Radomir S. Stanković,[*] Jaakko Astola,[**] Claudio Moraga[***]

PASCAL MATRICES, REED–MULLER EXPRESSIONS AND REED–MULLER ERROR CORRECTING CODES

Abstract. Reed–Muller expressions are a way to analytically describe binary sequences viewed as truth-vectors of switching functions. Reed–Muller codes are a mean to improve reliability in transmitting binary sequences. Both concepts can be defined in terms of Reed–Muller matrices. These matrices can be viewed as a Pascal matrix (matrix of binomial coefficients) computed modulo 2. The Pascal matrix is a key concept starting from which these two concepts, the Reed–Muller expressions and the Reed–Muller codes, can be developed. Further, the Pascal matrices give a way towards two different, but equally possible, interpretations of the Reed–Muller expressions. Their elements are coefficients in polynomial expressions, while their columns can be viewed as basis functions in particular spectral representations. Particular rows of the Pascal matrix modulo 2 are selected to form the generator matrix of the Reed–Muller code. The link to Pascal matrices permits generalizations of the binary Reed–Muller codes into p-ary Reed–Muller codes. This link can be used conversely to define new functional expressions for p-valued functions.

Mathematics Subject Classification (2010): Primary: 15-02, 43A32; Secondary 94B05.

Keywords: Reed-Muler expressions, Reed-Muler codes, Pascal matrices, spectral transforms

*Dept. of Computer Science, Faculty of Electronic Engineering, Niš, Serbia
Radomir.Stankovic@gmail.com

**Deptartment of Signal Processing, Tampere University of Technology, Tampere, Finland
Jaakko.Astola@tut.fi

***European Center for Soft Computing, Mieres, Spain
TU Dortmund University, Dortmund, Germany
Claudio.Moraga@udo.edu

Contents

1. Introduction

Reed–Muller expressions are usually related to the work of Muller [39, 40] and Reed [45], although originated in the work of Zhegalkin in 1927 and 1928 [72, 73]. This is possibly due to their concrete engineering applications suggested in [39] and [45] and partially because of the language barrier, since Zhegalkin published in Russian although with extended few pages abstracts in French in both of these publications. The widespread usage of Reed–Muller expressions in Switching theory and Logic design is related to the observation that AND-EXOR expressions on the average require a smaller number of product terms than AND-OR expressions [47].

The Reed–Muller expressions can be alternatively viewed as particular spectral representations of switching functions [4, 7, 8]. In these settings, these expressions are determined by using the Reed–Muller matrices viewed as matrices whose columns are defined by Boolean monomials. The spectral interpretation originates in a search for fast computing methods for coefficients in the Reed–Muller expressions. The Fast Fourier Transform (FFT) like algorithms have been devised for computing the coefficients in the Reed–Muller expressions, see [7] for the binary case and [23] for the ternary case. Generalizations of these methods to other multi-valued cases are straightforward.

In the binary case, the elements of the Reed–Muller matrices are, therefore, binomial coefficients computed modulo 2. Binomial coefficients are elements of the Pascal matrix, and then the Reed–Muller matrix is the Pascal matrix with elements computed modulo 2. In the literature, the Reed–Muller matrix is reported under different names depending on the area of application or aimed at pointing out some of its features. For example, it is called the conjunctive transform matrix [4] since its columns can be expressed in terms of the logic AND (conjunction) of Boolean variables. If elements of the Pascal matrix are written in the form of a triangle, the term Pascal triangle is often used. The Pascal triangle modulo 2, is also called the

Sierpiński triangle, Sierpiński gasket, and Sierpiński sieve, referring to the work by Waclaw Sierpiński in 1915 and 1916 [50, 51, 52]. For historical reasons, it should be noticed that the same patterns can be found in certain mosaics from the 13th century in two churches in Italy [1] [70]. In coding theory and cryptography, the term Algebraic Normal Form (ANF) is used to denote the Reed–Muller expressions with all variables in positive polarity, i.e., with variables represented by positive literals.

The same references of Muller [39] and Reed [45] are usually mentioned when discussing the Reed–Muller codes invented by Muller, while Reed provided an efficient decoding algorithm in terms of the majority logic. The Reed–Muller codes are defined in terms of Boolean monomials, or worded differently, by referring to rows of the Reed–Muller matrices.

Therefore, both concepts, the Reed–Muller expressions and Reed–Muller codes, are related to the Pascal matrix. This observation permits a unified view to these different but related concepts, which opens a way for further generalizations and extensions of these concepts to the p-valued case.

When discussing different approaches to the computation of Reed–Muller expressions in the context presented in this paper, it is interesting to mention algorithms using the transeunt triangles introduced in [64] and explored and elaborated later by several authors, see, for instance, [9, 10, 16, 38]. A discussion of relationships between the Sierpiński triangle and the transeunt triangle is presented in [63].

2. Pascal Matrices

The Pascal matrix is an infinite matrix $\mathbf{P} = [p_{i,j}]$ whose elements are binomial coefficients arranged as a two-dimensional array.

Binomial coefficients are the coefficients (positive integers) appearing in the algebraic expansion of non-negative integer powers of a binomial $(x + y)^n$. This expansion consists of terms $qx^r y^s$, with $r + s = n$. The power n determines the row of the Pascal matrix where the corresponding binomial coefficients q are located. For more discussions about the Pascal matrices, we refer to [11]. The history of the Pascal triangle is very interesting as can be seen from [69] and dates back to the 2nd century BC, and the work of the Indian scholar Pingala, as it was documented and used by Halayudha around 957, and latter by Bhattotpala in 1068. There we can read in the voluminous literature on the subject that the Pascal triangle was used by Persian scholars Al-Karaji and Omar Khayám in the 11th century. The notion was known also to the Chinese mathematician Yang Hui in the same century. The notion spread to Europe by the work of the German scholar Petrus Apianus in the 16th century. The triangle was related to Pascal by Pierre Raymond de Monmort who reported it as *Table de M. Pascal pour les combinaisons*, and Abraham de Moivre reported in Latin as *Triangulum Arithmeticum Pascalianum* [69].

For $y = 1$, binomial coefficients can be identified as coefficients of terms x^k in the polynomial expansion of $(1 + x)^n$. Since the Reed–Muller expressions can

[1] The cathedral of Anagni and also the Roman Basilica of Santa Maria in Cosmedin in Italy. The work is attributed to the family of architects Cosmati [68].

be compared to the Taylor expressions where x^k is replaced by the product of k binary variables, the links to the Reed–Muller expressions are clear and have been observed and discussed by several authors. At the same time, the links between the generator matrices in coding theory were disclosed.

2.1. Representations of Pascal Matrices. In the following, we will consider the extended binomial coefficients defined as $\binom{r}{s} = \frac{r!}{s!(r-s)!}$ if $r \geqslant s \geqslant 0$ and $q = 0$ if $r < s$.

There are three forms of the $(n \times n)$ Pascal matrix, $n \in N$, N-the set of natural numbers,

(1) Lower triangular $\mathbf{L}_n = [l_{i,j}]$, $i, j = 0, 1, \ldots, n-1$, with $l_{i,j} = \binom{i}{j}$,
(2) Symmetric $\mathbf{S}_n = [s_{i,j}]$, $i, j = 0, 1, \ldots, n-1$, with $s_{i,j} = \binom{i+j}{i}$,
(3) Upper triangular $\mathbf{U}_n = [u_{i,j}]$, $i, j = 0, 1, \ldots, n-1$, with $u_{i,j} = \binom{i}{j}$.

These matrices are unimodular, since their determinants are equal to 1. Further, $\mathbf{S}_n = \mathbf{L}_n \mathbf{U}_n$.

Example 1. The Pascal matrix is often written as a lower triangular matrix as in the case of a (5×5) matrix

$$\mathbf{L}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}.$$

Another way to write the Pascal matrix is as the upper triangular matrix as in the following example

$$\mathbf{U}_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 6 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Notice that lower and upper triangular Pascal matrices are related by transposition and, for example, $\mathbf{L}_5 = \mathbf{U}_5^T$.

Alternatively, the Pascal matrix can be written in symmetric form as

$$\mathbf{S}_5 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 6 & 10 & 15 \\ 1 & 4 & 10 & 20 & 35 \\ 1 & 5 & 15 & 35 & 70 \end{bmatrix}.$$

It is easy to check that $\mathbf{S}_5 = \mathbf{L}_5 \mathbf{U}_5$.

2.2. Matrix Exponentiation and the Pascal Matrix. The Pascal matrix is alternatively defined as the exponential of the matrix which has the sequence $1, 2, 3, 4, \ldots$ as elements of its subdiagonal and zero elsewhere. Recall that the matrix exponential is a matrix function on square matrices analogous to the ordinary exponential function e^x. If $\mathbf{X}$ is an $(n \times n)$ real or complex matrix, the exponential of $\mathbf{X}$, $e^{\mathbf{X}}$ is the $(n \times n)$ matrix determined by the power series

$$e^{\mathbf{X}} = \sum_{r=0}^{\infty} \frac{1}{r!}\mathbf{X}^r = \mathbf{I} + \mathbf{X} + \frac{1}{2!}\mathbf{X}^2 + \frac{1}{3!}\mathbf{X}^3 + \cdots,$$

where $\mathbf{I}$ is the $(n \times n)$ identity matrix.

A similar definition can be given in terms of the superdiagonal matrix with respect to the same sequence. It is clear that when $\mathbf{X}$ is a nilpotent matrix, i.e., there exist some k such that $\mathbf{X}^k = \mathbf{0}$ which is the zero matrix, i.e., an $(n \times n)$ matrix whose elements are equal to 0, the above series has k terms. The subdiagonal and superdiagonal matrices whose matrix exponential is the Pascal matrix with q rows and columns are nilpotent with $k = q$, which is a feature making the related computations feasible for reasonably large q.

Example 2. For $n = 2$, the (4×4) Pascal matrix can be generated as follows. In this case, consider the matrix

$$\mathbf{D} = \text{subdiag}(1, 2, 3) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \end{bmatrix}.$$

The direct computation shows that

$$\mathbf{D}^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix}, \qquad \mathbf{D}^3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 6 & 0 & 0 & 0 \end{bmatrix},$$

while $\mathbf{D}^4 = \mathbf{0}$. It follows that

$$e^{\mathbf{D}} = \mathbf{I} + \mathbf{D} + \frac{1}{2}\mathbf{D}^2 + \frac{1}{6}\mathbf{D}^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix},$$

which is the (4×4) Pascal matrix.

Example 3. The first 9 rows and columns of the Pascal matrix can be written as a (9×9) matrix

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 5 & 10 & 10 & 5 & 1 & 0 & 0 & 0 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 & 0 & 0 \\ 1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 & 0 \\ 1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1 \end{bmatrix}. \tag{2.1}$$

For all $n \in N$, the square matrices derived by selecting the first n rows and columns of the Pascal matrix are non-singular and, therefore, are used as transform matrices for the Pascal transform. This spectral transform has certain interesting applications. See [2, 21, 56, 57, 58, 75], and references therein. It is similar with the modified Pascal transform discussed below.

2.3. Gibbs Exponentiation and the Pascal Matrix. Another way to generate the Pascal matrix is the following.

Consider the group G of p-valued n-vectors $x = (x_1, \ldots, x_n)$ with the group operation defined as the componentwise addition modulo p. The group G has the order $g = |G| = p^n$.

For each $x \in G$, we define a p-adic contraction as $\sigma(x) = \sum_{i=1}^{n} x_i p^{n-i}$.

Notice that $\sigma(p^n - 1) = p^n - 1$.

We consider the space $P(G)$ of all functions $f : G \to Z_p$, where Z_p is the set of non-negative integers modulo p, and define the multiplication for f and g in $P(G)$ as [20]

$$\begin{aligned} &(fg)(0) = 0, \\ &(fg)(x) = \sum_{s=0}^{\sigma(x)-1} f(\sigma(x) - 1 - s)g(s), \quad \forall x \in G, \quad x \neq 0. \end{aligned} \tag{2.2}$$

Thus, we have

$$\begin{aligned} (fg)(0) &= 0, \\ (fg)(1) &= f(0)g(0), \\ (fg)(2) &= f(1)g(0) + f(0)g(1), \\ (fg)(3) &= f(2)g(0) + f(1)g(1) + f(0)g(2), \\ (fg)(4) &= f(3)g(0) + f(2)g(1) + f(1)g(2) + f(0)g(3), \\ &\vdots = \vdots \end{aligned}$$

This multiplication is called the convolutionwise Gibbs multiplication [59] referring to [20].

Columns of the Pascal matrix can be generated as the integer powers of a function $f(x)$ in $P(G)$ defined as

$$f(x) = 1, \quad \forall x \in G. \tag{2.3}$$

in terms of the Gibbs multiplication in the field of rational numbers.

2.4. Pascal Matrices over Finite Fields. The Pascal matrix can be computed also over finite fields of order p and then $(p^n \times p^n)$ matrices defined by the first p^n columns and rows of the Pascal matrix computed modulo p are used in functional expressions for n-variable p-valued functions and also to define non-binary Reed–Muller codes.

If the Pascal matrix over a finite field of order p^n, p-prime, is written as a lower triangular matrix, its elements are defined as

$$P_{i,j} = (i!)((i-j)!j!)^{-1} = \binom{i}{j} \mod p$$

for $i, j = 0, 1, \ldots, p^n - 1$ and, by convention, if $j > i$, then $p_{i,j} = 0$. In other words, $\mathbf{L}_{p^n}$ is a lower triangular matrix whose non-zero entries are the elements of the Pascal triangle taken mod p.

Example 4. The first 9 rows of the Pascal matrix calculated modulo 2 are

$$\mathbf{L}_{(3^2\times 3^2)} \mod 2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

It is obvious that first 8 rows and columns are identical to the binary Reed–Muller transform matrix of order $8 = 2^3$, which can be defined as the Kronecker product of the basic Reed–Muller matrix, i.e., the Pascal (2×2) matrix modulo 2. Thus, the Pascal $(2^n \times 2^n)$ matrix modulo 2 is the Reed–Muller matrix defined as

$$\mathbf{R}(n) = \bigotimes_{i=0}^{n} \mathbf{R}(1), \quad \mathbf{R}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}. \tag{2.5}$$

The $(p^n \times p^n)$ Pascal matrices over finite fields of a prime order p can be generated as the Kronecker product of the basic Pascal matrix of order p, $\mathbf{P}_p$, as [53], [54],

$$\mathbf{P}_{p^n} = \mathbf{P}_p \otimes \mathbf{P}_{p^{n-1}} \mod p.$$

Example 5. Consider the case $p = 2$ and $n = 3$. Then,

$$\mathbf{P}_{2^3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix},$$

which is the matrix defined by (2.5).

For the ternary case, $p = 3$ and $n = 2$,

$$\mathbf{P}_{3^2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 2 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \end{bmatrix}.$$

In [20], the binary Reed–Muller transform matrix has been generated by using the multiplication (2.2) modulo 2. The same Kronecker product structure of the Pascal matrix with entries computed modulo p is preserved for any prime p.

The Kronecker product structure however does not hold for a non-prime p, which is easy to verify by computing, for example, the third column of the Pascal matrix modulo 4.

Example 6. The first 9 rows of the Pascal matrix calculated modulo 3 are

$$\mathbf{L}_{(3^2\times 3^2)\mod 3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 2 & 2 & 0 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \end{bmatrix}. \tag{2.6}$$

It is obvious that this matrix can be generated as the Kronecker product

$$\mathbf{L}_{(3^2\times 3^2)\mod 3} = \mathbf{L}_3(1) \otimes \mathbf{L}_3(1), \quad \mathbf{L}_3(1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{2.7}$$

The inverse of the Pascal matrix over $GF(p)$, $\mathbf{Q}_p$, is defined as a matrix with elements

$$q_{ij} = \begin{cases} (-1)^{j-i}\binom{j}{i} \mod p, & \text{if } j \geqslant i, \\ 0, & \text{otherwise,} \end{cases}$$

for $i, j = 0, 1, \ldots, p-1$.

Further, for p-prime, $\mathbf{P}_p^p = \mathbf{I}_p$, where $\mathbf{I}_p$ is the identity matrix of order p, and $\mathbf{P}_p^{-1} \mathrm{mod}\ p = \mathbf{P}_p^{p-1} \mathrm{mod}\ p$ [55].

Example 7. The Pascal matrix over $GF(5)$ and its inverse

$$\mathbf{P}_p = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 & 1 \\ 0 & 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{P}_p^{-1} = \mathbf{P}_p^4 = \begin{bmatrix} 1 & 4 & 1 & 4 & 1 \\ 0 & 1 & 3 & 3 & 1 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

$$\mathbf{P}_{p^n}^{-1} = \mathbf{P}_p^{-1} \otimes \mathbf{P}_{p^{n-1}}^{-1} \mod p.$$

Example 8. For $p = 3$, if $\mathbf{J}_p$ is a $(p \times p)$ matrix with elements on the secondary diagonal equal to 1, skew identity matrix, then

$$\mathbf{P}_p^{-1} = \mathbf{P}_p^4 = \mathbf{J}_p \cdot \mathbf{P}_p^T \cdot \mathbf{J}_p,$$

and since

$$\mathbf{P}_{p^n}^{-1} = \mathbf{P}_p^{-1} \otimes \cdots \otimes \mathbf{P}_p^{-1} \mod p,$$

then

$$\mathbf{P}_p^{-1} = (\mathbf{J}_p \cdot \mathbf{P}_p^T \cdot \mathbf{J}_p) \otimes \cdots \otimes (\mathbf{J}_p \cdot \mathbf{P}_p^T \cdot \mathbf{J}_p) = (\mathbf{J}_p^{\otimes n} (\mathbf{P}_p^{\otimes n})^T \cdot \mathbf{J}_p^{\otimes n}).$$

In the following sections, we will discuss the application of basic $(p \times p)$ Pascal matrices computed modulo p in spectral representations of multiple-valued functions and definition of error-correcting codes.

3. Spectral Representations of p-valued Functions

A function $f \in P(G)$ is uniquely specified by enumerating the values it takes over the domain G. When the cardinality of the support set of G is large, enumeration is space inefficient, and often analytical representations are used. In other words, in many practical applications, it is convenient to decompose a function modeling a signal into a linear combination of functions modeling signals whose behavior is well known. These functions are called a basis, and it is desirable that they exhibit certain properties useful in practical applications. In this section, we will consider basis functions derived or related to Pascal matrices.

A basis X in $P(G)$ is any set of its p^n linearly independent vectors. In spectral representations of functions of p-valued variables, the main task is the following.

Given a function $f \in P(G)$, represent f as a linear combination of elements of a selected basis X.

TABLE 1. Basic Pascal transform matrices modulo p for $p = 2, 3, 4, 5$ and their inverses.

$\mathbf{G}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$	$\mathbf{G}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
$\mathbf{G}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix}$	$\mathbf{G}_5 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 1 & 3 & 3 & 1 & 0 \\ 1 & 4 & 1 & 4 & 1 \end{bmatrix}$
$(\mathbf{G}_2)^{-1} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$	$(\mathbf{G}_3)^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
$(\mathbf{G}_4)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 3 & 1 & 1 \end{bmatrix}$	$(\mathbf{G}_5)^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 4 & 1 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 4 & 3 & 2 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

Different bases are selected for different applications. In this paper, we are particularly interested in bases related to columns of the Pascal matrices over finite fields.

The set of p^n coefficients in this linear combination (the spectrum of f with respect to the basis X) is computed as the inner products of f and elements of X. Therefore, the spectral representation of f means the redistribution of the information content of f among the spectral coefficients. In other words, the spectral representation means assigning to a function f specified by the function vector $\mathbf{F} = [f(0), \ldots, f(p-1)]^T$ another vector of the same order $\mathbf{S}_f = [S_f(0), \ldots, S_f(p-1)]^T$ from the same or a different vector space.

For $p = 2$, a basis suitable from both theoretical and practical point of view is the Reed–Muller basis defined by (2.5), i.e., by columns of the Pascal matrix modulo 2. A direct generalization to $p > 2$ by using the corresponding basic matrices derived from the Pascal matrix computed modulo p is possible, however, as noticed in [37], a problem is that the property of self-inverseness for $\mathbf{R}(1)$ in (2.5) is not preserved as it can be seen from Table 1. Notice that in this table the inverse of $\mathbf{G}_4$ is not calculated in the finite field of order 4, but modulo 4, i.e., in the ring of integers modulo 4.

TABLE 2. Basic RMF-matrices for $p = 2, 3, 4, 5$.

$$\mathbf{R}_{2,RMF}(1) = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \qquad \mathbf{R}_{3,RMF}(1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{R}_{4,RMF}(1) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 1 & 3 & 3 \end{bmatrix} \quad \mathbf{R}_{5,RMF}(1) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 4 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 \\ 1 & 2 & 3 & 4 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The property of self-inverseness can be regained if instead of the constant function 1, we take the function $f(x) \equiv p - 1$, as it has been done in defining the Reed–Muller–Fourier (RMF) transforms [61, 62]. Table 2 shows the basic matrices for RMF-spectral transforms derived in this way for $p = 2, 3, 4, 5$.

The columns of the Pascal matrix modulo p can be described by monomials in terms of variables and powers of p-valued variables computed with the Gibbs multiplication [59]. For $p = 2$, these products can be identified as the logic AND of binary valued variables. For other values of p the powers of variables are computed in terms of the Gibbs multiplication, while products of variables and their powers are computed modulo p followed by the multiplication with $p - 1$.

Example 9. For $p = 2$, the first 8 columns of the Pascal matrix modulo 2 can be expressed as

$$\begin{aligned} \mathbf{X}_2 &= \begin{bmatrix} 1 & x_1 \end{bmatrix} \otimes \begin{bmatrix} 1 & x_2 \end{bmatrix} \otimes \begin{bmatrix} 1 & x_3 \end{bmatrix} \\ &= \begin{bmatrix} 1 & x_3 & x_2 & x_2x_3 & x_1 & x_1x_3 & x_1x_2 & x_1x_2x_3 \end{bmatrix}. \end{aligned}$$

For $p = 3$, we first generate the monomials

$$\begin{aligned} \mathbf{X}_3 &= \begin{bmatrix} x_1^{*0} & x_1^{*1} & x_1^{*2} \end{bmatrix} \otimes \begin{bmatrix} x_2^{*0} & x_2^{*1} & x_2^{*2} \end{bmatrix} \\ &= \begin{bmatrix} 2 & x_2 & x_2^{*2} & x_1 & x_1 \odot x_2 & x_1 \odot x_2^{*2} \\ & & & x_1^{*2} & x_1^{*2} \odot x_2 & x_1^{*2} \odot x_2^{*2} \end{bmatrix}, \end{aligned}$$

where $*$ states for the exponentiation based on the Gibbs multiplication defined in Table 3, and $\odot$ is the multiplication modulo 3 followed with the multiplication by 2. In this way, the basis functions in the RMF-transform are produced.

Example 10. Consider the switching function $f(x_1, x_2) = \overline{x}_1\overline{x}_2 \oplus x_1\overline{x}_2 \oplus x_1\overline{x}_2$, whose function vector is $\mathbf{F} = [1, 0, 1, 1]^T$. Table 4 shows the Reed–Muller, the Arithmetic, the Walsh, and the Vilenkin–Chrestenson transform matrices, and the corresponding spectra. The spectral coefficients are logic (Boolean) values, integers, and complex numbers, respectively. To perform the related computations, function values are correspondingly interpreted.

TABLE 3. 3EXP.

$*$	0	1	2
0	2	0	0
1	2	1	0
2	2	2	2

TABLE 4. Spectra for the function f in Example 10.

Transform	Spectrum	Field
$\mathbf{R}(2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$	$[1, 1, 1, 0]^T$	$GF(2)$
$\mathbf{A}(2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 1 \end{bmatrix}$	$[1, -1, 0, 1]^T$	Q
$\mathbf{W}(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$	$[3, 1, -1, 1]^T$	Q
$\mathbf{CV}(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & 1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$	$[3, i, 1, -i]^T$	C

As stated in [2], the discrete Legendre, Laguerre, Hermite, and binomial transforms are intimately related with the Pascal matrix [1, 3]. This feature is important, since procedures for the computation of these transforms can be defined in terms of the Pascal matrix, and its various decompositions in terms of sparse matrices or matrices which can be computed by the additions with no multiplication required [3].

In [2], it is defined a new discrete polynomial Pascal transform intended for applications in digital filter design [42, 43], three-dimensional object detection [33], and digital watermarking [35].

In this case, the transform matrix is defined by writing the rows of Pascal triangle into matrix form and alternating the signs of the columns. The basis functions are

defined as polynomials

$$P(x,k) = \frac{(-1)^k}{k!} x^{(k)}, \quad x \geqslant 1,$$

where $x^{(k)} = x(x-1)(x-2)\cdots(x-k+1)$.

Example 11. The first four polynomials are [2]

$$P(x,0) = 1, \;\; P(x,1) = -x, \;\; P(x,2) = \frac{1}{2}x(x-1), \;\; P(x,3) = -\frac{1}{6}\,x(x-1)(x-2).$$

Coefficients in this modified Pascal transform represent weighted differences in neighboring data values.

In matrix notation, the modified Pascal transform matrix is defined as [2]

$$\mathbf{P}_{\text{modified}}(2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -3 & 3 & -1 \end{bmatrix}.$$

An important property of this transform is that this is a self-inverse transform and, therefore, the same algorithm can be used to perform both the direct and the inverse transform.

3.1. Decomposition of the Pascal Matrix. The decomposition of the Pascal matrix is studied for the simplification of computing the related spectra and for the realization of the corresponding procedures in hardware. In the later case, an especial attention has been paid to the number of additions and multiplications, since this influence the performance of the produced hardware. For instance, in the case of FPGA based realizations, this directly determines the size of required adder and multiplier arrays.

Further, relationships of the Pascal transform with certain other transforms, as for example, the Hermite, binomial, and Laguerre transform, allows to speed up computations with these transforms by using various decompositions of the Pascal transform. When these relationship exist, then for a given transform, the transform matrix is expressed in terms of the Pascal matrix and some suitably defined auxiliary matrices. Then, the decomposition of the Pascal matrix is used to get expressions for the initial transform matrices simple to compute.

In this section, we will consider two out of many decompositions of the Pascal matrix that can be found in the literature.

The $(n \times n)$ Pascal matrix can be factorized as [3]

$$\mathbf{U}_n = \prod_{k=1}^{n-1} \mathbf{G}_k, \;\; \text{where } \mathbf{G}_k = \begin{bmatrix} \mathbf{I}_{n-1-k} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{k+1} \end{bmatrix},$$

and the $((k+1) \times (k+1))$ matrix $\mathbf{S}_{k+1}$ whose (i,j)-th element is given by

$$s_{i,j} = \begin{cases} 1, & \text{for } j \geqslant i, \\ 0, & \text{for } j < i. \end{cases} \quad i,j = 0,1,\ldots k.$$

By definition, $\mathbf{I}_1 = [1]$, and $\mathbf{S}_1 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$.

Example 12. Consider the (5×5) upper triangular Pascal matrix $\mathbf{U}_5$.

In this case, $n = 5$ and the matrix is factorized as

$$\mathbf{G}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{G}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{G}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Therefore, $\mathbf{U}_5 = \mathbf{G}_1\mathbf{G}_2\mathbf{G}_3\mathbf{G}_4$.

This decomposition reduces computing with Pascal matrix to the series of additions, which appears suitable for hardware realizations, for example on FPGAs.

Another interesting decomposition of the Pascal matrix is in terms of simple binary matrices with non-zero elements on the main diagonal and lower sub-diagonal while all other elements are 0 [34]. In this case,

$$\mathbf{L}_n = \prod_{k=1}^{n-1} \mathbf{H}_i, \text{ where } \mathbf{H}_k = \begin{bmatrix} \mathbf{I}_{n-1-k} & \mathbf{0} \\ \mathbf{0} & \mathbf{Y}_{k+1} \end{bmatrix}$$

and $\mathbf{Y}_{k+1}$ is a $((k+1) \times (k+1))$ matrix whose (i,j)-th element is defined as

$$y_{i,j} = \begin{cases} 1, & j = i,\ j = i-1, \\ 0, & j > i,\ j < i-1. \end{cases} \quad i, j = 0, 1, \ldots k.$$

This factorization is interesting since it leads to sparse factor matrices of a structure that enables derivation of fast algorithms resembling the so-called FFT algorithms with constant geometry [44].

Example 13. The (5×5) Pascal matrix in Example 1 can be factorized as

$$\mathbf{L}_5 = \mathbf{H}_1\mathbf{H}_2\mathbf{H}_3\mathbf{H}_4,$$

where

$$\mathbf{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix},$$

$$\mathbf{H}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \quad \mathbf{H}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

There are various decompositions of Pascal matrices defined in the literature. For example, the decomposition of Pascal matrices in terms of some other matrices with regular structure as Toeplitz and Henkel matrices leads to fast computation algorithms [65, 67, 74]. See, also [34]. For the decomposition of symmetric Pascal matrices in terms of Fibonacci matrices, see [71].

4. Error-Correcting Codes

A linear p-ary code C of length n is a linear subspace of the vector space $GF(p)^n$, where $GF(p)$ is the finite field of order p. The code has the length n (the codeword length) and the dimension k (message length), where k is the number of information digits, i.e., bits in the binary case. Such a code is denoted as an (n, k)-code.

In coding theory, block codes are an important class of error-correcting codes derived by assuming that the message (a sequence of symbols to be transmitted) is split into blocks of the specified length k and then to each block a codeword of length $n > k$ is assigned. In this case, another parameter is the minimum Hamming distance that also determines the number of errors a block code can detect and correct. If the Hamming distance is d, then the number of errors that can be detected and corrected are $d - 1$ and $\lfloor \frac{d-1}{2} \rfloor$, respectively.

An (n, k)-code can be defined by an $(k \times n)$ generator matrix $\mathbf{G}$ whose rows are k linearly independent vectors of the code C. The generator matrix can be represented in the so-called standard form as

$$\mathbf{G} = \left[\mathbf{I}_k \quad | \quad \mathbf{P}_{(k \times (n-k))} \right],$$

where $\mathbf{P}$ is a $(k \times (n-k))$ purposely selected matrix, and $\mathbf{I}_k$ is the $(k \times k)$ identity matrix.

Writing the generator matrix in this form, it becomes clear the statement that in coding theory there is no redistribution of the information content of represented functions and just a number of additional bits is added. If blocks are viewed as function vectors, there is a resemblance to the spectral representations where the spectrum S_f is assigned to a function f. The codewords assigned to blocks consists of two parts. The identity matrix $\mathbf{I}_k$ ensures that the first part of the codeword transmits the original message, while additional bits for error-correcting are computed by referring to the submatrix $\mathbf{P}$. Recall that linear transformation

over rows and columns of the generator matrix produce equivalent codes. The essential differences, however, are the following.

(1) In spectral representations the function vectors and spectra are of the same length. The code words are of length n larger than that of blocks k.
(2) In general, coding is an injective function from the space of information messages to the space of codewords. Therefore, there is not a redistribution of the information content of message over elements of the codewords as this is done in spectral representations. In the case of systematic codes, that are usually linear, there is a bit more of resemblance to spectral representations. In this case, also there is no a redistribution of the information content, since the information is transmitted as given. The additional bits that are assigned to each block are, however, computed from the information bits in a way resembling computing the spectrum.

Further differences between functional expressions and block codes are in linear spaces whose elements are function vectors and spectra, and blocks and codewords, respectively.

In the case of functional expressions, function vectors and their spectra are from linear spaces that are of the same length, but not necessarily over the same fields.

Example 14. A switching function f of n variables is specified by a binary string of length 2^n. The Reed–Muller spectrum S_f of f is another switching function represented by another binary string of the same length, while the Arithmetic transform assigns to f an integer-valued string of the same number of elements 2^n. There are also complex-valued transforms for switching functions [17, 18, 19], however, in all the cases the length of the spectrum is equal to the length of the function vector. The same feature of various spectral transforms holds for p-valued functions and in general for arbitrary discrete functions. The elements of the spectrum are computed by performing some operations over the elements of the function vectors. For example, the first element $S_f(0) = f(0)$ for the Reed–Muller and the Arithmetic transform, with the value interpreted as the Boolean-value and the integer, respectively. For the Walsh transform, $S_f(0)$ is the sum of all function values. The second element $S_f(1) = f(0) \oplus f(1)$ and $S_f(1) = f(0) + f(1)$ for the Reed–Muller and the Arithmetic transform, where $\oplus$ and $+$ denote the addition modulo 2 and the addition in the set of integers, respectively. For the Walsh transform in Paley ordering [60], $S_f(1)$ is the difference of the sum of the first and the second half of elements in the function vector. Other spectral coefficients in these transforms are computed in a similar way, with the operations over function values defined specifically for the corresponding transform.

In coding theory, in the case of linear block codes, the function spaces for blocks and codewords are over the same field, but of different order. That is important to notice since, this difference in the length of blocks and codewords is related to the number of errors that can be detected and corrected. The correspondence with the so-called extended spectra in transforms derived from various Ternary decision diagrams (TDDs) for the representation of switching functions [46] will be discussed later.

Example 15. The $(7,4)$-Hamming code is a block code where to each block of four information bits, a codeword of 7 bits is assigned. Thus, the information blocks can be viewed as the set of all 16 switching functions of two variables, while the codewords are particular 16 switching functions of 7 variables selected out of 128 possible functions such that they are at the minimum Hamming distance 3 of each other. Due to these extra bits, the code can correct any single error that might occur in any four-bit block of the information message.

If the four-bit block of the information message is $(m_0m_1m_2m_3)$, then the codeword is $(e_0e_1e_2e_3e_4e_5e_6)$, where $e_0 = m_0$, $e_1 = m_1$, $e_2 = m_2$, $e_3 = m_3$, and

$$e_4 = m_1 + m_2 + m_3, \quad e_5 = m_0 + m_2 + m_3, \quad e_6 = m_0 + m_1 + m_2,$$

which resembles computing the spectral coefficients as illustrated in Example 14. In matrix notation, computing the codewords can be performed by using the generator matrix

$$\mathbf{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right].$$

The redundant bits are computed by referring to the right submatrix in the generator matrix. Note that the equation $e = m_0 + m_1 + m_3$ can also be used instead of any of the three above equations to produce a valid Hamming code.

With respect to the analogy with spectral transforms, perhaps a better correspondence to the spectrum is in cyclic codes where a code can be defined as the set of vectors whose spectrum is zero for particular frequencies. There the spectrum is the Fourier spectrum over finite fields, so the corresponding transform matrix is the Vandermonde matrix of roots of unity in the finite field.

In cyclic q-ary codes with blocklength k, each codeword of length n is represented by a polynomial $c(x)$ of degree at most $k-1$. This polynomial can be written as $c(x) = a(x)g(x)$, where $g(x)$ is the generator polynomial for the code. If we consider the Fourier transform over the field $GF(q)$, then the Fourier spectrum of the codeword is $C_w = A_wG_w$. Recall that the Fourier coefficients take values in the extension field $GF(q^m)$. The spectrum of data block A_w is arbitrary, and the spectrum G_w of $g(x)$ specifies the indices w for which $C_w = 0$. For the given set of indices in the spectral domain $\chi = \{w_1, w_2, \ldots, w_{n-k}\}$, the cyclic code is the set of vectors over $GF(q)$ such that their spectrum is 0 in the components indexed by elements of χ. For more details about cyclic codes, see for instance [66].

Example 16. [22] Consider the generator polynomial $g(x) = 1 + x + x^3$ for the cyclic binary $(7,4)$ code. Coefficients of this polynomial determine the first row of the generator matrix. The other rows are determined by the cyclic shift of this row. Thus, the generator matrix of this code is

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

Regarding the remark about equality of fields for blocks and codewords, notice that, for example, in Walsh and Walsh-Hadamard codes that are derived from the Walsh matrices, the encoding $(1,-1) \to (1,0)$ is performed to make the codewords compatible with the binary blocks to be encoded, the elements of which take logic values 0 and 1.

Note that Hadamard codes can be defined with respect to Hadamard matrices of different orders, however, to ensure the linearity, the Hadamard matrices of orders 2^r, i.e., Walsh–Hadamard matrices $\mathbf{W}(r)$ are used. The code words are selected as rows of the matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{W}(r) \\ -\mathbf{W}(r) \end{bmatrix},$$

after encoding $(1,-1) \to (1,0)$. It should be explained that after encoding the unencoded $-\mathbf{W}(r)$ is encoded as $\mathbf{W}(r)+\mathbf{J}$, where $\mathbf{J}$ is matrix with all the elements equal to 1. Note that $\mathbf{W}(r)$ is the $(2^r \times 2^r)$ matrix and, therefore, there are 2^{r+1} codewords to encode blocks of length $r+1$. The ratio between the length of blocks and codewords $(r+1)/2^r$ is inconvenient, however, a very good feature of the code is that it can detect 2^{r-2} errors and correct an error less. The Walsh-Hadamard codes are particular examples of the more general class of codes, the Reed–Muller codes.

Example 17. For $r = 16$, the Hadamard code is defined by the rows of the matrix

$$\mathbf{C} = \begin{bmatrix} \mathbf{W}(4) \\ -\mathbf{W}(4) \end{bmatrix}.$$

The rows $1, 2, 3, 5, 9$ of this matrix in encoding $(1,-1) \to (1,0)$ are the basis of the $(16,5)$-code, thus, they are rows of its corresponding generator matrix reordered by decreasing Hamming weight. These rows in this encoding correspond to the constant vector 1 and Boolean variables x_1, x_2, x_3, and x_4, viewed as functions of four variables. Thus,

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

This code can detect 4 errors and correct three errors that might appear in any block of length 5.

In the case of Walsh–Hadamard codes to encode 32 blocks of length 5, the code words are selected from the space of 65536 binary vectors of length 16.

In this respect, there is a resemblance with extended spectra of transforms derived from Ternary decision diagrams used to represent binary switching functions [46]. Note that EXOR-TDDs, define the Extended Reed–Muller spectra. For a function of n binary variables, the Extended Reed–Muller spectrum has 3^n coefficients, and 2^n of them are the function values, while the remaining $3^n - 2^n$

coefficients are the coefficients in the Reed–Muller expressions for different polarities of variables and Kronecker expressions for the represented function f. These expressions all represent the same function f and if by evaluating them different values are obtained, it means that there is some error in the information bits of the message if the Extended Reed–Muller spectrum is viewed as a codeword. The value produced by the majority of expressions should be taken as the correct value.

Example 18. For $n = 2$, the Extended Reed–Muller spectrum is defined as

$$\mathbf{S}_{\mathrm{EXOR},f} = (\mathbf{E}(1) \otimes \mathbf{E}(1))\mathbf{F}, \quad \mathbf{E}(1) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix},$$

where $\mathbf{F}$ is the function vector of a switching function of two variables. This extended spectrum contains all function values and coefficients of all Fixed-polarity Reed–Muller expressions and all Kronecker expressions for switching functions of two-variables [60].

If $\mathbf{F}$ is interpreted as a four-bit block $[m_0 m_1 m_2 m_3]$ of an information message, then computing the Extended Reed–Muller spectrum resembles assigning a 9-bit codeword $\mathbf{c} = [c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8]$ determined as

$$\mathbf{c} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \right) \cdot \begin{bmatrix} m_0 \\ m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} m_0 \\ m_1 \\ m_0 \oplus m_1 \\ m_2 \\ m_3 \\ m_2 \oplus m_3 \\ m_0 \oplus m_2 \\ m_1 \oplus m_3 \\ m_0 \oplus m_1 \oplus m_2 \oplus m_3 \end{bmatrix}.$$

Table 5 shows the Extended Reed–Muller spectra for switching functions of two variables. These spectra are at the minimum Hamming distance 4 to each other. By using logic AND or logic OR instead of EXOR in the above computations, as this is done in the definition of the AND-TDDs and OR-TDDs [46], we get different codewords to be assigned to four-bit blocks. The usefulness of codes derived in this way is an open question probably worth of studying.

5. Reed–Muller Codes

Binary Reed–Muller codes are defined by selecting the codewords (rows of the generator matrix) from columns of the Pascal matrices computed modulo 2. These codes are usually denoted as $RM(d, r)$, where d is the order of the code, and r determines that the length of codewords is 2^r. Referring to the monomials describing the columns of the Pascal matrix, the order d of the code determines the number of variables appearing in the products corresponding to the columns of the Pascal matrix modulo 2 that should be taken as rows of the generator matrix [37].

TABLE 5. The Extended Reed–Muller spectra for 16 switching functions of two variables.

$m_0m_1m_2m_3$	$c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8$
0000	000000000
0001	000101011
0010	001001101
0011	001100110
0100	010010011
0101	010111000
0110	011011110
0111	011110101
1000	100010101
1001	100111110
1010	101011000
1011	101110011
1100	110000110
1101	110101101
1110	111001011
1111	111100000

TABLE 6. The correspondence between the Reed–Muller matrix (Pascal matrix modulo 2) and Reed–Muller codes.

0	00000000	$RM(0,3)$			
1	11111111				
x_1	00001111				
x_2	00110011		$RM(1,3)$		
x_3	01010101				
x_1x_2	00000011				
x_1x_3	00000101			$RM(2,3)$	
x_2x_3	00010001				
$x_1x_2x_3$	01010101				$RM(3,3)$

Example 19. For $p = 2$, Table 6 shows the correspondence between the columns of the Pascal matrix modulo 2, product terms in Reed–Muller expressions, and (n, k) Reed–Muller codes for $n = 3$ and $k = 0, 1, 2, 3$. Fig. 1 expresses the rows of the generator matrix for these codes as paths in the Reed–Muller decision tree for $n = 3$.

The generator matrix of the $(2, 4)$ Reed–Muller code, has rows defined by the Boolean monomials $1, x_1, x_2, x_3, x_4, x_1x_2, x_1x_3, x_1x_4, x_2x_3, x_2x_4, x_3x_4$.

The code $RM(n, k)$ is linear, comprises $2^{\sum_{i=0}^{k} \binom{n}{i}}$ codewords, and has minimum Hamming distance 2^{n-k}.

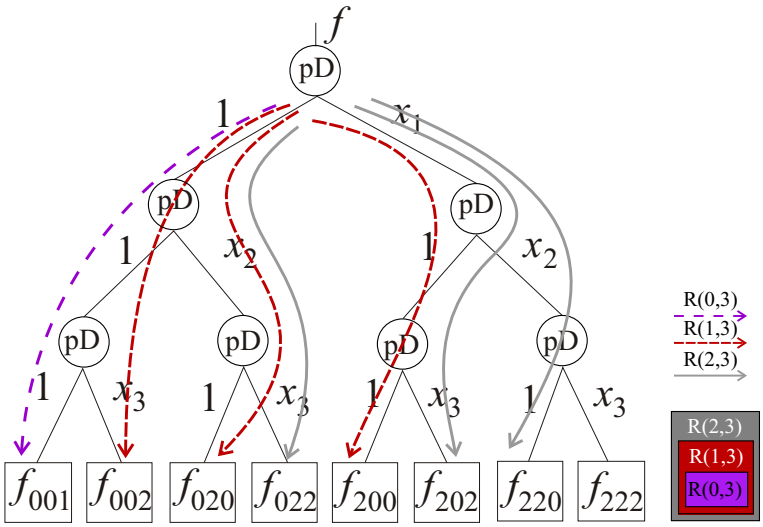

FIGURE 1. Rows of the generator matrices for the Reed–Muller codes $RM(0,3)$, $RM(1,3)$, and $RM(2,3)$.

There are several generalizations and the corresponding definitions of the Reed–Muller codes. Generalized or binary encoded Reed–Muller codes are defined in terms of generalized switching functions $f : Z_2^n \to Z_q$, where $q = 2^h$, $h \in N$, $h > 1$, where Z_k is the ring of integers modulo k. In other words, the integer coefficients from Z_q are assigned to Boolean monomials. The codes are defined in a manner similar to that used to define the classical binary Reed–Muller codes. It means, the rows of generating matrices are determined by Boolean monomials of certain orders with some of them multiplied by 2 [14] or powers of 2 [49] which are viewed as particular integers from the ring Z_{2^h}. For instance, the Boolean monomials of order k are multiplied by k. When extended into $(n \times n)$ matrices by following the same pattern in which they are defined, the generator matrices of such generalized Reed–Muller codes can be used as a basis to define the Reed–Muller expressions over the ring of integers. For $h = 2$, the quaternary Reed–Muller codes are obtained from thus generalized Reed–Muller codes [14]. This generalization of the Reed–Muller codes is useful for applications in *Orthogonal Frequency Division Multiplexing* (OFDM) since it is convenient with respect to the peak-to-mean envelope power ratio (PMEPR) [49]. In these applications, the main idea is to use block coding to transmit across the carriers polyphase sequences selected such that have small PMEPR. The problem is to select the appropriate linear codes. In [13], is it shown that the Reed–Muller codes generalized by attaching multiplicative weights to certain Boolean monomials offer good solutions permitting to select between few parameters relevant for such applications. The method can be used for binary, quaternary, octary, and also higher-order modulation schemes. Another

advantage is that the fast decoding schemes based on the Walsh-Hadamard matrices can be used. of Note that the multiplication by 2 and powers of 2 resembles the multiplication of certain monomials defined in terms of the Gibbs multiplication with $p-1$ to determine columns of the RMF-matrices as it is discussed above. The Gibbs multiplication was used to preserve the triangular form of the Reed–Muller transform matrices over finite fields, while the multiplication by $p-1$ ensures the self-inverseness. These modifications provided RMF-expressions to represent functions defined in finite fields. By borrowing bases derived by extending the generator matrices for the generalized Reed–Muller codes, the functional expressions for integer-valued functions are obtained. These expressions correspond to the arithmetic transform in the binary case, and express the triangular form of the transform matrices.

For example, the $(k \times n)$ generator matrices for the generalized Reed–Muller codes discussed above can be extended into the corresponding square matrices by adding rows corresponding to Boolean monomials of higher orders k multiplied by k. These matrices are a basis for the Reed–Muller expressions for functions of binary variables with function values in the ring of integers Z_{2^h} [13]. The multiplicative weights assigned to monomials of higher order can be useful in tuning the basis to produce functional expressions with a small number of terms.

Example 20. For $h > 1$, the so-called $ZRM_{2^h}(4,2)$ code discussed in [14] is defined by the generator matrix whose rows are defined by monomials

$$1, x_1, x_2, x_3, x_4, 2x_1x_2, 2x_1x_3, 2x_1x_4, 2x_2x_3, 2x_2x_4, 2x_3x_4.$$

This code contains $2^{5h} \cdot 2^{6(h-1)}$ codewords.

Many authors refer to [15, 24, 37] as sources of initial considerations of non-binary Reed–Muller codes. A particular example of a quaternary Reed–Muller code is presented in [5]. The present interest in Reed–Muller codes can be seen, for example, from [6, 12, 25, 26, 27]. An interesting approach to the generalization of Reed–Muller codes and study of their properties is carried out in [28, 29, 32, 30, 31].

In [48], it is defined a class of Lee metric codes that can be viewed as non-binary Reed–Muller codes with the binary Reed–Muller codes as a particular example. If p is an odd prime number, these codes are quasi-cyclic reversible [36]. In the context of discussions in this paper, the generator matrix of the codes $\mathbf{G}$ can be interpreted as follows.

The first row of $\mathbf{G}$ is the constant vector of length p^n whose all entries are 1. The following n rows are vectors representing the p-valued variables $x_1, x_2, \cdots, x_n$. The other rows are obtained as all possible products of distinct rows corresponding to p-valued variables. Therefore, the codewords are selected as certain rows of the Reed–Muller matrices [48]. The selected rows correspond to monomials in p-valued variables defined as products of variables, however, not their powers. Binary Reed–Muller codes are obtained for $p = 2$.

Example 21. For $p = 4$, and $n = 2$, the generator matrix of codes introduced in [48] is

$$\mathbf{G} = \begin{bmatrix} 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 \\ 0,1,2,3,0,1,2,3,0,1,2,3,0,1,2,3 \\ 0,0,0,0,1,1,1,1,2,2,2,2,3,3,3,3 \\ 0,0,0,0,0,1,2,3,0,2,0,2,0,3,2,1 \end{bmatrix}.$$

In this and other examples we presented, binary and various non-binary Reed–Muller codes are derived by selecting certain rows of the corresponding Reed–Muller matrices. It can be done conversely, the matrices appearing in definition of various Reed–Muller and related codes can be used to define new functional expressions for binary and p-valued, $p > 2$, logic functions. For example, the matrix $(\mathbf{G}_4)^{-1}$ in Table 1 can be used as the basic transform matrix to compute the spectra of quaternary functions in terms of the basis defined by columns of the matrix $\mathbf{G}_4$. In spectral techniques, the Kronecker product of basic matrices for a single variable permits extension to functions of an arbitrary number of variables. This is also possible when the basic matrices are taken from coding theory as these in Table 1 used in [37]. In that respect, it is possible and, moreover, convenient to look at construction schemes used in coding theory to extend the length of codes. In the following section, we will discuss this topic on the example of Plotkin constructions.

The good features of spectral representations derived from matrices in Table 1 is that the transform matrices are triangular (which is not the case for Galois-field transforms), and finite field arithmetic (for $p = 4$) is replaced by the modular arithmetic, which can be an important advantage in computing. For example, in computing on Graphics Processing Units (GPUs) reading arithmetic operations in finite fields for non-prime p from tables is time consuming, while modulo operations are directly implemented in both Nvidia CUDA and OpenCL programming environments. Further, codes are usually defined such that possess efficient coding and decoding algorithms, which can be a way to disclose fast computing algorithms for spectral transforms derived by using basic matrices from coding theory.

Example 22. Consider functional expressions for quaternary functions derived by using the matrix $(\mathbf{G}_4)^{-1}(1)$ in Table 1. The function $f(x_1, x_2) = x_1 \oplus x_2$ has the function vector $\mathbf{F} = [0,1,2,3,1,2,3,0,2,3,0,2,3,0,1,2]^T$. The spectrum of f is computed as

$$\mathbf{S}_f = (\mathbf{G}_4)^{-1}(2)\mathbf{F}, \quad (\mathbf{G}_4)^{-1}(2) = (\mathbf{G}_4)^{-1}(1) \otimes (\mathbf{G}_4)^{-1}(1),$$

and it is $\mathbf{S}_f = [0,0,0,0,1,1,0,0,0,3,0,0,0,1,0,0]^T$.

Due to the linearity of the code and its basic feature that it is derived from the Pascal matrix, it is clear that this spectral transform is useful in representing functions which can be represented analytically as sum of variables, their powers, and products of variables and their powers.

6. Plotkin Constructions and Spectral Transforms

In coding theory, Plotkin constructions are defined to obtain a binary code of length $2n$ from a given binary code of length n [41]. In general terms, if C_1 and C_2

TABLE 7. The Plotkin construction schemes for non-binary Reed–Muller codes for $p = 2, 3, 4, 5$.

p	Construction scheme
2	$(u + v \mid u)$
3	$(u + v + w \mid 2u + v \mid u)$
4	$(u + v + w + x \mid 3u + 2v + w \mid 3u + v \mid u)$
5	$(u + v + w + x + y \mid 4u + 3v + 2w + x \mid u + 3v + w \mid 4u + v \mid u)$

TABLE 8. The Plotkin constructions schemes for the RMF-codes for $p = 2, 3, 4, 5$.

p	Construction scheme
2	$(u + v \mid u)$
3	$(u + v + w \mid u + 2v \mid u)$
4	$(u + v + w + x \mid u + 2v + 3w \mid 3u + v \mid 3u)$
5	$(u + v + w + x + y \mid u + 2v + 3w + 4x \mid u + 3v + w \mid u + 4v \mid u)$

are two codes of length n, the code C of length $2n$ is obtained as

$$C = \{(u + v|u), u \in C_1, v \in C_2\},$$

where $|$ denotes the concatenation of sequences. The codes C_1 and C_2 do not need to be necessarily different codes. An important feature of Plotkin construction is that the dimension of the new code $d(C) \geqslant \min\{2d(C_1), d(C_2)\}$.

Referring to spectral techniques, it can be observed that the Plotkin construction is derived from the basic transform matrices used to define the corresponding functional expressions.

Example 23. For the binary Reed–Muller codes the Plotkin construction follows from $\mathbf{R}(1)$ since

$$\begin{bmatrix} v & u \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} u + v & | & u \end{bmatrix}$$

This example explains the way of defining non-binary Reed–Muller codes as suggested in [37], see also [27]. It also suggests that the following remark can be given.

Remark 1 (Reed–Muller codes and spectral transforms)**.** The Plotkin construction method for Reed–Muller codes is an alternative way to express the Kronecker product structure of the Reed–Muller transform and the same interpretation extends to non-binary cases.

Table 1 shows the generator matrices for non-binary Reed–Muller codes for $p = 2, 3, 4, 5$. Codes of higher dimensions can be defined by using the Plotkin construction schemes that are shown in Table 7.

The Plotkin construction can be used to define the non-binary Reed–Muller codes from the basic RMF-transform matrices in Table 2 in the same way as this is

done in [37] by using the basic matrices derived from the Pascal matrices modulo p in Table 1. Table 8 show the corresponding Plotkin construction schemes.

The symbolic notation for columns of the Pascal matrix modulo p in terms of p-valued variables and their powers with respect to the Gibbs multiplication directly leads to the definition of p-ary Reed–Muller codes in terms of thus generalized monomials of p-valued functions.

7. Closing Remarks

Studying relationships among important concepts might lead to interesting observations and certainly provides a deeper understanding of them. It often offers an alternative interpretation of well established concepts and can motivate their generalizations as well as extensions of various techniques dealing with particular concepts from an area to another. With this motivation, the present paper discusses Reed–Muller expressions and Reed–Muller error-correcting codes by starting from their matrix representations through relating these concepts to the Pascal matrices over finite fields.

Both concepts, the Reed–Muller expressions and the Reed–Muller codes, can be defined in terms of monomials (binary and p-valued) describing columns of the $(p^n \times p^n)$ matrix modulo p derived by using the Kronecker product of basic $(p \times p)$ Pascal matrices. The Pascal matrix can be generated from the constant function identically equal to 1 by using the Gibbs multiplication defined convolutionwise. The replacement of this function by the constant function $p-1$ leads to the Reed–Muller-Fourier expressions. The corresponding Reed–Muller-Fourier codes can be defined from the p-valued monomials used in these expressions. The Plotkin constructions in coding theory can be viewed as another way to express the Kronecker product structure of spectral transform matrices.

References

[1] M. F. Aburdene, J. E. Dorband, *Unification of Legendre, Laguerre, Hermite, and Binomial transforms using Pascal's matrix*, Multidimens. Syst. Signal Process. **5** (1994), 301–305.

[2] M. F. Aburdene, T. J. Goodman, *The discrete Pascal transform and its applications*, IEEE Signal Processing Letters **12**(7) (2005), 493–495.

[3] M. F. Aburdene, R. J. Kozick, R. S. Magargle, J. D. Maloney-Han, C. M. Coviello, *Discrete polynomial transform representation using binary matrices and flow diagrams*, Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. **2** (2001), 1141–1144.

[4] N. N. Aizenberg, O. T. Trofimljuk, *Conjunctive transforms for discrete signals and their applications of tests and the detection of monotone functions*, Kibernetika **5**, K, 1981, in Russian.

[5] A. E. Ashikhmin, S. N. Litsyn, *Fast Decoding of Non-Binary First Order Reed–Muller Codes*, Appl. Algebra Eng. Commun. Comput. **7** (1996), 299–308.

[6] A. Ashikhmin, E. Knill, *Nonbinary quantum stabilizer codes*, IEEE Trans. Inform. Theory **47** (2001), 3065–3072.

[7] Ph. W. Besslich, *Efficient computer method for XOR logic design*, IEE Proc., Part E **129** (1982), 15–20.

[8] Ph. W. Besslich, *Spectral processing of switching functions using signal flow transformations*, in: M. G. Karpovsky (ed.), *Spectral Techniques and Fault Detection*, Academic Press, Orlando, Florida, 1985.

[9] J. T. Butler, G. W. Dueck, S. Yanushkevich, V. Shmerko, *On the number of generators for transeunt triangles*, Discrete Appl. Math. **108** (2001), 309–316.
[10] ———, *On the use of transeunt triangles to synthesize fixed-polarity Reed–Muller expansions of functions*, Proc. Reed-Muler 2009 Wrokshop, Naha, Okinawa, Japan, May 23–24, 2009.
[11] G. S. Call, D. J. Velleman, *Pascal's matrices*, Amer. Math. Monthly **100** (1993), 372–376.
[12] D. J. Campbell, H. Anwar, D. E. Browne, *Magic-state distillation in all prime dimensions using Quantum Reed–Muller codes*, Phys. Rev. X **2**(4) (2012), 21–41.
[13] J. A. Davis, J. Jedwab, *Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed–Muller codes*, Tech. Rept., HP Laboratories Bristol, HPL-97–158, December 1997, 1–27.
[14] ———, *Peak-to-mean power control in OFDM, Golay complementary sequences, and Reed–Muller codes*, IEEE Trans. Inform. Theory **45**, 1999, 2397–2417.
[15] P. Delsarte, J.-M. Goethals, F. J. MacWilliams, *On generalized Reed–Muller codes and their Relatives*, Inf. Control **16** (1974), 403–442.
[16] G. W. Dueck, D. Maslov, J. T. Butler, S. N. Yanushkevich, V. P. Shmerko, *A method to find the best mixed polarity Reed–Muller expression using transeunt triangle*, Proc. 5th Reed–Muller 2001 Workshop, Starkville, Mississippi, USA, August 10–11, 2001, 82-92.
[17] B. J. Falkowski, S. Rahardja, *Properties and applications of unified complex Hadamard transforms*, Proc. 27th Int. Symp. on Multiple Valued Logic, Antigonish, Nova Scotia, Canada, May 28–30, 1997, 131–136.
[18] B. J. Falkowski, *Fast multi-polarity complex Hadamard transform for logic functions*, Proc. 28th Int. Symp. on Multiple-Valued Logic, May 27–29, Fukuoka, Japan, 1998, 180–185.
[19] ———, *Family of generalised multi-polarity complex Hadamard transforms*, IEE Proc. Vision, Image and Signal Processing **145**(6) (1998), 371–378.
[20] J. E. Gibbs, *Instant Fourier transform*, Electron. Lett. **13**(5) (1977), 122–123.
[21] T. J. Goodman, M. F. Aburdene, *On discrete Pascal transform, Poisson sequence and Laguerre polynomials*, Electron. Lett. **43**(14) (2007), 780–781.
[22] J. I. Hall, *Notes on Coding Theory*, classroom notes, Department of Mathematics Michigan State University, Michigan, USA, August 2012.
[23] B. Harking, C. Moraga, *Efficient derivation of Reed–Muller expansions in multiple-valued logic systems*, Proc. 22nd Int. Symp. on Multiple-Valued Logic, May 27–29, 1992, Sendai, Japan, 436–441.
[24] T. Kasami, S. Lin, W. W. Peterson, *New generalizations of the Reed–Muller codes Part I : Primitive codes*, IEEE Trans. Inform. Theory, **14**(2) (1968), 189–199.
[25] N. Kashyap, A. Thangaraj, *The treewidth of MDS and Reed–Muller codes*, IEEE Trans. Inform. Theory **58**(7) (2012), 4837–4847.
[26] T. Kaufman, S. Lovett, E. Porat, *Weight distribution and list-decoding size of ReedMuller codes*, IEEE Trans. Inform. Theory **58**(5) (2012), 2689–2696.
[27] F. R. Kschischang, S. Pasupathy, *Some ternary and quaternary codes and associated sphere packings*, IEEE Trans. Inform. Theory **38**(2) (1992), 227-246.
[28] É. Leducq, *Second weight codewords of generalized Reed–Muller codes*, arXiv:1203.5244 [math.NT].
[29] ———, *Rayon de recouvrement des codes de Reed–Muller généralisés*, Seminar of Number Theory of the Institut de Mathmatiques de Bordeaux, University of Bordeaux 1, Bordeaux, France, December, 4, 2011.
[30] ———, *Autour des codes de Reed–Muller généralisés*, École Doctorale Paris Centre, Thèse de doctorat, Université Paris VII-Denis Diderot, December 2, 2011.
[31] ———, *A new proof of Delsarte, Goethals and Mac Williams theorem on minimal weight codewords of generalized Reed–Muller code*, Finite Fields Appl. **18**(3) (2012), 581–586.
[32] ———, *On the covering radius of first order generalized Reed–Muller codes*, IEEE Trans. Inform. Theory **59**(3) (2013), 1590–1596.
[33] B. Li, J. Shen, *Range-image-based calculation of three-dimensional convex object moments*, IEEE Trans. Robot. Autom. **9**(4) (1993), 484–490.

[34] X.-G. Lv, T.-Z. Huang, Z.-G. Ren, *A new algorithm for linear systems of the Pascal type*, J. Comput. Appl. Math. **225** (2009), 309–315.
[35] J. R. H. Martin, M. Kutter, *Information retrieval in digital watermarking*, IEEE Commun. Mag. **39**(8) (2001), 110–116.
[36] J. L. Massey, *Reversible codes*, Inf. Control **7**(3) (1964), 369–380.
[37] J. L. Massey, D. J. Costello, Jr., J. Justesen, *Polynomial weights and code constructions*, IEEE Trans. Inform. Theory **IT-19**(1) (1973), 101–110.
[38] D. Maslov, *A Method to Find the Best Mixed Polarity Reed–Muller Expansion*, Master of Science Thesis, The Faculty of Computer Science, The Univeristy of New Brunswick, June 2001.
[39] D. E. Muller, *Application of Boolean algebra to switching circuits design and to error detection*, IRE Trans. Electron. Comp. **EC-3** (1954), 6–12.
[40] ———, *Boolean algebras in electric circuit design*, Am. Math. Mon. **61**(7), Part 11 (1954), 27–28.
[41] M. Plotkin, *Binary codes with specified minimum distances*, IEEE Trans. Inform. Theory **6**(4) (1960), 445–450.
[42] B. Pšenička, F. Garcia-Ugalde, *Z transform from lowpass to bandpass by Pascal matrix*, IEEE Signal Process. Lett. **11**(2) (2004), 282–284.
[43] B. Pšenička, F. Garcia-Ugalde, A. Herrera-Camacho, *The bilinear Z transform by Pascal matrix and its application in the design of digital filters*, IEEE Signal Process. Lett. **9**(11) (2002), 368–370.
[44] K. R. Rao, D. N. Kim, J.-J. Hwang, *Fast Fourier Transform-Algorithms and Applications*, Springer, 2010.
[45] S. M. Reed, *A class of multiple error correcting codes and their decoding scheme*, IRE Trans. Inf. Th. **PGIT-4** (1954), 38–49.
[46] T. Sasao, *Ternary decision diagrams and their applications*, in: T. Sasao, M. Fujita eds., *Representation of Discrete Functions*, Kluwer Academic Publishers, 1996, 269–292.
[47] T. Sasao, Ph. W. Besslich, *On the complexity of MOD-2 sum PLA's*, IEEE Trans. Comput. **33**(2) (1990), 262–266.
[48] Ch. Satyanarayana, *Lee metric codes over integer residue rings*, IEEE Trans. Inform. Theory **IT-25**(2) (1979), 250–254.
[49] K.-U. Schmidt, A. Finger, *New codes for OFDM with low PMEPR*, Proc. Int. Symp. on Information Theory, (ISIT 2005), September 4–9, 2005, 1136–1140.
[50] W. Sierpiński, *Sur une courbe dont tout point est un point de ramification*, Compt. Rendus Acad. Sci., Paris, **160** (1915), 302–305.
[51] W. Sierpiński, *On courves which contain the image of any other courve*, Mat. Sb. **30** (1916), 267–287.
[52] ———, *On a courve every point of which is a point of ramification*, Prace Mat.-Fiz. **27** (1916), 77-86 (in Polish).
[53] E. Sakk, S. Wicker, *Wavelet packets for error control coding*, Proceedings of the SPIE San Diego, CA, USA, Vol. 5207, 48th Annual Meeting (Wavelets X), August 3–8, 2003
[54] E. Sakk, S. Small, *The Fourier Convolution Theorem over Finite Fields: Extensions of Its Application to Error Control Coding*, in: Salih, S. ed., *Fourier Transform Applications*, InTech, 2012, 231–248, ISBN: 978-953-51-0518-3.
[55] E. Sakk, *Wavelet Packet Formulation of Generalized Reed–Muller Codes*, PhD thesis, Cornell University, Ithaca, NY, USA, 2002.
[56] A. N. Skodras, *On the computation of the Discrete Pascal transform*, Hellenic Open University: Technical Report HOU-CS-TR-2005-06-EN, December 2005, 1–9.
[57] ———, *Fast Discrete Pascal transform*, Electron. Lett. **42**(23) (2006), 1367–1368.
[58] ———, *Efficient computation of the Discrete Pascal transform*, Proc. 14th European Signal Processing Conference (EUSIPCO 2006), Florence, Italy, September 2006.

[59] R. S. Stanković, *Some remarks on Fourier transforms and differential operators for digital functions*, Proc. 22nd Int. Symp. on Multiple-Valued Logic, May 27–29, 1992, Sendai, Japan, 365–370.

[60] R. S. Stanković, J. T. Astola, *Spectral Interpretation of Decision Diagrams*, Springer, 2003.

[61] R. S. Stanković, C. Moraga, *Reed–Muller-Fourier representations of multiple-valued functions over Galois fields of prime cardinality*, in: Kebschull, U., Schubert, E., Rosentiel, W., Eds., *Proc. IFIP WG 10.5 Workshop on Applications of the Reed–Muller Expansion in Circuit Design*, 16.–17.9.1993, Hamburg, Germany, 115–124.

[62] R. S. Stanković, M. Stanković, C. Moraga, T. Sasao, *Calculation of Reed–Muller-Fourier coefficients of multiple-valued functions through multiple-place decision diagrams*, Proc. Twenty-Fourth International Symposium on Multiple-Valued Logic, May 25–27, 1994, 82–88.

[63] I. Stewart, *Four encounters with Sierpiński's gasket*, J. Math. Intell. **17**(1) (1999), 52–64.

[64] V. P. Suprun, *Fixed polarity Reed–Muller expressions of symmetric Boolean functions*, Proc. IFIP WG 10.5 Workshop on Application of the Reed–Muller Expansions in Circuit Design, Chiba, Tokyo, Japan, August 27–29, 1995, 246–249.

[65] Z. Tang, R. Duraiswami, N. Gumerov, *Fast algorithms to compute matrix-vector products for Pascal matrices*, Technical Reports from UMIACS UMIACS-TR-2004-08, Department of Mathematics and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, March 25, 2004, 1–19.

[66] J. H. van Lint, *An Introduction to Coding Theory*, 2nd ed., Springer, 1992.

[67] X. Wang, L. Z. Lu, *A fast algorithm for solving linear systems of Pascal type*, Appl. Math. Comput. **175** (2006) 441–451.

[68] Wikipedia, The Free Encyclopedia, `http://en.wikipedia.org/wiki/Sierpinski_triangle#cite_note-1`, accesed March 18, 2013.

[69] WikipediA, The Free Encyclopedia, `https://en.wikipedia.org/wiki/Pascal's_triangle#cite_note-8`, accesed March 24, 2013.

[70] S. Wolfram, *A New Kind of Science*, Champaign, IL, USA, Worfram Media, 2002, pages 870 and 931–931.

[71] Z. Zhang, X. Wang, *A factorization of the symmetric Pascal matrix involving the Fibonacci matrix*, Discrete Appl. Math. **155** (2007), 2371–2376.

[72] I. I. Zhegalkin, *O tekhnyke vychyslenyi predlozhenyi v symbolytscheskoi logykye*, Math. Sb. **34** (1927), 9–28, in Russian.

[73] ———, *Arifmetizatiya symbolytscheskoi logyky*, Math. Sb. **35** (1928), 311–377, in Russian.

[74] Z. Zhi-zheng, *The linear algebra of the generalized Pascal matrix*, Linear Algebr. Appl. **271** (1997) 169–177.

[75] Q.-C. Zhong, A. K. Nandi, M. F. Aburdene, *Efficient implementation of discrete Pascal transform using difference operators*, Electron. Lett. **43**(24) (2007), 1348–1350.