Tema

Tendências em Matemática Aplicada e Computacional, **18**, N. 1 (2017), 105-125

www.scielo.br/tema
doi: 10.5540/tema.2017.018.01.0105

Optimal Check Digit Systems Based on Modular Arithmetic[†]

L. FARIA, P.E.D. PINTO and N. PEDROZA*

Received on November 15, 2016 / Accepted on February 19, 2017

ABSTRACT. In this article, we discuss check digits systems based on modular arithmetic, which are used worldwide. Check digits were created to eliminate most errors in data input of computational systems. Though old, a discussion about the optimality of the systems used is not found in the literature. We describe the main existing systems worldwide and highlight those adopted in Brazil. We present the necessary improvements in order to make all systems optimal. We also propose a new optimal system with 3 permutations for systems with modular base 10.

Keywords: Check digits, error detection, modular arithmetic.

1 INTRODUCTION

Check digits (CDs) are one or more characters, numeric or alphanumeric, which are added to an original set of characters to verify that information has been correctly entered to computer systems. They are used everywhere such as in product bar codes, bank accounts, identity documents and in some important identifiers in Brazil, the CPF, witch is the Brazilian individual taxpayer registry identification and the CNPJ, witch is an identification number issued to Brazilian companies by the Secretariat of the Federal Revenue. The main problems are due to human error, which have been categorized by Verhoeff [15] and have been presented here, along with their frequencies in Table 1.

Phonetic errors are related to the English, Dutch and German languages, where there are numbers with very similar pronunciation, as in the case of thirty and thirteen in English. All random errors distinct from those described in Table 1 were grouped as Others and have not been examined.

We shall consider as optimal a system which allows detection, in weighted terms, of the maximum possible number of errors of the 6 first classes in Table 1.

The check digit systems found in the literature are based on three branches of Mathematics: Modular Arithmetic, Group Theory and Latin Squares, [2, 4, 14, 16]. Even though there are

[†] Partially financed by FAPERJ; Paper presented at CNMAC 2016.
*Corresponding author: Natália Pedroza – E-mail: npsnatalia@gmail.com
IME/DICC – Universidade do Estado do Rio de Janeiro, 20550-900 Rio de Janeiro, RJ, Brasil.
E-mails: luerbio@cos.ufrj.br; pauloedp@ime.uerj.br

Table 1: Error types and their frequencies.

Error type	Abrev	Format	Frequency (%)
Single errors	SNG	..a.. → ..b..	79.10
Adjacent transposition	TRA	..ab.. → ..ba..	10.20
Alternate transpostion	TRL	..abc.. → ..cba..	0.80
Adjacent twins	TWA	..aa.. → ..bb..	0.50
Alternate twin	TWL	..aca.. → ..bcb..	0.30
Phonetic	PHO	..1a.. → ..0a.., a≥ 2	0.50
Others	OTH		8.60

various theoretical proposals for the use of those last two branches in the check digit systems, we found only two practical applications using group theory. The first one is a Verhoeff system based on anti-symmetric mappings in the numbering used in Deutsche Mark banknotes. However, this system has been discontinued after the adoption of the euro in Germany, [13]. The second and more recent one is a check digit system used in *MISB Standard 1204.1* for the video community and is based on a system over an abelian group of arbitrary order created by [3] and is related to hexadecimal numbers.

Thus, the systems used in practice are in their vast majority based on Modular Arithmetic. However, for this class of codes, no article discussing their optimality has been found in the literature.

The goal of the present article is to promote this discussion. The conditions for optimality of the methods currently in use are characterized, particularly of methods with prime modular base and with base 10. Based on the examined conditions, proposals to improve the existing codes and which may also serve as a base for new methods are discussed. In the case of prime modular base, the problem of optimality is totally solved. In the case of modular base 10, optimality is solved in cases of small identifiers (with up to 6 digits) and a new system is presented which is optimal under certain restrictions, but it is conjectured to be optimal in general for larger identifiers (over 6 decimal digits).

This paper is structured as follows. In Section 2 we present various examples of practical uses of check digits based on modular arithmetic. In Section 3 we first deduce the theoretical capabilities of error detection for systems based on modular arithmetic and then we present a report of non-detection cases, especially for systems used in Brazil. In Section 4 we describe the methods developed in this work. First, an optimum system for prime bases and, next, two optimum systems for base 10. The first one was created by Verhoeff, and proved to be optimum by the authors, for identifiers with up to 6 digits. The second, created by the authors, is conjectured to be optimum for identifiers with more than 6 digits and proved to be optimum under the restriction of using only 3 permutations. In Section 5 we claim that the proposed methods can bring improvements for the check digit systems used worldwide.

2 SYSTEMS BASED ON MODULAR ARITHMETIC

They are the most widely disseminated [1], [5], [6], [7] and [11]. In these systems, arithmetic operations are conducted in the digits of identifiers and the results are considered as mod n, for $n \in \mathbb{N}$. Given an identifier with m digits $a_1a_2a_3 \cdots a_m$ and a sequence of weights p_i, the check digit a_m is determined by the *direct system* equation given by:

$$a_1p_1 + a_2p_2 + a_3p_3 + \cdots + a_{m-1}p_{m-1} \bmod n = a_m.$$

Or, in some other systems, by the equation of the *system by complement* given by:

$$a_1p_1 + a_2p_2 + a_3p_3 + \cdots + a_{m-1}p_{m-1} + a_m \equiv 0 \bmod n.$$

In a more comprehensive manner, we shall, rather than attributing weights to check digits, define functions σ_i, for each position $1 \leq i \leq m-1$. We justify below that, ideally, these functions are permutations applied to the modular base digits. Hence, the equation of the direct system may be re-written as follows:

$$\sum_{i=1}^{m-1} \sigma_i(a_i) \bmod n = a_m.$$

We describe the main variations of the methods which use modular base 10 or 11 in calculating the CDs, which are the most common.

2.1 Modulus 10

One of the most natural methods when working with identifiers composed of decimal digits is the one that uses modular base 10, since the check digit is also an element of the set. The effect of multiplying each digit by a weight p is equivalent to applying a function, which is a permutation whenever $mdc(10, p) = 1$. For instance, the multiplication by 3 of digits from 0 to 9 in mod 10 generates the same values as the application of permutation σ : (0 3 6 9 2 5 8 1 4 7) on digits 0 to 9.

The most common variants found use permutations indirectly through systems of 2 or 3 weights. The **product barcode identifier** internationally used for the commercial identification of products consists of 13 decimal digits. For the calculation of the check digit, an equation by complement is used, with weights 1 and 3, alternately applied. As an example, we show how the CD was calculated for the code 7 89102 711427 5. Initially, the 12 first digits are multiplied by 1 and 3:

	7	8	9	1	0	2	7	1	1	4	2	7
×	1	3	1	3	1	3	1	3	1	3	1	3
	7	24	9	3	0	6	7	3	1	12	2	21

Then the results are added, obtaining $7+24+9+3+0+6+7+3+1+12+2+21 = 95 \equiv 5 \bmod 10$. The value of the CD is the complement of 5 to 10, that is, 5.

Several systems that use mod 10 have been developed. In Mod 10 IBM, proposed by Luhn [9], once the algorisms of the identifier are alternately multiplied by weights 1 and 2, in each product, the "proof by nine" operation should be conducted, which consists in adding the algorisms of this product when it is above 9. This system is used in the document of the **General Register of the State of Rio de Janeiro**, which consists of a decimal number of 9 digits, used for civil identification. As an illustration, let us consider the identity number 21883353-1. We initially multiply each digit by its corresponding weight:

$$\begin{array}{cccccccccc} & 2 & 1 & 8 & 8 & 3 & 3 & 5 & 3 \\ \times & 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 \\ \hline & 2 & 2 & 8 & 16 & 3 & 6 & 5 & 6 \end{array}$$

We apply the "proof by nine" to 16, obtaining 1+6 = 7, and the sum of the results is $2+2+8+7+3+6+5+6 = 39 \equiv 9 \bmod 10$. We consider the check digit as the complement of 9 to 10, that is, 1, and therefore the equation of the system is satisfied.

Note that Mod 10 IBM system is equivalent to alternately applying two permutations, identity and σ : (0 2 4 6 8 1 3 5 7 9), to the identifiers' digits.

Another example of the use of mod 10 IBM system is in **credit card's** identification number, used in the bank system worldwide.

In order for a check digit system to detect alternate errors, it is necessary the use of at least three weights or that at least three permutations applied to the identifiers' digits. That is the case of the international **passport**, where the weights 7, 3 and 1 are alternately used. This identifier is formed by 10 characters, which may be capital letters from A to Z, digits from 0 to 9 and the symbol $<$. In order to calculate the CD, the alphanumeric characters must be converted into numbers. Letter A is mapped to 10, B to 11 and so on, and the character $<$ is mapped to 0.

It is clearly observed that, in this case, it is not possible to detect all single errors, since the mapping of several characters to the same decimal digit prevents detection of interchanges between these characters. For example, 0, A, K, U, $<$ are all mapped to multiples of 10 and, therefore, any interchange between them is not detected.

2.2 Modulus 11

When we use modulus 11 to calculate the check digit, considering numeric identifiers, we find a particularity. The rest of the division by 11 could be 10, which will require a special form of representing this situation. In order to use only one check digit, two solutions are adopted:

1. *Mod 11 complete* – Character X is used to represent rest 10.
2. *Mod 11 restrict* – Digit 0 is used to represent rest 10.

Each one of these solutions presents inconveniences in its use. In the case of mod 11 complete, a non numeric character is introduced, which may bring difficulties. That is the case, for instance,

of the transmission of information by telephone, where X has to be mapped to a decimal digit, transforming the situation into mod 11 restrict. When mod 11 restrict is adopted, it is not possible to detect error situations that change the calculation of 0 to 10, or vice-versa.

Examples of the use of mod 11 complete in Brazil are bank agency and bank account codes, with little variation between different banks. Cyclic weights from 1 to 10 are frequently used.

Another example of the use of complete mod 11 is the international book identifier **ISBN** (International Standard Book Number) adopted around the world, whose standardization is done by ISO (International Organization for Standardization). This standard has undergone variations over time, and the main ones are called ISBN10 (used until 2007), with 10 digits, and ISBN13 for 13 digits. In the identifier of ISBN10 standard, the first 9 digits denote the language, the editor and the serial number of the book. The calculation of the check digit is conducted with the mod 10 system by complement. The ISBN13 standard adopts the same mod 10 system that is used in product barcodes.

In some countries, variations of mod 11 system are used, with two check digits. For example, in Brazil, both the CPF and the CNPJ use 2 check digits. Considering the identifier as $a_1a_2a_3\cdots a_{m-1}a_m$, where a_{m-1} and a_m are the two check digits, the system's equations are as follows:

$$\begin{aligned} a_1p_1 + a_2p_2 + a_3p_3 \ldots a_{m-2}p_{m-2} \bmod 11 &= a_{m-1} \quad \text{and} \\ a_1p_1 + a_2p_2 + a_3p_3 \ldots a_{m-2}p_{m-2} + a_{m-1}p_{m-1} \bmod 11 &= a_m. \end{aligned}$$

In case the result is 10, then check digit is 0.

2.3 Other modular bases

Besides bases 10 and 11, which are the most used in practice, there are examples of the use of other modular bases in several countries. For instance, **IBAN** (International Bank Account Number), which is adopted in international bank transactions, use mod 97, [10]. In [7], the cases of an aviation company and of an American car rental are presented, which use mod 7.

3 ERROR DETECTION IN SYSTEMS BASED ON MODULAR ARITHMETIC

Considering a modular function σ_i base n applied to each digit of the identifier, whose image is contained in set $\{0, 1, 2, \ldots, n-1\}$, the error detection capability of a check digit modular system is reshaped by Theorem 1 which is partially presented, not as a theorem, in [15] and in [6].

Theorem 1. *A modular system of check digits is able to detect:*

1. *Single errors, the change of a with b if, and only if, the image of σ_i is a permutation of 0 to $n-1$ or p_i is relatively prime to n, for every i.*

2. *Transposition errors of digits a_i e a_j between positions i and j if, and only if,* $(a_i - a_j)(p_i - p_j) \not\equiv 0 \bmod n$ *or* $(\sigma_i - \sigma_j)(a_i) \not\equiv (\sigma_i - \sigma_j)(a_j) \bmod n$.

3. *Twin errors between a and b, in positions i and j if, and only if,* $(a - b)(p_i + p_j) \not\equiv 0 \bmod n$ *or* $(\sigma_i + \sigma_j)(a) \not\equiv (\sigma_i + \sigma_j)(b) \bmod n$.

4. *Phonetic errors in positions i and* $i + 1$ *if, and only if,* $p_i \not\equiv (p_i - p_{i+1})a \bmod n$ *or* $(\sigma_i - \sigma_{i+1})(a) \not\equiv \sigma_{i+1}(0) - \sigma_i(1) \bmod n$, *for* $a \neq 0, 1$.

Proof. In order for a check digit system to detect:

1. the change of a with b, $a \neq b$, it is sufficient that the image of σ_i be a permutation of digits from 0 to $n - 1$, as long as there are n digits and it is necessary that $\sigma_i(a) \not\equiv \sigma_i(b) \bmod n$. For the same reason it is necessary that $p_i a$, with a varying from 0 to 9, be a permutation mod n for each p_i. It is a well-known result in Modular Arithmetic that this occurs if and only if p_i is relatively prime to n.

2. the transposition of digits between positions i and j, besides $a_i \neq a_j$, we shall have

$$\sigma_i(a_i) + \sigma_j(a_j) \not\equiv \sigma_i(a_j) + \sigma_j(a_i) \bmod n.$$

   By the definition of the difference of functions, we may write,

$$(\sigma_i - \sigma_j)(a_i) \not\equiv (\sigma_i - \sigma_j)(a_j) \bmod n.$$

   Considering weights, the condition may be described as:

$$p_i a_i + p_j a_j \not\equiv p_i a_j + p_j a_i \bmod n.$$

   or, equivalently, $(a_i - a_j)(p_i - p_j) \not\equiv 0 \bmod n$.

3. twin errors from a to b in positions i and j, we shall have $a \neq b$ and

$$\sigma_i(a) + \sigma_j(a) \not\equiv \sigma_i(b) + \sigma_j(b) \bmod n.$$

   By the definition of sum of functions, we may write,

$$(\sigma_i + \sigma_j)(a) \not\equiv (\sigma_i + \sigma_j)(b) \bmod n.$$

   Considering weights, the condition may be re-written as:

$$p_i a + p_j a \not\equiv p_i b + p_j b \bmod n.$$

   or, equivalently, $(a - b)(p_i + p_j) \not\equiv 0 \bmod n$.

4. phonetic errors in positions i and $i+1$, we shall have $a \neq 0, 1$ and

$$\sigma_i(1) + \sigma_{i+1}(a) \not\equiv \sigma_i(a) + \sigma_{i+1}(0) \bmod n.$$

By the definition of sum of functions, we may write,

$$(\sigma_{i+1} - \sigma_i)(a) \not\equiv \sigma_{i+1}(0) - \sigma_i(1) \bmod n.$$

Considering weights, the condition may be re-written as:

$$p_i 1 + p_{i+1} a \not\equiv p_i a + p_{i+1} 0 \bmod n.$$

or, equivalently, $p_i \not\equiv (p_i - p_{i+1})a \bmod n$. □

3.1 Detection capability in current systems

We have conducted extensive practical experimentation in error detection of different systems used in Brazil. Table 2 shows the percentage of non-detected errors in these systems. Columns 2 to 7 (SNG, TRA, TRL, TWA, TWL, PHO, respectively) show the percentage of non-detected errors of each type according to Table 1. The last column, AVE, is a weighted average of non-detected errors, according to the weights of same table. For identifiers up to 9 digits, we have exhaustively considered all possible errors. For larger identifiers, such as CNPJ, we have conducted a large statistical sampling, considering samples of 10^9 values. Systems that use 2 check digits are represented only by the CNPJ, since it is the one that presents the best detection rate.

Table 2: Percentage of non-detected errors.

System	SNG	TRA	TRL	TWA	TWL	PHO	AVE (%)
10 - 2 weights	0	11.11	100.00	11.11	11.11	0	2.02
10 - 3 weights	0	11.11	11.11	49.23	40.73	0	1.59
10 alphanum. 3 weights	7.81	18.02	18.02	48.76	41.57	0	8.53
10 IBM	0	2.22	100.00	6.67	11.11	12.50	1.16
11 complete	0	0	0	14.29	0	6.25	0.10
11 restrict	1.82	1.82	1.82	15.85	1.82	14.61	1.80
11 CNPJ	0.04	0.14	0.14	0.83	0.03	2.36	0.06

It is observed that none of the systems detects all error types. The worst system observed was that of the passport, which uses mod 10 with 3 weights, and alphanumeric identifiers. The system that adopts restrict mod 11 is inferior to two of the mod 10 systems. By its turn, complete mod 11 system stands out among all systems with one check digit. However, it is not a totally satisfactory solution, as it is inadequate in situations that require numeric keyboards. The best of all methods is the CNPJ, which uses mod 11 with two check digits, as it would be expected.

We shall discuss below the conditions for optimality in check digit systems that use modular arithmetic. The discussion considers two cases: mod 10 and mod p, where p is a prime number, considering the case of mod 11, and generalizing it to other aforementioned situations.

4 OPTIMAL SYSTEMS FOR CHECK DIGITS WITH MODULAR ARITHMETIC

An optimal system is the one which fails to detect the lowest weighted percentage of errors. The equation for minimizing the weighted average of errors is given by:

$$\mu = 0,791\,p_1 + 0,102\,p_2 + 0,008\,p_3 + 0,005\,p_4 + 0,003\,p_5 + 0,005\,p_6.$$

Where p_i is the probability of non detection of the 6 first types of errors in Table 1, SNG, TRA, TRL, TWA, TWL and PHO, respectively.

As a reminder, a check digit system consists in applying k functions to the $m-1$ first digits of an identifier of digits, in order to obtain the last one of them. Given the high frequency of single errors, their detection is mandatory and, thus, these functions must be permutations, according Theorem 1.

4.1 Optimal system for prime modular bases

Next, we shall develop a general optimal system for prime modular bases. We are particularly interested in bases 11, which are appropriate for identifiers composed by decimal digits, and 37, for alphanumeric digits.

It has been observed that, when modular base n is a prime number, we may use systems of weights varying from 1 to $n-1$, as we then guarantee the detection of single errors. In order to detect transpositions, it suffices to have at least three distinct weights, repeatedly applied, since the difference between these weights will always be a number relatively prime to n. In order to consider twin errors, the sum of consecutive or alternate weights must not be equal to n. For the detection of phonetic errors, it is necessary that:

$$p_i \not\equiv (p_i - p_{i+1})a \bmod n, \quad \text{for } 0 \leq a \leq 9. \tag{1}$$

From modular arithmetic theory, we know that products pa, where p is an integer and a varies from 0 to $n-1$, generate a permutation of numbers from 0 to $n-1$ in modulo n. Hence, any number p_i from 0 to $n-1$ may be written as $p_i \equiv (p_i - p_{i+1})a \bmod n$, for some $0 \leq a \leq n-1$. Therefore, in order to satisfy the inequality 1, only numbers p_i remain, such that $p_i \equiv (p_i - p_{i+1})a \bmod n$, for values of a varying from 10 to $n-1$. We have thus determined a relation which p_i and p_{i+1} must satisfy.

In the case of mod 11, the inequality is fulfilled only for $a = 10$. Replacing this value in the equation, we have:

$$p_i \equiv 10(p_i - p_{i+1}) \bmod 11 \Longrightarrow p_{i+1} \equiv 2p_i \bmod 11.$$

Therefore, when choosing each weight as the double of its predecessor in mod 11, we meet the conditions for detecting phonetic errors. That is, by alternately applying any circular permutation of weights $\langle 1, 2, 4, 8, 5, 10, 9, 7, 3, 6\rangle$.

For $n > 11$, there are several other possible relations between two consecutive weights. Therefore, we shall use a different framework in order to generate these various combinations. We shall create a digraph with $n - 1$ vertices, where there is an edge between vertices u and v, $1 \leq u \neq v < n$, if weights u and v, applied to two consecutive positions of the identifier, are able to detect all phonetic errors. Note that the relation is not cumulative, the pair (u, v) may be able to detect all phonetic errors, while the pair (v, u) may not have this property. In creating this digraph, in order to avoid twin errors, we have not considered pairs where $u + v = n$.

After creating the digraph, we look for cycles in it, of the desired size, such that, for every two alternate elements of the cycle, their sum is never n.

The digraph for $n = 13$ is shown in Figure 1. We have highlighted, as examples, the cycle of size 3: $\langle 1, 2, 4 \rangle$; and the cycle of size 12: $\langle 1, 2, 3, 4, 6, 8, 12, 11, 9, 5, 10, 7 \rangle$.

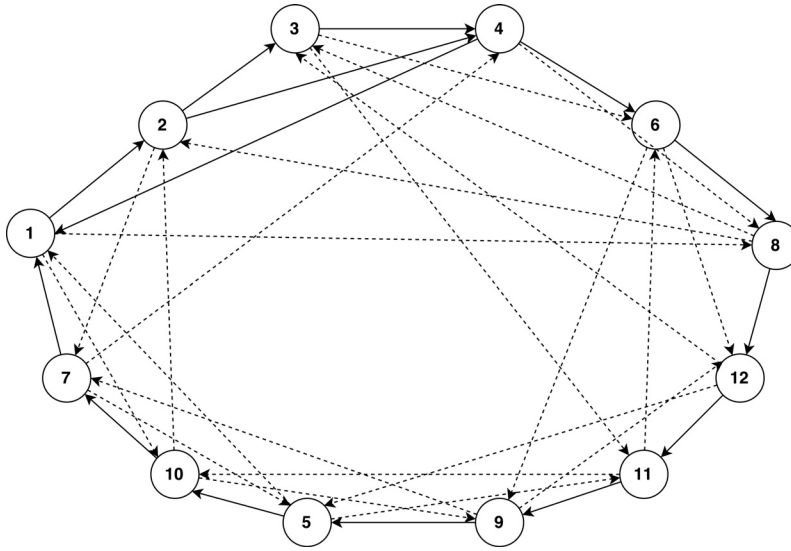

Figure 1: Digraph for $n = 13$.

In order to obtain these cycles, we could use the Dijkstra algorithm. However, as we have the special test for alternate twins, and also as, in general, we look for cycles of small size, we may use the backtracking algorithm shown in the sequence. In this algorithm, L is the set of weights aimed at, and $k > 2$ the size of the desired cycle. T is an auxiliary vector to indicate elements which have already entered the cycle. The variable *can* is true when it is possible to add a new element to the cycle. For that, it should be checked if the new element to be added has not already entered the cycle, if the sum of alternate elements is different from n and if the cycle appropriately closes when the kth element is placed in the cycle.

Table 3 shows some of the algorithm's outputs for four prime bases: 11, 13, 37 and 97 and values for k: 3, 6 and 10.

Algorithm 1:

```
Config();
for i from 1 to n − 1 :
    can ← true;
    if (not T[i]) then can ← false;
    else if (l > 0) and (E[L[l], i] = 0) then can ← false;
    else if (l > 1) and ((L[l − 1] + i) = n) then can ← false;
    else if (l = (k − 1)) and ((E(i, L[1]) = 0) or ((L[l] + L[1]) = n)) then
        can ← false;
    if (can) then
        l ← l + 1;   L[l] ← i;   T[i] ← false;
        if (l = k) then Print;
        else Config();
        l ← l − 1;   T[i] ← true;
Begin
    l ← 0;   T[∗] ← true;   Config();
```

Table 3: Examples of cycles with k weights for prime bases n.

n \ k	3	6	10
11	–	–	1 2 4 8 5 10 9 7 3 6
13	1 2 4	1 2 3 6 9 5	1 2 3 4 6 8 12 11 9 5
	3 4 8	1 8 3 6 12 5	1 2 3 4 6 12 5 10 9 7
	5 10 7	2 3 6 9 5 10	1 2 3 4 6 12 11 9 7 5
37	1 2 3	1 2 3 4 5 6	1 2 3 4 5 6 7 8 9 10
	1 2 8	1 2 3 4 5 8	1 2 3 4 5 6 7 8 9 11
	1 2 12	1 2 3 4 5 12	1 2 3 4 5 6 7 8 9 12
97	1 2 3	1 2 3 4 5 6	1 2 3 4 5 6 7 8 9 10
	1 2 5	1 2 3 4 5 9	1 2 3 4 5 6 7 8 9 12
	1 2 12	1 2 3 4 5 14	1 2 3 4 5 6 7 8 9 14

4.1.1 Optimal system for mod 11, with one check digit

It should be noted that currently found problems in the use of mod 11 are mainly due to an inadequate sequence of weights in the system for calculating CDs. In the case of complete mod 11, it suffices to use the sequence of weights ⟨1, 2, 4, 8, 5, 10, 9, 7, 3, 6, ⟩ and we obtain detection of all six types of errors considered in this paper.

In the case of restrict mod 11, non detected errors correspond to 1,82% of cases of each type of error, due to the situations where the calculation initially indicates 10 and, with the error, turns to 0, or vice-versa, that is, 2/110 of cases of error.

4.1.2 Optimal system for alphanumeric identifiers with one check digit

In this case, we can use the general system presented, with modular base 37. The check digit must also be alphanumeric. As we have 36 alphanumeric digits in the Latin alphabet, a natural solution is to consider an additional character. In the case of passports, this character corresponds to <. Besides, non-numeric characters, that is, those belonging to set $\{A, B, \ldots, Z, <\}$ would be mapped to the numeric set $\{10, 11, 12, \ldots, 36\}$. The simplest set of weights to be used consists of 3 weights $\langle 1, 2, 3\rangle$ used repeatedly.

This solution is associated with a significant improvement when compared to the system currently used for passports.

4.1.3 Optimal systems for mod 11 with two check digits

An optimal system consists in using the general system, as described in the case with one digit. Also, it is possible to conduct only one calculation and the result should be represented with two digits. That is, results from 0 to 9 would be represented with 0 in the first check digit and the result itself in the second. The result 10 would normally be represented with two digits.

This solution is much simpler and more efficient than the several existing cases with two check digits, since it detects all error situations.

Its advantage in relation to mod 11 with only one check digit is the fact that it avoids non-numeric character X, always dealing with decimal digits only. However, it has the disadvantage of using an additional digit in each identifier.

4.2 Optimal systems for mod 10 check digit

All mod 10 systems shown, which are effectively used in practice, adopt the strategy of modular multiplication by weights. This result corresponds to the use of permutation, as long as the weights are relatively prime to 10. These weights are 1, 3, 7 and 9, and the equivalent permutations are σ_1 : (0 1 2 3 4 5 6 7 8 9), σ_3 : (0 3 6 9 2 5 8 1 4 7), σ_7 : (0 7 4 1 8 5 2 9 6 3) and σ_9 : (0 9 8 7 6 5 4 3 2 1), respectively. It is easily observed that none of the pairs in these permutations satisfies the equations in Theorem 1 of Section 3. This explains the percentages of non-detected errors shown in Table 2. Aggravating the difficulties with mod 10 systems, we have the following theorem, proven in [8].

Theorem 2. *Suppose a check digit system mod n, where n is even, which detects all single errors. Then there is at least one change of adjacent digits that this system does not detect.*

Proof. Since the system must detect all single errors, $\sigma_i(a)$ is a permutation, for every i. In order to detect all adjacent transposition errors, one should have

$$\sigma_i(a_i) + \sigma_{i+1}(a_{i+1}) \not\equiv \sigma_i(a_{i+1}) + \sigma_{i+1}(a_i) \bmod n, \quad \text{for all } a_i \neq a_{i+1}.$$

Therefore, the difference $\sigma(a) = \sigma_i(a_i) - \sigma_{i+1}(a_i)$ should also be a permutation. However, the sum of elements in $\sigma_i(a)$ in $\mathbb{Z}_n$ is given by:

$$\sum_{i=0}^{n} i = \frac{n}{2} + (1 + n - 1) + (2 + n - 2) + (3 + n - 3) + \ldots + \left(\frac{n}{2} - 1 + \frac{n}{2} + 1\right) \equiv \frac{n}{2} \bmod n.$$

This implies that:

$$\frac{n}{2} \equiv \sum \sigma(a) \equiv \sum(\sigma_i(a_i) - \sigma_{i+1}(a_i)) = \sum \sigma_i(a_i) - \sum \sigma_{i+1}(a_i) \equiv \frac{n}{2} - \frac{n}{2} = 0 \bmod n.$$

According to the last equality, $\frac{n}{2} \equiv 0 \bmod n$, this is a contradiction. □

The considerations above show that there is no way to improve mod 10 systems used in practice. New methods should be considered. The search for optimal mod 10 systems has led to the strategy of searching for more adequate permutations than the ones shown above. We shall present two mod 10 systems which are optimal, given certain restrictions. The first one is Verhoeff system [15], described below, and the second is a new optimal system developed by the authors, when the use of only three permutations is desired.

We shall explore in detail the properties of the intended permutations. We first consider the calculation of non-detected transposition errors for two permutations p and q applied to digits a_i and a_j of an identifier. A transposition error is not detected if $p(a_i) + q(a_j) \equiv p(a_j) + q(a_i) \bmod 10$ or, in an equivalent manner, $p(a_i) - q(a_i) \equiv p(a_j) - q(a_j) \bmod 10$. Each equality in the equation corresponds to 2 non-detected errors, the change of a_i with a_j and vice-versa.

Let us illustrate with permutations p : (1 4 2 5 0 9 3 6 8 7) and q : (3 7 0 8 5 1 4 6 2 9). In order to find the cases of non-detected errors, the mod 10 difference between these permutations should be calculated: $d = p - q$: (8 7 2 7 5 8 9 0 6 8). This difference shows that the transpositions between numbers 1 and 3, in the positions where p and q were applied, are not detected, since $p(1) - q(1) = 4 - 7 = -3 \equiv p(3) - q(3) = 5 - 8 \bmod 10$, that is, the same result is obtained.

This shows that, in order to calculate all non-detected transposition errors, we shall count the amount s of each result in the difference function, and, for each distinct s, add the number of arrangements $A_{s,2} = s(s-1)$ to the total. In the example, there are two repetitions of number 7 and three of number 8. The total of transposition errors non-detected by the pair of permutations considered is, thus, $t = 2.1 + 3.2 = 8$. The 8 pairs of non-detected changes are (1, 3), (3, 1), (0, 5), (5, 0), (0, 9), (9, 0), (5, 9) and (9, 5).

In order to calculate non-detected twin errors, rather than subtracting the permutations involved, they should be added, since non-detection of the change of a digit a in positions where p and q are applied, by another digit $b \neq a$, occurs if $p(a) + q(a) \equiv p(b) + q(b) \bmod 10$.

Finally, a phonetic error, which occurs in consecutive positions where p and q are applied, is not detected when $p(1) + q(a) \equiv p(a) + q(0) \bmod 10$, for $a = 2, 3, \ldots, 9$. Similarly, when we have $p(1) - q(0) \equiv p(a) - q(a) \bmod 10$. Whenever equality occurs, 2 errors fail to be detected, the change $1a$ by $a0$ and vice-versa.

4.2.1 Verhoeff Mod 10 system

The best mod 10 check digit system known was, until recently, the one proposed by Verhoeff. He developed a system with the following characteristics:

1. it uses $m - 1$ distinct permutations, one for each digit in the identifier with m digits.
2. each pair of consecutive permutations fails to detect the least number of twin errors and transposition errors (2 cases in 90 of each type).
3. each pair of alternate permutations fails to detect 4 errors in 90 possible ones, for both twin and transposition errors.
4. only 2 in 16 phonetic errors fail to be detected for every set of 6 permutations used.

Due to computational limitations of the time of its development, this system was found through a mixed process where several mathematical properties of the system have been deduced and many results have been achieved computationally. Verhoeff has not proven the method's optimality. As shown below, this method is optimal only for identifiers up to 6 digits.

We present Algorithm 2 which finds a system equivalent to Verhoeff in the search for a set of permutations with the properties described.

Algorithm 2:

```
Config()
for i from 1 to s :
    q ← MapPerm(S[i], L[l]);   e ← ErrorsPhon(q, L[l]);
    if (l mod 6 = 0) then e ← e − 2;
    if (l > 1) then
        e ← e + ErrorsTwin(q, L[l − 1]) + ErrorsTrans(q, L[l − 1])
    else e ← e + 8;
    if (e = 8) then
        l ← l + 1;   L[l] ← q;
        if (l = m) then
            Print
        else Config();
        l ← l − 1;
Begin
    Obtain S;   Define m;   L[1] ← r;   l ← 1;   Config();
```

In Algorithm 2, s and l respectively refer to the sizes of S, the set of all permutations that minimize non-detection of both transpositions and twin errors in relation to the identity permutation, r : (0 1 2 3 4 5 6 7 8 9), and L, the set of permutations to be used in the system.

Procedure $Config$ successively fulfills set L, where, after each group of 6 permutations, 2 phonetic errors are admitted. Function $MapPerm$ conducts the mapping of one permutation of S, such that a new permutation has the desired properties in relation to the permutation already placed in L. Variable e counts the cases of non-detected phonetic and alternate errors. When

$e = 8$ the permutation is accepted in set L, according to the rules of the system. For the second permutation this test is not necessary, so 8 is artificially added to e to force the permutation to be accepted.

Table 4 shows an example of a set obtained by Verhoeff for $m = 18$.

Table 4: Set with 18 permutations.

1	0123456789	7	3456710298	13	6752498310
2	9460571382	8	8359472106	14	8310795264
3	3620845197	9	0397568421	15	2931568704
4	1290857436	10	0189723564	16	6582741903
5	8340971265	11	0562184739	17	0541823976
6	8142630579	12	4175908236	18	4189053627

Even though Verhoeff's system has been the best mod 10 system since 1969, we have not found any reference to its practical application. We suppose this is due to a psychological barrier that considers the system's implementation complicated and subject to errors, as it involves a large number of permutations.

4.2.2 The search for an optimal mod 10 system with 3 permutations

As shown, an optimal base 10 system should not be based on weights, but rather on the use of at least three permutations.

Until 2013 there have been several attempts to find such a system, all of them are described in [5], but none of them has achieved the optimal method.

An initial improvement attempted to resolve the problem of mod 10 IBM, by using powers of permutation σ : (0 2 4 6 8 1 3 5 7 9). The equation of the system, in this case, is given by:

$$\sum_{i=1}^{m-1} \sigma^{i-1}(a_{m-i}) \equiv a_m \bmod 10.$$

In another variant, called Generalized IBM code, the permutation σ is modified to $\mu(a) = \sigma(a) + 6 \bmod 10$, where $a = 0, 1, \ldots, 9$, that is, μ : (6 8 0 2 4 7 9 1 3 5). The detection of single, transposition and twin errors is the same as in mod IBM, but this method is better for other types of error.

Two variations have also been tried, called the Colenbrander code, which differ only in the permutation used in the system. The equation for calculating the check digit is once again as the one for generalized Mod 10 IBM, only changing the permutation. The Colenbrander method uses the permutation f : (9 2 4 6 8 0 1 3 5 7). In the Modified Colenbrander, permutation is λ : (1 3 5 7 9 0 2 4 6 8). Permutation λ is obtained from f through the equation $\lambda = (f((a + 1) \bmod 10) - 1) \bmod 11$.

Another method of this kind has been created by a German bank. The following three permutations are applied alternately to the digits of the identifier: σ_1 : (1 2 3 4 5 6 7 8 9 0), σ_2 : (2 4 6 8 0 1 3 5 7 9), and σ_3 : (3 6 9 1 4 7 0 2 5 8). These permutations are reached through the formula $\sigma_i(a) = ((ia + i) \bmod 11) \bmod 10$.

An optimal code was finally obtained in 2013 by the authors [12] and is described below.

4.2.3 Optimal system for mod 10 check digit with 3 permutations

In the optimal base 10 modular system with three permutations, these permutations are alternately applied to the digits of the identifier. We shall consider, without loss in generality, the first of these permutations as identity: σ_1 : (0 1 2 3 4 5 6 7 8 9).

Next, we show the stages in constructing this system and prove its optimality. Initially, the error detection capability of one of these systems is analyzed, formed by permutations σ_1 : (0 1 2 3 4 5 6 7 8 9), σ_2 : (0 8 6 4 2 7 9 1 3 5) and σ_3 : (1 6 3 2 8 7 4 0 5 9).

Lemma 3. *The mod 10 check digit system that uses three permutations* σ_1 : (0 1 2 3 4 5 6 7 8 9), σ_2 : (0 8 6 4 2 7 9 1 3 5) *and* σ_3 : (1 6 3 2 8 7 4 0 5 9) *fails to detect:*

1. *0 single errors;*
2. *2/90 of transposition errors;*
3. *16/270 of twin errors;*
4. *0 phonetic errors.*

This way, the rate of non detection of errors is

$$\mu = 0,102(2/90) + 0,008(2/90) + 0,005(16/270) + 0,003(16/270) = 0,29\%.$$

Proof. The system detects all single errors, as it uses permutations. A check digit system does not detect transposition errors in positions i and j when $(\sigma_i - \sigma_j)(a_i) \equiv (\sigma_i - \sigma_j)(a_j) \bmod 10$; twin errors in positions i and j when $(\sigma_i + \sigma_j)(a) \equiv (\sigma_i + \sigma_j)(b) \bmod 10$ and phonetic errors when $\sigma_i(1) - \sigma_{i+1}(0) \equiv (\sigma_i - \sigma_{i+1})(a) \bmod 10$, for $a \neq 0, 1$. Table 5 shows the cases where equalities are satisfied for each type of error.

It should be noted that, in order to analyze adjacent errors, permutations should be linked as follows: σ_1 with σ_2, σ_2 with σ_3 and σ_3 with σ_1. For alternate errors, we link σ_1 with σ_3, σ_2 with σ_1 and σ_3 with σ_2. In the general case of transposition and twin errors, it suffices to analyze combinations two by two of the three permutations.

Lines 1 to 3 in Table 5 show cases of non-detection of transposition errors. Every pair of values whose images of $\sigma_i - \sigma_j$ are equal will not be detected. Therefore, 2 ($= A_{2,2}$) transpositions in 90 possible ones will be undetected.

Table 5: Positions of non-detected errors

1	$\sigma_1 - \sigma_2$	0 3 6 9 2 8 7 6 5 4
2	$\sigma_2 - \sigma_3$	9 2 3 2 4 0 5 1 8 6
3	$\sigma_3 - \sigma_1$	1 5 1 9 4 2 8 3 7 0
4	$\sigma_1 + \sigma_2$	0 9 8 7 6 2 5 8 1 4
5	$\sigma_2 + \sigma_3$	1 4 9 6 0 4 3 1 8 4
6	$\sigma_1 + \sigma_3$	1 7 5 5 2 2 0 7 3 8
7	$\sigma_1(1) - \sigma_2(0)$	1
8	$\sigma_2(1) - \sigma_3(0)$	7
9	$\sigma_3(1) - \sigma_1(0)$	6

Lines 4 and 5 show cases of non-detection of twin errors in general. In line 4, there are 2 ($= A_{2,2}$) cases of non-detection. In line 5, 2 ($= A_{2,2}$) errors correspond to the change between elements whose images of $\sigma_2 + \sigma_3$ are equal to 1, and 6 ($= A_{3,2}$) errors correspond to the change between elements whose images are equal to 4, thus adding to a total of 8 non-detected errors.

In line 6, to the change between elements whose images $\sigma_1 + \sigma_3$ are equal to 2, 5 and 7, there is a correspondence of 3 ($= A_{2,2}$) errors.

Lines 7 to 9 show that no phonetic error fails to be detected, since $\sigma_i(1) - \sigma_{i+1}(0) \not\equiv (\sigma_i - \sigma_{i+1})(a) \bmod 10$, for $0 \leq a \leq 9$. This equation may be generalized for any value of a, since in cases of $a = 0$ or 1, the equation corresponds to single errors □

Lemma 4. *An optimal mod 10 check digit system with three permutations fails to detect 2/90 of transposition errors.*

Proof. Suppose an optimal system where all phonetic errors are detected and all pairs of permutations fail to detect 2/90 of twin errors (the best possible rate), two of 3 pairs fail to detect 2/90 of transposition errors and only one pair fails to detect 4/90 (the following lowest value possible). Hence, the percentage of non-detection $\mu = 0,102(6/270) + 0,008(6/270) + 0,005(2/90) + 0,003(2/90) = 0,0034$ is superior to the Lemma 3 system. This is a contradiction. □

Corollary 5. *An optimal mod 10 check digit system should minimize non detection of twin and phonetic errors.*

The optimal system was obtained through an exhaustive computational process, though with several optimizations. Being r the identity permutation, we have created vector S with permutations p such that $ErrorsTra(r, p) = 2$, where $ErrosTra$ indicates transposition errors. Thus, non-detection of transposition errors is minimized. Experimentally, it has been verified that S has 46.400 permutations. All triples $\{r, p, q\}$ have been examined, where $p, q \in S$, such that $ErrorsTrans(p, q) = 2$. For each value k, vector CG counts for each $0 \leq k \leq 90$, the total

amount of twin errors not detected by the permutation pairs of each triple. It was experimentally obtained that the total of triples where all pairs of permutations fail to detect 2 transposition errors is 12.654.000. The lowest rate not zeroed of vector CG is for $k = 16$, which is thus the minimum number of non-detected twin errors. The minimum of 16 errors is relative to the total of 270 errors, since we have three distinct pairs in each triple and the number of errors refer to the three permutations as a whole.

The minimization of phonetic errors has been treated independently of that of twin errors, since there are triples capable of detecting all phonetic errors, as is shown by the result of Algorithm 3 below.

Algorithm 3:

$r \leftarrow (0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)$; $p \leftarrow r$; $m \leftarrow 0$;
Begin
  for i *from 2 to 10!* **:**
    $p \leftarrow Nextperm(p)$;
    if $(ErrorsTrans(r, p) = 2)$ **then** $m \leftarrow m + 1$; $S[m] \leftarrow p$;
  for i *from 1 to* $m - 1$ **:**
    for j *from* $i + 1$ *to* m **:**
      if $(ErrorsTrans(S[i], S[j]) = 2)$ **then**
        $k \leftarrow ErrorsTwin(r, S[i]) + ErrorsTwin(S[i], S[j]) + ErrorsTwin(S[j], r)$;
        $f \leftarrow ErrorsPhon(r, S[i]) + ErrorsPhon(S[i], S[j]) + ErrorsPhon(S[j], r)$;
        if $((k = 16)$ *and* $(f = 0))$ **then Print** $r, S[i], S[j]$;

Algorithm 3 obtains 100 triples with the desired characteristics. It may be experimentally observed that the distribution of non-detected twin errors between pairs of permutations is only of two kinds: $\langle 2, 8, 6\rangle$ e $\langle 6, 8, 2\rangle$. Triples of the first kind are preferred, since they minimize non-detection for identifiers of all sizes. Based on this observation, it is possible to modify Algorithm 3 such that non-detection of twin errors for the first pair of each triple is equal to 2. This leaves a total of 48 optimal triples. Table 6 shows the last two permutations of the 48 triples, which has the identity as initial permutation, obtained with the use of this algorithm.

Lemma 6. *An optimal mod 10 check digit system with three permutations should detect the same errors as the Lemma 3 system.*

This result is experimentally proved by Algorithm 3.

4.2.4 Optimality of mod 10 methods

When Verhoeff proposed the system described above, he did not prove its optimality. His system was obtained through a mixed process of theory and experimentation. The present authors have recently developed a program which exhaustively searches for sets of up to 6 permutations that are better, in terms of error detection, than Verhoeff system. The program has not found such sets, thus proving the optimality of Verhoeff system for identifiers of up to 6 digits. Over 6 digits,

Table 6: Mod 10 optimal systems with three permutations

1	0864279135 1632874059	25	2086491357 9410652837
2	0864279135 2743985160	26	3197502468 3854096271
3	0864279135 3854096271	27	3197502468 4965107382
4	0864279135 4965107382	28	3197502468 5076218493
5	0864279135 5076218493	29	3197502468 6187329504
6	0864279135 6187329504	30	3197502468 7298430615
7	0864279135 7298430615	31	3197502468 8309541726
8	0864279135 8309541726	32	3197502468 9410652837
9	0864279135 9410652837	33	4208613579 4965107382
10	1975380246 2743985160	34	4208613579 5076218493
11	1975380246 3854096271	35	4208613579 6187329504
12	1975380246 4965107382	36	4208613579 7298430615
13	1975380246 5076218493	37	4208613579 8309541726
14	1975380246 6187329504	38	4208613579 9410652837
15	1975380246 7298430615	39	5319724680 6187329504
16	1975380246 8309541726	40	5319724680 7298430615
17	1975380246 9410652837	41	5319724680 8309541726
18	2086491357 2743985160	42	5319724680 9410652837
19	2086491357 3854096271	43	6420835791 7298430615
20	2086491357 4965107382	44	6420835791 8309541726
21	2086491357 5076218493	45	6420835791 9410652837
22	2086491357 6187329504	46	7531946802 8309541726
23	2086491357 7298430615	47	7531946802 9410652837
24	2086491357 8309541726	48	8642057913 9410652837

the system of three permutations described becomes better than Verhoeff and it is conjectured that it is not only an optimal system with three permutations, but that it is also optimal in general, with identifiers of 7 or more digits.

Next, we sum up the error detection capability of optimal methods.

4.3 Error detection capability of optimal methods

Table 7 shows the percentage of non-detected errors in several optimal methods. For the case of base 10 Verhoeff's method, two lines are presented, one for identifiers of up to 6 digits, and another for identifiers of more than 6 digits. The case of restrict mod 11 is also illustrated.

It is observed that methods with prime modular base may detect all errors considered in this paper, with the exception of restrict Modulus 11.

A comparison with Table 2, Section 3, shows the margin for possible improvement in check digit error detection, in relation to current applications. The case of passports is highlighted, where one could have benefited from the fact that it is a very adequate situation to use prime base 37, exactly the number of symbols used in this document.

Table 7: Percentage of non-detected errors for optimal methods

Systems	SNG	TRA	TRL	TWA	TWL	PHO	AVE %
Mod 10 3P optimal	0	2.22	2.22	5.93	5.93	0	0.29
Mod 10 Vehroeff (≤ 6)	0	2.22	4.44	2.22	4.44	0	0.28
Mod 10 Vehroeff (≥ 7)	0	2.22	4.44	2.22	4.44	3.57	0.30
Mod 11 restrict	1.82	1.82	1.82	1.82	1.82	1.82	1.66
Prime modular bases	0	0	0	0	0	0	0

5 CONCLUSION

In this work, we have presented and discussed check digit systems based on modular arithmetic, which are used in almost all computational records of modern life. In particular, we have examined their conditions of optimality, which is a theme that has not been treated in the literature of the field yet.

We have developed an optimal method for prime modular bases which is capable of detecting all main error types considered in this work. Recent works based in group theory, such as [3] have been proposed for the same purpose. Nevertheless, they have not arrived yet to optimal methods, since they can not detect all phonetic errors.

For modular base 10 systems, we have characterized the fact that it is imperfect to use weights in the calculations. We have presented two optimal systems, given certain restrictions. The first one is an old system, developed by Verhoeff, which we have proved, through an extensive computational experiment, to be optimal for identifiers of up to 6 digits. We have described a new optimal system, when only three permutations are used, developed by the present authors which is conjectured to be also optimal for identifiers of more than 6 digits. It should be noted that the Verhoeff system for base 10, based on group theory, is superior to both modular arithmetic systems discussed, but is not used in practice anymore.

In recent years a new base has been considered and is related to identifiers using hexadecimal digits. Examples are International Standard Audiovisual Number (ISAN) which enables the identification of any kind of audiovisual works and the International Mobile Equipment Identifier (MEID) which is unique for each mobile station. In both cases the check digit is calculated transforming base 16 to base 10. A more adequate method was created by [3] using group theory, as mentioned before.

We have verified that all check digit systems used in Brazil can be improved with the proposals presented in this work, considering the six most common error types in data input and their probability of occurrence.

For the reader's convenience, four programs which implement the four main algorithms presented in this work are available at: `http://www.ime.uerj.br/$\sim$pauloedp/MEST/TESES/NATALIA/Pagina.html`,

1. ALG8.pas: implementation of Algorithm 1 to obtain mod 10 systems equivalent to Verhoeff.
2. ALG9.pas: implementation of Algorithm 2 to obtain optimal mod 10 systems with three permutations.
3. ALG11.pas: implementation of Algorithm 3 to obtain optimal modular systems of prime bases.
4. ALG12.cpp: Verification of Verhoeff's system optimality.

RESUMO. Neste artigo discutimos os sistemas de dígitos verificadores baseados em aritmética modular, utilizados mundialmente. Dígitos verificadores são usados para eliminar a maioria dos erros na entrada de dados em sistemas computacionais. Embora antigos, não é encontrada na literatura uma discussão sobre a otimalidade dos esquemas utilizados. Descrevemos os principais esquemas existentes e destacamos aqueles adotados no Brasil. Apresentamos as melhorias necessárias para tornar os diversos esquemas ótimos. Propomos, também, um novo esquema ótimo com 3 permutações para sistemas com base modular 10.

Palavras-chave: Dígitos verificadores, detecção de erros, aritmética modular.

REFERENCES

[1] D. Argenta & R. Amorim. *Estudo e Implementação de Dígitos Verificadores*. 2012. 52 f. Monograph (Computer Science). Universidade do Estado do Rio de Janeiro, (2012).

[2] G.B. Belyavskaya, V.I. Isbash & G.L. Mullen. Check Character Systems Using Quasigroups: I. *Des. Codes Cryptography*, **37**(2) (2004), 215–227.

[3] Y. Chen, M. Niemenmaa & AJ. H. Vinck. A general check digit system based on finite groups – *Designs, Codes and Cryptography*, **80**(1) (2016), 149–163.

[4] H.M. Damm. Totally anti-symmetric quasigroups for all orders $n \neq 2, 6$. *Discrete mathematics*, **307**(6) (2007), 715–729.

[5] A. Ecker & G. Poch. Check Character Systems. *Computing*, **37**(4) (1984), 277–301.

[6] J.A. Gallian. Error detection methods. *ACM Computing Surveys (CSUR)*, **28**(3) (1996), 504–517.

[7] J.A. Gallian. *Contemporary Abstract Algebra*, 70. ed., Cengage Learning, (2010).

[8] P. Gumm. A new class of check digit methods for arbitrary number systems. *Information Theory, IEEE Transactions on* **31**(1) (1985), 102–105.

[9] H.P. Luhn. *Computer for verifying numbers*. US n. 2.950.048, 6 January 1954, 23 August (1960).

[10] IBAN, Available in: `http://www.tbg5-finance.org/?ibancheck.shtml`. Last accessed 10 october 2016.

[11] M. Malek. *Coding Theory*, Decimal Code. Available in: `http://www.mcs.csueastbay.edu/~malek/TeX/Decimal.pdf`, 2009. Last accessed 10 october 2016.

[12] N. Pedroza. *Uma análise dos esquemas de dígitos verificadores usados no Brasil*. 84f. Master's thesis. Universidade do Estado do Rio de Janeiro, (2013).

[13] R.-H. Schulz. On Check Digit Systems using Antisymmetric Mappings. *Mathematik und Informatik, Freie Universät Berlin*, (1999).

[14] R.-H. Schulz. Check Character Systems and Anti-symmetric Mappings. *Computational Discrete Mathematics, Advanced Lectures*, (2001), 136–147.

[15] J. Verhoeff. *Error Detecting Decimal Codes*. 122 f. Tese, Mathematical Centre Tract 29, The Mathematical Centre, (1969).

[16] E.F. Wood. Self-Checking Codes – An Application of Modular Arithmetic. *The Mathematics Teacher*, **80**(4) (1987), 312–316.