Fully-homomorphically encrypted gradient descent algorithms for quadratic programming

André Bertolace, Konstantinos Gatsis, Kostas Margellos

***Abstract*— Quadratic programming is a very successful tool for modeling and solving many real-life problems. Several control theoretic problems involve solving a quadratic programming. State estimation under minimum square error or model predictive control are just a few examples of applications. In this paper, we implement and numerically analyze the extent at which gradient descent algorithms can be used to solve quadratic programming in a homomorphic encryption setup. The limit on the multiplication depth of homomorphic encryption circuits is a major challenge for iterative procedures such as gradient descent algorithms. However, it is possible to achieve results with algorithms that exhibit fast convergence rates such as Nesterov's accelerated gradient descent method. The paper shows for the first time the feasibility of fully homomorphically encrypted gradient descent algorithms.**

I. Introduction

Homomorphic encryption (HE) is a ground-breaking mathematical method that enables the analysis or manipulation of encrypted data without revealing its content [14]. In doing so, HE permits the secure delegation of data processing to third-party cloud providers. Several encryption schemes, such as Paillier [24] or El Gamal [9] are partially HE schemes[1]. In 2009, Gentry [13] proposed the first fully HE scheme[1]. The computational overhead of the scheme was significant, but it showed that such schemes are indeed possible. Since then these approaches have been further developed. Currently, the most used schemes are BFV [3], [10], CKKS [8], BGV [4], and GSW [16]. The computational overhead remains large but it has been brought down to a level where these schemes can be implemented in practice. The benefits of delegating data processing without giving away access to the data are tremendous. In the area of control theory, encrypted linear controllers have been implemented. Most results use partially HE schemes [1], [2], [7], [11], [12], [26], but approaches using fully HE schemes also exist [20], [21], [27]. The authors in [25] provide a detailed overview of the current status of research in the encrypted control for networked systems. Having said that, the implementation of algorithms in a HE setup does not come without challenges. For instance, some HE schemes, like the CKKS, use random noise to guarantee the security of the encrypted data. This noise compounds at every arithmetic operation, resulting in a limited number of sequential arithmetic operations performed by an encryption circuit, in special, multiplication operations. This is a known issue and many methodologies have been proposed to adapt traditional algorithms to a HE setup.

In this paper, we study homomorphic encryption on the context of Quadratic Programming (QP). QP has been a very successful tool for modeling many real-life problems. Several problems in physics or engineering can be formulated as some form of a minimum squared error problem. For example, QP is extensively used in financial applications in the formulation of portfolio optimization. In the area of control theory, it is commonly used in state estimation under minimum square error and model predictive control. To this matter, in [27] presented an encrypted model predictive control (MPC) scheme for linear constrained systems, showing that homomorphic encryption can be used to realize a secure and private cloud-based evaluation of an MPC if the corresponding piecewise affine control law is explicitly given. We propose and adapt gradient descent (GD) and accelerated gradient descent (AGD) algorithms to solve a QP in a HE way. Furthermore, we analyze the effect of the noisy HE channel in these descent algorithms. Once an encryption circuit is defined, the number of sequential arithmetic operations is fixed. The result is that in the HE setup, the AGD will perform fewer steps when compared to the GD. In the case that high accuracy of solution is required, and under limited computational memory resources, our results show that both the encrypted plain GD or the encrypted AGD may need to be considered. This is in contract to plain-text optimization, where AGD is typically superior. Additionally, we show that the condition number of the matrix of the quadratic term of the objective function plays an important role in determining which algorithm is preferred. For matrices with higher condition numbers, the AGD is preferred, for its faster convergence properties, even if performing fewer iterations. Moreover, we implemented our own HE matrix multiplication algorithm that is more efficient, in terms of multiplication depth, than other algorithms proposed in the literature.

II. Descent algorithms for unconstrained quadratic programming

Quadratic optimization problems in an unconstrained form has a closed form solution, however they involve the inversion of a matrix, a procedure that could not be implemented with only additions and multiplications, posing hence a challenge for its implementation in a HE setup. An alternative is to use gradient descent methods to solve the QP problem. This is a class of iterative algorithms that provide a simple way [5] to minimize a differentiable function f:

$$\min_{x\in\mathbb{R}^n} f(x).$$

The authors are with the Department of Engineering Science, University of Oxford, Oxford OX1 3PJ, U.K. E-mail: kostas.margellos@eng.ox.ac.uk, konstantinos.gatsis@eng.ox.ac.uk, andre.bertolace@eng.ox.ac.uk (corresponding author)

[1]Partially HE schemes enable the implementation of either addition or multiplication on encrypted data, but not both, whereas fully HE schemes enable the implementation of both addition and multiplication operations.

Starting at an initial estimate, it iterates the following until reaching a desired tolerance in the solution:

$$x_{t+1} = x_t - \eta \nabla f(x_t),$$

where $\nabla f(x_t)$ denotes the gradient of f calculated at x_t.
Methods of this type have a convergence rate which is independent of the dimension n of the solution space. This feature makes them particularly attractive for optimization in very high dimensions. In addition, properties such as smoothness or strong convexity of the objective function f do play a relevant role in choosing a variant of the algorithm with faster convergence. A detailed review of these and other methods to numerically solve convex optimization problems can be found in [5], [23].
For QPs, $f(x) = \frac{1}{2}x^T Q x + p^T x$, where $Q \in \mathbb{R}^{n\times n}$ and $p \in \mathbb{R}^n$. The QP is also convex if $Q \succcurlyeq 0$, while the gradient needed for the iterative implementation is $\nabla f(x) = Qx + p$.
It is possible to speed-up the convergence of gradient descent algorithms if the objective function satisfies certain conditions on the smoothness and convexity.
Definition 2.1: A continuous differentiable function f is β-smooth if the gradient ∇f is β-Lipschitz, i.e., $\|\nabla f(x) - \nabla f(y)\| \leq \beta \|x - y\|$, $\forall x, y \in \mathbb{R}^n$.
Definition 2.2: A function f is α-strongly convex, with $\alpha > 0$, if for any x, y it satisfies the following sub-gradient inequality, i.e., $f(x) - f(y) \leq \nabla f(x)^T (x - y) - \frac{\alpha}{2}\|x - y\|^2$, $\forall x, y \in \mathbb{R}^n$.
Given these definitions, an immediate consequence is that if f is twice differentiable, then f is α-strongly convex if the eigenvalues of the Hessian of f are larger than or equal to α. In fact, $f(x) = \frac{1}{2}x^T Q x + p^T x$ is λ_{max}-smooth and λ_{min}-strongly convex, where $\lambda_{max}, \lambda_{min} > 0$ are, respectively, the maximum and minimum eigenvalues of the matrix Q. The ratio $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$ is the condition number of the matrix Q and it plays an important role in the convergence of descent algorithms[2]. For reference, both Nesterov's accelerated gradient descent (AGD) and Gradient Descent (GD) methods [5] for the smooth and strongly convex quadratic function converge exponentially fast (Table I).

f	Algorithm	Convergence	Iterations
α-strong. conv., β-smooth	AGD	$R^2 \exp\left(-\frac{t}{\sqrt{\kappa}}\right)$	$\sqrt{\kappa}\log\left(\frac{R^2}{\epsilon}\right)$
α-strong. conv., β-smooth	GD	$R^2 \exp\left(-\frac{t}{\kappa}\right)$	$\kappa \log\left(\frac{R^2}{\epsilon}\right)$

TABLE I: Convergence rate and other parameters for different algorithms. $R = \|x_1 - x^*\|$ is the distance from the initial estimate to the optimal value. The number of iterations is directly derived from the convergence rate for a fixed tolerance (in terms of distance from optimal value) ϵ.

[2]Both algorithms have a multiplication depth that depends on the gradient $\nabla f(x) = Qx + p$. Only additions and multiplications are needed to solve the QP using both algorithms.

III. Homomorphically encrypted arithmetic

Among the many HE schemes, the CKKS scheme is often quoted as being the most efficient method to perform approximate HE computations over real and complex numbers [19]. We focus on the Residue Number System (RNS) variant of the CKKS scheme [6], the method implemented on Microsoft SEAL [28], the encryption library used in the implementation of the gradient descent methods to solve a QP problem.
The HE [8] cipher is defined by a pair E, D, of encryption decryption algorithms respectively. E takes a public key pk along with a message m as inputs and outputs a cipher-text c, as $\mathsf{c} = E(pk, m) = pk \cdot u + (m + e_0, e_1)$, where e_0, e_1 are random variables with values sampled from a discrete Gaussian distribution with variance σ^2, and u, another random variable, sampled from a ternary distribution $\{-1, 0, 1\}$ with probability $\rho/2$ for $+1$ and -1 and $1 - \rho$ for 0.
The decryption algorithm, D, takes a secret key sk along with the cipher-text c as inputs and outputs a noisy message $\tilde{m} = D(sk, \mathsf{c}) = e \cdot u + m + e_0 + s \cdot e_1$, where e is sampled form a discrete Gaussian distribution [15] of variance σ^2.
The security of the CKKS scheme is based on the Ring Learning With Errors (RLWE) problem, where Gaussian noise is introduced to achieve the desired hardness properties [8]. The key generation algorithm [8] for determining the pair sk and pk works by choosing s as a random signed binary vector, $\{0, -1, +1\}^N$, with hamming weight[3] h. The algorithm is parameterized by a security parameter λ which plays a direct role in determining the constant h. In addition, the parameter a in the RLWE is a random polynomial.
CKKS exploits the structure of integer polynomial rings for its plain-text and cipher-text spaces. For a power-of-two N, let us define the cyclotomic rings $\mathcal{R} = \mathbb{Z}[X]/(X^N + 1)$, $\mathcal{S} = \mathbb{C}[X]/(X^N + 1)$ together with a canonical embedding transformation $\sigma : \mathcal{S} \to \mathbb{C}^N$ that encodes/decodes a vector[4] in $\mathbb{C}^N$ to/from the ring of cyclonomical complex polynomials $\mathcal{S}$. The full process works as follows:

$$x \xrightarrow{\sigma^{-1}(\cdot)} \mu \xrightarrow{\lfloor \cdot \times \Delta \rceil} m \xrightarrow{E(pk,\cdot)} \mathsf{c} \xrightarrow{D(sk,\cdot)} \tilde{m} \xrightarrow{\cdot \div \Delta} \tilde{\mu} \xrightarrow{\sigma(\cdot)} \tilde{x}$$

To encode a message $x \in \mathbb{C}^N$ one applies the inverse embedding transformation to get $\mu = \sigma^{-1}(x) \in \mathcal{S}$, then scale μ by a factor $\Delta = 2^p$ and round to obtain the plain-text $m = \lfloor \Delta \cdot \mu \rceil \in \mathcal{R}$.
The CKKS scheme can be considered as a noisy channel [22]. A simple encryption/decryption procedure adds noise to the original message. We refer to [19] for a more detailed description of the approximation errors for the different operations in the different implementations of the CKKS scheme. Consider first the error due to the encryption/decryption procedure only. In this case, the noise is composed of the encoding error r_{encode} and the encryption error e_{fresh}, more precisely, $\tilde{m} = D(sk, E(pk, m)) = m + e_{fresh} = \Delta \cdot \mu + r_{encode} + e_{fresh} = \Delta \cdot \mu + \epsilon_{enc}$.

[3]The Hamming weight is defined as the number of bits different form zero. It is equivalent to the Hamming distance of the binary vector to the zero vector of same size.

[4]The space size is actually $N/2$ because the roots of the cyclonomic polynomial lie on the unit circle and are pairwise complex conjugate.

It makes sense to look at ϵ/Δ since $\tilde{\mu} := \tilde{m}/\Delta$ and one can always reduce the noise by increasing the Δ at the expense of computational power.

Operations which can be done homomorphically, such as addition and multiplication, add up to the approximated errors of both input cipher-texts. The result is an increased approximation error in the output cipher-text. For instance, the multiplication of two cipher-texts $\mathtt{c}_1$ and $\mathtt{c}_2$ results in higher noise level, $\mathtt{c}_1 \bullet \mathtt{c}_2 = (\Delta\mu_1 + \epsilon_1)(\Delta\mu_2 + \epsilon_2) = \Delta^2\mu_1\mu_2 + \overbrace{\epsilon_1\epsilon_2 + \Delta(\mu_1\epsilon_2 + \mu_2\epsilon_1)}^{\epsilon_\times}$.

Notice that the resulting cipher-text has scaled by Δ^2. For this reason, in the CKKS scheme [8], it is necessary to re-scale the result of the the multiplication of two cipher-texts in order to have a comparable scale Δ. The CKKS scheme introduces its own distinct and unique characteristic re-scaling procedure [8].

This is an issue for GD and AGD algorithms that have a large multiplication or addition depth. For algorithms that have a high multiplication depth, the noise grows at a much higher rate when compared to the addition case. To illustrate this, consider the addition and multiplication of 2 or 3 cipher-texts, with a normalized error, $\phi = \epsilon/\Delta$, as summarized in the following table.

Operation	Normalized error
$\mathtt{c} \mathbf{+} \mathtt{c}$	2ϕ
$\mathtt{c} \mathbf{+} \mathtt{c} \mathbf{+} \mathtt{c}$	3ϕ
$\mathtt{c} \bullet \mathtt{c}$	$\phi^2 + 2\mu\phi$
$\mathtt{c} \bullet \mathtt{c} \bullet \mathtt{c}$	$\phi^3 + 3\mu\phi^2 + 3\mu^2\phi$
$a \odot \mathtt{c}$	$a\phi$

TABLE II: Arithmetic operations noise

The operators have the following meaning:

- $\mathbf{+}$ represent the addition of two cipher-texts;
- $\bullet$ the multiplication of two cipher-texts;
- $\odot$ the multiplication of a plain-text and a cipher-text;

IV. Homomorphically encrypted gradient descent algorithms

A. Algorithm description

For the HE version of gradient descent methods, let us assume that the user calculates λ_{min}, λ_{max}, and sends these as plain-text, i.e. not encrypted, values to the solver. Together with these constants, the user also sends the encrypted matrix and vector, $\mathtt{Q} = E(pk, Q)$ and $\mathtt{p} = E(pk, p)$ that determine the QP. The encrypted version of the descent algorithms will still proceed in an iterative fashion. The only different is that one would be iterating over cipher-texts $\mathtt{c}_\mathtt{t}$ instead of plain-text x_t. When iterating over cipher-texts, two steps deserve special attention, the stopping criteria $|\mathtt{c}_{\mathtt{t}+1} \mathbf{-} \mathtt{c}_\mathtt{t}| > \epsilon$ and the matrix multiplication procedure (referred to MMult (Algorithm 4) discussed in the sequel). The latter is relevant because of the exponential growth of the noise level with the multiplication depth. The authors in [18] do propose a fully-homomorphic matrix multiplication algorithm to perform these operations in a more efficient manner. However, determining whether an encrypted value is larger than another encrypted value or even a plain-text without decrypting both values is not feasible under the usual security definitions.

Given the challenge to implement the stopping rule in an HE setup, the HE version of the AGD algorithm has to be slightly modified:

- Instead of specifying the tolerance ϵ, the user hands in the number of iterations N. With N and the convergence rate of the algorithms it is possible to infer the number of iterations needed to reach a given precision (Table I).
- The user may hand in the initial estimate $\mathtt{x}_0$, although this is not necessary.

The HE versions of the the AGD and GD still follow an iterative procedure. These take the form of (Algorithm 1) and (Algorithm 2) respectively and are very similar to the usual AGD and GD algorithms. The main difference is the use of HE arithmetic operators and the special MMult matrix multiplication procedure.

Algorithm 1 HE AGD for an unconstrained QP

1: **function** HEAGDQP($\mathtt{Q}, \mathtt{p}, d, \lambda_{min}, \lambda_{max}, \mathtt{x}_0, N$)
2: $\quad\kappa \leftarrow \frac{\lambda_{max}}{\lambda_{min}}$
3: $\quad\mathtt{x}_- \leftarrow \mathtt{x}_0$
4: $\quad\mathtt{y}_- \leftarrow \mathtt{x}_0$
5: $\quad\eta \leftarrow \frac{-1}{\lambda_{max}}$
6: $\quad$**for** $t = 0$ to $N - 1$ **do**
7: $\quad\quad\mathtt{y}_+ \leftarrow \mathtt{x}_- \mathbf{+} \text{MMULT}(\mathtt{Q}, \mathtt{x}_-, d, \eta) \mathbf{+} \eta \odot \mathtt{p}$
8: $\quad\quad\mathtt{x}_+ \leftarrow \left(1 + \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) \odot \mathtt{y}_+ \mathbf{-} \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \odot \mathtt{y}_-$
9: $\quad\quad\text{RELINEARIZE}(\mathtt{x}_+)$
10: $\quad\quad\mathtt{y}_- \leftarrow \mathtt{y}_+$
11: $\quad\quad\mathtt{x}_- \leftarrow \mathtt{x}_+$
12: $\quad$**end for**
13: $\quad$**return** $\mathtt{x}_+$
14: **end function**

Algorithm 2 HE GD for an unconstrained QP

1: **function** HEGDQP($\mathtt{Q}, \mathtt{p}, d, \lambda_{min}, \lambda_{max}, \mathtt{x}_0, N$)
2: $\quad\kappa \leftarrow \frac{\lambda_{max}}{\lambda_{min}}$
3: $\quad\mathtt{x}_- \leftarrow \mathtt{x}_0$
4: $\quad\eta \leftarrow \frac{-2}{\lambda_{min}+\lambda_{max}}$
5: $\quad$**for** $t = 0$ to $N - 1$ **do**
6: $\quad\quad\mathtt{x}_+ \leftarrow \mathtt{x}_- \mathbf{+} \text{MMULT}(\mathtt{Q}, \mathtt{x}_-, d, \eta) \mathbf{+} \eta \odot \mathtt{p}$
7: $\quad\quad\mathtt{x}_+ \leftarrow \mathtt{x}_-$
8: $\quad$**end for**
9: $\quad$**return** $\mathtt{x}_+$
10: **end function**

The only difference between the two algorithms is the presence of the extra two $\odot$ and one $\mathbf{-}$ operations on line 8 of the (Algorithm 1) when compared to (Algorithm 2). These operations are intrinsic to the accelerated gradient method as

the method uses additional past information to update to the next step.

B. Matrix multiplication seen differently

Because we would like to study gradient descent methods, let us consider a simple matrix multiplication. Halevi and Shoup [17] introduced a method (Algorithm 3) to evaluate an arbitrary linear transformation on encrypted vectors. They exploit the diagonal encoding of a matrix to easily express the matrix-vector multiplication by combining rotation and constant multiplication operations

Algorithm 3 Halevi-Shoup LINTRANS algorithm

function LINTRANS($\mathtt{c}, U$)
  $n \leftarrow dim(U)$
  $\mathtt{cU} \leftarrow \mathtt{c} \odot u_0$
  for $l = 1$ to $n - 1$ **do**
    $\mathtt{cU} \leftarrow \mathtt{cU} + \text{ROT}(\mathtt{c}, l) \odot u_l$
  end for
  RELINEARIZE($\mathtt{cU}$)
  return $\mathtt{cU}$
end function

In [18] the authors elaborated further on the Halevi-Shoup method and proposed a new matrix multiplication scheme (JKLS) that allows for a ciphered-matrix multiplication that uses only one cipher-text per matrix following a row-ordering encoding $A \rightarrow a$. Their algorithm performs the following calculations [5]

$$a_o = U_\sigma \odot a,\ b_o = U_\tau \odot b$$
$$a_k = V_k \odot a_0, b_k = W_k \odot b_0\ ,\ k = 1, \ldots, d-1$$
$$ab = \sum_{k=0}^{d-1} a_k \bullet b_k$$

Methodology	No. of Ciphertexts	Complexity	Mult. depth	Relinearizations
Halevi-Shoup	d	$\mathcal{O}(d^2)$	$1\bullet$	1
JKLS	1	$\mathcal{O}(d)$	$1\bullet + 2\odot$	3
Our work	1	$\mathcal{O}(d)$	$1\bullet + 1\odot$	2

TABLE III: Comparison of the different matrix multiplication algorithms.

Although convenient, the JKLS algorithm needs 2 $\odot$ operations [18]. We propose a modified version of a matrix multiplication algorithm with 1 less $\odot$ multiplication step:

$$a_k = V_k \odot a\ , b_k = W_k \odot b\ ,\ k = 0, \ldots, d-1$$
$$ab = \sum_{k=0}^{d-1} a_k \bullet b_k$$

with [6]:

$$V_k(d \cdot i + j, l) = \begin{cases} 1 & \text{if } l = d \cdot i + [i + j + k]_d \\ 0 & \text{otherwise} \end{cases}$$

[5]We refer to [18] for the definitions of V_k, W_k, U_σ and U_τ.

[6]Note that V_k and W_k in Algorithm 4 are not the same matrices as specified in [18].

$$W_k(d \cdot i + j, l) = \begin{cases} 1, & \text{if } l = d \cdot [i + j + k]_d + j \\ 0, & \text{otherwise} \end{cases}$$

where $[\cdot]_d$ is a shortcut for $\cdot$ modulo d.

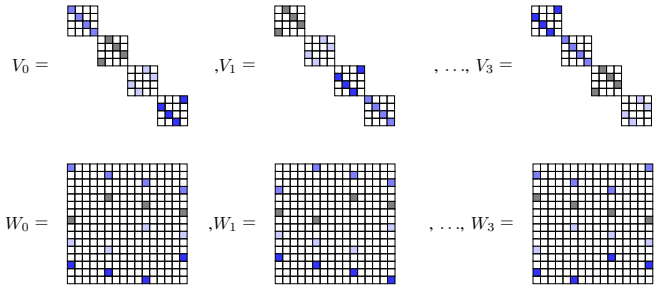

Fig. 1: Matrix multiplication - V_k and W_k examples for $d = 4$

The matrices V_k and W_k are permuting the row-encoded matrices A and B respectively such that the matrix multiplication algorithm as we know can be implemented with element-wise multiplication and additions. Table III summarizes the complexity differences of our method compared to Halevi-Shoup and JKLS.

Algorithm 4 HE Matrix Multiplication (MMULT)

function MMULT($\mathtt{A}, \mathtt{B}, d, a$)
  $\mathtt{AB} \leftarrow$ CIPHERTEXT()
  for $k = 0$ to $d - 1$ **do**
    $\mathtt{A_k} \leftarrow \text{LINTRANS}(\mathtt{A_0}, V_k(a))$ ▷ $\mathtt{A_k}$ scale: Δ
    $\mathtt{B_k} \leftarrow \text{LINTRANS}(\mathtt{B_0}, W_k(1))$ ▷ $\mathtt{B_k}$ scale: Δ
    $\mathtt{AB_k} \leftarrow \mathtt{A_k} \bullet \mathtt{B_k}$ ▷ $\mathtt{AB_k}$ scale: Δ^2
    RELINEARIZE($\mathtt{AB_k}$) ▷ $\mathtt{AB_k}$ scale: Δ
    $\mathtt{AB} \leftarrow \mathtt{AB} + \mathtt{AB_k}$
  end for
  return $\mathtt{AB}$
end function

V. NUMERICAL ANALYSIS

Prior to starting any computations, it is necessary to specify the capacity of the encryption circuit. In doing so, the user is implicitly specifying the multiplication depth, as these quantities are linked. A priory, it is possible to build circuits with as much multiplication depth as needed. However, in practical terms, the larger the capacity of a circuit, the larger the computing memory and computational power required at each arithmetic operation. Given our computing resources and the parameters of the Microsoft SEAL [28] library, the largest circuit we can implement has a multiplication depth of 18. At each iteration, AGD and GD have a multiplication depth of 3 and 2 respectively. The result is a cap of 6 and 9 steps for AGD and GD respectively.

In our first experiment, we consider a 2-dimensional QP with $x^* = (1, 1)$. We run both algorithms starting at the optimal value x^* for the same matrix Q and decrypted the outcomes of every iteration. The purpose is to analyze the bias introduced by the noise as described in Section III and

Table II. Indeed that is a noticeable bias responsible for the divergence from the optimal value (Figure 2)[7] but ideally we would expect the algorithm to stay at the optimal solution as a stationary point.

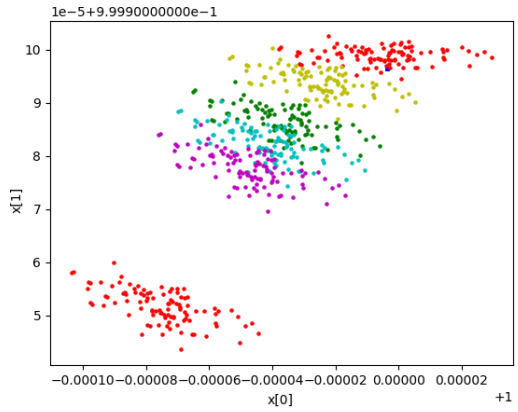

Fig. 2: Decrypted steps illustrating the deviation from the optimal value due to the bias for 100 repetitions HE-AGD with a 2-by-2 matrix with $\kappa = 2$ and optimal value at $x^* = (1, 1)$ (A similar behavior is observed for HE-GD).

We proceed by running both algorithms starting at initial condition $x_0 \neq x^*$ for the same matrix Q and decrypted the outcomes of every iteration. The numerical results in (Figure 3) show that at each iteration the solution gets closer to the optimal x^*. The faster convergence rate of the AGD

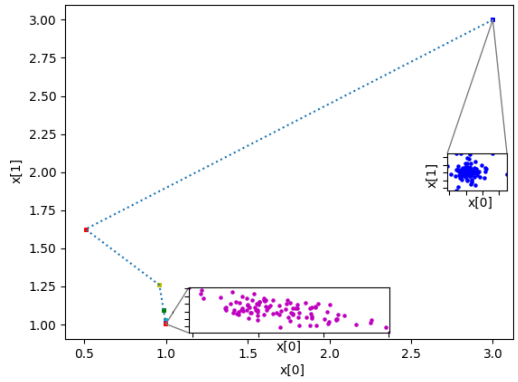

Fig. 3: Decrypted HE-AGD steps for 100 repetitions with a 2-by-2 matrix with $\kappa = 2$, optimal value at $x^* = (1, 1)$ initial condition $x_0 = (3, 3)$ (A similar behavior is observed for HE-GD).

algorithm makes it an attractive method for numerically solving our QP.

AGD exhibits a superior convergence rate, hence it allows meeting a given convergence (in terms of optimality) tolerance with fewer iterations. However, in case of encryption, the computational limits imposed by the allowable depth of the encryption circuit introduces a trade-off, as AGD involves more arithmetic operations compared to GD (see Section IV-A). As such, it might be computationally impossible to perform the number of iterations needed by AGD to meet a given tolerance. We investigate this trade-off numerically, and show that the preferred method depends on the condition number κ of the quadratic matrix Q.

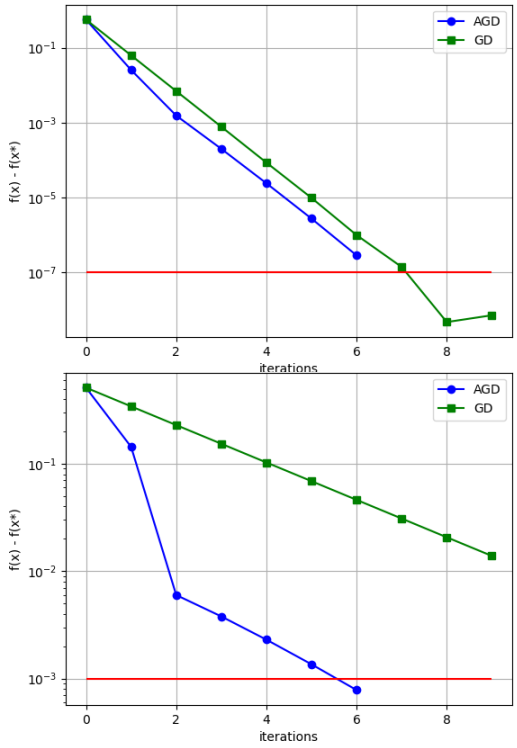

Fig. 4: Average convergence rate for 100 repetitions with a 2-by-2 matrix with $\kappa = 2$ (top) and $\kappa = 10$ (bottom). Optimal value at $x^* = (1, 1)$ and initial condition $x_0 = (2.0, 1.5)$.

Consider first the case of a low condition number, namely $\kappa = 2$. Our analysis is depicted in Figure 4 (upper panel), where we consider an optimality tolerance of $\epsilon = 10^{-7}$ (solid red line). For our encryption circuit, we can perform only 6 iterations of AGD prior to running into memory problems. This implies that within this number of iterations the solution returned by AGD does not meet the desired tolerance. In contrast we can perform more iterations with GD (due to the fewer operations required) thus meeting the desired tolerance despite the fact that the method is slower compared to AGD. For $\kappa = 10$ the situation is different (see Figure 4 (lower panel)), as AGD is sufficiently faster compared to GD so that within the limit of 6 iterations imposed by our circuit the returned solution has met the tolerance set for this case (note that here we changed the tolerance to $\epsilon = 10^{-3}$ as the convergence rate is lower with the higher conditional number of the matrix). This observation, i.e., the dependency of the preferred method depending on the condition number, extends to other quadratic examples.

For the numerical examples presented here we used our own Python wrapper of the Microsoft SEAL [28] C++ library and run the code in an Intel Xeon E5-1620 with 24GB of RAM

[7]Each iteration is marked in a different color (step 0: red, step 1: yellow, ...); for every color cloud, each dot corresponds to another run; all runs start at x^* and follows a different path due to the "noisy channel" behavior of the CKKS scheme.

running Debian 11. All the code for the implementation of the numerical results are available on Github:

Wrapper: https://github.com/b3rt01ac3/pSEAL
Gradient descent: https://github.com/b3rt01ac3/agd-he

VI. Conclusion

In this paper we studied, implemented and analyzed both gradient and accelerated gradient descent algorithms to solve a QP problem in a HE manner. In our implementation the accelerated gradient descent method takes an extra multiplication operation at each step when compared to the gradient descent method. This means that for an encryption circuit with same security parameters and channel capacity (i.e. multiplication depth), the accelerated gradient descent method can never be run for as many steps as the gradient descent one. We show that the condition number of the matrix of the quadratic term of the objective function plays an important role in determining the which algorithm is preferred. For higher values, the accelerated gradient descent is preferred, as it converges faster to the solution, even if performing less iterations. Whereas for lower values of the condition number of the matrix the gradient descent performs better due to the extra iterations. In addition, we proposed a new HE matrix multiplication algorithm that is more efficient, in terms of multiplication depth, than other algorithms proposed in the literature. These observations have been verified by means of a numerical study. Future work includes the investigation of the use of other HE schemes such as BFV, expand the solution to cover constrained QP problems and the furhter adapting of algorithms to reduce the multiplication depth.

References

[1] Andreea B. Alexandru, Konstantinos Gatsis, Yasser Shoukry, Sanjit A. Seshia, Paulo Tabuada, and George J. Pappas. Cloud-based quadratic optimization with partially homomorphic encryption. *IEEE Transactions on Automatic Control*, 66(5):2357–2364, 2021.

[2] Andreea B. Alexandru, Anastasios Tsiamis, and George J. Pappas. Encrypted distributed lasso for sparse data predictive control, 2021.

[3] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[4] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[5] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3–4):231–357, November 2015.

[6] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. A full rns variant of approximate homomorphic encryption. Cryptology ePrint Archive, Paper 2018/931, 2018. https://eprint.iacr.org/2018/931.

[7] Jung Hee Cheon, Kyoohyung Han, Hyuntae Kim, Junsoo Kim, and Hyungbo Shim. Need for controllers having integer coefficients in homomorphically encrypted dynamic system. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5020–5025, 2018.

[8] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[9] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[10] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. https://eprint.iacr.org/2012/144.

[11] Farhad Farokhi, Iman Shames, and Nathan Batterham. Secure and private cloud-based control using semi-homomorphic encryption. *IFAC-PapersOnLine*, 49(22):163–168, 2016. 6th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2016.

[12] Farhad Farokhi, Iman Shames, and Nathan Batterham. Secure and private control using semi-homomorphic encryption. *Control Engineering Practice*, 67:13–20, 2017.

[13] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[14] Craig Gentry. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105, March 2010.

[15] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Paper 2007/432, 2007. https://eprint.iacr.org/2007/432.

[16] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[17] Shai Halevi and Victor Shoup. Faster homomorphic linear transformations in helib. Cryptology ePrint Archive, Report 2018/244, 2018. https://ia.cr/2018/244.

[18] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 1209–1222, New York, NY, USA, 2018. Association for Computing Machinery.

[19] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. Approximate homomorphic encryption with reduced approximation error. Cryptology ePrint Archive, Report 2020/1118, 2020. https://ia.cr/2020/1118.

[20] Junsoo Kim, Chanhwa Lee, Hyungbo Shim, Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Encrypting controller using fully homomorphic encryption for security of cyber-physical systems. *IFAC-PapersOnLine*, 49(22):175–180, 2016. 6th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2016.

[21] Shane Kosieradzki, Yingxin Qiu, Kiminao Kogiso, and Jun Ueda. Rewrite rules for automated depth reduction of encrypted control expressions with somewhat homomorphic encryption. In *2022 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 804–809, 2022.

[22] Yongwoo Lee, Joonwoo Lee, Young-Sik Kim, HyungChul Kang, and Jong-Seon No. High-precision and low-complexity approximate homomorphic encryption by error variance minimization. Cryptology ePrint Archive, Report 2020/1549, 2020. https://ia.cr/2020/1549.

[23] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, 2nd edition, 2006.

[24] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[25] Moritz Schulze Darup, Andreea B. Alexandru, Daniel E. Quevedo, and George J. Pappas. Encrypted control for networked systems: An illustrative introduction and current challenges. *IEEE Control Systems Magazine*, 41(3):58–78, 2021.

[26] Moritz Schulze Darup, Adrian Redder, and Daniel E. Quevedo. Encrypted cooperative control based on structured feedback. *IEEE Control Systems Letters*, 3(1):37–42, 2019.

[27] Moritz Schulze Darup, Adrian Redder, Iman Shames, Farhad Farokhi, and Daniel Quevedo. Towards encrypted mpc for linear constrained systems. *IEEE Control Systems Letters*, 2(2):195–200, 2018.

[28] Microsoft SEAL (release 3.7). https://github.com/Microsoft/SEAL, September 2021. Microsoft Research, Redmond, WA.