of its type, to suit a particular class of problems. The timing results from the implementation of these techniques can then be compared to select the most competitive technique for the class of problems.

CHAPTER III

THE COMRADE MATRIX AND POLYNOMIAL GCD COMPUTATIONS

3.1 Introduction

This chapter reviews the theoretical background that have been established by Barnett in [6] and [7] regarding the application of the comrade matrix in the computation of the GCD of two polynomials relative to an orthogonal basis. Section 3.2 defines the relation that exists between the orthogonal basis and the comrade matrix. More theoretical results are investigated in Section 3.3 which includes the practical aspects of its implementation. A general algorithm for computing the GCD of polynomials using the comrade matrix approach is given in Section 3.4, with elaborations on algorithms for constructing the comrade matrix and the corresponding systems coefficient matrix. The algorithms apply rational number and multiprecision operations of SACLIB's library functions. The theoretical computing time analysis for these algorithms are investigated and included.

3.2 Orthogonal Basis and Congenial Matrices

Consider the set of orthogonal polynomials over the rational numbers given by

$$p_i(x) = \sum_{j=0}^{i} p_{ij} x^j, \tag{3.1}$$

with the set $\{p_i(x)\}$ known as an orthogonal basis satisfying the relationships (1.3)

$$\begin{aligned} p_0(x) &= 1, \\ p_1(x) &= \alpha_0 x + \beta_0, \\ p_{i+1}(x) &= (\alpha_i x + \beta_i) p_i(x) - \gamma_i p_{i-1}(x), \end{aligned}$$

for $i = 1, 2, .., n-1$ with $\alpha_i > 0$, $\gamma_i, \beta_i \geq 0$.

Any given nth degree integral polynomial $a(x)$ can be expressed uniquely as

$$a(x) = a_0 p_0(x) + a_1 p_1(x) + ... + a_n p_n(x),$$

Assume without loss of generality that $a_n = 1$ and for subsequent convenience define the monic polynomial [6]

$$\begin{aligned} \tilde{a}(x) &= a(x)/\alpha_0\alpha_1...\alpha_{n-1}, \\ &= \tilde{a}_0 + \tilde{a}_1 x + ... + x^n. \end{aligned} \tag{3.2}$$

We can assume without loss of generality that if $m = n$ then $b(x)$ can be replaced by [7]

$$(b_0 - \frac{a_0 b_n}{a_n})p_0(x) + ... + (b_{n-1} - \frac{a_{n-1} b_n}{a_n})p_{n-1}(x)$$

The comrade matrix associated with $a(x)$ [7] is given by

$$\begin{pmatrix} \frac{-\beta_0}{\alpha_0} & \frac{1}{\alpha_0} & \dots & 0 & \dots & \dots & \dots & 0 \\ \frac{\gamma_1}{\alpha_1} & \frac{-\beta_1}{\alpha_1} & \frac{1}{\alpha_1} & 0 & \dots & \dots & \dots & 0 \\ 0 & \frac{\gamma_2}{\alpha_2} & \frac{-\beta_2}{\alpha_2} & \frac{1}{\alpha_2} & \dots & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \dots & \dots & \vdots \\ 0 & 0 & 0 & 0 & \dots & \dots & \dots & \frac{1}{\alpha_{n-2}} \\ \frac{-a_0}{a_n\alpha_{n-1}} & \dots & \dots & \dots & \dots & \frac{-a_{n-3}}{a_n\alpha_{n-1}} & \frac{-a_{n-2}+a_n\gamma_{n-1}}{a_n\alpha_{n-1}} & \frac{-a_{n-1}-a_n\beta_{n-1}}{a_n\alpha_{n-1}} \end{pmatrix}, \tag{3.3}$$

has $a(x)$ of the form (3.2) as its characteristic polynomial. The comrade matrix generalizes the results for the special case of Chebyshev polynomials when $\alpha_i = 1, \beta_i = 0, \gamma_i = 1$ where the term colleague matrix for $a(x)$ is associated as analogue with the usual companion matrix

$$C = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ -\tilde{a}_0 & -\tilde{a}_1 & -\tilde{a}_2 & \dots & -\tilde{a}_{n-1} \end{pmatrix}, \tag{3.4}$$

in which case $\alpha_i = 1, \beta_i = 0, \gamma_i = 0$ for all i, Further generalization of the companion matrix corresponds to the confederate matrix, in which case the basis $p_i(x)$ are the more general real polynomials. The term congenial matrices which includes the colleague, comrade and confederate matrices is again introduced in [11].

3.3 GCD of Polynomials Relative to an Orthogonal Basis

Let

$$\begin{aligned}\tilde{d}(x) &= \tilde{d}_0 + \tilde{d}_1 x + ...x^k, \\ &= d_0 p_0(x) + d_1 p_1(x) + ... + p_k(x), \end{aligned} \tag{3.5}$$

be the monic GCD of $a(x)$ and $b(x)$ as in (1.4). If A is the comrade matrix given by (3.3), define

$$b(A) = b_0 I + b_1 p_1(A) + ... + b_m p_m(A), \tag{3.6}$$

then it is known that $k = n - \text{rank}(b(A))$ and the rows of $b(A)$ are given as

$$r_0 = [b_0, ..., b_{n-1}] \tag{3.7}$$

$$r_1 = r_0 p_1(A) \tag{3.8}$$

$$\vdots \tag{3.9}$$

$$r_{n-1} = r_0 p_{n-1}(A) \tag{3.10}$$

Using the defining relation (1.3), we obtain the rows in terms of the comrade matrix and the recurrence relation

$$\begin{aligned} r_0 &= (b_0, ..., b_{n-1}), \\ r_1 &= r_0(\alpha_0 A), \\ r_i &= r_{i-1}(\alpha_{i-1} A + \beta_{i-1} I) - \gamma_{i-1} r_{i-2}. \quad \text{for} \quad i = 2, ..., n-1 \end{aligned} \tag{3.11}$$

Theorem 3.1. *For $i = 1, 2, ..., n$ let c_i be the ith column of $b(A)$. The columns $c_{k+1}, ..., c_n$ are linearly independent and the coefficients $d_0, ..., d_{k-1}$ in (3.5) are given by*

$$c_i = d_{i-1} c_{k+1} + \sum_{k+2}^{n} x_{ij} c_j \quad i = 1, 2, ..., k \text{ for some } x_{ij}. \tag{3.12}$$

The k-systems of equations in (3.12) is described by the augmented matrix

$$\left(\begin{array}{ccccccc|ccccc} c_{k+1} & \vdots & c_{k+2} & \vdots & \dots & \vdots & c_n & c_1 & \vdots & \dots & \vdots & c_k \end{array} \right), \tag{3.13}$$

which is

$$\begin{pmatrix} c_{1,k+1} & c_{1,k+2} & \dots & c_{1,n} \\ c_{2,k+1} & c_{2,k+2} & \dots & c_{2,n} \\ \vdots & \vdots & \dots & \vdots \\ c_{n,k+1} & c_{n,k+2} & \dots & c_{n,n} \end{pmatrix} \begin{pmatrix} x_{i,k+1} \\ x_{i,k+2} \\ \vdots \\ x_{in} \end{pmatrix} = \begin{pmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ c_{ni} \end{pmatrix}, \tag{3.14}$$

for each $i = 1, 2, ..., k$ and $x_{i,k+1} = d_{i-1}$.

3.3.1 On The Construction of Systems of Equations

It is observed from (3.13) and (3.14) that the computation of the rank of $b(A)$ and the solution $d = (d_0, ..., d_{k-1})$, to (3.12) can be computed simultaneously. Rearrange the columns of matrix $b(A)$ so that the jth column of $b(A)$ is the $n-(j-1)$th column of a new matrix $L^{(0)}$,

$$l^0_{n-(j-1)} = c_j \quad \text{for} \quad j = 1, 2, ..., n. \tag{3.15}$$

Reducing $L^{(0)}$ to upper row echelon form by s steps gives the matrix

$$L^{(s)} = \begin{pmatrix} l^{(s)}_{11} & l^{(s)}_{12} & \dots & l^{(s)}_{1r} & \vdots & l^{(s)}_{1,r+1} & \dots & l^{(s)}_{1,r+k} \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & l^{(s)}_{rr} & \vdots & l^{(s)}_{r,r+1} & \dots & l^{(s)}_{r,r+k} \\ 0 & 0 & \dots & 0 & \vdots & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 & \vdots & 0 & \dots & 0 \end{pmatrix}, \tag{3.16}$$

such that $r = \text{rank}(L^{(0)})$. Let inv, be the inverse function. If $k = n - r$, the solution to the coefficients of $d(x)$ is given by

$$d_{k-i} = l^{(s)}_{r,r+i}\text{inv}(l^{(s)}_{rr}), \tag{3.17}$$

$$d_k = 1, \tag{3.18}$$

for each $i = 1, 2, ..., k$. If $r = n$ then degree of $d(x)$=0, which implies that GCD is a unit element.

3.4 A General Algorithm

Let $a(x)$ and $b(x)$ be polynomials in the generalized form as in (1.4). The main algorithm that applies the comrade matrix approach consists of the following procedure:

1. constructing the comrade matrix (3.3) associated with $a(x)$.
2. applying the recursive equation (3.11) to construct the system of linear equation described in 3.3.1.
3. reducing the coefficient matrix obtained from step 2 to upper echelon form and finding the desired solution.

In constructing the nonmodular algorithms, the application of rational number arithmetic is assumed. The modular approach can be applied at step 3 alone or from an earlier step. In this section we therefore present the algorithms composing steps 1 and 2, considering the application of rational number arithmetic that will commonly be used in the modular and nonmodular algorithms developed in later chapters.

3.4.1 On the Construction of the Comrade Matrix

The parameter lists for the algorithm CDEMOFP are $alph, bet, gamm, a$ which is of variable type Word (in list representation) and h which is a beta digit integer. The arguments passed to these parameters have the following values:
$alph = (\alpha_0, ..., \alpha_{n-1})$ so, $alph(i) = \alpha_{i-1}$ for $1 \leq i \leq n$, is the ith element of list $alph$. Likewise is the declaration for the arguments bet and $gamm$ respectively.
$a = (a_0, ..., a_{n-1}, 1)$ such that each a_i is the coefficient of a(x) in the form (1.4) and $a(i+1) = a_i$ for $i = 0$ to $n-1$ and $a(n+1) = 1$.
Let $A \in \mathrm{M}(n, n, Q)$ be a matrix in array representation such that if $a_{ij} = r/s \in Q$ then $A_{i-1,j-1,0} = A[i-1][j-1][0] = r$ and $A_{i-1,j-1,1} = A[i-1][j-1][1] = s$.

Algorithm CDEMOFP

/*The algorithm construct the comrade matrix (3.3)*/

begin
Step 1: set $a_{ij} = 0$ for each $i = 1, ..., n$ and $j = 1, ..., n$.
for $i = 0$ to $n-1$ and $j = 0$ to $n-1$, **do** set $A_{i,j,0} = 0$ and $A_{i,j,1} = 1$
Step 2: assign the diagonal entries for rows 1 to $n-1$
for $i = 0$ to $n-2$ **do**
$t \leftarrow \text{RNQ}(-bet(i+1), alph(i+1))$
if $t = 0$ **then** continue; **else** $A_{i,i,0} \leftarrow t(1)$ and $A_{i,i,1} \leftarrow t(2)$
Step 3: assign the entries for $a_{i,i-1}$ for rows 2 to $n-1$.
for $i = 1$ to $n-2$ **do**
$t \leftarrow \text{RNQ}(gamm(i+1), alph(i+1))$
if $t = 0$ **then** continue; **else** $A_{i,i-1,0} \leftarrow t(1)$ and $A_{i,i-1,1} \leftarrow t(2)$
Step 4: assign the entries for $a_{i,i+1}$ for rows 1 to $n-1$.
for $i = 0$ to $n-2$ **do**
if $alph(i+1) = 0$ **then** continue
else $A_{i,i+1,0} \leftarrow alph_2(i+1)$ and $A_{i,i+1,1} \leftarrow alph_1(i+1)$
Step 5: assign the entries for row n in columns $j = 1$ to $n-2$
for $j = 0$ to $n-3$ **do**
$t \leftarrow \text{RNQ}(-a(j+1), alph(n))$
if $t = 0$ **then** continue; **else** $A_{n-1,j,0} \leftarrow t(1)$ and $A_{n-1,j,1} \leftarrow t(2)$
Step 6: assign the entries $a_{n,n-1}$ and $a_{n,n}$.
for $j = n-2$ to $n-1$ **do**
if $j = n-2$ **then** $t \leftarrow \text{RNQ}(\text{RNSUM}(-a(j+1), gamm(n)), alph(n))$
if $j = n-1$ **then** $t \leftarrow \text{RNQ}(\text{RNSUM}(-a(j+1), -bet(n)), alph(n))$
if $t = 0$ **then** continue; **else** $A_{n-1,j,0} \leftarrow t(1)$ and $A_{n-1,j,1} \leftarrow t(2)$
Step 7: convert array form to list form: A convert to $comL$
Step 8: output($comL$)
end

In addition to the properties of the length of the integers given in [20] we derive the following property that will be applied in the succeeding computing time analysis.

Proposition 3.1. *Let $x, y \in Z$ such that $\text{L}(x) \sim \text{L}(y) \sim 1$. Let a_i be the coefficient of*

a in the input set $\mathrm{P}(u, v, n, (p_i)_{i=0}^{n})$ *described in Chapter 1 such that* (p_i) *is the set of Legendre basis. Then* $\mathrm{L}(a_i \pm \frac{x}{y}) \preceq \mathrm{L}(\frac{u}{v} + 1)$.

Proof:

$$\begin{aligned} \mathrm{L}(a_i \pm \frac{x}{y}) &\leq \mathrm{L}(\frac{u}{v} + \frac{x}{y}) \\ &= \mathrm{L}(\frac{yu + vx}{vy}) \\ &\preceq \mathrm{L}(u) + \mathrm{L}(v) \sim \mathrm{L}(\frac{u+v}{v}) \end{aligned}$$

Theorem 3.2. *For the polynomials a and b in* $\mathrm{P}(u, v, n, (p_i)_{i=0}^{n})$ *such that* (p_i) *is the set of Legendre basis,* $t^{+}_{\mathrm{CDEMOFP}}(u, v, n) \preceq n^2 + n\mathrm{L}(u) + \mathrm{L}(u)\mathrm{L}(\mathrm{L}(u)) + \mathrm{L}(v)$.

Proof: The cost of each step consists of the total costs of the operations involved and the cost of copying the digits of the integers involved.

1. Step 1 involves copying n^2 entries. Each entry comprises of two components which are 0 and 1. Therefore $t_1 \sim n^2$.

2. Since $t = 0$ in step 2, $t_2 \sim n$.

3. $t_{\mathrm{RNQ}}(\gamma_i, \alpha_i) \sim 1$. $\sum_{i=1}^{n-2} t_{\mathrm{RNQ}}(\gamma_i, \alpha_i) \preceq n$. For each $1 \leq i \leq n-2$, $\frac{\gamma_i}{\alpha_i} = \frac{i}{2i+1}$. The total cost of copying the numerators and denominators of the entries of the comrade matrix $A(i, i-1)$ for $1 \leq i \leq n-2$ is denominated by $\mathrm{L}(2n+1)\ \mathrm{L}(n)$. $t_3 \preceq n + \mathrm{L}(n) \sim n$.

4. For $0 \leq i \leq n-2$, $\frac{1}{\alpha_i} = \frac{i+1}{2i+1}$. The cost of copying the numerators and denominators equals $\sum_{i=0}^{n-2} \mathrm{L}(i+1) + \sum_{i=0}^{n-2} \mathrm{L}(2i+1) \sim \mathrm{L}(2n-1) \sim 1$Therefore, $t_4 \sim 1$.

5. For $0 \leq j \leq n-3$, the step computes the values $A(n-1, j) = \frac{-a_j}{\alpha_{n-1}}$. $\mathrm{L}(\mathrm{num}(-a_j)) \leq \mathrm{L}(u)$ and $\mathrm{L}(\mathrm{den}(-a_j)) \leq \mathrm{L}(v)$. $\alpha_{n-1} = \frac{2n-1}{n}$. Let $d = \mathrm{L}(\gcd(u, 2n-1))$. $d \leq \mathrm{L}(2n-1) \sim 1$. Let $e = \mathrm{L}(\gcd(v, n))$. $e \leq \mathrm{L}(n) \sim 1$. Therefore $k = \min(d, e) = 1$. $m = \max(\mathrm{L}(-a_j), \mathrm{L}(\alpha_{n-1})) = \mathrm{L}(u)$ and

$n = \min(\mathrm{L}(-a_j), \mathrm{L}(\alpha_{n-1})) = 1$. From Theorem **??**, $t_{RNQ}(-a_j, \alpha_{n-1}) \preceq \mathrm{L}(u)$. For each $0 \leq j \leq n-3$, the cost of copying the numerators and denominators of $A(n-1, j)$ is dominated by $\mathrm{L}(u)$. The total cost of all these operations repeated $n-3$ times gives $t \preceq n\mathrm{L}(u)$.

6. Let $x = -a_{n-2} + \gamma_{n-1}, y = -a_{n-1} - \beta_{n-1}$. We know that $t_{\mathrm{RNNEG}}(a_{n-2}) \preceq \mathrm{L}(u)$ and $t_{\mathrm{RNSUM}}(-a_{n-2}, \gamma_{n-1}) \preceq \mathrm{L}(u)\mathrm{L}(\mathrm{L}(u))$. Since both of the numerator and denominator of γ_{n-1} is of $\mathrm{L}(1)$ and $\mathrm{L}(\mathrm{num}(-a_{n-2})) \leq \mathrm{L}(u)$, $\mathrm{L}(\mathrm{den}(-a_{n-2})) \leq \mathrm{L}(v)$, from Proposition 3.1, $\mathrm{L}(x) \preceq \mathrm{L}(u) + \mathrm{L}(v)$. From $\mathrm{L}(\alpha_{n-1}) = 1$, this gives $t_{\mathrm{RNQ}}(x, \alpha_{n-1}) \preceq \mathrm{L}(u) + \mathrm{L}(v)$. The total cost in computing $\dfrac{x}{\alpha_{n-1}}$ is thus dominated by $\mathrm{L}(u)\mathrm{L}(\mathrm{L}(u)) + \mathrm{L}(v)$. Similarly we obtain $t_{\mathrm{RNNEG}}(a_{n-1}) \preceq \mathrm{L}(u)$. Since $\beta_i = 0$ for all i, $t_{\mathrm{RNSUM}}(-a_{n-1}, -\beta_{n-1}) \sim \mathrm{L}(u)$. From $\mathrm{L}(-a_{n-1} - \beta_{n-1}) \leq \mathrm{L}(u) + \mathrm{L}(v)$, we obtain $t_{\mathrm{RNQ}}(y, \alpha_{n-1}) \preceq \mathrm{L}(u) + \mathrm{L}(v)$. So, the total cost of computing the values of $\dfrac{y}{\alpha_{n-1}}$ is dominated by $\mathrm{L}(u)$. Now let $a = \dfrac{u}{v} + \dfrac{n-1}{n}$. $\mathrm{L}(x) \leq \mathrm{L}(a)$. $\dfrac{a}{\alpha_{n-1}} = \dfrac{nu + v(n-1)}{v(2n-1)} \leq \dfrac{u+v}{v}$. Therefore, the cost of copying their numerator and denominator of $\dfrac{x}{\alpha_{n-1}}$ is dominated by $\mathrm{L}(u) + \mathrm{L}(v)$. The cost of copying the numerator and denominator of $\dfrac{y}{\alpha_{n-1}}$ is dominated by $\mathrm{L}(u)$. This gives $t_6 \preceq \mathrm{L}(u)\mathrm{L}(\mathrm{L}(u)) + \mathrm{L}(v)$.

7. For $0 \leq i \leq n-2$ and $0 \leq j \leq n-2$, $\mathrm{L}(A(i,j)) \sim 1$. For $0 \leq j \leq n-3$, $\mathrm{L}(A(n-1,j)) \leq \mathrm{L}(\dfrac{u}{v}) = \mathrm{L}(u)$. From step 6, $\mathrm{L}(A(n-1,n-2)) \leq \mathrm{L}(\dfrac{u+v}{v}) \preceq \mathrm{L}(u) + \mathrm{L}(v)$. Also $\mathrm{L}(A(n-2,n-1)) \leq \mathrm{L}(\dfrac{u}{v}) = \mathrm{L}(u)$. Thus, $t_7 \preceq (n-1)^2 + n\mathrm{L}(u) + \mathrm{L}(v) \sim n^2 + n\mathrm{L}(u) + \mathrm{L}(v)$. From steps 1 to 7, $t_{\mathrm{CDEMOFP}}(u, v, n) \preceq n^2 + n\mathrm{L}(u) + \mathrm{L}(u)\mathrm{L}(\mathrm{L}(u)) + \mathrm{L}(v)$, from which Theorem 3.2 is immediate.

3.4.2 The Coefficient Matrix for a System of Equations

The algorithm applies the recurrence relation (3.11) to construct the coefficient matrix described in Theorem 3.12 and the new coefficient matrix (3.15). The parameter lists for the algorithm CSFCDEM are $alph, bet, gamm, b$ which is of variable

type Word (in list representation) and h which is a beta digit integer. The arguments passed to these parameters have the following values:
$alph = (\alpha_0, ..., \alpha_{n-1})$ so that $alph(i) = \alpha_{i-1}$ for $1 \leq i \leq n$ is the ith element of list $alph$. The declaration for bet and $gamm$ is similar.
$b = (b_0, ..., b_m, ..., b_{n-1})$ such that each b_i is the coefficient of b(x) in the form (1.4). So b is the list representing an $n \times 1$ vector. $CMDE$ is the comrade matrix obtained from Algorithm CDEMOFP.

Algorithm CSFCDEM

/*To construct the coefficient matrix in the form (3.15) via (3.11)*/

begin
Step 1: obtain the first row of coefficient matrix from b.
$CS \leftarrow \text{COMP}(b, NIL)$
Step 2: initializing and obtaining the second row of CS.
$R0_{1\times n} \leftarrow CS$
$R1_{1\times n} \leftarrow \text{MRNPROD}(R0_{1\times n}, \text{MRNSPR}(alph(1), CMDE_{n\times n}))$
$CS \leftarrow \text{COMP}(\text{FIRST}(R1_{1\times n}), CS)$
Step 3: apply recurrence equation for remaining rows of CS.
Step3a: composing ith row
for $i = 3$ to n **do**
$S_{n\times n} \leftarrow \text{MRNSPR}(alph(i-1), CMDE_{n\times n})$
$T_{n\times n} \leftarrow \text{MRNSPR}(bet(i-1), I_{n\times n})$
$U_{1\times n} \leftarrow \text{MRNSPR}(-gamm(i-1), R0_{1\times n})$
$R_{1\times n} \leftarrow \text{MRNSUM}(\text{MRNPROD}(R1_{1\times n}, \text{MRNSUM}(S_{n\times n}, T_{n\times n})), U)$
$CS \leftarrow \text{COMP}(\text{FIRST}(R_{1\times n}), CS)$
Step3b: initialization: to set $R0 = R1$
$R0 \leftarrow NIL$ and $R1 \leftarrow \text{FIRST}(R1_{1\times n})$ /*$R1$ in vector representation*/
for $j = 1$ to n **do**
$R0 \leftarrow \text{COMP}(R1(j), R0)$ /*$R0$ is in vector representation*/
$R0 \leftarrow \text{INV}(R0)$ and $R0_{1\times n} \leftarrow \text{COMP}(R0, NIL)$
Step3c: to set $R1 = R$
(similar steps to Step 3b)
$CS \leftarrow \text{INV}(CS)$
Step4: integerize CS

Step5: sorting columns of CS to obtain $CSNEW$ as in (3.15)
Step6: output($CSNEW$)
end

Referring to algorithm CSFCDEM, the matrix $R0_{1\times n}$ comprising a single row is represented as a list comprising of a single list such as $((r_1, ..., r_n))$. On the other hand $R0$ is in vector form and is therefore represented as a single list of objects, that is $(r_1, ..., r_n)$. With this representation, $R0_{1\times n}$ can be obtained by applying the function COMP($R0, NIL$) where NIL is the empty list (). The conversion from vector form to matrix form is required in the application of MRNSPR, MRNSUM or MRNPROD which are the matrix rational number scalar product, matrix rational number product and matrix rational number sum algorithm respectively. These auxiliary rational number matrix operation algorithms are not presently available in SACLIB's library and are constructed for its application in the algorithms CDEMOFP and CSFCDEM.

The construction of algorithm CSFCDEM is based upon the recursive equations [47],

$$R_0 = (b_0, ..., b_m, b_{m+1} + .. + b_{n-1}), \tag{3.19}$$

$$R_1 = R_0(\alpha_0 A), \tag{3.20}$$

$$R_i = R_{i-1}(\alpha_{i-1} A) - \gamma_{i-1} R_{i-2}, \tag{3.21}$$

for $2 \leq i \leq n-1$. In (3.19), $b_j = 0$ for $m < j \leq n-1$ if $m < n-1$. In (3.21), we assume that $\beta_i = 0$ for all i and is omitted from the equation. We begin by first computing the length of R_i for each $0 \leq i \leq n-1$.

Lemma 3.1. *Let the polynomials a and b be in* $\mathrm{P}(u, v, n, (p_i)_{i=0}^n)$ *such that (p_i) is the set of Legendre basis. Let A be the comrade matrix associated with a. Referring to (3.19) - (3.21),* $\mathrm{L}(R_i(j)) \leq \mathrm{L}(\frac{f(u,v)}{g(v)}) \preceq \mathrm{L}(u) + \mathrm{L}(v)$, *for $0 \leq i \leq n-1$ and $1 \leq j \leq n$ where f and g are polynomials in the variables u or v with single precision coefficients and with degree bound $i+1$.*

Proof: For $1 \leq i, j \leq n$, let c_{ij} be the entries of the comrade matrix obtained from algorithm CDEMOFP. But referring to steps 6 and 7 in the proof of Theorem 3.2,

$\mathrm{L}(c_{ij}) \leq \mathrm{L}(z)$ such that $z = \dfrac{u+v}{v}$. Considering the values of the numerators and denominators that will contribute to the greatest possible length and replacing an element of single precision with 1, we have for $1 \leq j \leq n$, $\mathrm{L}(R_0(j)) \leq \mathrm{L}(\dfrac{u}{v})$ and $\mathrm{L}(R_1(j)) \leq \mathrm{L}(\dfrac{u}{v} * z)$. For a fixed j, the function whose length dominates $\mathrm{L}(R_0(j))$ and $\mathrm{L}(R_1(j))$ is a quotient of polynomials in u and v whose coefficients of length 1 and has degree bounds 1 and 2 respectively. This gives $\mathrm{L}(R_2) \leq \mathrm{L}(y)$ such that $y = \dfrac{u}{v} * z^2 + \dfrac{u}{v}$, a quotient of polynomials in u and v with coefficient of length 1 and degree bound equals to 3. Proceeding by induction, suppose that for $1 \leq j \leq n$, $\mathrm{L}(R_{l-1}(j)) \leq \mathrm{L}(x1)$ and $\mathrm{L}(R_{l-2}(j)) \leq \mathrm{L}(x2)$ such that

$$\begin{aligned} x1 &= \frac{f_{l-1}(u,v)}{g_{l-1}(v)}, \\ x2 &= \frac{f_{l-2}(u,v)}{g_{l-2}(v)}, \end{aligned}$$

such that f_{l-k}, g_{l-k}, $k = 1, 2$, has single precision coefficients and degree bounds equal to l and $l-1$ respectively. Then $\mathrm{L}(R_l)(j) \leq \mathrm{L}(y)$ such that $y = x1 * z + x2$. Hence, y is of the form $\dfrac{f(u,v)}{g(v)}$, where the coefficients of f and g are single precision integers with degree bound equals to $l+1$. Applying the properties of the length of the nonzero integers, that is $\mathrm{L}(a \pm b) \preceq \mathrm{L}(a) + \mathrm{L}(b)$, $\mathrm{L}(ab) \sim \mathrm{L}(a) + \mathrm{L}(b)$ and $\mathrm{L}(a^b) \sim b\mathrm{L}(a)$, $a \geq 2$ in the later, we then have for $0 \leq i \leq n-1$, and $1 \leq j \leq n$, $\mathrm{L}(R_i(j)) \preceq \mathrm{L}(u) + \mathrm{L}(v)$

Theorem 3.3. *Let $a, b \in \mathrm{P}(u, v, n, (p_i)_{i=0}^n)$ with (p_i) the Legendre basis. Then $t^+_{\mathrm{CSFCDEM}}(u, v, n) \preceq n^3\mathrm{L}^2(u)(\mathrm{L}(q) + \mathrm{L}(r))$ such that $q = \max(r, n\mathrm{L}(v))$ and $r = \mathrm{L}(u) + \mathrm{L}(v)$.*

Proof: Let $t(V)$ be the computing time to compute the components of the vector V and let the jth component of the vector V be denoted $V(j)$. Let $A = (a_{ij})$ be comrade matrix obtained from Algorithm CDEMOFP.

1. From $\mathrm{L}(b_i) \leq \mathrm{L}(u)$, $0 \leq i \leq m$, $t_1 \preceq m\mathrm{L}(u)$.

2. $t(R0) \preceq m\mathrm{L}(u)$. To compute

$$t(R1) = t_{\mathrm{MRNSPR}}(\alpha_0, A) + t_{\mathrm{MRNPROD}}(R0, \alpha_0 A),$$

(a) $t_{\mathrm{MRNSPR}}(\alpha_0, A) = \sum_{i=1}^{n-1}\sum_{j=1}^{n} t_{\mathrm{RNPROD}}(\alpha_0, a_{ij}) + \sum_{j=1}^{n} t_{\mathrm{RNPROD}}(\alpha_0, a_{nj})$ where $\mathrm{L}(a_{ij}) \sim 1$ for $1 \le i \le n-1$ and $1 \le j \le n$ and $\mathrm{L}(a_{nj}) \preceq \mathrm{L}(u) + \mathrm{L}(v)$ for $1 \le j \le n$. Thus, $t_{\mathrm{MRNSPR}}(\alpha_0 A) \preceq n^2 + n(\mathrm{L}(u) + \mathrm{L}(v))\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$.

(b) Let $1 \le j \le n$. The jth column of $R1$ is given by $\sum_{i=1}^{n} R0(i)\alpha_0 a_{ij}$. The cost of computing $R1(j)$ is given by

$$\sum_{i=1}^{n} t_{\mathrm{RNPROD}}(R0(i), \alpha_0 a_{ij}) + \sum_{i=1}^{n-1} t_{\mathrm{RNSUM}}(R0(i)\alpha_0 a_{ij}, R0(i+1)\alpha_0 a_{i+1,j}).$$

For each $1 \le i \le n$, $t_{\mathrm{RNPROD}}(R0(i), \alpha_0 a_{ij}) \preceq$
$(\mathrm{L}(u) + \mathrm{L}(v))\mathrm{L}(u)\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v)) \preceq \mathrm{L}^2(u)\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$. For each j, the total cost of computing these products is dominated by
$n\mathrm{L}^2(u)\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$. Taking $\mathrm{L}(R0(i)\alpha_0 a_{ij}) \preceq \mathrm{L}(u) + \mathrm{L}(v)$ for each i and j, $t_{\mathrm{RNSUM}}(R0(i)\alpha_0 a_{ij}, R0(i+1)\alpha_0 a_{i+1,j}) \preceq (\mathrm{L}(u) + \mathrm{L}(v))^2\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v)) \preceq \{\mathrm{L}^2(u) + \mathrm{L}^2(v)\}\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$. The cost of the $n-1$ sums is dominated by $n\{\mathrm{L}^2(u) + \mathrm{L}^2(v)\}\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$. Summing the cost for all $1 \le j \le n$, the total cost of computing the vector $R1$ is such that
$t(R1) \preceq n^2\{\mathrm{L}^2(u) + \mathrm{L}^2(v)\}\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$.

3. For $1 \le i, j \le n$, the respective length of $R_{i-1}(j)$ and $\alpha_{i-1}c_{ij}$ is dominated by $\mathrm{L}(u) + \mathrm{L}(v)$. For $2 \le i \le n-1$, we compute the cost of computing row $i+1$ given by the equation

$$R_i = R_{i-1}\alpha_{i-1}A - \gamma_{i-1}R_{i-2}.$$

For a fix value of i, the jth component of R_i is given by

$$R_i(j) = \sum_{k=1}^{n} R_{i-1}(k)\alpha_{i-1}a_{kj} - \gamma_{i-1}R_{i-2}(j)$$

The cost of computing $\sum_{k=1}^{n} R_{i-1}(k)\alpha_{i-1}a_{kj}$ is dominated by
$n(\mathrm{L}(u) + \mathrm{L}(v))^2\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))$.
$\mathrm{L}(\sum_{k=1}^{n} R_{i-1}(k)\alpha_{i-1}a_{kj}) \sim \sum_{k=1}^{n} \mathrm{L}(R_{i-1}(k)\alpha_{i-1}a_{kj}) \preceq n(\mathrm{L}(u) + \mathrm{L}(v))$.
$\mathrm{L}(\gamma_{i-1}R_{i-2}(j)) \preceq \mathrm{L}(u) + \mathrm{L}(v)$. Therefore,

$$t_{\mathrm{RNSUM}}(\sum_{k=1}^{n} R_{i-1}(k)\alpha_{i-1}a_{kj}, -\gamma_{i-1}R_{i-2}(j)) \preceq n(\mathrm{L}(u) + \mathrm{L}(v))^2\mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v)).$$

For each $2 \leq j \leq n-1$, the total cost of computing $R_i(j)$ is dominated by $n^2(\mathrm{L}(u)+\mathrm{L}(v))^2\mathrm{L}(\mathrm{L}(u)+\mathrm{L}(v))$. It follows that the cost of computing the vector R_i for $2 \leq i \leq n-1$ is dominated by $n^3(\mathrm{L}(u)+\mathrm{L}(v))^2\mathrm{L}(\mathrm{L}(u)+\mathrm{L}(v)) = t'_3$

4. The integerization of CS in step 4 of algorithm CSFCDEM transforms the rational number matrix CS to its row equivalent integer matrix $CSNEW$. For each row $i = 1$ to n,

   (a) find the integer least common multiple (Saclib ILCM) of the n denominators of the n column elements. From Lemma 3.1, the denominators of $R_i(j)$ for $0 \leq i \leq n-1$ and $1 \leq j \leq n$, that will constitute the maximum length of the elements of the comrade matrix is a polynomial $g(v)$ with single precision coefficients and degree, that is, dominated by $\mathrm{L}(v)$. Suppose that at the first $k-2$ iterations, we have computed the LCM of the integers $x_1x_2...x_{k-2}$ and x_{k-1} where x_i constitute the denominators of a certain row of A and suppose that the LCM is of the maximum possible length, which is equal to $x_1x_2...x_{k-1}$. At the end of the $k-1$ iteration we compute the least common multiple, that is $\mathrm{lcm}(x_1x_2...x_{k-1}, x_k) = \dfrac{x_1x_2...x_{k-1}x_k}{\gcd(x_1x_2...x_{k-1}, x_k)}$, (refer to Theorem 1.4.8 in [48]). Applying the Euclidean algorithm to compute the GCD and applying the results of Theorem 3 in [20], we have

$$\begin{aligned} t_{\mathrm{M}}(x_1x_2...x_{k-1}, x_k) &\sim \mathrm{L}(x_1x_2...x_{k-1})\mathrm{L}(x_k) \\ &\sim k\mathrm{L}^2(v) \end{aligned} \tag{3.22}$$

   and

$$\begin{aligned} t_{\mathrm{E}}(x_1x_2...x_{k-1}, x_k) &\preceq \mathrm{L}(x_k)\{\mathrm{L}(x_1...x_{k-1}) - \mathrm{L}(y) + 1\} \\ &\preceq \mathrm{L}(x_k)\{\mathrm{L}(x_1) + ... + \mathrm{L}(x_{k-1}) - \mathrm{L}(y) + 1\} \\ &\preceq k\mathrm{L}^2(v) \end{aligned} \tag{3.23}$$

   where $y = \gcd(x_1x_2...x_{k-1}, x_k)$ and $\mathrm{L}(x_i), \mathrm{L}(y) \preceq \mathrm{L}(v)$. Likewise, it can be calculated from the results for the computing time of the classical division algorithm [20], which gives $t_{\mathrm{D}}(x_1x_2...x_k, y) \preceq k\mathrm{L}^2(v)$. Thus, the computing time for the $n-1$ iterations in computing the LCM of a row is dominated by $n\mathrm{L}^2(v)$. The computing time for n rows is then dominated by $n^2\mathrm{L}^2(v)$.

(b) multiply the LCM of the denominators of row i with each of the elements in row i to integerize the row. Suppose that the LCM of a row is $x_1x_2...x_n$ whose length is therefore dominated by $n\mathrm{L}(v)$. The length of each numerator of A is dominated by $\mathrm{L}(u)+\mathrm{L}(v)$. For each $1 \leq i,j \leq n$,

$$t_{\mathrm{RNPROD}}(y, a_{ij}) \preceq n(\mathrm{L}(u)+\mathrm{L}(v))\mathrm{L}(v)\mathrm{L}(\max(\mathrm{L}(u)+\mathrm{L}(v), n\mathrm{L}(v)))$$

which gives a total cost of $n^3(\mathrm{L}(u)+\mathrm{L}(v))\mathrm{L}(v)\mathrm{L}(\max(\mathrm{L}(u)+\mathrm{L}(v), n\mathrm{L}(v)))$.

The analysis implies that t_4 is dominated by $n^3\{\mathrm{L}(u)\mathrm{L}(v)+\mathrm{L}^2(v)\}\mathrm{L}(q)$ such that $q = \max(\mathrm{L}(u)+\mathrm{L}(v), n\mathrm{L}(v))$.

$t_{\mathrm{CSFCDEM}}(x) \preceq n^3(\mathrm{L}^2(u)+\mathrm{L}^2(v))\mathrm{L}(q) + n^3(\mathrm{L}^2(u)+\mathrm{L}^2(v)+\mathrm{L}(u)\mathrm{L}(v))\mathrm{L}(r)$. Thus the computing time for CSFCDEM is cubic, that is

$$t_{\mathrm{CSFCDEM}}(x) \preceq n^3\mathrm{L}^2(u)(\mathrm{L}(q)+\mathrm{L}(r)) \tag{3.24}$$

such that $q = \max(r, n\mathrm{L}(v))$ and $r = \mathrm{L}(u)+\mathrm{L}(v)$.

3.5 Concluding Remark

In the computing time analysis, we let P of the the set generalized polynomials relative to an orthogonal basis $(p_i(x))_{i=0}^n$ and decompose P into

$$\begin{aligned}\mathrm{P}(u,v,n,(p_i)_{i=0}^n) &= \{a = \sum_{i=0}^{k} a_i p_i(x) \in P : |\mathrm{num}(a_i)| \leq u, 0 < \mathrm{den}(a_i) \leq v,\\ &\ni \gcd(u,v) = \gcd(\mathrm{num}(a_i), \mathrm{den}(a_i)) = 1 \text{ and } k \leq n\}.\end{aligned}$$

where num and den is the numerator and denominator of a_i respectively. For high degree polynomials the coefficients of a will involve multiprecision integers or rational numbers. However, the defining terms always remain single precision, which reduce the complexity of the computations involved in constructing the comrade matrix and the corresponding systems matrix. If the input polynomials belong to $\mathrm{P}(u,v,n,(p_i)_{i=0}^n)$, the computing time analysis shows that the cost of the algorithm for constructing the comrade matrix is dominated by n^2. The cost of the algorithm for constructing the systems matrix is given by $t_{\mathrm{CSFCDEM}}(x) \preceq n^3\mathrm{L}^2(u)(\mathrm{L}(q)+\mathrm{L}(r))$ such that $q = \max(r, n\mathrm{L}(v))$ and $r = \mathrm{L}(u)+\mathrm{L}(v)$. If $\mathrm{L}(q) > \mathrm{L}(r)$ and $q = n\mathrm{L}(v)$, then $t_{\mathrm{CSFCDEM}}(x) \preceq n^4\mathrm{L}^2(u)\mathrm{L}(v)$.

CHAPTER IV

THE NONMODULAR GENERALIZED POLYNOMIAL GCD COMPUTATIONS

4.1 Introduction

Referring to the general algorithm described in 3.4, steps 1 and 2 involve the application of algorithm CDEMOFP and algorithm CSFCDEM in the construction of the comrade matrix and the formulation of the coefficient of the corresponding systems of equations (3.12). These basic algorithms directly apply the rational number or multiprecision integer arithmetic which are based on the theories that contribute to its derivation. As a consequence, a natural approach to developing a nonmodular algorithm for the generalized polynomial GCD computation using the comrade matrix approach depends on the nonmodular approach to solving step 3 of the general algorithm, which reduces the matrix coefficient of the systems obtained from step 2 to its upper echelon form. The desired coefficients are then obtained in Q by a direct rational number computation in an equation involving entries of the last nonzero row of the echelon form described in the form (3.17) and (3.18) in 3.3.1.

At the beginning of this chapter we elaborate on the techniques involved in the exact computation of the row echelon form applied in step 3 of 3.4. At the end of this chapter the corresponding nonmodular algorithms for the computation of the GCD of generalized polynomials in an orthogonal basis are described.

4.2 The Computation of Row Echelon Form of a Matrix

In this section, we present the one and two step Gauss elimination method, which we referred to as the single or double step Gauss elimination method respectively.

4.2.1 Single Step Gauss Elimination

Let $A \in \mathrm{M}(m, n, Z)$ and

$$\begin{aligned} A^{(0)} &= [a_{ij}^{(0)}], \\ a_{00}^{-1} &= 1, \\ a_{ij}^{(k)} &= a_{kk}^{(k-1)} a_{ij}^{(k-1)} - a_{kj}^{(k-1)} a_{ik}^{(k-1)} / a_{k-1,k-1}^{(k-2)}, \end{aligned} \tag{4.1}$$

$k+1 \leq i \leq m, k+1 \leq j \leq n$.
for $k = 1, 2, ..., m-1$, with the convention that the elements $a_{lk}^{(k)}$, such that $k+1 \leq l \leq m$ is assumed to be zero at the end of step k. Using standard terminology, at step k, the element $a_{kk}^{(k-1)}$ is called the pivot element and row k is called the pivot row, (refer to [28]). In the following matrix reduction algorithms to its RE form, let $A_{i,j}$ be the abbreviation for the array $A[i][j]$. We then have for $1 \leq i \leq m, 1 \leq j \leq n$, $a_{ij} = A_{i-1,j-1}$.

Algorithm SSGSE

/*The algorithm reduces the matrix A to upper echelon form via (4.1).*/
Input:
A : array representation of an integral matrix of size $m \times n$.
Output:
Upper Echelon form of A

begin
Step1: (*initialization*)
  divisor $\leftarrow 1$
Step2: (*reduction step*)
  for $k = 0$ to $m-2$ **do**
    /**step 2a: find a nonzero pivot*/*
    $p \leftarrow k$
    while $A_{p,k} = 0$ and $p < m-1$ **do**
      $p \leftarrow p+1$
    if $p \leq m-1$ and $A_{p,k} \neq 0$ **then**
      if $p > k$ **then** interchange rows p and k

*/*step 2b: reduce so that the elements under pivot row in pivot column is zero*/*
for $i = k+1$ to $m-1$
for $j = k+1$ to $n-1$ **do**
$A_{i,j} \leftarrow A_{k,k}A_{i,j} - A_{i,k}A_{k,j}/divisor$
$A_{i,k} \leftarrow 0$
$divisor \leftarrow A_{k,k}$
/*since $p > m-1$, find pivot element in the next column*/
end

4.2.2 Double Step Gauss Elimination

The two step fraction free scheme [28] is described.

$$
\begin{aligned}
a_{00}^{-1} &= 1, && (4.2)\\
a_{ij}^{(0)} &= A^{(0)}, && (4.3)\\
c_0^{(k-2)} &= a_{k-1,k-1}^{(k-2)}a_{kk}^{(k-2)} - a_{k-1,k}^{(k-2)}a_{k,k-1}^{(k-2)}/a_{k-2,k-2}^{(k-3)}, && (4.4)\\
c_{i1}^{(k-2)} &= a_{k-1,k}^{(k-2)}a_{i,k-1}^{(k-2)} - a_{k-1,k-1}^{(k-2)}a_{ik}^{(k-2)}/a_{k-2,k-2}^{(k-3)}, && (4.5)\\
c_{i2}^{(k-2)} &= a_{k,k-1}^{(k-2)}a_{ik}^{(k-2)} - a_{kk}^{(i,k-1)}a_{k-2}^{(k-2)}/a_{k-2,k-2}^{(k-3)}, && (4.6)\\
a_{ij}^{k} &= c_0^{(k-2)}a_{ij}^{(k-2)} + c_{i1}^{(k-2)}a_{kj}^{(k-2)} + c_{i2}^{(k-2)}a_{k-1,j}^{(k-2)}/a_{k-2,k-2}^{(k-3)}, && (4.7)
\end{aligned}
$$

for $1 \le i \le m$ and $1 \le j \le n$;

$$
\begin{aligned}
a_{kk}^{(k-1)} &= c_0^{(k-2)}, && (4.8)\\
a_{kl}^{(k-1)} &= a_{kl}^{(k)}\\
&= a_{k-1,k-1}^{(k-2)}a_{kl}^{(k-2)} - a_{k-1,l}^{(k-2)}a_{k,k-1}^{(k-2)}/a_{k-2,k-2}^{(k-3)}, && (4.9)
\end{aligned}
$$

for $k+1 \le l \le n$,
which is applied for $k = 2, 4, ..., 2 * \text{floor}(\frac{m-1}{2})$. When n is even, the last elimination is performed by the single step method.

For the input matrix $A = (a_{ij}) \in \text{M}(m, n, \text{P}(d))$, let the vector valued function row be define such that $\text{row}(i) = (a_{i+1,1}, ..., a_{i+1,n})$ for each integer $0 \le i \le m-1$. The algorithm DSGSE is given as:

Algorithm DSGSE

/*The algorithm reduces the matrix A to upper echelon form via (4.7) and (4.9).*/
Input: A: array representation of an integral matrix of size $m \times n$.
Output: Upper Echelon form of A

begin
Step 1: $divisor \leftarrow 1$
Step 2: (*pivoting and reducing*)
for $(k = 1, k \leq 2 * \text{floor}(\frac{m-1}{2}) - 1; k+ = 2)$
Step 2a: (*find a nonzero pivot*) set $q = 1$
Step 2b: set $p = k$
while $p \leq m - 1$ **do**
$c_0 \leftarrow A_{k-1,k-1}A_{p,k} - A_{k-1,k}A_{p,k-1}/divisor$
if $c_0 \neq 0$ **then** break(i.e goto *Step 2c*)
else $p \leftarrow p + 1$
Step 2c:
if $p > m - 1$ and $k + q \leq m - 1$ **then**
interchange row $k - 1$ with row $k + q$
$q \leftarrow q + 1$
goto *Step 2b*
Step 2d:
if $k + q > m - 1$ **then** apply Algorithm SSGSE so that row $k - 1$ is the pivot row
Step 2e: (*elements in columns k-1 and k below respective pivot row set to 0*)
if $c_0 \neq 0$ and $p \leq m - 1$ **then**
if $p > k$ **then** interchange row k with row p
for $i = k + 1$ to $i = m - 1$ **do** (*work on row k+1 to row m-1*)
$c1_i \leftarrow A_{k-1,k}A_{i,k-1} - A_{k-1,k-1}A_{i,k}/divisor$
$c2_i \leftarrow A_{k,k-1}A_{i,k} - A_{k,k}A_{i,k-1}/divisor$
for $j = k + 1$ to $n - 1$ **do**
$A_{i,j} \leftarrow c_0A_{i,j} + c1_iA_{k,j} + c2_iA_{k-1,j}/divisor$
for $j = k - 1$ to k **do**
set $A_{i,j} = 0$
for $l = k + 1$ to $n - 1$ **do** (*work on row k*)
$A_{k,l} \leftarrow A_{k-1,k-1}A_{k,l} - A_{k-1,l}A_{k,k-1}/divisor$
$A_{k,k-1} \leftarrow 0$
$A_{k,k} \leftarrow c_0; \quad divisor \leftarrow A_{k,k}$
end

4.2.3 The Rational Number Gauss Elimination

The Gauss elimination approach in the preceding section reduces an integral matrix to its row echelon form by applying integer addition and multiplication and exact integer quotient, in order to preserve the integer entries of the echelon form matrix. It is also possible to reduce a matrix over the rational number without first transforming the matrix to a row equivalent integer matrix but the complexity of rational number arithmetic has to be applied. This approach shall not be omitted as it can be regarded as a basis of comparison with the less complex algorithms involving exact integer and modular arithmetic. The scheme for the rational number Gauss elimination takes a naive approach in reducing the elements underneath the pivot element to zero. Let $A \in \mathrm{M}(m, n, Z)$ be such that

$$\begin{aligned} A^{(0)} &= [a_{ij}^{(0)}], \\ a_{ij}^{(k)} &= a_{kk}^{(k-1)} a_{ij}^{(k-1)} - a_{kj}^{(k-1)} a_{ik}^{(k-1)}, \end{aligned} \tag{4.10}$$

$k+1 \leq i \leq m, k+1 \leq j \leq n$.
for $k = 1, 2, ..., m-1$, with the convention that the elements $a_{lk}^{(k)}$ such that $k+1 \leq l \leq m$ is assumed to be zero at the end of step k as adopted by the single step Gauss elimination scheme whereby at step k, the element $a_{kk}^{(k-1)}$ is called the pivot element and row k is called the pivot row.

Algorithm RNGSE

/*The algorithm reduces the matrix A to upper echelon form via (4.1).*/
Input:
A : array representation of an integral matrix of size $m \times n$.
Output:
Upper Echelon form of A

begin

for $k = 0$ to $m - 2$ **do**
    /**step 1a: find a nonzero pivot**/
    $p \leftarrow k$
    while $A_{p,k} = 0$ and $p < m - 1$ **do**
        $p \leftarrow p + 1$
    if $p \leq m - 1$ and $A_{p,k} \neq 0$ **then**
        if $p > k$ **then** interchange rows p and k
        /**step 1b: reduce so that the elements under pivot row in pivot column is zero**/
        for $i = k + 1$ to $m - 1$ **do**
            for $j = k + 1$ to $n - 1$ **do**
                $A_{i,j} \leftarrow A_{k,k}A_{i,j} - A_{i,k}A_{k,j}$
            $A_{i,k} \leftarrow 0$
    /*since $p > m - 1$, find pivot element in the next column**/
end

4.3 On the Nonmodular OPBGCD Algorithms

From the general algorithm of 3.4, a general nonmodular algorithm for the exact computation of the GCD of two polynomials relative to an orthogonal basis is described.

Algorithm OPBGCD (Orthogonal-Basis Polynomial GCD)

Input:
$\alpha = (\alpha_0, ..., \alpha_{n-1})$, $\beta = (\beta_0, ..., \beta_{n-1})$ and $\gamma = (\gamma_0, ..., \gamma_{n-1})$.
The polynomials, $a = (a_0, a_1, ..., a_{n-1}, 1)$, $b = (b_0, b_1, ..., b_m, ..., b_{n-1})$.
$h = n - 1$.
Output:
The monic generalized polynomial $c = \gcd(a, b)$ of degree k.

---

begin
Step1: (Calculation of the comrade matrix)
$CMR \leftarrow$ CDEMOFP$(\alpha, \beta, \gamma, a, b, n - 1)$

Step2: (Calculation of the systems coefficient matrix)

$CSYS \leftarrow$ CSFCDEM$(\alpha, \beta, \gamma, b, CMR, n-1)$

Step3: (Find RE form of $CSYS$ and solution to GCD of a and b)

Step4: output(solution)
end

Algorithm OPBGCD can be modified by the application of a different technique in reducing the array representation of the matrix $CSYS$ obtained from step 2, to its row echelon form. This leads to the construction of 3 different nonmodular algorithms, namely the OPBGCD-SSGSE, OPBGCD-DSGSE and the classical OPBGCD-RNGSE algorithms. The algorithms derived its name from the three different matrix reduction techniques described in 4.2. In the following discussion, we describe the subalgorithm SSGSE-SOLN, which is the subalgorithm called in step 3 of OPBGCD-SSGSE algorithm. The other two algorithms, namely the OPBGCD-DSGSE and OPBGCD-RNGSE calls the DSGSE-SOLN algorithm and RNGSE-SOLN algorithm respectively, which are extensions of the DSGSE and RNGSE algorithms respectively. The reduction algorithms are extended in order to compute the coefficients of the GCD from the RE form of the reduced matrix.

Algorithm SSGSE-SOLN

/*The algorithm reduces the matrix A to upper echelon form via (4.1) and computes gcd(a, b)*/
Input:
A : array representation of an integral matrix of size $m \times n$.
Output:
$c = \gcd(a, b)$ such that $c_i \in Q$.

begin
Step1: (*initialization*)
$divisor \leftarrow 1$
Step2: (*reduction step*)
for $k = 0$ to $m - 2$ **do**
/**step 2a: find a nonzero pivot*/

$p \leftarrow k$
while $A_{p,k} = 0$ and $p < m-1$ **do**
$p \leftarrow p+1$
if $p \leq m-1$ and $A_{p,k} \neq 0$ **then**
if $p > k$ **then** interchange rows p and k
/**step 2b: reduce so that the elements under pivot row in pivot column is zero*/*
for $i = k+1$ to $m-1$ **do**
for $j = k+1$ to $n-1$ **do**
$A_{i,j} \leftarrow A_{k,k}A_{i,j} - A_{i,k}A_{k,j}/divisor$
$A_{i,k} \leftarrow 0$
$divisor \leftarrow A_{k,k}$
/*since $p > m-1$, find pivot element in the next column*/
Step3: (*Calculate the solution*)
$degd \leftarrow n - r$
for $i = 1$ to $degd$ **do**
$d(degd - i) \leftarrow \mathrm{RNQ}(A_{r-1,r-1+i}, (A_{r-1,r-1}))$
$d(degd) \leftarrow 1$
$isoln \leftarrow (d(0), ..., d(degd), 1)$
Step4: output($isoln$)
end

4.4 Empirical Computing Time Analysis

In all the empirical tests for the nonmodular and modular techniques, the algorithms are coded in SACLIB version 2.1, Suse Linux 7.0 run on an i386 machine, that is, an intel based Pentium III, 800 Mhz PC. SACLIB [22] is a library of C programs for computer algebra derived from the Fortran version SAC2. The polynomials for the test problems are generated by converting power series polynomials with random integer roots less than 20, which are of variable multiplicity. The empirical computing time for the nonmodular and modular algorithms are observed for input polynomials of degree up to 100, with multiprecision integer coefficients up to 153 decimal digits. Three classes of problems for the Legendre polynomial basis are examined; that is the small GCD and big GCD solutions and the relatively prime polynomials. For the small GCD solutions, the degree of the GCD is slightly less then half the degree of the input polynomials. Observing the class of dense polynomials and systems, the degree of b is taken to be one less than the degree of a. Let $c = \gcd(a, b)$.

4.4.1 Computing Time for OPBGCD-SSGSE and OPBGCD-DSGSE

The timing results (in seconds) for the OPBGCD-SSGSE and OPBGCD-DSGSE for the three classes of problems are tabulated below:

Table 4.1: Timing Results (in secs) for the Legendre Polynomial Basis: Small GCD Case

deg(a)	deg(c)	OPBGCD-DSGSE	OPBGCD-SSGSE
20	3	0.23	0.29
40	15	10.44	14.22 (+N2000000)
60	25	133.96 (+N2000000)	197.68 (+N2000000)
80	38	599.48 (+N4000000)	975.20 (+N4000000)
100	38	3859.00 (+N8000000)	-

Table 4.2: Timing Results (in secs) for the Legendre Polynomial Basis: Big GCD Case

deg(a)	deg(c)	OPBGCD-DSGSE	OPBGCD-SSGSE
20	16	0.06	0.06
40	33	1.11	1.11
60	54	3.13	3.89
80	72	13.93	19.81
100	85	134.00 (+N4000000)	170.61 (+N4000000)

Remark: In the above results, +N2000000 allocates an additional two million words of memory to list storage. If no memory allocation is written, +N1000000 words of memory to list storage have been used, which is the allocation by default.

From the empirical results in Table 4.1, the ratio of the average computing time for OPBGCD-SSGSE to OPBGCD-DSGSE is approximately equals to 1.6. The ratio of the average computing time of OPBGCD-SSGSE to OPBGCD-DSGSE for the big GCD case is reduced to about 1.3. The performance of OPBGCD-DSGSE for the three different class of problems is shown in Table 4.4.

Table 4.3: Timing Results (in secs) for the Legendre Polynomial Basis: Relatively Prime Case

deg(a)	OPBGCD-DSGSE	OPBGCD-SSGSE
20	0.34	0.43
40	46.4 (+N2000000)	67.80 (+N2000000)
60	197.5 (+N2000000)	300.78 (+N2000000)
80	1884.00 (+N4000000)	-

Table 4.4: OPBGCD-DSGSE Timing Results (in secs) for the Legendre Polynomial Basis

deg(a)	small GCD	big GCD	relatively prime
20	0.23	0.06	1.00
40	10.44	1.11	46.4
60	133.96	3.13	197.5
80	599.48	13.93	1884.00
100	3859.00	134.00	-

4.5 Discussions

The empirical results emphasize the superiority of OPBGCD-DSGSE over OPBGCD-SSGSE. The results correspond to the superiority of the double step Gauss elimination algorithm which is about twice the efficiency of the single step Gauss elimination algorithm. As a comparison in the performance of OPBGCD-DSGSE algorithm between the three different class of problems, the results show that for a fixed degree of the polynomial inputs a, the algorithm takes the shortest time in the big GCD case, when the rank of the corresponding systems coefficient matrix is small in size. In other words, for same degree inputs, the execution time for the relatively prime case (rank=$n = \deg(a)$) is greater than the small GCD case (big rank but smaller than n) which is in fact also greater than the big GCD case (small rank).

CHAPTER V

A MODULAR GENERALIZED POLYNOMIAL GCD COMPUTATION

5.1 An Introduction to the Modular Approach

The following discussion is based on the theories that have been established in Barnett [7] and [11]. Consider the situation when the polynomials $a(x)$ and $b(x)$ are in the generalized form (1.4), such that the degree of a is greater than or equal to degree of b. We refer to the general algorithm described in 3.4 involving the comrade matrix approach which comprises of:

1. constructing the comrade matrix (3.3) associated with $a(x)$.
2. applying the recursive equation (3.11) to construct the systems of linear equations described in 3.3.1.
3. reducing the coefficient matrix in step 2 to upper echelon form and finding the desired solution.

From the procedure described above, it is observed that the incorporation of the modular technique is possible at 3 different stages, that is either from the stage of the construction of the comrade matrix at step 1 onwards, or from the application of the recursive equation at step 2 onwards, or solely from step 3 which reduces the matrix obtained from step 2 (which may include multiprecision integer entries) to its row echelon form followed by the computation of the coefficients of the GCD. The application of the modular approach computes the solution to the GCD in several finite fields which eventually converges to the true solution when the bound for the solution has been exceeded. The method involves the application of 2-congruence Chinese remainder algorithm (CRA) followed by the rational number reconstruction algorithm (RNRA), since the true solution is in the form of a monic generalized

polynomial with rational number coefficients. In this Chapter, we present a modular technique that applies modular arithmetic only at step 3, that is, in the Gauss elimination procedure on a matrix $A \pmod{p_i}$ followed by the computation of the solution gcd $\pmod{p_i}$, for $i = 1, ..., n$ for some n.

The algorithm MOPBGCD in [2] suggests a termination test that can eliminate the application of unlucky primes. The algorithm [2] has not been tested for polynomials with multiprecision coefficients. A complete form of the application of the modular approach is presented in the following algorithm.

5.2 Algorithm MOPBGCD

Input:
The defining terms α_i, β_i and γ_i for $0 \leq i \leq n-1$.
The generalized polynomial, $a = (a_0, a_1, ..., a_{n-1}, 1)$.
The generalized polynomial $b = (b_0, b_1, ..., b_m, ..., b_{n-1})$, $m \leq n$.
Output:
The monic generalized GCD c such that $c = \gcd(a, b)$ is of degree k.

begin
Step1: (Calculation of the comrade matrix)
$CMR \leftarrow$ CDEMOFP$(\alpha, \beta, \gamma, a, b, n-1)$

Step2: (Calculation of the systems coefficient matrix)
$CSYS \leftarrow$ CSFCDEM$(\alpha, \beta, \gamma, b, CMR, n-1)$

Step3: (Initialization)
set $k = 1$ and set $n = 1$

Step4: (Chinese remainder algorithm)
find a prime p_0 and the matrix $CSYS \pmod{p_0}$
find the reduced matrix and the solution C_0 in Z_{p_0}
if $\deg(C_0) = 0$ **then** output(1)
set $p_{new} = p_0$ and set $C_{new} = C_0$

Step5:
Step 5a:
set $k = k + 1$
find a prime p_0 and the matrix $CSYS \pmod{p_0}$
find the reduced matrix and the solution C_0 in Z_{p_0}
if $\deg(C_0) > \deg(C_{new})$ **then**(p_0 is unlucky. Find another prime)
set $k = k - 1$; goto Step5
else if $\deg(C_0) < \deg(C_{new})$ **then**(all previous primes unlucky)
set $k = 1$; $n = 1$; goto step4
Step 5b:
Solve for the 2-congruence CRA :
$C_1 \cong C_0 \pmod{p_0}$ and $C_1 \cong C_{new} \pmod{p_{new}}$
set $C_{new} = C_1$ and set $p_{new} = p_0 * p_{new}$
if $k \leq 2^n$ **then** goto step5

Step6: (rational number reconstruction)

Step7: (if rational reconstruction fails)
set $n = n + 1$; goto step5

Step8: (divisor test and preparing output)
if test true **then** output(solution)
else
if there is still enough primes **then**
set $n = n + 1$ and goto Step 5
else (the primes are exhausted)
output(all primes were unlucky)
end

In short, CSFCDEM applies a recursive equation involving the defining terms and the comrade matrix resulting as the matrix coefficient of a certain systems of equations. This rational number matrix coefficient obtained from the recursive equations is transformed into an equivalent integral matrix $CSYS$, which is returned by CSFCDEM. In this approach, steps 1 and 2 applies rational number and multiprecision integer arithmetic operations.

5.2.1 The Selection of Primes

The MOPBGCD requires a list of distinct odd single precision primes. The list can be generated by the application of SACLIB's function DPGEN(m, k) so that the prime p_i in the list is such that $m \leq p_i < m + 2k$. SACLIB2.1 also provides such a list called LPRIME. The list LPRIME is used in the MOPBGCD algorithm and is such that LPRIME $= \{536830337, ..., 536870909\}$. There are 2055 elements in this list where each prime p_i is nearly as large as the maximum single precision integer. MOPBGCD calls for the subroutine FDPRIME which calls for LPRIME(i), the ith element of LPRIME. For each row $1 \leq i \leq n$ of the matrix A obtained from the subroutine CSFCDEM, the element a_{ij} of each column $1 \leq j \leq n$ is converted to $a_{ij} \pmod{p_i} \in \mathrm{GF}(p_i)$ by applying SACLIB's function MDHOM. If there exists a single row such that all the column elements $\pmod{p_i}$ are equal to 0, the process of taking the homomorphic images is terminated and a new prime is selected. The whole procedure is then repeated until there is no single row whose column elements are all

divisible by the selected prime. $A \pmod{p_i}$ is thus obtained and applied to the subroutine MGSEGCD. By this technique we are able to simultaneously check that the homomorphic images of the elements of A does not alter the rank of the matrix A.

5.2.2 Finding the Row Echelon Form and Solution in GF(p)

The algorithm reduces the matrix $A \pmod{p}$ to upper echelon form in GF(p) and computes the coefficients of the GCD in GF(p).

Algorithm MGSEGCD

Input:
A : array representation of an $m \times n$ matrix in GF(p).
Output:
$isoln = (d(0), ..., d(\deg(d)), 1)$, the coefficients of $\gcd(a, b) \pmod{p}$.

begin
Step 1: (*reduction step*)
  for $k = 0$ to $m - 2$ **do**
    /**step 1a: find a nonzero pivot**/
    $p \leftarrow k$
    while $A_{p,k} = 0$ and $p < m - 1$ **do**
      $p \leftarrow p + 1$
    if $p \leq m - 1$ and $A_{p,k} \neq 0$ **then**
      if $p > k$ **then** interchange rows p and k
      /**step 1b: reduce so that the elements under pivot row in pivot column is zero**/
      for $i = k + 1$ to $m - 1$ **do**
        for $j = k + 1$ to $n - 1$ **do**
          $A_{i,j} \leftarrow (A_{k,k} * A_{i,j} - A_{i,k} * A_{k,j}) \pmod{p}$
        $A_{i,k} \leftarrow 0$
    /*since $p > m - 1$, find pivot element in the next column**/
Step2: calculate the rank r of A
Step3: (*calculate the solution* $\pmod{p}$)
  $degd \leftarrow n - r$
  for $i = 1$ to $degd$ **do**
    $d(degd - i) \leftarrow A_{r-1,r-1+i} * \mathrm{INV}(A_{r-1,r-1}) \pmod{p}$
  $d(degd) \leftarrow 1$

$isoln \leftarrow (d(0), ..., d(degd), 1)$
Step4: output($isoln$) **end**

The computation of $A_{i,j} \pmod p \leftarrow (A_{k,k} * A_{i,j} - A_{i,k} * A_{k,j}) \pmod p$ at step 1b applies SACLIB's function MDSUM, MDNEG and MDPROD which computes the sum, negative and product of elements in GF(p) for single precision prime p. Let A be the RE form of the matrix obtained from step 1. The rank of A can be calculated by checking if a_{ij} is 0, for each i from m to 1 and for each column $1 \leq j \leq n$. If a_{rs} for some s, is the first nonzero element encountered, then rank of A is r. Obviously, since all the computations in GF(p) is of order 1, the computing time of step 1 is dominated by m^2n. The computation of the rank is dominated by mn and the computation of the solution is dominated by $degd \preceq n$. If $m = n$ then $t_{\text{MGSEGCD}} \preceq n^3$.

5.2.3 The Application of CRA to MOPBGCD Algorithm

Step 5 of algorithm MOPBGCD dominates the cost of applying the Chinese remainder algorithm. Let γ be the number of good primes used. Assuming that all the primes are of good reduction, we compute the cost of the $(k-1)$st iteration of the CRA as follows:

1. if $|CSYS|_\infty$ then the cost of computing $CSYS \pmod{p_k}$ is dominated by $n^2\text{L}(C)$.
2. The cost of finding the solution $c_k \pmod{p_k}$ where

   $$c_k \pmod{p_k} = (d_{k,0} \pmod{p_k}, d_{k,1} \pmod{p_k}, ..., d_{k,\delta} \pmod{p_k})$$

   is the cost of applying algorithm MGSEGCD which is dominated by $n^3 + \delta$.
3. Let C_k be the solution to the $(k-1)$st iteration of the CRA. The cost of computations in step 5b is the cost of solving the 2-congruence CRA given by

   $$\begin{aligned} c &\cong c_k \pmod{p_k}, \\ c &\cong C_{k-1} \pmod{P_{k-1}}, \end{aligned}$$

such that C_{k-1} is the solution modulo $P_{k-1} = p_1 p_2 ... p_{k-1}$ obtained from the $(k-2)$st iteration. C_k is obtained by solving

$$\begin{aligned} x_{k-1} &= \text{INV}(P_{k-1} \pmod{p_k}, p_k)(c_k - C_{k-1} \pmod{p_k}) \pmod{p_k}, \\ C_k &= C_{k-1} + x_{k-1} P_{k-1}. \end{aligned} \tag{5.1}$$

The cost of computing x_{k-1} is dominated by $\text{L}(P_{k-1})$. Now,

$$\begin{aligned} C_{k-1} &= x_0 + p_1 x_1 + ... + p_1 p_2 ... p_{k-2} x_{k-2} \\ &\leq p_1 - 1 + p_1(p_2 - 1) + ... + p_1 p_2 ... (p_{k-1} - 1) \\ &= P_{k-1}, \end{aligned}$$

which implies that the computing time for computing C_k for the $(k-1)$st iteration of the CRA is dominated by $\text{L}(P_{k-1})$. Since each x_{k-1} and C_k of (5.1) has δ terms, the cost of computing these terms is dominated by $\delta \text{L}(P_{k-1})$

Therefore the cost of computing step 5, if γ primes are used is such that $t_5 \preceq \gamma(n^2 \text{L}(C) + n^3 + \delta \text{L}(p_1 p_2 ... p_\gamma))$.

Let a and b be in the generalized form such that $\gcd(a, b) = d$ and $|\text{num}(d_i)| < D$, $0 < \text{den}(d_i) < D$. Let n_1 be number of primes of good reduction used. Then, $D < \sqrt{\frac{M}{2}}$ such that $M = p_1 ... p_{n_1}$. From $M > 2D^2$ and taking the least possible prime, i.e $p_i = 2$, we have $2^{n_1} > 2D^2 \Leftrightarrow n_1 > 1 + 2log_2(D)$. So, if $p > 2$ then $n_1 \leq 2 \log_2(D) + 2 \sim \log_2(D)$. The total cost of applying the CRA is dominated by $\log_2(D)(n^2 \text{L}(C) + n^3 + \delta \text{L}(M))$ such that $M = p_1 p_2 ... p_{\log_2(D)}$, that is when taking $\gamma = \log_2(D)$.

5.2.4 The Rational Number Reconstruction Algorithm

Given $u \in \text{GF}(M)$ such that M is the product of several primes, the algorithm finds the integers a and b that satisfies $a \cong ub \pmod{M}$ and outputs a solution if it exists or outputs NIL if the solution does not exist. MOPBGCD applies SACLIB's function RNFMR to reconstruct rational numbers from the coefficients of the GCD which is an output of MILLCRA obtained in $\text{GF}(M)$. If the number of primes

composing M is not sufficient, we abort the attempt to find the rest of the coefficients of the GCD. It can be shown that shown that the computing time for the classical rational number reconstruction algorithm, given that $u \in \mathrm{GF}(M)$ is dominated by $\mathrm{L}(u)\mathrm{L}(M) + \mathrm{L}^2(M)$. If the degree of GCD d is δ then the total cost for each execution is dominated by $\delta\mathrm{L}^2(M)$.

If k is the number of times that the rational number reconstruction algorithm is attempted, that is for every $2^k + 1$, we therefore have $k \preceq \log_2(\log_2(D))$. The total cost of applying the RNRA to obtain a correct solution if all the n_1 primes are lucky is dominated by $\delta \log_2(\log_2(D))\mathrm{L}^2(M)$. In this case, the computing time for the application of CRA and RNRA algorithms is dominated by

$$\begin{aligned} & \log_2(D)(n^2\mathrm{L}(C) + n^3 + \delta\mathrm{L}(M)) + \delta \log_2(\log_2(D))\mathrm{L}^2(M) \\ \preceq \quad & \mathrm{L}(D)(n^2\mathrm{L}(C) + n\mathrm{L}^2(M) + n^3) \\ \preceq \quad & n^3\mathrm{L}(D)(\mathrm{L}(C) + \mathrm{L}^2(M)) \end{aligned}$$

such that $M = p_1 p_2 ... p_{\mathrm{L}(D)}$. Note that $\log_2(D) \leq \log_2(D) + 1 \sim \mathrm{L}(D)$.

5.2.5 The Termination Criterion

Once the function RNFMR returns a polynomial with all of its coefficients having a solution, MOPBGCD applies a divisor test to check if the candidate solution divides each of the input polynomial. Let $\gcd(a, b) = d$ such that $\deg(d) = \delta$. The OPDIV algorithm described in the preceding chapter generates the generalized polynomial remainder sequence $(a = r_0, r_1, ..., r_{k+1})$, in which r_{i+1} is the remainder when r_i is divided by $x^{\deg(r_i)-\delta} * d$ and $r_0 = a$. Assuming classical matrix multiplication, the cost of obtaining the $\deg(r_i) + 2$ components of $x^{\deg(r_i)-\delta} * d$, comprising the coefficients of the remainder is dominated by $\deg(r_i)^3 \preceq n^3$. Summing up the cost for $1 \leq i \leq n - \delta$ gives a total cost that is dominated by n^3. We therefore claim in **??** that OPDIV is quartic in n, where n is the degree bound of the input polynomials. However, further work will be required for finding the computing time bound in terms of the parameters of its input variables.

5.3 Computing Time Results for Algorithm MOPBGCD

In this section we examine the theoretical and empirical computing time analysis of the modular algorithm MOPBGCD.

5.3.1 Theoretical Computing Time Analysis

We tabulate the computing time results that have been obtained for the respective subalgorithms. For the analytical results in Table 5.1, the input

Table 5.1: Computing Time Analysis for the Legendre Basis Case

Algorithm	Dominating Function
CDEMOFP	$n^2 + n\mathrm{L}(u) + \mathrm{L}(u)\mathrm{L}(\mathrm{L}(u)) = t_1'$
CSFCDEM	$n^3\mathrm{L}^2(u)\{\mathrm{L}(\max(\mathrm{L}(r), n\mathrm{L}(u))) + \mathrm{L}(\mathrm{L}(u) + \mathrm{L}(v))\} = t_2'$
CRA & RNRA	$n^3\mathrm{L}(D)(\mathrm{L}(C) + \mathrm{L}^2(M)) = t_3'$

polynomials represented relative to the generalized basis are of the form $a, b \in \mathrm{P}(u, v, n, (p_i)_{i=0}^n)$ where (p_i) is the set of Legendre basis. Suppose $c = \gcd(a, b)$ such that $\deg(c) = \delta$ and $|\mathrm{num}(c_i)| < D$, $0 < \mathrm{den}(c_i) < D$. We let $M = p_1 p_2 ... p_{n_1}$ be primes of good reduction, so that $n_1 \preceq \log_2(D)$. Comparing $t_1' + t_2'$ and t_3', we have

$$\frac{t_1' + t_2'}{t_3'} = \frac{\mathrm{L}^2(u)(\mathrm{L}(q) + \mathrm{L}(r))}{\mathrm{L}(D)(\mathrm{L}(C) + \mathrm{L}^2(M))}. \tag{5.2}$$

where $q = \max(\mathrm{L}(r), n\mathrm{L}(u))$ and $r = \mathrm{L}(u) + \mathrm{L}(v)$. The result (5.2) depicts the efficiency of the modular technique applied in the Gauss elimination procedure and calculating the solution to the problem, in comparison with the rational number computations applied in obtaining the comrade and systems coefficient matrix.

5.3.2 Empirical Computing Time Analysis

The timing results (in seconds) of algorithm MOPBGCD for the three classes of problems are given in the following tables.

Table 5.2: MOPBGCD for the Legendre Polynomial Basis: Small GCD Case

deg(a)	deg(c)	maxdigit$\{a_i, b_i\}$	execution time (secs) MOPBGCD	no. of primes	memory
20	3	30	0.07	3	default
40	15	59	0.68	5	default
60	25	88	3.09	9	default
80	38	113	9.07	17	default
100	38	153	17.7	17	+N2000000

Table 5.3: MOPBGCD for the Legendre Polynomial Basis: Big GCD Case

deg(a)	deg(c)	execution time (secs) MOPBGCD	no. of primes	memory
20	16	0.06	5	default
40	33	0.86	17	default
60	54	3.68	33	default
80	72	8.74	33	default
100	85	19.77	33	+N2000000

As in 4.4.1, +N2000000 allocates an additional two million words of memory to list storage. The default allocation is 1000000 words. As expected, the big GCD case requires more primes and execution time compared to the small GCD case. For a fixed degree of the polynomials tested, although the number of primes required for the big GCD case is about twice that of the small GCD case, the differences in computing time or space required in the two cases is not so significant. For the relatively prime case, the program terminates upon encountering a solution c such that $\deg(c) = 0$, since the degree of the actual solution cannot be any smaller. Unless a bad prime is encountered before this actual solution is obtained, in which case, the rank of the systems coefficient matrix is less than n, the number of primes used will be equal to 2. Therefore, the relatively prime case may be expected to consume the shortest time, assuming that the first encounter is a good prime. The theoretical computing time

Table 5.4: MOPBGCD for the Legendre Polynomial Basis: Relatively Prime Case

deg(a)	execution time (secs) MOPBGCD
20	0.07
40	1.06
60	1.83
80	6.49

Table 5.5: MOPBGCD Timing Results in secs for the Legendre Polynomial Basis

deg(a)	small GCD	big GCD	relatively prime
20	0.07	0.06	0.07
40	0.68	0.86	1.06
60	3.09	3.68	1.83
80	9.07	8.74	6.49
100	17.77	19.74	-

results of the preceding section, compares the computing time required by the rational number algorithms (in steps 1 and 2 of 3.4), to the modular algorithm which applies the CRA and RNRA algorithms prior to the application of the divisor test OPDIV algorithm. The empirical results of the test problems were obtained so that empirical comparisons can be made with respect to this application.

5.4 Discussions

Table 5.2, 5.3 and 5.4 respectively, presents the total execution time of MOPBGCD with respect to 3 classes of problems described in 4.4. For a fixed degree input of the test problems, the sum of the execution time of the subalgorithms of MOPBGCD in columns 3 and 4 of Table 5.6 and 5.7, respectively, is subtracted from the total execution time for the problem to obtain the respective execution time for OPDIV. The efficiency of OPDIV with respect to MOPBGCD is clearly indicated from the timing results of the last columns of Tables 5.6 and 5.7. Moreover, Table 5.6

Table 5.6: Execution time in secs : MOPBGCD sub-algorithms for Small GCD Legendre Basis Inputs

deg(a)	deg(c)	Rational Arithmetic	Modular arithmetic	OPDIV
20	3	0.04	0.01	0.01
40	15	0.4	0.13	0.15
60	25	1.62	0.84	0.63
80	38	3.87	3.81	1.39
100	38	7.52	7.40	2.78

Table 5.7: Execution time in secs : MOPBGCD sub-algorithms for Big GCD Legendre Basis Inputs

deg(a)	deg(c)	Rational Arithmetic	Modular arithmetic	OPDIV
20	16	0.04	0.01	0.01
40	33	0.42	0.31	0.13
60	54	1.52	2.00	0.16
80	72	3.74	4.68	0.32
100	85	8.09	10.70	0.98

shows that the execution time for the application of rational number arithmetic in steps 1 and 2 of 3.4 exceeds the execution time for the application of modular arithmetic of the succeeding steps, only when the size of the GCD is sufficiently small. When the size of the GCD is sufficiently big, the modular approach of reducing the systems coefficient matrix to its RE form and computing the rational number solution is as time consuming as the application of the rational number arithmetic of the preceding steps. A hybrid of these two approaches can be of advantage to the algorithm, whereby the rational number functions were already made available.

CHAPTER VI

SUMMARY, CONCLUSIONS AND FURTHER RESEARCH

6.1 Introduction

We now reflect on what have been the results of this research, summarizing our findings and suggesting areas for further investigations. The main goal of this study is to produce a useful tool for computing the GCD of polynomials in the generalized form such that the general basis satisfies the three term recurrence relation. In the empirical comparisons, we study the performance of the algorithms on polynomials represented relative to an orthogonal basis, which are a special class of the polynomials in the general basis.

6.2 Significant Findings and Conclusions

The initial task of the work is to construct and implement the nonmodular algorithms from a general OPBGCD algorithm. This is achieved by the construction of the algorithms for constructing the comrade matrix of a polynomial (Algorithm CDEMOFP) and the coefficient systems for the comrade matrix (Algorithm CSFCDEM). This algorithms apply SACLIB's rational number or multiprecision integer arithmetics. Auxiliary algorithms for matrix computations have also been constructed and included in the appendix. The auxiliary algorithms are called by the comrade matrix and coefficient systems algorithms.

The modular techniques have been known for its competitiveness in dense class of problems related to the GCD and matrix related problems such as finding the determinants or solutions to linear equations. Therefore, it is our objective to emphasize the superiority of the implementation of the modular techniques in our algorithms, focusing only at the class of dense polynomials with multiprecision

coefficients in our empirical inputs.

According to [43], the small GCD class of power series polynomials is known to be the worst case for the Euclidean algorithm but is usually the best case for the modular algorithms. In the case of computing the GCD of polynomials in the generalized form, the modular techniques that have been developed are especially appreciated for being markedly superior then the nonmodular techniques for the class of dense rational number polynomials with small GCD solutions. The empirical computing time results show that the modular algorithms are markedly superior to the nonmodular algorithms when the polynomials are sufficiently dense with degree at most 100, the numerators of the coefficients at most 153 digits and the magnitude or degree of the GCD is about less than half the degree of the input polynomials (small GCD case). In this case, the modular algorithms are on the average about 60 times faster than the nonmodular algorithms.

The empirical time results also suggest that a hybrid application of the modular approach and multiprecision arithmetic can be regarded as efficient for both the small and big GCD solutions such that the bound of the true GCD solution are not predetermined, provided also that a correct solution can be acquired from the technique.

The efficiency of the modular approach requires efficient integration and implementation of the devisor test and rational number reconstruction algorithms. The divisor test acts as a termination criterion that can also determines the effectiveness of the method. A bad prime p is encountered if p divides the leading coefficient of the input polynomials or p divides the entries of the last row of the RE form of the equivalent integral systems coefficient matrix. Thus, the divisor test is an essential device that determines the correctness of the final solution and eliminates the use of a bad reduction.

Given two polynomials in its generalized form, the polynomials can be converted to its power series form for the GCD computations and converted back to its generalized form. The transformation from a polynomial in the generalized form to a polynomial in the power series form requires multiplication of an $(n+1) \times (n+1)$ matrix with and $n+1$ column vector and is therefore of magnitude $O(n^2)$. The

transformation of a polynomial in the power series representation to a polynomial in the general basis representation requires finding the unique solution to an $n \times n$ systems of equations, which is roughly $O(n^3)$. The application of the Gauss elimination algorithm is basically $O(n^3)$ with the coefficient of n^3 determined by the length of multiprecision integers involved in the computation. The contribution of the research into constructing and implementing an effective and efficient OPBGCD algorithms is not a trivial one. With the results of this research, the applications of the exact Gauss elimination methods (modular or nonmodular) have also been extended to the exact computations of the GCD of generalized polynomials.

6.3 Future Research

The empirical computing time results in the preceding chapter also illustrate the efficiency of the modular technique in computing the GCD of 2 polynomials relative to the Legendre basis. The computing time takes only seconds when the degree bound for the input polynomial is 100 and the coefficient bound 153 decimal digits. With the value of the defining terms dependent on the degree bound, so that these values remain single precision, we are able to construct an efficient algorithm for calculating the comrade and systems coefficient matrix using classical rational number arithmetic and devote the application of the modular technique to the Gauss elimination procedure, which is then followed by the computations of the integer solution to the problem in simpler domains and finding the rational number solution in the original domain. We will work on the extension of the application of the modular technique to further reduce the number of multiprecision computations and the rational number arithmetic involved in steps 1 and 2 of the general algorithm in 3.4. However, further properties of the homomorphic image scheme have to be carefully investigated to enable the application of the modular technique in the construction of the comrade and the systems coefficient matrices.

BIBLIOGRAPHY

[1] Ali A. Rahman(1992). "The Use of GCD Computation to Remove Repeated Zeroes from a Floating Point Polynomial." SCAN92 Conference. Oldenberg, Germany.

[2] Ali Abdul Rahman and Nor'aini Aris(2001). "Alkhwarizmi Modulo Pembahagi Sepunya Terbesar Dua Polinomial Relatif Terhadap Asas Berortogon." Simposium Kebangsaan Sains Matematik ke-9, Julai 2001 UKM, Bangi, Malaysia.

[3] Ali Abdul Rahman and Nor'aini Aris(2002). "The State of the Art in Exact Polynomial GCD Computations." MSTC 2002. Symposium A: Physical Sciences Engineering and Technology. 19-21 September 2002. Johor Bahru, Malaysia.

[4] Bareiss, E.H.(1968). "Sylvester's identity and Multistep Integer Preserving Gaussian Elimination." *Math.Comp.* 22(103). 565-578.

[5] Bareiss, E.H.(1972). "Computational Solutions of Matrix Problems over an Integral Domain." *J. Inst. Math. Applic.* 10. 68-104.

[6] Barnett, S.(1975). "A Companion Matrix Analogue for Orthogonal Polynomials." *Linear Algebra and its Applications.* 12. 197-208.

[7] Barnett, S. and Maroulas, J.(1978). "Greatest Common Divisor of Generalized Polynomial and Polynomial Matrices." *Linear Algebra and its Applications.* 22. 195-210.

[8] Barnett, S. and Maroulas, J.(1979). "Polynomials with Respect to a General Basis. I. Theory." *J. of Math. Analysis and Applications.* 72. 177-194.

[9] Barnett, S. and Maroulas, J.(1979). "Polynomials with Respect to a General Basis. II. Applications." *J. of Math. Analysis and Applications.* 72. 599-614.

[10] Barnett, S. and Maroulas, J.(1979). "Further Results on the Qualitative Theory of Generalized Polynomial." *J. Inst. Maths Applics.* 23. 33-42.

[11] Barnett, S.(1983). "Polynomials and Linear Control Systems." New York and Basel:Marcel Dekker.

[12] Borosh, I., Fraenkel, A.S.(1966). "On Modular Approaches to Solving Systems of Linear Equations with Rational Coefficients." *Math.Comp.* 20. 107-112.

[13] Brown, W.S.(1971). "On Euclid's Algorithm and Polynomial Greatest Common Divisors." *J. ACM.* 18(1). 478 - 504.

[14] Brown, W.S. and Traub, J.F.(1971). "On Euclid's Algorithm and the Theory of Subresultants" *J. ACM.* 18, 1971.

[15] Cabay, S.(1971). "Exact Solution of linear Equations." Proc. Second Symp. on Symbolic and Algebraic Manipulation, ACM, New York. 392 - 398.

[16] Cabay, S. and Lam, T.P.L.(1977). "Congruence Techniques for the Exact Solution of Integer Systems of Linear Equations." *ACM TOMS.* 3(4). 386-397.

[17] Cabay, S. and Lam, T.P.L. (1977). "Algorithm 523, ESOLVE: Congruence Technique for the exact solution of integer systems of linear equations." *ACM Trans. Math. Software* 3(4). (Dec. 1977). 404 - 410.

[18] Collins, G.E.(1967). "Subresultants and Reduced Polynomial Remainder Sequences." *J. ACM.* 14(1).

[19] Collins, G.E.(1971). "The Calculation of Multivariate Polynomial Resultants." *J. ACM.* 18(4) (Oct.1971). 144-152.

[20] Collins, G.E.(1974). "The Computing Time of the Euclidean Algorithm." *SIAM Journal on Computing.* 3(1). 1-10.

[21] Collins, G.E., Mignotte, M and Winkler, F.(1982). "Arithmetic in Basic Algebraic Domains." *Computing Suppl.* 4. 189-220.

[22] Collins, G.E. et al.(1993). "SACLIB1.1 User's Guide." Research Institute for Symbolic Computation, Johannes Kepler University: RICS-Linz Report Series Technical Report Number 93-19.

[23] Collins, G.E. and Encarnacion, M.J.(1995). "Efficient Rational Number Reconstruction." J.Symbolic Computation. 20, 287 - 297.

[24] Char, B.W., Geddes, K.O. and Gonnet, G.H.(1989). "GCDHEU: Heuristic Polynomial GCD Algorithm Based on Integer GCD Computation." *J. Symbolic Comput.* 7. 31-48.

[25] Davenport J.H.(1987). "How Elementary Upper Bounds Generate Cheaper Data." in Caviness,B.(ed): Proc. EUROCAL. 2.

[26] Davenport, J.H., Siret,Y. and Tournier, E.(1988). "Computer Algebra: Systems and Algorithms for Algebraic Computation." London:Academic Press.

[27] Encarnacion, M.J.(1994). "On a Modular Algorithm for Computing GCDs of polynomials over Algebraic Number Fields." in Von Zur Gathen, J. and Giesbrecht, M.(eds): Proceedings of the International Symposium on Symbolic and Algebraic Computation. ACM, New York. 58 - 65.

[28] Geddes, K.O., Czapor, S.R. and Labahn.(1992). "Algorithms for Computer Algebra." , Boston: Kluwer Academic Publishers.

[29] Grant, J.A. and Rahman, A.A.(1992). "Determination of the zeros of a linear combination of generalised polynomials." *J. of Computational and Applied Mathematics.* 42. 269-278.

[30] Howell, J.A. and Gregory, R.T.(1969). "An Algorithm for Solving Linear Algebraic Equations Using Residue Arithmetic." *BIT.* 9. 200-234. 324-337.

[31] Howell, J.A.(1971). "Algorithm 406: Exact Solutions of Linear Equations Using Residue Arithmetic." *Comm. ACM.* 14, 3 (March 1971). 180-184.

[32] Jebelean, T.(1993). "Improving the Multi-Precision Euclidean algorithm." in Proceedings of DISCO '93.

[33] Langemyr, L.(1989). "Computing the GCD of two Polynomials over an Algebraic Number Field." The Royal Institute of Technology, Stockholm, Sweden: Ph.D. Thesis.

[34] Lauer, M.(1982). "Computing by Homomorphic Images." in Buchberger, B., Collins, G.E. and Loos,R.(eds) in cooperation with Albrecht, R: Computer Algebra Symbolic and Algebraic Computation, Computing Suppl. 4. New York: Springer Verlag. 139-168.

[35] Lipson, J.D.(1981). "Elements of Algebra and Algebraic Computing." Menlo Park, California: Benjamin/Cummings. Publishing Company.

[36] Kaltofen, E. and Monagan, M.B.(1999). "On the Genericity of the Modular Polynomial GCD Algorithms." *J. ACM.* 59-65.

[37] Knuth, D.E.(1981). "The Art of Computer Programming. vol II." Reading, Mass: Addison Wesley.

[38] Mazukelli, D.(1972). "Multistep Elimination over Commutative Rings." Northwestern University, Illinois: Ph.D. Thesis.

[39] McClellan, M.T.(1971). "The Exact Solution of Systems of Linear Equations with Polynomial Coefficients." U. of Wisconsin, Madison, Wis: Ph.D. Thesis.

[40] McClellan, M.T.(1973). "The Exact Solution of Systems of Linear Equations with Polynomial Coefficients." *J.ACM.* 20(4). 563-588.

[41] McClellan, M.T.(1977). "A Comparison of Algorithms for the Exact Solution of Linear Equations." *ACM Trans Math Software.* 3(2). 147-158.

[42] Mohamed Omar Rayes(1995). "Parallel Integral Multivariate Polynomial Greatest Common Divisor Computation." Kent State University: Ph.D. Thesis.

[43] Monagan, M.B. and Margot, R.(1998). "Computing Univariate GCDs Over Number Fields." Proc Symposium Discrete Algorithm, Soc. Indust. Appl. Math. 42-49.

[44] Monagan, M. B. and Wittkopf, A.(2000). "On the Design and Implementation of Brown's Algorithm over the Integers and Number Fields." Proc. ISSAC 2000. St. Andrews, Scotland.

[45] Musser, D.R.(1971). "Algorithms for Polynomial Factorization."Computer Sciences Department, University of Wisconsin, Madison: Ph.D. Thesis. Tech. Report. 134.

[46] Newman, M.(1967). "Solving Equations Exactly." *J. Res.* (NBS71B). 4. 171-179.

[47] Nor'aini Aris and Ali Abdul Rahman.(2001). "On the Computation of the GCD of Two Polynomials Relative to an Orthogonal Basis." Department of Mathematics, Faculty of Science, UTM. Technical Report. LT/M Bil. 1/2001.

[48] Rubald, C.M.(1973). "Algorithms for Polynomials over a Real Algebraic Number Field." University of Wisconsin: Ph.D. Thesis.

[49] Smedley, T.J.(1989). "A New Modular Algorithm for Computation of Algebraic Number Polynomial GCDs." *J. ACM.* 91-94.

[50] Von Zur Gathen, J. and Gerhard, J.(1999). "Modern Computer Algebra." Cambridge: University Press.

[51] Wang, P.S.(1980). "The EEZ-Algorithm." SIGSAM bull. 14. 50-60.

[52] Wang, P.S.(1981). "A p-adic Algorithm for Univariate Partial Fractions." Proc. of the 1981 ACM Symposium on Symbolic and Algebraic Computation, New York, USA: ACM Inc. 212-217.

[53] Stromberg, K.R.(1981). "An Introduction to Classical Real Analysis." Belmont, California: Wadsworth Inc.

[54] Theodore, J.R.(1990)."Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory. " 2nd Ed. New York, U.S.A: Johm Wiley and Sons Inc.

[55] Yun, D.Y.Y.(1973). "The Hensel Lemma in Symbolic Manipulation." Dept. of Mathematics, M.I.T., Cambridge, Mass: Ph.D. Thesis.

[56] Zippel, R.E.(1979). "Probabilistic Algorithms for Sparse Polynomials." Proc. of the 1979 European Symposium on Symbolic and Algebraic Computation. Springer LNCS. 72. 216-226.

CHAPTER VII

PAPERS PUBLISHED

Papers presented in Symposiums or Conferences during the duration of study are listed in the following:

1. Nor'aini Aris and Ali Abdul Rahman(2001). "On the Computation of GCD of Polynomials Relative to an Orthogonal Basis." Department of Mathematics, Faculty of Science, UTM. Technical Report. LT/M Bil.1.

2. Ali Abdul Rahman and Nor'aini Aris(2001). "Alkhwarizmi Modulo Pembahagi Sepunya Terbesar Dua Polinomial Relatif terhadap Asas Berortogon." Prosiding Sains Kebangsaan Matematik ke-9. 18-20 Julai 2001. UKM, Bangi.

3. Ali Abdul Rahman and Nor'aini Aris(2002). "The State of the Art in Exact Polynomial GCD Computations." Proceedings Malaysian Science and Technology Conference 2002. pages 923-933. 19-21 September 2002. Johor Bahru, Malaysia.

4. Nor'aini Aris and Ali Abdul Rahman(2001). "On the Division of Generalized Polynomials." Asean Symposium on Computer Mathematics 2003. 17-19 April 2003. (Postponed indefinitely due to SARS outbreak) Proceedings ASCM03 will be published as a volume in Lecture Note Series on Computing, World Scientific Publisher, Singapore/River Edge, USA.