On modular inverse matrices a computation approach

DOI: 10.46932/sfjdv3n3-005

Received in: March 22nd, 2022
Accepted in: April 21st, 2022

Fausto Abraham Jacques-García
Doctor of Educational Technology
Institution: Information Science School, Querétaro State University, México
Address: Av. de las Ciencias s/n, col. Juriquilla. Querétaro, Qro, México. C.P. 76230
E-mail: jacques@uaq.edu.mx

Daniel Uribe-Mejía
Matematician
Address: Querétaro, Qro. México, C.P. 76000
E-mail: uribe.d@live.com.mx

Gonzalo Macías-Bobadilla
Doctor of Engineering
Institution: Engineering School, Querétaro State University, México
Address: C.U. Cerro de las Campanas s/n, Col. Centro, Querétaro, Qro, México. C.P. 76010
E-mail: gonzalo.macias@uaq.mx

Ricardo Chaparro-Sánchez
Doctor of Innovation in Educational Technology
Institution: Information Science School, Querétaro State University, México.
Address: Av. de las Ciencias s/n, col. Juriquilla. Querétaro, Qro, México. C.P. 76230
E-mail: rchapa@uaq.mx

ABSTRACT
This paper describes the proposal of a numerical method and its extension, to compute Modular Inverse Matrices and so, Modular Linear Equations Systems (with one, infinite or no-solution set), with no theoretical limit, in \( \mathbb{Z}_n \); considering polynomial and logarithmic time computational complexity. The geometric interpretation of this, implies that elements, such as planes of these vector spaces, interact in the \( n \)-dimensional grid. The interaction and ‘movement’ inside the Grid, can only be possible in a discrete way; from one point to another, like digital states. On the other hand, this work also considers applied mathematics in fields such as cryptography. Based on research, it was observed that this method is an algorithm, because it is precise, defined and finite, so it can be programmed in any computer language. This work constitutes a new approach in numerical analysis for modular inverse matrix computation, plotted in 3-axis linearly. Uses and applications of this proposal are diverse.

Keywords: numerical analysis, modular inverse matrices, gauss-jacques, linear arithmetic spaces, symmetric cryptography, gauss-jacques and montante-jacques algorithms.

1 INTRODUCTION
Modular inverse matrices have one characteristic: all their elements must be integers and positive. By the contrary, the elements in a Natural inverse matrix may be negative and rational. That is the
difference between a Natural inverse matrix and a Modular Inverse Matrix. Not every square matrix is invertible. There is one condition to assure that a n x n square matrix is modular invertible. It is the greater common divisor (GCD) between two values. Those two values are the determinant of the matrix det(K), and the Euclidean modulus m. In other words, if GCD(det(K), m) = 1, then that matrix K, working with modulus mod m is absolutely invertible. In the other hand, if GCD(det(K), n) > 1, then that matrix K, working with mod n, is not invertible. We can have a matrix K that is invertible with mod n, but we can have the same matrix K that is not invertible with modulus m. In the same way, we can have the same mod m that works for a different matrix. There are many uses and applications for modular inverse matrices in Science, Engineering and Education. But, specifically in Computer Science, modular inverse matrices may be applied in Symmetric Cryptography.

This work is based on the Gauss-Jacques method [1]. In symmetric crypto-algorithms, such as Hill Cipher, a square matrix is used as a key (K) to encrypt some data, and the modular inverse of that matrix (K_\text{m}^{-1}((-1))), is used to decrypt the encrypted data. We call a modular invertible K, a candidate key. As we can see in [2], Hill Cipher is a poly-alphabetic and substitution crypto system. We can encrypt data with bigger square candidate keys using the proposed algorithm. Montante-Jacques method is, as well, an approach for achieving Shannon’s perfect secrecy [2].

This proposal is open for feedback and contributions; not only for computer science, but for any science field that requires the use of modular inverse matrices.

2 THEORY

A. Linear Arithmetic Spaces

Linear Arithmetic Spaces (LAS) is the intersection between matrix theory, number theory and superior algebra according to [3]. In the following paragraphs we describe the mathematical background and foundations of the proposed methods related to LAS.

We have modular linear equations. As seen in [1], modular linear equation is a linear equation of the form:

\[(ax) \mod m = c\]  \hspace{1cm} (1)

Where a is the known coefficient of x, x is the incognita we have to find, n is an integer number, and c is a constant number, such as 1; in our context. That is because det(K) and mod m must be co-primes. So, an x must be found, so that multiplying by a, and applying modulus n, result is equal 1. So, we have:
\[ ax = (nq) + 1 \quad (2) \]

Clearing \( x \), then we have:

\[ x = (nq + 1)/a \quad (3) \]

Equation (3) has two unknown variables, which means that \( x \) can have infinite values. To find \( x \), we have to know the value of \( q \), and to find \( q \), we have to find to value of \( x \). This can be done using a brute force method, or, the extended Euclidean algorithm with Diophantine equations.

The integer number \( s \) is divisor of the integer number \( n \), if \( n=st \) for some \( t \in \mathbb{Z} \). The divisibility of \( n \) times \( s \) can be written as \( s|n \). Divisibility is a transitive relation in \( \mathbb{Z} \). If \( m|n \) and \( n|m \), then \( n = \pm m \), and numbers \( m \) and \( n \) can be named associated. According to Euclid’s Theorem, the set \( P = \{2, 3, 5, 7, 11, 13, \ldots\} \) of all prime numbers is infinite [3]. If \( m \) and \( n \) are two integer numbers both not equal zero, then all the integer numbers that divide \( m \) and \( n \), there is a greater divisor, known as the Greater Common Divisor of \( m \) and \( n \), it can be written as \( GCD(m, n) \) [4]. If we have \( a, b \in \mathbb{Z} \), and \( b > 0 \), we can find a \( q, r \in \mathbb{Z} \), so that:

\[ a = bq + r, 0 \leq r \leq b \quad (4) \]

In (4), we have an algorithm to find \( b \) and \( r \) in a finite number steps. Applying this algorithm to \( a, b \) and \( r \), we have that \( b = rq_1 + r_1 \) having \( 0 \leq r_1 > r \). If \( r_1 = 0 \), then, \( r \) is the \( GCD \) \((b, r)\). If \( r_1 \neq 0 \), then \((b, r) = (r, r_1)\), and now the process of finding \( GCD(b, r) \) has been reduced to find the \( GCD(r, r_1) \). To proceed in this way, we get:

\[
\begin{align*}
  a &= bq + r \text{ con } 0 \leq r > b \\
  b &= rq_1 + r_1 \text{ con } 0 \leq r_1 > r \\
  r &= r_1q_2 + r_2 \text{ con } 0 \leq r_2 > r_1 \\
  r_1 &= r_2q_3 + r_3 \text{ con } 0 \leq r_3 > r_2 \\
  &\vdots \\
  r_j &= r_{j+1}q_{j+2} + r_{j+2} \text{ con } 0 \leq r_{j+2} < r_{j+1} \\
\end{align*}
\]

In this way, we will have a \( r_{n+1} = 0 \).

And, we have that \( rn = (a, b) = (b, r) = (r, r_1) = \ldots = (r_{n-1}, r_n) \), so, the \( GCD(a, b) = r_n \) [5].
A Diophantine equation in this context is as follows:

\[ ax + by = c \quad (5) \]

Numbers a, b and c are integers not equal zero, and x and y integers numbers. It is important to note it, because the modular inverse of a square matrix must have all of its elements positive integers, that is why Diophantine equations was used. The equation in (5) has a solution only if d|c, where d = (a, b). If d = (a, b), d|x, and x0, y0, is a particular solution of equation (5), then the rest of solutions (infinite solutions) for x and y is given by equations:

\[ x = x_0 + \frac{b}{d}t \quad (6) \]
\[ y = y_0 - \frac{a}{d}t \quad (7) \]

Where t is also a positive integer number. As I have said before in this paper, when we find a x value and y value, then we can find all the rest of the values, all the infinite solutions with equations (6) and (7). In fact, y is our q in the Euclidean form, so \( q = -y_0 + \frac{a}{d}t \) [6]. There is a change in positive and negative for obvious reasons. If we make a substitution, we have that \( a(x_0 + \frac{b}{d}t) + b(y_0 - \frac{a}{d}t) = ax_0 + \frac{ab}{d}t + by_0 - \frac{ab}{d}t = ax_0 + by_0 \).

Because x0, y0 is a particular solution of \( ax + by = c \), it is concluded that x, y are solutions of (5).

Now, there is a concept used to solve problems in number theory, which depends on the residue properties obtained by dividing two integer positive numbers. Let a, b and m be integer numbers, having m>0, then a is congruent to b modulus m. This can be written as:

\[ a \equiv b \pmod{m} \quad (8) \]

If m|a-b, the number m can be called as the congruence modulus. By the contrary, we can say that a and b are incongruous modulus m. We also have that two integers a and b, have the same residue by being divided into a positive integer m if and only if a \( \equiv b \pmod{m} \) [7].

As we can see, LAS, Euclidean algorithm, Diophantine equations and congruencies, are intimately related each other as seen in [8], and are fundamental elements of the algorithms described below.

3 THE METHODS

As we have mentioned before, not every square matrix has a modular inverse for some modulus m. If the GCD between det(K) and m is equals 1, then, that specific matrix with that specific modulus m,
does have a modular inverse associated. By the contrary, if the result after applying the GCD is greater than 1, then, that specific matrix with that specific modulus m, does not have an associated modular inverse. There is a way to guarantee the modular invertibility. If m is a prime number, then any matrix has its associated modular inverse matrix. It is obvious to know it, because the GCD between those two numbers will always be 1; so det(K) and m are called co-primes. Having this, we may use a statistical-tested random number generator (RNG) to generate de candidate key.

First, the Gauss-Jacques method [1] is based on Gaussian elimination, but plotted in the 3-axis linearly. This method does not use neither determinants nor the adjoint matrix to find the modular inverse of a matrix. This method/algorithm is described in the following steps:

1. Set the size of the key-matrix (n x n).
2. Use RNG to generate the key-matrix (K).
3. Select modular value m as a prime number.
4. Write the identity matrix (I) next to the candidate key-matrix found.
5. Find x, according to (1) and (8), such as, kijx \equiv 1 \pmod{m}.
6. Apply the formula to the entire row, such as, rnx \mod m = new rn. Where rn is n-row.
7. Now, apply Gaussian elimination.
8. Repeat from step 5 to the next pivot.
9. The right-sided matrix is the modular inverse of K.

Applying the algorithm, an inverse matrix is found plotted in 3-axis linearly, this is, in the natural numbers set.

Using the matrix:

\[
\begin{bmatrix}
61 & 31 & 46 & 1 & 0 & 0 \\
98 & 21 & 37 & 0 & 1 & 0 \\
47 & 35 & 45 & 0 & 0 & 1
\end{bmatrix}
\]

as an example, and applying the Gauss-Jacques method modulus 512, K_m^{(-1)}:

\[
\begin{bmatrix}
1 & 0 & 0 & 502 & 445 & 327 \\
0 & 1 & 0 & 187 & 397 & 313 \\
0 & 0 & 1 & 377 & 478 & 257
\end{bmatrix}
\]

In \( Z_n \), so the modular inverse of the matrix is computed as:
Where \( e(k_{ii}, m) \) represents the Euclidean function, which returns the modular inverse \( x \), according to (1) and (5) with \( c = 1 \), both; also with \((a, b)\), having the pivot \((k_{ii})\) as \(a\), and the modular value \((m)\) as \(b\). In this way, these elements constitute the arguments of the Euclidean function. Now, \( k_{ij} \) and \( k_{ji} \), represent elements of matrix \( K \) at \(i\)-row and \(j\)-column.

Its polynomial can be expressed as:

\[
f(n, m_j) = (n^3 + (n^2 - n) \log \phi m_j)
\]

Where \( n \) represents the matrix size, and \( m \) the modular value being used, and \( \phi = (1 + \sqrt{5})/2 \), the logarithmic base. Another way to express it in functional form is \( f(x) = \phi^x \); which derivative \( \frac{\partial}{\partial x} \phi^x = \phi^x \log(\phi) \) indicates the rate of change with respect of time(\( x \)) for each ‘\( x \)’; in a time-complexity algorithmic analysis. And its integral \( \int \phi^x \partial x = \phi^x / \log(\phi) + C \) indicates the area under the curve formed by the two reference points, which is the density or amount of resources [physical and energetic] needed, to execute the algorithm [information].

It’s computational complexity, using big O notation is \( O(n^3 \log m) \).

As we can see in Fig.1, it is a third-degree polynomial; modular inverse matrix computation of this algorithm, is of variable sized with no theoretical limit.

![Fig. 1. The Gauss-Jacques polynomial complexity.](image)

Now, as an extension of the Gauss-Jacques method, we have that the Montante method, also known as Bareiss algorithm, to obtain the inverse of a matrix, can be also a proposal. In this context, we also use matrix product, combinatorial theory, logic propositions, mathematic relations, set theory, linear
equations, divisibility, linear dependency, to mention some, to get the modular inverse of a matrix, but in this case, considering determinants. This method is described in the following steps:

1. Set the size of the key-matrix (n x n).
2. Use RNG to generate the key-matrix (K).
3. Select modular value m as a prime number.
4. Write the identity matrix (I) next to the candidate key-matrix found.
5. Apply the Montante (Bareiss) method to obtain the inverse of that matrix.
6. Instead of dividing the right-side matrix by element a11 (a1i), find its multiplicative inverse x. This can be expressed as a11x (mod m)=1 or a11x ≡ 1 (mod m). Because it is a modular linear equation, there are infinite x’s we could find and use. To do this, we apply the Euclidean extended algorithm as a formal method. We can also find x using a brute force method.
7. Multiply x value found by the augmented resulting matrix.
8. Apply mod m to the augmented resulting matrix.
9. The final right-sided matrix, corresponds to the modular inverse of K (K_m^(-1)) with the selected modulus m.

This methods is called, the Montante-Jacques method.

To verify that K_m^(-1) is the modular inverse of K, we can multiply [(K*K_m^(-1)) mod m] = [(K_m^(-1) *K) mod m] = I.

The following example shows the method step by step: Let’s say we have:

\[
\begin{array}{ccc}
61 & 31 & 46 \\
98 & 21 & 37 \\
47 & 35 & 45 \\
\end{array}
\]

Applying the Montante-Bareiss method, \([ K] m^(-1)\) ::

\[
\begin{array}{ccc|ccc}
8227 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 8227 & 0 & -350 & 215 & 181 \\
0 & 0 & 8227 & -2671 & 583 & 2251 \\
\end{array}
\]

Now, we apply the extended Euclid’s algorithm to find the multiplicative inverse x:

\[
\begin{align*}
8227 &= 512(16) + 35 \\
512 &= 35(14) + 22 \\
35 &= 22(1) + 13 \\
22 &= 13(1) + 9 \\
13 &= 9(1) + 4 \\
9 &= 4(2) + 1 \\
4 &= 1(4) + 0 \\
\end{align*}
\]
Then,
\[
8227 - 512(16) = 35
\]
\[
512 - 35(14) = 22
\]
\[
512 - [8227 - 512(16)](14) = 22
\]
\[
512(1) + [8227(-14) + 512(224)] = 22
\]
\[
512(225) + 8227(-14) = 22
\]

And after,
\[
35 - 22(1) = 13
\]
\[
[8227 - 512(16)] - [512(225) + 8227(-14)] = 13
\]
\[
8227(15) + 512(-241) = 13
\]
\[
22 - 13(1) = 9
\]
\[
[512(225) + 8227(-14)] - [8227(15) + 512(-241)] = 9
\]
\[
512(466) + 8227(-29) = 9
\]

And after then,
\[
13 - 9(1) = 4
\]
\[
[8227(15) + 512(-241)] - [512(466) + 8227(-29)] = 4
\]
\[
8227(44) + 512(-707) = 4
\]
\[
9 - 4(2) = 1
\]
\[
[512(466) + 8227(-29)] - [8227(44) 512(-707)](2) = 1
\]
\[
512(1880) + 8227(-117) = 1
\]
So, \(x_0 = -117\).

We prove it by \([8227(-117)] = 1 \text{ mod } 512\).

The rest of \((x, y)\) solutions can be found by applying (6) and (7). Let’s remember that 
\(d = (a, b)\)
and \(x\) and \(y\) must satisfy \(ax + by = 1\). Where \(t \in \mathbb{Z}\).

So, applying steps 7 and 8, we get the computed \(K_m^{-1}\):

\[
\begin{array}{ccc}
502 & 445 & 327 \\
187 & 397 & 313 \\
377 & 478 & 257 \\
\end{array}
\]

In \(\mathbb{Z}_n\), to verify it, we have that:

\[
\begin{array}{ccccccc}
61 & 31 & 46 & 502 & 445 & 327 & 53761 \\
98 & 21 & 37 & 187 & 397 & 313 & 67072 \\
47 & 35 & 45 & 377 & 478 & 257 & 47104 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
61 & 31 & 46 & 502 & 445 & 327 & 61440 \\
98 & 21 & 37 & 187 & 397 & 313 & 69633 \\
47 & 35 & 45 & 377 & 478 & 257 & 56320 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
61 & 31 & 46 & 502 & 445 & 327 & 41472 \\
98 & 21 & 37 & 187 & 397 & 313 & 48128 \\
47 & 35 & 45 & 377 & 478 & 257 & 37889 \\
\end{array}
\]
Operating each element mod 512 we get the identity matrix:

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

This method has got a logarithmic complexity base \( \phi \). Where \( \phi = (1+\sqrt{5})/2 \). The logarithmic is:

\[
(n\log(n!)/\log(\phi)) \quad (11)
\]

Using big O notation, we have:

\[
O(n \log n!) \quad O(\log m!) \quad (12)
\]

As we can see in Fig. 2, it is a logarithmic-time algorithm; modular inverse matrix computation of this algorithm, is of variable sized with no theoretical limit.

Fig. 2. The Montante-Jacques logarithmic complexity.

4 CONSIDERATIONS

So we have the function \( c = f(n, m) \) for each method. Where \( c \) represents its complexity in computational-time; \( n \) represents the matrix size; and \( m \) represents the modular value. Comparing both algorithms, the plot we have, also using the GeoGebra© software [9], is given below:
As we can see in Fig. 3, the blue functional form corresponds to the Gauss-Jacques algorithm, while the orange one, corresponds to the Montante-Jacques algorithm.

Their behavior is very efficient. The z axis in Fig. 3, represents the amount of operations, as n and m grows. The difference between them is significant; we can observe that the Montante-Jacques is more efficient. However, the limitation we observe for this algorithm is that, its determinant for a big matrix is a very long integer, so, programming techniques must be used to manage it in the computer memory for the later computations. On the other hand, Gauss-Jacques algorithm does not have that issue, but its linear arithmetic computations are more numerous. Each one provides advantages and limitations; it all depends on the technological or scientific purposes to apply.

We are also able to observe its tendency when using bigger matrices. Having known that a common CPU operates millions of arithmetic operations per second, the time-consuming for really big matrices is quite efficient.

Definitely, Computer Sciences need to apply mathematics methods [10] for technology innovation, in this case, based on LAS. According to Grossman [11], matrices represent a topic of interest in many research areas, so research on this field, is more relevant than ever.

5 CONCLUSION

These algorithms represent a good approach in numerical analysis and computational complexity in $\mathbb{Z}_n$. This is because they are either polynomial or logarithmical, while, others are n-degree polynomial, or even factorial. Analyzing the derivative and integral of the algorithm’s behavior, we can observe computational encouraging results.

This makes the proposal as a good option when working with computer science projects, such as symmetric cryptography in digital computers. On the other hand, quantum computers can be used for bigger matrices generation and modular inverse computation. Both methods can definitely be used for Quantum Computing; Quantum Cryptography.
A Key-Matrix ($K$) may have infinite number of modular inverse matrices ($K_m^{-1}$) for infinite modular values ($m$). In the same way, a Key-Matrix ($K$) is related with only one modular inverse matrix ($K_m^{-1}$), with a specific modular value $m$. A Key-Matrix may not have a modular inverse if the modular value used is not a prime.

Applying the Gauss-Jacques method to a particular System of Modular Linear Equations, in dimension ‘n’ ($dim\ n$) the solution set can be found as follows: If a certain system has got more equations than incognita, -an one solution system-, the solution set belongs to the Natural Numeric Set ($N$). If a particular system has got more incognita than equations, -an infinite solution system-, all the solution sets belongs to the Natural Numeric Set ($N$) as well. And, if a particular system has got the same number of equations as incognita, -a square system-, it can be either the above if ‘$m$’ is a prime number, or a no solution system (inconsistent).

Now, Using combinatorial theory, the number of Key-Matrix ($K$) can be generated with 1, 2, 3, or more digits for each element, is $n^2(m^n)$, where $n$ is the size of the square matrix, and $m$ is the amount of digits for each element $k_{ij}$ of $K$. Counting in this way, with transfinite square matrices or Keys -in Cryptography context-, based on Cantor’s transfinite number theory.

For each square-matrix $K$, where $k_{ij} \in N$, and any given $m$ prime modular value, there is a modular inverse matrix ($K_m^{-1}$), ready to be found by the presented method, in $Z_n$.

For each square-matrix $A$, where $a_{ij} \in N$, and any given $m$ prime modular value, there is at least a solution set that satisfies the System of Modular Linear Equations [(Ax $\equiv$ b) mod m] by the presented method, in $Z_n$.

**ACKNOWLEDGMENT**

Thanks to my Heavenly Father in Jesus Christ, who has strengthened me throughout the journey. Thanks to my wife daughter and sons, for their patience and hope. Thanks to my mother, for her continuously support.
REFERENCES


