Songklanakarin J. Sci. Technol.
40 (6), 1361-1367, Nov. – Dec. 2018

http://www.sjst.psu.ac.th

Original Article

Numerical methods for finding multiplicative inverses of a modulo N

Bencharat Prempreesuk, Prapanpong Pongsriiam, and Nairat Kanyamee*

Department of Mathematics, Faculty of Science,
Silpakorn University, Sanam Chandra Palace, Mueang, Nakhon Pathom, 73000 Thailand

Received: 2 May 2017; Revised: 14 July 2017; Accepted: 22 August 2017

Abstract

In this work, we propose new methods based on the root-finding methods in numerical analysis to calculate the inverse of an integer a modulo N for any positive integer N. We apply Newton's and secant methods with a power-reduction process and Newton's and secant methods together with the binary representation and Zeckendorf representation to determine the inverse of an integer a modulo N. The numerical results confirm an accuracy when compared to analytical results. In addition, the two methods with the binary and Zeckendorf representations give better CPU times than Newton's and secant methods with power-reduction, for large N.

Keywords: modular multiplicative inverse, Newton's method, secant method, binary representation, Zeckendorf representation

1. Introduction

Modular multiplicative inverse is often seen in number theory and it is desirable to be able to calculate it quickly. This motivates us to search for an efficient method for finding the modular multiplicative inverse.

On the other hand, numerical analysis provides algorithms for approximate solutions to particular problem. So, there should be some connections between number theory and numerical analysis fields.

Let $N \in \mathbb{Z}^+$ and $a, b \in \mathbb{Z}$. We write $a \equiv b \pmod N$ if $N|a-b$. Assume further that $(a, N) = 1$. Then it is a well-known result (Rosen, 2005) in number theory that there exists an integer x such that

$$ax \equiv 1 \pmod N.$$

The integer x is called the modular multiplicative inverse (or simply the inverse) of a modulo N and it is denoted by a^{-1} or $\frac{1}{a}$. For example, $2(3) \equiv 1 \pmod 5$, so the inverse of 2 modulo 5 is 3, that is, $3 \equiv 2^{-1} \pmod 5$. By the fundamental theorem of arithmetic (Rosen, 2005), we can write

$$N = {p_1}^{n_1}{p_2}^{n_2} \dots {p_k}^{n_k},$$

where $p_1, p_2, \dots, p_k$ are distinct primes and $n_1, n_2, \dots, n_k$ are positive integers. We can first calculate the inverse of a modulo ${p_1}^{n_1}, {p_2}^{n_2}, \dots, {p_k}^{n_k}$, respectively, and then apply the Chinese remainder theorem (Rosen, 2005) to obtain the inverse of a modulo N.

Apart from the Euclidean algorithm to calculate the inverse, some researchers have introduced an interesting application of numerical analysis to number theory. In numerical analysis, we have the classical root-finding methods such as Newton's method and secant method to find a zero of a function f or to find a solution of $f(x) = 0$. The concept of iterative root-finding methods is to construct a sequence $x_0, x_1, x_2, \dots$ that will converge to a zero of $f(x)$ in $[a, b]$. In 2010, Knapp and Xenophontos calculated the inverses of an integer a modulo p^n using root-finding methods, such as Newton's method, secant method, fixed-point iteration and a higher-order convergent method (Knapp & Xenophontos, 2010). Newton's method is suitable for calculating the inverse of an integer a modulo p^n when $n = m \cdot 2^i$ for $i = 1, 2, 3, \dots$ and some positive integer m. The secant method is used to find the inverse of a modulo $p^{\alpha+\beta}$, when the inverses

*Corresponding author
Email address: kanyamee_n@su.ac.th

of $a \bmod p^{\alpha}$ and $a \bmod p^{\beta}$ are already known. The higher-order convergent method of order r is suitable for calculating an inverse of a modulo p^n when $n = m \cdot r^i$ for $i = 1, 2, 3, \ldots$ and some positive integer m. In 2014, Dumas studied fast algorithms for the computation of modular inverses by using Newton-Raphson iteration (Dumas, 2014). He designed four algorithms with a few operations used. He claimed that his hybrid algorithm is 26% faster than any direct method.

In this paper, we extend the work of Knapp *et al*. by using Newton's and secant methods with a power-reduction calculation and Newton's and secant methods together with the binary representation and the Zeckendorf representation to determine the inverse of an integer a modulo N for any positive integer N, by splitting N into its prime factor form and calculating the inverse in terms of a modulo p^n. To do this, we consider two representations for n. In the first case, we represent n as the sum of distinct powers of 2. In the second case, we use the Zeckendorf representation for it (Griffiths, 2015). Then we use root-finding methods to calculate inverses modulo p^n. Furthermore, we designed the algorithms in MATLAB to calculate the inverse of a modulo p^n, and compared the CPU times of each method.

2. Root-finding methods for the inverse of a modulo p^n

In this section, we establish root-finding methods with a power-reduction calculation to calculate the inverse of an integer a modulo p^n. The methods are based on the root-finding methods in Knapp *et al*. (Knapp & Xenophontos, 2010). We are looking for an integer x for which $ax \equiv 1 \pmod{p^n}$ or $x \equiv \frac{1}{a} \pmod{p^n}$. Following the methods in Knapp *et al*., a suitable choice for the function f is $f(x) = \frac{1}{x} - a$. The goal is to solve the equation $f(x) = 0$.

2.1 Newton's method with a power-reduction calculation

The well-known Newton's method is one of the most powerful numerical methods for root-finding problems. We combine this method with a process of modular calculations to reduce the power of p, in order to find the inverse mod p^n.

Newton's method relies on the continuity of $f'(x)$ and $f''(x)$. According to (Burden & Faires, 1997), the iteration for Newton's method to locate a zero of f is given by

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad , \quad i = 1, 2, 3, \ldots$$

with a suitable initial value x_1. This iteration generates a sequence $\{x_i\}$ which converges to a zero of f. To apply the method, we first find $f'(x) = -\frac{1}{x^2}$. The iteration for Newton's method becomes

$$x_{i+1} = x_i - \frac{\frac{1}{x_i} - a}{-\frac{1}{x_i^2}}$$

$$x_{i+1} = x_i(2 - ax_i), \; i = 1, 2, 3, \ldots \tag{2.1}$$

Here, we obtain the sequence $\{x_i\}$ which converges to a solution of $f(x) = 0$.

Theorem 2.1.1. Let $\alpha > 0$ and suppose that x_i is an inverse of a modulo p^{α}. Then x_{i+1} given by (2.1) is an inverse of a modulo $p^{2\alpha}$.

Proof. See Knapp and Xenophontos (2010).

Theorem 2.1.1 implies that if we know the inverse of a modulo p^{α}, then we can use Newton's method to calculate the inverse of a modulo $p^{2\alpha}$, a modulo $p^{4\alpha}$, a modulo $p^{8\alpha}$ and so on. For simplicity, we choose the initial guess x_1 to be the inverse of a modulo p and apply Newton's method in (2.1) to determine the inverse of a modulo p^2, a modulo p^4, a modulo p^8 and so on. As a result, this method is suitable for calculating the inverse of an integer a modulo p^n when $n = 2^k$, $k = 1, 2, 3, \ldots$. It is known that the method converges at a quadratic rate for a simple root (Burden & Faires, 1997).

In addition to Newton's method in Knapp *et al*., we construct an algorithm to find the inverse of an integer a modulo p^n for any integer n, not necessarily of the form p^{2^k}. We first apply Newton's method in (2.1) to calculate the inverse x of an integer a modulo p^{2^k} where $2^{k-1} < n < 2^k$ for some positive integer k. Then, we begin to reduce the power of p from 2^k down to n. Since $\equiv 1 \pmod{p^{2^k}}$, it follows that $ax \equiv 1 \pmod{p^n}$. The reduction process applies this idea together with the congruence property to search for the smallest positive number in the same congruent class as x modulo p^n (shown in line 20 (referring to line 14) in Algorithm 1). By continuing this process, we finally arrive with the inverse of an integer a modulo p^n. The algorithm (Algorithm 1) to determine such an inverse is shown in Table 1.

Example 1: Let $p = 3, a = 2$ and $n = 6$. We want to calculate the inverse of 2 modulo 3^6. We choose the initial value $x_1 = 2$ since $2 \cdot 2 \equiv 1 \pmod 3$. Observe that $4 < 6 < 8$. We apply Newton's method to find the inverse of 2 modulo 3^2 and so on until 3^8 as follows:

$$x_2 \equiv 5 \pmod{3^2}$$
$$x_3 \equiv 41 \pmod{3^4}$$
$$x_4 \equiv 3281 \pmod{3^8}.$$

After we obtain x_4, we use the power-reduction calculation and finally get the inverse of 2 modulo 3^6 as 365.

2.2 The secant method with a power-reduction calculation

Secant method is designed to converge almost as fast as Newton's method, but involves only $f(x)$ not $f'(x)$ (Atkinson, 1988). In a similar way to Newton's method, we also combine this method with a modular calculation to reduce the power of p. The iteration for secant method is given by

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} \quad , \quad i = 2, 3, 4, \ldots$$

Table 1. Algorithm for Newton's method with a power-reduction calculation

Algorithm 1: Newton's method with a power-reduction calculation

Input: $a, p, n \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value x_1 (inverse of a mod p);
for $k = 1$ **to** M
  compute : $x_{k+1} = x_k(2 - ax_k)$; $r = 2^k$;
  if $x_{k+1} < 0$
  set $m = 1; z_1 = x_{k+1} + p^r$;
    while $z_m < 0$
      $m = m + 1; z_m = x_{k+1} + (m \cdot p^r)$;
    end ; display z_{end}
  else if $x_{k+1} > 0$ and $x_{k+1} < p^r$
    display x_{k+1} ; stop the process
  else if $x_{k+1} > 0$ and $x_{k+1} > p^r$
    set $m = 1; z_1 = x_{k+1} - p^r$;
    while $z_m > p^r$
      $m = m + 1; z_m = x_{k+1} - (m \cdot p^r)$;
    end ; display z_{end}
  end
  if $r = n$
    stop the process
  else if $r > n$
    repeat line 4-16 by replacing r with n
  end
end

The method requires two initial values x_1 and x_2 to generate a sequence $\{x_i\}$ which converges to a zero of f. By setting $f(x) = \frac{1}{x} - a$, the iteration for secant method is given by

$$x_{i+1} = x_i - \frac{\left(\frac{1}{x_i} - a\right)(x_i - x_{i-1})}{\left(\frac{1}{x_i} - a\right) - \left(\frac{1}{x_{i-1}} - a\right)}$$

$$x_{i+1} = x_i + x_{i-1} - ax_ix_{i-1}, \qquad i = 2, 3, 4, \ldots \tag{2.2}$$

Equation (2.2) provides a sequence $\{x_i\}$ which converges to a root of $f(x) = 0$.

Theorem 2.2.1. Suppose that $x_{i-1} \equiv \frac{1}{a} \pmod{p^\alpha}$ and that $x_i \equiv \frac{1}{a} \pmod{p^\beta}$. Then, for x_{i+1} given by (2.2) we have $x_{i+1} \equiv \frac{1}{a} \pmod{p^{\alpha+\beta}}$.

Proof. See Knapp and Xenophontos (2010).

Based on this theorem, we can use secant method to find the inverse of a modulo $p^{\alpha+\beta}$ if we know the inverse of a modulo p^α and a modulo p^β. We first choose x_1 and x_2 such that $x_1 \equiv \frac{1}{a} \pmod{p^\alpha}$, and $x_2 \equiv \frac{1}{a} \pmod{p^\beta}$. Then, we iterate x_3 from secant method, which gives the inverse of a modulo $p^{\alpha+\beta}$, and continue the process. Since it is simpler to calculate the inverse of a modulo p, both initial values x_1 and x_2 for the secant method are chosen to be the inverse of a modulo p. Then, we apply the secant method to calculate x_3 that is the inverse of a modulo p^2. By continuing the process, we obtain the inverse of a modulo p^3, a modulo p^5, a modulo p^8 and so on. In fact, the iterated values, x_i, are the inverses in which the powers of p that are the Fibonacci numbers. This method is known to provide convergence of order $\frac{1+\sqrt{5}}{2} \approx 1.618$ (Atkinson, 1988).

In addition to the secant method in Knapp *et al.*, we construct an algorithm to find the inverse of an integer a modulo p^n for any positive integer n, other than the form of $p^{\alpha+\beta}$ or p^{F_k} where F_k is a Fibonacci number. Recall that the Fibonacci sequence $(F_k)_{k\geq 1}$ is defined by the recurrence relation $F_k = F_{k-1} + F_{k-2}$ for $k \geq 3$ and the initial values $F_1 = F_2 = 1$. We first apply the secant method in (2.2) to calculate the inverse of an integer a modulo p^{F_k} where $F_{k-1} < n < F_k$ for some positive integer k. Then, we use the reduction process (mentioned in section 2.1) to reduce the power of p from F_k down to n in order to obtain the inverse of integer a modulo p^n. The algorithm (Algorithm 2) to determine such an inverse is shown in Table 2.

Table 2. Algorithm for the secant method with a power-reduction calculation

Algorithm 2: The secant method with a power-reduction calculation

Input: $a, p, n \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value x_1, x_2 (inverse of a mod p);
$r_1 = 1; r_2 = 1$;
for $k = 2$ **to** M
  compute : $x_{k+1} = x_k + x_{k-1} - ax_kx_{k-1}$; $r_{k+1} = r_{k-1} + r_k$; $r = r_{k+1}$;
  if $x_{k+1} < 0$
  set $m = 1; z_1 = x_{k+1} + p^r$;
    while $z_m < 0$
      $m = m + 1; z_m = x_{k+1} + (m \cdot p^r)$;
    end ; display z_{end}
  else if $x_{k+1} > 0$ and $x_{k+1} < p^r$
    display x_{k+1} ; stop the process
  else if $x_{k+1} > 0$ and $x_{k+1} > p^r$
    set $m = 1; z_1 = x_{k+1} - p^r$;
    while $z_m > p^r$
      $m = m + 1; z_m = x_{k+1} - (m \cdot p^r)$;
    end ; display z_{end}
  end
  if $r = n$
    stop the process
  else if $r > n$
    repeat line 5-17 by replacing r with n
  end
end

Example 2: Let $p = 3, a = 2$ and $n = 11$. We want to calculate the inverse of 2 modulo 3^{11}. We choose the initial values $x_1 = 2, x_2 = 2$ because $2 \cdot 2 \equiv 1 \pmod 3$. Notice that $8 < 11 < 13$. We apply the secant method to find the inverse of 2 modulo 3^2, 2 modulo 3^3 and so on until 2 modulo 3^{13} as follows:

$$x_3 \equiv 5 \pmod{3^2}$$
$$x_4 \equiv 14 \pmod{3^3}$$
$$x_5 \equiv 122 \pmod{3^5}$$
$$x_6 \equiv 3281 \pmod{3^8}$$
$$x_7 \equiv 797162 \pmod{3^{13}}.$$

Once we obtain x_7, we use the power-reduction calculation and finally get the inverse of 2 modulo 3^{11} as 88574.

3. Calculating the Inverse of a Mod N and a Mod p^n

In this section, we use the methods in section 2.1 and section 2.2 to calculate the inverse of a modulo p^n for any positive integer n. We consider n in two cases. For the first case, we represent n as the sum of distinct powers of 2. For the second case, we represent n in the form of Zeckendorf representation.

3.1 Using binary representation

Since we know that every positive integer can be represented by the sum of distinct powers of 2 (Rosen, 2005), we first express n in the binary system. If n is a positive integer, the binary expansion of n is given by (Mathew, 1992)

$$n = \left(b_j \times 2^j\right) + \left(b_{j-1} \times 2^{j-1}\right) + \cdots + (b_1 \times 2^1) + (b_0 \times 2^0),$$

where each digit b_j is either a 0 or a 1. This can also be written in the notation

$$n = b_j b_{j-1} \ldots b_2 b_1 b_{0\,\text{two}}.$$

Therefore

$$p^n = p^{b_j 2^j + b_{j-1} 2^{j-1} + \cdots + 2b_1 + b_0} \tag{3.1}$$

or

$$p^n = p^{b_j 2^j} \cdot p^{b_{j-1} 2^{j-1}} \ldots p^{2b_1} \cdot p^{b_0}. \tag{3.2}$$

The form in (3.2) consists of a prime number to powers 2^k. We can apply Newton's method in Algorithm 1 to calculate the inverse of an integer a modulo p^{2^k} for each k in which b_k in (3.1) is not zero. The secant method in Theorem 2.2.1 and Algorithm 2 are then applied to determine the inverse of each pair of a modulo p^{2^k} until the largest power term. The algorithm (Algorithm 3) to calculate the inverse using the binary representation is presented in Table 3.

Example 3: Let $p = 2, a = 3$ and $n = 10$. We want to calculate the inverse of 3 modulo 2^{10}. We choose the initial value $x_1 = 1$, since $3 \cdot 1 \equiv 1 \pmod 2$. The binary representation of n in this problem is $n = 10 = 2^3 + 2 = 1010_{two}$. Observe that the representation contains the terms 2^3 and 2. We apply Newton's method and Theorem 2.1.1. to find the inverses of 3 modulo 2^{2^i}, $i \leq 3$ as follows:

$x_2 \equiv 3 \pmod{2^2}$ or $x_2 \equiv 3 \pmod{2^{2^1}}$

$x_3 \equiv 11 \pmod{2^4}$ or $x_3 \equiv 11 \pmod{2^{2^2}}$

$x_4 \equiv 171 \pmod{2^8}$ or $x_4 \equiv 171 \pmod{2^{2^3}}$.

Table 3. Algorithm for calculating the inverse using binary representation

Algorithm 3: Inverse of a mod p^n using binary representation

Input: $a, p, n \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value x_1 (inverse of a mod p);
A = vector representing n in base two ; NA =size of A ;
IdA = vector containing indices of nonzero elements in A ; NId =size of IdA ;
Apply **Algorithm 1** from line 1-16
  Input: $a, p, NA \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value x_1 (inverse of a mod p)
  Output: X
if $NId = 1$
  display $X(end)$; stop the process
else for $i = 1$ **to** NId
    $newX_i = X(IdA_i)$;
  end
  $P_1 = 2^{(IdA_1 - 1)} + 2^{(IdA_2 - 1)}$;
  Apply **Algorithm 2** from line 1-17
    Input: $a, p, P_1 \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value $newX_1, newX_2$;
    Output: S_1
  for $j = 2$ **to** $NId - 1$
    $P_j = P_{j-1} + 2^{(IdA_{j+1} - 1)}$;
    Apply **Algorithm 2** from line 1-17
      Input: $a, p, P_j \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value $S_{j-1}, newX_{j+1}$;
  end
  display $S(end)$; stop the process
end

Using the secant method and Theorem 2.2.1. with x_2 and x_4 as the initial conditions to calculate the inverse of 3 modulo 2^{10} yields

$$x = 683 \pmod{2^{2^3+2}}$$

Therefore, 683 is the inverse of 3 modulo 2^{10}.

3.2 Using the Zeckendorf representation

Consider the fact that any positive integer can be expressed as the sum of Fibonacci numbers (Rosen, 2005). In this section, we represent the integer n in the form of the Zeckendorf representation.

Theorem 3.2.1. (Zeckendorf's Theorem (Rosen, 2005; Griffiths, 2015)**)** Every positive integer can be represented uniquely as the sum of one or more distinct Fibonacci numbers in such a way that the sum does not include any two consecutive Fibonacci numbers. More precisely, if n is any positive integer, there exist positive integers $c_i \geq 2$, with $c_{i+1} > c_i + 1$ for every $i \in \{0,1,2, \ldots, k\}$ such that

$$n = \sum_{i=0}^{k} F_{c_i}$$

where F_k is the k^{th} Fibonacci number. Such a sum is called the Zeckendorf representation of n. In fact, the greedy algorithm always leads to Zeckendorf representation.

Proof. See Rosen (2005) and Griffiths (2015).

Example 4: Consider the number $n = 95$. There are several ways to express this number as a sum of Fibonacci numbers. Notice that $95 = 89 + 3 + 2 + 1 = F_{11} + F_4 + F_3 + F_2$ and $95 = 55 + 34 + 5 + 1 = F_{10} + F_9 + F_5 + F_2$ are not Zeckendorf representations because $3, 2, 1$ and $55, 34$ are consecutive Fibonacci numbers. However, $95 = 89 + 5 + 1 = F_{11} + F_5 + F_2$ is a Zeckendorf representation which is uniquely determined.

From the greedy algorithm and Zeckendorf's theorem, we can construct an algorithm to write n in form of the Zeckendorf representation. We first construct a Fibonacci sequence starting from 1 and contains numbers closest to n. The binary search algorithm is applied to find the largest Fibonacci numbers, F_{n_1}, that is less than or equal to n. Then repeat the process to find the largest Fibonacci numbers, F_{n_2}, that is less than or equal to $n - F_{n_1}$. By continuing this process, we arrive with the Zeckendorf representation $n = F_{n_1} + F_{n_2} + \cdots + F_{n_k}$ for some $k \in \mathbb{N}$. Algorithm 4 in Table 4 provides the process for finding such a representation.

Table 4. Algorithm for representing n with the Zeckendorf representation

Algorithm 4: Representing n with the Zeckendorf representation

Input: $n \in \mathbb{N}$;
set $F_1 = 1$; $F_2 = 2$; $i = 3$; $f = F_2$;
while $f \leq 2n$
    $F_i = F_{i-1} + F_{i-2}$; $f = F_i$; $i = i + 1$; (constructing Fibonacci sequence)
end
while $n \neq 0$
    Find $C = F_{n_1}$, the closet value to the number n, using **binary search algorithm**
        Input: F, n
        Output: C and I (index of value C in F)
    if $C > n$
        $x = F_{Id-1}$; $S = [S, x]$; $ID = [ID, I - 1]$;
    else
        $x = F_{Id}$; $S = [S, x]$; $ID = [ID, I]$;
    end
    $n = n - x$;
end
S = flip left/right of S ; ID = flip left/right of ID ; display S and ID
(Return S, the Zeckendorf representation of n and corresponding indices ID)

Once we have the Zeckendorf representation from Algorithm 4. The secant method in section 2.2 is applied to find the inverse of an integer a modulo p^{F_i} for each Fibonacci number F_i in Algorithm 4. Then, we apply the secant method again to find the inverse of an integer a modulo $p^{F_{n_1}+F_{n_2}+\cdots+F_{n_k}}$. The algorithm (Algorithm 5) that includes the secant method to find the inverse of an integer a modulo p^n is shown in Table 5.

Table 5. Algorithm for calculating the inverse using Zeckendorf representation

Algorithm 5: Inverse of a mod p^n using Zeckendorf representation

Input: $a, p \in \mathbb{Z}$; $n, M \in \mathbb{N}$; intitial value x_1 (inverse of a mod p)
Apply **Algorithm 4**
    Input : n
    Output : F, S, ID
Find the closet value to $S(end)$, A , and its index , IdA , using **binary search algorithm**
    Input: $F, S(end)$
    Output : A, IdA
Apply **Algorithm 2** from line 1-17, Change M in line 3 to IdA (inv of a mod $p^1, p^1, p^2, \ldots$)
    Input: $a, p, S(end) \in \mathbb{Z}$; $M, IdA \in \mathbb{N}$; intitial value x_1, x_1 ;
    Output: Y
$Y = Y_{2:end}$
if length of $S = 1$
    display $Y(end)$
else $power_1 = S_1 + S_2$;
    Apply **Algorithm 2** from line 1-17
        Input: $a, p, power_1 \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value Y_{ID_1}, Y_{ID_2} ;
        Output: sum_1
    for $j = 2$ **to** (length of $S - 1$)
        $power_j = power_{j-1} + S_{j+1}$;
        Apply **Algorithm 2** from line 1-17
            Input: $a, p, power_j \in \mathbb{Z}$; $M \in \mathbb{N}$; intitial value $sum_{j-1}, Y_{ID_{j+1}}$;
            Output: sum_j
    end
display $sum(end)$
end

Example 5: Let $p = 2, a = 3$ and $n = 12$. We want to calculate the inverse of 3 modulo 2^{12}. We choose the initial values $x_1 = 1$ and $x_2 = 1$ because $3 \cdot 1 \equiv 1 \pmod 2$. We have that $12 = 8 + 3 + 1$. Then, we use the secant method and Theorem 2.1.1. to find the inverses of 3 modulo 2^k, where k is a Fibonacci number less than or equal 12 as follows:

$$x_3 \equiv 3 \pmod{2^2}$$
$$x_4 \equiv 3 \pmod{2^3}$$
$$x_5 \equiv 11 \pmod{2^5}$$
$$x_6 \equiv 171 \pmod{2^8}.$$

Next, we apply the secant method and Theorem 2.2.1. again to find the inverse of 3 modulo 2^{1+3} and 3 modulo 2^{4+8}. This yields the inverse

$$x = 2731 \pmod{2^{8+3+1}}.$$

Therefore, 2731 is the inverse of 3 modulo 2^{12}.

In the next example, we compare the CPU times of each method used to find the inverse of a modulo p^n by using MATLAB. The calculations are carried out with an Intel Core i7 CPU@4 GHz RAM 32 GB computer. The results are presented in Example 6.

Example 6: Let $p = 3$, $a = 2$. We choose the initial values $x_1 = 2$ and $x_2 = 2$, since $2 \cdot 2 \equiv 1 \pmod 3$. The CPU times are recorded in seconds.

Table 6. CPU times in finding the inverse of a modulo p^n by the four methods.

Method	Newton	Secant	Base 2	Zeckendorf
$a \pmod{p^{32}}$	0.005446	0.008523	0.018183	0.017999
$a \pmod{p^{128}}$	0.005586	3.393821	0.018325	0.017747
$a \pmod{p^{21}}$	0.010100	0.000300	0.025102	0.013143
$a \pmod{p^{55}}$	0.009226	0.006529	0.023589	0.013082
$a \pmod{p^{233}}$	>1000	0.006682	0.023929	0.013357
$a \pmod{p^{40}}$	313.999486	1.805201	0.021543	0.017441
$a \pmod{p^{111}}$	10.129003	>1000	0.023383	0.017748
$a \pmod{p^{115}}$	0.134732	>1000	0.023891	0.017690

From Table 6, Newton's method is the fastest in finding inverses when n is a power of 2 and not too large, due to lesser number of operations than that in the other methods. In this example, we can see that when $n = 32$ or 128, Newton's method gives the smallest calculation time. If we consider the secant method, it has the best time when n is a Fibonacci number and not too large. We can see that this method gives the shortest time to find the inverses when $n = 21, 55$ and 233. The binary and Zeckendorf representation methods can calculate the inverses for any n and the CPU time increased a bit as n becomes larger. These last two methods are the best way to calculate the inverses of a modulo p^n, especially when n is not in a power of 2 or a Fibonacci number, or when n is large.

The figures below confirm the discussion above. Figure 1(a) shows the graphs of the CPU times for the four methods versus n when $n = 2,3, \ldots ,30$. The CPU time in the graph of Newton's method drops when $n = 2, 4, 8, 16, 24$ and it jumps up due to the power-reduction process. For example, the method uses comparatively little CPU time when $n = 16$, but it takes a much longer time when $n = 17$ since it first calculates the inverse of a modulo p^{24} and then reduces the power down to $n = 17$. Similarly, the CPU time in the graph for the secant method drops when $n = 2, 3, 5, 8, 13, 21$ and it jumps up in a similar manner. If we consider the graphs of CPU times for the binary and Zeckendorf representation methods, they are quite stable and do not increase much; these two methods become faster than the first two methods as n increases. Figure 1(b) illustrates the behavior of CPU times of these methods when $n = 2, 3, \ldots, 100$. We can see from the graphs that the times taken do not grow rapidly as n gets larger. These are time-efficient and very stable methods to calculate the inverse of a modulo p^n for any number n.

We end the section with an example of calculating the inverse of an integer a modulo N where $N = p_1^{n_1} p_2^{n_2} \ldots p_m^{n_m}$.

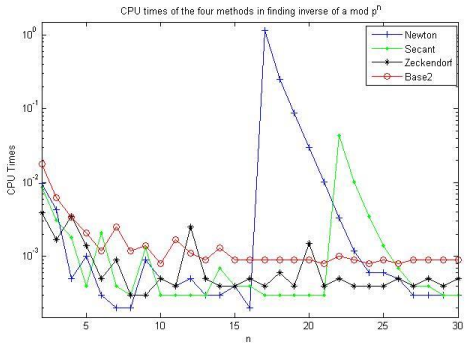
Example 7: Let $a = 2$ and $N = 33075$. The prime factorization of N is $N = 3^3 \cdot 5^2 \cdot 7^2$. We would like to find an integer x such that

$$2x \equiv 1 \pmod{3^3 \cdot 5^2 \cdot 7^2}.$$

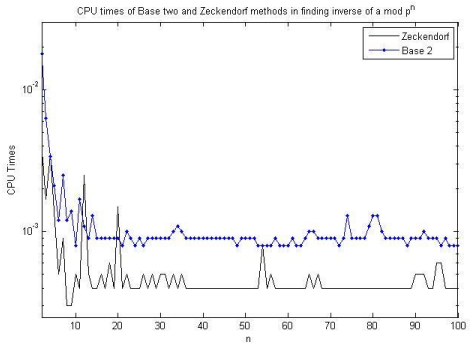
We break it down into finding

$$2x \equiv 1 \pmod{3^3},$$
$$2x \equiv 1 \pmod{5^2},$$
$$2x \equiv 1 \pmod{7^2}.$$

By using our algorithms, we can find the inverses in each case as



(a)



(b)

Figure 1. (a) The CPU times of the four methods versus n when $n = 2,3, \ldots ,30$. (b) The CPU times with binary and Zeckendorf representations versus n when $n = 2,3, \ldots ,100$.

$$x \equiv 14 \pmod{3^3},$$
$$x \equiv 13 \pmod{5^2},$$
$$x \equiv 25 \pmod{7^2}.$$

Then we combine them by using the Chinese remainder theorem.

Let $a_1 = 14, a_2 = 13, a_3 = 25, m_1 = 3^3, m_2 = 5^2, m_3 = 7^2$. For the Chinese remainder theorem (Rosen, 2005) we define

$$M_1 = \frac{M}{m_1} = \frac{3^3 \cdot 5^2 \cdot 7^2}{3^3} = 5^2 \cdot 7^2 = 1225$$
$$M_2 = \frac{M}{m_2} = \frac{3^3 \cdot 5^2 \cdot 7^2}{5^2} = 3^3 \cdot 7^2 = 1323$$
$$M_3 = \frac{M}{m_3} = \frac{3^3 \cdot 5^2 \cdot 7^2}{7^2} = 3^3 \cdot 5^2 = 675.$$

To determine the inverse of M_1, denoted by M_1', we solve $1225M_1' \equiv 1 \pmod{3^3}$, which is equivalent to $10M_1' \equiv 1 \pmod{3^3}$. So $M_1' \equiv 19 \pmod{3^3}$. Next, we find the inverse of M_2, denoted by M_2', by solving $1323M_2' \equiv 1 \pmod{5^2}$, which is equivalent to $23M_2' \equiv 1 \pmod{5^2}$. So $M_2' \equiv 12 \pmod{5^2}$. The inverse of M_3, denoted by M_3', is calculated by solving $675M_3' \equiv 1 \pmod{7^2}$, or equivalently, $38M_3' \equiv 1 \pmod{7^2}$. Then $M_3' \equiv 40 \pmod{7^2}$.

Therefore, according to the Chinese remainder theorem, the inverse of 2 modulo 33075 is

$$x \equiv a_1 M_1 M_1' + a_2 M_2 M_2' + a_3 M_3 M_3' \pmod{M}$$

$$\equiv 14(1225)(19) + 13(1323)(12) + 25(675)(40) \pmod{3^3 \cdot 5^2 \cdot 7^2}$$
$$\equiv 1207238 \pmod{3^3 \cdot 5^2 \cdot 7^2}$$
$$\equiv 16538 \pmod{3^3 \cdot 5^2 \cdot 7^2}.$$

4. Conclusions and Discussion

This work introduces new methods to calculate inverses modulo N for any positive integer N. We express N in its prime factorization as $N = p_1^{n_1} p_2^{n_2} \dots p_m^{n_m}$, where $p_1, p_2, \dots, p_m$ are distinct prime numbers and $n_1, n_2, \dots, n_m$ are positive integers. Then, we calculate the inverse by breaking it down to find the inverses of $a \bmod p_1^{n_1}$, $a \bmod p_2^{n_2}$, ..., and $a \bmod p_m^{n_m}$ and combine them using the Chinese remainder theorem. The four methods proposed to calculate the inverse of $a \bmod p^n$ are based on Newton's and secant methods.

We also construct an algorithm for each method to help calculating the inverses of a modulo p^n in MATLAB and present an example of each method. In addition, we compare the CPU times of each method in Example 6. Newton's method with power-reduction calculation is good for the case when $n = 2^i$ for $= 1, 2, 3, \dots$. The secant method with power-reduction calculation is used to find the inverse of a modulo $p^{\alpha+\beta}$ when the inverses of $a \bmod p^{\alpha}$ and $a \bmod p^{\beta}$ are already obtained. In fact, it is good for the case when n is a Fibonacci number. However, these two methods may take time when n is not in such a form because of the power-reduction process. The method that expresses n as a binary representation followed by Newton's and secant methods is suitable for any number n because every number can be represented in base 2. The method does not involve power-reduction calculation. As a result, it is time efficient and very stable for calculating the inverse for any number n, especially when n is large or is not in a power of 2 or a Fibonacci number. Similarly, the method that expresses n in Zeckendorf representation followed by the secant method is suitable for any number n because every number has a Zeckendorf representation. This method is also as good as that using the binary representation.

Acknowledgements

We would like to thank Dr. Pinyo Taeprasartsit for his helpful suggestions in programming throughout this work.

References

Atkinson, K. E. (1988). *An introduction to numerical analysis* (2nd ed.). Hoboken, NJ: John Wiley & Sons.

Burden, R. L., & Faires, J. D. (1997). *Numerical analysis*. Pacific Grove, CA: Brooks/Cole.

Dumas, J. G. (2014). On Newton-Raphson iteration for multiplicative inverses modulo prime powers. *IEEE Transations on Computers*, *63*(8), 2016-2109. doi:10.1109/TC.2013.94

Griffiths, M. (2015). The Zeckendorf representation of a Beatty-Related Finonacci sum. *Fibonacci Quart, 53*(3), 230-236. MSC2010: 11B39, 11B83.

Knapp, M. P., & Xenophontos, C. (2010). Numerical analysis meets number theory: Using rootfinding methods to calculate inverses mod p^n. *Applicable Analysis Discrete Mathematics, 4*(1), 23-31. doi:10.2298/AADM100201012K

Mathew, J. H. (1992). *Numerical methods for mathematics, science, and engineering*. Upper Saddle River, NJ: Prentice-Hall.

Rosen, K. H. (2005). *Elementary number theory and its application*. Boston, MA: Addison-Wesley.