ขั้นตอนวิธีทำซ้ำแบบนิวตันสำหรับการคำนวณพหุนามมอดุลาร์ผกผันภายใต้มอดุโล $x^n \pm 1$ สำหรับบางรูปแบบของ n

นายศมากร ศรีพัฒนกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์และวิทยาการคณนา
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

NEWTON ITERATIVE ALGORITHM FOR POLYNOMIAL MODULAR INVERSION MODULO $x^n \pm 1$ FOR SOME PATTERNS OF n

Mr. Samakorn Sripatthanakul

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Applied Mathematics and

Computational Science

Department of Mathematics and Computer Science

Faculty of Science

Chulalongkorn University

Academic Year 2021

Thesis Title	NEWTON ITERATIVE ALGORITHM FOR POLYNOMIAL MODULAR INVERSION MODULO $x^n \pm 1$ FOR SOME PATTERNS OF n
By	Mr. Samakorn Sripatthanakul
Field of Study	Applied Mathematics and Computational Science
Thesis Advisor	Wutichai Chongchitmate, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.... Dean of the Faculty of Science

(Professor Polkit Sangvanich, Ph.D.)

THESIS COMMITTEE

.... Chairman

(Associate Professor Krung Sinapiromsaran, Ph.D.)

.... Thesis Advisor

(Wutichai Chongchitmate, Ph.D.)

.... Examiner

(Thap Panitanarak, Ph.D.)

.... External Examiner

(Assistant Professor Wittawat Kositwattanarerk, Ph.D.)

ศมากร ศรีพัฒนกุล : ขั้นตอนวิธีทำซ้ำแบบนิวตันสำหรับการคำนวณพหุนามมอดุลาร์ผกผันภายใต้มอดุโล $x^n \pm 1$ สำหรับบางรูปแบบของ n. (NEWTON ITERATIVE ALGORITHM FOR POLYNOMIAL MODULAR INVERSION MODULO $x^n \pm 1$ FOR SOME PATTERNS OF n) อ.ที่ปรึกษาวิทยานิพนธ์หลัก : อ.ดร.วุฒิชัย จงจิตเมตต์, 33 หน้า.

วิทยานิพนธ์ฉบับนี้นำเสนอขั้นตอนวิธีสำหรับการคำนวณมอดุลาร์ผกผันของพหุนามในริงของพหุนามเหนือฟิลด์จำกัด $\mathbb{F}_q$ ที่มีลักษณะเฉพาะ p เมื่อกำหนดพหุนาม f และจำนวนนับ r โดยใช้แนวคิดของขั้นตอนวิธีทำซ้ำแบบนิวตัน จะได้ว่าเราสามารถหาขั้นตอนวิธีการหารแบบเร็วที่ใช้หาตัวผกผันของ f ภายใต้มอดุโล $x^{p^r} - 1$, $x^{p^r} + 1$, $x^{2p^r} - 1$ และ $x^n - 1$ เมื่อ $n = 2^r d$ และ $r, d \in \mathbb{N}$ ได้ โดยเราได้มีการวิเคราะห์ความซับซ้อนในการคำนวณของขั้นตอนวิธีภายใต้มอดุโลเหล่านี้ไว้ที่ $\mathcal{O}(n \log n)$ ซึ่งมีประสิทธิภาพมากกว่าขั้นตอนวิธีแบบ Half-GCD ในแง่ของการคำนวณสำหรับ n ขนาดใหญ่

ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	ลายมือชื่อนิสิต	ศมากร ศรีพัฒนกุล
		ลายมือชื่อ อ.ที่ปรึกษาหลัก	วุฒิชัย จงจิตเมตต์
สาขาวิชา	คณิตศาสตร์ประยุกต์และวิทยาการคณนา		
ปีการศึกษา	2564		

6270102823 : MAJOR APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCE
KEYWORDS : DIVISION ALGORITHM / FINITE FIELD / ITERATIVE ALGORITHM / MULTIPLICATION TIME

SAMAKORN SRIPATTHANAKUL : NEWTON ITERATIVE ALGORITHM FOR POLYNOMIAL MODULAR INVERSION MODULO $x^n \pm 1$ FOR SOME PATTERNS OF n. ADVISOR : WUTICHAI CHONGCHITMATE, Ph.D., 33 pp.

This thesis presents an algorithm for computing the modular inverse of a polynomial in a ring of polynomials over a finite field $\mathbb{F}_q$ with a characteristic p. Given a polynomial f and a natural number r, by applying the idea of the Newton iteration algorithm, the fast division algorithm used to find the inverse of f under modulo $x^{p^r}-1$, $x^{p^r}+1$, $x^{2p^r}-1$ and x^n-1 where $n = 2^r d$ for some $r, d \in \mathbb{N}$, is established. The cost analysis for these cases show that the algorithm has the computational complexity of $\mathcal{O}(n \log n)$ which is more efficient than the Half-GCD algorithm in terms of computational complexity for large n.

Department	: Mathematics and Computer Science	Student's Signature	Samakorn Sripatthanakul
Field of Study	: Applied Mathematics and Computational Science	Advisor's Signature	วุฒิชัย จงจิตเมตต์
Academic Year	: 2021		

ACKNOWLEDGEMENTS

It is difficult to express my gratitude to my advisor, Wutichai Chongchitmate, Ph.D. for his enthusiasm, to inspire and efforts in explaining and clarify important things related to this research. Throughout research writing period, he has provided advice, taught basic knowledge for research and given lots of ideas with kindness. This research would not have been completed without him.

I further would like to thank all of my thesis committees: Associate Professor Krung Sinapiromsaran, Ph.D., Thap Panitanarak, Ph.D. and Assistant Professor Wittawat Kositwattanarerk, Ph.D. for their insightful suggestions and improving the quality of this research.

I wish to thank all of my teachers for sharing their knowledge and would like to thank all other lecturers and staffs of the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, especially, Ms. Wipa Pimpapan and Acting Sub Lt. Thinnakrit Sirisaengphraiwan, for their patience, encouragement and impressive advising. Moreover, I would like to thank all friends and my seniors in Applied Mathematics and Computer Science (AMCS) program for their useful advice, helpful comments and friendship over the course of my study.

Finally, I am also grateful to the Development and Promotion of Science and Technology Talents Project (DPST), Institute of the Promotion of Teaching Science and Technology (IPST), Thailand, which has supported my study and given a scholarship since 2015.

CONTENTS

CHAPTER I

INTRODUCTION

1.1 Motivation and Literature Surveys

Finding the modular inverse of a polynomial in a polynomial ring over a finite field $\mathbb{F}_q$ with a characteristic p is a classic problem in number theory. However, the algorithms for finding this problem is not efficiency in term of computational complexity, especially for high degree polynomials. This modular inverse plays an important role in error correcting codes and cryptography.

Let $\mathbb{F}_q$ be a finite field where $q = p^m$ and p is a prime number and $\mathbb{F}_q[x]$ be the set of polynomials over $\mathbb{F}_q$. The Euclidean algorithm is a well-known method for computing the modular inversion polynomial over $\mathbb{F}_q$. However, its complexity is $\mathcal{O}(n^2 \log n)$, which is not efficient for large n, where n is the maximum of degrees of the dividend and the divisor. The algorithm was improved for $p = 2$ by [1]. The improvement for more general p was carried out by R.T.Moenck [2]. This algorithm, named Half-GCD, reduces the performing steps by roughly half compared with the classic Euclidean algorithm. The cost analysis showed that it has the complexity of $\mathcal{O}(n \log^2 n)$ which is more efficient than the original Euclidean algorithm for every degree n. Cao and Cao [3] proposed an iterative algorithm with the complexity of $\mathcal{O}(n \log n)$ based on Newton idea [4] to find a modular inverse for the modulo x^n of an integer $n \geq 0$.

1.2 Research Objective

Following the same idea as that of Cao and Cao, this paper proposes an algorithm for finding the modular inverse of a polynomial under modulo $x^{p^r} - 1$, $x^{p^r} + 1$, $x^{2p^r} - 1$ for characteristic p, and $x^n - 1$ where p is a prime number and $n = 2^r d$ for some $r, d \in \mathbb{N}$ for characteristic 2. In other words, given a polynomial $f \in \mathbb{F}_q[x]$, we want to find the polynomial $g \in \mathbb{F}_q[x]$ such that $fg \equiv 1 \pmod{h_i}$ for $i = 1, 2, 3, 4$, where $h_1 = x^{p^r} - 1$, $h_2 = x^{p^r} + 1$, $h_3 = x^{2p^r} - 1$ and $h_4 = x^n - 1$.

1.3 Thesis Overview

This thesis is separated into four chapters organized as follows. First, Chapter 1 gives the motivation, objective and the overall works. Chapter 2 will describe the previous works and the properties of an algebraic concept used on our works. The main result of our thesis is organized in Chapter 3, which presents the algorithm for finding a modular inversion of a given polynomial in a polynomial ring over a finite field. The use of the algorithm is also illustrated by examples in this chapter. The rest of this thesis is a conclusion for summarizing our works and the future works which described in Chapter 4.

CHAPTER II

BACKGROUND KNOWLEDGE

2.1 Algebraic Concept

Algorithms used for calculating the inverse of polynomial modulo $x^n - 1$ where n is a form of $2^r d$ and r, d is a positive integer, rely on algebra of polynomial rings. So, in order to explain those, we need to be clear about understanding of fields and polynomial rings with some more in-depth concepts like finite field and its characteristic. Along with the algebra, a good understanding of some properties of polynomials is required.

2.1.1 Finite Field

We begin this section by recalling the notion of the finite field and their properties. For the reference, see [5–9]. Some related definitions are shown in the following.

Definition 2.1. A **field** is a set $\mathbb{F}$ together with two binary operations $+$ and $\cdot$ on $\mathbb{F}$ such that $(\mathbb{F}, +)$ is an abelian group with the identity 0 and $(\mathbb{F} - \{0\}, \cdot)$ is also an abelian group satisfying the following distributive law holds:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c), \quad \text{for all} \quad a, b, c \in \mathbb{F},$$

and for any field $\mathbb{F}$, let $\mathbb{F}^{\times} = \mathbb{F} - \{0\}$.

Definition 2.2. Let $\mathbb{F}$ be a field. If the number of element in $\mathbb{F}$ is infinite, $\mathbb{F}$ is called an **infinite field**. If the number of elements in $\mathbb{F}$ is finite, $\mathbb{F}$ is called a **finite field** or **Galois field**.

So, for any finite field, there are additional characteristic starting as follow.

Definition 2.3. Let $\mathbb{F}$ be a field and e be its identity. If for any positive integer m, we have $me \neq 0$, then we say that the **characteristic** of $\mathbb{F}$ is 0 or that $\mathbb{F}$ is a field of characteristic 0. If there exists a positive integer m such that $me = 0$, then the smallest positive integer p satisfying $pe = 0$ is called the characteristic of $\mathbb{F}$ and $\mathbb{F}$ is called a field of characteristic p.

Example 2.4. All of $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ are fields of characteristic 0, and $\mathbb{Z}_p := \{\overline{0}, \overline{1}, \ldots, \overline{p-1}\}$ is a field of characteristic p.

Theorem 2.5. *Let $\mathbb{F}$ be any field, then the characteristic of $\mathbb{F}$ is either 0 or a prime p.*

Corollary 2.6. *If $\mathbb{F}$ is a finite field, then the characteristic of $\mathbb{F}$ is not equal to 0.*

Theorem 2.7. *Let $\mathbb{F}_q$ be a field of characteristic $p, p \neq 0$, and a, b be any two polynomials of $\mathbb{F}_q[x]$, then*

$$(a+b)^p = a^p + b^p.$$

Similarly, we get the following corollary.

Corollary 2.8. *Let $\mathbb{F}_q$ be a field of characteristic $p, p \neq 0$ and a, b be any two polynomials of $\mathbb{F}_q[x]$, then*

$$(a-b)^p = a^p - b^p.$$

2.1.2 The ring of polynomials over a finite field

Let $\mathbb{F}_q$ be a finite field of q elements, whose characteristic is p. This section introduces the polynomial over $\mathbb{F}_q$. The structure of the set of all polynomials over $\mathbb{F}_q$ are characterized and their properties are presented. For the general reference, we refer to [8].

Consider all polynomials of the form

$$a_0 + a_1x + a_2x^2 + ... + a_nx^n, a_i \in \mathbb{F}_q$$

Here a_i is called the i^{th} **coefficient** of the polynomial. In this polynomial n is the largest integer for which $a_i \neq 0$. As such a_n is called the **leading coefficient** and n is called the **degree** of the polynomial. When the leading coefficient is 1, the polynomial is said to be **monic**. A part of a polynomial $a_i x^i$ is called a **term**. In addition, the set of all polynomials over $\mathbb{F}_q$ forms a ring, with addition and multiplication, called **polynomial ring over** $\mathbb{F}_q[x]$ and it is denoted by $\mathbb{F}_q[x]$, where x is called the **indeterminate** or **variable**. The next lemma presents a property on the degree of a polynomial as follows.

Lemma 2.9. *Let $f \in \mathbb{F}_q[x]$ be a polynomial of degree $m \geq 1$ with $f(0) \neq 0$. Then there exists a positive integer $e \leq q^m - 1$ such that $f(x)$ divides $x^e - 1$.*

Definition 2.10. Let $f \in \mathbb{F}_q[x]$ be a nonzero polynomial. If $f(0) \neq 0$, then the least positive integer e for which $f(x)$ divides $x^e - 1$ is called the **order** of f and denoted by $\operatorname{ord}(f) = \operatorname{ord}(f(x))$. If $f(0) = 0$, then $f(x) = x^h g(x)$, where $h \in \mathbb{N}$ and $g \in \mathbb{F}_q[x]$ with $g(0) \neq 0$ are uniquely determined; $\operatorname{ord}(f)$ is then defined to be $\operatorname{ord}(g)$.

Theorem 2.11. *Let c be a positive integer. Then the polynomial $f \in \mathbb{F}_q[x]$ with $f(0) \neq 0$ divides $x^c - 1$ if and only if ord(f) divides c.*

The greatest common divisor of two polynomials over a finite field is defined in the following.

Definition 2.12. Let f_1 and f_2 be polynomials over a finite field $\mathbb{F}_q[x]$. The polynomial $g \in \mathbb{F}_q[x]$ is a **greatest common divisor** of f_1 and f_2 which is denoted by $\gcd(f_1, f_2)$ if and only if g divides f_1 and f_2 and for every other element $d \in \mathbb{F}_q[x]$ such that d divides f_1 and f_2, then g is a divisor of d.

We get some results on the divisor of certain polynomials related to the greatest common divisor of their degrees.

Theorem 2.13. *If e_1 and e_2 are positive integers, then the greatest common divisor of $x^{e_1} - 1$ and $x^{e_2} - 1$ in $\mathbb{F}_q[x]$ is $x^d - 1$, where d is the greatest common divisor of e_1 and e_2.*

Theorem 2.14. *Let $g, f \in \mathbb{F}[x]$, where $f \neq 0$. Then there exists a unique $z \in \mathbb{F}[x]$ such that $z \equiv g \pmod{f}$ and $\deg(z) < \deg(f)$, namely, $z \coloneqq g \bmod f$.*

Theorem 2.15. *Let $g, f \in \mathbb{F}[x]$, with $f \neq 0$, and let $d := \gcd(g, f)$.*

(i) For every $h \in \mathbb{F}[x]$, the congruences $gz \equiv h$ (mod f) has a solution $z \in \mathbb{F}[X]$ if and only if $d \mid h$.

(ii) For every $z \in \mathbb{F}[x]$, we have $gz \equiv 0$ (mod f) if and only if $z \equiv 0$ (mod f/d).

(iii) For all $z, z' \in \mathbb{F}[x]$, we have $gz \equiv gz'$ (mod f) if and only if $z \equiv z'$ (mod f/d).

Part *(iii)* of Theorem 2.15 gives a cancellation law for polynomial congruences:

$$\text{if } \gcd(g, f) = 1 \text{ and } gz \equiv gz' \pmod{f}, \text{ then } z \equiv z' \pmod{f}.$$

We may generalize the "mod" operation for accordance with our work as follow. Suppose $g, h, f \in \mathbb{F}[x]$, with $f \neq 0$, $g \neq 0$, and $\gcd(g, f) = 1$. If s is the rational function $h/g \in \mathbb{F}[x]$, then we define s mod f to be the unique polynomial $z \in \mathbb{F}[x]$ satisfying

$$gz \equiv h \pmod{f} \text{ and } \deg(z) < \deg(f).$$

Theorem 2.16. *(Chinese Remainder Theorem) Let $\{f_i\}_{i=1}^k$ be a pairwise relatively prime family of non-zero polynomials in $\mathbb{F}[x]$, and let $g_1, \ldots, g_k$ be arbitrary polynomial in $\mathbb{F}[x]$. Then there exists a solution $g \in \mathbb{F}[x]$ to the system of congruences*

$$g \equiv g_i \ (mod\ f_i) \ (i = 1, \ldots, k).$$

Moreover, any $g' \in \mathbb{F}[x]$ is a solution to this system of congruences if and only if $g \equiv g'$ (mod f_i), where $f := \prod_{i=1}^k f_i$.

2.2 Euclidean Algorithm

The usual and well-known Euclidean division for integers is stated that for integers $a, b > 0$, there exists unique integers $q > 0$ and $0 \leq r < a$ such that $a = bq + r$. This statement can be translated to an analogous version in the ring of polynomials as follows.

Theorem 2.17. *Let $\mathbb{F}_q$ be a finite field with characteristic p, let $f, g \in \mathbb{F}_q[x]$ be polynomials of degrees ≥ 0 Then there exists unique polynomials pair $q, r \in \mathbb{F}_q[x]$ such that $f = gq + r$ and* $\deg(r) < \deg(g)$.

Proof. In order to prove the following theorem, we first let

$$f(x) = a_0 + a_1x + \cdots + a_nx^n,$$
$$g(x) = b_0 + b_1x + \cdots + b_mx^m,$$

where $n = \deg(f), m = \deg(g)$, with $a_n \neq 0$, and $b_m \neq 0$ is a unit in $\mathbb{F}_q$. It should be noted that b_m is the unit which guarantees that the existence of its inverse even though inverses for other elements do not necessarily exist in R. According to [10], induction on the degree n is applied to construct the proof.

As the base step of the induction, in the case of $n = 0$ and $\deg(g) > \deg(f)$, we let $r = f$ and $q = 0$. Also, if $\deg(f) = \deg(g) = 0$, we let $r = 0$ and $q = a_nb_m^{-1}$.

Next, we assume the theorem is proved for all polynomials which the degree is less than n. Also, we assume that $\deg(g) \leq \deg(f)$, because if this is not true, we just let $q = 0$ and $r = f$. Now, it can be written as

$$f(x) = a_nb_m^{-1}x^{n-m}g(x) + f_1(x),$$

where $n > \deg(f_1)$. By applying the induction, we can find q_1, r and write f as

$$f(x) = a_nb_m^{-1}x^{n-m}g(x) + q_1(x)g(x) + r(x)$$

with $\deg(r) < \deg(g)$. Finally, to achieve the proof, we thus define

$$q(x) = a_nb_m^{-1}x^{n-m} + q_1(x).$$

In this respect, the previous theorem above proves only the existence of both q and r.

However, this makes no claim whether q and r are unique or not. To prove the uniqueness, see more details in [10], Lang starts by assuming there exists two instances of q and r such that

$$f = q_1 g + r_1 = q_2 g + r_2,$$

where $\deg(g) > \deg(r_1)$ and $\deg(g) > \deg(r_2)$. Reformatting the above equation yields

$$(q_1 - q_2)g = r_2 - r_1.$$

The leading coefficient of g was assumed to be a unit in R, we can conclude that

$$\deg\left((q_1 - q_2)g\right) = \deg(q_1 - q_2) + \deg(g).$$

However, we know that $\deg(g) > \deg(r_2 - r_1)$, and also know that it can be only if $\deg(q_1 - q_2) = 0$, which means that $q_1 = q_2$. Therefore, we have consequently $r_1 = r_2$. $\square$

Euclidean algorithm is the process of applying Euclidean Division in succession several times to find the greatest common divisor of polynomials and it is the well-know method for computing the modular inversion polynomial over $\mathbb{F}_q$ which is the main point of our project. The process of Euclidean algorithm is shown as follow.

Let $\mathbb{F}_q$ be a finite field with characteristic p and $P_0, P_1 \in \mathbb{F}_q[x]$ where $n = \deg P_0 > \deg P_1 \geq 0$. Then

$$\begin{bmatrix} P_0 \\ P_1 \end{bmatrix} \xrightarrow{M_1} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \xrightarrow{M_2} \begin{bmatrix} P_2 \\ P_3 \end{bmatrix} \xrightarrow{M_3} \cdots \xrightarrow{M_{h-1}} \begin{bmatrix} P_{h-1} \\ P_h \end{bmatrix} \xrightarrow{M_h} \begin{bmatrix} P_h \\ 0 \end{bmatrix}$$

where $M_i = \begin{bmatrix} Q_i & 1 \\ 1 & 0 \end{bmatrix}$ for some $Q_i \in \mathbb{F}_q[x]$, $\begin{bmatrix} P_i \\ P_{i+1} \end{bmatrix} = M_{i+1} \begin{bmatrix} P_{i+1} \\ P_{i+2} \end{bmatrix}$,
for all $i \in \{1, \ldots, h-1\}$.

We illustrate this Euclidean division algorithm with an example.

Example 2.18. Let $\mathbb{F}_q = \mathbb{Z}_2, a(x) = x^5 + x^4 + x^2 + 1$ and $b(x) = x^3 + x + 1$.

The division algorithm can be performed with the help of the following scheme.

$$
\begin{array}{r|rrrrrr|l}
x^3+x+1 & x^5 & +x^4 & & +x^2 & & +1 & x^2+x+1 \\
 & x^5 & & +x^3 & +x^2 & & & \\ \cline{2-7}
 & & x^4 & +x^3 & & & +1 & \\
 & & x^4 & & +x^2 & +x & & \\ \cline{3-7}
 & & & +x^3 & +x^2 & +x & +1 & \\
 & & & x^3 & & +x & +1 & \\ \cline{4-7}
 & & & & x^2 & & &
\end{array}
$$

So we can obtain the quotient $q(x) = x^2 + x + 1$ and the remainder $r(x) = x^2$ and the result can be written as

$$x^5 + x^4 + x^2 + 1 = (x^2 + x + 1)(x^3 + x + 1) + x^2.$$

For the convenience of writing and computing we denoted the polynomial

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

by simplified form $a_n a_{n-1} ... a_1 a_0$. For example, one can write the polynomials $x^5 + x^4 + x^2 + 1$ and $x^3 + x + 1$ as 1 1 0 1 0 1 and 1 0 1 1 respectively. Doing this, the above scheme can be simplified as

$$
\begin{array}{r|cccccc|l}
1011 & 1 & 1 & 0 & 1 & 0 & 1 & 111 \\
 & 1 & 0 & 1 & 1 & & & \\ \cline{2-7}
 & & 1 & 1 & 0 & 0 & 1 & \\
 & & 1 & 0 & 1 & 1 & & \\ \cline{3-7}
 & & & 1 & 1 & 1 & 1 & \\
 & & & 1 & 0 & 1 & 1 & \\ \cline{4-7}
 & & & & 1 & & &
\end{array}
$$

However, its complexity is $\mathcal{O}(n^2 \log n)$, which is not efficient for large n, where n is the maximum of degrees of the dividend and the divisor. The algorithm was improved for $p = 2$ by [1].

2.3 Fast division of polynomials

In Section 2.2, the Euclidean division is one of the key building blocks for the factorization algorithm which presented later. Hence, it is important to have a fast implementation for it. Using the long division algorithm of polynomials, will yield an asymptotic complexity of $\mathcal{O}(\deg(a)\deg(b))$, where $a, b \in \mathbb{F}_q$, which for all practical purposes is the same as $\mathcal{O}(n^2)$, where n is the maximum degree of their polynomials. Newton's method is a numerical method for finding a root of a real-valued function $f(x)$. It starts by approximating, or just selecting any starting point x_0 and then computing next approximation by forming a tangent line through point $(x_0, f(x_0))$ and using the point where this tangent intersects x-axis as the next approximation. This iterative step can be expressed as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}.$$

Next, when we are given integers $a, b > 0$, we would like to compute integers $q, r \geq 0$ such that $a = bq + r$ and $r < b$. We observe that q may be computed as $q = [a/b]$, and when q is known, we can compute $r = a - bq$. So, to determine the value of q, it suffices to get a close enough approximation of $c = b^{-1}$ and then multiply ac and round it down to the closest integer. This we can achieve by using Newton's method on function $f(x) = x^{-1} - b$. With this, the iterative step is as follows:

$$x_{i+1} = x_i - \frac{x_i^{-1} - b}{-x_i^{-2}} = 2x_i - bx_i^2.$$

Now, as it turns out, this Newton's method translates to polynomials over commutative rings with unity too. In their publication Zhengjun Cao and Hanyue Cao [3] improve upon some earlier version of this algorithm and provide all missing steps for implementing it. Details of their work are out of the scope of this thesis, but the resulting algorithm will be introduced next.

In 2012, an algorithm relied on the first *reversing* coefficients of a polynomial and computing its modulo with a large power of variable x is proposed by Cao and Cao. Their main idea is based on the iterative step of the Newton's method. In their results, the reversing coefficients of a polynomial $f(x)$ is denoted with $rev(f) = rev_{\deg(f)}(f)$ and can be achieved with simply calculating $x^{\deg(f)}f(x^{-1})$. For example:

$$\begin{aligned} f(x) &= a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \\ rev(f) &= x^n f\left(x^{-1}\right) \\ &= x^n \left(a_0 + a_1x^{-1} + a_2x^{-2} + \ldots + a_nx^{-n}\right) \\ &= a_0x^n + a_1^{n-1} + \ldots + a_{n-1}x + a_n. \end{aligned}$$

According to Theorem 2.17, the Euclidean division for a polynomial, which states that let R be a commutative ring, $a, b \in R[x]$ be two polynomials with degrees greater than 0, and b be monic. Up to unit factors, there exists a unique pair of polynomials $q, r \in R[x]$ such that $a = bq + r$ where $\deg(r) < \deg(b)$.

Cao and Cao begin setting $a = bq+r$ by substituting x with x^{-1} and by multiplying with x^n, $n = \deg(a)$ and $m = \deg(b)$. We have

$$\begin{gathered} x^n a(x^{-1}) = (x^{n-m}q(x^{-1}))(x^m b(x^{-1})) + x^{n-m+1}(x^{m-1}r(x^{-1})) \\ \Leftrightarrow \\ rev_n(a) = rev_{n-m}(q) \cdot rev_m(b) + x^{n-m+1}rev_{m-1}(r), \end{gathered}$$

which becomes

$$rev_n(a) = rev_{n-m}(q) \cdot rev_m(b) \mod x^{n-m+1}.$$

Roughly speaking, Cao and Cao mention that because b is monic, $rev_m(b)$ has a constant coefficient 1, and thus, $rev_m(b)$ is invertible modulo x^{n-m+1}. If we set $g \in R[x]$ to be invertible mod f, we then have to be capable to find a polynomial $h \in R[x]$ such that $gh \equiv 1 \mod f$. This makes

$$rev_{n-m}(q) = rev_n(a) \cdot rev_m(b)^{-1} \mod x^{n-m+1},$$

and hence, we have

$$q = rev_{n-m}(rev_{n-m}(q)).$$

In brief, Cao and Cao [3] proposed an iterative algorithm with the complexity of $\mathcal{O}(n \log n)$ based on Newton idea [4] to find a modular inverse for the modulo x^n for an integer $n \geq 0$. In the other word, a problem of finding $g(x) \in R[x]$ such that $fg \equiv 1 \mod x^n$, when $f(x) \in R[x]$ is given, $f(0) = 1$ and $n \in \mathbb{N}$. Furthermore, they observe that when l is a power of two, if $fg_i \equiv 1 \mod x^{2^i}$, then

$$x^{2^i} | (1 - fg_i),$$
$$x^{2^{i+1}} | (1 - fg_i)^2,$$
$$x^{2^{i+1}} | 1 - f(2g_i - fg_i^2).$$

Hence, the iteration step to solve the problem is $g_{i+1} = 2g_i - fg_i^2$ and $i \in \{0, 1, 2, \ldots, l-1\}$. This leads to the following result:

Theorem 2.19. *Let R be a commutative ring and $f, g_0, g_1, \ldots, \in R[x]$, with $f(0) = 1, g(0) = 1$ and*

$$g_{i+1} \equiv 2g_i - fg_i^2 \mod x^{2^{i+1}}$$

for all i. Then $fg_i \equiv 1 \mod x^{2^i}$ for all $i \geq 0$.

By Theorem 2.19, we can obtain the following algorithm to compute the inverse of $f \mod x^l$. Note that the log in the pseudo code refers to the binary logarithm.

Algorithm 1 Newton Iteration Algorithm

Input: $f \in R[x]$ with $f(0) = 1$, and $l \in \mathbb{N}$.

Output: $g \in R[x]$ satisfying $fg \equiv 1 \pmod{x^l}$.

Set initial $g(0) \leftarrow 1, r \leftarrow \lceil \log l \rceil$

for $i = 1, 2, 3, \ldots, r$ **do**

  $g_i \leftarrow (2g_{i-1} - fg_{i-1}^2) \text{ rem } x^{2^i}$

return g_r

When all these are combined, we get the following algorithm:

Algorithm 2 Fast Division Algorithm

If $\deg(a) < \deg(b)$ **return** $q = 0$ and $r = a$
Let $m = \deg(a) - \deg(b)$ and $r = \lceil \log(m) + 1 \rceil$
Let $f = rev(b)$
for $i = 1, 2, 3, \ldots, r$ **do**
    $g_i \leftarrow (2g_{i-1} - fg_{i-1}^2) \bmod x^{2^i}$
Let $s = rev(a)g_r \mod x^{m+1}$
return $q = x^{m-\deg(s)} rev(s)$ and $r = a - bq$

Definition 2.20. Let R be a ring. A function $\mathrm{M} : \mathbb{N} \to \mathbb{R}^+$ is called a multiplication time for $R[x]$ if polynomials in $R[x]$ of degree less than n can be multiplied by using at most $M(n)$ operations in R.

Assume the multiplicative time satisfies

$$M(n)/n \geq M(m)/m \text{ if } n \geq m, \quad M(mn) \leq m^2 M(n),$$

for all $n, m \in \mathbb{N}_{>0}$. So the first inequality yields the superlinearity properties

$$M(mn) \geq mM(n), \quad M(m+n) \geq M(m) + M(n) \text{ and } M(n) \geq n.$$

We determine the multiplicative time $M(n)$ using Fast Fourier Transform (FFT) is the same order as $\mathcal{O}(n \log n)$, where n is the degree of polynomial.

Theorem 2.21. *Algorithm 1 improves computational complexity which uses at most* $3M(2^r) + 2^r \in \mathcal{O}(M(n)) = \mathcal{O}(n \log n)$ *operations in* R*, where* $n = \ell = 2^r$*.*

Proof. Assume that $fg_i \equiv 1 \pmod{x^{2^i}}$ for all $i \geq 0$, then

$$fg_{i+1} \equiv 1 \pmod{x^{2^{i+1}}}.$$

It follow that we can reduce to $fg_{i+1} \equiv 1 \pmod{x^{2^i}}$. Since $\gcd(f, x^{2^i}) = 1$, then

$$g_{i+1} \equiv g_i \pmod{x^{2^i}} \tag{2.1}$$

for all $i \geq 0$. Consider the equation

$$g_i \equiv 2g_{i-1} - fg_{i-1}^2 \pmod{x^{2^i}}. \tag{2.2}$$

In step 2 of Algorithm 1 and equation 2.1. The negative of the upper half of fg_{i-1}^2 modulo x^{2^i} is the same as g_i and the lower half of g_i is the same as the g_{i-1}. So, the cost for one iteration of the ith step is

- $M(2^i)$ for the computation of g_{i-1}^2
- $M(2^{i-1})$ for the computation of product fg_{i-1}^2
- 2^{i-1} for the computation of the negative of upper half of fg_{i-1}^2

Thus, we have $M(2^i) + M(2^{i-1}) + 2^{i-1}$ operations in step 2 of Algorithm 1. So the total running time for this algorithm is

$$\begin{aligned}
\sum_{i=1}^{r} \left(M(2^i) + M(2^{i-1}) + 2^{i-1}\right) &\leq \sum_{i=1}^{r} \left(\frac{3}{2}M(2^i) + 2^{i-1}\right) \\
&\leq \left(\sum_{i=1}^{r} 2^{i-r}\right) \left(\frac{3}{2}M(2^r) + 2^{r-1}\right) \\
&< 3M(2^r) + 2^r \\
&\in \mathcal{O}\left(M(2^r)\right) = \mathcal{O}\left(n \log n\right)
\end{aligned}$$

Hence the complexity of Newton iterative algorithm is $\mathcal{O}(n \log n)$ as required. $\square$

2.4 Half-GCD algorithm

Let $a \in \mathbb{F}_q[x]$ and $b \in \mathbb{F}_q[x]$ be the dividend and the modulus, respectively. The algorithm first determine a regular 2×2 matrix, M, which reduces $\gcd(a, b)$ to $\gcd(c, d)$ where $c, d \in \mathbb{F}_q[x]$, $\deg c \geq (\deg a)/2 > \deg d$, and $M[a\ b]' = [c\ d]'$. Its complexity is $\mathcal{O}(n \log^2 n)$ which is more efficient than the original Euclidean algorithm for all degree n. The following figure describe the algorithm of Half-GCD and the Euclidean steps.

Let $\mathbb{F}_q$ be a finite field with characteristic p and $P_0, P_1 \in \mathbb{F}_q[x]$ where $n = \deg P_0 > \deg P_1 \geq 0$. Then

$$\begin{bmatrix} P_0 \\ P_1 \end{bmatrix} \xrightarrow{M_1} \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \xrightarrow{M_2} \cdots \xrightarrow{M_k} \begin{bmatrix} P_k \\ P_{k+1} \end{bmatrix} \xrightarrow{M_{k+1}} \begin{bmatrix} P_{k+1} \\ P_{k+2} \end{bmatrix} \xrightarrow{M_{k+2}} \cdots \xrightarrow{M_{h-1}} \begin{bmatrix} P_{h-1} \\ P_h \end{bmatrix} \xrightarrow{M_h} \begin{bmatrix} P_h \\ 0 \end{bmatrix}$$

where $M_i = \begin{bmatrix} Q_i & 1 \\ 1 & 0 \end{bmatrix}$ for some $Q_i \in \mathbb{F}_q[x]$, $\begin{bmatrix} P_i \\ P_{i+1} \end{bmatrix} = M_{i+1} \begin{bmatrix} P_{i+1} \\ P_{i+2} \end{bmatrix}$,
for all $i \in \{1, \ldots, h-1\}$ and $\deg P_k \geq \dfrac{\deg P_0}{2} > \deg P_{k+1}$.

The Half-GCD algorithm can be written in the following pseudo code with these notations.

- Let $\|A\|$ denote the degree of polynomial $A \in \mathbb{F}[x]$.
- A regular matrix M is a product of zero or more elementary matrices
  $M = M_1 M_2 \ldots M_k, \quad k \geq 0$
- (A **div** B) donote the quotient of A divided by B.
- (A **mod** B) denote the remainder of A divided by B.

Algorithm 3 Algorithm Polynomial HGCD(A, B)

Input: A, B are univariate polynomials with $\|A\| > \|B\| \geq 0$.

Output: a regular matrix M which reduces (A, B) to (C', D') where $\|C'\|, \|D'\|$ straddle $\|A\|/2$.

$m \leftarrow \lceil \frac{\|A\|}{2} \rceil$; $\{$This is the magic threshold$\}$

if $\|B\| < m$ then **return** (E);

$\begin{bmatrix} A_0 \\ B_0 \end{bmatrix} \leftarrow \begin{bmatrix} A\mathbf{div}X^m \\ B\mathbf{div}X^m \end{bmatrix}$.

$\{$now $\|A_0\| = m'$ where $m + m' = \|A\|\}$

$R \leftarrow \text{hGCD}(A_0, B_0)$;

$\{\lceil \frac{m'}{2} \rceil$ is the magic threshold for this recursive call$\}$

$\begin{bmatrix} A' \\ B' \end{bmatrix} \leftarrow R^{-1} \begin{bmatrix} A \\ B \end{bmatrix}$;

if $\|B'\| < m$ then **return** (R);

$\mathbb{Q} \leftarrow A'\mathbf{div}B'$; $\begin{bmatrix} C \\ D \end{bmatrix} \leftarrow \begin{bmatrix} B' \\ A' \mod B' \end{bmatrix}$;

$l \leftarrow \|C\|; k \leftarrow 2m - l$; $\{$now $l - m < \lceil \frac{m'}{2} \rceil\}$

$C_0 \leftarrow C\mathbf{div}X^k; D_0 \leftarrow D\mathbf{div}X^k$; $\{$now $\|C_0\| = 2(l - m)\}$

$S \leftarrow \text{hGCD}(C_0, D_0)$;

$\{l - m$ is magic threshold for this recursive call.$\}$

$M \leftarrow R \cdot \langle \mathbb{Q} \rangle \cdot S$; **return** (M);

Algorithm 4 Polynomial co-GCD Algorithm

Input: A pair of polynomials with $\deg P_0 > \deg P_1$.

Output: A regular matrix $M = \text{co-GCD}(P_0, P_1)$ such that

$$\begin{bmatrix} P_0 \\ P_1 \end{bmatrix} \xrightarrow{M} \begin{bmatrix} \text{GCD}(P_0, P_1) \\ 0 \end{bmatrix}.$$

1: Compute $M_0 \leftarrow \text{hGCD}(P_0, P_1)$.

2: Recover P_2, P_3 via

$$\begin{bmatrix} P_2 \\ P_3 \end{bmatrix} \leftarrow M_0^{-1} \begin{bmatrix} P_0 \\ P_1 \end{bmatrix}.$$

3: if $P_3 = 0$ then **return** (M_0).

else, perform one step of the Euclidean algorithm,

$$\begin{bmatrix} P_2 \\ P_3 \end{bmatrix} \xrightarrow{M_1} \begin{bmatrix} P_3 \\ P_4 \end{bmatrix}.$$

where M_1 is an elementary matrix.

4: if $P_4 = 0$ then **return** $(M_0 M_1)$.

else, recursively compute $M_2 \leftarrow \text{co-GCD}(P_3, P_4)$

return $(M_0 M_1 M_2)$.

For the complexity analysis, the Half-GCD algorithm make two recursive calls to itself. The work in each call to the algorithm, exclusive of recursion, is $\mathcal{O}(n \log n)$. Hence the computational complexity $T'(n)$ of this Half-GCD algorithm satisfies

$$T'(n) = 2T'(n/2) + \mathcal{O}(n \log n) = \mathcal{O}(n \log^2 n).$$

Assume that $T'(\alpha n) = \mathcal{O}(\alpha T'(n))$ for all constant α, then the computational complexity $T(n)$ of the co-GCD algorithm satisfies

$$\begin{aligned} T(n) &= T'(n) + \mathcal{O}(n \log n) + T(n/2) \\ &= \mathcal{O}(T'(n) + T'(n/2) + T'(n/4) + \cdots) \\ &= \mathcal{O}(T'(n) + \frac{1}{2}T'(n) + \frac{1}{4}T'(n) + \cdots) \\ &= \mathcal{O}(2T'(n)) = \mathcal{O}(T'(n)) \end{aligned}$$

Theorem 2.22. *The computational complexity of the Half-GCD algorithm is $\mathcal{O}(n \log^2 n)$.*

CHAPTER III

MAIN RESULTS

3.1 Polynomial Modular Inversion

This section proposes the main result given in Theorem 3.1, 3.3, 3.4 and 3.5 for finding a modular inverse of a polynomial f in $\mathbb{F}_q[x]$ under the modulo $x^{p^r}-1$, $x^{p^r}+1$, $x^{2p^r}-1$ and x^n-1 where $n=2^r d$ for some $r,d\in\mathbb{N}$. The key idea of the proof is similar to that presented by Cao and Cao; see [3]. The two main differences between their results and ours are the modulo and its domain. Their modulo is x^n and the problem domain is a polynomial ring over a ring, while our modulo is x^n-1 and our problem domain is a polynomial ring over a finite field.

Theorem 3.1. *Let $f(x)$ be a polynomial over $\mathbb{F}_q$ of characteristic p. If $f(1)\neq 0$ and there exists a sequence $\{g_i(x)\}_{i\geq 0}$ of polynomials in $\mathbb{F}_q[x]$ with $g_0=f(1)^{-1}$ satisfying the iterative congruent relation*

$$g_{i+1}\equiv f^{p-1}g_i^p \pmod{x^{p^{i+1}}-1} \quad (i\geq 0), \tag{3.1}$$

if and only if g_i is an inverse of f satisfying $fg_i\equiv 1 \pmod{x^{p^i}-1}$ for all $i\geq 0$.

Proof. Assume that $f(1)\neq 0$. Let $\{g_i(x)\}_{i\geq 0}$ be a sequence of polynomials over $\mathbb{F}_q$ with $g_0=f(1)^{-1}$ satisfying the iterative congruent relation

$$g_{i+1}\equiv f^{p-1}g_i^p \pmod{x^{p^{i+1}}-1} \quad (i\geq 0).$$

With the assumption $g_0=f(1)^{-1}$, we have $f(x)g_0(x)=f(x)/f(1)$. Since $f(1)\neq 0$, we obtain that 1 is a root of $f(x)/f(1)-1$. i.e., $f(x)g_0(x)\equiv 1 \pmod{x-1}$. Next, let $i\geq 0$.

Assume that $fg_i \equiv 1 \pmod{x^{p^i} - 1}$. Then, there exists $h \in \mathbb{F}_q[x]$ such that

$$f^p g_i^p - 1 = (fg_i - 1)^p = ((x^{p^i} - 1)h)^p = (x^{p^{i+1}} - 1)h^p.$$

This implies that $f^p g_i^p \equiv 1 \pmod{x^{p^{i+1}} - 1}$. Since, by assumption, $g_{i+1} \equiv f^{p-1} g_i^p \pmod{x^{p^{i+1}} - 1}$, we get $fg_{i+1} \equiv 1 \pmod{x^{p^{i+1}} - 1}$.

Conversely, assume that $fg_i \equiv 1 \pmod{x^{p^i} - 1}$ for all $i \geq 0$. For $i = 0$, we immediately obtain that $f(1) \neq 0$ and it follows that $g_0(x)$ can be formed a constant $1/f(1) \in \mathbb{F}_q$. Set $g_0 = 1/f(1)$. For $i \geq 1$, by assumption, we have $fg_i \equiv 1 \pmod{x^{p^i} - 1}$, i.e. there exists $h \in \mathbb{F}_q[x]$ such that $1 - fg_i = (x^{p^i} - 1)h$. This implies that

$$(x^{p^{i+1}} - 1)h^p = ((x^{p^i} - 1)h)^p = (1 - fg_i)^p = 1 - f^p g_i^p = 1 - f(f^{p-1} g_i^p).$$

Since $fg_{i+1} \equiv 1 \pmod{x^{p^{i+1}} - 1}$, we can choose $g_{i+1} = f^{p-1} g_i^p$. By the induction on i, the proof is complete. $\square$

The computational algorithm for Theorem 3.1 is shown in the following pseudo code.

Algorithm 5 Iterative algorithm for Theorem 3.1

Input: $r \in \mathbb{N}_0$ and $f \in \mathbb{F}_q[x]$ with $f(1) \neq 0$.
Output: $g \in \mathbb{F}_q[x]$ satisfying $fg \equiv 1 \pmod{x^{p^r} - 1}$.
1: Set initial $g_0 \leftarrow (f(1))^{-1}$
2: **for** $i = 1, 2, 3, \ldots, r$ **do**
  $g_i \leftarrow f^{p-1} g_{i-1}^p \text{ rem } (x^{p^i} - 1)$
3: **return** g_r

The implementation of the algorithm is illustrated in the examples below.

Example 3.2. Given $f(x) = x^2 + x + 2$ and $h(x) = x^{27} - 1$ under $\mathbb{F}_q[x] = \mathbb{F}_9[x]$. To seek a polynomial g which makes $fg \equiv 1 \pmod{h}$. Applying Theorem 3.1 yields $g_0 = 1/f(1) = 1$, and the sequence $\{g_i\}_{i\geq 1}$ can be calculated as follows.

- For $g_1 \equiv f^{p-1}g_0^p = (x^2+x+2)^2 \equiv 2x^2+2x \pmod{x^3-1}$, there exists $g_1 = 2x^2+2x$.

- For $g_2 \equiv f^{p-1}g_1^p = (x^2+x+2)^2(2x^2+2x)^3 \equiv x^8+x^7+x^5+2x^4+2x^3+2x+1 \pmod{x^9-1}$, there exists $g_2 = x^8+x^7+x^5+2x^4+2x^3+2x+1$.

- For $g_3 \equiv f^{p-1}g_2^p = (x^2+x+2)^2(x^8+x^7+x^5+2x^4+2x^3+2x+1)^3 \equiv 2x^{26}+2x^{25}+2x^{23}+x^{22}+x^{21}+x^{19}+2x^{18}+2x^{17}+2x^{15}+x^{14}+x^{13}+x^{11}+2x^{10}+2x^9+2x^7+x^6+x^5+x^3+2x^2+2x \pmod{x^{27}-1}$, there exists $g_3 = 2x^{26}+2x^{25}+2x^{23}+x^{22}+x^{21}+x^{19}+2x^{18}+2x^{17}+2x^{15}+x^{14}+x^{13}+x^{11}+2x^{10}+2x^9+2x^7+x^6+x^5+x^3+2x^2+2x$.

It should be noted that this process provides the sequence $\{g_i\}_{i\geq 1}$ which are the inversion of f under modulo $x^{3^i}-1$, respectively. For instance, given $r = 100$ or other words $h(x) = x^{3^{100}}-1$, we can continue this process until we have g_{100}. Thus, $g = g_{100}$ is a polynomial modular inversion modulo $x^{3^{100}}-1$, as required.

We change our focus to another interesting case of modulo, i.e., $\bmod\,(x^{p^i}+1)$. Note that $x^{p^i}+1$ is congruent to $x^{p^i}-1$ in the case of $p=2$. Theorem 3.3 gives an iterative method for polynomials modular inversion modulo $x^{p^i}+1$ over a finite field. In addition, Theorem 3.4 presents an iterative method for polynomials modular inversion modulo $x^{2p^i}-1$ over a finite field by applying Theorems 3.1 and 3.3.

Theorem 3.3. *Let $f(x)$ be a polynomial over $\mathbb{F}_q$. If $f(-1) \neq 0$ and there exists a sequence $\{h_i(x)\}_{i\geq 0}$ of polynomials in $\mathbb{F}_q[x]$ with $h_0 = f(-1)^{-1}$ satisfying the iterative congruent relation*

$$h_{i+1} \equiv f^{p-1}h_i^p \pmod{x^{p^{i+1}}+1} \quad (i \geq 0), \tag{3.2}$$

if and only if h_i is an inverse of f satisfying $fh_i \equiv 1 \pmod{x^{p^i}+1}$ for all $i \geq 0$.

Proof. Assume that $f(-1) \neq 0$. Let $\{h_i(x)\}_{i\geq 0}$ be a sequence of polynomials over $\mathbb{F}_q$ with $h_0 = f(-1)^{-1}$ satisfying the iterative congruent relation

$$h_{i+1} \equiv f^{p-1}h_i^p \pmod{x^{p^{i+1}}+1} \quad (i \geq 0).$$

With the assumption $h_0 = f(-1)^{-1}$, we have $f(x)h_0(x) = f(x)/f(-1)$. Since $f(-1) \neq 0$, we obtain that -1 is a root of $f(x)/f(-1) - 1$. i.e., $f(x)h_0(x) \equiv 1 \pmod{x+1}$. Next, let $i \geq 0$. Assume that $fh_i \equiv 1 \pmod{x^{p^i}+1}$. Then, there exists $k \in \mathbb{F}_q[x]$ such that

$$f^p h_i^p + 1 = (fh_i + 1)^p = ((x^{p^i}+1)k)^p = (x^{p^{i+1}}+1)k^p.$$

This implies that $f^p h_i^p \equiv 1 \pmod{x^{p^{i+1}}+1}$. Since, by assumption, $h_{i+1} \equiv f^{p-1}h_i^p \pmod{x^{p^{i+1}}+1}$, we get $fh_{i+1} \equiv 1 \pmod{x^{p^{i+1}}+1}$.

Conversely, assume that $fh_i \equiv 1 \pmod{x^{p^i}+1}$ for all $i \geq 0$. For $i = 0$, we immediately obtain that $f(-1) \neq 0$ and it follows that $h_0(x)$ can be formed a constant $1/f(-1) \in \mathbb{F}_q$. Set $h_0 = 1/f(-1)$. For $i \geq 1$, by assumption, we have $fh_i \equiv 1 \pmod{x^{p^i}+1}$, i.e. there exists $k \in \mathbb{F}_q[x]$ such that $1 - fh_i = (x^{p^i}+1)k$. This implies that

$$(x^{p^{i+1}}+1)k^p = ((x^{p^i}+1)k^p = (1+fh_i)^p = 1 + f^p h_i^p = 1 + f(f^{p-1}h_i^p).$$

Since $fh_{i+1} \equiv 1 \pmod{x^{p^{i+1}}+1}$, we can choose $h_{i+1} = f^{p-1}h_i^p$. By the induction on i, the proof is complete. □

Theorem 3.4. *Given a prime number p greater than 2. Let $i \geq 1$ and assume that g_i and h_i be polynomials modular inversion modulo $x^{p^i}-1$ and modulo $x^{p^i}+1$, respectively. If*

$$j_i \equiv -\frac{1}{2}h_i(x^{p^i}-1) + \frac{1}{2}g_i(x^{p^i}+1) \pmod{x^{2p^i}-1},$$

then j_i is a polynomial modular inversion modulo $x^{2p^i}-1$, i.e.,

$$fj_i \equiv 1 \pmod{x^{2p^i}-1}.$$

Proof. Suppose that

$$fj_i \equiv f\left(-\frac{1}{2}h_i(x^{p^i}-1) + \frac{1}{2}g_i(x^{p^i}+1)\right) \pmod{x^{2p^i}-1}.$$

Since $fh_i \equiv 1 \pmod{x^{p^i}+1}$ and $fg_i \equiv 1 \pmod{x^{p^i}-1}$, we obtain that

$$fh_i(x^{p^i}-1) \equiv (x^{p^i}-1) \pmod{x^{2p^i}-1}$$
$$fg_i(x^{p^i}+1) \equiv (x^{p^i}+1) \pmod{x^{2p^i}-1}.$$

This implies that $fj_i \equiv -\frac{1}{2}(x^{p^i}-1) + \frac{1}{2}(x^{p^i}+1) = 1 \pmod{x^{2p^i}-1}$, as required. $\square$

Let $\mathbb{F}$ be a finite field with characteristic 2. Consider the case of polynomial modular inversion modulo $x^{2^r d}-1$. Let

$$n = 2^r d \in \mathbb{N} \quad \text{for some } r, d \in \mathbb{N}.$$

We can use Half-GCD algorithm (for the best now) for computing the inverse of f modulo $x^d - 1$ and continue with the problem of polynomial modular inversion modulo $x^{2^r d}-1$ which describe in the next theorem.

Theorem 3.5. *Let $\mathbb{F}_q$ be a finite field with characteristic 2. Let $f, g_0 \in \mathbb{F}_q[x]$ satisfying $fg_0 \equiv 1 \pmod{x^d-1}$, where d is a natural number. If*

$$g_{i+1} \equiv fg_i^2 \pmod{x^{2^{i+1}d}-1}, \quad \textit{for all } i \in \{0, 1, 2, \ldots, r-1\}.$$

Then g_i is a polynomial modular inversion modulo $x^{2^i d}-1$, i.e.,

$$fg_i \equiv 1 \pmod{x^{2^i d}-1}, \quad \textit{for all } i \in \{0, 1, 2, \ldots, r\}.$$

Proof. Let $\mathbb{F}_q$ be a finite field with characteristic 2 and let $\{g_i(x)\}_{i\geq 0}$ be a sequence of polynomials over $\mathbb{F}_q$ satisfying the iterative congruent relation

$$g_{i+1} \equiv fg_i^2 \pmod{x^{2^{i+1}d}-1}, \quad (i \geq 0).$$

In the case of $i = 0$ has claimed by the assumption, $fg_0 \equiv 1 \pmod{x^d - 1}$. Next, let $i > 0$. Assume that $fg_i \equiv 1 \mod x^{2^i d} - 1$. Then, there exists $k \in \mathbb{F}_q[x]$ such that

$$\begin{aligned} f^2 g_i^2 - 1 &= ((x^{2^i d} - 1)k)^2 \\ &= (x^{2^{i+1} d} - 1)k^2 \end{aligned}$$

This implies that $f^2 g_i^2 \equiv 1 \pmod{x^{2^{i+1}d} - 1}$. By assumption, $g_{i+1} \equiv f g_i^2 \pmod{x^{2^{i+1}d} - 1}$, we have $fg_{i+1} \equiv f^2 g_i^2 \pmod{x^{2^{i+1}d} - 1}$. Hence, $fg_{i+1} \equiv 1 \pmod{x^{2^{i+1}d} - 1}$ as required. □

The computational algorithm for the polynomial modular inversion modulo $x^n - 1$ where n is a natural number, which used Theorem 3.5 is shown in the following pseudo code.

Algorithm 6 Iterative algorithm for modular inversion of f modulo $x^n - 1$

Input: $n \in \mathbb{N}$ with the form of $n = 2^r d$ for some $r, d \in \mathbb{N}$, and $f \in \mathbb{F}_q[x]$ with $f(1) \neq 0$.

Output: $g \in \mathbb{F}_q[x]$ satisfying $fg \equiv 1 \pmod{x^n - 1}$.

```
Compute inverse of f modulo x^d − 1 by using Half-GCD algorithm.
Set initial g_0 be an inverse of f modulo x^d − 1
for i = 1, 2, 3 ..., r do
    g_i ← f g_{i−1}^2 rem (x^{2^i d} − 1)
return g_r
```


3.2 On Cost Analysis

Referring to the cost analysis of Cao and Cao [3], the definition of multiplication time and its properties are required to analyze the multiplication time of Algorithm 5 and Algorithm 6.

Definition 3.6. Let $\mathbb{F}_q$ be a finite field of characteristic p. A function $\mathrm{M} : \mathbb{N} \to \mathbb{R}^+$ is called a multiplication time for $\mathbb{F}_q$ if polynomials in $\mathbb{F}_q[x]$ of degree less than n can be multiplied by using at most $M(n)$ operations in $\mathbb{F}_q$.

Throughout this section, the function M is defined by the above definition, $\mathbb{F}_q$ is a finite field of characteristic p and we determine the multiplicative time $M(n)$ using Fast Fourier Transform (FFT) as $\mathcal{O}(n \log n)$, where n is the degree of polynomial. [11] To find the multiplication time of the above algorithms, the following sufficient properties are needed.

$$M(mn) \geq mM(n), \quad M(m+n) \geq M(n) + M(m), \quad \text{and} \quad M(n) \geq n,$$

for all $n, m \in \mathbb{N}$.

Lemma 3.7. *Let $\mathbb{F}_q$ be a finite field of characteristic p. Let $\{g_i\}_{i \geq 0}$ be a sequence of polynomials over $\mathbb{F}_q[x]$ with $g_0 = 1$ and f be a polynomial over $\mathbb{F}_q[x]$ with $f(1) = 1$. If $fg_i \equiv 1 \pmod{x^{p^i} - 1}$ for all $i \geq 0$, then the sequence $\{g_i\}$ satisfies the iterative congruent relation*

$$g_{i+1} \equiv g_i \pmod{x^{p^i} - 1} \quad \text{for all } i \geq 0. \tag{3.3}$$

Proof. Assume that $fg_i \equiv 1 \pmod{x^{p^i} - 1}$ for all $i \geq 0$, then, $fg_{i+1} \equiv 1 \pmod{x^{p^{i+1}} - 1}$. We can reduce to $fg_{i+1} \equiv 1 \pmod{x^{p^i} - 1}$. Since $\gcd(f, x^{p^i} - 1) = 1$, then we have $g_{i+1} \equiv g_i \pmod{x^{p^i} - 1}$ for all $i \geq 0$. □

Theorem 3.8. *Algorithm 5 correctly computes the inverse of f modulo $x^{p^r} - 1$ which uses $\mathcal{O}(M(p^r)) = \mathcal{O}(n \log n)$ multiplicative operations in $\mathbb{F}_q$ with characteristic p.*

Proof. In step 2 of Algorithm 5. For the ith step,

$$g_i \equiv f^{p-1} g_{i-1}^p \pmod{x^{p^i} - 1}. \tag{3.4}$$

The cost for one iteration of the ith step is

- $\lceil \log_2 p \rceil M(p^i)$ for the computation of g_{i-1}^p
- $\lceil \log_2 (p-1) \rceil M(p^i)$ for the computation of f^{p-1}

- $M(p^i)$ for the product $f^{p-1}g_{i-1}^p$ modulo $(x^{p^i}-1)$

Thus, we have $(\lceil \log_2 p\rceil + \lceil \log_2(p-1)\rceil + 1)M(p^i)$ operations in step 2 of Algorithm 5. So the total running time for this algorithm is

$$\begin{aligned}
&\sum_{i=1}^{r}\left(\lceil \log_2 p\rceil + \lceil \log_2(p-1)\rceil + 1\right) M(p^i)\\
&< \sum_{i=1}^{r}\left(\log_2 p + \log_2(p-1) + 3\right) M(p^i)\\
&= \sum_{i=1}^{r}\big(\log_2(8p^2-8p)\big) M(p^i)\\
&\leq \left(\sum_{i=1}^{r} p^{i-r}\right)\big(\log_2(8p^2-8p)M(p^r)\big)\\
&< \left(\left(\frac{p}{p-1}\right)\log_2(8p^2-8p)\right) M(p^r)
\end{aligned}$$

Since p is constant, then $\left(\frac{p}{p-1}\right)\log_2(8p^2-8p)$ is a constant too. So,

$$\begin{aligned}
\left(\left(\frac{p}{p-1}\right)\log_2(8p^2-8p)\right) M(p^r) &\in \mathcal{O}(M(p^r))\\
&= \mathcal{O}(M(n))\\
&= \mathcal{O}(n\log n)
\end{aligned}$$

which for the practical purpose is the same as $\mathcal{O}(n\log n)$, where $n = p^r$ is the degree of polynomials as required. □

Theorem 3.9. *The computational complexity for computing the inverse of f modulo $x^{p^r}+1$ is $\mathcal{O}(n\log n)$ where $n = p^r$ is a degree of the polynomial.*

Proof. In the same way of Theorem 3.8, we can complete the proof. □

Consider in Theorem 3.5 with characteristic 2, we obtain the complexity of this lemma using the relation of g_i as follow.

Lemma 3.10. *Let $\mathbb{F}_q$ be a finite field of characteristic 2. Let $\{g_i\}_{i\geq 0}$ be a sequence of polynomials over $\mathbb{F}_q[x]$, f be a polynomial over $\mathbb{F}_q[x]$ with $f(1) = 1$ satisfying $fg_0 \equiv 1 \pmod{x^d - 1}$, where d is a natural number. If $fg_i \equiv 1 \pmod{x^{2^i d} - 1}$ for all $i \geq 0$, then the sequence $\{g_i\}$ satisfies the iterative congruent relation*

$$g_{i+1} \equiv g_i \pmod{x^{2^i d} - 1} \quad \text{for all } i \geq 0. \tag{3.5}$$

Proof. Assume that $fg_i \equiv 1 \pmod{x^{2^i d} - 1}$ for all $i \geq 0$. Then $fg_{i+1} \equiv 1 \pmod{x^{2^{i+1} d} - 1}$. So $fg_{i+1} \equiv 1 \pmod{x^{2^i d} - 1}$. Since $\gcd(f, x^{2^i d} - 1) = 1$, then we have $g_{i+1} \equiv g_i \pmod{x^{2^i d} - 1}$ for all $i \geq 0$. □

Theorem 3.11. *Theorem 3.5 yields the complexity to compute an inverse of f modulo $x^{2^r d} - 1$ for some $r, d \in \mathbb{N}$, which uses at most $\mathcal{O}(d \log^2 d) + \mathcal{O}(n \log n)$ multiplicative operation in $\mathbb{F}_q$ where $n = 2^r d$ for some $r, d \in \mathbb{N}$.*

Proof. We first compute the complexity of computing inverse of f modulo $x^d - 1$ using Half-GCD algorithm, which is $\mathcal{O}(d \log^2 d)$. For the ith step of step 3 of Algorithm 6,

$$g_i \equiv fg_{i-1}^2 \pmod{x^{2^i d} - 1}. \tag{3.6}$$

The cost for one iteration of the ith step is

- $M(2^i d)$ for the computation of g_{i-1}^2
- $M(2^i d)$ for the product of fg_{i-1}^2 modulo $(x^{2^i d} - 1)$

Thus, we have $M(2^i d) + M(2^i d) = 2M(2^i d)$ operations in step 3 of Algorithm 6. So the total running time for this algorithm is

$$\begin{aligned}
\mathcal{O}(d \log^2 d) + \sum_{i=1}^{r} \left(2M(2^i d)\right) &\leq \mathcal{O}(d \log^2 d) + \left(\sum_{i=1}^{r} 2^{i-r}\right) \left(2M(2^i d)\right) \\
&< \mathcal{O}(d \log^2 d) + 4M(2^r d) \\
&\in \mathcal{O}(d \log^2 d) + \mathcal{O}\left(M(2^r d)\right)
\end{aligned}$$

which the same order as $\mathcal{O}(d\log^2 d) + \mathcal{O}(n\log n)$, where $n = 2^r d$ for some $r, d \in \mathbb{N}$. We can classify the computational complexity of the above algorithm,

$$\mathcal{O}(d\log^2 d) + \mathcal{O}(n\log n)$$

depending on the cases of d as follows.

- If d is constant, then the term of $\mathcal{O}(d\log^2 d)$ is constant too and it will be absorbed to the order of $\mathcal{O}(n\log n)$. So the computational complexity of this algorithm is $\mathcal{O}(n\log n)$.
- If d is linear, the order of d is the same as n, then the term of $\mathcal{O}(n\log n)$ will be absorbed to the first term, $\mathcal{O}(d\log^2 d) = \mathcal{O}(n\log^2 n)$. So the computational complexity of this algorithm is $\mathcal{O}(n\log^2 n)$.
- If d is the form of $n^{1-\epsilon}$ for some $\epsilon > 0$, then we can write the term of $\mathcal{O}(d\log^2 d)$ as $\mathcal{O}(n^{1-\epsilon}\log^2(n^{1-\epsilon})) = \mathcal{O}(n^{1-\epsilon}(1-\epsilon)^2\log^2(n))$. Since $(1-\epsilon)^2$ is a constant term, so $\mathcal{O}(n^{1-\epsilon}\log^2(n)) = \mathcal{O}(\frac{\log n}{n^{\epsilon}}(n\log n))$ which has the growth rate slower than $\mathcal{O}(n\log n))$ as $n \to \infty$. Hence the term of $\mathcal{O}(d\log^2 d)$ will be absorbed to the $\mathcal{O}(n\log n)$. So the computational complexity of this algorithm is $\mathcal{O}(n\log n)$.

□

3.3 Experiments and results

This table shows the example of running times for computing the polynomial modular inversion modulo $x^n - 1$ where $n = 2^r d$ for some $r, d \in \mathbb{N}$, comparing with the Half-GCD algorithm.

(n , r , d)	Running Times (Seconds)	
	Half-GCD algorithm	Algorithm 6
(12 , 2 , 3)	0.3124387	0.3112994
(96 , 5 , 3)	0.3384296	0.3414478
(768 , 8 , 3)	0.3677804	0.3419618
(3072 , 10 , 3)	0.4399936	0.3586432

As the above table, we use the Sagemath program for computing the results, set $\mathbb{F}_2$ be the field and then we compute by using $f = x^7 + x^3 + 1 \in \mathbb{F}_2[x]$ as a fixed input.

We can deduce that at the large degree n of the polynomials, Algorithm 6 can enhance the less times for computing the polynomial modular inversion modulo $x^n - 1$ where $n = 2^r d$ for some $r, d \in \mathbb{N}$, which has the more efficient algorithm than the original Half-GCD algorithm.

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

4.1 Conclusions

In this work, studies a modular inversion problem for a polynomial in the ring of polynomial under particular modulo. Rigorously, let $\mathbb{F}_q$ be a field with characteristic p, given $f \in \mathbb{F}_q[x]$, this thesis presents the iterative algorithm, as shown in Theorem 3.1, 3.3, 3.4 and 3.5 to find its modular inverse $g \in \mathbb{F}_q[x]$, where $fg \equiv 1 \pmod{h_i}$ for $i = 1, 2, 3, 4$, where $h_1 = x^{p^r} - 1$, $h_2 = x^{p^r} + 1$, $h_3 = x^{2p^r} - 1$ and for characteristic 2, $h_4 = x^n - 1$ where $n = 2^r d$ for some $r, d \in \mathbb{N}$. In addition, the cost analysis in term of computational complexity of the algorithm for these modulus is in the same order as that of Cao and Cao, $\mathcal{O}(n \log n)$. This indicates that the algorithm is computationally cheaper than the Half-GCD algorithm.

4.2 Future work

We know that every natural number $n \geq 0$ can be written as $v+1$ or $2^r d$ where v is an even number and $d, r \in \mathbb{N}$. So we can find the polynomial modular inversion modulo $x^n - 1$ by classifying and repeating between these cases: polynomial modular inversion modulo $x^{v+1} - 1$ and $x^{2^r d} - 1$ with the assumptions of existing of polynomial modular inversion modulo $x^v - 1$ and $x^d - 1$ respectively.

However, we focus on only the case of $n \in \mathbb{N}$ with the form of $n = 2^r d$ for some $r, d \in \mathbb{N}$. We used the Half-GCD algorithm for computing the first step (inverse of f modulo $x^d - 1$) and continue with the problem of the polynomial modular inversion modulo $x^{2^r d} - 1$. So, the case of finding algorithm for the polynomial modular inversion modulo $x^{v+1} - 1$ with the assumption of existing of the polynomial modular inversion modulo $x^v - 1$ instead of recalling the Half-GCD algorithm, we will left as a future work.

Moreover, we provide some possible future works related to this thesis. Similar to this work, the idea can be extended to obtain an algorithm for computing the modular inverse of a polynomial in a ring of polynomials over a finite field $\mathbb{F}_q$ with a characteristic p under modulo $x^\ell - 1$ and $x^\ell + 1$, where $\ell \in \mathbb{N}$.

REFERENCES

[1] K. Kobayashi, N. Takagi, and K. Takagi, "An algorithm for inversion in gf (2^ m) suitable for implementation using a polynomial multiply instruction on gf (2)," in *18th IEEE Symposium on Computer Arithmetic (ARITH'07)*, pp. 105–112, IEEE, 2007.

[2] R. T. Moenck, "Fast computation of gcds," in *Proceedings of the fifth annual ACM symposium on Theory of computing*, pp. 142–151, 1973.

[3] Z. Cao and H. Cao, "On fast division algorithm for polynomials using Newton iteration," in *International Conference on Information Computing and Applications*, pp. 175–180, Springer, 2012.

[4] L. W. Ehrlich, "A modified newton method for polynomials," *Communications of the ACM*, vol. 10, no. 2, pp. 107–108, 1967.

[5] V. Shoup, *A computational introduction to number theory and algebra.* Cambridge university press, 2009.

[6] Z.-X. Wan, *Lectures on finite fields and Galois rings.* World Scientific Publishing Company, 2003.

[7] D. S. Dummit and R. M. Foote, *Abstract algebra*, vol. 1999. Prentice Hall Englewood Cliffs, NJ, 1991.

[8] R. Lidl and H. Niederreiter, *Introduction to finite fields and their applications.* Cambridge university press, 1994.

[9] T. E. Stenhammar, "Factorization of binary polynomials," Master's thesis, Tampere University of Technology, 2018.

[10] S. Lang, *Algebra*, vol. 211. Springer Science & Business Media, 2012.

[11] H. J. Nussbaumer, "The fast fourier transform," in *Fast Fourier Transform and Convolution Algorithms*, pp. 80–111, Springer, 1981.

BIOGRAPHY

Name	Mr. Samakorn Sripatthanakul
Date of Birth	23 January 1997
Place of Birth	Prachinburi, Thailand
Educations	B.Sc. (Mathematics) (Second Class Honours), Chulalongkorn University, 2015–2018 M.Sc. (Applied Mathematics and Computational Science), Chulalongkorn University, 2019–Present
Scholarships	Development and Promotion of Science and Technology Talents Project (DPST), Institute of the Promotion of Teaching Science and Technology (IPST)

Publications

- **Samakorn Sripatthanakul** and Wutichai Chongchitmate, Iterative algorithm for polynomial modular inversion modulo $x^{p^r} - 1$ over finite field of order p, *Proceeding of Annual Meeting in Mathematics (AMM) 2022*, pp.21. 2022.