Sine Series Approximation of the Mod Function for Bootstrapping for Approximate HE

Charanjit S. Jutla
IBM T. J. Watson Research Center

Nathan Manohar
UCLA

Abstract

While it is well known that the sawtooth function has a point-wise convergent Fourier series, the rate of convergence is not the best possible for the application of approximating the mod function in small intervals around multiples of the modulus. We show a different sine series, such that the sine series of order n has error $O(\epsilon^{2n+1})$ for approximating the mod function in ϵ-sized intervals around multiples of the modulus. Moreover, the resulting polynomial, after Taylor series approximation of the sine series, has small coefficients, and the whole polynomial can be computed at a precision that is only slightly larger than $-(2n+1)\log\epsilon$, the precision of approximation being sought. This polynomial can then be used to approximate the mod function to almost arbitrary precision, and hence allows practical CKKS-HE bootstrapping with arbitrary precision.

1 Introduction

The work of [5, 4] presented a new homomorphic encryption (HE) scheme for approximate arithmetic (called the CKKS HE scheme) over real/complex numbers. The CKKS HE scheme was considerably more efficient than other schemes for evaluating arithmetic circuits and leveraged properties of approximate arithmetic to achieve these efficiency gains. It has found many applications, among them privacy-preserving machine learning and secure genome analysis (see [10, 15, 1, 12, 17, 11] for some examples). However, the initial CKKS HE scheme lacked a bootstrapping procedure, and, thus, it was not a *fully* homomorphic encryption (FHE) scheme. This was remedied when [3] introduced the first bootstrapping procedure for the CKKS HE scheme, which followed the general template introduced by Gentry [6] of evaluating the decryption circuit homomorphically. The challenge here is that the decryption procedure for CKKS requires computing the mod function, which is not easily representable via an arithmetic circuit. In fact, the mod function modulo q on the interval $[-Kq, Kq]$ for some integer K is not even a continuous function. However, [3] made the clever observation that in the

CKKS HE scheme, we have an upper bound m on the size of the message, which can be made much smaller than q. In this situation, we actually only need to be able to compute the mod function on points in $[-Kq, Kq]$ that are a distance at most m from a multiple of q. In this case, the mod function is periodic with period q and is linear on each of the small intervals around a multiple of q. Figure 1 shows the mod function along with the small intervals for approximation.

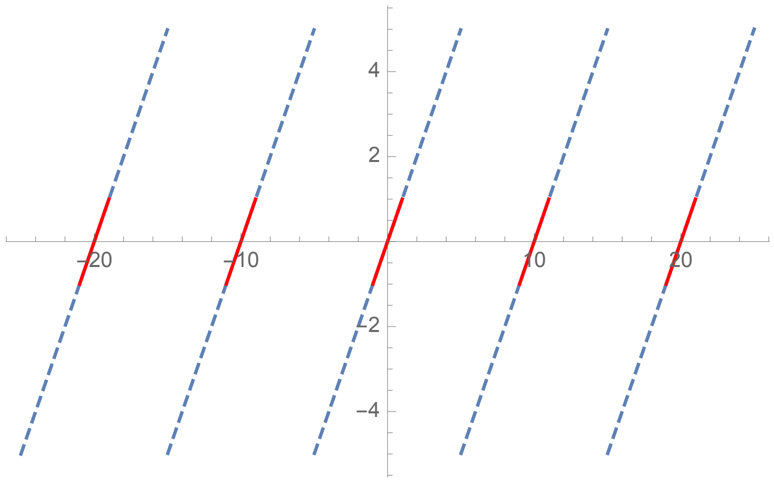

Figure 1: The mod function with modulus $q = 10$. The solid red lines represent the small intervals on which we need to approximate.

The work of [3] observed that the mod function $[t]_q$ on these intervals can be approximated via a scaled sine function $S(t) = \frac{q}{2\pi} \sin\left(\frac{2\pi t}{q}\right)$. This approximation introduces an inherent error that depends on the message upper bound m. Let ϵ denote the ratio $\frac{m}{q}$. Then, it can be shown that

$$|[t]_q - S(t)| \leq \frac{2\pi^2}{3} q\epsilon^3.$$

If ϵ is small enough, then this error can be sufficiently small for use in bootstrapping provided that $S(t)$ can be well-approximated by a low degree polynomial. The work of [3] along with several followup works [2, 8, 13] proceeded to provide methods of approximating this scaled sine function (or scaled cosine function in the case of [8] and scaled sine/cosine and inverse sine in the case of [13]) by a low-degree polynomial, which can then be plugged into the bootstrapping procedure of [3]. However, due to the inherent error between the mod function $[t]_q$ and the scaled sine function $S(t)$, this approach has a "fundamental error" that will occur regardless of how $S(t)$ is approximated. One of the problems with this is that in order for the error to be $O(1)$ (and, therefore, not destroy the message), m must be $O(q^{2/3})$. This means that we must begin bootstrapping while the size of the encrypted message is considerably smaller than q, which is a source of inefficiency in the bootstrapping procedure, particularly in applications that require high precision. Compounding this problem is the fact that for large q, the size of the coefficients of the approximation of $S(t)$ grow. This requires the basis polynomials to be computed to higher precision and affects the stability of the computation, where the errors introduced by the approximate arithmetic are amplified due to the large coefficients.

The reason obtaining high-precision bootstrapping for CKKS is important is that one of the main applications for CKKS is privacy-preserving machine learning. However, many ML algorithms require high precision computation in order to converge. This may be especially true during the learning phase of neural networks, which involves back propagation and integer division by private integers. Additional nonlinear steps involve pooling functions, threshold functions, etc. Moreover, due to their high depth, computing these ML algorithms homomorphically without bootstrapping is infeasible. Thus, for privacy-preserving ML applications, high-precision bootstrapping is required.

Recently, the work of [9] was able to obtain high-precision bootstrapping by finding direct polynomial approximations of the mod function on small intervals around the modulus via a new technique called modular Lagrange interpolation. This avoided the fundamental error inherent in prior works that occurred due to first approximating the mod function by a scaled sine function. The coefficients of these polynomials were small enough to enable high-precision bootstrapping. However, the coefficients were still large enough that in order to evaluate the polynomial approximations in a stable manner, one would need to operate at a higher precision than the input ciphertext. Ultimately, this fact corresponded to the bootstrapping procedure losing additional levels, since the computations during bootstrapping were operating at a higher precision.

1.1 This Work

In this work, we show how to obtain arbitrary precision bootstrapping via a different method from that of [9]. Instead of approximating the mod function directly, we first approximate the mod function by a sine series and then approximate each term in the sine series using a Taylor series. We show that the sine series converges to the mod function in small intervals around the modulus. In particular, our sine series of order n has error $O(\epsilon^{2n+1})$ for approximating the mod function in ϵ-sized intervals around multiples of the modulus.

Thus, we avoid the fundamental error of the scaled sine approach and are able to obtain an approximation with arbitrarily small error in the desired intervals. Furthermore, the coefficients of the sine series are small (in fact, they have norm < 1). This, combined with the fact that the Taylor series expansion of $\sin x$ has small coefficients, leads to a polynomial approximation of the mod function with small coefficients. Due to these small coefficients, the whole polynomial can be computed at a precision only slightly larger than $(-2n+1)\log\epsilon$, the precision of the approximation being sought. This means that during bootstrapping, we can operate at a precision level that is approximately the same as that of the input ciphertext and do not lose additional levels due to having to operate at a higher precision during bootstrapping.

2 Sine Series Approximation

In this section, we will show the following theorem and corollary, giving a sine series approximation to the mod function in small intervals around the modulus.

Theorem 1 *For every $n \geq 1$, there exist a sequence of rational numbers $\beta_1, ...\beta_n$ such that for every ϵ, $0 < \epsilon < 1/\sqrt{n}$, for every $|x| < \epsilon$,*

$$\left| x - \sum_{k=1}^{n} \beta_k \sin(kx) \right| < \frac{e^2}{2} * (n+1) * (\epsilon/2)^{2n+1}$$

Corollary 2 *For every $n \geq 1$, there exist a sequence of rational numbers $\beta_1, ...\beta_n$ such that for every ϵ, $0 < \epsilon < 1/\sqrt{n}$, for every integer m, for every x such that $|x - 2m\pi| < \epsilon$,*

$$\left| (x \; mod \, 2\pi) - \sum_{k=1}^{n} \beta_k \sin(kx) \right| < \frac{e^2}{2} * (n+1) * (\epsilon/2)^{2n+1}$$

To prove Theorem 1, for each n, we will determine the rational numbers $\{\beta_i\}_{i\in[n]}$. In particular, they are *not* the same as the Fourier coefficients of the sawtooth function, as we are focused on x that is potentially much smaller than the period of the sawtooth function.

We start by giving explicit formulas for the determinant of some Vandermonde-like matrices. For every $n > 0$, for every sequence of n distinct integers $\mathbf{a} = (a_1, ..., a_n)$, let $V^{(n)}(\mathbf{a})$ denote the Vandermonde matrix of $\mathbf{a}$, i.e. it is the $n \times n$ matrix with the (i,j)-th element a_i^{j-1} (for $i, j \in [n]$). Define $S^{(n)}(\mathbf{a})$ to be the $n \times n$ matrix with the (i,j)-th element a_i^{2j-1}, i.e. each row is the odd powers of the elements of $\mathbf{a}$. Note that the first column of this matrix is just $\mathbf{a}$. Also, define a related matrix $\hat{S}^{(n)}(\mathbf{a})$ to be the $n \times n$ matrix which is same as $S^{(n)}(\mathbf{a})$ except that the first column (i.e. $\mathbf{a}$) is replaced by $(2n+1)$-th powers of $\mathbf{a}$. In other words, the $(i,1)$-th element of this matrix is $a_i^{(2n+1)}$.

Lemma 3 *The determinant of the matrix* $S^{(n)}(\mathbf{a})$ *defined above is*

$$\left(\prod_{i=1}^{n} a_i\right) * \prod_{i=1}^{n} \prod_{1\le j<i} (a_i^2 - a_j^2).$$

The determinant of the matrix $\hat{S}^{(n)}(\mathbf{a})$ *defined above is*

$$(-1)^{n-1} \; * \; \det(S^{(n)}(\mathbf{a})) \; * \; \prod_{i=1}^{n} a_i^2.$$

Proof: We will first focus on the matrix $S^{(n)}(\mathbf{a})$. For computing the determinant, for each row i, we get a contribution of a factor a_i towards the determinant, and the remaining matrix is then just a Vandermonde matrix with all powers of a_i^2. Thus,

$$\det(S^{(n)}(\mathbf{a})) = \left(\prod_{i=1}^{n} a_i\right) * \det(V^{(n)}(\mathbf{a}')),$$

where $\mathbf{a}' = (a_1^2, \ldots, a_n^2)$. The result then follows from the well-known determinant of Vandermonde matrices.

As for the claim for the matrix $\hat{S}^{(n)}(\mathbf{a})$, first consider a modified matrix that is obtained by moving the first column to the last. Since this can be accomplished by $(n-1)$ column exchanges, the determinant of the modified matrix is $(-1)^{n-1}$ times the determinant of $\hat{S}^{(n)}(\mathbf{a})$. Furthermore, the determinant of the modified matrix is easily related to determinant of $S^{(n)}(\mathbf{a})$ by noting that i-th row in the modified matrix is a_i^2 times the i-th row in $S^{(n)}(\mathbf{a})$. □

If the sequence of integers $\mathbf{a}$ are in increasing order and lower bounded by one, then the determinant of $S^{(n)}(\mathbf{a})$ is positive. Let $\vec{\beta}$ be an n-vector of rational numbers. For the sine series approximation, we would like to determine $\vec{\beta}$ so that the transpose of the matrix $S^{(n)}(\mathbf{a})$ multiplied by $\vec{\beta}$ is a vector with all entries zero except the first, which is one. Since β_j refers to the coefficient of the $\sin(a_j x)$ term in the sine series, the above requirement ensures that when we Taylor expand each sine term in the sine series about the origin (or a multiple of 2π) and sum the terms, the resulting polynomial will be $x + x^{2n+1}p(x)$ for some polynomial $p(x)$. Thus, the $x^3, x^5, \ldots, x^{2n-1}$ terms in the Taylor series expansions of the $\sin(jx)$'s cancel out. We note that since our sine series will include $\sin x, \sin 2x, \sin 3x, \ldots$ terms, we will later instantiate $\mathbf{a}$ with $(1, 2, \ldots, n)$. The required condition is drawn below.

$$\begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ a_1^3 & a_2^3 & \cdots & a_n^3 \\ & & \vdots & \\ a_1^{2n-1} & a_2^{2n-1} & \cdots & a_n^{2n-1} \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Let d_i denote the $(i,1)$-th minor of $S^{(n)}(\mathbf{a})$. In other words, the list $\{d_i\}_i$ is the list of minors of the first column of $S^{(n)}(\mathbf{a})$.

Lemma 4

$$\beta_i = (-1)^{i+1} * \frac{d_i}{\det(S^{(n)}(\mathbf{a}))}.$$

Proof: From the above equation, $\vec{\beta}$ is just the first column of the inverse of $(S^{(n)}(\mathbf{a}))^T$. Note that the $(i,1)$-th element of inverse of transpose of $S^{(n)}(\mathbf{a})$ is $(-1)^{i+1} * d_i$ divided by the determinant of $S^{(n)}(\mathbf{a})$. □

We will show below that for any x in the domain of approximation, the absolute value of the error due to the sine series approximation of order n is upper bounded by the absolute value of

$$\sum_{i=1}^{n} \beta_i * i^{2n+1} * \frac{x^{2n+1}}{(2n+1)!}.$$

Thus, the value of $\sum_{i=1}^{n} \beta_i * i^{2n+1}$ is of interest to us.

Lemma 5 *For the matrix $S^{(n)}(\mathbf{a})$ with $\mathbf{a}$ set to the sequence of integers from one to n,*

$$\sum_{i=1}^{n} \beta_i * i^{2n+1} = (-1)^{n-1} * (n!)^2.$$

Proof: $\sum_{i=1}^{n} \beta_i * i^{2n+1}$ is the inner product of the first column of $\hat{S}^{(n)}(\mathbf{a})$ and $\vec{\beta}$. By Lemma 4, we have

$$\begin{aligned}
\vec{\beta}^\top \cdot (\hat{S}^{(n)}(\mathbf{a}))_1 &= ((S^{(n)}(\mathbf{a}))^{-\top})_1)^\top \cdot (\hat{S}^{(n)}(\mathbf{a}))_1 \\
&= \frac{1}{\det(S^{(n)}(\mathbf{a}))} * \sum_{i=1}^{n} (-1)^{i+1} d_i * (\hat{S}^{(n)}(\mathbf{a}))_{i,1} \\
&= \frac{\det(\hat{S}^{(n)}(\mathbf{a}))}{\det(S^{(n)}(\mathbf{a}))} \\
&= (-1)^{n-1} * \prod_{i=1}^{n} a_i^2 \\
&= (-1)^{n-1} * (n!)^2,
\end{aligned}$$

where we have used Lemma 8 in the second-to-last equality. □

We now show that the partial sums of the sine series satisfy Leibniz's alternating series test. For any x in the domain of approximation, consider the series $\sum_{m=n+1}^{\infty}(-1)^m * b_m$, where

$$b_m = (-1)^{n-1} * \sum_{i=1}^{n} \beta_i * \frac{(ix)^{2m-1}}{(2m-1)!} \tag{1}$$

The alternating series test requires that the b_m satisfy the following three conditions

1. $\lim_{m\to\infty} b_m = 0$
2. All b_m are positive (or all b_m are negative)
3. $|b_m| \geq |b_{m+1}|$ for all natural numbers $m \geq n+1$.

Theorem 6 *Alternating Series Test [Leibniz]. If the series above satisfies the alternating series test then $\sum_{m=n+1}^{\infty}(-1)^m * b_m$ converges. Moreover, for all $k \geq 0$,*

$$|\sum_{m=n+1}^{\infty}(-1)^m * b_m - \sum_{m=n+1}^{n+1+k-1}(-1)^m * b_m| \leq |b_{n+1+k}|.$$

Lemma 7 *For every $|x| < n^{-1/2}$, the above series given by b_m satisfies the Leibniz alternating series test.*

We prove this lemma in the next subsection, but before that we need formulas for determinant of some matrices related to $S^{(n)}(\mathbf{a})$. Define $V^{(n,k)}(\mathbf{a})$ to be a $n \times n$ matrix, which is same as the Vandermonde matrix $V^{(n)}(\mathbf{a})$ except the last column is replaced by $(n-1+k)$-th powers (instead of $(n-1)$-th powers).

Lemma 8 *For $k \geq 1$, the determinant of the matrix $V^{(n,k)}(\mathbf{a})$ is*

$$det\, V^{(n)}(\mathbf{a}) \; * \; s_k(\mathbf{a}),$$

where $s_k(\mathbf{a})$ is a (symmetric) polynomial in $\mathbf{a}$ given by

$$s_k(\mathbf{a}) = \sum_{1 \leq i_1 \leq \ldots \leq i_k \leq n} a_{i_1} * \cdots * a_{i_k}$$

Remark: The polynomials $s_k(\mathbf{a})$ differ from the well-known elementary symmetric polynomials $e_k(\mathbf{a})$ in that the latter has summation over $1 \leq i_1 < \ldots < i_k \leq n$.

Proof:

Fix any $k \geq 1$. Consider an $n \times n$ matrix M which is same as $V^{(n,k)}(\mathbf{a})$ except that the last row is powers of an indeterminate x. Let $\mathbf{a}'$ stand for a $(n-1)$ length truncation of $\mathbf{a}$. Treating the elements of $\mathbf{a}'$ as scalars, the determinant of this matrix M is a polynomial in x, of degree $n-1+k$. Call this polynomial $f(x)$. Since the determinant of a matrix with two equal (or even scaled by a constant) rows is zero, the polynomial $f(x)$ has roots $\mathbf{a}'$. Thus,

$$f(x) = g(x) * \prod_{i=1}^{n-1}(x - a_i), \tag{2}$$

where $g(x)$ is a polynomial (to be determined) of degree k . However, $f(x)$, the degree $n-1+k$ polynomial, has zero coefficients for all monomials x^j with j in $[n-1..n-1+k-1]$. Introduce a new formal variable $t = 1/x$, and then the above equation (2) can be written as

$$\tilde{f}(t) = \tilde{g}(t) * \prod_{i=1}^{n-1}(1 - ta_i). \tag{3}$$

where $\tilde{f}$ is the polynomial f with coefficients reversed, and similarly for $\tilde{g}$. Note, all the zero coefficients of $f(x)$ described above implies that $\tilde{f}(t) = f_{n-1+k} \bmod t^{k+1}$. Thus,

$$f_{n-1+k} \; * \prod_{i=1}^{n-1}(1 - ta_i)^{-1} = \tilde{g}(t) \bmod \; t^{k+1}. \tag{4}$$

Hence,

$$f_{n-1+k} \; * \prod_{i=1}^{n-1}\sum_{j=0}^{k}(1 + (ta_i)^j) = \tilde{g}(t) \bmod \; t^{k+1}. \tag{5}$$

Since $g(x)$ is of degree k, $\tilde{g}(t)$ has degree at most k as well. Denote by $\tilde{g}_z$ the coefficient of t^z in $\tilde{g}_z$, which is same as g_{k-z}. Then, by comparing coefficients of t^z on both sides we get, that for each $z \in [0..k]$,

$$g_{k-z} = \tilde{g}_z = f_{n-1+k} \; * \; s_z(\mathbf{a}'),$$

where $s_0(\mathbf{a}')$ is defined to be 1.

Thus, having determined $g(x)$, we also have $f(x)$ by (2). Letting $x = a_n$, then we get

$$\begin{aligned}
\det\ V^{(n,k)}(\mathbf{a}) &= f(a_n) \\
&= \prod_{i=1}^{n-1}(a_n - a_i)\ *\ g(a_n) \\
&= *\prod_{i=1}^{n-1}(a_n - a_i)\ *\ f_{n-1+k}\ *\ \sum_{z=0}^{k} a_n^{k-z} s_z(\mathbf{a}') \\
&= \prod_{i=1}^{n-1}(a_n - a_i)\ *\ f_{n-1+k}\ *\ s_k(\mathbf{a}) \\
&= \det V^{(n)}(\mathbf{a}) * s_k(\mathbf{a}),
\end{aligned}$$

where the last equality follows by noting that the top coefficient of $f(x)$, i.e. f_{n-1+k} is the (n,n)-minor of $V^{(n,k)}(\mathbf{a})$, which is same as the (n,n)-minor of Vandermonde matrix $V^{(n)}(\mathbf{a})$, which in turn is $(-1)^{n+n} *\ \det\ V^{(n-1)}(\mathbf{a}')$. $\square$

Lemma 9 *For* $\mathbf{a}$ *that is the sequence of squares of numbers from* 1 *to* n*, for all* $k \geq 0$*,*

$$\frac{s_{k+1}(\mathbf{a})}{s_k(\mathbf{a})} \leq n^3$$

Proof: First note that $s_{k+1}(\mathbf{a}) = \sum_{i=1}^{n} a_i * s_k(\mathbf{a}_{(i)})$, where $\mathbf{a}_{(i)}$ is $\mathbf{a}$ restricted to first i entries. Since a_i are monotonically increasing, it follows that $s_{k+1}(\mathbf{a}) \leq n * a_n * s_k(\mathbf{a})$, from which the claim follows. $\square$

2.1 Alternating Series Test

Proof: (of Lemma 7) In this proof we will fix $\mathbf{a}$ to be the sequence of integers from one to n. Note, each b_m can be written as $b_m = c_m * \frac{(x)^{2m-1}}{(2m-1)!}$, where $c_m = (-1)^{n-1} * \sum_{i=1}^{n} \beta_i * i^{2m-1}$.

1. Since n is fixed and all β_i are bounded by lemma 8, we just need to show that for every x in the domain of approximation, for every $i \in [n]$, $\frac{(ix)^{2m-1}}{(2m-1)!}$ goes to zero as m goes to infinity. Since the domain of approximation is bounded, $|x|$ itself is bounded. Since, $k! \geq e(k/e)^k$, the above is upper bounded by $e^{-1} * (iex/(2m-1))^{2m-1}$, which goes to zero as m goes to infinity.

2. To show that all b_m are positive (or all are negative), it suffices to show that all c_m are positive (or all c_m are negative resp.). We first show that c_{n+1} is positive (i.e. m set to $n+1$). By lemma 5, this quantity is just $(-1)^{2(n-1)} * (n!)^2$, and hence is positive.

   Let $\hat{S}^{(n,k)}(\mathbf{a})$ be the matrix that is the same as $\hat{S}^{(n)}(\mathbf{a})$ except that the first column is replaced by $(2n-1+2k)$ powers of $\mathbf{a}$. Thus, $\hat{S}^{(n,1)}(\mathbf{a})$ is same as $\hat{S}^{(n)}(\mathbf{a})$. As in the proof of lemma 5,

$$\begin{aligned}(-1)^{n-1} * c_{n+k} &= \sum_{i=1}^{n} \beta_i * i^{2n-1+2k} \\ &= \vec{\beta}^\top \cdot (\hat{S}^{(n,k)}(\mathbf{a}))_1 \\ &= ((S^{(n)}(\mathbf{a}))^{-\top})_1)^\top \cdot (\hat{S}^{(n,k)}(\mathbf{a}))_1 \\ &= \frac{1}{\det S^{(n)}(\mathbf{a})} * \sum_{i=1}^{n} (-1)^{i+1} d_i * (\hat{S}^{(n,k)}(\mathbf{a}))_{i,1} \\ &= \frac{\det \hat{S}^{(n,k)}(\mathbf{a})}{\det S^{(n)}(\mathbf{a})}\end{aligned}$$

   Note that $\det \hat{S}^{(n,k)}(\mathbf{a})$ is what is called a Schur polynomial in $\mathbf{a}$. It is an alternant polynomial, i.e. its sign changes if any two entries of $\mathbf{a}$ are exchanged. All alternant polynomials are divisible by the Vandermonde polynomial, i.e. $\det S^{(n)}(\mathbf{a})$; this is clear by noting that every $(a_i - a_j)$ is also a factor of $\det \hat{S}^{(n,k)}(\mathbf{a})$, as this determinant is zero if $a_i = a_j$. Regardless, we will not pursue this approach here, and directly use lemma 8. From that lemma it follows (after some maneuvering) that $\det \hat{S}^{(n,k)}(\mathbf{a})$ is

$$(-1)^{n-1} * s_{k-1}(\mathbf{a}^{(2)}) * \prod_{i=1}^{n} \prod_{1 \le j < i} (a_i^2 - a_j^2) \ * \ \prod_{i=1}^{n} a_i^3,$$

   where $\mathbf{a}^{(2)}$ is the sequence $\mathbf{a}$, but with each entry squared. Thus, all $c_{n,k}$ are positive, for $k \ge 1$.

3. We now show that $|b_m| \ge |b_{m+1}|$, for all $m \ge n+1$. In the following $\mathbf{a}$ is the sequence of

numbers from 1 to n. We have,

$$\begin{aligned}
\frac{|b_{m+1}|}{|b_m|} &= \frac{(-1)^{n-1} * s_{m+1-(n+1)}(\mathbf{a}^{(2)}) * \prod_{i=1}^{n}\prod_{1\le j<i}(a_i^2 - a_j^2) \; * \; \prod_{i=1}^{n} a_i^3 \; * \; \frac{(x)^{2m+1}}{(2m+1)!}}{(-1)^{n-1} * s_{m-(n+1)}(\mathbf{a}^{(2)}) * \prod_{i=1}^{n}\prod_{1\le j<i}(a_i^2 - a_j^2) \; * \; \prod_{i=1}^{n} a_i^3 \; * \; \frac{(x)^{2m-1}}{(2m-1)!}} \\
&= \frac{s_{m+1-(n+1)}(\mathbf{a}^{(2)}) \; * \; \frac{(x)^{2m+1}}{(2m+1)!}}{s_{m-(n+1)}(\mathbf{a}^{(2)}) \; * \; \frac{(x)^{2m-1}}{(2m-1)!}} \\
&= \frac{s_{m+1-(n+1)}(\mathbf{a}^{(2)})}{s_{m-(n+1)}(\mathbf{a}^{(2)})} \; * \; \frac{x^2}{2m(2m+1)} \\
&\le n^3 \; * \; \frac{x^2}{2m(2m+1)} \text{ (by lemma 9)} \\
&\le 1 \text{ (for } x < n^{-1/2})
\end{aligned}$$

□

Proof: (of Theorem 1) We have $|x - \sum_{i=1}^{n} \beta_i \sin(nx)| \le (n!)^2 * \frac{x^{2n+1}}{(2n+1)!}$ by Lemmas 5, 7 and Leibniz's alternating series test theorem . The claim follows by using the bounds

$$\left(\frac{n}{e}\right)^n < n! < \left(\frac{n+1}{e}\right)^{n+1} e$$

□

3 Application to Bootstrapping for Approximate HE

In Section 1, we explained that approximating the mod function on small intervals around the modulus is a necessary step in bootstrapping for approximate homomorphic encryption (CKKS). In this section, we will briefly overview the bootstrapping procedure for the CKKS-FHE scheme introduced in [3] and explain how our sine series approximation to the mod function enables high-precision bootstrapping.

Notation and Necessary Preliminaries: Let M be a power of 2 and $\Phi_M(X) = X^N + 1$ be the Mth cyclotomic polynomial of degree $N = M/2$. Let $\mathcal{R} = \mathbb{Z}[X]/\Phi_M(X)$. For an integer q, let $\mathcal{R}_q = \mathbb{Z}_q[X]/\Phi_M(X)$. Using the canonical embedding σ, it is possible to map an element $m(X) \in \mathcal{R}$ into $\mathbb{C}^N$ by evaluating $m(X)$ at the Mth primitive roots of unity. Using the same canonical embedding, it is also possible to define an isometric ring isomorphism between $\mathcal{S} = \mathbb{R}[X]/\Phi_M(X)$ and $\mathbb{C}^{N/2}$, where for an element $m(X) \in \mathcal{S}$, it has the canonical embedding norm $||m||_\infty^{\mathsf{can}} = ||\sigma(m)||_\infty$.

Overview of the CKKS-FHE Scheme: The CKKS-FHE scheme [5] is an FHE scheme for approximate arithmetic over real/complex numbers. Its security is based on the ring-LWE (RLWE) assumption. The message space of the scheme is polynomials $m(X)$ in $\mathcal{R}$ with $||m||_\infty^{\mathsf{can}} < q/2$ for a prime q. Using the canonical embedding and appropriate scaling, one can map a vector in $\mathbb{C}^{N/2}$ of fixed precision into $\mathcal{R}$. The fact that canonical embedding induces an isometric ring isomorphism between $\mathcal{S}$ and $\mathbb{C}^{N/2}$ implies that operations on the message space $\mathcal{R}$ map to the same operations performed coordinate-wise on $\mathbb{C}^{N/2}$. Thus, the CKKS-FHE scheme supports packing $N/2$ complex numbers into a single plaintext and operating on them in single instruction multiple data (SIMD) manner. Please refer to [5] for more details on this encoding procedure. We will refer to $m(X) \in \mathcal{R}$ as the plaintext/message and the corresponding vector in $\mathbb{C}^{N/2}$ as the plaintext "slots."

A ciphertext ct encrypting a message $m \in \mathcal{R}$ is an element of $\mathcal{R}_{q_\ell}^2$ for some $\ell \in \{0, \ldots, L\}$. ℓ refers to the "level" of the ciphertext. In [5], $q_\ell = p^\ell * q$ for primes p and q. However, q_ℓ can be set in other ways (such as via an RNS basis [4]). The decryption structure is $\langle \mathsf{ct}, \mathsf{sk} \rangle \bmod q_\ell = m + e$ for some small error $e \in \mathcal{R}$. Observe that there is no way to remove e and some of the least significant bits of m are unrecoverable. A fresh ciphertext is generated at the highest level L. Homomorphic operations increase the magnitude of the error and the message and one must apply a rescaling procedure or modular reduction to bring a ciphertext to a lower level to continue homomorphic computation. Eventually, a ciphertext is at the lowest level (an element of $\mathcal{R}_q^2$), and no further operations can be performed.

Bootstrapping Procedure for CKKS-FHE: [3] introduced the first bootstrapping procedure for the CKKS-FHE scheme. Subsequent works [2, 7, 8] improved various aspects of bootstrapping, but the overall procedure remains the same. The goal is to take a ciphertext at the lowest level and bring it up to a higher level so that homomorphic computation can continue. Thus, given a ciphertext ct at the lowest level, we want to obtain another ciphertext ct' such that

$$\langle \mathsf{ct}, \mathsf{sk} \rangle \bmod q \approx \langle \mathsf{ct}', \mathsf{sk} \rangle \bmod q_\ell$$

for some $\ell > 1$. For simplicity in the following, we will include the starting decryption error in the message m. That is, we will assume that $\langle \mathsf{ct}, \mathsf{sk} \rangle \bmod q = m$.

Bootstrapping is done via the following sequence of steps:

1. **Modulus Raising:** By simply considering ct as a ciphertext at the highest level, it follows that $\langle \mathsf{ct}, \mathsf{sk} \rangle \bmod q_L = qI + m$ for some $I \in \mathcal{R}$.

2. **Coefficients to Slots:** We need to perform the modular reduction on the polynomial coefficients of $t = qI + m$. However, recall that homomorphic computations evaluate

coordinate-wise on the plaintext "slots," not the polynomial coefficients. Thus, we need to transform our ciphertext so that the polynomial coefficients are in the "slots." This can be done by evaluating a linear transformation homomorphically.

3. **Compute the Mod Function:** We need a procedure to compute/approximate the mod function homomorphically. This is a significant challenge since we can only compute arithmetic operations homomorphically.

4. **Slots to Coefficients:** Finally, we need to undo the coefficients to slots step. This can be done by homomorphically evaluating the inverse of the previous linear transform.

Observe that if we can approximate the mod function, then the above procedure will give us a ct' at some higher level ℓ that decrypts to $m + e$ for some small error e. Since we are dealing with approximate arithmetic, this error from bootstrapping can be absorbed into the other errors that occur during approximate arithmetic and homomorphic evaluation.

Prior Approaches to Approximating the Mod Function: We can upper bound $|I| < K$ for some integer K (a typical value is $K = 12$) so that we only need to approximate the mod function on the interval $[-Kq - m, Kq + m]$, where we have overloaded notation to make m an upper bound on the size of the message for consistency of notation with prior works. However, finding a good polynomial approximation for the mod function on this interval is difficult since it is not even a continuous function.

As described in the introduction, [3] observed that if m is sufficiently small, then the mod function $[t]_q$ can be approximated by the scaled sine function $S(t) = \frac{q}{2\pi}\sin\left(\frac{2\pi t}{q}\right)$. This approximation introduces a "fundamental error" of $\frac{2\pi^2}{3}q\epsilon^3$, where $\epsilon = m/q$. Thus, to obtain $O(1)$ error, we require $m = O(q^{2/3})$, meaning that we must begin bootstrapping prior to m becoming too large.

The work [3] then proceeded by approximating $S(t)$ using a Taylor expansion to degree $O(Kq)$ so that the error of approximation with $S(t)$ is about the same as the error between $S(t)$ and $[t]_q$. Since they are approximating a scaled sine function, they are able to use double-angle formulas for sine to reduce the computational cost of evaluating the approximation polynomial by first approximating a scaled-down version $\sin\left(\frac{2\pi t}{2^r * q}\right)$ to a degree $d_0 = O(1)$ and then using this approximation to approximate $S(t)$. The required setting of r is $O(\log Kq)$ and so the multiplicative depth (alternatively, the ciphertext levels consumed) remains the same.

The work [2] improved upon this method by instead using Chebyshev interpolation to approximate $S(t)$, which lowered the error of approximation and the required degree. In Chebyshev interpolation, instead of working with the polynomial basis $\{1, x, x^2, \ldots\}$, one works with the Chebyshev basis $\{T_0(x), T_1(x),$
$T_2(x), \ldots\}$ and uses the Chebyshev nodes as points for interpolation. Approximating $S(t)$ via Chebyshev interpolation $p_n(t)$ of degree $\leq n$ gives an error of

$$|S(t) - p_n(t)| \leq qK^{n+1} \frac{\pi^n}{(n+1)!}.$$

Observe that the above error does not depend on ϵ (that is, it is a good approximation on the entire space $[-Kq, Kq]$ and does not utilize the fact that we only need a good approximation close to multiples of q). However, for a typical parameter setting $K = 12$, this error bound only improves on the trivial bound of q for degree $n \geq 98$.

The work [8] improved on the approximation of the scaled sine function by leveraging the fact that we only care that our approximation is good near multiples of q. To do this, [8] uses Chebyshev interpolation on the union of these small intervals instead of the entire space $[-Kq, Kq]$. Implicit in this, they consider the ratio between the maximum size of a message and q. This procedure allows them to reduce the degree of the polynomial required for approximation and allows the error of approximation to depend on the ratio $\epsilon = m/q$. For approximating the scaled sine function on $2K + 1$ intervals near multiples of q (near $-Kq, \ldots, Kq$), the error of approximation in any particular interval is $O\left(\epsilon^d\right)$, where d is the number of points chosen for Chebyshev interpolation in that interval. However, due to the constants hidden in the big-O notation (which can depend exponentially on K), choosing the same number of points for Chebyshev interpolation in all intervals does not give the best approximation, and the authors choose d for each interval via a greedy algorithm.

The above approaches all require first approximating $[t]_q$ via a scaled sine function, and, therefore, will always at least have error $\frac{2\pi^2}{3} q\epsilon^3$. If we want to have a smaller error, it is necessary to use a different method that avoids the scaled sine function. A pair of recent works by the same authors [14, 13] attempt to avoid the scaled sine function by instead trying to find the optimal minimax polynomial of a fixed degree that approximates the mod function via algorithmic search. [14] uses L2-norm minimization and [13] uses a variant of the Remez algorithm [16] to obtain an approximation to the optimal minimax polynomial of a given degree that approximates the modular reduction function on the union of intervals containing points close to multiples of q. However, in both of these works, the polynomial is found via algorithmic search. Moreover, the degree of the polynomial is fixed a priori before any approximation is computed. Without any bounds showing trade-offs between the polynomial degree, size of the coefficients, and the error of approximation, it is hard to develop strategies for picking the degree. Unfortunately, as observed by [13], the size of the coefficients of these polynomials are too large to enable high-precision bootstrapping. By using a composition of sine/cosine and the inverse sine function, [13] are able to improve on [8], but their bootstrapping is only capable of up to 28 bit message precision.

Recently, the work of [9] was able to obtain high-precision bootstrapping by finding direct polynomial approximations of the mod function on small intervals around the modulus via a new technique called modular Lagrange interpolation. This avoided the fundamental error inherent in prior works that occurred due to first approximating the mod function by a scaled sine function. The coefficients of these polynomials were small enough to enable high-precision bootstrapping. However, the coefficients were still large enough that in order to evaluate the polynomial approximations in a stable manner, one would need to operate at a higher precision than the input ciphertext. Ultimately, this fact corresponded to the bootstrapping procedure losing additional levels, since the computations during bootstrapping were operating at a higher precision.

Our Approach to Approximating the Mod Function: Instead of giving a direct polynomial approximation of the mod function, we first approximate the mod function via a sine series and then approximate each sine term in the sine series using a Taylor series. We avoid the fundamental error associated with previous approaches since our sine series of order n has error $O(\epsilon^{2n+1})$, and, thus, we can set n to obtain the desired error. Moreover, the coefficients of our sine series and the Taylor series expansion of $\sin jx$ are small, which allows us to evaluate these polynomials at almost the same precision as the desired approximation. This improves over the modular Lagrange approach by saving levels during bootstrapping, since we can operate at a lower precision when evaluating the polynomial approximation of the mod function.

4 Polynomial Approximation of the Mod Function

Using the sine series approximation of the mod function given by Corollary 2 and the well-known Taylor series expansion of the sine function, we now give explicit low-degree polynomial approximations of the mod function on small intervals around multiples of the modulus to (almost) arbitrary precision. The resulting polynomials have small coefficients, as the Taylor series of the sine function has small coefficients, and the sine series itself has all coefficients less than one. Small coefficients are beneficial in contrast to large coefficients, as in the latter case one is forced to compute the different power monomials to much higher precision, wasting critical ciphertext space. To evaluate the sine series, we first compute a Taylor series approximation of $\sin x$. From this, the other higher order $\sin kx$ terms are computed conveniently using the double angle formula, or, more precisely, by squaring e^{ix} and taking the real component (recall that CKKS-FHE allows us to compute over complex numbers).

As for computing the Taylor series approximation of the sine function, note that the domain of approximation is small intervals around ℓq, where $\ell \in [-L..L]$ and q is the modulus. The bound L comes from the bound on the Hamming-weight of the secret key and is typically 8 to 24. If our input is $X = x + \ell q$ for some small offset x and $\ell \in [-L..L]$, our goal is to compute $\sin(2\pi(x+\ell q)/q)$. This then requires a Taylor series that has powers of $2\pi(x+\ell q)/q$, which can be more than one. Earlier works noted that one can instead first compute $\sin(2\pi(x+\ell q)/(q2^r))$ using a Taylor series expansion (for some $r > 0$) and then compute $\sin(2\pi(x+\ell q)/q)$ using multiple applications of double angle formula (or, more precisely, one can first compute $e^{2\pi i(x+\ell q)/(q2^r)}$, and then use r squarings). For instance, by setting $r = \lceil \log((L+1)2\pi) \rceil$, we assure that $2\pi i(x+\ell q)/(q2^r))$ has norm less than one, and the higher powers in the Taylor series rapidly approach zero. However, the Taylor series of sine also has $m!$ in the denominator of the coefficient of x^m terms. More precisely, the Taylor series gives

$$e^{2\pi i(x+\ell q)/(q2^r)} = \sum_{m=0}^{\infty} (2\pi i(x+\ell q)/(q2^r))^m/m!.$$

We now determine for which range of values of $(x+\ell q)$ the above restricted to the sine terms, i.e. the imaginary terms or odd powers of x, satisfies the alternating series test (so that the partial series error can be bound by the absolute value of the next missing term). Thus, we need to determine the conditions under which

$$\begin{aligned} 1 &> \frac{(2\pi|(x+\ell q)|/(q2^r))^{(2m+1)}/(2m+1)!}{(2\pi|(x+\ell q)|/(q2^r))^{(2m-1)}/(2m-1)!} \\ &= \frac{(2\pi|(x+\ell q)|/(q2^r))^2}{(2m+1)2m} \end{aligned}$$

Assuming $x << q$ and $2^r \geq (L+1)$, the above reduces to $(2m+1)2m > 2\pi$, or $m > \sqrt{\pi/2}$. Thus, if the Taylor series is computed partially up to any degree $2m-1$, $m > \sqrt{\pi/2}$, then the error in the approximation of sine is at most

$$(2\pi)^{2m+1}/(2m+1)! < (2\pi e/(2m+1))^{2m+1}$$

which is at most 2^{-2m+1}, if we further require that $m > 2\pi e\,(> \sqrt{\pi/2})$. Note, this calculation was based on 2^r being no larger than $L+1$.

Thus, having computed $\sin(2\pi(x+\ell q)/(q2^r))$ partially up to m terms, we now investigate the error for the higher order terms in the sine series, i.e. $\sin(2\pi k(x+\ell q)/q)$ for $k \geq 1$. If the error in the approximation of the original term is small, say $\epsilon << 1$, then the error for this k-th term is approximately $k2^r * \epsilon$. Thus, the total error in the sine series due to the Taylor series approximation of $\sum_{k=1}^{n} \beta_k \sin(2\pi k(x+\ell q)/q)$ is upper bounded in absolute value by $\sum_{k=1}^{n} |\beta_k| * k2^r |\epsilon|$, which is at most $n^2(L+1)\epsilon$, which, in turn, is at most $n^2(L+1)2^{-2m+1}$.

Finally, using Corrollary 2, the total error in the mod function approximation, for an input $X = x + \ell q$ with $\ell \in [-L..L]$ and $|x| < q/(2\pi) * \epsilon$ for any $\epsilon < 1/\sqrt{n}$ is

$$n^2(L+1)2^{-2m+1} + \frac{e^2}{2} * (n+1) * (\epsilon/2)^{2n+1} * q/(2\pi).$$

Thus, it makes sense to have m about $-\log_2(\epsilon/2) * n$ (which is typically greater than $2\pi e$ for $n > 1$; if this value is less than $2\pi e$, then one must use an r such that $2^r/(L+1)$ is correspondingly larger than one).

References

[1] Bergamaschi, F., Halevi, S., Halevi, T.T., Hunt, H.: Homomorphic training of 30,000 logistic regression models. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) Applied Cryptography and Network Security. pp. 592–611. Springer International Publishing, Cham (2019) 1

[2] Chen, H., Chillotti, I., Song, Y.: Improved bootstrapping for approximate homomorphic encryption. In: EUROCRYPT. pp. 34–54 (2019) 1, 3, 3

[3] Cheon, J., Han, K., Kim, A., Kim, M., Song, Y.: Bootstrapping for approximate homomorphic encryption. In: EUROCRYPT. pp. 360–384 (01 2018) 1, 1, 3, 3, 3

[4] Cheon, J.H., Han, K., Kim, A., Kim, M., Song, Y.: A full rns variant of approximate homomorphic encryption. In: Selected Areas in Cryptography – SAC 2018 (2018) 1, 3

[5] Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: ASIACRYPT (2017) 1, 3

[6] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC. pp. 169–178 (2009) 1

[7] Han, K., Hhan, M., Cheon, J.H.: Improved homomorphic discrete fourier transforms and fhe bootstrapping. IEEE Access 7, 57361–57370 (2019) 3

[8] Han, K., Ki, D.: Better bootstrapping for approximate homomorphic encryption. In: Jarecki, S. (ed.) Topics in Cryptology – CT-RSA 2020. pp. 364–390. Springer International Publishing, Cham (2020) 1, 3, 3

[9] Jutla, C.S., Manohar, N.: Modular lagrange interpolation of the mod function for bootstrapping for approximate he. Cryptology ePrint Archive, Report 2020/1355 (2020), https://eprint.iacr.org/2020/1355 1, 1.1, 3

[10] Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H.: Logistic regression model training based on the approximate homomorphic encryption. BMC Medical Genomics 11(4), 83 (2018), https://doi.org/10.1186/s12920-018-0401-7 1

[11] Kim, M., Harmanci, A., Bossuat, J.P., Carpov, S., Cheon, J., Chilotti, I., Cho, W., Froelicher, D., Gama, N., Georgieva, M., Hong, S., Hubaux, J.P., Kim, D., Lauter, K., Ma, Y., Ohno-Machado, L., Sofia, H., Son, Y., Song, Y., Jiang, X.: Ultra-fast homomorphic encryption models enable secure outsourcing of genotype imputation. bioRxiv (2020) 1

[12] Kim, M., Song, Y., Wang, S., Xia, Y., Jiang, X.: Secure logistic regression based on homomorphic encryption: Design and evaluation. JMIR Med Inform 6(2), e19 (Apr 2018), http://www.ncbi.nlm.nih.gov/pubmed/29666041 1

[13] Lee, J., Lee, E., Lee, Y., Kim, Y., No, J.: High-precision bootstrapping of rns-ckks homomorphic encryption using optimal minimax polynomial approximation and inverse sine function. IACR Cryptol. ePrint Arch. 2020, 552 (2020) 1, 3

[14] Lee, Y., Lee, J., Kim, Y., No, J.: Near-optimal polynomial for modulus reduction using l2-norm for approximate homomorphic encryption. IEEE Access 8, 144321–144330 (2020) 3

[15] Masters, O., Hunt, H., Steffinlongo, E., Crawford, J., Bergamaschi, F., Rosa, M.E.D., Quini, C.C., Alves, C.T., de Souza, F., Ferreira, D.G.: Towards a homomorphic machine learning big data pipeline for the financial services sector. In: RWC (2020) 1

[16] Remez, E., G.: Sur la determination des polynomes d'approximation de degre' donnee'. Comm. of the Kharkov Math. Soc. 10(196), 41–63 (1934) 3

[17] Sav, S., Pyrgelis, A., Troncoso-Pastoriza, J.R., Froelicher, D., Bossuat, J.P., Sousa, J.S., Hubaux, J.P.: Poseidon: Privacy-preserving federated neural network learning (2020) 1