Transactions on Computational and Applied Mathematics (2024) **DOI: 10.23977/tracam.2024.040101**
Clausius Scientific Press, Canada **ISSN 2616-1826 Vol. 4 Num. 1**

Linear Congruence and Reduction on the Learning with Errors Problem

Lanxuan Xia

St. Mark's School, 25 Marlboro Road Southborough, Massachusetts, United States
lauraxia8@gmail.com

Keywords: Linear Diophantine equations, linear congruence, LU decomposition, LWE, lattice-based cryptography, matrices

Abstract: We propose an algorithm to solve general linear Diophantine equations and an algorithm to solve linear congruence problems efficiently using LU decomposition, which means unsafety of cryptography systems based on linear congruence equations. Thus, we focus on the generalization of the argument for a specific reduction of the Learning with Error (LWE) problem established in a previous work ([BLP13]) so that LWE can accommodate for more general choices of matrices. More specifically, we relaxed [BLP13]'s constraint on the choice of the identity matrix to general diagonal matrices. Two examples are presented here to show the validity of our results further.

1. Introduction

Many classic cryptography systems rely on the hardness to solve certain problems, such as large integer factorization, discrete logarithm, and elliptic curves to ensure information security. One of such problems of topic in our paper is solving the linear congruence equation $ax \equiv b(mod m)$ for large values of m. Linear congruence serves as the basis of many cryptographic algorithms, including the Rivest Shamir Adleman (RSA) encryption and decryption algorithms.

As a result, it is of interest for mathematicians to develop efficient algorithms for solving linear congruence. Some existing methods of solving linear congruence include root testing on the residue classes $mod m$, finding $a^{-1}(mod m)$, and converting the equation to the corresponding linear Diophantine equation $ax + cm = b$ and then applying the Euclidean Algorithm. In 1980 John R. Silvester also describes a novel method of combining matrix elementary row-operation with solutions to linear congruence equations.[1]

However, all these methods fell short when it comes to a system of linear congruence equations, namely those $Ax \equiv b(mod m)$ with A being a n-dimensional full rank matrix, having significantly large values of n and m. For these cases, we propose an algorithm that utilizes LU decomposition, taking advantage of the known efficient algorithms for calculating LU decomposition.

With more and more efficient algorithms for computing linear congruence, it is essential for mathematicians to devise new problems with a hardness level stronger than the current problems. Following the introduction of quantum computers, lattice-based cryptography has become increasingly important for post-quantum cryptography. Different from the classic encryption methods such as RSA and Diffie-Hellman, lattice-based cryptography is currently believed to be theoretically not breakable by quantum computers—that is, under the assumption that specific computational lattice problems have no polynomial time algorithm solutions.

In our paper, we will focus on a fundamental problem which most lattice-based cryptographic systems are based upon: the learning with error (LWE) problem. Two distributions are concerned in the problem:

$$((a_i, < a_i, s > + e_i mod q))_i and ((a_i, u_i))_i$$

where the s and a_i are chosen uniformly at random from $\mathbb{Z}_q^n$, $u_i$uniformly at random from$\mathbb{Z}_q$, and $e_i \in \mathbb{Z}$ from an error distribution which is typically a discrete Gaussian distribution. The decision variant of LWE aims to distinguish between the two distributions, while the search version aims to find the secret s.

Regev's groundbreaking work in 2005 shows a reduction from lattice problems such as finding relatively short vectors in general lattices to LWE, proving that the latter is a hard problem.[2] More specifically, it turns out a solving algorithm for n-dimensional LWE implies an efficient quantum algorithm to find short enough vectors in n-dimensional lattices.

However, since Regev's work only suggests the existence of an efficient quantum algorithm from any, even non-quantum, algorithms for LWE, this reduction is not the end point of such reductions. Other works thus succeed Regev's work in an attempt to find a better, namely classical (non-quantum) reductions. C. Perkert's 2005 work shows exponential modulus LWE can be classically reduced to standard lattice problems.[3] Z. Brakerski et al.'s 2013 work establishes a classical reduction on polynomial modulus LWE.[4] More importantly, it shows that the existence of an efficient classical algorithm for LWE implies an efficient algorithm for worst-case standard lattice problems.

In short, many reductions upon different versions of LWE have been established thus far, all with their own strengths and weak points. They all however, aim to either study the hardness of LWE or to adjust the problem to be more practical in the actual applications of cryptography.

The main contribution of this paper is that we provided an algorithm for solving linear congruence equations using LU decomposition and justify its efficiency by running the algorithm on large matrices and modules on MATLAB.

For the reduction on LWE, we provide here a more general result based on the conclusions reached in Z. Brakerski et al.'s 2013 work. The results can accommodate more matrices than corresponding results as stated by Brakerski et al.

Our algorithm for solving linear congruence will be preceded by a method of solving general linear Diophantine equations, some illustrations of that method, and will then be followed by three examples of linear congruence equations solved using our algorithm that will demonstrate the amount of efficiency our algorithm can achieve.

In order to obtain our results for the reduction on LWE, we used the same idea as the one in the proof of Corollary 3.4 by Brakerski et al. Moreover integer diagonal matrices other than the identity matrix which is used in the corollary 3.4 are also employed here. At last, the results that more generalized matrices can be used here are done in this paper.

2. Preliminaries

In this section, some useful preliminaries are presented.

2.1 Linear Diophantine Equation

The general solvable linear Diophantine equation has form:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

Where $a_1, a_2, \cdots, a_n$ and b being integers, with $(a_1, a_2, \cdots, a_n)|b$ being the necessary and sufficient condition for which there exist integer solutions for the equation. We call a linear Diophantine equation solvable if and only if there exist integer solutions, since those are the only solutions of interest.

2.2 Linear Congruence Equations

For A a n-dimensional full rank matrix, x and b n-dimensional vectors, and m an integer. The linear congruence equation of interest in this paper is denoted:

$$Ax \equiv b(modm)$$

Where the necessary and sufficient condition for there to exist integer solutions is for $(det(A), m)|b$. For convenience sake we will denote the determinant of A (commonly denoted as $det(A)$) only as $|A|$ from now on.

2.3 LU Decomposition

For any given full rank n-dimensional matrix A, the LU decomposition of it finds $LU = A$ where L, U are lower triangular matrix and upper triangular matrix respectively. The decomposition can be achieved efficiently by repeatedly adding a multiple of one row of A to another.

2.4 Lattice

An n-dimensional lattice Λ is a subgroup of $\mathbb{R}^n$, and is the set of all integer coefficient linear combinations of some n linearly independent vectors called the set of basis vectors B:

$$\Lambda = \mathcal{L}(B) = \{ \sum_{i=1}^{n} c_i b_i \mid c_i \in \mathbb{Z}^n \}$$

2.5 Gaussian Distributions

We define n-dimensional Gaussian function ρ_r for $r > 0$ as

$$\rho_r(x) := exp(-\pi||x||^2/r^2).$$

Then the continuous Gaussian distribution D_r is defined to be the distribution with density function proportional to ρ_r.

Also define for n-dimensional lattice Λ and vector $u \in \mathbb{R}^n$, $D_{\Lambda+u,r}$ to be the discrete distribution with support on coset $\Lambda + u$ with probability mass function proportional to ρ_r.

2.6 Learning with Errors

Let $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ be the additive group of reals under modulo 1, and $\mathbb{T}_q$ be $\mathbb{T}$'s subgroup given by $\{0, 1/q, \ldots, (q-1)/q\}$.

For integers $n, q \geq 1$, integer vector $s \in \mathbb{Z}^n$, and probability distribution ϕ on $\mathbb{R}$, define $A_{q,s,\phi}$ to be the distribution over $\mathbb{T}_q^n \times \mathbb{T}$ acquired by selecting $a \in \mathbb{T}_q^n$ uniformly at random and choosing an error term e from ϕ, outputting $(a, b = \langle a, s \rangle + e) \in \mathbb{T}_q^n \times \mathbb{T}$.

Lemma 2.1 ([BLP13, Theorem 3.1]) *Let $m, n, q, n', q' \geq 1$ be integers, let $G \in \mathbb{Z}^{n' \times n}$ satisfy that $\Lambda = \frac{1}{q'} G^T \mathbb{Z}^{n'} + \mathbb{Z}^n$ has a known B as a basis, and let $\mathcal{D}$ be a certain (B, δ)-bounded distribution over $\mathbb{Z}^n$. Let $\alpha, \beta > 0$, $\varepsilon \in (0,1/2)$ be so that*

$$\beta^2 \geq \alpha^2 + (4/\pi) ln(2n(1 + 1/\varepsilon)) \cdot (max\{q^{-1}, ||B||\} \cdot B)^2.$$

Then exist a transformation reduction from $LWE_{n,m,q,\leq\alpha}(\mathcal{D})$ to $LWE_{n',m,q',\leq\beta}(G \cdot \mathcal{D})$ which reduces the advantage by $\leq \delta + 14\varepsilon m$.

Lemma 2.2 ([BLP13, Corollary 3.4]) *For $n, m, q \geq 1, k \geq 1, \alpha, \beta > 0$ a divisor of $n, \mathcal{D}$ a (B, δ)-bounded distribution over $\mathbb{Z}^n$, and $\varepsilon \in (0,1/2)$ for which*

$$\beta^2 \geq \alpha^2 + (4/\pi) ln(2n(1 + 1/\varepsilon)) \cdot (B/q)^2,$$

Then exist a transformation reduction from $LWE_{n,m,q,\leq\alpha}(\mathcal{D})$ to $LWE_{n/k,m,q^k,\leq\beta}(G \cdot \mathcal{D})$ which reduces the advantage by $\leq \delta + 14\varepsilon m$, with $G = I_{n/k} \otimes (1, q, q^2, \ldots, q^{k-1})^T$.

3. Linear Diophantine Equations

The main result of this section is Theorem 3.1, which is a method of solving general solvable linear Diophantine Equations using integral elementary row operations. Elementary row operations include swapping rows, scalar multiplication on rows, and row sum, which is the act of adding multiples of one row to another. For our case, we will be using integral row sums, which is when the multiples of the row is strictly an integral multiple. We use $u \cdot v$ for vectors u and v to denote the dot product of these two vectors.

Theorem 3.1. *For integers $a_1, a_2, \cdots, a_n$ and b, let $(a_1, a_2, \cdots, a_n) = g$. If $g|b$, perform integral row sums on the following matrix until g appears in the first entry in a certain row i:*

$$A = \begin{pmatrix} a_1 & m & & & \\ a_2 & & m & & \\ \vdots & & & \ddots & \\ a_n & & & & m \end{pmatrix}$$

Where the leftmost column are the entries a_1 to a_n and the next n columns form the matrix $m \cdot I_n$, where $g \cdot m = b$. Suppose the entries of row i after the operation are $(gc_1c_2 \cdots c_n)$, then the solutions to the linear Diophantine equation $a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$ are $x_1 = c_1 x_2 = c_2 \cdots x_n = c_n$.

Proof. Note that for each step of the integral row sum, A is multiplied by a $n \times n$matrix with determinant equating 1. Thus, the matrix after the operation is $A \cdot B$ for a $n \times n$matrix B with $|B| = 1$. Let row i of B be $(b_1 b_2 \cdots b_n)$, then row i of $B \cdot A$ has first entry: $(b_1 b_2 \cdots b_n) \cdot (a_1 a_2 \cdots a_n)$, the other n entries be $(mb_1 mb_2 \cdots mb_n)$. Since we picked row i such that $(b_1 b_2 \cdots b_n) \cdot (a_1 a_2 \cdots a_n) = g$, then $(mb_1 mb_2 \cdots mb_n) \cdot (a_1 a_2 \cdots a_n) = m \cdot (b_1 b_2 \cdots b_n) \cdot (a_1 a_2 \cdots a_n)$
$= mg = b$. So $(mb_1 mb_2 \cdots mb_n)$, which is $(c_1 c_2 \cdots c_n)$ under our notation, are the solutions $(x_1 x_2 \cdots x_n)$ to the linear Diophantine equation.

Next we will be providing an example of solving linear Diophantine equations with the method we proposed.

Example 3.2. We attempt to solve the linear Diophantine equation $22x + 48y = 8$ using Theorem 3.1. Build matrix A as needed:

$$\begin{pmatrix} 22 & 4 & 0 \\ 48 & 0 & 4 \end{pmatrix}$$

Then perform the elementary integral row-operations as described in Theorem 3.1:

$$\begin{pmatrix} 22 & 4 & 0 \\ 48 & 0 & 4 \end{pmatrix} \to \begin{pmatrix} 22 & 4 & 0 \\ 4 & -8 & 4 \end{pmatrix} \to \begin{pmatrix} 2 & 44 & -20 \\ 4 & -8 & 4 \end{pmatrix} \to \begin{pmatrix} 2 & 44 & -20 \\ 0 & -96 & 44 \end{pmatrix}$$

Since $(22,48) = 2$, we take the row $(244 - 20)$, by Theorem 3.1 the solution to $22x + 48y = 8$ is then $x = 44, y = -20$. A quick check shows that indeed $22 \cdot 44 + 48 \cdot (-20) = 8$.

4. LU Decomposition Method for Linear Congruence Equations

The main theorem of this section which utilizes LU decomposition to solve linear congruence equations is listed below. Denote the adjoint matrix of any matrix A to be $A^{\star}$, for which $A^{\star} = |A| \cdot A^{-1}$.

Theorem 4.1. *For full rank n-dimensional matrix A, vector b, and module m, let $(|A|, m) = g$. If $g|b$, then the general solution for x is:*

$$x \equiv (\frac{|A|}{g})^{-1}|A|A^{-1}\frac{b}{g} + \frac{m}{g}k(modm)k \in \mathbb{Z}, k = 0,1,\cdots, g-1$$

Proof. The calculation process that directly leads to the result is listed below.

$$AA^{\star}x \equiv A^{\star}b(modm)$$

$$|A|x \equiv A^{\star}b(modm)$$

$$\frac{|A|}{g}x \equiv A^{\star}\frac{b}{g}(mod\frac{m}{g})$$

$$x \equiv (\frac{|A|}{g})^{-1}A^{\star}\frac{b}{g}(mod\frac{m}{g})$$

$$x \equiv (\frac{|A|}{g})^{-1}|A|A^{-1}\frac{b}{g}(mod\frac{m}{g})$$

$$x \equiv (\frac{|A|}{g})^{-1}|A|A^{-1}\frac{b}{g} + \frac{m}{g}k(modm)k \in \mathbb{Z}, k = 0,1,\cdots, g-1$$

The below corollary is a special case of Theorem 4.1 that follows directly from the proof above.

Corollary 4.2. *For full rank n-dimensional matrix A, vector b, and module m, if $(|A|, m) = 1$, then the general solution for x is unique:*

$$x \equiv |A|^{-1}|A|A^{-1}b(modm)$$

5. Numerical Tests

In this section we will provide three cases of the application of Theorem 4.1, the first case with a small value of n where n is the dimension of matrix A, the second case with large value of$n$ but still with $|A|$ coprime with m, while the last case is most complicated, with $(|A|, m) \neq 1$.

In order to test the efficiency of our algorithm presented in Theorem 4.1, we do these tests on

MATLAB to demonstrate its level of efficiency, and the corresponding computer is equipped with 8-core cpu and 8GB memory. The following three examples each have different levels of complexity yet are all based on our algorithm.

Example 5.1. For small values of n where n is the dimension of matrix A in the linear congruence equation $Ax \equiv b(modm)$, we restrict $|A|$ to be coprime to m, and base our example upon the results of Corollary 4.2.

For runtime efficiency testing purposes, we build the equations as follows. We first assign the solution x to be the n-dimensional vectors of all ones. Then we construct A by first constructing matrices L, U that are lower and upper triangular matrices such that $L \cdot U = A$. Let the main diagonal of U be of all ones except for the last element, which for our case we set it equal to 7. Then let the main diagonal of L be of all ones, Lastly we let the other possible nonzero elements of both L and U be integers uniformly selected at random from $[0,3]$. This selection was made possible by first selecting those elements randomly from the real numbers between 0 and 1, then multiplying each element by 3 and rounding them to the nearest integer. Note that since the determinant of triangular matrices is just the product of the elements of the diagonal, $|U| = 7, |L| = 1$, giving $|A| = 7$. In fact, our choice for the matrices L and U is to ensure we know the determinant of A and we can control the condition number of A. In our tests, let $m = 5$ where m is the modules in $b \equiv Ax(modm)$. Afterwards we compute $|A|^{-1} \equiv 7^{-1} \equiv 3(mod5)$, then lastly we use the results of Corollary 4.2 to compute $x \equiv 3 \cdot 7 \cdot A^{-1}b(mod5)$.

We run this code 1000 times and saved the runtime, condition number of A, and the norm of the difference between the achieved answer x and the expected answer e of each run. The latter data set consists of small real numbers > 0, specifically for our run the range of the data set is $[6.5034 \cdot 10^{-11}, 0.0590]$, with a mean of $3.6 \cdot 10^{-4}$. Although those numbers are not exactly 0, as it is supposed to be, we are only concerned about integer solutions, and these errors are small enough to be all rounding to 0 instead of 1. Keeping this in mind, we also save the norm of the difference between the rounded version of x and the expected answer e. This resulting data set is unsurprisingly of all 0s.

The datasets for runtime and condition numbers of A for different values of n (each for 1000 tests) can be seen in the table below(Table 1).

Table 1: Results of Example 5.1.

n Data	$n = 10$	$n = 15$	$n = 20$	$n = 25$
Max runtime	0.0210	0.0401	0.0096	0.0146
Min runtime	$1.4208 \cdot 10^{-5}$	$1.20 \cdot 10^{-5}$	$3.4666 \cdot 10^{-5}$	$5.2375 \cdot 10^{-5}$
Average runtime	$6.4240 \cdot 10^{-5}$	$6.4240 \cdot 10^{-5}$	$6.6944 \cdot 10^{-5}$	$9.4918 \cdot 10^{-5}$
Min condition #	1	1	1	1
Max condition #	$1.3299 \cdot 10^{7}$	$3.5225 \cdot 10^{9}$	$7.1558 \cdot 10^{12}$	$2.6156 \cdot 10^{13}$
Average condition #	$2.4785 \cdot 10^{5}$	$3.3985 \cdot 10^{7}$	$1.200 \cdot 10^{10}$	$3.5861 \cdot 10^{11}$

As shown, values for runtime are small enough for us to deem our algorithm efficient for small dimensions of A such as in this example.

Regarding the condition number of A. Since A is a relatively small matrix with minimum $n = 10$ and maximum $n = 25$, these condition numbers are high enough to deem A to be a very ill-conditioned matrix in every value of n. However, the resulting norm of the difference of the achieved solution x and the expected solution e is still low enough to be acceptable under the condition that we only care about the discrete integer solutions. This shows the stability of our algorithm, as it remains accurate when a small matrix A is very ill-conditioned.

Example 5.2. For this example matrix A of higher dimensions are concerned, namely matrices of dimension $n = 30000$. Again for this example we restrict $|A|$ to be coprime with the module m, so this example is still based on the results of Corollary 4.2, but on a larger scale.

Similar to the previous example, we build this example as follows. The expected solution for x is still the n-dimensional vector of all 1s. Then L is constructed to have main diagonal with all 1 entries and the diagonal immediately under the main diagonal with also all 1 entries. U on the other hand is the matrix with main diagonal with all 1s except the last entry, which is assigned to be 5, and the diagonal immediately above the main diagonal has all 1 entries. A is then $L \cdot U$. Consequently, we have $|L| = 1, |U| = 5 \Rightarrow |A| = |L| \cdot |U| = 5$. Again we assign L and U this way so that the determinant of A is known. Next we set $m = 7$ and calculate $b \equiv Ax(modm)$. This time we ask the code to find $|A|^{-1}(modm)$, and since $m = 7$ has a relatively small multiplicative group, we simply run the code to traverse through all residue classes $mod7$ to find that $3 \cdot |A| \equiv 1(modm)$. Finally we conclude that $x \equiv 3 \cdot 5 \cdot A^{-1} \cdot b(modm)$.

We run this code 1000 times again, saving the runtime, condition number of A, and the norm of the difference between the achieved solution x and the expected solution e. This time since L and U are completely determined, the latter data is the same each run, and for our run the result is $2.1755 \cdot 10^{-13}$. This is again a small number concerning the fact that we only care about integer solutions. Therefore, we again calculate the rounded value of x and is not surprised to see that it is 0 every test.

The datasets for runtime and condition numbers of A for different values of n (each for 1000 tests) can be seen in the table below (table 2).

Table 2: Results of Example 5.2.

n Data	$n = 100$	$n = 200$	$n = 300$	$n = 30000$
Max runtime	0.0657	0.0462	0.0085	0.1398
Min runtime	$2.50 \cdot 10^{-4}$	$3.17 \cdot 10^{-4}$	$4.00 \cdot 10^{-4}$	0.0227
Average runtime	$4.17 \cdot 10^{-4}$	$4.47 \cdot 10^{-4}$	$5.36 \cdot 10^{-4}$	0.0249
Condition #	$3.50 \cdot 10^{4}$	$1.40 \cdot 10^{5}$	$3.14 \cdot 10^{5}$	$3.15 \cdot 10^{9}$

Although we see quite a significant increase in average runtime from the previous example to this example, it is acceptable considering the drastic increase in n. Thus, we conclude that the algorithm is still efficient when it comes to sizes of n as large as $n = 30000$. Note that although n is larger in this example, the norm of the difference between the achieved solution x and the expected solution e is smaller in this example than the previous, this is due to the fact that the previous example involved randomly generated L and U while this example has known simple cases of L and U.

As shown above, the condition numbers for each case are not that large of numbers considering A having dimension up to $n = 30000$. Thus, we are working with a well-conditioned matrix here.

Example 5.3. In this example we work with the most complicated case, with a large dimension of A, namely $n = 30000$, and moreover $(|A|, m) > 1$. As a result we will be employing Theorem 4.1 directly for this example.

The buildup process for this example is almost identical to the previous example. The solution x is again assigned to be the n-dimensional vector of all 1s, L is the same matrix as in the previous example with the main diagonal and the diagonal immediately under it being of all 1s and everything else being 0, whilst U is the matrix with the main diagonal having all 1 entries except for the last entry being 6 and the diagonal immediately above it having all 1 entries. In this example we assign $m = 8$. Note that $|A| = |L| \cdot |U| = 6$, and $(|A|, m) = 2 > 1$. Lastly we compute $b \equiv Ax(modm)$. Then we calculate $\frac{|A|}{g}, \frac{m}{g}$, and $\frac{b}{g}(mod\frac{m}{g})$ with $g = (|A|, m) = 2$, also finding $(\frac{|A|}{g})^{-1}(mod\frac{m}{g})$ using the same process as described in the previous example, which is iterating through the equivalence classes $mod\frac{m}{g}$. Finally, by Theorem 4.1, $x \equiv (\frac{|A|}{g})^{-1}|A|A^{-1}\frac{b}{g} \equiv 3 \cdot 6 \cdot A^{-1}\frac{b}{2}(mod\frac{m}{2})$

This code is runned 1000 times again, saving runtime and the norm of the difference between the achieved solution x and the expected solution e. The latter is a set value just as in the previous example, and for our run the value turns out to be $1.6316 \cdot 10^{-13}$. Again this value is low enough to be rounded to 0 since we only care about integer solutions.

The datasets for runtime and condition numbers of A for different values of n (each for 1000 tests) can be seen in the table below(table 3).

Table 3: Results of Example 5.3.

n Data	$n = 100$	$n = 200$	$n = 300$	$n = 30000$
Max runtime	0.0342	0.0124	0.0100	0.1843
Min runtime	$2.55 \cdot 10^{-4}$	$3.21 \cdot 10^{-4}$	$4.10 \cdot 10^{-4}$	0.0227
Average runtime	$3.34 \cdot 10^{-4}$	$3.92 \cdot 10^{-4}$	$5.36 \cdot 10^{-4}$	0.0250
Condition #	$3.97 \cdot 10^{4}$	$1.59 \cdot 10^{5}$	$3.59 \cdot 10^{5}$	$3.60 \cdot 10^{9}$

The results for runtime in this example is similar to the result in the previous example, so it's safe to conclude that the algorithm is also efficient for large values of n and $(|A|, m) > 1$.

The condition numbers for A as can be seen above are also not large considering A has dimension $n = 30000$. Thus, in this example A is also well-conditioned.

Discussion of Results. As we have shown, the algorithm works for cases of small and large n regardless of the value of $(|A|, m)$. However, for all these above examples, note that we forcefully manipulated L and U so that $|A|$ is a known small value. However, the algorithm fails when $|A|$ is a random large value, large values of $|A|$ enlarges any existing error. When $|A|$ is on the level of 10^{11}, which can easily happen when the diagonals of both U and L are randomly generated, errors that were insignificant and negligible in cases of small $|A|$ become enlarged to a significantly large error. Other things that can affect the accuracy of the algorithm are the dimension of A and the condition number of A. Namely, the accuracy of the algorithm decreases when A is a significantly ill-conditioned matrix. Our algorithm is decently stable though since Example 5.3 shows it to be resistant to very ill-conditioned matrices as long as the dimensions of A is small enough. When high dimensions of A is combined with large condition numbers of A, the algorithm succumbs and returns high error. For example, we twist the code for Example 5.3 slightly and increase the dimension to $n = 300$. One run shows the condition number to be $6.4582 \cdot 10^{18}$, which means A is very ill-conditioned in this case. Under the condition that A is a high dimension ill-conditioned matrix, the norm of the difference between the achieved solution x and the expected solution e is as high as 36.0918, meaning the algorithm fails short in this case.

In conclusion, our algorithm works in cases of A with small determinants, and either having a small dimension or being well-conditioned. It fails however, when A has a large determinant or when it is both large and ill-conditioned.

6. Reduction

As our algorithm to solve linear congruence equations in the previous sections shows, problems previously thought to have strong hardness levels for which classical cryptography systems were based are starting to become increasingly breakable. Thus, it is of high importance for researchers to devise new problems that are currently unbreakable, especially under the circumstances that we are entering the post-quantum world.

The main result of this section is the following theorem, which is a special case of Lemma 2.1, and a generalization of Lemma 2.2. Let A_n be a diagonal matrix of dimension $n \times n$, and $max(A_n)$ be the largest element in the matrix, i.e.

$$max(A_n) = a_{mm} for 1 \leq m \leq n such that \nexists 1 \leq r \leq n with a_{rr} > a_{mm}.$$

$$\boldsymbol{A_n} = \begin{pmatrix} a_{11} & \cdots & & \\ \vdots & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix}$$

Theorem 6.1. *For $n, m, q \geq 1, k \geq 1, \alpha, \beta > 0$ a divisor of $n, \mathcal{D}$ a (B, δ)-bounded distribution over $\mathbb{Z}^n$, and $\varepsilon \in (0,1/2)$ for which*

$$\beta^2 \geq \alpha^2 + (4/\pi) ln(2n(1 + 1/\varepsilon)) \cdot (aB/q)^2,$$

there exist a transformation reduction from $LWE_{n,m,q,\leq\alpha}(\mathcal{D})$ to $LWE_{n/k,m,q^k,\leq\beta}(G \cdot \mathcal{D})$ that reduces the advantage by $\leq \delta + 14\varepsilon m$, with $G = A_{n/k} \otimes (1, q, q^2, \dots, q^{k-1})^T$. and $max(A_{n/k}) = a$.

Proof. Let $n' = n/k$, $g = (1, q, q^2, \dots, q^{k-1})^T$, we first represent G in matrix form.

$$G = \begin{pmatrix} a_{11}g & \cdots & & \\ \vdots & a_{22}g & & \\ & & \ddots & \\ & & & a_{n'n'}g \end{pmatrix}$$

Let $\Lambda = q^{-k} G \mathbb{Z}^{n'} + \mathbb{Z}^n$, which means Λ's lattice points are those that are linear combinations of the column vectors $q^{-k}G$ plus an arbitrary n-dimensional integer vector.

We now claim that a basis B for Λ has the form:

$$B = A_{n'} \otimes \begin{pmatrix} q^{-1} & q^{-2} & \cdots & q^{-k} \\ & q^{-1} & \cdots & q^{1-k} \\ & & \ddots & \vdots \\ & & & q^{-1} \end{pmatrix} \in \mathbb{R}^{n \times n}$$

This is true because all the column vectors in B belong in Λ and they are linearly independent, which is clear by calculating the determinant of B.

Orthogonalization from left to right, we get:

$$\tilde{B} = A_{n'} \otimes q^{-1} I_{n'}$$

Thus $\left||B'|\right| = max(A_{n'}) \cdot q^{-1}$, since we assumed $a = max(A_{n'}) > 0$, then $max\{q^{-1}, \left||B'|\right|\} = \left||B'|\right| = aq^{-1}$. By Lemma 2.2, we done the proof.

7. Conclusion

This paper first establishes an algorithm to solve general solvable linear Diophantine equations, then proposes an algorithm to solve the linear congruence equation $Ax \equiv b(modm)$ which works for A with a small determinant and either a small dimension or the property of being well-conditioned.

Next the paper generalizes the corollary 3.4 in Brakerski et al.'s 2013 paper by using a general diagonal matrix other than the identity matrix.

This generalization makes more types of matrices useful in the reduction. It is helpful for us to understand the hardness of LWE.

Open questions. As discussed in Section 5, our algorithm for solving linear congruence problems fails short when it comes to matrices with large determinant or with both a large dimension and the property of being ill-conditioned. Thus, we wonder if it is possible to overcome those problems and improve our proposed algorithm so that it accommodates these situations where our original algorithm fails in. Our first instinct is to use some sort of iterative method such as the Jacobi method, but we do notice that the Jacobi method on its own is also not sufficient since it only works for diagonally dominant matrices.

Additionally, as we expand Brakerski et al.'s Corollary 3.4 from only discussing the case for which the matrix is the identity to discussing it for any diagonal matrix, we could not help but wonder if it is possible to extend the argument further and have meaningful results for other choices of matrices. We do not know what results will the expanding of argument bring us regarding our understanding of the hardness of LWE, which is a crucial motivation for any types of reduction upon LWE, but it is certainly satisfactory to extend the proof simply for the purpose of making it more general.

References

[1] John R. Silvester (1980) A Matrix Method for Solving Linear Congruences, Mathematics Magazine, 53:2, 90-92, DOI: 10.1080/0025570X.1980.11976833

[2] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. J. ACM, 56(6):1–40, 2009. Preliminary version in STOC 2005.

[3] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In STOC, pages 333–342. 2009.

[4] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, D. Stehlé, Classical hardness of learning with errors, In Proc. of 45th STOC, pp. 575–584 (ACM, 2013)