Optimal Minimax Polynomial Approximation of Modular Reduction for Bootstrapping of Approximate Homomorphic Encryption*

Joon-Woo Lee[1], Eunsang Lee[1], Yongwoo Lee[1], Young-Sik Kim[2], and Jong-Seon No[1]

[1] Seoul National University, Republic of Korea
[2] Chosun University, Republic of Korea

Abstract. Approximate homomorphic encryption, called Cheon-Kim-Kim-Song (CKKS) scheme [6], is a fully homomorphic encryption scheme that supports arithmetic operations for real or complex number data encrypted. Since the bootstrapping of the CKKS scheme is a big bottleneck for practical use, this makes many cryptographers optimize its bootstrapping, but they cannot obtain its optimal solution. One of the core procedures in the bootstrapping is to approximate the modular reduction function, and several works have been done for the polynomial approximation of this function in the minimax aspect [2, 4, 13].

In this paper, we obtain a fast algorithm to derive the optimal minimax generalized approximate polynomial of any continuous functions over any union of the finite number of intervals, which uses a variant of the Remez algorithm, called modified Remez algorithm. Using that results, we derive the optimal minimax approximate polynomial for the modular reduction function rather than the sine or cosine function in the CKKS scheme. From the numerical analysis, there is some gap of the approximation error by two in the logarithm scale between the cosine minimax approximation and the proposed direct minimax approximation. There is some inherent flat error region for the cosine minimax approximation such that the minimax approximation error does not decrease as the degree of the approximation polynomial increases, but the approximation error for the proposed one is improved as the degree of approximation polynomial increases. Further, we propose a composite function approximation using inverse sine function to obtain approximation error smaller than the fundamental flat error with a small number of the non-scalar multiplications. For the direct approximation, we reduce the number of non-scalar multiplications by 30% by using odd function property of the minimax approximate polynomial of modular reduction function. From the numerical analysis, it is known that the composite function approximation is desirable when the running time of bootstrapping is important, and the direct approximation with odd function optimization is desirable when the depth is important.

* This work is supported by Samsung Advanced Institute of Technology.

Keywords: Approximate homomorphic encryption · Bootstrapping · Cheon-Kim-Kim-Song (CKKS) scheme · Fully homomorphic encryption (FHE) · Minimax approximate polynomial · Remez algorithm.

1 Introduction

Fully homomorphic encryption (FHE) is the encryption scheme enabling any logical operations [1, 8, 9, 12, 21] or arithmetic operations [5, 6] with encrypted data. The FHE scheme makes it possible to preserve security in data processing. However, in the traditional encryption schemes, they are not encrypted to enable the processing of encrypted data, which causes clients to be dissuaded from receiving services and prevent companies from developing various related systems because of the lack of clients' privacy. FHE solves this problem clearly so that clients can receive many services with ensuring their privacy.

First, Gentry constructed the FHE scheme by coming up with the idea of bootstrapping [11]. After this idea was introduced, cryptographers constructed many FHE schemes using bootstrapping. Approximate homomorphic encryption, which is also called a Cheon-Kim-Kim-Song (CKKS) scheme [6], is one of the promising FHE schemes, which deals with any real and complex numbers. The CKKS scheme is particularly in the spotlight for much potential power in many applications such as machine learning, in that data is usually represented by real numbers. Lots of researches for the optimization of the CKKS scheme have been done actively for practical use.

However, the slow running time of this scheme prevents it to be used practically in many applications. Especially, the bootstrapping operation is the biggest bottleneck in the slow running time of the CKKS scheme. Thus, the reduction of the running time for the bootstrapping operation of the CKKS scheme has actively been researched [2, 4, 13]. Especially, homomorphic modular reduction operation is one of the crucial parts of the bootstrapping of the CKKS scheme.

Since modular reduction operation cannot be performed exactly by operations that the CKKS scheme can support, the modular reduction function is approximated as some polynomials which can be performed by these operations. Many researches suggested the sub-optimal approximation polynomials that are close to the modular reduction function in the minimax aspect. Finding the exact minimax approximate polynomial for the modular reduction function is an important research topic that answers the exact trade-off between the degree of the approximate polynomial and the minimax approximation error. Besides, the algorithm to find the exact minimax approximate polynomial is directly applicable to the CKKS scheme, which gives the best performance in terms of the minimax approximation error.

There are some algorithms for obtaining the minimax approximate polynomial of some functions. One of them is the Remez algorithm [19] that returns the minimax generalized approximate polynomial for any continuous function on a closed interval. Parks-McClellan algorithm [16] uses the Remez algorithm for the optimal filter design in electrical engineering, which returns the minimax cosine polynomial for any filter conditions for two or three separate bands.

1.1 Previous Works

The CKKS scheme was firstly proposed in [6] without bootstrapping, which was a somewhat homomorphic encryption scheme supporting only the finite number of multiplications. Cheon et al. [4] firstly suggested bootstrapping operation with the homomorphic linear transformation between enabling transformation between slots and coefficients, and approximation of homomorphic modulus reduction function as the sine function. In homomorphic modular reduction, they approximated the sine function by evaluating its Taylor approximation and applying the double-angle formula. Although the double-angle formula reduces the number of operations compared to the direct Taylor approximation, it requires large depths. Chen et al. [2] improved the running time of bootstrapping operation by making homomorphic linear transformation and homomorphic modular reduction efficient. They applied a modified fast Fourier transform (FFT) algorithm to evaluate homomorphic linear transformation and used Chebyshev interpolation and Paterson-Stockmeyer algorithm to approximate the sine function. Chebyshev interpolation is known as a good sub-optimal approximate polynomial in the minimax aspect, but this is not the optimal approximate polynomial. Han et al. [13] proposed the residue number system (RNS) variant CKKS scheme using the generalized key switching method and improved the homomorphic modular reduction. While Chen et al. approximated the sine function in one interval, Han et al. approximated the cosine function only in the separated approximation regions, which enables to reduce the degree of polynomials and use simpler double-angle formula than that of the sine function. They obtained the upper bound on their maximum approximation error, but their approximate polynomial is not optimal. In the previous researches for the bootstrapping in the CKKS scheme, the modular reduction function was approximated by sine or cosine function, and then, the sine or cosine function was approximated by the approximate polynomial again. Up to now, the optimal algorithm to directly obtain the minimax approximate polynomial for the modular reduction function in the multiple approximation regions has not been introduced yet.

However, there is an example of using the exact minimax approximate polynomial of sign function for comparison operation in the CKKS scheme. That is, Cheon et al. [7] used an algorithm to obtain the minimax approximate polynomial only for sign function to accelerate their composition method for homomorphic comparison operations. They only used the well-known minimax approximation algorithm for sign function as a subroutine. There is no research to use the exact minimax approximate polynomial for bootstrapping of the CKKS scheme.

The Remez algorithm [19], also called the Remez exchange algorithm, is an iterative algorithm which obtains the minimax generalized approximate polynomial for a given continuous function on an interval $[a, b]$. There is its variant that obtains the minimax generalized approximate polynomial for a given continuous function on the multiple sub-intervals of an interval [10, 15, 19], which is less well-known than the original one. There is a crucial difference between the two algorithms in determining the new set of reference points in each iteration,

which is used to construct a generalized approximate polynomial in the next iteration. While the new set of reference points can be chosen naturally in the original one for an interval, there are many candidate points for the new set of reference points in the variant for the multiple sub-intervals of an interval. The variant algorithm chooses the new set of reference points which alternates in the sign of error and includes the global extreme point. Although this selection method ensures the convergence to the minimax generalized approximate polynomial, there are yet many candidate points for the new set of reference points satisfying these criteria. The Parks-McClellan filter design algorithm uses this variant of the Remez algorithm to design the optimal filter for a given condition, where its approximation domain is usually the union of two or three intervals.

1.2 Our Contribution

In this paper, we modify the Remez algorithm to directly obtain the optimal minimax generalized approximate polynomial for the modular reduction function in bootstrapping of the CKKS scheme. The proposed algorithm, called a modified Remez algorithm, enables us to find the minimax approximation error for the optimal minimax generalized approximate polynomial of any degree. For accelerating the Remez algorithm on multiple approximation intervals, we modify the criteria for the selection of new reference points for each iteration. Instead of the new reference points including the global extreme point in the new reference points, we choose the new reference points whose absolute sum is maximum. Although this new reference points for each iteration may not include the global maximum point, we prove that it is enough for the proposed approximation algorithm to converge to the optimal minimax generalized approximate polynomial, and this can reduce the number of iterations of the approximation algorithm in the bootstrapping of the CKKS scheme.

We also derive the optimal minimax approximation error of the approximate polynomial of the modular reduction function by the proposed modified Remez algorithm. From the numerical results, it is shown that the modified Chebyshev interpolation algorithm in [13] approximates to the optimal minimax approximation error pretty well until some degree of the approximate polynomial in terms of the approximation error, and for the higher degree of polynomials, there is a gap of two in the logarithmic scale with base two between the cosine function and the modular reduction function. We also find that there is some flat region in which the minimax approximation error does not decrease meaningfully as the degree of approximate polynomial increases. Precisely, in this flat region, the minimax approximate error seems to be saturated, and after reaching some value, it suddenly decreases linearly in the logarithm scale again. This phenomenon is related to the fundamental property of the modular reduction function. With this observation, note that we have to find approximate polynomials with higher degrees when we need somewhat higher precision for approximation.

The cosine approximation in [13] can use the double-angle formula to reduce the number of operations, but there is a fundamental limitation on the approximation error compared to the direct approximation for the modular reduction

function because they first approximate the modular reduction function by the sine or cosine function, and then, the sine or cosine function is approximated by the approximate polynomial again. Although Han et al. analyzed the minimax approximate error of their approximate polynomials with degrees higher than 76 for the cosine function, these minimax approximate errors are all smaller than the fundamental flat error between the modular reduction function and the cosine function. Thus, it is necessary to devise a better method to overcome the fundamental limitation. We propose a composite function approximation by composing the cosine function and the inverse sine function. By this method, the fundamental limitation of the error is removed, and thus we can obtain arbitrarily small minimax approximation error by this composite function approximation. We also show that this composite function approximation is effective in reducing the number of non-scalar multiplications. As a special case, we propose the best proportional constant of the original cosine approximation by using the composite function approximation, and this proportional constant gives the reduction of the fundamental flat error by 1/4 at no cost.

There are also cases where the depth is more important than the number of non-scalar multiplications. In this case, the direct approximation of modular reduction function is more desirable, in that the composite function approximation usually consumes some additional depths. We propose the two methods to reduce the number of non-scalar multiplications in the direct approximation. In these two methods, we use the fact that the optimal minimax approximate polynomials of the modular reduction function are odd functions. This ensures that the coefficients of even degree terms are always zero in the minimax approximate polynomial. The first method uses the Paterson-Stockmeyer algorithm, and the number of non-scalar multiplications is reduced by at most 30%. This method consumes one more depth compared to the original Paterson-Stockmeyer algorithm. The second method uses the Baby-step Giant-step algorithm, and also the number of non-scalar multiplications is reduced by at most 20%. This method does not need additional depth, and it ensures the optimal depth. By these various kinds of optimization of evaluation of the minimax approximate polynomials, we offer many options for various situations.

1.3 Outline

The outline of the paper is given as follows. Section 2 deals with some preliminaries for the CKKS scheme, approximation theory, and the Remez algorithm. In Section 3, we propose a modified Remez algorithm for the CKKS scheme with theoretical proof of its convergence to the minimax generalized approximate polynomial. Section 4 shows several simulation results including not only performance improvement of the minimax approximation error but also fundamental properties of the optimal minimax approximation error. In Section 5, we propose the composite function approximation which makes it possible to have arbitrarily small minimax error, and also propose the best proportional constant of the cosine approximation function. Section 6 shows improvements in the number of operations for the approximate polynomial directly obtained for

the modular reduction function using the odd function property of its optimal minimax approximate polynomial. Section 7 concludes the paper and discusses future works.

2 Preliminary

2.1 Notation

Let $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$, and $\mathbb{C}$ be sets of integers, rational numbers, real numbers, and complex numbers, respectively. Let $C[D]$ be a set of continuous functions on the domain D. Let $[d]$ be a set of positive integers less than or equal to d, i.e., $\{1, 2, \cdots, d\}$. Let $\mathsf{round}(x)$ be the function that outputs the integer nearest to x, and we do not have to consider the case of tie in this paper. For a power of two, M, let $\Phi_M(X) = X^N + 1$ be an M-th cyclotomic polynomial, where $M = 2N$. Let $\mathcal{R} = \mathbb{Z}[X]/\langle \Phi_M(X) \rangle$ and $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$. Let $\mathbb{Q}[X]/\langle \Phi_M(X) \rangle$ be a M-th cyclotomic field. For positive real number α, $\mathcal{DG}(\alpha)$ is defined as the distribution in $\mathbb{Z}^N$ whose entries are sampled independently from discrete Gaussian distribution of variance α^2. $\mathcal{HWT}(h)$ is a subset of $\{0, \pm 1\}^N$ with Hamming weight h. $\mathcal{ZO}(\rho)$ is the distribution in $\{0, \pm 1\}^N$ whose entries are sampled independently with probability $\rho/2$ for each of ± 1 and probability being zero, $1 - \rho$. The Chebyshev polynomials $T_n(x)$ are defined by $\cos n\theta = T_n(\cos \theta)$. The base of logarithm in this paper is two.

2.2 The CKKS Scheme

We introduce an overview of the CKKS scheme [6] in this section. The CKKS scheme supports several operations for encrypted data of real numbers or complex numbers. Since it deals with usually real numbers, the noise that ensures the security of the CKKS scheme can be embraced in the outside of the significant figures of the data, which is the crucial concept of the CKKS scheme.

Several independent messages are encoded into one polynomial by the canonical embedding before encryption. The canonical embedding σ embeds $a \in \mathbb{Q}[X]/\langle \Phi_M(X) \rangle$ into an element of $\mathbb{C}^N$ whose elements are values of a evaluated at the distinct roots of $\Phi_M(X)$. It is a well-known fact that the roots of $\Phi_M(X)$ are exactly the power of odd integers of the M-th root of unity, and $\mathbb{Z}_M^* = \langle -1, 5 \rangle$. Let $\mathbb{H} = \{(z_j)_{j \in \mathbb{Z}_M^*} : z_j = \overline{z_{-j}}\}$, and π be a natural projection from $\mathbb{H}$ to $\mathbb{C}^{N/2}$. Then, it is easily known that the range of σ is exactly $\mathbb{H}$. When $N/2$ messages of complex number constitute an element in $\mathbb{C}^{N/2}$, each coordinate is called a slot. The encoding and decoding procedures are given as follows.

- $\mathsf{Ecd}(\boldsymbol{z}; \Delta)$. For a vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$, return

$$m(X) = \sigma^{-1}\left(\lfloor \Delta \cdot \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}\right) \in \mathcal{R},$$

  where Δ is the scaling factor and $\lfloor \pi^{-1}(\boldsymbol{z}) \rceil_{\sigma(\mathcal{R})}$ denotes the discretization (rounding) of $\pi^{-1}(\boldsymbol{z})$ into an element of $\sigma(\mathcal{R})$.

– $\mathsf{Dcd}(m; \Delta)$. For a polynomial $m(X) \in \mathcal{R}$, return a vector $\boldsymbol{z} \in \mathbb{C}^{N/2}$ whose entry of index j is $z_j = \left\lfloor \Delta^{-1} \cdot m(\zeta_M^{5^j}) \right\rceil$ for $j \in \{0, 1, \cdots, N/2-1\}$, where ζ_M is the M-th root of unity.

Then, each procedure of the CKKS scheme is given as follows.

– $\mathsf{KeyGen}(1^\lambda)$:
  - Given λ as the security parameter, choose M as a power of two, an integer h, an integer P, a real positive number α, the fresh ciphertext modulus q_L, and the big ciphertext modulus Q, which will be the maximum ciphertext modulus.
  - Set the public key and the secret key as

$$\mathsf{sk} \coloneqq (1, s), \mathsf{pk} \coloneqq (-as + e, a) \in \mathcal{R}_{q_L}^2,$$

    where $s \leftarrow \mathcal{HWT}(h), a \leftarrow \mathcal{R}_{q_L}, e \leftarrow \mathcal{DG}(\alpha^2)$.
  - Set the evaluation key as

$$\mathsf{evk} \coloneqq (-a's + e' + Ps^2, a') \in \mathcal{R}_{Pq_L}^2,$$

    where $a' \leftarrow \mathcal{R}_{Pq_L}$ and $e' \leftarrow \mathcal{DG}(\alpha^2)$.
– $\mathsf{Enc}_{pk}(m \in \mathcal{R})$: Sample $v \leftarrow \mathcal{ZO}(0.5), e_0, e_1 \leftarrow \mathcal{DG}(\alpha^2)$, and return

$$\boldsymbol{c} = v \cdot pk + (m + e_0, e_1) \mod q_L.$$

– $\mathsf{Dec}_{sk}(\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2)$: Return $\bar{m} = \langle \boldsymbol{c}, sk \rangle \mod q_\ell$.
– $\mathsf{Add}(\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{q_\ell}^2)$: Return

$$\boldsymbol{c}_{\mathsf{add}} = \boldsymbol{c}_1 + \boldsymbol{c}_2 \mod q_\ell.$$

– $\mathsf{Mult}_{evk}(\boldsymbol{c}_1, \boldsymbol{c}_2 \in \mathcal{R}_{q_\ell}^2)$: For $\boldsymbol{c}_1 = (b_1, a_1), \boldsymbol{c}_2 = (b_2, a_2)$, let

$$(d_0, d_1, d_2) \coloneqq (b_1 b_2, a_1 b_2 + a_2 b_1, a_1 a_2) \mod q_\ell.$$

  Return

$$\boldsymbol{c}_{\mathsf{mult}} = (d_0, d_1) + \left\lfloor P^{-1} \cdot d_2 \cdot evk \right\rceil \mod q_\ell.$$

– $\mathsf{RS}_{l \to l'}(\boldsymbol{c} \in \mathcal{R}_{q_\ell}^2)$. Return

$$\boldsymbol{c}' = \left\lfloor \frac{q_{\ell'}}{q_\ell} \boldsymbol{c} \right\rceil \mod q_{\ell'}.$$

Each ciphertext has a level ℓ representing the maximum number of possible multiplications without bootstrapping. The modulus q_ℓ for each ciphertext of level ℓ is $p^\ell q_0$, where p is the scaling factor and q_0 is the base modulus.

There are additional homomorphic operations, rotation, and complex conjugation, which are used for homomorphic linear transformation in the bootstrapping of the CKKS scheme. Since these operations are not used in this paper, we omit these operations in this section.

Note that the RNS-variant CKKS scheme is inherently identical to the original CKKS scheme except the representation of each ciphertext or key. In this paper, we do not have to deal with these variants separately.

2.3 Bootstrapping for CKKS Scheme

The framework of the bootstrapping of the CKKS scheme was introduced in [6]. The purpose of bootstrapping is to refresh the ciphertext of level 0, whose multiplication cannot be performed anymore, to the fresh ciphertext of level L having the same messages. The bootstrapping is composed of the following four steps:

i) Modulus raising
ii) Homomorphic linear transformation; CoeffToSlot
iii) Homomorphic modular reduction
iv) Homomorphic linear transformation; SlotToCoeff

Modulus Raising: The starting point of bootstrapping is modulus raising, where we simply consider the ciphertext of level 0 as an element of $\mathcal{R}_Q^2$, instead of $\mathcal{R}_{q_0}^2$. Since the ciphertext of level 0 is supposed to be $\langle \mathsf{ct}, \mathsf{sk} \rangle \approx m \mod q_0$, we have $\langle \mathsf{ct}, \mathsf{sk} \rangle \approx m + q_0 I \mod Q$ for some $I \in \mathcal{R}$ when we try to decrypt it. We are assured that the absolute values of coefficients of I are rather small, for example, usually smaller than 12, because coefficients of sk consist of small numbers. The crucial part of the bootstrapping of the CKKS scheme is to make ct' such that $\langle \mathsf{ct}', \mathsf{sk} \rangle \approx m \mod q_L$. This is divided into two parts: homomorphic linear transform and homomorphic evaluation of modular reduction function.

Homomorphic Linear Transformation: The ciphertext ct after modulus raising can be considered as the ciphertext encrypting $m + q_0 I$, and thus we now have to perform modular reduction to coefficients of message polynomial homomorphically. However, the operations we have are all for slots, not coefficients of the message polynomial. Thus, to perform some meaningful operations on coefficients, we have to convert ct into a ciphertext that encrypts coefficients of $m + q_0 I$ as its slots, and after evaluation of homomorphic modular reduction function, we have to reversely convert this ciphertext into the other ciphertext ct' that encrypts the slots of the previous ciphertext as the coefficients of its message. These two operations are called CoeffToSlot and SlotToCoeff operations. These operations are regarded as homomorphic evaluation of encoding and decoding of messages, which are a linear transformation by some variants of Vandermonte matrix for roots of $\Phi_M(x)$. This can be performed by general homomorphic matrix multiplication [4], or FFT-like operation [2].

Homomorphic Modular Reduction Function: After CoeffToSlot is performed, we now have to perform modular reduction homomorphically on each slot in modulus q_0. This procedure is called EvalMod. This modular reduction function is not an arithmetic function, and even not a continuous function. Fortunately, by restricting the range of the messages such that m/q_0 is small enough, the approximation region can be given only near multiples of q_0. This allows us to approximate the modular reduction function more effectively. Since

the operations that the CKKS supports are arithmetic operations, most of the researches [2,4,13] dealing with CKKS bootstrapping approximate the modular reduction function with some polynomials, which is sub-optimal approximate polynomials.

2.4 Approximation Theory

Approximation theory is needed to prove the convergence of the minimax polynomial obtained by the proposed modified Remez algorithm. Assume that functions are defined on a union of the finite number of closed and bounded intervals in the real line. From the following well-known theorem [20] in real analysis, we are convinced that this domain of functions is a compact set.

Theorem 2.1 (**[20] Bolzano-Weierstrass Theorem**). *A subset of $\mathbb{R}^n$ is a compact set if and only if it is closed and bounded.*

A union of the finite number of closed and bounded intervals in the real line is trivially closed and bounded, and thus this domain is a compact set by Bolzano-Weierstrass theorem. This theorem will be used in the convergence proof of the modified Remez algorithm in Section 3.

The next theorem [20] states that any continuous function on compact set in the real line can be approximated with an arbitrarily small error by polynomial approximation. In fact, the theorem includes the case of continuous functions on more general domains, but we only use the special case on compact sets in the real line in this paper.

Theorem 2.2 (**[20] Stone-Weierstrass Theorem**). *Assume that f is a continuous function on the compact subset D of the real line. For every $\epsilon > 0$, there is a polynomial p such that $\|f - p\|_\infty < \epsilon$.*

There are many theorems for the minimax approximate polynomials of a function defined on a compact set in approximation theory. Before the introduction of these theorems, we refer to a definition of Haar condition of a set of functions that deals with the generalized version of power bases used in polynomial approximation and its equivalent statement. We also define a generalized polynomial with regard to bases that satisfy Haar condition. It is a well-known fact that the power basis $\{1, x, x^2, \cdots, x^d\}$ satisfies the Haar condition. Thus, if an argument deals with the generalized polynomials with regard to a set of basis functions satisfying Haar condition, it naturally includes the case of polynomials.

Definition 2.3 (**[3] Haar's Condition and Generalized Polynomial**). *A set of functions $\{g_1, g_2, \cdots, g_n\}$ satisfies the Haar condition if each g_i is continuous and if each determinant*

$$D[x_1, \cdots, x_n] = \begin{vmatrix} g_1(x_1) & \cdots & g_n(x_1) \\ \vdots & \ddots & \vdots \\ g_1(x_n) & \cdots & g_n(x_n) \end{vmatrix}$$

for any n distinct points $x_1, \cdots, x_n$ is not zero. Then, a linear combination of $\{g_1, g_2, \cdots, g_n\}$ is called a generalized polynomial.

Lemma 2.4 ([3]). *A set of functions $\{g_1, \cdots, g_n\}$ satisfies the Haar condition if and only if zero polynomial is the only generalized polynomial $\sum_i c_i g_i$ that has more than $n-1$ roots.*

Firstly, we are convinced that there is the unique minimax generalized approximate polynomial in the union of the finite number of closed and bounded intervals as in the following two theorems.

Theorem 2.5 ([3] **Existence of Best Approximations**). *Let $\mathcal{F}$ be a normed linear space, and f is any fixed element in $\mathcal{F}$. If $\mathcal{S}$ is a linear subspace of $\mathcal{F}$ with finite dimension, $\mathcal{S}$ contains at least one element of minimum distance from f.*

Theorem 2.6 ([3] **Haar's Unicity Theorem**). *Let f be any continuous function on a compact set K. Then the minimax generalized polynomial $\sum_i c_i g_i$ of f is unique if and only if $\{g_1, g_2, \cdots, g_n\}$ satisfies the Haar condition.*

In Theorem 2.5 for the existence of the best approximation, consider a set $\mathcal{F}$ of continuous functions on a union D of the finite number of closed and bounded intervals. We can easily know that $\mathcal{F}$ is a linear space with a max-norm $\|f\|_\infty = \max_{x \in D} |f(x)|$. The set $\mathcal{P}_d$ of generalized polynomials with regard to the finite number of basis functions on D is a finite-dimensional linear subspace. Then, from Theorem 2.5, there is at least one minimax generalized approximate polynomial for any $f \in \mathcal{F}$.

We now introduce the core property of the minimax generalized approximate polynomial for a function on D.

Theorem 2.7 ([3] **Chebyshev Alternation Theorem**). *Let $\{g_1, \cdots, g_n\}$ be a set of functions in $C[a,b]$ satisfying the Haar condition, and let D be a closed subset of $[a,b]$. A generalized polynomial $p = \sum_i c_i g_i$ is the minimax approximation on D to any given $f \in C[D]$ if and only if there are $n+1$ distinct elements $x_0 < \cdots < x_n$ in D such that for the error function $r = f - p$ restricted on D,*

$$r(x_i) = -r(x_{i-1}) = \pm \|r\|_\infty.$$

This condition is also called equioscillation condition. This means that if we find a generalized polynomial satisfying the equioscillation condition, then this is the unique minimax generalized approximate polynomial, needless to compare with the maximum approximation error of any polynomials.

The following three theorems are used to prove the convergence of the modified Remez algorithm in Section 3.

Theorem 2.8 ([3] **de La Vallee Poussin Theorem**). *Let $\{g_1, \cdots, g_n\}$ is a set of continuous functions on $[a,b]$ satisfying the Haar condition. Let f be a continuous on $[a,b]$, and p be a generalized polynomial such that $p - f$ has alternately positive and negative values at $n+1$ consecutive points x_i in $[a,b]$.*

Let p^* be a minimax generalized approximate polynomial for f, and $e(f)$ be the minimax approximation error of p^*. Then, we have

$$e(f) \geq \min_i |p(x_i) - f(x_i)|.$$

Lemma 2.9 ([3]). *Let $\{g_1, \cdots, g_n\}$ be a set of continuous functions satisfying the Haar condition. Assume that $x_1 < \cdots < x_n$ and $y_1 < \cdots < y_n$. Then the determinants $D[x_1, \cdots, x_n]$ and $D[y_1, \cdots, y_n]$, defined by Definition 2.3, have the same sign.*

Theorem 2.10 ([3] **Strong Unicity Theorem**). *Let $\{g_1, \cdots, g_n\}$ be a set of functions satisfying the Haar condition, and let p^* be the minimax generalized polynomial of a given continuous function u. Then, there is a constant $\gamma > 0$ determined by f such that for any generalized polynomial p, we have*

$$\|p - f\|_\infty \geq \|p^* - f\|_\infty + \gamma \|p - p^*\|_\infty.$$

2.5 Algorithms for Minimax Approximation

Remez Algorithm Remez algorithm [3, 18, 19] is an iterative algorithm that always returns the minimax approximate polynomial for any continuous function on an interval of $[a, b]$. This algorithm strongly uses Chebyshev alternation theorem [3] in that its purpose is finding the polynomial satisfying equioscillation condition. In fact, the Remez algorithm can be applied to obtain the minimax approximate generalized polynomial, whose basis function $\{g_1, \cdots, g_n\}$ satisfies the Haar condition. The following explanation includes the generalization of the Remez algorithm, and if we want to obtain the minimax approximate polynomial of degree d, we choose the basis function $\{g_1, \cdots, g_n\}$ by the power basis $\{1, x, \cdots, x^d\}$, where $n = d + 1$.

Remez algorithm firstly initializes the set of reference points $\{x_1, \cdots, x_{n+1}\}$, which will be converged to the extreme points of the minimax generalized approximate polynomial. Then, it obtains the minimax generalized approximate polynomial in regard to only the set of reference points. Since the set of reference points is the set of finite points in $[a, b]$, it is a closed subset of $[a, b]$, and thus Chebyshev alternation theorem holds on the set of reference points. Let $f(x)$ be a continuous function on $[a, b]$. The minimax generalized approximate polynomial on the set of reference points is exactly the generalized polynomial $p(x)$ with the basis $\{g_1, \cdots, g_n\}$ satisfying

$$p(x_i) - f(x_i) = (-1)^i E \quad i = 1, \cdots, d + 2$$

for some real number E. This forms a system of linear equations having $n + 1$ equations and $n + 1$ variables of n coefficients of $p(x)$ and E, which is ensured to be not singular by Haar's condition, and thus we can obtain the generalized

polynomial $p(x)$. Then, we can find n zeros of $p(x)-f(x)$, z_i between x_i and x_{i+1}, $i = 1, 2, \cdots, n$, and we can find $n+1$ extreme points $y_1, \cdots, y_{n+1}$ of $p(x) - f(x)$ in each $[z_{i-1}, z_i]$, where $z_0 = a$ and $z_{n+1} = b$. That is, we choose the minimum point of $p(x)-f(x)$ in $[z_{i-1}, z_i]$ if $p(x_i)-f(x_i) < 0$, and we choose the maximum point of $p(x) - f(x)$ in $[z_{i-1}, z_i]$ if $p(x_i) - f(x_i) > 0$. Thus, we find a new set of extreme points $y_1, \cdots, y_{n+1}$. If this satisfies equioscillation condition, the Remez algorithm returns $p(x)$ as the minimax generalized approximate polynomial from the Chebyshev alternation theorem. Otherwise, it replaces the set of reference points with these extreme points $y_1, \cdots, y_{n+1}$ and processes above steps again. This is the Remez algorithm in Algorithm 1. The Remez algorithm is proved to be always converged to the minimax generalized approximate polynomial by the following theorem.

Algorithm 1: Remez

Input : An input domain $[a, b]$, a continuous function f on $[a, b]$, an approximation parameter δ, and a basis $\{g_1, \cdots, g_n\}$.
Output: The minimax generalized approximate polynomial p for f

1 Select $x_1, x_2, \cdots, x_{d+2} \in [a, b]$ in strictly increasing order.
2 Find the generalized polynomial $p(x)$ in terms of $\{g_1, \cdots, g_n\}$ with $p(x_i) - f(x_i) = (-1)^i E$ for some E.
3 Divide the interval into $n+1$ sections $[z_{i-1}, z_i]$, $i = 1, \cdots, n+1$, from zeros $z_1, \cdots, z_n$ of $p(x) - f(x)$, where $x_i < z_i < x_{i+1}$, and boundary points $z_0 = a, z_{n+1} = b$.
4 Find the maximum (resp. minimum) points for each section when $p(x_i) - f(x_i)$ has positive (resp. nagative) value. Denote these extreme points $y_1, \cdots, y_{n+1}$.
5 $\epsilon_{\mathsf{max}} \leftarrow \max_i |p(y_i) - f(y_i)|$
6 $\epsilon_{\mathsf{min}} \leftarrow \min_i |p(y_i) - f(y_i)|$
7 **if** $(\epsilon_{\mathsf{max}} - \epsilon_{\mathsf{min}})/\epsilon_{\mathsf{min}} < \delta$ **then**
8 | **return** $p(x)$
9 **else**
10 | Replace x_i's with y_i's and go to line 2.
11 **end**

Theorem 2.11 (**[3] Convergence of Remez Algorithm**). *Let $\{g_1, \cdots, g_n\}$ be a set of functions satisfying the Haar condition, p_k be a generalized polynomial generated in the k-th iteration of Remez algorithm, and p^* be the minimax generalized polynomial of a given f. Then, p_k converges uniformly to p^* by the following inequality,*

$$\|p_k - p^*\|_\infty \leq A\theta^k,$$

where A is a non-negative constant, and $0 < \theta < 1$.

A Variant of the Remez Algorithm The above Remez algorithm can be extended to the multiple sub-intervals of an interval [10, 15, 19]. The variant of

the Remez algorithm is the same as Algorithm 1, except Step 3 and 4. For each iteration, firstly, we find all of the local extreme points of the error function $p - f$ whose absolute error values are larger than the absolute error values at the current reference points. Then, we choose $n + 1$ new extreme points among these points satisfying the following two criteria:

i) The error values alternate in sign.
ii) A new set of extreme points includes the global extreme point.

These two criteria are known to ensure the convergence to the minimax generalized polynomial, even though we could not find the exact proof of its convergence. However, it is noted that there are many choices of sets of extreme points satisfying these criteria. In the next section, we modify the variant of the Remez algorithm, where one of the two criteria is changed.

3 Modified Remez Algorithm

In this section, we propose the modified Remez algorithm to efficiently obtain the optimal minimax generalized approximate polynomial for continuous function on the union of finitely many closed intervals. The function we are going to approximate is the normalized modular reduction function defined in only near finitely many integers given as

$$\mathsf{normod}(x) = x - \mathsf{round}(x), \quad x \in \bigcup_{i=-(K-1)}^{K-1} [i - \epsilon, i + \epsilon],$$

where K determines the number of intervals in the domain. normod function corresponds to the modular reduction function scaled for both its domain and range.

In addition, Han et al. [13] uses the cosine function to approximate $\mathsf{normod}(x)$ to use double-angle formula for efficient homomorphic evaluation. If we use double-angle formula ℓ times, we have to approximate the following cosine function

$$\cos\left(\frac{2\pi}{2^\ell}\left(x - \frac{1}{4}\right)\right), \quad x \in \bigcup_{i=-(K-1)}^{K-1} [i - \epsilon, i + \epsilon].$$

To design an approximation algorithm that deals with the above two functions, we assume the general continuous function defined on an union of finitely many closed intervals, which is given as

$$D = \bigcup_{i=1}^{t} [a_i, b_i] \subset [a, b] \subset \mathbb{R}$$

where $a_i < b_i < a_{i+1} < b_{i+1}$ for all $i = 1, \cdots, t - 1$.

When we modify the variant of Remez algorithm to approximate a given continuous function on D with a polynomial having a degree less than or equal

to d, called the modified Remez algorithm, we have to consider two crucial points. One is to establish an efficient criterion for choosing new $d+2$ reference points among several extreme points, and the other is to make some steps in the modified Remez algorithm efficient. We deal with these two issues for the modified Remez algorithm in Sections 3.1 and 3.3, respectively.

3.1 Modified Remez Algorithm with Criteria for Choosing Extreme Points

Assume that we apply the variant of Remez algorithm on D and use $\{g_1, \cdots, g_n\}$ satisfying Haar condition on $[a, b]$ as the basis of generalized polynomial. After obtaining the minimax generalized approximate polynomial in regard to the set of reference points for each iteration, we have to choose a new set of reference points for the next iteration. However, there are many boundary points in D and all these boundary points have to be considered as extreme points of the error function. For this reason, there are many cases of selecting $n+1$ points among these extreme points. For bootstrapping in the CKKS scheme, there are many intervals to be considered, and thus there are lots of candidate extreme points. Since the criterion of the variant of the Remez algorithm cannot determine the unique new set of reference points for each iteration, it is necessary to make how to choose $n+1$ points for each iteration to reduce the number of iterations as small as possible. Otherwise, it requires a large number of iteration numbers for convergence to the minimax generalized approximate polynomial. In addition, it is very important to ensure that this criterion always leads to convergence to the minimax generalized approximate polynomial. If the criterion is not designed properly, the modified Remez algorithm may not converge into a single generalized polynomial in some cases.

In order to set the criterion for selecting $n+1$ reference points, we need to define a simple function for extreme points, $\mu_{p,f} : D \to \{-1, 0, 1\}$ as follows,

$$\mu_{p,f}(x) = \begin{cases} 1 & p(x) - f(x) \text{ is a local maximum value at } x \text{ on } D \\ -1 & p(x) - f(x) \text{ is a local minimum value at } x \text{ on } D \\ 0 & \text{otherwise,} \end{cases}$$

where $p(x)$ is a generalized polynomial obtained in that iteration and $f(x)$ is a continuous function on D to be approximated. We abuse the notation $\mu_{p,f}$ as μ.

If we gather all extreme points of $p(x) - f(x)$ into a set B, we can assume that B is a finite set, that is, $B = \{x_1, x_2, \cdots, x_m\}$. If there is an interval in B, we can choose a point in that interval. Assume that B is ordered in increasing order, $x_1 < x_2 < \cdots < x_m$, and then the values of μ at these points are always 1 or -1. We can ensure that $m \geq n+1$, which will be proved in Theorem 3.1. Let $\mathcal{S}$ be a set of functions defined as

$$\mathcal{S} = \{\sigma : [n+1] \to [m] \mid \sigma(i) < \sigma(i+1) \text{ for all } i = 1, \cdots, n\}.$$

Clearly, $\mathcal{S}$ has only the identity function if $n+1 = m$.

Then, we set three criteria for selecting $n+1$ extreme points as follows:

i) *Local extreme value condition.* If E is the absolute value of error at points in the set of reference points, then we have

$$\min_i \mu(x_{\sigma(i)})(p(x_{\sigma(i)}) - f(x_{\sigma(i)})) \geq E.$$

ii) *Alternating condition.* $\mu(x_{\sigma(i)}) \cdot \mu(x_{\sigma(i+1)}) = -1$ for $i = 1, \cdots, n$.
iii) *Maximum absolute sum condition.* Among σ's satisfying the above two conditions, choose σ maximizing the following value

$$\sum_{i=1}^{n+1} |p(x_{\sigma(i)}) - f(x_{\sigma(i)})|.$$

It is noted that the local extreme value condition in i) means that the extreme points are discarded if the local maximum value of $p(x) - f(x)$ is negative or the local minimum of $p(x) - f(x)$ is positive.

As we will see in the convergence proof in Section 3.2, the absolute value of error at current reference points $x_1, \cdots, x_{n+1}$ should be less than the minimax approximation error, and it increases and converges to the minimax approximation error as the number of iterations increases. Further, this value is weighted average of the absolute values of errors of the generalized approximate polynomial in previous iteration at the $x_1, \cdots, x_{n+1}$. The maximum absolute condition helps for the absolute value of error at current reference points to converge to the minimax approximation error fast.

Note that the first two conditions are also included in the variant of the Remez algorithm, and the third condition, the maximum absolute sum condition, is the replacement of the condition that the new set of reference points includes the global extreme point. The numerical analysis will show that the third condition makes the proposed modified Remez algorithm to converge to the optimal minimax generalized approximate polynomial fast. Although there are some cases in which the global maximum point is not included in the new set of reference points chosen by the maximum absolute sum condition, we will prove that the maximum absolute sum condition is enough for the modified Remez algorithm to converge to the minimax generalized approximate polynomial. Further, it can be found in the proof that the maximum absolute sum condition is the best and natural strategy for choosing the new set of reference points.

We have to check that $\mathcal{S}$ always contains at least one element σ_0 that satisfies the local extreme value condition and the alternating condition, and has $\sigma_0(i_0)$ for some i_0 such that $|p(x_{\sigma_0(i_0)}) - f(x_{\sigma_0(i_0)})| = \|p - f\|_\infty$. This existence is in fact the basic assumption of the variant of Remez algorithm, but we prove this existence for mathematical clarification.

Theorem 3.1. *Let B and $\mathcal{S}$ be the sets defined above. Then, there is at least one element in $\mathcal{S}$ which satisfies the local extreme value condition and the alternating condition and has $\sigma_0(i_0)$ for some i_0 such that $|p(x_{\sigma_0(i_0)}) - f(x_{\sigma_0(i_0)})| = \|p - f\|_\infty$.*

Proof. Let a_i and b_i be the boundary points in D defined above and let $t_1, t_2, \cdots, t_{n+1} \in D$ be the reference points used to construct $p_k(x)$ at the k-th iteration. Without loss of generality, $t_i < t_{i+1}$ for all $i = 1, \cdots, n$, and the following equation for some proper positive value E is satisfied as

$$p(t_i) - f(t_i) = (-1)^{i-1}E.$$

Let u_{2i-1} be the largest point among all a_j and t_{2j}'s which are less than or equal to t_{2i-1}, and let v_{2i-1} be the smallest point among all b_j and t_{2j}'s which are larger than or equal to t_{2i-1}. Then, firstly, we prove that there exists at least one local maximum point c_{2i-1} of $p_k(x)-f(x)$ in $[u_{2i-1}, v_{2i-1}]$, and $c_{2i-1} < t_{2i} < c_{2i+1}$ for all possible i. From the extreme value theorem for continuous function on interval [20], there exists at least one maximum point of $p_k(x) - f(x)$ in $[u_{2i-1}, v_{2i-1}]$, since $p_k(x) - f(x)$ is continuous on D. We denote this value at the maximum point as c_{2i-1}. Since t_{2i-1} is in $[u_{2i-1}, v_{2i-1}]$, $p_k(c_{2i-1}) - f(c_{2i-1}) \geq E > -E = p_k(t_{2j}) - f(t_{2j})$ for all possible j, and thus c_{2i-1} cannot be equal to any t_{2i}'s. Since elements appeared more than once in intervals $[u_{2i-1}, v_{2i-1}]$, $i = 1, 2, \cdots, \lfloor \frac{n+2}{2} \rfloor$, are only t_{2i}'s and $v_{2i-1} \leq t_{2i} \leq u_{2i+1}$ for all possible i, we now prove that $c_{2i-1} < t_{2i}$ and $t_{2i} < c_{2i+1}$.

Let u_{2i} be the largest point among all a_j and c_{2j-1}'s which are less than or equal to t_{2i}, and let v_{2i} be the smallest point among all b_j and c_{2j-1}'s which are larger than or equal to t_{2i}. Then, we prove that there exists at least one local minimum point c_{2i} of $p_k(x) - f(x)$ in $[u_{2i}, v_{2i}]$, and c_i's are sorted in strictly increasing order. Again, from the extreme value theorem for continuous function on interval, there exists at least one minimum point of $p_k(x) - f(x)$ in $[u_{2i}, v_{2i}]$. We denote this value at the minimum point as c_{2i}. Since t_{2i} is in $[u_{2i}, v_{2i}]$, $p_k(c_{2i}) - f(t_{2j}) \leq -E < E \leq p_k(c_{2j-1}) - f(c_{2j-1})$ for all possible j, and thus c_{2i} cannot be equal to any c_{2j-1}. Since elements appeared more than once in intervals $[u_{2i}, v_{2i}]$, $i = 1, 2, \cdots, \lfloor \frac{n+2}{2} \rfloor$, are only c_{2i-1}'s and $v_{2i} \leq c_{2i+1} \leq u_{2i+2}$ for all possible i, we now prove that c_i's are sorted in strictly increasing order.

Since c_i's are all local extreme points, $c_i \in B$ for all i. Then, we can set $\sigma' \in \mathcal{S}$ such that $x_{\sigma'(i)} = c_i$. Since c_{2i-1}'s are local maximum points and c_{2i}'s are local minimum points, σ' satisfies alternating condition. Since $\mu(c_i)(p(c_i)-f(c_i)) \geq E$, σ' also satisfies the local extreme value condition. If one of c_i has the maximum absolute value of $p - f$, we are done.

Assume that all of c_i's do not have the maximum absolute value of $p_k - f$. Let x_m be the global extreme point of $p_k - f$. If there is some k such that $c_k < x_m < c_{k+1}$, either c_k or c_{k+1} has the same value of μ at x_m. Then, we replace it with x_m and define this function as σ_0. Since σ_0 satisfies all of conditions in Theorem 3.1, we are done.

If $x_m < c_1$ or $x_m > c_{n+1}$, we separate it into two cases again. If $\mu(x_m) = \mu(c_1)$ (resp. $\mu(x_m) = \mu(c_{n+1})$), we just replace c_1 (resp. c_{n+1}) with x_m and define this function as σ_0, and σ_0 satisfies all these conditions. If $\mu(x_m) \neq \mu(c_1)$ (resp. $\mu(x_m) \neq \mu(c_{n+1})$), we replace c_{n+1} (resp. c_1) with x_m, and relabel the points to define the new function σ_0. This also satisfies all three conditions. Thus, we prove it.

□

Remark. This theorem also ensures that $m \geq n+1$. If $m < n+1$, $\mathcal{S}$ has to be empty. This theorem ensures that there is at least one element in $\mathcal{S}$, we can be convinced that $m \geq n+1$.

Now we propose the modified Remez algorithm for the continuous function on the union of finitely many closed intervals as in Algorithm 2. The local extreme value condition is reflected in Step 3, and the alternating condition and the maximum absolute sum condition are reflected in Step 4. Theorem 3.1 ensures the basic validity of Algorithm 2.

Algorithm 2: ModifiedRemez

Input : An input domain $D = \bigcup_{i=1}^{t}[a_i, b_i] \subset \mathbb{R}$, a continuous function f on D, an approximation parameter δ, and a basis $\{g_1, \cdots, g_n\}$
Output: The minimax generalized approximate polynomial p for f

1 Select $x_1, x_2, \cdots, x_{n+1} \in D$ in strictly increasing order.
2 Find the generalized polynomial $p(x)$ with $p(x_i) - f(x_i) = (-1)^i E$ for some E.
3 Gather all extreme and boundary points such that $\mu_{p,f}(x)(p(x) - f(x)) \geq |E|$ into a set B.
4 Find $n+1$ extreme points $y_1 < y_2 < \cdots < y_{n+1}$ with alternating condition and maximum absolute sum condition in B.
5 $\epsilon_{\mathsf{max}} \leftarrow \max_i |p(y_i) - f(y_i)|$
6 $\epsilon_{\mathsf{min}} \leftarrow \min_i |p(y_i) - f(y_i)|$
7 **if** $(\epsilon_{\mathsf{max}} - \epsilon_{\mathsf{min}})/\epsilon_{\mathsf{min}} < \delta$ **then**
8 | **return** $p(x)$
9 **else**
10 | Replace x_i's with y_i's and go to line 2.
11 **end**

3.2 Convergence of Modified Remez Algorithm

We now have to prove that the proposed algorithm always converges to the minimax generalized approximate polynomial for given piecewise continuous function on D. This proof is similar to the convergence proof of the original Remez algorithm on one closed interval [3, 18], but there are a few more general arguments than the original proof. This convergence proof includes the proof for both the variant of the Remez algorithm and the modified Remez algorithm.

Before proving the convergence, we have to generalize the de La Vallee Poussin theorem, which was used to prove the convergence of the original Remez algorithm on an interval. Since original de La Vallee Poussin theorem [3] only deals with a single interval, we generalize it in the following theorem that deals with the closed subset of an interval, whose proof is almost the same as that of the original theorem.

Lemma 3.2 (**Generalized de La Vallee Poussin Theorem**). *Let* $\{g_1, \cdots, g_n\}$ *is a set of continuous functions on* $[a, b]$ *satisfying the Haar condition, and let* D *be a closed subset of* $[a, b]$. *Let* f *be a continuous on* D, *and* p *be a generalized polynomial such that* $p - f$ *has alternately positive and negative values at* $n + 1$ *consecutive points* x_i *in* D. *Let* p^* *be a minimax generalized approximate polynomial for* f *on* D, *and* $e(f)$ *be the minimax approximation error of* p^*. *Then, we have*

$$e_D(f) \geq \min_i |p(x_i) - f(x_i)|.$$

Proof. Assume that the above statement is false. Then, there is a generalized polynomial p_0 such that $p_0 - f$ has alternately positive and negative values at $n + 1$ consecutive points in D, and

$$\|p^* - f\|_\infty < |p_0(x_i) - f(x_i)| \tag{1}$$

for all i. Then, $p_0 - p^* = (p_0 - f) - (p^* - f)$ has alternately positive and negative values at the consecutive x_i, which leads to the fact that there is n roots in $[a, b]$. From Lemma 2.4, $p_0 - p^*$ has to be zero, which is contradiction. □

We now prove the convergence of Algorithm 2.

Theorem 3.3. *Let* $\{g_1, \cdots, g_n\}$ *be a set of functions satisfying the Haar condition on* $[a, b]$, D *be the multiple sub-intervals of* $[a, b]$, *and* f *be a continuous function on* D. *Let* p_k *be a generalized approximate polynomial generated in the* k*-th iteration of the modified Remez algorithm, and* p^* *be the optimal minimax generalized approximate polynomial of* f. *Then, as* k *increases,* p_k *converges uniformly to* p^* *as in the following inequality*

$$\|p_k - p^*\|_\infty \leq A\theta^k,$$

where A *is a non-negative constant and* $0 < \theta < 1$.

Proof. Let $\{x_1^{(0)}, \cdots, x_{n+1}^{(0)}\}$ be the initial set of reference points and $\{x_1^{(k)}, \cdots, x_{n+1}^{(k)}\}$ be the new set of reference points chosen at the end of iteration k. Let $r_k = p_k - f$ be the error function of p_k and $r^* = p^* - f$ be the error function of p^*. Since p_k is generated such that the absolute values of the error function r_k at the reference points $x_i^{(k-1)}$, $i = 1, 2, \cdots, n + 1$ are the same. For $k \geq 1$, we define

$$\alpha_k = \min_i |r_k(x_i^{(k-1)})| = \max_i |r_k(x_i^{(k-1)})|,$$

$$\beta_k = \|r_k\|_\infty,$$

$$\gamma_k = \min_i |r_k(x_i^{(k)})|. \tag{2}$$

Define $\beta^* = \|r^*\|_\infty$. We have $\beta^* \geq \gamma_k$ from Lemma 3.2, $\beta_k \geq \beta^*$ by definition of p^*, and $\gamma_k \geq \alpha_k$ by the local extreme value condition for new set of reference points. Then, we have

$$\alpha_k \leq \gamma_k \leq \beta^* \leq \beta_k.$$

Let $c^{(k)} = [c_1^{(k)}, \cdots, c_n^{(k)}]^T$ be the coefficient vector of p_k. Then, $c^{(k)}$ is the solution vector of the following system of linear equations

$$(-1)^{i+1}h^{(k)} + \sum_{j=1}^{n} c_j^{(k)} g_j(x_i^{(k-1)}) = f(x_i^{(k-1)}), \quad i = 1, \cdots, n+1 \tag{3}$$

for the $n+1$ unknowns $h^{(k)}$ and $c_j^{(k)}$'s, and $|h^{(k)}| = \alpha_k$. From Theorem 2.6, we assure that the system of linear equations in (3) is nonsingular, which can be rewritten as in the matrix equation for $k+1$, instead of k,

$$\begin{bmatrix} 1 & g_1(x_1^{(k)}) & \cdots & g_n(x_1^{(k)}) \\ -1 & g_1(x_2^{(k)}) & \cdots & g_n(x_2^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^n & g_1(x_{n+1}^{(k)}) & \cdots & g_n(x_{n+1}^{(k)}) \end{bmatrix} \begin{bmatrix} h^{(k+1)} \\ c_1^{(k+1)} \\ \vdots \\ c_n^{(k+1)} \end{bmatrix} = \begin{bmatrix} f(x_1^{(k)}) \\ f(x_2^{(k)}) \\ \vdots \\ f(x_{n+1}^{(k)}) \end{bmatrix}. \tag{4}$$

From Cramer's rule, we can find $h^{(k+1)}$ as

$$h^{(k+1)} = \begin{vmatrix} f(x_1^{(k)}) & g_1(x_1^{(k)}) & \cdots & g_n(x_1^{(k)}) \\ f(x_2^{(k)}) & g_1(x_2^{(k)}) & \cdots & g_n(x_2^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ f(x_{n+1}^{(k)}) & g_1(x_{n+1}^{(k)}) & \cdots & g_n(x_{n+1}^{(k)}) \end{vmatrix} \Bigg/ \begin{vmatrix} 1 & g_1(x_1^{(k)}) & \cdots & g_n(x_1^{(k)}) \\ -1 & g_1(x_2^{(k)}) & \cdots & g_n(x_2^{(k)}) \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^n & g_1(x_{n+1}^{(k)}) & \cdots & g_n(x_{n+1}^{(k)}) \end{vmatrix}. \tag{5}$$

Let $M_i^{(k)}$ be the minor of the matrix in (4) removing the first column and the i-th row. Then, (5) can be written as

$$h^{(k+1)} = \frac{\sum_{i=1}^{n+1} f(x_i^{(k)}) M_i^{(k)} (-1)^{i+1}}{\sum_{j=1}^{n+1} M_j^{(k)}}. \tag{6}$$

If f is replaced by any generalized polynomial $p = \sum_{j=1}^{n} c'_j g_j$ in (4), the minimax approximation on $\{x_1^{(k)}, \cdots, x_{n+1}^{(k)}\}$ is p itself. This leads to

$$\frac{\sum_{i=1}^{n+1} p_k(x_i^{(k)}) M_i^{(k)} (-1)^{i+1}}{\sum_{j=1}^{n+1} M_j^{(k)}} = 0. \tag{7}$$

By substracting (6) from (7), and $r_k = p_k - f$, we have

$$-h^{(k+1)} = \frac{\sum_{i=1}^{n+1} r_k(x_i^{(k)}) M_i^{(k)} (-1)^{i+1}}{\sum_{j=1}^{n+1} M_j^{(k)}}.$$

By the fact that $r_k(x_i^{(k)})$'s alternate in sign by the alternating condition for new set of reference points, and all minors M_i have the same sign by Lemma 2.9, we have

$$\left| \sum_{i=1}^{n+1} r_k(x_i^{(k)}) (-1)^i M_i^{(k)} \right| = \sum_{i=1}^{n+1} |r_k(x_i^{(k)})| |M_i^{(k)}|$$

$$\alpha_{k+1} = |h^{(k+1)}| = \frac{\sum_{i=1}^{n+1} |M_i^{(k)}||r_k(x_i^{(k)})|}{\sum_{j=1}^{n+1} |M_j^{(k)}|}. \tag{8}$$

Now let

$$\theta_i^{(k)} = \frac{|M_i^{(k)}|}{\sum_{j=1}^{n+1} |M_j^{(k)}|}.$$

Since α_{k+1} is weighted average of $|r_k(x_i^{(k)})|$ by $\theta_i^{(k)}$'s as weights, we have $\alpha_{k+1} \geq \gamma_k$ from (2). Note that $\alpha_k \leq \gamma_k \leq \alpha_{k+1}$, and thus α_k is a non-decreasing sequence. This fact is used in the last part of the proof.

Note that there are some $n+1$ alternating points, where the approximate error values alternates, including the global extreme point by Theorem 3.1, and the approximate error values at $x_1^{(k)}, \cdots, x_{n+1}^{(k)}$ have the maximum absolute sum by the maximum absolute sum condition for new set of reference points. That is, $\sum_{i=1}^{n+1} |r_k(x_i^{(k)})|$ is larger than or equal to sum of any $n+1$ absolute error values including β_k and thus we have

$$\sum_{i=1}^{n+1} |r_k(x_i^{(k)})| \geq \beta_k. \tag{9}$$

First, we will prove that $\theta_i^{(k)}$ is larger than a constant $1-\theta > 0$ throughout the iterations. It is known that $M_i \neq 0$ for all i from the Haar condition. We firstly prove an inequality

$$x_{i+1}^{(k)} - x_i^{(k)} \geq \epsilon > 0, \quad i = 0, \cdots, n, \tag{10}$$

where ϵ does not depend on k. Assume that (10) is not true. Let $x^{(k)} = (x_1^{(k)}, \cdots, x_{n+1}^{(k)})$ be a sequence defined on D^{n+1}. Then $x^{(k)}$ has its subsequence such that $\min_i |x_{i+1}^{(k)} - x_i^{(k)}|$ converges to zero. Since D^{n+1} is a closed and bounded subset of $\mathbb{R}^{n+1}$, it is a compact set, and thus this subsequence also has its subsequence converging to a point $(x_1^*, \cdots, x_{n+1}^*)$. Since $\min_i |x_{i+1}^* - x_i^*| = 0$, there is some i such that $x_i^* = x_{i+1}^*$. Let p be the minimax generalized polynomial of f on $(x_1^*, \cdots, x_{n+1}^*)$. Since there is actually less than or equal to n points, p is the generalized approximate polynomial generated by the Lagrange interpolation, and thus

$$p(x_i^*) = f(x_i^*), \quad i = 1, \cdots, n+1.$$

It is known that $\alpha_1 > 0$ by the fact that α_k is weighted average of absolute approximation errors at the previous set of reference points. Then, there exists a number $\delta > 0$ such that whenever $|y_1 - y_2| < \epsilon$ and $y_1, y_2 \in D$, we have

$$|(p-f)(y_1) - (p-f)(y_2)| < \alpha_1$$

because $p - f$ is a continuous function on the compact set, and thus it is also uniformly continuous. Since there is a subsequence of $(x_1^{(k)}, \cdots, x_{n+1}^{(k)})$ converging to $x^{(k)} = (x_1^*, \cdots, x_{n+1}^*)$, there is some k_0 such that

$$|x_i^{(k_0)} - x_i^*| < \delta, \quad i = 1, \cdots, n+1.$$

Then,

$$|(p-f)(x_i^{(k_0)}) - (p-f)(x_i^*)| = |p(x_i^{(k_0)}) - f(x_i^{(k_0)})| < \alpha_1$$

because $(p-f)(x_i^*) = 0$. In fact, p is not the minimax generalized approximate polynomial in regard to the k-th set of reference points $\{x_1^{(k_0)}, \cdots, x_{n+1}^{(k_0)}\}$. Since α_{k+1} is the error value of the minimax generalized approximate polynomial on $\{x_1^{(k_0)}, \cdots, x_{n+1}^{(k_0)}\}$, we have

$$\alpha_{k+1} \leq \max_i |p(x_i^{(k_0)}) - f(x_i^{(k_0)})| < \alpha_1.$$

This contradicts the fact that α_k is non-decreasing sequence, and thus we have (10).

Now, we will prove that $\theta_i^{(k)}$ is larger than a constant $1-\theta$. Consider the subset D' of D^{n+1} such that for $(y_1, \cdots, y_{n+1}) \in D'$, $y_{i+1} - y_i \geq \epsilon$. We easily see that D' is a closed and bounded subset in $\mathbb{R}^{n+1}$, and thus D' is a compact set. Then, $|M_i|$, which is the same function as $|M_i^{(k)}|$ except that the inputs are y_i's instead of $x_i^{(k)}$'s, is a continuous function on D^{n+1}, so is on D', and thus there is an element at which $|M_i|$ has the mininum value on D' from the extreme value theorem. From the Haar condition in Definition 2.3, $|M_i|$ cannot be zero because y_i's are distinct and the minimum value of $|M_i|$ is not zero. Since we consider the finite number of functions $|M_i|$'s, the lower bound of all $|M_i|$'s is bigger than zero. In addition, since $\sum_{j=1}^{n+1} |M_j|$ is also a continuous function on D', there is an upper bound of $\sum_{j=1}^{n+1} |M_j|$ on D' from the extreme value theorem. This leads to the fact that θ_i's are lower-bounded beyond zero.

From $\gamma_{k+1} \geq \alpha_{k+1}$, (8), and (9), we have

$$\begin{aligned}
\gamma_{k+1} - \gamma_k &\geq \alpha_{k+1} - \gamma_k \\
&= \sum_{i=1}^{n+1} \theta_i^{(k)} (|r_k(x_i^{(k)})| - \gamma_k) && (11) \\
&\geq (1-\theta)(\beta_k - \gamma_k) && (12) \\
&\geq (1-\theta)(\beta^* - \gamma_k). && (13)
\end{aligned}$$

From (13), we have

$$\begin{aligned}
\beta^* - \gamma_{k+1} &= (\beta^* - \gamma_k) - (\gamma_{k+1} - \gamma_k) \\
&\leq (\beta^* - \gamma_k) - (1-\theta)(\beta^* - \gamma_k) \\
&= \theta(\beta^* - \gamma_k).
\end{aligned}$$

Then, we obtain the following inequality for some nonnegative B as

$$\beta^* - \gamma_k \leq B\theta^k. \tag{14}$$

From (12) and (14), we have

$$\begin{aligned}\beta_k - \beta^* &\le \beta_k - \gamma_k \\ &\le \frac{1}{1-\theta}(\gamma_{k+1} - \gamma_k) \\ &\le \frac{1}{1-\theta}(\beta^* - \gamma_k) \\ &\le \frac{1}{1-\theta} B\theta^k \\ &\le C\theta^k. \end{aligned} \tag{15}$$

From Theorem 2.10, there is a constant $\gamma > 0$ such that

$$\|p^* - f\|_\infty + \gamma\|p_k - p^*\|_\infty \le \|p_k - f\|_\infty.$$

Since $\beta_k = \|p_k - f\|_\infty$, $\beta^* = \|p^* - f\|_\infty$, and (15), we complete the proof by the following inequality

$$\begin{aligned}\|p_k - p^*\|_\infty &= \frac{\beta_k - \beta^*}{\gamma} \\ &\le A\theta^k\end{aligned}$$

for nonnegative constant A.

□

Note that (11) and Theorem 3.3 can be satisfied if we include the global extreme point to the new set of reference points in the variant of Remez algorithm, instead of the maximum absolute sum condition. Thus, this proof naturally includes the convergence proof of the original variant of the Remez algorithm. From (11), we can easily know that the maximum absolute sum condition is better for the choice of the new set of reference points than the simple inclusion of the global extreme point, in that the rate of the convergence is determined by the value of the absolute sum of errors values, which is confirmed as in Table 1 for the power basis $\{1, x, x^2, \cdots, x^d\}$.

Table 1 shows the number of iterations required to converge to the optimal minimax approximate polynomial in the variant of the Remez algorithm and the modified Remez algorithm. The initial set of reference points is selected uniformly in each interval as soon as possible since we want to observe their performances in the worst case. While the selection for new reference points is not unique for each iteration in the variant of the Remez algorithm, the modified Remez algorithm selects the new reference points uniquely for each iteration. Thus, when we analyze the variant of the Remez algorithm, we select the new reference points randomly for each iteration among the possible sets of reference points that satisfy the local extreme value condition and the alternating condition and have the global extreme point. We set the approximation parameter δ in Algorithm 2 as 2^{-40} and repeat this simulation with 100 times. It shows that

Degree	Modified Remez algorithm	Variant of Remez algorithm			
		Average	Standard deviation	Max	Min
79	**28**	60.0	9.38	82	41
99	**8**	17.1	3.34	28	11
119	**26**	53.4	8.10	79	37
139	**39**	60.3	4.71	79	48
159	**39**	72.1	9.71	98	42
179	**48**	72.3	9.72	105	53
199	**56**	80.4	7.28	94	60

Table 1: Comparison of iteration numbers between the modified Remez algorithm and the variant of Remez algorithm for $\delta = 2^{-40}$.

the modified Remez algorithm is much better to reduce the iteration number of the Remez algorithm.

Note that the number of iterations depends on the initial set of reference points. In fact, the uniformly distributed reference points are not desirable as an initial set of reference points because these reference points are far from the converged reference points. In fact, the node selection method in the modified Chebyshev algorithm in [13] is adequate for selection for the initial reference points.

3.3 Efficient Implementation of Modified Remez Algorithm

In this section, we have to consider the issues in each step of Algorithm 2 and suggest how to implement Steps 2, 3, and 4 of Algorithm 2 as follows.

Finding Approximate Polynomial A naive approach is finding coefficients of the approximate polynomial with power basis at the current reference points for the continuous function $f(x)$, i.e., we can obtain c_j's in the following equation

$$\sum_{j=0}^{d} c_j x_i^j - f(x_i) = (-1)^i E,$$

where E is also an unknown variable in this system of linear equations. However, this method suffers from the precision problem for the coefficients. It is known that as the degree of the basis of approximate polynomial increases, the coefficients usually decreases, and we have to set higher precision for the coefficients of the higher degree basis. Han et al. [13] use the Chebyshev basis for this coefficient precision problem since the coefficients of a polynomial with the Chebyshev basis usually have the almost same order. Thus, we also use the Chebyshev basis instead of the power basis. From the perspective of approximation theory, we point out the following lemma.

Lemma 3.4. *Chebyshev basis* $\{T_0(x), T_1(x), \cdots, T_d(x)\}$ *satisfies Haar condition on any closed and bounded interval.*

Proof. Note that the degree of $T_n(x)$ is exactly n, and thus $\sum_{i=0}^{d} c_i T_i(x)$ has the degree d. If $\sum_{i=0}^{d} c_i T_i(x)$ has more than d roots, it has to be the zero polynomial. □

This gives the theoretical basis for using the Chebyshev basis in Step 2 of Algorithm 2. In short, using the $d+2$ reference points, we solve the following system of $d+2$ linear equations to obtain c_j's and E as

$$\sum_{j=0}^{d} c_j T_j(x_i) - f(x_i) = (-1)^i E.$$

Obtaining Extreme Points Since we are dealing with a very small minimax approximation error, we have to obtain the extreme points as precisely as possible. Otherwise, we cannot reach the extreme point for the minimax generalized approximate polynomial precisely, and then the minimax approximation error obtained with this algorithm becomes large. Basically, in order to obtain the extreme points, we can scan $p(x) - f(x)$ with a small scan step and obtain the extreme points where the increase and decrease are exchanged. A small scan step increases the accuracy of the extreme point but causes a long scan time accordingly. To be more specific, it takes approximately 2^ℓ proportional time to find the extreme points with the accuracy of ℓ-bit. Therefore, it is necessary to devise a method to obtain high accuracy extreme points more quickly.

In order to obtain the exact point of the extreme value, we use a method of finding the points where the increase and decrease are exchanged and then finding the exact extreme point using a kind of binary search. The specific fast search algorithm is described as follows. Let $r(x) = p(x) - f(x)$ and sc be the scan step. If we can find x_0 where $\mu(x_0) r(x_0) \geq |E|$, and $(r(x_0) - r(x_0 - \mathsf{sc}))(r(x_0 + \mathsf{sc}) - r(x_0)) \leq 0$, we obtain the i-th extreme points using the following process successively ℓ times,

$$x_{i,k} = \underset{x \in \{x_{i,k-1} - \mathsf{sc}/2^k, x_{i,k-1}, x_{i,k-1} + \mathsf{sc}/2^k\}}{\arg\max} |r(x)|, k = 1, 2, \cdots, \ell.$$

Then, we obtain the extreme point with $O(\log(\mathsf{sc}) + \ell)$-bit precision. The reason is as follows. If sc is small enough, $|r(x)|$ behaves similarly to $a(x - x^*)^2 + b$ for some $a > 0$ and b near the point x^*. This behavior ensures that if $|x_1 - x^*| < |x_2 - x^*|$ near x^*, $|r(x_1)| > |r(x_2)|$ holds, and the converse also holds. It is easy to check that when sc is determined, it is sufficient if $r(x)$ behaves similarly to $a(x - x^*)^2 + b$ in the interval $[x^* - \mathsf{sc}, x^* + \mathsf{sc}]$, but sc needs not to be too small value. Then, we can find the extreme point with arbitrary precision with linear time to precision ℓ. In summary, we propose that the ℓ-bit precision of the extreme points can be obtained by the linear time of ℓ instead of 2^ℓ.

Obtaining New Reference Points When we find the new reference points satisfying the local extreme value condition, the alternating condition, and maximum absolute sum condition, there is a naive approach: among local extreme points which satisfy the local extreme value condition, find all $n+1$ points satisfying the alternating condition and choose the $n+1$ points which have the maximum absolute sum value. If we have m local extreme points, we have to investigate $\binom{m}{n+1}$ points, and this value is too large, and thus it makes this algorithm impractical. Thus, we have to find a more efficient method than this naive approach.

We propose very efficient and provable algorithm for finding the new reference points. The proposed algorithm always gives the $n+1$ points satisfying the three criteria. It can be considered as an elimination method, in that we eliminate some elements for each iteration in the proposed algorithm until we obtain $n+1$ points. It is clear that as long as $m > n+1$, we can find at least one element which may not be included in the new reference points. This proposed algorithm is given in Algorithm 3. Algorithm 3 takes $O(m \log m)$ running time, which is a quasi-linear time algorithm. To understand the last part of Algorithm 3, the example can be given that if the extreme point x_2 is removed, $T = \{|r(x_1)| + |r(x_2)|, |r(x_2)| + |r(x_3)|, |r(x_3)| + |r(x_4)|, \cdots\}$ is changed to $T = \{|r(x_1)| + |r(x_3)|, |r(x_3)| + |r(x_4)|, \cdots\}$.

It is assumed that whenever we remove an element in the ordered set B in Algorithm 3, the remaining points remain sorted and indices are relabeled in increasing order. When we compare the values to remove some extreme points, there are the cases that the compared values are equal or the smallest element is more than one. In such cases, we randomly remove one of these elements.

Theorem 3.5. *Algorithm 3 always returns the $n+1$ points satisfying alternating condition and maximum absolute sum condition.*

Proof. Let $B_{\mathsf{init}} = \{t'_1, t'_2, \cdots, t'_m\}$ be the initial elements in B , and let t'_ℓ be an element removed in the first while statement. We first show that each element removed in the first while statement in Algorithm 3 is not included in the new reference; that is, if the subset A of B_{init} having $n+1$ elements satisfies alternating condition and contain this removed element, there is another subset A' of B having $n+1$ elements satisfying alternating condition, not containing the removed element, and having absolute sum larger than or equal to A. Since it is removed in the first while statement, there is an element $t'_{\ell'}$ such that $|r(t'_{\ell'})| \geq |r(t'_\ell)|$, $\mu(t'_{\ell'}) = \mu(t'_\ell)$, and $|\ell' - \ell| = 1$. Clearly, A cannot contain $t'_{\ell'}$, because A satisfies the alternating condition. Let A' be the same set as A except that t'_ℓ is replaced with $t'_{\ell'}$. Then A' also satisfies alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

We now observe that at the end of the first while statement, B itself satisfies the alternating condition. Then we now have to prove that elements removed in the second while statement in Algorithm 3 are not included in the new reference points. In other words, if the subset A of B_{init} having $n+1$ elements satisfies the alternating condition and contains these removed elements, there is another

Algorithm 3: NewReference

Input : An increasing ordered set of extreme points $B = \{t_1, t_2, \cdots, t_m\}$ with $m \geq n+1$, and number of basis n.

Output: $n+1$ points in B satisfying alternating condition and maximum absolute sum condition.

$i \leftarrow 1$
while t_i is not the last element of B **do**
  if $\mu(t_i)\mu(t_{i+1}) = 1$ **then**
    Remove from B one of two points t_i, t_{i+1} having less value among $\{|r(t_i)|, |r(t_{i+1})|\}$.
  else
    $i \leftarrow i+1$
  end
end
if $|B| > n+2$ **then**
  Calculate all $|r(t_i)| + |r(t_{i+1})|$ for $i = 1, \cdots, |B|-1$ and sort and store this values into the array T.
while $|B| > n+1$ **do**
  if $|B| = n+2$ **then**
    Remove from B one of two points $t_1, t_{|B|}$ having less value among $\{|r(t_1)|, |r(t_{|B|})|\}$.
  else if $|B| = n+3$ **then**
    Insert $|r(t_1)| + |r(t_{|B|})|$ into T and sort T. Remove from B the two element having the smallest value in T.
  else
    if t_1 or $t_{|B|}$ is included in the smallest element in T **then**
      Remove from B only t_1 or $t_{|B|}$.
    else
      Remove from B the two elements having the smallest element in T.
    end
    Remove from T all elements related to the removed extreme points, and insert into T the sum of absolute error values of the two newly adjacent extreme points.
  end
end

subset A' of B having $n+1$ elements satisfying alternating condition, not containing these removed elements, and having absolute sum larger than or equal to A. Let t'_ℓ be an element removed in the second while statement be.

Then, there are three cases: at the time of removal of t'_ℓ, the remaining set B can have $n+2$, $n+3$, or larger than $n+3$ elements as in Algorithm 3. We consider each case separately. By the induction argument, we can assume that the remaining set B in each iteration has $n+1$ points that satisfy the alternating condition and have the maximum absolute sum among all possible $n+1$ points in B_{init}. In other words, we can assume that if we have $n+1$ points in B_{init} that satisfy the alternating condition and contain at least one of the removed elements before that time, there are $n+1$ points in the remaining set B at that time such that they satisfy the alternating condition and have absolute sum larger than or equal to the previous ones. This inductive assumption makes us consider only the remaining set B at that iteration instead of all B_{init} in the proof.

i) Case of $n+2$: If the remaining set B has $n+2$ elements at the time of removal of t'_ℓ, elements in B at that time are labeled as $t_1, t_2, \cdots, t_{n+2}$, and t'_ℓ is labeled as t_1 or t_{n+2}. Then, $|r(t)|$ at one of the two points t_1 and t_{n+2}, which is not t'_ℓ has the value of $|r(x)|$ larger than or equal to the value at t'_ℓ. We denote this element $t'_{\ell'}$. If we have a subset A of $n+1$ points in the remaining set B that satisfy alternating condition and contain t'_ℓ, $t'_{\ell'}$ must not be in these $n+1$ points due to alternating condition. Let A' be the same set as A except that t'_ℓ is replaced with $t'_{\ell'}$. Then, A' also satisfies the alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

ii) Case of $n+3$: If the remaining set B has $n+3$ elements at the time of removal of t'_ℓ, the elements in B at that time are labeled as $t_1, t_2, \cdots, t_{n+3}$, and we have to remove two elements. Then, there must be a different element t'_p which is also removed at the time of removal of t'_ℓ. $\{t'_\ell, t'_p\}$ can be $\{t_i, t_{i+1}\}$ for some i or $\{t_1, t_{n+3}\}$ as in Algorithm 3. Since all of the subsets of B having $n+1$ elements that satisfy the alternating condition are the cases of $B\backslash\{t_i, t_{i+1}\}$ for some i or $B\backslash\{t_1, t_{n+3}\}$, one subset of B with the alternating condition that has the maximum absolute sum has to be $B\backslash t'_\ell, t'_p$. Therefore, we can obtain the resulting subset by removing these two elements.

iii) Case of larger than $n+3$: If the remaining set B has elements larger than $n+3$ elements at the time of removal of t'_ℓ, the elements in B at that time are labeled as $t_1, t_2, \cdots, t_j$, where $j > n+3$. Then, there are two cases: One is that t'_ℓ is labeled as t_1 or t_j, and the other is not the first case.

iii)-1 If t'_ℓ is labeled as t_1 or t_j, let t'_p be the adjacent element in B. If the subset A in B that satisfies the alternating condition and contains t'_ℓ also contains t'_p, there is at least one pair of adjacent elements $t'_{\ell'}$ and $t'_{p'}$ in B that is not contained in A, since A satisfies the alternating condition and more than three elements are removed from B. Note that $|r(t'_{\ell'})| + |r(t'_{p'})| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set as A except that t'_ℓ and t'_p are replaced with $t'_{\ell'}$ and $t'_{p'}$. A' also satisfies

alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.
If A does not contain t'_p, the adjacent element of t'_p which is not t'_ℓ cannot be contained in A, since A satisfies the alternating condition. Let $t'_{\ell'}$ be the adjacent element of t'_p. Note that $|r(t'_{\ell'})| + |r(t'_p)| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set with A except that t'_ℓ is replaced with $t'_{\ell'}$. A' also satisfies the alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

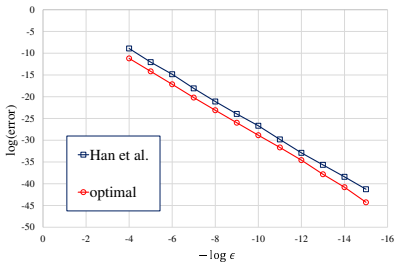
iii)-2 If t'_ℓ is not labeled as t_1 or t_j, the adjacent element t'_p of t'_ℓ in B which $|r(t'_\ell)| + |r(t'_p)|$ is the smallest value in T cannot be t_1 or t_j. If this is the case, t'_ℓ cannot be removed but t'_p is removed in that iteration. If the alternated subset A in B that contains t'_ℓ also contain t'_p, there is at least one pair of adjacent elements $t'_{\ell'}$ and $t'_{p'}$ in B that is not contained in A. Note also that $|r(t'_{\ell'})| + |r(t'_{p'})| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set as A except that t'_ℓ and t'_p are replaced with $t'_{\ell'}$ and $t'_{p'}$. A' also satisfies the alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.
If A does not contain t'_p, there is the adjacent element of t'_p which is not t'_ℓ, since t'_p is not t_1 or t_j. Let $t'_{\ell'}$ be adjacent element of t'_p. Then, $t'_{\ell'}$ cannot be contained in A, since A satisfies the alternating condition. Note that $|r(t'_{\ell'})| + |r(t'_p)| \geq |r(t'_\ell)| + |r(t'_p)|$. Let A' be the same set as A except that t'_ℓ is replaced with $t'_{\ell'}$. A' also satisfies alternating condition, does not contain t'_ℓ, and has absolute sum larger than or equal to that of A.

Thus, we prove the theorem. □

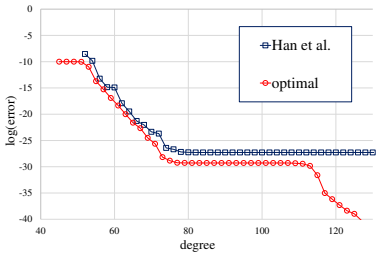
4 Numerical Analysis of Approximation Error

In this section, we will use the Chebyshev basis $\{T_0(x), T_1(x), \cdots, T_d(x)\}$ and the minimax approximate polynomial. We show and discuss the simulation results for the minimax approximation errors of the normalized modular reduction function and the scaled cosine function. The simulation results are all theoretically optimal minimax approximation error of the corresponding function.

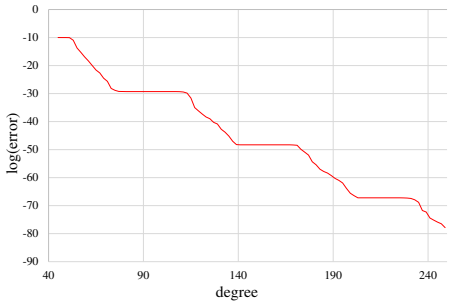
We now compare the simulation results in [13], which is the state-of-the-art method in the CKKS homomorphic bootstrapping. Han et al. mostly deal with the minimax approximation error of their approximate polynomial for the cosine function. Even if the cosine function is very close to the normalized modular reduction function near the integers, there exists a fundamental error between the cosine function and the normalized modular reduction function. Since our purpose is to approximate the normalized modular reduction function by the polynomial, we convert their result to the minimax approximation error between their polynomials and the normalized modular reduction function, instead of the cosine function. However, they used the approximation of $\mathsf{normod}(x)$ by $\frac{1}{2\pi}\cos\left(2\pi\left(x-\frac{1}{4}\right)\right)$.



(a) For various half-width of approximate interval and the degree of approximation polynomial 76



(b) For various degree of approximation polynomials and the half-width of approximation interval 2^{-10}



(c) Several flat error regions in modular reduction function

Fig. 1: Minimax approximation errors of normalized modular reduction function.

Fig. 1(a) shows the optimal minimax approximation error derived by the proposed modified Remez algorithm and the approximation error of the approximate polynomials in [13] as the half-width of intervals varies for the minimax approximate polynomial of degree 76. Figs. 1(b) and 1(c) also show the optimal minimax approximation error as the degree of the minimax approximate polynomial varies for half-width 2^{-10}. Fig. 1(a) shows that the proposed optimal minimax approximation algorithm is better than that in [13] and the log-scale difference is almost the same as the half-width varies. Note that the degrees of the proposed minimax approximate polynomial in [13] are almost all even, but the degrees of the optimal minimax approximate polynomial are all odd, which will be dealt in Section 5. For this reason, we compare the optimal minimax approximate polynomial of degree 75 with the minimax approximate polynomial in [13] of degree 76 in Fig. 1(a).

In Fig. 1(c), if the degree of the minimax approximate polynomial is less than or equal to 74, the optimal minimax approximation error derived by the proposed modified Remez algorithm is very close to that in [13]. It also shows that if the degree of minimax approximate polynomial is larger than 74, it reaches the fundamental flat error between the cosine function and the normalized modular reduction function. Although they insisted that the minimax approximation error continuously decreases even after the degree of 74, the meaningful minimax approximation error does not decrease after degree 74. If we need precision higher than this fundamental flat error, we cannot use the cosine function. In this case, this precision can be interpreted as significant figures of about five in the decimal number, which is not that high precision.

On the other hand, we observe that the minimax approximation error does not decrease after the degree of 73 until the degree of 113. Note that this flat error phenomenon is not caused by the limitation of the proposed algorithm, but it is the fundamental property of the normalized modular reduction function. However, if we evaluate the minimax approximate polynomials with degrees higher than 117, it is possible for the proposed optimal minimax approximation algorithm to obtain the higher precision of the minimax approximation errors. Thus, we cannot use the cosine function as an approximation of the normalized modular reduction function. Note that there are several flat regions in Fig. 1(c). We find that these flat regions are closely related to the minimax approximation error of inverse sine function by the numerical analysis as in the next section.

Fig. 2 shows the minimax approximation error of scaled cosine function with the scaling factor 1. While the optimal minimax approximation error of the proposed modified Remez algorithm continuously decreases as the degree of optimal minimax approximate polynomial increases, the minimax approximation error of the modified Chebyshev algorithm in [13] gives step-wise shape. This is because the modified Chebyshev algorithm cannot obtain the approximate polynomials of all degrees. The minimax approximation errors of the proposed algorithm for the scaled cosine function is slightly better than that of the modified Chebyshev algorithm. Note that there is no flat region in the minimax approximation error of the scaled cosine function, and the degree of the minimax approximate polynomial of the scaled cosine function is significantly smaller than that of the normalized modular reduction function of Fig. 1(c).

5 Composite Function Approximation of Modular Reduction Function

It is shown that the cosine approximation in [13] has the fundamental error from the modular reduction function as in Fig. 1(b). Thus, we cannot use the cosine approximation if some applications require approximation error smaller than this fundamental flat error. The necessity of the approximation error smaller than the fundamental flat error is shown in [14]. Since the fundamental flat error occurs from approximating the modular reduction function by the cosine function, we

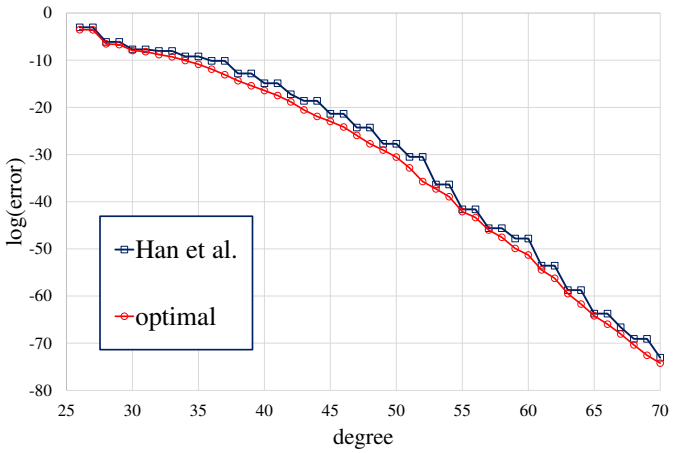


Fig. 2: Minimax approximation errors of scaled cosine function with scaling factor 1.

know that the fundamental flat error is the order of $O(\epsilon^3)$ with some constant when $\left|\frac{m}{q}\right| \leq \epsilon$ by the following equation

$$\epsilon - \frac{1}{2\pi}\sin(2\pi\epsilon) = \frac{1}{2\pi} \cdot \frac{1}{3!}(2\pi\epsilon)^3 + O(\epsilon^5).$$

Then, we can use roughly $2\log(1/\epsilon)$ bits of significant figures if we reach the fundamental flat error by some approximate polynomial. In other words, if we want to use ℓ bits of significant figures, we have to use at least roughly $\frac{\ell}{2} + \log m$ bit length of modulus in level 0. This additional $\frac{\ell}{2}$ bits of modulus does not have any information, and it causes some inefficiency for the running time and the memory uses. Furthermore, the size of the scaling factor in the bootstrapping of the CKKS scheme should be similar to the modulus in level 0 [4], and thus the number of possible multiplications after the bootstrapping becomes somewhat small. Thus, it is desirable to reduce the size of the additional bits of modulus while the size of the significant figures of the message remains unchanged. It can be accomplished by approximating the modular reduction function with approximation error smaller than the fundamental flat error.

Since the cosine approximation can reduce the number of non-scalar multiplications by using the double-angle formula of the cosine function, it is preferable to use the cosine approximation with double-angle formula even in the cases that the smaller approximation error is required. In this section, we propose the composite function approximation to approximate the modular reduction function with arbitrarily small approximation errors.

In [7], they use the composition method to perform the homomorphic comparison operation. Instead of evaluating the minimax approximate polynomial of the sign function, they use the composition method of several polynomials of small degree. Although the degree of the resulting approximate polynomials in the composition method is usually higher than the minimax approximate polynomial having the same maximum error, the required number of operations in evaluating the approximate polynomial of the composite method is smaller than that in evaluating the minimax approximate polynomial. Note that the cosine approximation with double-angle formula can also be considered as a composition. It can be a natural question if we can find some approximate polynomials for the modular reduction function by the general composition method so that we reduce the number of non-scalar multiplications for the bootstrapping of the CKKS scheme.

It is easy to check that if we have two functions f ad g for $0 < \epsilon < \frac{1}{4}$ as

$$f : \bigcup_{k=-\infty}^{\infty} [2\pi(k-\epsilon), 2\pi(k+\epsilon)] \to [-\sin 2\pi\epsilon, \sin 2\pi\epsilon], \quad f(x) = \sin x$$

$$g : [-\sin 2\pi\epsilon, \sin 2\pi\epsilon] \to [-2\pi\epsilon, 2\pi\epsilon], \quad g(x) = \arcsin x$$

then the following equation holds as

$$x - 2\pi \cdot \mathsf{round}\left(\frac{x}{2\pi}\right) = (g \circ f)(x), \quad x \in \bigcup_{k=-\infty}^{\infty} [2\pi(k-\epsilon), 2\pi(k+\epsilon)].$$

If we substitute $t = \frac{x}{2\pi}$, then we have

$$\mathsf{normod}(t) = \frac{1}{2\pi}(g \circ f)(2\pi t), \quad t \in \bigcup_{k=-\infty}^{\infty} [k-\epsilon, k+\epsilon]. \tag{16}$$

We will show that if we approximate both f and g more tightly by some approximate polynomials, we can approximate the modular reduction function with any small approximate error by the composition of f and g. Note that $g(x)$ can be approximated very well with some approximate polynomials of small degree, even with the linear polynomial for small ϵ. Indeed, the cosine approximation with double-angle formula in [13] can be regarded as the special case of the proposed composite function approximation, in that they approximate $g(x)$ with x, that is, the identity function. Note that the cosine function in [13] is merely a parallel shift of the sine function. Thus, it is said that they approximate the sine function instead of the cosine function.

Since the inverse sine function is odd, we will use the property of the minimax approximate polynomials of an odd function. From the following lemma, we are convinced that the minimax approximate polynomials of an odd (resp. even) function are odd (resp. even) functions.

Lemma 5.1. *Let D be a union of finite numbers of intervals that has symmetry to origin, and let $f : D \to \mathbb{R}$ be an odd (resp. even) function. If p is the minimax*

approximate polynomial of degree d for f, the coefficients of even (resp. odd) degree terms of p is always zero.

Proof. For an odd function f, let $q(x) = -p(-x)$. Then, clearly, $q(x)$ is also a polynomial of degree less than or equal to d, and $q(x)$ has the same minimax approximation error as $p(x)$, since f is an odd function. Thus, $q(x)$ is also the minimax approximate polynomial of f. From Theorem 2.6, we have $p(x) = q(x)$, and thus $p(x) = -p(-x)$, which shows that $p(x)$ is an odd function. In other words, coefficients of even degree in $p(x)$ are all zero.

For an even function f, let $q(x) = p(-x)$. Similarly, $p(x) = q(x) = p(-x)$, which shows that $p(x)$ is an even function. Thus, the coefficients of odd degree in $p(x)$ are all zero. □

It is noted that the identity function x is not the minimax approximate linear polynomial for the inverse sine function. Since the minimax approximate polynomial of the inverse sine function among the polynomials of degree less than or equal to two is a linear polynomial by Lemma 5.1, there are four global extreme points of error function which alternates in sign by the Chebyshev alternating theorem. However, x has only two global extreme points of error function at both ends of the interval. Of course, we can obtain the minimax approximate linear polynomial by the Remez algorithm. We can obtain the simple closed-form of the minimax approximate linear polynomial with very small approximation error as in the following theorem. The proof of Theorem 5.2 is included in Appendix A.

Theorem 5.2. *The minimax approximate linear polynomial of* $\arcsin x$ *in* $[-\sin\epsilon, \sin\epsilon]$ *is* $c_{\mathsf{min}}x$ *such that*

$$c_{\mathsf{min}} = 1 + \frac{\epsilon^2}{8} + O(\epsilon^4).$$

Further, we have

$$\frac{\|(1+\epsilon^2/8)x - \arcsin x\|_\infty}{\|x - \arcsin x\|_\infty} = \frac{\|c_{\mathsf{min}}x - \arcsin x\|_\infty}{\|x - \arcsin x\|_\infty} + O(\epsilon^2) = \frac{1}{4} + O(\epsilon^2),$$

where the domain of all functions is $[-\sin\epsilon, \sin\epsilon]$.

Let $f(x) = \sin x$ and $g(x) \approx \left(1 + \frac{\pi^2}{2}\epsilon^2\right)x$. From Theorem 5.2 and some scaling factor, the normalized modular reduction function in (16) can be rewritten as

$$\mathsf{normod}(t) \approx \frac{1}{2\pi}\left(1 + \frac{\pi^2}{2}\epsilon^2\right)\sin(2\pi t) = \frac{1}{2\pi}g_0(\sin 2\pi t),$$

where $g_0(x) = \left(1 + \frac{\pi^2}{2}\epsilon^2\right)x$, which is the minimax linear polynomial of $g(x)$. This is just multiplication of constant $1 + \frac{\pi^2}{2}\epsilon^2$ at the original approximation formula. Thus, this allows a reduction of the fundamental limitation of the approximation error for the cosine approximation in [13] by 1/4 at no cost. This means that we can obtain two more bit precision by only adjusting the multiplicative factor. We denote this minimax approximation flat error as δ_0. The

approximation error by the modification of the proportional constant is shown in Fig. 3. While the fundamental flat error of the cosine approximation in Fig. 3 is higher than the first flat error of the normod function by two in logarithm scale, the fundamental flat error of the composite function approximation with linear polynomial is almost the same as the first flat error of the normod function.

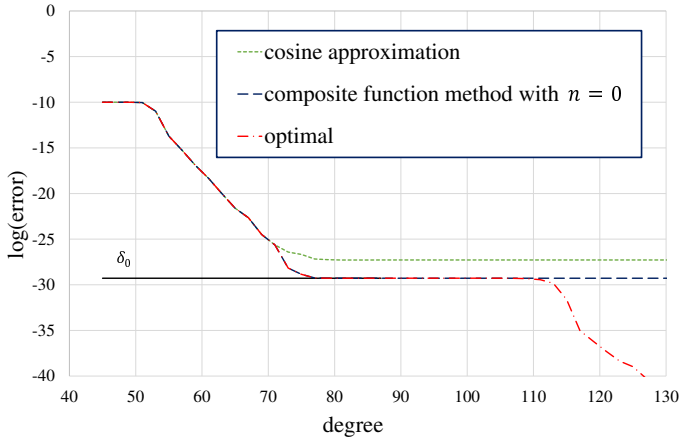


Fig. 3: Approximation errors by the cosine approximation and the approximation errors by the composite function approximation with $n = 0$.

Next, the smaller minimax approximation error can be obtained by approximating inverse sine function as the minimax approximate polynomial of degree more than one. The minimax approximate polynomial for the inverse sine function can be obtained by the original Remez algorithm because the approximation domain is one interval. From Theorem 5.1, we know that there are no terms of even degree for the optimal minimax approximate polynomial for the inverse sine function. Thus, we approximate $g(x)$ with the following formula to obtain the minimax approximation error smaller than the fundamental flat error as

$$g(x) \approx c_1 x + c_3 x^3 = g_1(x),$$

where c_1 and c_3 are obtained by the original Remez algorithm. We denote the minimax approximate error of this approximation for $g(x)$ as δ_1. Then, we approximate $\mathsf{normod}(t)$ as

$$\mathsf{normod}(t) \approx \frac{1}{2\pi}\left(c_1 \sin 2\pi t + c_3 \sin^3 2\pi t\right).$$

To be more specific, we firstly approximate the sine or cosine function with some minimax approximate polynomial having the corresponding minimax approximate error, and then evaluate $g_1(x)$ successively. The double-angle formula can be used in the approximation of sine or cosine function. This requires two more non-scalar multiplications and two more depth after the sine or cosine function is approximated, and this approximation makes it possible to obtain the minimax approximation error for normod function between δ_0 and δ_1.

Let $g_n(x)$ be the optimal minimax approximate polynomial of degree $2n+1$ for $g(x)$, which has only odd degree terms, and δ_n be the minimax approximation error of $g_n(x)$. Then, if we want to obtain the minimax approximate error between δ_n and δ_{n+1}, we have to use the following approximation formula

$$\mathsf{normod}(t) \approx \frac{1}{2\pi} g_n(\sin 2\pi t).$$

From Theorem 2.2, δ_n goes to zero as n increases. Thus, we can obtain arbitrarily small minimax approximation error by this composite function approximation. In the next section, we will deal with the evaluation method of polynomials with odd degree terms such as $g_n(x)$.

With the numerical analysis, it can be checked that the minimax approximation errors of inverse sine function by its minimax approximate polynomials of degree three and five are almost exactly $\frac{9}{4}$ and $\frac{25}{4}$ times of the minimax approximation error of the second and third flat regions of normod function, respectively, as in Fig. 4. We conjecture that the minimax approximation error of inverse sine function is closely related to the flat regions of Fig. 4. We put the mathematical relation between the minimax approximation error of inverse sine function and that of the normalized modular reduction function as the future work. Since we will propose the method which can reduce the number of operations in the direct approximation in the next section, the number of operations of the composite function approximation and the direct approximation will be compared in the next section.

6 Optimization of Evaluation of Approximate Polynomial with Odd Function Property

The composite function approximation can reduce the number of multiplications in the evaluation of approximate polynomial, but this method usually consumes additional depths. However, low depth is more important than the number of operations in many situations. In these cases, it is desirable to evaluate the optimal minimax approximate polynomial directly.

We obtain the tightest trade-off between the degree of the approximate polynomial and the minimax approximation error. However, the degree of the approximate polynomial is not a unique consideration in determining the running time of modular reduction function. The algorithm evaluating an approximate polynomial can also be taken into consideration and finding a special form of the approximate polynomial with the minimax approximation error which can

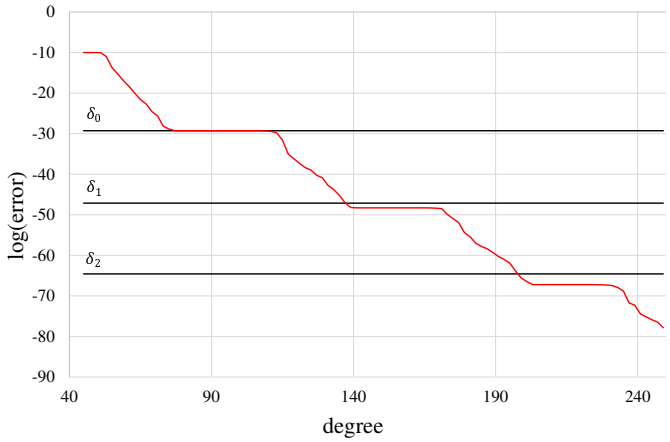


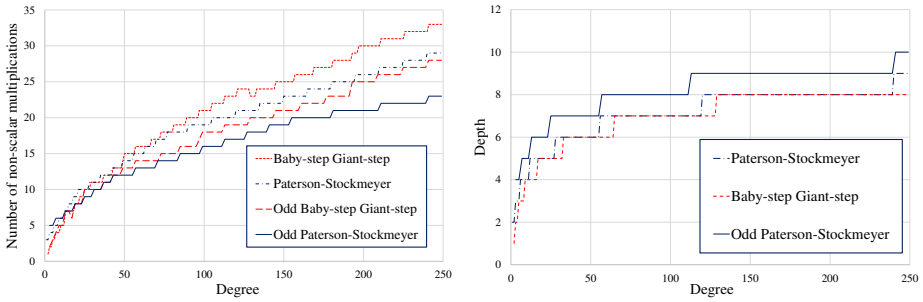
Fig. 4: Approximation errors of optimal minimax approximate polynomial and approximate errors of inverse sine function.

be evaluated faster than other polynomials is a desirable approach of reducing the running time. There are two fast evaluation algorithms for the minimax approximate polynomials, that is, the Paterson-Stockmeyer algorithm [2, 17] and Baby-step Giant-step algorithm [13]. The number of non-scalar multiplications in the Paterson-Stockmeyer algorithm is usually smaller than that in the Baby-step Giant-step algorithm.

The second consideration for the fast evaluation is to find a special form of polynomials of odd function. We prove that the minimax approximate polynomial of an odd function has to be an odd function in Lemma 5.1. The odd function property will be used to optimize the running time of evaluation.

In the bootstrapping of the CKKS scheme, we have to approximate the normalized modular reduction function on near integers. This function and the domain satisfy the conditions in Lemma 5.1. This fact can be used in terms of two aspects. First, when we obtain the coefficients of minimax approximate polynomial, we just consider the coefficients of odd degree terms. This can reduce the running time of obtaining the minimax approximate polynomial. In fact, the running time of the Remez algorithm to obtain the minimax approximate polynomial is not sensitive in the homomorphic scheme, because the minimax approximate polynomial is obtained not on ciphertext region, but plaintext region, and the minimax approximate polynomial may be obtained just once before performing homomorphic operations.

Second, the running time of evaluation of the minimax approximate polynomial can be reduced from the odd function property. This reduction of the



(a) The number of non-scalar multiplications of various evaluation algorithms

(b) The depth of various evaluation algorithms

Fig. 5: The performance of evaluation algorithms for the odd degree versions of approximate polynomials.

running time is far more important than the first aspect, because the evaluation is performed in the ciphertext region, and has to be performed whenever the bootstrapping operation is performed. Now, we propose two methods of reducing the number of non-scalar multiplications for the evaluation of approximate polynomials in case of odd function. The first method consumes one more depth compared to the original Paterson-Stockmeyer algorithm, and the second method does not consume additional depth, ensuring the optimal depth.

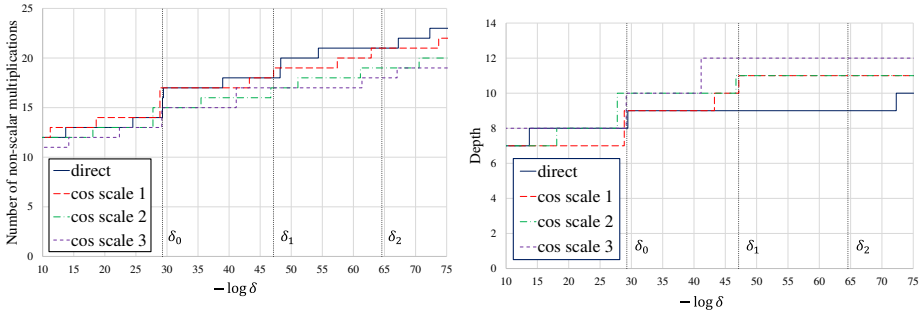
The first method uses either the Paterson-Stockmeyer algorithm to reduce the number of non-scalar multiplications. We propose to reduce the number of non-scalar multiplications by 30% with the odd function property of the minimax approximate polynomial of the modular reduction function.

Using odd function property, we can transform the approximate polynomial $f(x)$ of degree $(2n+1)$ as

$$f(x) = xg(x^2),$$

where $g(x)$ is a polynomial of degree n whose coefficient of ith term is equal to that of $(2i+1)$th term of $f(x)$. This allows evaluating only n-th degree polynomial $g(x)$ with two additional operations: squaring x for the first time and multiplication with x at the last time. The number of non-scalar multiplications is reduced from $\sqrt{2n} + O(\log n)$ to $\sqrt{n} + O(\log n)$, which means that we can reduce it by 30%. The concrete comparison is shown in Fig. 5. Note that the proposed method consumes one more depth compared to the original Paterson-Stockmeyer algorithm. We prefer to use the proposed method for polynomials with high degrees because we can reduce many non-scalar multiplications at the cost of one depth.

The second method uses the Baby-step Giant-step algorithm. Note that this method does not consume any additional depth, where we may prefer the Baby-step Giant-step algorithm instead of the Paterson-Stockmeyer algorithm. Before modifying the Baby-step Giant-step method, we introduce the following theorem.



(a) The number of non-scalar multiplications

(b) The depth

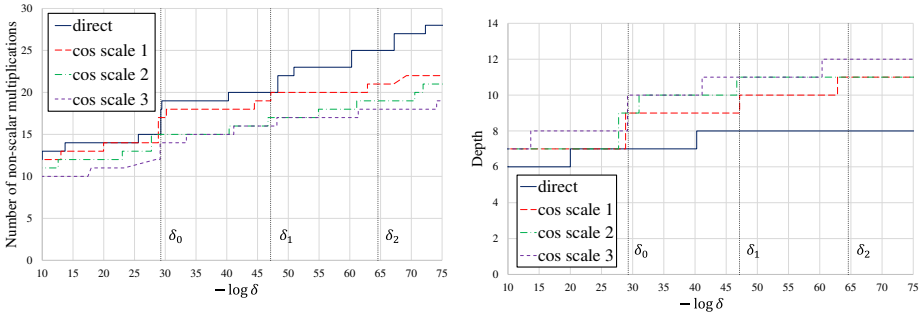
Fig. 6: The performance of the composite function approximation and the direct approximation when minimizing the non-scalar multiplications.

Theorem 6.1. *Let f be an polynomial with odd degree terms, and let g be an polynomial with even degree terms. If $f = gq + r$, where* $\deg r < \deg g$*, then q and r are both polynomials of odd degree terms.*

Proof. Since $0 = f(x) + f(-x) = (g(x)q(x) + r(x)) + (g(-x)q(-x) + r(-x)) = g(x)(q(x)+q(-x))+(r(x)+r(-x))$, we have $r(x)+r(-x) = -g(x)(q(x)+q(-x))$. Then, $g(x)|(r(x)+r(-x))$. Since the degree of $g(x)$ is larger than that of $r(x)+r(-x)$, $r(x)+r(-x) = 0$ and thus, we have $r(-x) = -r(x)$ and $q(-x) = -q(x)$, which proves the theorem. □

The Baby-step Giant-step method recursively uses the quotient and remainder polynomial when we divide the approximate polynomial by the even-degree Chebyshev polynomials. Since the approximate polynomial is an odd function and the even-degree Chebyshev polynomials are even functions, we know that the quotient and remainder polynomials are both odd functions. After a certain number of successive divisions, all polynomials which are evaluated with baby-step Chebyshev polynomial are odd functions. Then, we do not use the even-degree Chebyshev polynomials in Babysteps, and thus we do not have to evaluate the even degree Chebyshev polynomials. This directly leads to the reduction of non-scalar multiplications. Algorithm 4 describes the modification of the Baby-step Giant-step algorithm in case of the approximate polynomials with odd degree terms. Note that this modification does not consume additional depth.

We also note that in the original Baby-step Giant-step algorithm in [13], the length of baby steps is restricted to only the power of two. However, this restriction can be removed with the remaining correctness of the algorithm, and thus the length of baby steps can be any positive integers. This allows optimizing the number of non-scalar multiplications more finely without consuming the additional depths. In the odd function case, the length of baby steps is restricted



(a) The number of non-scalar multiplications

(b) The depth

Fig. 7: The performance of the composite function approximation and the direct approximation when minimizing the non-scalar multiplications with minimal depth.

to be positive even integers, in that the giant step Chebyshev polynomials have to be even functions. This fact is also reflected in the Algorithm 4.

Furthermore, we do not need to evaluate the highest giant-step polynomial in some cases. Let d be the degree of the input polynomial $p(x)$, k be the length of baby steps, and m be an integer such that $2^m \cdot k > d \geq 2^{m-1} \cdot k$. If $m > 1$ and $d \leq 3 \cdot 2^{m-2} \cdot k$, the degree of the quotient polynomial $q(x)$ of $p(x)$ divided by $T_{2^{m-1}k}(x)$ is less than or equal to $2^{m-2}k$. Then, we do not need the non-scalar multiplication with $T_{2^{m-2}k}(x)$ for evaluating the quotient polynomial. The non-scalar multiplication with $T_{2^{m-2}k}(x)$ is used only in the evaluation of the remainder polynomial $r(x)$ of $p(x)$ divided by $T_{2^{m-1}k}(x)$. Instead, if we divide $p(x)$ by $T_{2^{m-2}k}(x)$, the degree of the quotient polynomial $q(x)$ is less than or equal to $2^{m-1}k$ and the degree of the remainder polynomial $r(x)$ is less than or equal to $2^{m-2}k$. That is, for $m > 1$ and $d \leq 3 \cdot 2^{m-2} \cdot k$,

$$p(x) = T_{2^{m-1}k}(x)q(x) + r(x)$$

is changed to

$$p(x) = T_{2^{m-2}k}(x)q(x) + r(x).$$

Then, we divide $q(x)$ by $T_{2^{m-2}k}(x)$ again. With this change, we remove the necessity of $T_{2^{m-1}k}(x)$ and leave the depth unchanged, and thus we can reduce one non-scalar multiplication. Actually, this optimization can also be applied to the original Baby-Step Giant-Step algorithm for the general polynomial.

If the degree of input polynomial is d and the length of baby steps is k, the number of non-scalar multiplications is $k + m + \lceil \frac{d}{k} \rceil - 4$ when $m > 1$ and $d \leq 3 \cdot 2^{m-2} \cdot k$, and $k + m + \lceil \frac{d}{k} \rceil - 3$ otherwise, and the depth is $\lceil \log k \rceil + m$ in the original Baby-step Giant-step algorithm [13,14]. In the odd function case, the number of non-scalar multiplications is $\lceil \log k \rceil + \frac{k}{2} + m + \lceil \frac{d}{k} \rceil - 4$ when $m > 1$ and $d \leq 3 \cdot 2^{m-2} \cdot k$, and $\lceil \log k \rceil + \frac{k}{2} + m + \lceil \frac{d}{k} \rceil - 3$ otherwise, and the depth is also $\lceil \log k \rceil + m$.

The result of these two optimization methods for an odd function is shown in Fig. 5. The odd variant of the Paterson-Stockmeyer algorithm reduces the number of the non-scalar multiplications by 30% compared to the original Paterson-Stockmeyer algorithm but consumes an additional one depth. The odd degree variant of Baby-step Giant-step reduces the number of the non-scalar multiplications by 22% compared to the original Baby-step Giant-step algorithm, without consuming any additional depth.

Algorithm 4: OddBabystepGiantstep

Input : A polynomial of degree d, $p = \sum_{i=0}^{d} c_i T_i$, and a real number t.
Output: The value of $p(t)$.

$k, m \leftarrow$ the integer such that $2^m \cdot k > d \geq 2^{m-1} \cdot k$ and $k \approx \sqrt{d}$ which is even.
Evaluate $T_2(t), T_4(t), \cdots, T_{2\lfloor \log k \rfloor}(t)$ using $T_{2i}(t) = 2T_i(t)^2 - 1$.
Evaluate $T_3(t), T_5(t), \cdots, T_{k-1}(t)$ inductively.
Evaluate $T_k(t)$ inductively and evaluate $T_{2k}(t), T_{4k}(t), \cdots, T_{2^{m-2}k}(t)$ using $T_{2i}(t) = 2T_i(t)^2 - 1$.
if $m > 1$ *or* $d \leq 3 \cdot 2^{m-2} \cdot k$ **then**
  Find quotient $q(t)$ and remainder $r(t)$ of $p(t)$ divided by $T_{2^{m-2}k}$.
else
  Evaluate $T_{2^{m-1}k}(t)$.
  Find quotient $q(t)$ and remainder $r(t)$ of $p(t)$ divided by $T_{2^{m-1}k}$.
end
if $\deg q < k$ **then**
  Evaluate $q(t)$ with $T_1(t), T_3(t), T_5(t), \cdots, T_{k-1}(t)$.
else
  Evaluate $q(t)$ recursively.
end
if $\deg r < k$ **then**
  Evaluate $r(t)$ with $T_1(t), T_3(t), T_5(t), \cdots, T_{k-1}(t)$.
else
  Evaluate $r(t)$ recursively.
end
Evaluate $p(t)$ with $q(t), r(t)$.

We now compare the number of the non-scalar multiplications of composite function approximation and the direct approximation. We compare these approximations in two aspects. One is the case of minimizing the number of non-scalar multiplications without any consideration of depth, and the other is the case of minimizing the number of non-scalar multiplications with minimal depth. We use the Paterson-Stockmeyer algorithm and its odd function variant in the first aspect, and use the Baby-step Giant-step algorithm and its odd function variant in the second aspect. For the composite function approximation, we separate the three cases where the scaling factor of the cosine function is 1, 2, or 3, respectively.

Fig. 6 shows the number of non-scalar multiplications and the depth for the various required minimax approximation error in the first aspect and Fig. 7 shows the number of non-scalar multiplications and the depth for the various required minimax approximation error in the second aspect. We observe that the composite function approximation reduces effectively the number of the non-scalar multiplications compared to the direct approximation. On the other hand, the direct approximation usually consumes the smallest depth.

7 Conclusion and Future Works

We obtained the fast algorithm to derive the generalized minimax approximate polynomial of any piecewise continuous functions over any union of the finite number of intervals, which generalizes the Remez algorithm, called modified Remez algorithm. Using that results, we directly derived the optimal minimax approximate polynomial for the modular reduction function rather than the sine or cosine function in the CKKS scheme. From the numerical analysis, there is some gap of the approximation error by two in the logarithm scale between the cosine minimax approximation and the proposed direct minimax approximation. Further, there is some inherent flat error region for the cosine minimax approximation such that the minimax approximation error does not decrease as the degree of the approximate polynomial increases, but the approximation error for the proposed one is improved as the degree of approximate polynomial increases.

Then, we proposed the composite function approximation to obtain approximation error smaller than the fundamental flat error with a small number of the non-scalar multiplications by using the inverse sine function. For the direct approximation, we reduced the number of non-scalar multiplications by 30% by using odd function property of the minimax approximate polynomial of modular reduction function. By the numerical analysis, it was known that the composite function approximation is desirable when the running time of bootstrapping is important, and the direct approximation with odd function optimization is desirable when the depth is important.

The theoretical relation between the composite function approximation and the minimax approximate polynomials of the normalized modular reduction function is an important open problem to understand the minimax approximation error in the CKKS encryption scheme. We put this relation as future work.

References

1. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory* **6**(3), 13 (2014)
2. Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: *Advances in Cryptology-EUROCRYPT 2019*. pp. 34–54. Springer (2019)

3. Cheney, E.: *Introduction to approximation theory*. McGraw-Hill (1966)
4. Cheon, J., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: *Advances in Cryptology-EUROCRYPT 2018*. pp. 360–384. Springer (2018)
5. Cheon, J., Han, K., Kim, A., Kim, M., Song, Y.: A full rns variant of approximate homomorphic encryption. In: *International Conference on Selected Areas in Cryptography-SAC 2018*. pp. 347–368. Springer (2018)
6. Cheon, J., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: *Advances in Cryptology-ASIACRYPT 2017*. pp. 409–437. Springer (2017)
7. Cheon, J., Kim, D., Kim, D.: Efficient homomorphic comparison methods with optimal complexity. Cryptology ePrint Archive, Report 2019/1234 (2019)
8. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. *Journal of Cryptology* **33**(1), 34–91 (2020)
9. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive p. Report 2012/144 (2012)
10. Filip, S.: A robust and scalable implementation of the Parks-McClellan algorithm for designing FIR filters. *ACM Transactions on Mathematical Software* **43**(1), 1–24 (2016)
11. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*. pp. 169–178 (2009)
12. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: *Advances in Cryptology-CRYPTO 2013*. pp. 75–92. Springer (2013)
13. Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: *Cryptographers' Track at the RSA Conference*. pp. 364–390. Springer (2020)
14. Lee, Y., Lee, J., Kim, Y., No, J.: Near-optimal polynomial for modulus reduction using L_2-norm for approximate homomorphic encryption. Cryptology ePrint Archive p. Report 2020/488 (2020)
15. McClellan, J., Parks, T.: A personal history of the Parks-McClellan algorithm. *IEEE Signal Processing Magazine* **22**(2), 82–86 (2005)
16. Parks, T., M.J.: Chebyshev approximation for nonrecursive digital filters with linear phase. *IEEE Transactions on Circuit Theory* **19**(2), 189–194 (1972)
17. Paterson, M., Stockmeyer, L.: On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM Journal on Computing* **2**(1), 60–66 (1973)
18. Powell, M.: *Approximation theory and methods*. Cambridge University Press (1981)
19. Remez, E.: Sur la détermination des polynômes d'approximation de degré donnée. *Communications of the Kharkov Mathematical Society* **10**(196), 41–63 (1934)
20. Rudin, W.: *Principles of mathematical analysis*, vol. 3. McGraw-Hill New York (1964)
21. Van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: *Advances in Cryptology-EUROCRYPT 2010*. pp. 24–43. Springer (2010)

A Proof of Theorem 5.2

Theorem A.1 (Restatement of Theorem 5.2)**.** *The minimax approximate linear polynomial of* $\arcsin x$ *in* $[-\sin\epsilon, \sin\epsilon]$ *is* $c_{\mathsf{min}}x$ *such that*

$$c_{\mathsf{min}} = 1 + \frac{\epsilon^2}{8} + O(\epsilon^4).$$

Further, we have

$$\frac{\|(1+\epsilon^2/8)x - \arcsin x\|_\infty}{\|x - \arcsin x\|_\infty} = \frac{\|c_{\mathsf{min}}x - \arcsin x\|_\infty}{\|x - \arcsin x\|_\infty} + O(\epsilon^2) = \frac{1}{4} + O(\epsilon^2),$$

where the domain of all functions is $[-\sin\epsilon, \sin\epsilon]$.

Proof. Since $\arcsin x$ is an odd function, the minimax approximate linear polynomial is an odd function. Thus, the constant term of the minimax approximate linear polynomial is zero, and we can write the minimax approximate linear polynomial of $\arcsin x$ as $c_{\mathsf{min}}x$. Let $y = \arcsin x$. Then $c_{\mathsf{min}}\sin y$ is the minimax approximate function for the identity function y among functions of the form $c\sin y$ with some c. Then we have to prove the following equation

$$\frac{\|(1+\epsilon^2/8)\sin y - y\|_\infty}{\|\sin y - y\|_\infty} = \frac{\|c_{\mathsf{min}}\sin y - y\|_\infty}{\|\sin y - y\|_\infty} + O(\epsilon^2) = \frac{1}{4} + O(\epsilon^2), \tag{17}$$

where the domain of these function of y is $[-\epsilon, \epsilon]$.

It is sufficient to consider only the non-negative values in the domain, that is, $[0, \epsilon]$, due to the symmetry of functions. We can have $c_{\mathsf{min}} \geq 1$, because if $c_{\mathsf{min}} < 1$, we have $c_{\mathsf{min}}\sin y < \sin y < y$ in $(0, \epsilon]$ and then the approximation error by $c_{\mathsf{min}}\sin y$ is larger than that of $\sin y$. Thus, we can represent $c_{\mathsf{min}} = 1 + \delta_{\mathsf{min}}$ for some non-negative value δ_{min}. For making the following proof more simple, we set $\delta_{\mathsf{min}} = \mu_{\mathsf{min}}\epsilon^2$ for some non-negative μ_{min}.

Let $(1+\delta_u)\sin y$ be a function such that $(1+\delta_u)\sin\epsilon = \epsilon$. Note that $(1+\delta_u)\sin y \geq y$ in $[0, \epsilon]$. If $\delta_{\mathsf{min}} > \delta_u$, we have $(1+\delta_{\mathsf{min}})\sin y > (1+\delta_u)\sin y \geq y$ in $[0, \epsilon]$ and thus we have $\delta_{\mathsf{min}} \leq \delta_u$. Thus, we have $0 \leq \delta_{\mathsf{min}} \leq \delta_u$. Note that $e_+(\delta) = \max_{y\in[0,\epsilon]}((1+\delta)\sin y - y)$ is increasing function of δ in $[0, \delta_u]$ with having 0 at $\delta = 0$, and $e_-(\delta) = \max_{y\in[0,\epsilon]}(y - (1+\delta)\sin y)$ is a decreasing function of δ in $[0, \delta_u]$ with having 0 at $\delta = \delta_u$. Since $\|(1+\delta)\sin y - y\|_\infty = \max\{e_+(\delta), e_-(\delta)\}$, δ_{min} is the point where $e_+(\delta) = e_-(\delta)$, that is, the minimax approximation error occurs.

We will now obtain a non-zero lower bound for μ_{min} using the following inequality

$$\sin y \leq y - \frac{y^3}{12} \leq y$$

for small enough y. We can assume that this inequality holds in $[0, \epsilon]$. Let $\tilde{r}_\delta(y) = (1+\delta)(y - \frac{1}{12}y^3) - y = \delta y - \frac{1}{12}(1+\delta)y^3$ for small enough δ. Then, the local maximum point of $\tilde{r}_\delta(y)$ is located at $y = 2\sqrt{\delta/(1+\delta)}$ by obtaining the zero of

the first derivative of $\tilde{r}_\delta(y)$. Thus, the maximum value of $\tilde{r}_\delta(y)$ is $\frac{4}{3}\sqrt{\delta^3/(1+\delta)}$. Since the minimum value of $\tilde{r}_\delta(y)$ is at $y=\epsilon$, the minimum value of $\tilde{r}_\delta(y)$ is $\delta\epsilon - \frac{1}{12}(1+\delta)\epsilon^3$. It is easy to check that $e_+(\delta) \le \frac{4}{3}\sqrt{\delta^3/(1+\delta)}$, and $e_-(\delta) \ge -\delta\epsilon + \frac{1}{12}(1+\delta)\epsilon^3$. If we set $\delta = \frac{1}{16}\epsilon^2$, we can obtain

$$\begin{aligned} e_+(\delta) \le \frac{4}{3}\sqrt{\frac{\delta^3}{1+\delta}} < \frac{4}{3}\delta^{\frac{3}{2}} &= \frac{1}{48}\epsilon^3 \\ &= -\delta\epsilon + \frac{1}{12}\epsilon^3 < -\delta\epsilon + \frac{1}{12}(1+\delta)\epsilon^3 \le e_-(\delta). \end{aligned}$$

Thus, we have $e_+(\delta) < e_-(\delta)$ for $\delta = \frac{\epsilon^2}{16}$. This means that $\delta_{\mathsf{min}} > \frac{1}{16}\epsilon^2$, and thus $\mu_{\mathsf{min}} > \frac{1}{16}$.

We will now obtain the exact formula for μ_{min}. Let $r_\delta(y) = (1+\delta)\sin y - y$. The maximum point of $r_\delta(y)$ is located at the zero of the first derivative of $r_\delta(y)$, and the zero is $\arccos(1/(1+\delta)) = \arcsin(\sqrt{\delta^2+2\delta}/(1+\delta))$. Since the minimum point of $r_\delta(y)$ is located at $y=\epsilon$, and $e_+(\delta_{\mathsf{min}}) = e_-(\delta_{\mathsf{min}})$, μ_{min} can be obtained from the following equation

$$\sqrt{\mu^2\epsilon^4 + 2\mu\epsilon^2} - \arcsin\frac{\sqrt{\mu^2\epsilon^4+2\mu\epsilon^2}}{1+\mu\epsilon^2} + (1+\mu\epsilon^2)\sin\epsilon - \epsilon = 0,$$

which is not easy to solve.

Let $h(\mu,\epsilon)$ be the left-hand side of the above equation as a function of μ and ϵ. Note that $h(\mu,\epsilon)$ is the differentiable function for $\mu,\epsilon > 0$. By the mean value theorem, there is a constant μ_c between μ_{min} and $\frac{1}{8}$ such that the following equation holds as

$$h\left(\frac{1}{8},\epsilon\right) = \frac{\partial h}{\partial\mu}(\mu_c,\epsilon)\left(\frac{1}{8} - \mu_{\mathsf{min}}\right). \tag{18}$$

By applying L'Hôpital's rule 5 times at the following formula, we have

$$\lim_{\epsilon\to 0+}\frac{1}{\epsilon^5}h\left(\frac{1}{8},\epsilon\right) = -\frac{19}{1280} \tag{19}$$

and thus we have $h(1/8,\epsilon) = O(\epsilon^5)$. On the other hand, since $\mu_{\mathsf{min}} > \frac{1}{16}$, we have $\mu_c > \frac{1}{16} > 0$ and then the following equation holds as

$$\frac{\partial h}{\partial\mu}(\mu_c,\epsilon) = \frac{\epsilon}{\sqrt{\mu_c(\mu_c\epsilon^2+2)}}\left(\mu_c\epsilon^2 + 1 - \frac{1}{\mu_c\epsilon^2+1}\right) + \epsilon^2\sin\epsilon \tag{20}$$

$$> \epsilon^2\sin\epsilon > \epsilon^3 - \frac{1}{6}\epsilon^5. \tag{21}$$

From (18), (19), and (21), we have $1/8 - \mu_{\mathsf{min}} = -19/1280\epsilon^2 + o(\epsilon^2)$ and

$$\mu_{\mathsf{min}} = \frac{1}{8} + O(\epsilon^2)$$

$$c_{\mathsf{min}} = 1 + \frac{1}{8}\epsilon^2 + O(\epsilon^4).$$

In fact, we already have $\mu_{\mathsf{min}} > \frac{1}{8}$ and thus $r_c(y) = (1 + \epsilon^2/8)\sin y - y$ has the maximum absolute error at $y = \epsilon$. Since $\epsilon - \sin\epsilon = O(\epsilon^3)$, the left-hand side of (17) can be rewritten as

$$\frac{\|c_{\mathsf{min}} \sin y - y\|_\infty}{\|\sin y - y\|_\infty} = \frac{\left(1 + \epsilon^2/8\right)\sin\epsilon - \epsilon}{\sin\epsilon - \epsilon} + \frac{O(\epsilon^5)}{\sin\epsilon - \epsilon} = \frac{\left(1 + \epsilon^2/8\right)\sin\epsilon - \epsilon}{\sin\epsilon - \epsilon} + O(\epsilon^2).$$

Using the Taylor series of $\sin x$, we have

$$\sin\epsilon - \epsilon = -\frac{1}{6}\epsilon^3 + O(\epsilon^5)$$

$$\left(1 + \frac{1}{8}\epsilon^2\right)\sin\epsilon - \epsilon = -\frac{1}{24}\epsilon^3 + O(\epsilon^5).$$

Then, we have

$$\frac{\|(1 + \epsilon^2/8)\sin y - y\|_\infty}{\|\sin y - y\|_\infty} = \frac{\left(1 + \frac{1}{8}\epsilon^2\right)\sin\epsilon - \epsilon}{\sin\epsilon - \epsilon} = \frac{1}{4} + O(\epsilon^2).$$

□