Proyecciones
Vol. 26, N^o 3, pp. 309-339, December 2007.
Universidad Católica del Norte
Antofagasta - Chile
DOI: 10.4067/S0716-09172007000300007

A TECHNIQUE BASED ON THE EUCLIDEAN ALGORITHM AND ITS APPLICATIONS TO CRYPTOGRAPHY AND NONLINEAR DIOPHANTINE EQUATIONS

LUIS A. CORTÉS VEGA
DANIZA E. ROJAS CASTRO
UNIVERSIDAD DE ANTOFAGASTA, CHILE
and
YOLANDA S. SANTIAGO AYALA
SANTIAGO C. ROJAS ROMERO
UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS, PERÚ
Received : November 2006. Accepted : March 2007

Abstract

The main objective of this work is to build, based on the Euclidean algorithm, a "matrix of algorithms"

$$\Phi_{\mathbf{B}} : \mathbf{N}^*_{m\times n} \to \mathbf{N}^*_{m\times n} \quad , \quad with \quad \Phi_{\mathbf{B}}(\mathbf{A}) = (\Phi_{b_{ij}}(a_{ij})),$$

where $\mathbf{B} = (b_{ij})_{1\leq i\leq m 1\leq j\leq n}$ *is a fixed matrix on* $\mathbf{N}^*_{m\times n}$. *The function* $\Phi_{\mathbf{B}}$ *is called the algorithmic matrix function. Here we show its properties and some applications to Cryptography and nonlinear Diophantine equations.*

The case $n = m = 1$ *has particular interest. On this way we show equivalences between* $\Phi_{\mathbf{B}}$ *and the Carl Friedrich Gauβ's congruence module p.*

Key Words : *Algorithmic matrix function, Euclidean algorithm, non-linear Diophantine equations, message codification and decoding, Gauβ's congruence module p*

1. Introduction and Motivation

A good representative of the concept *algorithm* – which frequently appears in mathematical works and is a fundamental tool– is the so called **Euclidean algorithm** (-330 a -227).

The word *algorithm* has the connotation of an exact method that requires to be followed step by step, in order to solve a problem in a finite number of "iterations".

Nowadays, the understanding of the more abstract concepts of Mathematics can be made easy, in some way, with the support of algorithmic programs.

From a computational viewpoint, an algorithm is a programmed computing mechanism, which has to execute a determined number of "iterations".

An algorithm, not always is represented by a computational program. Its implementation can be made by other types of automata or by the human being. Several, and different algorithms, can made the same task using a distinguishing set of commands, executed in an adequate time.

The concept *algorithm* was formalized in 1936 by *The Turing Machine*, of the mathematician Alan Turing [0]. All These ideas about algorithms became the basis of Scientific Computation, see [0].

The central part of this article consist in building an algorithmic mathematical technique –based on the Euclidean algorithm– which we later apply to Cryptography and problems of Number Theory. Here we show the richness and importance of several mathematical and computational concepts, among them : the concept of computational algorithm, the Euclidean algorithm, the concept of divisibility, the concept of isomorph function, the concept of matrix and its properties, and the concept of congruence module p. In this context, we also put an special emphasis in studying some problems included in the nonlinear Diophantine equations Theory.

The article has four sections and an appendix. The first section is basically devoted to building a rectangular matrix with order $m \times n$, where certain scalar functions $\phi_{b_{ij}}(\cdot)$ – which we call "*algorithmic scalar functions*"– act in every component of it. More exactly, here we build a matrix application

$$\Phi_{\mathbf{B}} : \mathbf{N}^*_{m\times n} \rightarrow \mathbf{N}^*_{m\times n} \quad , \text{ with } \Phi_{\mathbf{B}}(\mathbf{A}) = (\phi_{b_{ij}}(a_{ij}))$$

where $\mathbf{A}$ is a rectangular matrix with order $m \times n$, given by

$$\mathbf{A}=(\mathrm{a}_{ij}) = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n-1} & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn-1} & a_{mn} \end{pmatrix} \in \mathbf{N}^*_{m\times n}$$

and $\Phi_{\mathbf{B}}(\mathbf{A})$ is the algorithmic matrix

$$\Phi_{\mathbf{B}}(\mathbf{A}) \underset{=}{\text{def}} (\phi_{b_{ij}}(a_{ij})) = \begin{pmatrix} \phi_{b_{11}}(a_{11}) & \phi_{b_{12}}(a_{12}) & \dots & \phi_{b_{1n}}(a_{1n}) \\ \phi_{b_{21}}(a_{21}) & \phi_{b_{22}}(a_{22}) & \dots & \phi_{b_{2n}}(a_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{b_{m1}}(a_{m1}) & \phi_{b_{m2}}(a_{m2}) & \dots & \phi_{b_{mn}}(a_{mn}) \end{pmatrix} \in$$

$\mathbf{N}^*_{m\times n}$

where $\phi_{b_{ij}} : \mathbf{N}^* \to \{0, 1, 2, \dots, b_{ij} - 1\}$ and $\mathbf{B} = (b_{ij})$ is a fixed matrix in $\mathbf{N}_{m\times n}$ with all their components $b_{ij} \geq 2$, $1 \leq i \leq m$, $1 \leq j \leq n$. The function $\Phi_{\mathbf{B}}$ built on this way, is called "*algorithmic matrix function*". In the second section we show properties and theorems of this function, as well as its applications to message codification and decoding. In the third section we have some applications to the nonlinear Diophantine equations. Then, in the fourth section, we face the particular case $n = m = 1$ and show some equivalences between the concepts of algorithmic scalar function and *Carl Friedrich Gauß*'s congruence module p. We Finalize the work by attaching an appendix in which a program that computes the residue of division of two positive entire numbers is given.

2. Building the algorithmic matrix function

We start the work by reviewing some concepts and results which will be used throughout it.

The following Theorem is one of the more important results in Arithmetic, [0].

Theorem 2.1. (*Euclidean Algorithm*)

Let a and b be entire numbers, $b > 0$. Then there exist, and be unique, entire numbers q and r such that a=bq+r, with $0 \leq r < b$.

This numbers q and r are called the quotient and residue of the entire division of a by b.

The proof of this theorem is a classic one, but we give it with the purpose of benefit of the article's reading.

Proof.

If b divides a, the result is valid with $r = 0$. So, we only consider the case when b does not divide a.

Let $S = \{a - tb/t \in \mathbf{Z}, a - tb > 0\}$. If $a > 0$ and $t = 0$, then $a \in S$ and S is not empty. If $a \leq 0$, take $t = a-1$, then $a-tb = a-(a-1)b = a(1-b)+b$, with $(1-b) \leq 0$, since $b \geq 1$. Thus, $a-tb > 0$ and S is not empty. Therefore, for all $a \in \mathbf{Z}$, S is a non-empty set in $\mathbf{Z}$. For the well-ordering Principle, S contains a least element r, such that $0 < r = a - qb$, for some $q \in \mathbf{Z}$. If $r = b$, then $a = (q + 1)b$ and b divides a, which contradicts the fact that b does not divide a. If $r > b$, then $r = b + c$ for some $c \in \mathbf{Z}$ and $a - qb = r = b + c$, and this implies that $c = a - (q + 1)b \in S$, which contradicts the fact that r is the least element of S. Therefore, $r < b$. □

A consequence of this Theorem is Corollary 2.2 given below. Before stating, let us denote $\mathbf{N}$, the set of all natural numbers and $\mathbf{N}^* = \mathbf{N} \cup \{0\}$; Also, the symbol $\times$ will denote the usual product in $\mathbf{N}$.

We have to mention that even though Theorem 2.1 guarantees the existence of the residue r when an entire a (dividend) is divided by b (divisor), it does not indicate how to calculate this residue. Just with the purpose of helping readers, in the appendix we put a program that allow us to calculate the residue of a division; this program is an adaptation from an algorithm given in [0], pp. 217-218.

Corollary 2.2. Let $a, b \in \mathbf{N}^*$, $b \geq 2$. There exist, and be unique, the values r_i, $0 \leq i \leq n$, $0 \leq r_i \leq b - 1$, such that

$\mathrm{a}=\mathrm{r}_0 + r_1 b + r_2 b^2 + \ldots + r_{n-1} b^{n-1} + r_n b^n$

where n satisfies $a < b^{n+1}$.

Corollary 2.2 and Theorem 2.1 allow us define a mapping on which is based all this article.

Definition 2.3. Given $a, b \in \mathbf{N}^*$ with $b \geq 2$, such that

$\mathrm{a}=\mathrm{r}_0 + r_1 b + r_2 b^2 + \ldots + r_{n-1} b^{n-1} + r_n b^n$,

with $0 \leq r_i \leq b - 1$, for all $0 \leq i \leq n$, we define the following mapping on $\mathbf{N}^*$

$$\phi_b : \mathbf{N}^* \rightarrow \mathbf{N}^*, \text{ tal que } \phi_b(a) = \begin{cases} a, & \text{if } 0 \leq a \leq b-1, \\ r_0, & \text{if } a \geq b. \end{cases}$$

Just to fix ideas, we give some examples of how ϕ_b operates.

Example 2.4. From definition of ϕ_b , we have :
(a) $\phi_3(9) = 0$, (b) $\phi_{17}(23) = 6$, (c) $\phi_8(25) = 1$,
(d) $\phi_{13}(5) = 5$, (e) $\phi_8(9) = 1$, (f) $\phi_5(\phi_8(21)) = 0$,
(g) $\phi_7(\phi_{15}(\phi_6(21))) = 3$.

Remark 2.5.

(i) In Definition 2.3, if $a \geq b$, then r_0 represents the residue of the division of a by b, and so, $0 \leq r_0 \leq b - 1$.

(ii) From Definition 2.3, we can get the following algorithm :

Start denoting $\eta_0 = \phi_{b_0}(a)$, where $b_0 \geq 2$ and $a \geq b_0$. Then, for $b_1 \geq 2$ we have

$$\phi_{b_1}(\eta_0) = \begin{cases} \eta_0, & \text{if } 0 \leq \eta_0 \leq b_1 - 1, \\ \\ r_1, & \text{if } \eta_0 \geq b_1, \end{cases}$$

thus,

$$\phi_{b_1}(\phi_{b_0}(a)) = \begin{cases} \eta_0, & \text{if } 0 \leq \eta_0 \leq b_1 - 1, \\ \\ r_1, & \text{if } \eta_0 \geq b_1. \end{cases}$$

Now, take $\eta_1 = \phi_{b_1}(\phi_{b_0}(a))$ and $\eta_0 \geq b_1$, then for $b_2 \geq 2$, we have

$$\phi_{b_2}(\eta_1) = \begin{cases} \eta_1, & \text{if } 0 \leq \eta_1 \leq b_2 - 1, \\ \\ r_2, & \text{if } \eta_1 \geq b_2, \end{cases}$$

thus,

$$\phi_{b_2}(\phi_{b_1}(\phi_{b_0}(a))) = \begin{cases} \eta_1, & \text{if } 0 \leq \eta_1 \leq b_2 - 1, \\ \\ r_2, & \text{if } \eta_1 \geq b_2. \end{cases}$$

Continue on this way, until the $(k-1)$-th step, with $\phi_{b_{k-1}}(\phi_{b_{k-2}}(\ldots \phi_{b_0}(a))) = \eta_{k-1}$, then for $b_k \geq 2$ and $\eta_{k-2} \geq b_{k-1}$, we have

$$\phi_{b_k}(\eta_{k-1}) = \begin{cases} \eta_{k-1}, & \text{if } 0 \leq \eta_{k-1} \leq b_k - 1, \\ \\ r_k, & \text{if } \eta_{k-1} \geq b_k, \end{cases}$$

thus,

$$\eta_k = \phi_{b_k}(\phi_{b_k-1}(\phi_{b_{k-2}}(\dots\phi_{b_0}(a)))) = \begin{cases} \eta_{k-1}, & \text{if } 0 \le \eta_{k-1} \le b_k - 1, \\ r_k, & \text{if } \eta_{k-1} \ge b_k. \end{cases}$$

Henceforth, we will denote it by

$$\eta_k = \phi_{b_k} \circ \phi_{b_k-1} \circ \phi_{b_{k-2}} \circ \dots \circ \phi_{b_0}(a) = \begin{cases} \eta_{k-1}, & \text{if } 0 \le \eta_{k-1} \le b_k - 1, \\ r_k, & \text{if } \eta_{k-1} \ge b_k. \end{cases}$$

Note, this algorithm stops on the k-th iteration when $0 \le \eta_{k-1} < b_k$, and can be programmed similarly to the algorithm given in the appendix.

The following example illustrates the former algorithm.

Example 2.6.
It is evident $\phi_9(\phi_7(\phi_8(\phi_{24}(\phi_{62}(100))))) = 6$. In fact, we have

$$\begin{aligned} \phi_9(\phi_7(\phi_8(\phi_{24}(\phi_{62}(100))))) &= \phi_9(\phi_7(\phi_8(\phi_{24}(38)))) \\ &= \phi_9(\phi_7(\phi_8(14))) = \phi_9(\phi_7(6)) = \phi_9(6) = 6. \end{aligned}$$

In conclusion:

$$\phi_9 \circ \phi_7 \circ \phi_8 \circ \phi_{24} \circ \phi_{62}(100) = 6.$$

Clearly, in the example we have: $b_0 = 62$, $b_1 = 24$, $b_2 = 8$, $b_3 = 7$ and $b_4 = 9$. Also note, if $b_0 = b_1 = \dots = b_k = b$, the algorithm stops on the first iteration, since by definition of ϕ_b, the formula $\phi_b(a) = \phi_b(\phi_b(a))$ is always satisfied.

Remark 2.7.

(i) Given $x, y \in \mathbf{N}^*$, $x \ge b, y \ge b$ such that $x = y$, then $\phi_b(x) = \phi_b(y)$. It makes ϕ_b a function for each $b \in \mathbf{N}^*$, with $b \ge 2$.

(ii) In this context, the domain of ϕ_b for each $b \in \mathbf{N}^*$, with $b \ge 2$, is the set $\mathbf{N}^*$, and its range is the set $\phi_b(\mathbf{N}^*) = \{0, 1, 2, \dots, b-1\} \subset \mathbf{N}^*$.

(iii) The function ϕ_b acting on $a \in \mathbf{N}^*$, with $a \ge b$, gives the residue r of dividing a by b.

(iv) The function ϕ_b , for each $b \in \mathbf{N}^*$, with $a \geq b$, $b \geq 2$, **it is not injective** (see example 2.4, parts (c) and (e), above).

Theorem 2.8.

Let a, b and $c \in \mathbf{N}^*$, where $b \geq 2$, $a \geq b$ and $c \geq b$. Let also r_{0a} and r_{0c} be the first 0-th coefficients appearing in the decomposition of a and c on base b, respectively. Then ϕ_b verifies the following properties :

(a) $\phi_b(0) = 0$,

(b) $\phi_b(db) = 0$, para todo $d \in \mathbf{N}^*$,

(c) $\phi_b(a) = \phi_b(\phi_b(a))$,

(d) $\phi_b(a + c) = \phi_b(\phi_b(a) + \phi_b(c))$,

(e) $\phi_b(a \times c) = \phi_b(\phi_b(a) \times \phi_b(c))$

(f) $\phi_b(a \times c) = \phi_b(a \times \phi_b(c))$

(g) $\phi_b(a \times c) = \phi_b(\phi_b(a) \times c)$

(h) $\phi_b(a + b) = \phi_b(a)$ ("periodicity" of ϕ_b)

Remark 2.9.

It is east to see that – from the Euclidean algorithm – for every $a \in \mathbf{N}^*$ such that $a \geq b$, we can characterize the function ϕ_b on the next way :

$\phi_b(a) = r$, if and only if, there exists $q \in \mathbf{N}^*$, such that $a = r + bq$, with

$0 \leq r \leq b - 1$.

For that and Remark 2.5, we name the mapping ϕ_b as "*algorithmic scalar function*". This function can be seen like an iterative process, from which we get, after k iterations, a natural number η_k, such that $0 \leq \eta_k < b_k$.

Now, we use the characterization of ϕ_b given in Definition 2.3 to prove Theorem 2.8.

Proof of Theorem 2.8

(a) There exists $q = 0 \in \mathbf{N}^*$, such that $0 = 0 + 0b$, that is, $\phi_b(0) = 0$. This part of Theorem 2.8 can also be proved using Definition 2.3.

(b) There exists $q = d \in \mathbf{N}^*$, such that $db = 0 + db$, that is, $\phi_b(db) = 0$.

(c) Let $\phi_b(a) = r_{0a}$. Then, $r_{0a} \leq b - 1$. This and Remark 2.5 imply that $\phi_b(r_{0a}) = r_{0a}$, it means, $\phi_b(\phi_b(a)) = r_{0a} = \phi_b(a)$.

(d) Let $r_{0a} = \phi_b(a)$ and $r_{0c} = \phi_b(c)$, then there exist q_1 and $q_2 \in \mathbf{N}^*$, such that $a = r_{0a} + q_1 b$ and $c = r_{0c} + q_2 b$, with $0 \leq r_{0a} \leq b-1$ y $0 \leq r_{0c} \leq b-1$. It means, there exists $q = q_1 + q_2 \in \mathbf{N}^*$, such that

$$(a+c) = (r_{0a} + r_{0c}) + qb. \tag{2.1}$$

By other hand, let $\phi_b(a+c) = r_{0(a+c)}$, then there exists q_3, such that

$$(a+c) = r_{0(a+c)} + q_3 b, \tag{2.2}$$

with $0 \leq r_{0(a+c)} \leq b-1$. Now, from (1) and (2) we get a $q_4 = q - q_3 \in \mathbf{N}^*$, (ó $q_4 = q_3 - q \in \mathbf{N}^*$) such that $(r_{0a} + r_{0b}) = r_{0(a+c)} + q_4 b$, that is $\phi_b(r_{0a} + r_{0c}) = r_{0(a+c)}$. Hence

$$\phi_b(\phi_b(a) + \phi_b(c)) = \phi_b(a+c).$$

This proves item (d).

(f)–(g) Let $\phi_b(a) = r_{0a}$, then there exists $q_1 \in \mathbf{N}^*$, such that $a = r_{0a} + q_1 b$, with $0 \leq r_{0a} \leq b-1$. Thus, for each $z \in \mathbf{N}^*$ we have

$$\begin{aligned}
\phi_b(z \times c) &= \phi_b \underbrace{(c + c + c + ... + c)}_{\mathsf{z-times}} \overset{(d)}{=} \phi_b(c + (c + c + ...c)) \\
&\overset{(d)}{=} \phi_b\left(\phi_b(c) + \phi_b \underbrace{(c + c + ... + c)}_{\mathsf{z-1\ times}}\right) \overset{(d)}{=} \\
&\phi_b\left(\phi_b(c) + \phi_b(c) + \phi_b \underbrace{(c + c + ... + c)}_{\mathsf{z-2\ times}}\right) \overset{(d)}{=} \\
&\overset{(d)}{=} ... \overset{(d)}{=} \phi_b \underbrace{(\phi_b(c) + \phi_b(c) + ... + \phi_b(c))}_{\mathsf{z\ times}} = \phi_b(z\phi_b(c))
\end{aligned} \tag{2.3}$$

Now, making the change of variable $z = a$ in equation (3), it follows that $\phi_b(a \times c) = \phi_b(a\phi_b(c))$. The proof of item (g) is analogous.

(e) Let $\phi_b(a) = r_{0a}$, then there exists $q_1 \in \mathbf{N}^*$, such that $a = r_{0a} + q_1 b$, with $0 \leq r_{0a} \leq b-1$. Using item (f), already proved, we have

$$\begin{aligned}
\phi_b(a \times c) &= \phi_b(a\phi_b(c)) = \phi_b((r_{0a} + q_1 b)\phi_b(c)) = \\
&= \phi_b(r_{0a}\phi_b(c) + q_1 b\phi_b(c))(d) - (b) = \phi_b(r_{0a}\phi_b(c))(c) = \phi_b(\phi_b(a)\phi_b(c)).
\end{aligned}$$

This proves item (e).

To prove item **(h)**, is enough to use items (d), (b) and (c). Therefore, Theorem 2.8 was entirely proved. □

Corollary 2.10.
Let $a_1, a_2, \ldots, a_n \in \mathbf{N}^*$, and p, $b \in \mathbf{N}^*$, with $b \geq 2$ and $p \geq 1$. Then:

(i) $\phi_b(\sum_{k=0}^{p} a_k) = \phi_b(\sum_{k=0}^{p} \phi_b(a_k))$,

(ii) $\phi_b(\prod_{k=o}^{p} a_k) = \phi_b(\prod_{k=o}^{p} \phi_b(a_k))$.

Proof: It is straightforward from Theorem 2.8, items (c) y (d).

By other hand, the next corollary comes evident

Corollary 2.11.
Let a, p, $b \in \mathbf{N}^*$, such that $b \geq 2$ and $p \geq 1$. Then:
$\phi_b(a^p) = \phi_b([\phi_b(a)]^p)$.

Proof: It is immediate from Corollary 2.10, item (ii).

Definition 2.12 (Algorithmic matrix Function)
Given $a_{ij}, b_{ij} \in \mathbf{N}^*$ with $b_{ij} \geq 2$, for all $1 \leq i \leq m, 1 \leq j \leq n$. Let ϕ_{bij}, $1 \leq i \leq m, 1 \leq j \leq n$, be algorithmic scalar functions of the form

$$\phi_{bij} : \mathbf{N}^* \to \mathbf{N}^*, \text{ such that } \phi_{b_{ij}}(a_{ij}) = \begin{cases} a_{ij}, & \text{if } 0 \leq a_{ij} \leq b_{ij} - 1, \\ r_{0_{ij}}, & \text{if } a_{ij} \geq b_{ij}. \end{cases}$$

From them, we build a matrix mapping $\Phi_{\mathbf{B}}$:

$$\Phi_{\mathbf{B}} : \mathbf{N}^*_{m\times n} \to \mathbf{N}^*_{m\times n}, \text{ with } \Phi_{\mathbf{B}}(\mathbf{A}) = (\phi_{b_{ij}}(a_{ij})),$$

where $\mathbf{A}$ is a matrix with order $m \times n$ built from the a_{ij}:

$$\mathbf{A} = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n-1} & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \ldots & a_{mn-1} & a_{mn} \end{pmatrix} \in \mathbf{N}^*_{m\times n},$$

with

$$\Phi_{\mathbf{B}}(\mathbf{A}) \underset{=}{\text{def}} (\phi_{b_{ij}}(a_{ij})) = \begin{pmatrix} \phi_{b_{11}}(a_{11}) & \phi_{b_{12}}(a_{12}) & \dots & \phi_{b_{1n}}(a_{1n}) \\ \phi_{b_{21}}(a_{21}) & \phi_{b_{22}}(a_{22}) & \dots & \phi_{b_{2n}}(a_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{b_{m1}}(a_{m1}) & \phi_{b_{m2}}(a_{m2}) & \dots & \phi_{b_{mn}}(a_{mn}) \end{pmatrix} \in \mathbf{N}^*_{m\times n}$$

Here, $\mathbf{B} = (b_{ij})$ is a "base matrix" fixed in $\mathbf{N}^*_{m\times n}$, with all their components $b_{ij} \geq 2$. The mapping $\Phi_{\mathbf{B}}$, built on this way, is called *algorithmic matrix function*. This matrix function has interesting properties.

Theorem 2.13.

Let $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ be matrices in $\mathbf{N}^*_{m\times n}$, with $\mathbf{A} = (a_{ij})$, $\mathbf{B} = (b_{ij})$ and $\mathbf{C} = (c_{ij})$, such that $b_{ij} \geq 2$, with $a_{ij} \geq b_{ij}$ and $c_{ij} \geq b_{ij}$. Then, the algorithmic matrix function $\Phi_{\mathbf{B}}$ satisfies the following properties:

(a) $\Phi_{\mathbf{B}}(\mathbf{O}) = \mathbf{O}$,

(b) $\Phi_{\mathbf{B}}(d\mathbf{B}) = \mathbf{O}$, for all $d \in \mathbf{N}^*$,

(c) $\Phi_{\mathbf{B}}(\mathbf{A}) = \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{A}))$,

(d) $\Phi_{\mathbf{B}}(\mathbf{A} + \mathbf{C}) = \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{A}) + \Phi_{\mathbf{B}}(\mathbf{C}))$,

(e) $\Phi_{\mathbf{B}}(d\mathbf{A}) = \Phi_{\mathbf{B}}(d\Phi_{\mathbf{B}}(\mathbf{A}))$,

(f) Let $\mathbf{B} = (b_{ij})$, $\mathbf{E} = (e_{ij})$, $\mathbf{F} = (f_{ij}) \in \mathbf{N}^*_{p\times p}$, with $b_{ij} \geq 2$. Let $\mathbf{E}\cdot\mathbf{F} = \mathbf{D} = (d_{ij})$, thus $d_{ij} = \sum_{k=1}^{p} e_{ik}f_{kj}, 1 \leq i, j \leq p$, then

$$\Phi_{\mathbf{B}}(\mathbf{E}\cdot\mathbf{F}) = (\phi_{b_{ij}}(d_{ij})) = (\phi_{b_{ij}}(\sum_{k=1}^{p} \phi_{b_{ij}}(e_{ik}) \times \phi_{b_{ij}}(f_{kj}))).$$

(g) Let $\mathbf{G} = (g_{ij}) \in \mathbf{N}^*_{p\times p}$, a matrix whose components g_{ij} satisfy $g_{ij} < b_{ij}$, with $b_{ij} \geq 2$, for all $1 \leq i \leq m$, $1 \leq j \leq n$. Then,

$$\Phi_{\mathbf{B}}(\mathbf{G}) = \mathbf{G}.$$

In particular, we have

(h) $\Phi_{\mathbf{B}}(\mathbf{I}) = \mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix in $\mathbf{N}^*_{m\times n}$,

(i) $\Phi_{\mathbf{B}}(\mathbf{A} + \mathbf{B}) = \Phi_{\mathbf{B}}(\mathbf{A})$ ("periodicity" of $\Phi_{\mathbf{B}}$).

Proof:

The proofs of **(a)**-**(g)** are straightforward from Theorem 2.8 and definition of $\Phi_{\mathbf{B}}$. The proof of item **(h)** is straightforward from definition of $\Phi_{\mathbf{B}}$ and the fact that $b_{ij} \geq 2 > 0$, $b_{ij} \geq 2 > 1$, for all $1 \leq i \leq m, 1 \leq j \leq n$. Item **(i)** is straightforward from item **(h)** of Theorem 2.8. □

Remark 2.14.

It is interesting ro remark that when $\mathbf{A} = (a_{ij})$, $\mathbf{B} = (b_{ij})$ and $\mathbf{C} = (c_{ij}) \in \mathbf{N}^*_{m\times n}$ with $b_{ij} \geq 2$, $c_{ij} \geq 2$, for all $1 \leq i \leq m$, $1 \leq j \leq n$, then the following matrix operation can be used :

$$\Phi_{\mathbf{C}}(\Phi_{\mathbf{B}}(\mathbf{A})) \underset{=}{(d)}$$

$$\begin{pmatrix} \phi_{c_{11}}(\phi_{b_{11}}(a_{11})) & \phi_{c_{12}}(\phi_{b_{12}}(a_{12})) & \dots & \phi_{c_{1n}}(\phi_{b_{1n}}(a_{1n}) \\ \phi_{c_{21}}(\phi_{b_{21}}(a_{21})) & \phi_{c_{22}}(\phi_{b_{22}}(a_{22}) & \dots & \phi_{c_{2n}}(\phi_{b_{2n}}(a_{2n})) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{c_{m1}}(\phi_{b_{m1}}(a_{m1})) & \phi_{c_{m2}}(\phi_{b_{m2}}(a_{m2})) & \dots & \phi_{c_{mn}}(\phi_{b_{mn}}(a_{mn})) \end{pmatrix}$$

Now, using this identity, we can generalize the whole algorithm presented in remark 2.5 to algorithmic matrix functions.

We illustrate this idea with the following example.

Example 2.15.

Let $\mathbf{A} = (a_{ij}) \in \mathbf{N}^*_{2\times 3}$, the matrix

$$\mathbf{A} = \begin{pmatrix} 21 & 79 & 45 \\ 49 & 5 & 39 \end{pmatrix} \in \mathbf{N}^*_{2\times 3}.$$

Let $\mathbf{B} = (b_{ij})$ and $\mathbf{C} = (c_{ij}) \in \mathbf{N}^*_{2\times 3}$ the following base matrices

$$\mathbf{B} = \begin{pmatrix} 5 & 12 & 23 \\ 12 & 4 & 35 \end{pmatrix}, \ \mathbf{C} = \begin{pmatrix} 2 & 7 & 9 \\ 34 & 8 & 15 \end{pmatrix} \in \mathbf{N}^*_{2\times 3}.$$

Then

$$\begin{aligned}\Phi_{\mathbf{C}}(\Phi_{\mathbf{B}}(\mathbf{A})) &= \begin{pmatrix} \phi_2(\phi_5(21)) & \phi_7(\phi_{12}(79)) & \phi_9(\phi_{23}(45)) \\ \phi_{34}(\phi_{12}(49)) & \phi_8(\phi_4(5)) & \phi_{15}(\phi_{35}(39)) \end{pmatrix} \\ &= \begin{pmatrix} \phi_2(1) & \phi_7(7) & \phi_9(22) \\ \phi_{34}(1) & \phi_8(1) & \phi_{15}(4) \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 4 \\ 1 & 1 & 4 \end{pmatrix}.\end{aligned}$$

3. An algorithm to codify and decode messages: An approach.

In Cryptography –from Greek kryptos (hide) and grafos (write), literally "hidden writing" – there are a great variety of interesting problems related to codification and decoding of information. Essentially, Cryptography is the art or science of codifying and decoding information **using mathematical techniques** which make possible the interchanging of messages so that **they only can be read by the people to whom they are sent**, see [2] and its references.

In this section we will use the concepts and results of the former section. Here, we face some problems concerned to Cryptography, specifically consider codification and decoding of Spanish language (this should not restrict the analysis, for any reason). Thus the main objective of this section is to show that, by the algorithmic matrix function, we can codify and decode messages.

The authors of this article think that techniques and methods here developed have great part of originality, and can be very useful when they are applied to other fields of knowledge, like Genetics, specifically in reading and interpretation of human DNA sequences.

An important point to be noted in this section – which we will soon describe – is the concept of "*Levels of Codification and Decoding*". Also, from a didactic viewpoint, this section was conceived self-sufficient. All the group of work has the sensation that the exposed ideas can germinate and be taken to other areas such as Didactic of Mathematics, Mathematics and Finances, Discrete Mathematics, Games Theory, Computation and Informatics, Artificial Intelligence, Arithmetic, Engineering, and so on.

Definition 3.1. (Language Matrix)

Denote by $S_{m\times n}$ the space of all matrices whose components only contain symbols coming from the Spanish alphabet and/or an insignificant

symbol, which will be denoted by &. The symbols in the components of the matrix can have or not a pre-established order.

We will call the matrices in $S_{m\times n}$ "Language matrices". If $\mathbf{L} \in S_{m\times n}$, then :

$$\mathbf{L} = \begin{pmatrix} l_{11} & l_{12} & l_{13} & \dots & l_{1n-1} & l_{1n} \\ l_{21} & l_{22} & l_{23} & \dots & l_{2n-1} & l_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ l_{m1} & l_{m2} & l_{m3} & \dots & l_{mn-1} & l_{mn} \end{pmatrix}_{m\times n}.$$

Below, we illustrate Definition 3.1 with some examples.

Example 3.2.

$$\mathbf{L}=\begin{pmatrix} c & w & w & s \\ \& & q & p & m \\ t & y & u & z \end{pmatrix} \in S_{3\times 4}\,, \quad , \mathbf{G} = \begin{pmatrix} d & e & j & a & m & e \\ q & u & e & \& & t & e \\ c & u & e & n & t & e \end{pmatrix} \in S_{3\times 6}.$$

Definition 3.3. (Isomorphism)

Let S be the set of all the Spanish alphabet letters, more an insignificant sign, ordered in the following way:

$$S = \{\&, a, b, c, \dots, x, y, z\} = \{s_1, s_2, s_3, \dots, s_{26}, s_{27}, s_{28}\}.$$

Let C be the set of the first twenty-seven natural numbers, and the zero, that is:

$$C = \{0, 1, 2, \dots, 25, 26, 27\} = \{c_1, c_2, c_3, \dots, c_{26}, c_{27}, c_{28}\}.$$

With this sets, we define the following isomorphism:

$$f : S \to C, \text{ where } f(s_k) = c_k,\ 1 \le k \le 28.$$

Note, for example: $f(s_5) = f(d) = 4$, $f(s_{21}) = f(s) = 20$, $f(s_1) = f(\&) = 0$. Also note that the inverse function of f, $g = f^{-1}$ works like this

$$g : C \to S, \text{ where } s_k = g(c_k),\ 1 \le k \le 28.$$

Which let us conclude that $d = g(4)$, $s = g(20)$ and $\& = g(0)$.

Remark 3.4. :

(i) Definition 3.3 is not restrictive, to the effect of that always is possible to use other symbols.

(ii) To the known Spanish alphabet letters, we have added –with the idea of simplifying the reading of the message– a new symbol "&", which will denote a blank space between the words or letters, and form part of message codification and/or decoding.

(iii) We have to mention that message codification and decoding, are frequently used to building codes of public and private domains.

Definition 3.5. (Transformed Matrix)

Denote by $\mathbf{T} = (t_{ij})$ to the "transformed matrix" of a language matrix $\mathbf{L}$. This matrix will be composed of $m \times n$ numerical characters t_{ij}, such that $t_{ij} < 28$, for all $1 \leq i \leq m$, $1 \leq j \leq n$.

This characters will be attached to the components of matrix $\mathbf{T}$, by the following mapping:

$$F : S_{m\times n} \to \mathbf{N}^*_{m\times n}, \text{ such that } F(\mathbf{L}) = \mathbf{T}, \text{ with } f(l_{i,j}) = t_{i,j}, \ 1 \leq i \leq m,$$
$$1 \leq j \leq n.$$

Here, f is the isomorphism given by Definition 3.3.

In order to illustrate this, we will use the same matrix of example 3.2. In fact, note that the language matrix

$$\mathbf{L} = \begin{pmatrix} c & w & w & s \\ \& & q & p & m \\ t & y & u & z \end{pmatrix}$$

has its transformed matrix

$$\mathbf{T} = F(\mathbf{L}) = \begin{pmatrix} 3 & 24 & 24 & 20 \\ 0 & 18 & 17 & 13 \\ 21 & 26 & 22 & 27 \end{pmatrix}.$$

Definition 3.6. We will call "return matrix" to the matrix obtained from the mapping

$$R : N^*_{m\times n} \to \mathbf{S}_{m\times n}, \text{ with } R(\mathbf{T}) = (g(t_{i,j})), \ 1 \leq i \leq m, 1 \leq j \leq n,$$

where g is the inverse function of the isomorphism f given in Definition 3.3. For illustration, we use the former example. Thus, we have:

$$R(\mathbf{T}) = R\left(\begin{pmatrix} 3 & 24 & 24 & 20 \\ 0 & 18 & 17 & 13 \\ 21 & 26 & 22 & 27 \end{pmatrix}\right) = \begin{pmatrix} c & w & w & s \\ \& & q & p & m \\ t & y & u & z \end{pmatrix}.$$

Now, let $\mathbf{L}_0 \in \mathbf{S}_{m\times n}$ a language matrix. We will suppose this matrix contains the message we want to codify or decode. Also, let $\mathbf{C}_0 = (q_{ij}) = F(\mathbf{L}) \in \mathbf{N}^*_{m\times n}$, a matrix, which we call *"fount matrix"*. This matrix has the particularity of being the transformed of a language matrix $\mathbf{L}_0 \in \mathbf{S}_{m\times n}$, for that, its components are numerical characters $q_{ij} < 28$, $1 \leq i \leq m$, $1 \leq j \leq n$. Also, denote by $\mathbf{C}_p = (p_{ij}) \in \mathbf{N}^*_{m\times n}$, the matrix which we will call "*personal code matrix*"; This matrix has the particularity of being given to the person who will codify the hidden message in $\mathbf{L}_0$. Now, we are ready to give the next definition.

Definition 3.7. Let $\mathbf{L}_0 \in \mathbf{S}_{m\times n}$, $\mathbf{C}_0 \in \mathbf{N}^*_{m\times n}$ the matrices already characterized. Let $\mathbf{B} \in \mathbf{N}^*_{m\times n}$ a base matrix of the form

$$\mathbf{B} = \begin{pmatrix} 28 & 28 & 28 & \dots & 28 & 28 \\ 28 & 28 & 28 & \dots & 28 & 28 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 28 & 28 & 28 & \dots & 28 & 28 \end{pmatrix}_{m\times n}$$

and $\mathbf{C}_p^{(1)}$, $\mathbf{C}_p^{(2)}$, $\mathbf{C}_p^{(3)}, \dots, \mathbf{C}_p^{(k)}$ a sequence of k personal code matrices, with $\mathbf{C}_p^{(k)} = (c_{ij}^{(k)}) \in \mathbf{N}^*_{m\times n}$. The level in where is computed the matrix $\mathbf{C}_k$, resulting of the operation

$$\Phi_{\mathbf{B}}(\sum_{m=1}^{k} \mathbf{C}_p^{(m)} + \mathbf{C}_0) = \mathbf{C}_k,$$

is called the $k-th$ **level of codification of matrix $\mathbf{C}_0$ using the personal code $\mathbf{C}_p^{(k)}$.**

This definition surges from the analysis of the following scheme of codification of matrix $\mathbf{C}_0$. For that we use the personal code matrices $\mathbf{C}_p^{(m)}$, $m = 1, 2, \dots, k$ and the algorithmic matrix function $\Phi_{\mathbf{B}}$. For example, If we are at the kth -level of codification of matrix $\mathbf{C}_0$, the scheme takes the form:

$$\boxed{\begin{array}{c}\mathbf{L}o, \mathbf{C}_0, \left\{\mathbf{C}_p^{(m)}\right\}_{m=1}^{k}, \mathbf{B} \\ \text{Data for the k-th level}\end{array}} \overset{\rightarrow}{\rightarrow} \boxed{\begin{array}{c}\phi_{\mathbf{B}}\left(\sum_{m=1}^{k}\mathbf{C}_p^{(m)} + \mathbf{C}_0\right) \\ \text{Algorith at the k-th level}\end{array}} \overset{\rightarrow}{\rightarrow} \boxed{\begin{array}{c}\mathbf{C}_k, R(\mathbf{C}_k) = \mathbf{L}_k \\ \text{output of the k-th level}\end{array}}$$

Repeat the process if we want to codify $\mathbf{C}_0$ again

In other words,

Initial data: $\mathbf{L}_0 \in \mathbf{S}_{m\times n}$, $\mathbf{C}_0 \in \mathbf{N}^*_{m\times n}$, $\mathbf{C}_p^{(1)} \in \mathbf{N}^*_{m\times n}$, and $\mathbf{B} \in \mathbf{N}^*_{m\times n}$.
Then

$\Phi_{\mathbf{B}}(\mathbf{C}_p^{(1)} + \mathbf{C}_0) = \mathbf{C}_1$ is the first level of codification of $\mathbf{C}_0$.
We take $R(\mathbf{C}_1) = \mathbf{L}_1$

Initial data: $\mathbf{L}_1 \in \mathbf{S}_{m\times n}$, $\mathbf{C}_1 \in \mathbf{N}^*_{m\times n}$, $\mathbf{C}_p^{(2)} \in \mathbf{N}^*_{m\times n}$, and $\mathbf{B} \in \mathbf{N}^*_{m\times n}$.
Then

$\Phi_{\mathbf{B}}(\mathbf{C}_p^{(2)} + \mathbf{C}_1) = \mathbf{C}_2$ is the second level of codification of $\mathbf{C}_0$.
We take $R(\mathbf{C}_2) = \mathbf{L}_2$

Initial data: $\mathbf{L}_2 \in \mathbf{S}_{m\times n}$, $\mathbf{C}_2 \in \mathbf{N}^*_{m\times n}$, $\mathbf{C}_p^{(3)} \in \mathbf{N}^*_{m\times n}$, and $\mathbf{B} \in \mathbf{N}^*_{m\times n}$.
Then

$\Phi_{\mathbf{B}}(\mathbf{C}_p^{(3)} + \mathbf{C}_2) = \mathbf{C}_3$ is the third level of codification of $\mathbf{C}_0$.
We take $R(\mathbf{C}_3) = \mathbf{L}_3$,

And so on.

Note that after $k-1$ levels of codification of matrix $\mathbf{C}_0$, we have

Initial data: $\mathbf{L}_{k-1} \in \mathbf{S}_{m\times n}$, $\mathbf{C}_{k-1} \in \mathbf{N}^*_{m\times n}$, $\mathbf{C}_p^{(k)} \in \mathbf{N}^*_{m\times n}$, and $\mathbf{B} \in \mathbf{N}^*_{m\times n}$.
Then

$\Phi_{\mathbf{B}}(\mathbf{C}_p^{(k)} + \mathbf{C}_{k-1}) = \mathbf{C}_k$ is the $k-$–th level of codification of$\mathbf{C}_0$.
We take $R(\mathbf{C}_k) = \mathbf{L}_k$

Also note that at the $k-$ th level of codification, we get:

$$\begin{aligned}
\mathbf{C}_k &= \Phi_{\mathbf{B}}(\mathbf{C}_p^{(k)} + \mathbf{C}_{k-1}) = \Phi_{\mathbf{B}}(\mathbf{C}(_p^{(k)} + \underbrace{\Phi_{\mathbf{B}}(\mathbf{C}_p^{(k-1)} + \mathbf{C}_{k-2})}_{\mathbf{C}_{k-1}}) \\
&= \Phi_{\mathbf{B}}(\mathbf{C}_p^{(k)} + \Phi_{\mathbf{B}}(\mathbf{C}_p^{(k-1)} + \underbrace{\Phi_{\mathbf{B}}(\mathbf{C}_p^{(k-2)} + \mathbf{C}_{k-3})}_{\mathbf{C}_{k-2}})) \\
&= \ldots \underset{=}{\text{Theo.}} 3.13\ \Phi_{\mathbf{B}}(\underbrace{\mathbf{C}_p^{(k)} + \mathbf{C}_p^{(k-1)} + \mathbf{C}_p^{(k-2)} + \ldots + \mathbf{C}_p^{(1)}}_{k- \text{ addends}} + \mathbf{C}_0) \\
&= \Phi_{\mathbf{B}}(\textstyle\sum_{m=1}^{k}\mathbf{C}_p^{(m)} + \mathbf{C}_0).
\end{aligned}$$

This fact motivated Definition 3.7.

On the next, we will denote by $[x]_{pe}$ the entire part of a real number x, that is

$$[x]_{pe} = máx\left\{k \in \mathbf{Z}, \text{ such that } k \leq x\right\}.$$

The next Theorem is fundamental since it gives a way to decode a matrix $\mathbf{C}_k$ codified at the $k-th$ level. Without loosing generality, here we give the theorem at the first level of codification.

Theorem 3.8. Given $\mathbf{C}_p^{(1)} = (c_{ij}^{(1)}) \in \mathbf{N}_{m\times n}^*$, personal code matrix , and $\mathbf{B} \in \mathbf{N}_{m\times n}^*$ a base matrix of the type

$$\mathbf{B} = \begin{pmatrix} 28 & 28 & 28 & \dots & 28 & 28 \\ 28 & 28 & 28 & \dots & 28 & 28 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 28 & 28 & 28 & \dots & 28 & 28 \end{pmatrix}_{m\times n},$$

$\mathbf{L}_0 \in \mathbf{S}_{m\times n}$ a language matrix, and $\mathbf{C}_d^{(1)} = d\mathbf{B} - \mathbf{C}_p^{(1)} \in \mathbf{N}_{m\times n}^*$ a matrix of "decoding", with

$$d = \left[\frac{\max\left\{c_{ij}^{(1)} : 1 \leq i \leq m;\ 1 \leq j \leq n\right\}}{28} + 1\right]_{pe}.$$

Initially take $\mathbf{C}_0 = F(\mathbf{L}_0) \in \mathbf{N}_{m\times n}^*$, such that

$$\begin{cases} \Phi_{\mathbf{B}}(\mathbf{C}_p^{(1)} + \mathbf{C}_0) = \mathbf{C}_1, \\ R(\mathbf{C}_1) = \mathbf{L}_1 \in \mathbf{S}_{m\times n}. \end{cases}$$

Then

$$\begin{cases} \Phi_{\mathbf{B}}(\mathbf{C}_d^{(1)} + \mathbf{C}_1) = \mathbf{C}_0, \\ R(\mathbf{C}_0) = \mathbf{L}_0 \in \mathbf{S}_{m\times n}. \end{cases} \tag{3.1}$$

Remark 3.9.

(i) Do not forget, in identity (4), matrix $\mathbf{C}_1 = (r_{ij}) \in \mathbf{N}_{m\times n}^*$ with $r_{ij} < 28$, $1 \leq i \leq m$, $1 \leq j \leq n$, represents the resulting matrix of codifying of message contained in $\mathbf{L}_0$, by $\Phi_{\mathbf{B}}$ using the personal code matrix $\mathbf{C}_p^{(1)} \in \mathbf{N}_{m\times n}^*$. Matrix $\mathbf{C}_d^{(1)}$, in (4) induces the decoding, by $\Phi_{\mathbf{B}}$, of matrix $\mathbf{C}_1$. Theorem 3.8 guarantees that, with this choosing of $\mathbf{C}_d^{(1)}$ we return to matrix $\mathbf{C}_0$.

(ii) Theorem 3.8 gives an algorithmic mathematical technique (the algorithmic matrix function), based on the Euclidean algorithm, which let us decode a codified message. This is carried out by the algorithmic matrix function $\Phi_{\mathbf{B}}$. In other words, what is possible codify by $\Phi_{\mathbf{B}}$ using $\mathbf{C}_p^{(1)}$, can be decoded by $\Phi_{\mathbf{B}}$ using $\mathbf{C}_d^{(1)}$, and conversely.

Proof of Theorem 3.8. Suppose the first level of codification of matrix $\mathbf{C}_0$ was carried out, that is, identity $\Phi_{\mathbf{B}}(\mathbf{C}_p^{(1)} + \mathbf{C}_0) = \mathbf{C}_1$ holds on, then

$$\Phi_{\mathbf{B}}(\mathbf{C}_1 + \mathbf{C}_d^{(1)}) Hip = \Phi_{\mathbf{B}}(\underbrace{\Phi_{\mathbf{B}}(\mathbf{C}_p^{(1)} + \mathbf{C}_0)}_{\mathbf{C}_1} + \mathbf{C}_d^{(1)})$$

$$\underset{=}{(d)}\ \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{C}_0 + \mathbf{C}_p^{(1)})) + \Phi_{\mathbf{B}}(\mathbf{C}_d^{(1)}))$$

$$\underset{=}{(d-g)}\ \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{C}_0 + \mathbf{C}_p^{(1)}) + \Phi_{\mathbf{B}}(\mathbf{C}_d^{(1)})).$$

Hence,

$$\begin{aligned}
\Phi_{\mathbf{B}}(\mathbf{C}_1 + \mathbf{C}_d^{(1)}) &= \Phi_{\mathbf{B}}((\mathbf{C}_0 + \mathbf{C}_p^{(1)}) + \mathbf{C}_d^{(1)}) \\
&= \Phi_{\mathbf{B}}(\mathbf{C}_0 + \mathbf{C}_p^{(1)} + d\mathbf{B} - \mathbf{C}_p^{(1)}) \\
&= \Phi_{\mathbf{B}}(\mathbf{C}_0 + d\mathbf{B} + \mathbf{O}).
\end{aligned}$$

Then,

$$\begin{aligned}
\Phi_{\mathbf{B}}(\mathbf{C}_1 + \mathbf{C}_d^{(1)}) &= \Phi_{\mathbf{B}}(\mathbf{C}_0 + d\mathbf{B}) \\
&\underset{=}{(d)}\ \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{C}_0) + \Phi_{\mathbf{B}}(d\mathbf{B})) \\
&\underset{=}{(b)}\ \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{C}_0) + \mathbf{O}) \\
&= \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{C}_0)).
\end{aligned}$$

And finally,

$$\Phi_{\mathbf{B}}(\mathbf{C}_1 + \mathbf{C}_d^{(1)}) = \Phi_{\mathbf{B}}(\Phi_{\mathbf{B}}(\mathbf{C}_0))(c) = \Phi_{\mathbf{B}}(\mathbf{C}_0)(g) = \mathbf{C}_0.$$

The last equality is due to item (g) of Theorem 2.13, since all the components of matrix $\mathbf{C}_0 = (q_{ij}) = F(\mathbf{L}_0) \in \mathbf{N}^*_{m\times n}$ are $q_{ij} < 28$, $1 \leq i \leq m, 1 \leq j \leq n$. □

To try reading a codified message, before all, we have to decode it; the resulting matrix of decoding would have its components in $\mathbf{N}^*$. To try reading this matrix and, for it, the message, we have to apply to this last matrix, the return function R. All this ideas are illustrated in the next example.

Example 3.10. Suppose a colleague from Universidad de Antofagasta (Chile) send us, by e-mail, a codified message which is contained in the language matrix

$$\mathbf{L} = \begin{pmatrix} i & a & k & x & f & o & c & n & d & e & z & r & z & n \\ v & \& & l & z & o & z & c & p & u & l & r & y & p & y \end{pmatrix} \in S_{2\times 14}.$$

Our colleague, tell us the message has been codified by $\Phi_{\mathbf{B}}$, and should be read, it means decoded, only at the third level of decoding; and each level of decoding should be carried out using a personal code matrix of the type:

$\mathbf{C}_p = (c_{ij}^{(p)}) =$

$$\begin{pmatrix} 19 & 23 & 9 & 8 & 2 & 3 & 4 & 13 & 17 & 20 & 21 & 25 & 37 & 14 \\ 2 & 4 & 7 & 9 & 13 & 21 & 22 & 24 & 29 & 32 & 34 & 12 & 17 & 15 \end{pmatrix} \in \mathbf{N}^*_{2\times 14}.$$

This information gets by Internet. Therefore, if we want to read the message, we should be able of building the matrix $\mathbf{C}_1$ and then $R(\mathbf{C}_1)$.

If this is the purpose, our first step should be to determinate the transformed of $\mathbf{L} = \mathbf{L}_4$ by F. If we do so, we get the codifying matrix $F(\mathbf{L}_4) = \mathbf{C}_4$, and then

$$\mathbf{C}_4 = \begin{pmatrix} 9 & 1 & 11 & 25 & 6 & 16 & 3 & 14 & 4 & 5 & 27 & 19 & 27 & 14 \\ 23 & 0 & 12 & 27 & 16 & 27 & 3 & 17 & 22 & 12 & 19 & 26 & 17 & 26 \end{pmatrix} \in \mathbf{N}^*_{2\times 14}.$$

To decoding the initial message, we will use Theorem 3.8 several times. In fact, let $\mathbf{B} \in \mathbf{N}^*_{2\times 14}$, a base matrix given by

$$\mathbf{B} = \begin{pmatrix} 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 \\ 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 & 28 \end{pmatrix}_{2\times 14},$$

the decoding matrix $\mathbf{C}_d$, is calculated using the formula $\mathbf{C}_d = d\mathbf{B} - \mathbf{C}_p$, with

$$d = \left[\frac{\max\left\{c_{ij}^{(p)} : 1 \le i \le 2;\, 1 \le j \le 14\right\}}{28} + 1\right]_{pe} = \left[\frac{37}{28} + 1\right]_{pe} = [1.3214 + 1]_{pe} = 2 \tag{3.2}$$

thus $\mathbf{C}_d \in \mathbf{N}^*_{2\times 14}$, and takes the form:

$$\mathbf{C}_d = \begin{pmatrix} 37 & 33 & 47 & 48 & 54 & 53 & 52 & 43 & 39 & 36 & 35 & 31 & 19 & 42 \\ 54 & 52 & 49 & 47 & 43 & 35 & 34 & 32 & 27 & 24 & 22 & 44 & 39 & 41 \end{pmatrix}.$$

Hence we have,
$\Phi_{\mathbf{B}}(\mathbf{C}_4 + \mathbf{C}_d) = \mathbf{C}_3 =$

$$\begin{pmatrix} 18 & 6 & 2 & 17 & 4 & 13 & 27 & 1 & 15 & 13 & 6 & 22 & 18 & 0 \\ 21 & 24 & 5 & 18 & 3 & 6 & 9 & 21 & 21 & 8 & 13 & 14 & 0 & 11 \end{pmatrix},$$

this induces the language matrix

$$\mathrm{R}(\mathbf{C}_3) = \mathbf{L}_3 = \begin{pmatrix} q & f & b & p & d & m & z & a & \tilde{n} & m & f & u & q & \& \\ t & w & e & q & c & f & i & t & t & h & m & n & \& & k \end{pmatrix}.$$

It is evident the hidden message in $\mathbf{L}_3$, at the first level of decoding, is incomprehensible in the Spanish language, so we have to go to the second level of codification, for this, we again considerate the matrix:

$$\mathbf{C}_3 = F(\mathbf{L}_3) = \begin{pmatrix} 18 & 6 & 2 & 17 & 4 & 13 & 27 & 1 & 15 & 13 & 6 & 22 & 18 & 0 \\ 21 & 24 & 5 & 18 & 3 & 6 & 9 & 21 & 21 & 8 & 13 & 14 & 0 & 11 \end{pmatrix}$$

Now, using again the decoding matrix $\mathbf{C}_d$,

$$\mathbf{C}_d = \begin{pmatrix} 37 & 33 & 47 & 48 & 54 & 53 & 52 & 43 & 39 & 36 & 35 & 31 & 19 & 42 \\ 54 & 52 & 49 & 47 & 43 & 35 & 34 & 32 & 27 & 24 & 22 & 44 & 39 & 41 \end{pmatrix}$$

we get,

$\Phi_{\mathbf{B}}(\mathbf{C}_3 + \mathbf{C}_d) = \mathbf{C}_2 =$

$$\begin{pmatrix} 27 & 11 & 21 & 9 & 2 & 10 & 23 & 16 & 26 & 21 & 13 & 25 & 9 & 14 \\ 19 & 20 & 26 & 9 & 18 & 13 & 15 & 25 & 20 & 4 & 7 & 2 & 11 & 24 \end{pmatrix}.$$

Therefore,

$$R(\mathbf{C}_2) = \mathbf{L}_2 = \begin{pmatrix} z & k & t & i & b & j & v & o & y & t & m & x & i & n \\ r & s & y & i & q & m & \tilde{n} & x & s & d & g & b & k & w \end{pmatrix}.$$

Again, we observe the hidden message in $\mathbf{L}_2$, at the second level of decoding, keeps being imprecise; this take us to the third level of decoding. For it we determine a third matrix

$$\mathbf{C}_2 = F(\mathbf{L}_2) = \begin{pmatrix} 27 & 11 & 21 & 9 & 2 & 10 & 23 & 16 & 26 & 21 & 13 & 25 & 9 & 14 \\ 19 & 20 & 26 & 9 & 18 & 13 & 15 & 25 & 20 & 4 & 7 & 2 & 11 & 24 \end{pmatrix}$$

and using again the decoding matrix

$$\mathbf{C}_d = \begin{pmatrix} 37 & 33 & 47 & 48 & 54 & 53 & 52 & 43 & 39 & 36 & 35 & 31 & 19 & 42 \\ 54 & 52 & 49 & 47 & 43 & 35 & 34 & 32 & 27 & 24 & 22 & 44 & 39 & 41 \end{pmatrix}$$

we get,

$$\Phi_{\mathbf{B}}(\mathbf{C}_2 + \mathbf{C}_d) = \mathbf{C}_1 = \begin{pmatrix} 8 & 16 & 12 & 1 & 0 & 7 & 19 & 3 & 9 & 1 & 20 & 0 & 0 & 0 \\ 17 & 16 & 19 & 0 & 5 & 20 & 21 & 1 & 19 & 0 & 1 & 18 & 22 & 9 \end{pmatrix}.$$

From this operation, a new language matrix surges:

$$R(\mathbf{C}_1) = \mathbf{L}_1 = \begin{pmatrix} h & o & l & a & \& & g & r & c & i & a & s & \& & \& & \& \\ p & o & r & \& & e & s & t & a & r & \& & a & q & u & i \end{pmatrix}.$$

Hence, the message sent by our colleague is:

HOLA GRCIAS POR ESTAR AQUI.

Therefore, the process has to finish, and we must be able to perceive the message sent by our colleague is:

HOLA, GRACIAS POR ESTAR AQUI.

Remark 3.11.

(i) Note, the personal code matrix can change at each level of decoding or codification. Just to simplify operations, here we keep it constant.

(ii) Also – as it will be showed in a future work – the algorithmic matrix function $\Phi_{\mathbf{B}}$ has other applications, one of them relates to the possibility of codifying and decoding a message in a partial way.

4. Applications of the algorithmic scalar function to non-linear Diophantine equations: a world of conjectures

The expression "Diophantine Equation" proceeds from Diofantos of Alejandría (255 A.D.), one of the great mathematicians of the Greek civilization. He was the first mathematician who began a systematic study of equations with positive entire solutions. He found the first great results on this area. He wrote three important works on this line, being the most important what nowadays is known as "Arithmetic". In this work, Diofantos gives entire solutions for linear equations (even of higher order); this results were the mathematical guiding light which guided to the Number Theory, until the French mathematician Pierre de Fermat (1601–1665) entered the mathematic scenery.

A Diophantine equation is a mathematical expression which have to be solved only with natural (entire) numbers , [0, 7, 0].

A fundamental problem of the theory associated to Diophantine equations, surges from the next question: Given a Diophantine equation (or system of equations), is there or not possibility of solutions?.

Here we face some problems related to this kind of questions. In our treatment we use the algorithmic scalar function as a basic tool.

On this context, the problems here studied are associated to the so called "Beal's and Fermat's Conjectures", see [1, 4, 5, 6] and their references.

Beal's Conjecture

Let A, B, C, x, y, z be all natural numbers, x, y, $z > 2$. If
$\mathrm{A}^x + B^y = C^z$,
then A, B y C have a common factor.

An equivalent way of writing the Beal's Conjecture is:

The equation $\mathrm{A}^x + B^y = C^z$, has not solution for natural numbers A, B, C, x, y, z , with x, y, $z > 2$ and A, B , C co-prime.

Fermat's Conjecture

This conjecture states that equation
$\mathrm{A}^n + B^n = C^n$, has not solutions A, B, $C \in \mathbf{N}$ with $n \in \mathbf{N}$ and $n > 2$.

Remark 4.1.

(i) To prove the Beal's and Fermat's conjectures is enough to considerate all exponents x, y, z and n prime.

(ii) Fermat's conjecture – also called by the number theoreticians as "Last Fermat's Theorem"– has been recently proved (or proved again) using mathematic tools of high level. The mathematician who achieved this deed is the English A. Walis (1996) by proving for it another conjecture called The Taniyama–Shimura-Weil Conjecture.

(iii) We do not pretend to solve this conjectures; however, we give a short Theorem which involves both of them. To prove this Theorem, we only use the concept of algorithmic scalar function. In the immediate future we hope to continue studying these and other interesting problems.

Theorem 4.2. Let $A \in \mathbf{N}$ be an even number and $B \in \mathbf{N}$, $C \in \mathbf{N}$ odd numbers, such that $\phi_8(B \times C) \neq 1$. Also, let x, y, z be all prime greater than 2. Then, the Diophantine equation $\mathrm{A}^x + B^y = C^z$, does not have solution.

Corollary 4.3. Let $A \in \mathbf{N}$ be an even number and $B \in \mathbf{N}$, $C \in \mathbf{N}$ odd numbers. If $\phi_8(B \times C) \neq 1$, then the Fermat's equation $\mathrm{A}^p + B^p = C^p$, does not have solution when p is a prime greater than 2.

Theorem 4.4 Let $A \in \mathbf{N}$ be an even number and $B \in \mathbf{N}$, $C \in \mathbf{N}$ odd numbers, such that $\phi_7(A \times B \times C) \neq 0$. Then the Diophantine equation $\mathrm{A}^3 + B^3 = C^3$, does not have solution.

Before proving this results, we will need two technical Lemmas:

Lemma 4.5. Let $B \in \mathbf{N}$ be an odd number, then $\phi_8(B^2) = 1$, and in general $\phi_8(B^{2n}) = 1$ for all $n \in \mathbf{N}^*$.

Proof: We use Theorem 2.8, together to the definition of algorithmic scalar function ϕ_b and Corollaries 2.10–2.11. In fact, if $B \in \mathbf{N}$ is odd, then it has the form $B = 2k + 1$, with $k \in \mathbf{N}$. Now suppose that $k \in \mathbf{N}$ is even, this implies that $k = 2t$, for some $t \in \mathbf{N}$. Thus, $B^2 = 16t^2 + 8t + 1$. Therefore, for each even $k \in \mathbf{N}$, we have

$$\phi_8(B^2) = \phi_8(16t^2 + 8t + 1) \underset{(2.10)}{=} \phi_8(\phi_8(16t^2) + \phi_8(8t) + \phi_8(1)) = \underset{(b)}{=} \phi_8(0 + 0 + \phi_8(1)) = \phi_8(\phi_8(1)) \underset{(c)}{=} \phi_8(1) \underset{def}{=} 1.$$

If $k \in \mathbf{N}$ is odd, that is having the form $k = 2t + 1$, then

$$B^2 = 4(2t + 1)^2 + 4(2t + 1) + 1,$$

and then we have

$$B^2 = 16t^2 + 16t + 4 + 8t + 4 + 1 = 16t^2 + 24t + 8 + 1$$

Hence, for all odd $k \in \mathbf{N}$ we have

$$\begin{aligned} \phi_8(B^2) &= \phi_8(16t^2 + 24t + 8{+}1) \underset{=}{2.10} \\ &= \phi_8(\phi_8(16t^2) + \phi_8(24t) + \phi_8(8) + \phi_8(1)) = \\ \underset{=}{(b)}\ \phi_8(0 + 0 + 0 + \phi_8(1)) &= \phi_8(\phi_8(1)) = \phi_8(1) \underset{=}{def}\ 1. \end{aligned}$$

This prove the firs part of Lemma 4.5.

To the second part, we use Corollary 2.11. In fact, let $z = B^2$ where $B \in \mathbf{N}$ is an odd number, then $\phi_8(z) = 1$. Hence,

$$\phi_8(B^{2n}) = \phi_8(z^n) \underset{=}{2.11}\ \phi_8([\phi_8(z)]^n) = \phi_8(1^n) = \phi_8(1) \underset{=}{def}\ 1$$

for all $n \in \mathbf{N}^*$. This proves the Lemma.

Corollary 4.6. Let $B \in \mathbf{N}$ an odd number, then $\phi_8(B^3) = \phi_8(B)$, and in general $\phi_8(B^{2n+1}) = \phi_8(B)$ for all $n \in \mathbf{N}^*$.

Proof:

Note that

$$\begin{aligned} \phi_8(B^3) &= \phi_8(B^2 \times B) = \phi_8(\phi_8(B^2) \times \phi_8(B)) \\ \underset{=}{\text{lema } 4.5}\ \phi_8(1 \times \phi_8(B)) &= \phi_8(\phi_8(B)) = \phi_8(B). \end{aligned}$$

In a similar way, we have,

$$\begin{aligned} \phi_8(B^{2n+1}) &= \phi_8(B^{2n} \times B) = \phi_8(\phi_8(B^{2n}) \times \phi_8(B)) \\ \underset{=}{\text{lema } 4.5}\ \phi_8(1 \times \phi_8(B)) &= \phi_8(\phi_8(B)) = \phi_8(B). \end{aligned}$$

Lemma 4.7. Let $B \in \mathbf{N}$ an odd number, such that $B < 8$, then $\phi_8(B) = B$.

Proof: The proof of this Lemma is evident, it comes straightforward from definition of the algorithmic scalar function ϕ_b with $b = 8$.

Lemma 4.8. Let $B \in \mathbf{N}$ an even number, then $\phi_8(B^3) = 0$, and in general $\phi_8(B^{2n+1}) = 0$ for all $n \in \mathbf{N}$.

Proof If $B \in \mathbf{N}$ is an even number, then $B = 2k$, thus $\phi_8(B^3) = \phi_8(8k^3) = \phi_8(8z) = 0$, where $z = k^3 \in \mathbf{N}$. Similarly, we have

$$\phi_8(B^{2n+1}) = \phi_8(2^{2n+1}k^{2n+1}) = \phi_8(\phi_8(2^{2n+1}) \times \phi_8(k^{2n+1})) =$$

$$\phi_8(0 \times \phi_8(k^{2n+1})) = 0.$$

Lemma 4.9. Let $B \in \mathbf{N}$, such that $\phi_7(B) \neq 0$, then $\phi_7(B^3) = 1$ ó $\phi_7(B^3) = 6$.

Proof: The proof is similar to the proof of Lemma 4.5, and will be omitted.

Now, we prove Theorem 4.2. We shall assume it to be false and get a contradiction.

Proof of Theorem 4.2 Suppose there are $A_0 \in \mathbf{N}$ an even number and $B_0 \in \mathbf{N}$, $C_0 \in \mathbf{N}$ odd numbers with $\phi_8(B_0 \times C_0) \neq 1$, such that

$$A_0^x + B_0^y = C_0^z,$$

where the exponents x, y and z, are all prime greater than two. Then, since ϕ_8 is function, we have that

$$\phi_8(A_0^x + B_0^y) = \phi_8(C_0^z),$$

hence $\phi_8(\phi_8(A_0^x) + \phi_8(B_0^y)) = \phi_8(C_0^z)$. Now, since A_0 is even, Lemma 4.8 implies $\phi_8(A_0^x) = 0$. Therefore, we straightly have:

$$\begin{aligned} (4.1)\ \phi_8(C_0^z) &= \phi_8(A_0^x + B_0^y) \\ (4.2) &= \phi_8(\phi_8(A_0^x) + \phi_8(B_0^y)) = \phi_8(0 + \phi_8(B_0^y)) = \phi_8(B_0^y), \end{aligned}$$

because of $\phi_8(\phi_8(B_0^y)) = \phi_8(B_0^y)$. Now, due to Corollary 4.6 and the oddness of B_0 and C_0, we can replace in expression (4.1) $\phi_8(B_0^y)$ by $\phi_8(B_0)$ and by other hand, replace $\phi_8(C_0^z)$ by $\phi_8(C_0)$. Therefore, from (4.1) we deduce $\phi_8(B_0) = \phi_8(C_0)$. Now, multiply this equality by C_0, and have

$$(4.2) \qquad C_0 \times \phi_8(B_0) = C_0 \times \phi_8(C_0).$$

One more time, apply ϕ_8 to the expression (4), and get

$$\phi_8(C_0 \times \phi_8(B_0)) = \phi_8(C_0 \times \phi_8(C_0)) = \phi_8(\phi_8(C_0) \times \phi_8(C_0)) = \phi_8(C_0^2) = 1.$$

But, by other hand,
$\phi_8(C_0 \times \phi_8(B_0)) = \phi_8(\phi_8(C_0) \times \phi_8(B_0)) = \phi_8(C_0 \times B_0)$.
That is,

$$1 = \phi_8(C_0 \times B_0) = \phi_8(B_0 \times C_0).$$

This clearly contradicts the hypothesis of Theorem 4.2. Consequently, we can say that Beal's Conjecture, under our assumptions, is true. □

Remark 4.10:

(i) There are endless products of two odd numbers which satisfy the condition $\phi_8(B_0 \times C_0) \neq 1$, for example, 3×7, 5×11, etc. But there also have endless products of two odd numbers which satisfy $\phi_8(B_0 \times C_0) = 1$, for example, 3×11, 17×1, etc.

(ii) Proofs of Corollary 4.3 and Theorem 4.4, are similar to the former proof, so they are omitted here. Though, did note that Corollary 4.3 is a straight consequence of Theorem 4.2. In fact, it is enough to do $x = y = z = p$

(iii) On this part of the work, we wanted to motivate using the algorithmic scalar function for facing these kind of conjectures. For other non-linear problems, linear diophantine equations, we think the algorithmic scalar and matrix function should be useful. Showing this, will be theme for another investigation.

5. Equivalences between the algorithmic scalar function and congruence module p of Carl F. Gauβ

The mathematician Johann Carl Friedrich Gauβ (Gauss, 1777–1855), nicknamed the prince of mathematics, did very important contributions in several fields of Mathematics and Physics, that is why he es considered as one of the more important scientists of nineteenth century.

Among his multiples findings there are many results on divisibility. A very interesting contribution on this area was to define the concept of **Congruence Module** p, which is associated to division of two entire numbers. Specifically,

Definition 5.1. (K. F. Gauβ).
Let a and b two entire numbers. We will say a is **congruent** to b module p if $a - b$ is divisible by p.

Remark 5.2.

(a) From definition 5.1 a is **congruent** to b module p if and only if there is a $k \in \mathbf{Z}$ such that $a = b + kp$.

(b) Given a certain entire number p, the set of the entire numbers can be divided in classes of equivalences in which the elements of the same class are those having the same residue when are divided by p. In Number theory, the set of classes of equivalences is represented by $\mathbf{Z}/[p]$, which must be read $\mathbf{Z}$ module p. When two entire numbers belongs to the same class $\in \mathbf{Z}/[p]$ we say that are congruent module p,

Theorem 5.3 (Equivalence between ϕ_b and the congruence module p)
Let $a, m \in \mathbf{N}^*$ with $a \geq p$ and $p \geq 2$, such that,

$$a = r_0 + r_1 p + r_2 p^2 + \ldots + r_{n-1} p^{n-1} + r_n p^n,$$

with $0 \leq r_i \leq p - 1$, for all $0 \leq i \leq n$, then

$$\phi_m(a - r_0) = 0 \text{ if and only if } a \text{ is congruent to } r_0 \text{ módule } p.$$

Remark 5.4. The proof of Theorem 5.3 is straightforward. Also note, working with natural numbers, we get the triple equivalence:

$$\begin{aligned} a \text{ is congruent to } r_0 \text{ module } p &\iff \phi_p(a - r_0) = 0 \\ &\iff \text{There is } q \in \mathbf{N}^*, \text{ such that } a = r_0 + qp. \end{aligned}$$

In other words, the algorithmic scalar function defined in section §2 , is between the congruence module and the Euclidean algorithm. And If we only work in $\mathbf{N}$, is logic to think that, in some way, it must be possible to show, by ϕ_m, the great part of results of Arithmetic related to the concept of congruence module p. Because of its importance, this has to be a theme for future investigation. On this way, our group of work has made some advances with the pre print titled "Entre la congruencia de Gauβ y el algoritmo de la división", see [8].

6. Appendix

```
Program Resto(input, output) ;
    Var  a, b, q, r:  Integer ;
    Begin
    Writeln ( 'queremos determinar el resto r = φ_b(a)') ;
    Writeln ('cuando el entero a se divide entre el entero positivo b') ;
    Write ('a =') ;
    Read (a) :
    Write ('b =') ;
    Read (b) :
    If a = 0 then
      Writeln ( 'En este caso, a = 0, por lo que q = 0 y r = 0.')
    Else
    Begin
        r := abs(a) ;
        q := 0 ;
    While (r >= b) do
     Begin
        r := r − b ;
        q := q + 1 ;
      End
      if a > 0 then
      Begin
       Writeln ( 'cuando dividimos', a : 0, 'entre', b : 0.  '.') ;
       Writeln (´el resto r =′, r : 0,'.') ;
      End
      Else if r = 0 then
      Begin
       Writeln ( 'cuando dividimos', a : 0, 'entre', b : 0.  '.') ;
       Writeln ('el resto r = 0.') ;
      End
      Else
      Begin
       Writeln ( 'cuando dividimos', a : 0, 'entre', b : 0.  '.') ;
       Writeln ('el resto r = φ_b(a) =′, b − r : 0,'.') ;
       End: End : End
```

Queremos determinar el resto $r = \phi_b(a)$
cuando el entero a se divide entre el entero positivo b.

$a = 43$
$b = 8$
Cuando dividimos 43 entre 8,
el resto $r = \phi_b(43) = 3$

Acknowledgment

This work is supported by projects of the Dirección de Investigación de la **Universidad de Antofagasta**, Chile, DIRINV-UA, N^o 1319-06 and a Grant by CONICYT–FONDECYT N^o 1040067. In Perú: Consejo Superior de Investigación de la **Universidad Nacional Mayor de San Marcos**, Perú, CSI–Estudio N^o 071401051, respectively.

References

[1] R. D, Mauldin, A Generalization of Fermat's Last Theorem: The Beal's Conjeture and Prize Problem. Notices of the AMS, 44, N^o 11, pp, 1436–1437, (1997).

[2] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, (1996).

[3] C. Mercado, Historia de las Matemáticas, Ed. Universitaria, Santiago de Chile, (1972).

[4] S. Lang, Old and new conjetured diophantine inequalities, Bull. Amer. Math. Soc, 23, pp, 37–75, (1990)

[5] A. Walis, Modular elliptic curves and Fermat's Last Theorem, Ann. Math. 141 (1995), pp. 443–551, (1995).

[6] S. Singh, O último Teorema de Fermat, Ed. Record, Rio de Janeiro, (1999).

[7] N. L. Biggs, Matemática Discreta, Ed. Vicens–Vives, (1994).

[8] L. A. Cortés Vega, D. E. Rojas Castro, Y. S. Santiago Ayala and S. C. Rojas Romero, Entre la congruencia de Gauβ y el Algoritmo de Euclides: Una Observación, Pre-Print, (2007).

[9] L. A. Cortés–Vega, D. E. Rojas–Castro, The Quotient Function and its applications to some problems yielding in Discrete Mathematics and Artificial Intelligence, Pre-print, (2007).

[10] B. Kolman, R. S. Busby y S. C. Ross, Estructura de matemáticas discretas para la computación, Prentice Hall, (1887).

[11] R. Graham, D. E. Knuth y O. Patashnik, Concrete Mathematics: A fondation for Computer Science, Addison-Wesley, (1994).

[12] D. Burton, Elementary Number Theory, Ed. Allyn–Bacon, (1980).

[13] A. Hodges, Alan Turing: The Enigma of Intelligence, Ed. Unwin–Paperbacks, (1983).

[14] R. P. Grimaldi, Discrete and Combinatorial Mathematics. An Applied Introduction, Addison-Wesley, (1994).

Luis A. Cortés Vega
Departamento de matemáticas
Facultad de Ciencias Básicas
Universidad de Antofagasta
Casilla 170
Chile
e-mail : lcortes@uantof.cl

Daniza E. Rojas Castro
Departamento de matemáticas
Facultad de Ciencias Básicas
Universidad de Antofagasta
Casilla 170
Chile
e-mail : drojas@uantof.cl

Yolanda S. Santiago Ayala
Facultad de Ciencias Matemáticas
Universidad Nacional Mayor de San Marcos
Lima-Perú
e-mail: ysantiagoa@unmsm.edu.pe

and

Santiago C. Rojas Romero
Facultad de Ciencias Matemáticas
Univesidad Nacional Mayor de San Marcos
Lima
Perú
e-mail: srojasr@unmsm.edu.pe