Cheap and Fast Iterative Matrix Inverse in Encrypted Domain

Tae Min Ahn, Kang Hoon Lee, Joon Soo Yoo, and Ji Won Yoon(✉)

School of Cybersecurity, Korea University, Seoul 02841, Korea
{xoals3563,hoot55,sandiegojs,jiwon_yoon}@korea.ac.kr

Abstract. Homomorphic encryption (HE) is a promising technique for preserving the privacy of sensitive data by enabling computations to be performed on encrypted data. However, due to the limitations of arithmetic HE schemes, which typically only support addition and multiplication, many nonlinear operations must be approximated using these basic operations. As a result, some nonlinear operations cannot be executed in the same manner as they would be in the plain domain. For instance, the matrix inverse can be calculated using the Gaussian elimination method in the plain domain, which is not possible using only the usual arithmetic. Therefore, much literature has turned to iterative matrix inverse algorithms such as the Newton method, which can be implemented using only additions and multiplications. In this paper, we propose a new matrix inversion method with better performance and prove that the new method outperforms the existing method; the number of depths of the new method is fewer than that of the existing method. Thus, we can evaluate more operations and design the algorithm efficiently since the number of operations is limited in HE. We experiment on ML algorithms such as linear regression and LDA to show that our matrix inverse operation is more efficient than Newton's in HE. Our approach exhibits approximately twice the performance improvement compared to the Newton's method.

Keywords: inverse matrix · homomorphic encryption · machine learning

1 Introduction

Privacy-preserving data mining (PPDM) is becoming significantly vital as more and more data is collected, analyzed, and shared. In addition to this trend, privacy regulations such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) have boosted the significance of privacy-preserving techniques by necessitating organizations to protect sensitive and personal information. In this regard, various techniques such as differential privacy and homomorphic encryption (HE) are proposed to protect sensitive information. Among these techniques, homomorphic encryption, which is based

https://doi.org/10.1007/978-3-031-50594-2_17

on lattice-based cryptography, is considered a post-quantum resistant encryption algorithm and one of the most promising solutions to attacks from quantum computers. The technique is often referred to as the “holy grail” of cryptography, as it allows for computation on encrypted data. Hence, much literature [3–5] has focused on the execution of privacy-preserving data analysis using homomorphic encryption.

However, homomorphic encryption is often considered impractical due to its poor time performance, albeit with its promising properties. Specifically, the computation of HE circuits typically requires more time at least by the several order of magnitude compared to the construction of plain circuits. In [3], the evaluation of a logistic regression model on the Edinburgh Myocardial Infarction dataset, which consists of 1,253 observations and 10 features, has been reported to require 116 min when implemented within the HE scheme. In contrast, the same process in the plain domain, utilizing a personal computer, can be completed in a matter of seconds. In light of this limitation, much of the recent research has focused on algorithmically [14,15] improving the time performance of HE, as well as on the use of parallel structures [6] in the hardware construction of HE schemes, in an effort to make the technique practical for deployment.

The most efficient and practical implementation of a fully homomorphic encryption scheme based on the Learning with Errors (LWE) problem is the CKKS scheme with leveled homomorphic encryption (LHE) setting. While the CKKS scheme offers the ability to perform arbitrary computations through the use of bootstrapping techniques, its practical deployment is limited to the LHE setting. In this context, the depth of the circuit must be pre-determined, with the number of multiplications per ciphertext serving as the determining factor. Furthermore, as the scheme is based on arithmetic homomorphic encryption, the majority of algorithms must be approximated using only the basic operations of addition and multiplication.

The primary concern when executing privacy-preserving data mining algorithms in HE is the low latency of matrix operations. Among these operations, the most challenging and time-consuming task to construct within HE is the inverse operation. In the plain domain, the inverse of a matrix can be easily obtained through the use of Gaussian elimination. However, in the encrypted domain, all HE circuits must be designed for the worst-case scenario. Additionally, the encrypted elements in a matrix necessitate comparison operations for all elementary row (or column) operations and time-consuming divisions.

In order to overcome this problem, there are several attempts at designing matrix inverse operations in the context of HE. However, they have been met with limited success due to their naïve implementation, resulting in a significant increase in computational time and multiplicative depth. Cheon et al. [7] use a matrix version of Goldschmidt’s algorithm described in [8] since it can only be operated using additions and multiplications. However, it is not practical to use as it requires knowledge of a threshold value in advance, which is infeasible in the encrypted domain. Therefore, much literature generally uses Newton’s method [9] for matrix inversion in HE [11,12] because it obtains an approximate matrix inverse using only additions and multiplications in an iterative manner. However, it also has a drawback as it requires many iterations and multiplications.

The issue of multiplicative depth is also crucial when designing homomorphic circuits, as most algorithms in practice use leveled homomorphic encryption (LHE), in which the multiplicative depth is predetermined. In the encrypted domain, the matrix inverse must be approximated using a sequence of matrix multiplications, which significantly increases the multiplicative depth of the circuit. For example, the Newton method requires a multiplicative depth of 43 to approximate the inverse matrix, taking up most of the circuit's depth and preventing further operations. Although a technique called bootstrapping can increase the multiplicative depth of the ciphertext, it requires a much greater amount of time and is therefore avoided in practical circuit construction.

Therefore, it is crucial to design an efficient matrix inverse operation with fewer depths in leveled homomorphic encryption (LHE). By reducing the number of multiplications per ciphertext in the matrix inverse algorithm, one can design an HE circuit with a shallower multiplicative depth. Additionally, with the same security parameter set, more operations can be added for further computations within a leveled homomorphic encryption scheme or smaller parameters can be chosen for more efficient computation of the circuit.

In short, our contributions are summarized as the following:

- We present a novel iterative matrix inverse operation. Our technique can reduce the number of depths by nearly a half compared to the Newton's method, mostly used algorithms in the current literature.
- We provide mathematical proofs and experimental result comparing two approaches—ours and Newton's method. Specifically, we demonstrate the convergence speed and required depths of both approaches in theory and implementation.
- Our matrix inverse algorithm seamlessly integrates with the inverse matrix in HE. We substantiate our claim by presenting experimental results.

2 Background

2.1 Homomorphic Encryption

Homomorphic encryption (HE) is a technique that allows for computations to be performed on the encrypted data without the need for decryption, utilizing a one-to-one model between the client and the server. This is achieved by designing the encryption scheme based on the Learning with Errors (LWE) problem [10], which uses noise as a means of ensuring security. However, as computations are performed on the encrypted data, the noise in the ciphertext accumulates, and if this noise exceeds a certain threshold, the correctness of the decryption process can no longer be guaranteed.

Let $\mathcal{M}$ and $\mathcal{C}$ denote the spaces of plaintexts and ciphertexts, respectively. The process of HE is typically composed of four algorithms: key generation, encryption, decryption, and evaluation.

1. **Key generation**: Given the security parameter λ, this algorithm outputs a public key pk, a public evaluation key evk and a secret key sk.
2. **Encryption**: Using the public key pk, the encryption algorithm encrypts a plaintext $m \in \mathcal{M}$ into a ciphertext $ct \in \mathcal{C}$.
3. **Decryption**: For the secret key sk and a ciphertext ct, the decryption algorithm outputs a plaintext $m \in \mathcal{M}$.
4. **Evaluation**: Suppose a function $f : \mathcal{M}^k \rightarrow \mathcal{M}$ is performed over the plaintexts $m_1, \cdots, m_k$. Then, the evaluation algorithm takes in ciphertext $c_1, \cdots, c_k$ corresponding to $m_1, \cdots, m_k$ and the evaluation key evk to output c^* such that $\mathsf{Dec}(c^*) = f(m_1, \cdots, m_k)$.

In the field of homomorphic encryption, there are two primary categories of encryption schemes: fully homomorphic encryption (FHE) and leveled homomorphic encryption (LHE). FHE permits any computation to be executed on the encrypted data, while LHE is more restricted in the types of computations that can be performed. These distinctions are due to the various methods used to handle the accumulation of noise in the ciphertext.

FHE utilizes a specialized technique known as *bootstrapping* to reduce the noise in the ciphertext and increase the multiplicative level of the ciphertext, allowing for further computations to be performed on the encrypted data. However, the use of bootstrapping is a computationally expensive technique and can be time-consuming. In practical applications, LHE is often preferred for its faster performance when working with limited depth circuits. This is because LHE does not rely on the use of bootstrapping and thus is less computationally intensive.

Homomorphic encryption can be categorized in terms of evaluation based on the type of computations that can be performed on the encrypted data. Arithmetic homomorphic encryption allows for basic arithmetic operations such as addition and multiplication to be performed on the encrypted data. Two popular examples are CKKS encryption [1] and BFV [13] encryption schemes, where the CKKS encryption scheme is the latest and the most practical HE solution providing real number arithmetics. Boolean-based homomorphic encryption allows for Boolean operations, such as AND, OR, and NOT, to be performed on the encrypted data. TFHE [15] and FHEW [14] are two examples. The choice of the homomorphic encryption scheme depends on the specific application and the type of computations that need to be performed on the data.

2.2 Arithmetic HE

Arithmetic HE generally uses the usual arithmetic such as addition and multiplication within the limited multiplicative depth which is pre-defined by the encryption parameters. Therefore, one needs to consider the depth of the circuit in advance for the optimal performance since the more depth of the circuit requires larger parameter set resulting in the performance degradation. In the BFV and CKKS schemes, the depth of the circuit is mostly determined by the number of multiplications per chiphertext required for the HE circuit.

Moreover, the multiplication operation is more complex designed than the addition in HE. In BFV and CKKS, the multiplication between two ciphertexts entails auxiliary procedures such as relinearization and modulus switching. Therefore, the time gap between such operations differs in a significant amount. As an illustration, within the CKKS scheme, the computational time required for multiplication exceeds that of addition by a factor greater than 46 (time for mult. : $649\,ms$, add: $14\,ms$)[1]. Hence, it is important to note that reducing number of multiplication is crucial in HE circuit design.

One of the key features of the CKKS scheme is the use of the Single Instruction Multiple Data (SIMD) structure. SIMD [17] is a structure that enables the packing of vector plaintexts into a single ciphertext, and operations are performed in vector units. Another feature is additional functionalities such as slot rotation. Rotations enable us to interact with values located in different ciphertext slots. These features allow for efficient operations on vectors and matrices. Halevi et al. [16] introduce a matrix encoding method based on diagonal decomposition, where the matrix is arranged in diagonal order. This method requires $O(n)$ ciphertexts to represent the matrix, and the matrix multiplication can be computed using $O(n^2)$ rotations and multiplications and two circuit depths given the multiplication of two square matrices of size n.

Additionally, Jiang et al. [18] propose the matrix multiplication method that reduces the complexity of multiplications and rotations to $O(n)$ by employing three levels of computational depth. These approaches are beneficial in terms of computational efficiency. Nevertheless, within the scope of this paper, we employ a naive matrix multiplication approach that necessitates $O(n^3)$ multiplicative operations for the computation of the inverse matrix. The evaluation of an inverse matrix typically entails substantial computational depth. Utilizing a naive matrix multiplication method is advantageous in this regard, as it necessitates only a single depth.

2.3 Circuit Depth

In leveled homomorphic encryption, the total count of multiplication evaluations for a single ciphertext is predetermined by the initial depth parameter of the HE system. For example, when a ciphertext is assigned a depth level denoted as L, it is intrinsically constrained to execute a maximum of L multiplicative operations. Beyond this specified threshold of L multiplications, the ciphertext ceases to support further multiplication operations.

The design of HE circuits can significantly influence the multiplicative depths, making it a crucial consideration. To illustrate this point, consider four distinct ciphertexts denoted as x, y, z, and w, each initially possessing a depth level of L. When these ciphertexts are multiplied sequentially, it consumes 3 depth levels, resulting in a ciphertext denoted as $xyzw$ with a reduced depth of $L-3$.

Alternatively, we can initially perform a multiplication between x and y, yielding xy with a depth decrement of 1; likewise, we can evaluate a multipli-

[1] $\lambda = 128$, $N = 2^{16}$, $\Delta = 2^{50}$, $L = 50$.

cation on z and w. Finally, the multiplication of xy and zw results in a ciphertext $xyzw$ with a reduced depth of $L-2$. Importantly, both approaches yield equivalent results and require an identical count of 3 multiplication operations. However, the depth level of the resulting ciphertext differs by a factor of 1.

Note that when multiplying ciphertexts with different levels, the multiplication operations are executed based on the lowest level among them.

2.4 Conventional Iterative Matrix Inverse

There are mainly two approaches in implementing the iterative matrix inverse operation: Goldschmidt's method [8] and Newton's method.

Goldschmidt Algorithm. (See the details in Algorithm 3) Let $\mathbf{A}$ be an invertible square matrix that satisfies $\|\bar{\mathbf{A}}\| \leq \epsilon < 1$ for $\bar{\mathbf{A}} = \mathbf{I} - \frac{1}{2^t}\mathbf{A}$ for some non-negative integer t. It follows that

$$\frac{1}{2^t}\mathbf{A}(\mathbf{I}+\bar{\mathbf{A}})(\mathbf{I}+\bar{\mathbf{A}}^2)\cdots(\mathbf{I}+\bar{\mathbf{A}}^{2^{r-1}}) = \mathbf{I} - \bar{\mathbf{A}}^{2^r}$$

where $\mathbf{I}$ is the identity matrix. Additionally, we note that $\|\bar{\mathbf{A}}^{2^r}\| \leq \|\bar{\mathbf{A}}\|^{2^r} \leq \epsilon^{2^r}$, which implies that $\frac{1}{2^t}\prod_{i=0}^{r-1}(\mathbf{I}+\bar{\mathbf{A}}^{2^i}) = \mathbf{A}^{-1}(\mathbf{I}-\bar{\mathbf{A}}^{2^r})$ is an approximate inverse of $\mathbf{A}$ when $\epsilon^{2^r} \ll 1$.

The algorithm is able to correctly output the approximate matrix inverse for some sufficiently large $r \in \mathbb{N}$. Using the Goldschmidt algorithm, Cheon et al. propose a matrix inverse method over HE schemes [7].

Newton's Method. (See the details in Algorithm 4) Likewise, let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be any invertible square matrix, and let α be the reciprocal of the dominant eigenvalue of $\mathbf{AA}^T$. Newton's method computes the following sequence of matrices $\{X_k\}_{k\geq 0}$ as:

$$\mathbf{X}_0 = \alpha\mathbf{A}^T \quad \text{and} \quad \mathbf{X}_{k+1} = \mathbf{X}_k(2\mathbf{I} - \mathbf{AX}_k),$$

until $\mathbf{X}_k$ converges to $\mathbf{A}^{-1}$. We will dive into the details including the proof for convergence in Theorem 1.

Newton's method for obtaining an approximate inverse matrix consists of three steps: (1) computing $\mathbf{AA}^T$, (2) computing the dominant eigenvalue of $\mathbf{AA}^T$, and (3) calculating a sequence of $\mathbf{X}_n$ to approximate the inverse of $\mathbf{A}$. It is worth noting that α is the reciprocal of a dominant eigenvalue of $\mathbf{AA}^T$ which can be approximated using the Goldschmidt's algorithm through a combination of addition and multiplication operations.

In fact, it is difficult to directly obtain the dominant eigenvalue from homomorphic encryption. However, in this paper, we demonstrate that convergence can be proven even when a larger value is used rather than the exact value of the dominant eigenvalue. Therefore, some literature uses a trace instead of a dominant eigenvalue when obtaining the inverse matrix in homomorphic encryption by Newton's method [12]. The trace of a square matrix is the sum of its main diagonal elements. Thus, the trace of $\mathbf{A}\mathbf{A}^T$ is always greater than the dominant eigenvalue of $\mathbf{A}\mathbf{A}^T$ since the trace is the sum of eigenvalues and $\mathbf{A}\mathbf{A}^T$ is positive-definite.

3 Problems in Two Popular Methods

In this section, we will delve into the details and challenges associated with the implementation of the iterative matrix inverse operation in HE using two distinct approaches: Goldschmidt's method and Newton's method.

First, a major limitation of Goldschmidt's method is that the value of $\bar{\mathbf{A}}$ must be known in advance in order to satisfy the condition where $\|\bar{\mathbf{A}}\|$ is less than 1. This is infeasible, as all values—including input, intermediate, and output—are processed in an encrypted state. In other words, it is not possible to find t such that $\|\bar{\mathbf{A}}\| = \|\mathbf{I} - \frac{1}{2^t}\mathbf{A}\| < 1$. As a result, the algorithm cannot be initiated at all. It may be suggested to raise the value of t sufficiently large to match the condition of $\|\bar{\mathbf{A}}\| < 1$, however, this would highly likely zero out the elements of $\bar{\mathbf{A}} = \mathbf{I} - \frac{1}{2^t}\mathbf{A}$, thus the approach cannot provide the approximate matrix inverse for all $\mathbf{A}$.

Next, a drawback of Newton's method is the significant computational complexity in terms of overall time consumption. Upon examination of Newton's method, the sequence of $\mathbf{X}_n$ requires two matrix multiplications in one iteration; assuming that the process converges in r iterations, the time complexity of step (3) in Sect. 2.4 is $O(n^2 r)$. Additionally, step (3) consumes $2r$ circuit-depth. As a result, the time complexity of Newton's method and its depth-consumption are significant. To provide an intuitive example, for a small matrix of size $n = 10$ and iteration number $r = 15$, the total number of multiplications in a HE setting is 4,500. If we assume that each multiplication takes 649 ms, the expected time for the inverse matrix operation would be at least 2,920 s.

4 Proposed Approach

We propose a novel matrix inverse method by combining elements from both Goldschmidt's method and Newton's method.

4.1 Motivation

Goldschmidt's approach requires the value of t for the convergence of $\bar{\mathbf{A}} = \mathbf{I} - \frac{1}{2^t}\mathbf{A}$, however, as previously mentioned, finding this value in the encrypted domain is infeasible.

In contrast, Newton's method relates the dominant eigenvalue of $\mathbf{A}\mathbf{A}^T$ to the scaling of $\mathbf{A}\mathbf{A}^T$, where the scaling by α ensures that the norm of $\mathbf{A}\mathbf{A}^T$ is less than 1.

4.2 Efficient Matrix Inverse

Based on this observation, we posit that the dominant eigenvalue λ_1 is correlated with the role of t in Goldschmidt's method. To address this issue, (1) we first find the dominant eigenvalue of $\mathbf{AA}^T$, and (2) scale $\mathbf{AA}^T$ by its dominant eigenvalue. (3) We then use the normalized $\mathbf{AA}^T$ to iteratively approximate the matrix inverse using the Goldschmidt's sequence for $\mathbf{Y}_i$, as detailed in Algorithm 1.

Algorithm 1. Our Approach

```
1: Input: $n \times n$ invertible matrix $\mathbf{A}$, iteration number $r$
2: Output: approximate inverse matrix $\mathbf{Y}_r$
3: $\lambda_{\mathbf{1}} \leftarrow$ a dominant eigenvalue or trace of $\mathbf{AA}^T$
4: $\mathbf{Y}_0 \leftarrow \frac{1}{\lambda_{\mathbf{1}}}\mathbf{A}^T$
5: $\bar{\mathbf{A}}_0 \leftarrow \mathbf{I}_{n\times n} - \frac{1}{\lambda_{\mathbf{1}}}\mathbf{AA}^T$
6: for $i = 1$ to $r$ do
7:     $\mathbf{Y}_i \leftarrow \mathbf{Y}_{i-1}(\mathbf{I}_{n\times n} + \bar{\mathbf{A}}_{i-1})$
8:     $\bar{\mathbf{A}}_i \leftarrow \bar{\mathbf{A}}_{i-1}^2$
9: end for
```

In summary, our approach diverges from Newton's method in two fundamental ways: (1) we employ the Goldschmidt algorithm to approximate the inverse of matrix $\mathbf{A}$, and (2) our technique incurs a multiplicative depth of only 1 per iteration, while Newton's method entails a depth of 2 per iteration.

It is worth emphasizing that both methods involve the same number of multiplications per iteration, namely, 2. However, the discrepancy in depth utilization per iteration between the two methods arises from the fact that our approach permits the computation of multiplications independently, incurring a depth cost of 1 for each operation. In contrast, Newton's method conducts matrix multiplications sequentially, incurring a depth cost of 2 per iteration.

Furthermore, it is crucial to note that both Newton's method and our approach require an equivalent number of iterations to achieve convergence. Consequently, given that Newton's method necessitates a depth of 2 per iteration, our approach ultimately requires only half the depth cost to achieve convergence compared to Newton's method. Further details regarding this matter will be addressed in the subsequent proof section.

The reason for finding the dominant eigenvalue of $\mathbf{AA}^T$, instead of $\mathbf{A}$ itself, is because not all eigenvalues of the input matrix $\mathbf{A}$ are necessarily positive. For convergence, it is essential that the norm of $\bar{\mathbf{A}}_{\mathbf{0}}$ (the matrix used in the Algorithm 1) be less than 1. $\mathbf{AA}^T$ has the property that all of its eigenvalues are positive. By using the dominant eigenvalue of $\mathbf{AA}^T$, we ensure that the norm of $\bar{\mathbf{A}}_{\mathbf{0}}$ remains less than 1 for any invertible matrix $\mathbf{A}$. In the case that the input matrix $\mathbf{A}$ is positive definite, it is unnecessary to calculate $\mathbf{AA}^T$. Under such circumstances, we can directly evaluate the inverse matrix using the following approach.

Algorithm 2. Our Approach

```
1: **Input:** $n \times n$ positive-definite invertible matrix $\mathbf{A}$, iteration number $r$
2: **Output:** approximate inverse matrix $\mathbf{Y}_r$
3: $\lambda_{\mathbf{1}} \leftarrow$ a dominant eigenvalue or trace of $\mathbf{A}$
4: $\mathbf{Y}_0 \leftarrow \frac{1}{\lambda_{\mathbf{1}}}\mathbf{I}_{n\times n}$
5: $\bar{\mathbf{A}}_0 \leftarrow \mathbf{I}_{n\times n} - \frac{1}{\lambda_{\mathbf{1}}}\mathbf{A}$
6: for $i = 1$ to $r$ do
7:     $\mathbf{Y}_i \leftarrow \mathbf{Y}_{i-1}(\mathbf{I}_{n\times n} + \bar{\mathbf{A}}_{i-1})$
8:     $\bar{\mathbf{A}}_i \leftarrow \bar{\mathbf{A}}_{i-1}^2$
9: end for
```

5 Convergence and Depth Analysis

In this work, we demonstrate that our proposed method converges to the inverse matrix, and it does so at the same rate as Newton's method. To support our claim, we provide the following lemma, which establishes the convergence of a matrix $\mathbf{A}$ under a specific condition.

Lemma 1. *Suppose $\mathbf{A}$ is an $n \times n$ complex matrix with spectral radius $\rho(\mathbf{A})$. Then, $\lim_{k\to\infty} \mathbf{A}^k = 0$ if $\rho(\mathbf{A}) < 1$.*

5.1 Proof of Convergence

Suppose that the eigenvalues of an $n \times n$ matrix $\mathbf{A}$ by $\lambda_i(\mathbf{A}), i = 1, \ldots, n$. When $\mathbf{A}$ is positive-definite, we can order its eigenvalues in a non-decreasing order as follows:

$$\lambda_1(\mathbf{A}) \geq \lambda_2(\mathbf{A}) \geq \cdots \geq \lambda_n(\mathbf{A}) > 0.$$

It is worth noting that the eigenvalues of a positive definite matrix are real and positive.

We first state the convergence of Newton's iterative algorithm. We provide details of the proof of Theorem 1 in Appendix B.1 since it is used in other theorems.

Theorem 1. *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be an invertible matrix and define the sequence $\{\mathbf{X}_k\}_{k\geq 0}$ of matrices as follows:*

$$\begin{cases} \mathbf{X}_0 = \alpha \mathbf{A}^T, \\ \mathbf{X}_{k+1} = \mathbf{X}_k(2\mathbf{I} - \mathbf{A}\mathbf{X}_k). \end{cases}$$

where $\alpha = \frac{1}{\lambda_1(\mathbf{A}\mathbf{A}^T)}$. Then, $\mathbf{X}_k \to \mathbf{A}^{-1}$ as $k \to \infty$.

Next, we prove that the sequence in our approach (in Algorithm 1) converges to an inverse matrix, i.e., $\mathbf{Y}_i \to \mathbf{A}^{-1}$.

Theorem 2. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be an invertible matrix and define the sequence* $\{\mathbf{Y}_k\}_{k\geq 0}$ *of matrices as follows:*

$$\begin{cases} \mathbf{Y}_0 = \alpha \mathbf{A}^T, \ \textit{with } \alpha = \frac{1}{\lambda_1(\mathbf{A}\mathbf{A}^T)}, \\ \bar{\mathbf{A}} = \mathbf{I} - \alpha \mathbf{A}\mathbf{A}^T, \\ \mathbf{Y}_{k+1} = \mathbf{Y}_k(\mathbf{I} + \bar{\mathbf{A}}^{2^k}). \end{cases}$$

Then, $\mathbf{Y}_k \to \mathbf{A}^{-1}$ *as* $k \to \infty$.

Proof. From the definition of Theorem 2, we get

$$\mathbf{Y}_k = \alpha \mathbf{A}^T(\mathbf{I} + \bar{\mathbf{A}})(\mathbf{I} + \bar{\mathbf{A}}^2)\cdots(\mathbf{I} + \bar{\mathbf{A}}^{2^{k-1}}) = \mathbf{A}^{-1}(\mathbf{I} - \bar{\mathbf{A}}^{2^k}). \tag{1}$$

We show that $\rho(\bar{\mathbf{A}}) = \rho(\mathbf{I} - \alpha\mathbf{A}\mathbf{A}^T) < 1$. We note that the eigenvalues $\lambda_i(\bar{\mathbf{A}})$ are given by, $\lambda_i(\bar{\mathbf{A}}) = 1 - \alpha\lambda_i(\mathbf{A}\mathbf{A}^T)$. Since $\mathbf{A}\mathbf{A}^T$ is positive-definite and $\alpha = \frac{1}{\lambda_1(\mathbf{A}\mathbf{A}^T)}$, we have $|\lambda_i(\bar{\mathbf{A}})| < 1$. Thus, we can get $\rho(\bar{\mathbf{A}}) < 1$. Therefore, by Lemma 1 we have $\lim_{k\to\infty} \bar{\mathbf{A}}^k = 0$. We note that $\mathbf{Y}_k = \alpha\mathbf{A}^T \prod_{i=0}^{k-1}(\mathbf{I} + \bar{\mathbf{A}}^{2^i}) = \mathbf{A}^{-1}(\mathbf{I} - \bar{\mathbf{A}}^{2^k})$ follows from Eq. (1). Therefore,

$$\lim_{k\to\infty} \mathbf{Y}_k = \alpha\mathbf{A}^T \prod_{i=0}^{\infty}(\mathbf{I} + \bar{\mathbf{A}}^{2^i}) = \mathbf{A}^{-1}(\mathbf{I} - \lim_{k\to\infty} \bar{\mathbf{A}}^{2^k}) = \mathbf{A}^{-1}.$$

In the context of our method, we posit the use of the trace of $\mathbf{A}\mathbf{A}^T$ in place of the dominant eigenvalue of $\mathbf{A}\mathbf{A}^T$. Our method still guarantees convergence of the iterative process, as the spectral radius of the modified matrix $\bar{\mathbf{A}}$, denoted as $\rho(\bar{\mathbf{A}})$, remains less than one under this assumption.

5.2 Convergence Comparison

We prove that our method has the same convergence rate as Newton's method.

Theorem 3. *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be an invertible matrix. Suppose* $\{\mathbf{X}_k\}_{k\geq 0}$ *is the sequence of matrices generated from Newton's method of Theorem 1 and* $\{\mathbf{Y}_k\}_{k\geq 0}$ *generated from Theorem 2 with* $\mathbf{A}$*. Then for any* $0 < \epsilon \ll \|\mathbf{A}^{-1}\|$*, let* $R_1, R_2 \in \mathbb{N}$ *be the smallest integers that satisfy* $\|\mathbf{A}^{-1} - \mathbf{X}_i\| < \epsilon$ *for all* $i > R_1$ *and* $\|\mathbf{A}^{-1} - \mathbf{Y}_j\| < \epsilon$ *for all* $j > R_2$ *respectively. Then we have* $R_1 = R_2$*. That is, the method illustrated in Theorem 2 converges with the same iterations as Newton's method.*

Proof. From the proofs of Theorem 1 and Theorem 2, we have

$$\mathbf{X}_k = \mathbf{A}^{-1}(\mathbf{I} - \mathbf{R}_k) = \mathbf{A}^{-1}\left(\mathbf{I} - \left(\mathbf{I} - \frac{1}{\lambda_1(\mathbf{A}\mathbf{A}^T)}\mathbf{A}\mathbf{A}^T\right)^{2^k}\right),$$

$$\mathbf{Y}_k = \mathbf{A}^{-1}(\mathbf{I} - \bar{\mathbf{A}}^{2^k}).$$

We first prove that R_1 and R_2 always exist for $0 < \epsilon < \|\mathbf{A}^{-1}\|$. Define two sequences $\{x_k\}_{k\geq 0}$ and $\{y_k\}_{k\geq 0}$ with $x_k = \|\mathbf{A}^{-1} - \mathbf{X}_k\|$ and $y_k = \|\mathbf{A}^{-1} - \mathbf{Y}_k\|$.

For simplicity, we denote the greatest eigenvalue of $\mathbf{A}\mathbf{A}^T$ as λ_1, and the smallest eigenvalue as λ_n. Then we have

$$\begin{aligned} x_k = \|\mathbf{A}^{-1} - \mathbf{X}_k\| &= \left\|\mathbf{A}^{-1}\left(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}\mathbf{A}^T\right)^{2^k}\right\| \\ &\leq \|\mathbf{A}^{-1}\| \cdot \left\|\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}\mathbf{A}^T\right\|^{2^k} \\ &= \|\mathbf{A}^{-1}\| \cdot \left(\frac{\lambda_1 - \lambda_n}{\lambda_1}\right)^{2^k}. \end{aligned}$$

Also for y_k, we have

$$\begin{aligned} y_k = \left\|\mathbf{A}^{-1} - \mathbf{Y}_k\right\| = \left\|\mathbf{A}^{-1}\mathbf{A}\bar{\mathbf{A}}^{T^{2^k}}\right\| &= \left\|\mathbf{A}^{-1}\left(\mathbf{I} - \frac{1}{\lambda_1}\cdot\mathbf{A}\mathbf{A}^T\right)^{2^k}\right\| \\ &\leq \|\mathbf{A}^{-1}\| \cdot \left\|\mathbf{I} - \frac{1}{\lambda_1}\cdot\mathbf{A}\mathbf{A}^T\right\|^{2^k} \\ &= \|\mathbf{A}^{-1}\| \cdot \left(\frac{\lambda_1 - \lambda_n}{\lambda_1}\right)^{2^k}. \end{aligned}$$

Then by the definition of λ_1 and λ_n, we have the inequality

$$0 < \frac{\lambda_1 - \lambda_n}{\lambda_1} < 1.$$

From the results, we can observe that both sequences x_n and y_n monotonically decrease and both converge to 0 as $k \to \infty$. Thus, for any $0 < \epsilon \ll \left\|\mathbf{A}^{-1}\right\|$, there always exist $R_1, R_2 \in \mathbb{N}$ such that

$$x_i < \epsilon \text{ for all } i > R_1, \text{ and } y_j < \epsilon \text{ for all } j > R_2.$$

We further investigate the behavior of x_k and y_k to compare the minimal iteration required, namely R_1 and R_2:

$$\begin{aligned} x_{R_1} &= \left\|\mathbf{A}^{-1}\left(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}\mathbf{A}^T\right)^{2^{R_1}}\right\| < \epsilon, \\ y_{R_2} &= \left\|\mathbf{A}^{-1}\left(\mathbf{I} - \frac{1}{\lambda_1}\mathbf{A}\mathbf{A}^T\right)^{2^{R_2}}\right\| < \epsilon. \end{aligned}$$

It is readily evident that, for a given ϵ value, R_1 is equal to R_2.

Our proposed method, despite relying on the trace instead of the dominant eigenvalue when compared to Newton's method, demonstrates an equivalent convergence rate. The proof for this is similar to Theorem 3.

5.3 Depth Comparison

From Theorem 3, we confirm that our method converges at the same rate as Newton's method. It implies that our method uses less multiplicative depth for matrix inverse operation.

Specifically, let t_{div} denote the iteration number required for the division algorithm method. Moreover, let $\mathbf{X}_k$ and $\mathbf{Y}_k$ represent the previous two algorithms, and assume that $\mathbf{X}_k$ and $\mathbf{Y}_k$ converge at iterations of R_1 and R_2, respectively. Then, the total number of multiplications required for $\mathbf{X}_k$ is $2t_{div} + n^3 + n^2 + 2n^3R_1$ and the total number of multiplications required for $\mathbf{Y}_k$ is $2t_{div} + 2n^2 + 2n^3R_2$. Since the division algorithm requires the same amount of multiplications for both algorithms, we only compare the remaining terms. Hence, $\mathbf{Y}_k$ requires almost the same number of multiplications since $R_1 = R_2$.

For a depth comparison, we analysis the sequence equation $\mathbf{X}_{k+1} = \mathbf{X}_k(2\mathbf{I} - \mathbf{A}\mathbf{X}_k)$ in Theorem 1 and the sequence equation $\mathbf{Y}_{k+1} = \mathbf{Y}_k(\mathbf{I} + \bar{\mathbf{A}}^{2^k})$ in Theorem 2. First, assuming the depth level of the input matrix $\mathbf{A}$ is denoted as L, and the level of $\mathbf{X}_0$ is assumed to be $L - 5$, we can observe that $\mathbf{X}_1$ is computed by multiplying $\mathbf{A}$ and $\mathbf{X}_0$, then subtracting it from $2\mathbf{I}$, followed by another multiplication with $\mathbf{X}_0$. Considering only the multiplication operations (since addition and subtraction do not affect the level), the level of $\mathbf{A}\mathbf{X}_0$ becomes $L - 6$, and after another multiplication with $\mathbf{X}_0$, the resulting matrix $\mathbf{X}_1$ has a level of $L-7$. Following this pattern, we can see that $\mathbf{X}_2$ has a $L - 9$ level, $\mathbf{X}_3$ has a $L - 11$ level, and so on. Since the level difference between $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$ $(k \geq 0)$ is 2, we can conclude that the Newton method consumes 2 depths per iteration.

Next, assuming the level of $\mathbf{Y}_0$ is L, then $\bar{\mathbf{A}}$ has a $L-1$ level. $\mathbf{Y}_1$ is computed by adding $\bar{\mathbf{A}}$ and $\mathbf{I}$ and then multiplying it by $\mathbf{Y}_0$, resulting in a $L - 2$ level. $\bar{\mathbf{A}}^2$ is the square of $\bar{\mathbf{A}}$, which has $L - 2$ level. $\mathbf{Y}_2$ is the result of multiplying $\mathbf{Y}_1$ and $\bar{\mathbf{A}}^2$, which makes its level $L-3$. This pattern continues, and we can observe that $\mathbf{Y}_3$ has a $L - 4$ level, $\mathbf{Y}_4$ has a $L - 5$ level, and so on. The level difference between $\mathbf{Y}_k$ and $\mathbf{Y}_{k+1}$ $(k \geq 1)$ is always 1. Therefore, our method consumes 1 depth per iteration.

Based on the observation, the total depths required for $\mathbf{X}_k$ is $t_{div} + 2 + 2R_1$ and the total depths required for $\mathbf{Y}_k$ is $t_{div} + 3 + R_2$. Since $R_1 = R_2$, and assuming that $R_1 = R_2 \geq 2$, our method can achieve the inverse matrix with fewer depths compared to the Newton method.

6 Experiment

In this section, we conduct a comparative analysis to evaluate the performance of the proposed algorithm and Newton's method when applied to invertible matrices in both the plain and encrypted domains. The evaluation focuses on two critical metrics: circuit depth and iteration number. Subsequently, the proposed algorithm is applied to linear regression and LDA in the encrypted domain to validate its computational efficiency.

6.1 Experiment Setting

Environment. In our cryptographic experiments, we employed OpenFHE [2] library for implementing the CKKS scheme. All experiments were evaluated on a system consisting of Intel Core i9-9900K CPU 3.60GHz $\times$ 16, 62.7 GiB RAM, Ubuntu 20.04.4 LTS.

CKKS Scheme Setting. We employed a 128-bit security level for all CKKS implementations. The other encryption parameters, including the ring dimension N, scaling factor Δ, and circuit depth D, were pre-determined to perform the inverse matrix operations or machine learning algorithms. Furthermore, we exclusively used a leveled approach and avoided the use of bootstrapping during the evaluation of homomorphic circuits.

6.2 Invertible Matrix and Machine Learning

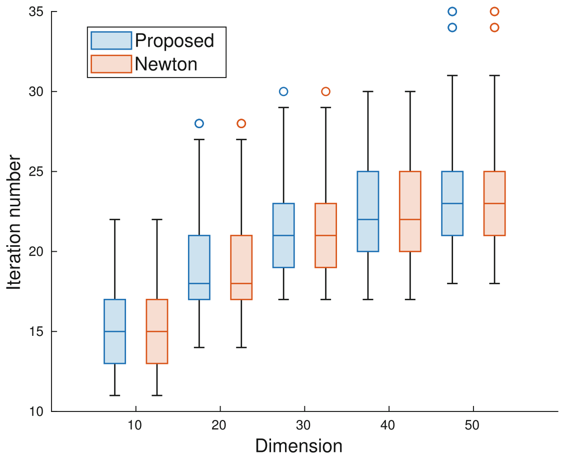

Fig. 1. The distribution of iteration numbers required for convergence to the inverse matrix across various dimensions for two algorithms—ours and Newton method.

Iteration Number Distribution. Figure 1 demonstrates the distributions of the iteration numbers for our proposed algorithm and the Newton's method. We conducted 100 experiments for matrix dimensions of $10, 20, \ldots, 50$ and depicted their distributions using box plots. We randomly generated square invertible matrices of varying sizes, with the smallest eigenvalue greater than 10^{-7} to avoid being recognized as zero. We recorded the iteration numbers at which convergence was achieved, with ϵ set to 0.001 and compared the approaches in the plain domain using Matlab R2022b. The results of our experiments show that our proposed algorithm converges identically to Newton's method regardless of dimension.

Time and Memory w.r.t. Circuit Depth. The reduction in depths has a significant impact on both the multiplication time and the memory size of ciphertext and keys in the encrypted domain. For example, in the CKKS scheme, with the same parameter set (λ, N, Δ), the multiplication time increases proportionally with respect to the depth D of the circuit (see Table 1). Specifically, multiplication time for $D = 1$ is 0.037, while 0.649 for $D = 50$; the latter is approximately 17 times greater.

Table 1. Impact of circuit depth on the multiplication and key generation time in the CKKS encryption scheme with fixed encryption parameters (λ, N, Δ).

λ	N	Δ	Depth	Mult. Time (s)	KeyGen Time (s)
128	2^{16}	2^{50}	1	0.037	0.225
128	2^{16}	2^{50}	10	0.14	0.763
128	2^{16}	2^{50}	20	0.251	1.323
128	2^{16}	2^{50}	30	0.361	1.884
128	2^{16}	2^{50}	40	0.486	2.461
128	2^{16}	2^{50}	50	0.649	3.062

Additionally, in the leveled-CKKS scheme, the size of the ciphertext and key are linearly determined by the circuit depth. This is due to the fact that the CKKS scheme uses rescaling (or similarly modulus-reduction in other schemes) procedure, which reduces the ciphertext size (modulus) after multiplication. Consequently, a larger initial ciphertext size is necessary to accommodate the entire circuit multiplications. Therefore, the depth of the circuit is a crucial factor that determines both the time performance and memory capacity in leveled encryption schemes.

Comparison of Implementation in Encrypted Domain: Time and Depth. We compare our proposed algorithm with the Newton's method for a randomly generated square matrix of size 5 with regards to error at specific iterations, under varying circuit depths (as seen in Table 2) in the encrypted domain. We use the same set of parameters (λ, N, Δ) as in Table 1 and measure the error of the approximated inverse matrix using the spectral norm. For the convergence of the approximated inverse, we set $\epsilon = 0.001$.

The results indicate that our algorithm converges at iteration number 16, which can be efficiently implemented with a circuit depth of $D = 27$. In contrast, the Newton's method converges at the same iteration number 16; however, it requires a circuit depth of $D = 43$.

Therefore, we conclude that our proposed algorithm has the same convergence speed as the Newton's method in the encrypted domain. However, as our algorithm can be implemented with a smaller circuit depth, its total execution

Table 2. Evaluation of our approach and Newton's method in the encrypted domain based on iteration number, circuit depth, and error (both use trace instead of dominant eigenvalue).

	Our Method			**Newton Method**		
Depth	#Iter	Error	Time(s)	#Iter	Error	Time(s)
20	9	0.4957	242.36	4	2.3868	177.97
25	14	0.0036	442.15	7	1.5761	301.48
27	16	$4.59e^{-6}$	596.62	8	1.0687	409.17
35	16	$4.59e^{-6}$	1054.35	12	0.1296	791.84
40	16	$4.59e^{-6}$	1242.98	14	0.0036	1042.87
43	16	$4.59e^{-6}$	1410.56	16	$4.59e^{-6}$	1256.37

time is about 596 s, whereas the Newton's method's execution time is about 1,256 s, making our method 2.1 times faster.

Table 3. Comparison of our proposed approach and Newton's method in performing ML algorithms—linear regression and LDA (both use trace instead of dominant eigenvalue).

	Our Method		**Newton Method**	
ML. Alg.	Iter. (Depth)	Time (s)	Iter. (Depth)	Time (s)
Linear Regression	22(58)	14921.42	22(58)	13352.61
	22(37)	7541.7	N/A	N/A
LDA	9(36)	1902.33	9(36)	1884.76
	9(28)	1481.71	N/A	N/A

Application to ML Algorithms. We demonstrate the efficiency of our approach through two popular ML algorithms, linear regression and LDA, that utilize a positive definite matrix as input to evaluate its inverse. We compare the efficiency of our method with the Newton's algorithm in terms of circuit depth and time performance in the encrypted domain; we show that our algorithm significantly enhances the overall performance.

For our evaluation of linear regression in the encrypted domain, we employed 100 samples with 8 features from the well-known public dataset "Diabetes dataset". We used the same encryption parameters λ, N, Δ and set $\epsilon = 0.001$ for the convergence of the matrix inverse operation. The linear regression of the dataset requires an inverse of a 8×8 square matrix. Our method and Newton's method both required 22 iteration number (see Table 3). However, our method requires less depth per iteration than Newton's method. This results in a circuit depth optimization of 37 for our method, compared to 58 for the Newton's method.

Initially, we conducted an experiment using the same circuit depth of 58 for both our method and Newton's method. Our approach closely resembles Newton's method in terms of the number of iterations required for convergence. However, it is noteworthy that as the depth level of the ciphertext decreases, the ciphertext modulus decreases as well, resulting in an increase in multiplication speed. In contrast to our method, which consumes only one depth in a single iteration, Newton's method consumes two depths in a single iteration. Consequently, even when performing the same number of operations, the multiplication of ciphertexts with a relatively lower depth level in Newton's method takes less time than in our approach. This phenomenon results in a decrease in the total execution time of Newton's method, reducing it by 1568.81 s compared to the execution time of our method. However, our method can perform additional 21 multiplications followed by the acquisition of the inverse matrix. Conversely, in the case of the Newton's method, further multiplication was no longer feasible upon obtaining the inverse matrix.

Subsequently, we measured the execution time of our approach with an optimal circuit depth of 37. Our approach demonstrated approximately 1.8 times less execution time compared to the Newton's method. It is important to note that the Newton's method cannot be implemented with a depth of 37; a minimum circuit depth of 58 is required to ensure correctness of the result.

In the evaluation of LDA, we used a subset of 150 samples from Iris flower dataset, which consists of 4 features and 3 species. With the same setting as in the linear regression, the LDA algorithm has to compute over an inverse of 4×4 matrix. Our method and Newton's method both required 9 iterations. Hence, the total depth required for each approach was 28 and 36, respectively, for constructing the optimal circuit. The evaluation time for the optimal circuit for each approach was approximately 1481.71 s for our method and 1884.46 s for the Newton's method, indicating a 1.27 times improvement in time performance of our proposed algorithm.

7 Conclusion

This paper presents a novel iterative matrix inverse algorithm that reduces multiplicative depths compared to the widely used Newton's method in the homomorphic encryption domain. Our algorithm offers significant improvements in computational time efficiency, with about 2 times reduction, and is advantageous in machine learning algorithms requiring the inverse of matrices.

Acknowledgements. This work was supported by an Institute of Information & Communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00558-003, Development of National Statistical Analysis System using Homomorphic Encryption Technology).

A Iterative Matrix Inverse Methods

A.1 Goldschmidt's Matrix Inverse Method

Algorithm 3. Goldschmidt's Matrix Inverse Approach

```
1: Input: $n \times n$ invertible matrix $\mathbf{A}$, iteration number $r$
2: Output: approximate inverse matrix $\mathbf{B}_r$
3: $t \leftarrow 1$
4: while true do
5:    $\bar{\mathbf{A}}_0 \leftarrow \mathbf{I}_{n\times n} - \frac{1}{2^t}\mathbf{A}$
6:    if $\|\bar{\mathbf{A}}_0\| < 1$ then
7:       break;
8:    end if
9:    $t \leftarrow t + 1$
10: end while
11: $\mathbf{B}_0 \leftarrow \frac{1}{2^t}\mathbf{I}_{n\times n}$
12: for $i = 1$ to $r$ do
13:    $\mathbf{B}_i \leftarrow \mathbf{B}_{i-1}(\mathbf{I}_{n\times n} + \bar{\mathbf{A}}_{i-1})$
14:    $\bar{\mathbf{A}}_i \leftarrow \bar{\mathbf{A}}_{i-1}^2$
15: end for
```

A.2 Newton's Matrix Inverse Method

Algorithm 4. Newton's Matrix Inverse Approach

```
1: Input: $n \times n$ invertible matrix $\mathbf{A}$, iteration number $r$
2: Output: approximate inverse matrix $\mathbf{B}_r$
3: $\lambda_{\mathbf{1}} \leftarrow$ a dominant eigenvalue of $\mathbf{A}\mathbf{A}^T$
4: $\mathbf{B}_0 \leftarrow \frac{\mathbf{1}}{\lambda_{\mathbf{1}}}\mathbf{A}^T$
5: for $i = 1$ to $r$ do
6:    $\mathbf{B}_i \leftarrow \mathbf{B}_{i-1}(2\mathbf{I}_{n\times n} - \mathbf{A}\mathbf{B}_{\mathbf{i-1}})$
7: end for
```

B Detailed Proof

B.1 Proof of Theorem 1

Proof. Let $\mathbf{R}_k = \mathbf{I} - \mathbf{A}\mathbf{X}_k$. Then, we note that $\mathbf{X}_{n+1} = \mathbf{X}_k(\mathbf{I} + \mathbf{R}_k)$. We first show that $\rho(\mathbf{R}_0) = \rho(\mathbf{I} - \alpha\mathbf{A}\mathbf{A}^T) < 1$. We note that the eigenvalues $\lambda_i(\mathbf{R}_0)$ are given by, $\lambda_i(\mathbf{R}_0) = 1 - \alpha\lambda_i(\mathbf{A}\mathbf{A}^T)$. Since $\mathbf{A}\mathbf{A}^T$ is positive-definite and $\alpha = \frac{1}{\lambda_1(\mathbf{A}\mathbf{A}^T)}$, we have $|\lambda_i(\mathbf{R}_0)| < 1$. Thus, we get $\rho(\mathbf{R}_0) < 1$. Therefore, by Lemma 1, we have

$$\lim_{k\to\infty} \mathbf{R}_0^k = 0. \tag{2}$$

Next, we note that,

$$\begin{aligned}\mathbf{R}_k = \mathbf{I} - \mathbf{A}\mathbf{X}_k &= \mathbf{I} - \mathbf{A}\mathbf{X}_{k-1}(\mathbf{I} + \mathbf{R}_{k-1}) \\ &= \mathbf{I} - \mathbf{A}\mathbf{X}_{k-1} - \mathbf{A}\mathbf{X}_{k-1}\mathbf{R}_{k-1} \\ &= \mathbf{R}_{k-1} - \mathbf{A}\mathbf{X}_{k-1}\mathbf{R}_{k-1} \\ &= (\mathbf{I} - \mathbf{A}\mathbf{X}_{k-1})\mathbf{R}_{k-1} = (\mathbf{R}_{k-1})^2.\end{aligned}$$

Therefore, inductively, we have $\mathbf{R}_k = \mathbf{R}_0^{2^k}$. Hence, $\lim_{n\to\infty} \mathbf{R}_k = \lim_{k\to\infty} \mathbf{R}_0^{2^k} = 0$, where the last equality follows from Eq. (2). Finally, from the definition of $\mathbf{R}_k$, we note that $\mathbf{X}_k = \mathbf{A}^{-1}(\mathbf{I} - \mathbf{R}_k)$. Therefore,

$$\lim_{n\to\infty} \mathbf{X}_k = \lim_{n\to\infty} \mathbf{A}^{-1}(\mathbf{I} - \mathbf{R}_k) = \mathbf{A}^{-1}.$$

Consider the scenario in which the trace of the matrix product $\mathbf{A}\mathbf{A}^T$ is utilized in place of the dominant eigenvalue. Despite the replacement of the scalar parameter alpha with the reciprocal of the trace of $\mathbf{A}\mathbf{A}^T$, the spectral radius of the matrix $\mathbf{R}_0$ remains less than one. This ensures that the iterative process converges.

References

1. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017. LNCS, vol. 10624, pp. 409–437. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70694-8_15
2. Al Badawi, A., et al.: OpenFHE: open-source fully homomorphic encryption library. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pp. 53–63 (2022)
3. Kim, A., Song, Y., Kim, M., Lee, K., Cheon, J.H.: Logistic regression model training based on the approximate homomorphic encryption. BMC Med. Genomics **11**(4), 23–31 (2018)
4. Sun, X., Zhang, P., Liu, J.K., Yu, J., Xie, W.: Private machine learning classification based on fully homomorphic encryption. IEEE Trans. Emerg. Top. Comput. **8**(2), 352–364 (2018)
5. Wood, A., Najarian, K., Kahrobaei, D.: Homomorphic encryption for machine learning in medicine and bioinformatics. ACM Comput. Surv. (CSUR) **53**(4), 1–35 (2020)
6. Jung, W., Kim, S., Ahn, J. H., Cheon, J. H., Lee, Y.: Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with GPUS. ACM Comput. Surv. (CSUR), 114–148 (2021)
7. Cheon, J.H., Kim, A., Yhee, D.: Multi-dimensional packing for HEAAN for approximate matrix Arithmetics. Cryptology ePrint Archive (2018)
8. Cetin, G.S., Doroz, Y., Sunar, B., Martin, W.J.: Arithmetic using word-wise homomorphic encryption. Cryptology ePrint Archive (2015)
9. Guo, C.H., Higham, N.J.: A schur-newton method for the matrix\boldmath p th Root and its Inverse. SIAM J. Matrix Analy. Appl. **28**(3), 788–804 (2006)

10. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM) **56**(6), 1–40 (2009)
11. Mital, N., Ling, C., Gündüz, D.: Secure distributed matrix computation with discrete Fourier transform. IEEE Trans. Inf. Theory **68**(7), 4666–4680 (2022)
12. Cock, M.D., Dowsley, R., Nascimento, A.C., Newman, S.C.:Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security, pp. 3–14 (2015)
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
14. Ducas, L., Micciancio, D.: FHEW: bootstrapping homomorphic encryption in less than a second. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 617–640. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46800-5_24
15. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. **33**(1), 34–91 (2020)
16. Halevi, S., Shoup, V.: Faster homomorphic linear transformations in HElib. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 93–120. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_4
17. Smart, N.P., Vercauteren, F.: Fully homomorphic SIMD operations. Des. Codes Crypt. **71**, 57–81 (2014)
18. Jiang, X., Kim, M., Lauter, K., Song, Y.: Secure outsourced matrix computation and application to neural networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 1209–1222 (2018)