mathematics

MDPI

Article

Some Results on Self-Complementary Linear Codes

Maria Pashinska-Gadzheva [1], Iliya Bouyukliev [1,*] and Valentin Bakoev [2]

[1] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, 5000 Veliko Tarnovo, Bulgaria; mariqpashinska@math.bas.bg
[2] Faculty of Mathematics and Informatics, St. Cyril and St. Methodius University of Veliko Tarnovo, 5000 Veliko Tarnovo, Bulgaria; v.bakoev@ts.uni-vt.bg
* Correspondence: iliyab@math.bas.bg

Abstract: Binary codes have a special place in coding theory since they are one of the most commonly used in practice. There are classes of codes specific only to the binary case. One such class is self-complementary codes. Self-complementary linear codes are binary codes that, together with any vector, contain its complement as well. This paper is about binary linear self-complementary codes. A natural goal in coding theory is to find a linear code with a given length n and dimension k such that the minimum distance d is maximal. Codes with these properties are called optimal. Another important issue is classifying the optimal codes, i.e., finding exactly one representative of each equivalence class. In some sense, the classification problem is more general than the minimum distance bounds problem. In this work, we summarize the classification results for self-complementary codes with the maximum possible minimum distance and a length of up to 20. For the classification, we developed a new algorithm that is much more efficient compared to existing ones in some cases.

Keywords: binary self-complementary codes; optimal codes; classification

MSC: 94B05

Citation: Pashinska-Gadzheva, M.; Bouyukliev, I.; Bakoev, V. Some Results on Self-Complementary Linear Codes. *Mathematics* **2023**, *11*, 4950. https://doi.org/10.3390/math11244950

Academic Editor: Patrick Solé

Received: 14 November 2023
Revised: 9 December 2023
Accepted: 12 December 2023
Published: 14 December 2023

1. Introduction

In our previous work [1], we studied binary linear self-complementary codes meeting the Grey–Rankin bound. We considered the relations among six infinite families of binary linear codes with two and three nonzero weights that are closely connected to these self-complementary codes. Furthermore, a large part of our research is related to binary self-dual codes, which are also self-complementary. Reed–Muller codes, which are related to Boolean functions, are also self-complementary. This motivated us to take a closer look at the class of self-complementary codes and determine when these codes for a given length and dimension are optimal, i.e., when they have the largest minimum distance among all binary linear $[n, k]$ codes.

Classification is an important problem concerning many types of combinatorial objects. Due to its complexity, it is possible only for structures with relatively small or very specific parameters. Classification results have been obtained for different classes of linear codes: self-dual, self-orthogonal, LCD, etc. For self-complementary codes, so far there have only been partial results and no systematic classification.

A previous systematic study of self-complementary codes was presented in [2], but the authors mainly dealt with the problem of bounds for the minimum distance. In our study, we extend this research in two directions. One is presenting the exact number of inequivalent optimal self-complementary codes with given parameters. The other is exploring codes with larger lengths. We classify all self-complementary codes of length $n \leq 20$. As can be seen in the attached tables, in most cases, the constructed self-complementary codes are optimal in general (as linear codes with the same length and dimension), but there are exceptions.

A binary linear code is self-complementary if it contains the all-ones vector $\mathbf{1} = (11\ldots1)$. Thus, for each codeword $c \in C$, the complementary vector $\bar{c}$ is also a codeword in C, where $\bar{c}$ is obtained by replacing each 0 in c with 1 and each 1 with 0. For example, all self-dual codes are also self-complementary. Furthermore, the special classes of self-complementary codes that meet the Gray–Rankin bound and the two- and three-weight codes connected to them are of great interest [1,3,4]. Such codes are connected to many combinatorial structures such as SDP designs (a combinatorial design has the symmetric difference property, or is an SDP design if the symmetric difference of any two blocks is either a block or the complement of a block), bent functions, strongly regular graphs, and others. As seen in [5], self-complementary codes are also important for error detection. The research in this area is generally focused on codes meeting the Gray–Rankin bound.

Let $d(n,k)$ be the maximum possible Hamming distance of a binary linear code for given values of n and k. Codes with parameters $[n,k,d(n,k)]$ are called optimal. An important aspect of research in this field is the calculation of the number of inequivalent optimal codes. The problems for bounds of minimum distance of codes and code classification have been considered by many authors. Summary results for optimal binary self-orthogonal and self-dual codes are presented in [6]. Classification results of binary linear codes can be found in the research project in [7] and many others. A very accessible and important source of information on minimum distance bounds for linear codes over fields of up to nine elements is *Code Tables: Bounds on the parameters of various types of codes* [8].

In this work, we extend the existing results on the bounds for the maximum possible minimal distance of linear self-complementary codes presented in [2]. Let $d_{SC}(n,k)$ be the maximum possible minimum distance among all self-complementary $[n,k]$ codes. We present the bounds for $d_{SC}(n,k)$ and the classification results for optimal self-complementary codes with $n \leq 20$. Some relations to known families of codes are also presented.

To find both the bounds and the classification results, we mainly use two algorithms. Both are of the isomorph-free generation type and are based on the concept of canonical augmentation. However, there is a big difference between them. One algorithm is described in detail in [9] and is implemented in the program GENERATION. The other algorithm is new and much more effective in cases with a large number of inequivalent codes, and its detailed description is given below.

This paper is organized as follows. Section 2 presents some basic notations. Section 3 gives information about the algorithm strategies that are used. Section 4 presents the main algorithm. Section 5 presents the obtained results. A brief conclusion is given in Section 6.

2. Basic Notations

A binary linear $[n,k,d]$ code C is a k-dimensional subspace of the vector space $\mathbb{F}_2^n$. The *Hamming weight* of a binary vector is the number of its nonzero coordinates. The *minimum weight* (or distance) d of a binary linear code is the smallest weight among all nonzero codewords in the code. The elements of C are called *codewords*. All linear codes are defined by a $k \times n$ matrix whose rows form a basis of C, called a *generator matrix* of the code. Two binary linear codes, C_1 and C_2, are equivalent if the codewords of C_2 can be obtained from the codewords of C_1 by permutation of the coordinates of C_1. A permutation $\sigma \in S_n$ is an *automorphism* of C if $C = \sigma(C)$, and the set of all automorphisms of C forms a group called the *automorphism group* of C, which is denoted by $\mathrm{Aut}(C)$ in this paper. If C has length n, then the number of codes equivalent to C is $n!/|\mathrm{Aut}(C)|$.

The classification problem can be considered as finding the generator matrices of all inequivalent $[n,k,d]$ binary codes for given values of n, k, and d. The *dual code* of C is $C^{\perp} = \{u \in \mathbb{F}_2^n : u \cdot v = u_1v_1 + u_2v_2 + \cdots + u_nv_n = 0 \text{ for all } v \in C\}$. If C is an $[n,k,d]$ code, then $C^{\perp}$ is an $[n,n-k,d^{\perp}]$ code, and $d^{\perp}$ is called the *dual distance* of C. A generator matrix H for the dual code is called a parity check matrix of the code C. A linear code is *self-dual* if $C = C^{\perp}$. All self-dual codes have an even length. Codes consisting of codewords with only even weights are called *even*, and if all codewords have weights divisible by 4, the code is called doubly even. The polynomial $W(y) = 1 + A_1y + A_2y^2 + \cdots + A_ny^n$ is called the

weight enumerator of the linear $[n, k]$ code C if A_i is equal to the number of codewords in C of weight i.

The next proposition gives a connection between the weights of the rows of a generator matrix and the columns of a parity check matrix.

Lemma 1. *Any even linear code C has a parity check matrix whose columns have odd weights, and its dual is a self-complimentary code.*

Proof. Let $G = [I_k \; P]$ be a generator matrix for C in systematic form. Then, each row of P has an odd weight. And accordingly, the parity check matrix $H = \left[P^T \; I_{n-k}\right]$ has only odd weight columns. The sum of all rows of H gives the all-ones vector. □

A *subcode* of C is a linear subspace of the code. The residual code Res(C,c) with respect to a codeword $c \in C$ is the restriction of C to the zero coordinates of c. If C is self-complementary, then its residual code with respect to a codeword $c \neq (1 \ldots 1)$ is also self-complementary. A lower bound on the minimum distance of the residual code is given by the following theorem.

Theorem 1 ([10], Lemma 3.9)**.** *Suppose C is a binary $[n, k, d]$ code and suppose $c \in C$ has weight w, where $d > w/2$. Then, $Res(C, c)$ is an $[n - w, k - 1, d']$ code with $d' \geq d - w + \lceil w/2 \rceil$.*

The following proposition gives a lower bound on the dual distance.

Lemma 2. *Suppose C is a binary $[n, k, d]$ code with dual distance $d^\perp$, $c \in C$, and the dimension of $Res(C, c)$ is $k - 1$. Then, the dual distance of $Res(C, c)$ is at least $d^\perp$.*

For any length n and minimal distance d, the Gray–Rankin bound [11–13] is an upper bound for the cardinality of a binary self-complementary code C. It states that

$$|C| \leq \frac{8d(n-d)}{n-(n-2d)^2} \tag{1}$$

provided that the right-hand side is positive. The bound also holds for nonlinear codes, but we only consider linear codes here. This bound gives a relation between the number of codewords and the length and minimum distance of a code. Linear codes meeting the bound exist only for specific parameters, as shown in [14]. Subcodes and residuals of codes meeting the Gray–Rankin bound are of great research interest since they are connected to many other combinatorial structures [15].

3. Strategy Used in the Algorithms

In this section, we present the main ideas we follow in developing the code classification algorithm. This algorithm works for linear codes in general. In the beginning, we note that linear codes have a good description in terms of generator matrices. A generator matrix G defines a code C, which can be considered a representative of its equivalence class. Therefore, for the classification of binary codes with parameters n and k, it is necessary to construct exactly one generator matrix of a linear code from each equivalence class.

The algorithms for the classification of codes are characterized by the fact that exhaustive generation of all matrices that generate codes with the desired parameters and properties must be carried out. The construction of the matrices can be carried out row by row or column by column depending on whether we have more constraints on the weight distribution and minimum distance or on the dual distance. In order for the algorithms to be effective, it is necessary to consider as few sub-objects as possible in the generation process, without losing non-equivalent solutions at the last step. We can consider that the matrices we want to construct are in a systematic form. Then, the unknown part of the matrix consists of $n - k$ columns. If we want to construct only the set of these inequivalent

$[n,k,d]$ codes that have as a residual code, with respect to a codeword with weight d, a given code C_r with parameters $[n-d,k-1,d' \geq d/2]$, then the number of unknown columns is $n-d-1$ because we consider a generator matrix G in the following form

$$G = \left(\begin{array}{c|c|c|c} 1 & 0\ldots0 & 0\ldots0 & 11\cdots1 \\ \hline 0 & I_{k-1} & G_r & X \end{array}\right) \tag{2}$$

where $(I_{k-1}|G_r)$ generates the residual code C_r of C.

In the same notations, the following matrix is a parity check matrix of the code C.

$$G^{\perp} = \left(\begin{array}{c|cc|cc} 0 & G_r^T & & I_{n-k-d+1} & 0 \\ \hline 1 & & & & \\ \vdots & X^T & & 0 & I_{d-1} \\ 1 & & & & \end{array}\right). \tag{3}$$

The use of residual codes, as seen in [16], has proven to be very effective. The problem of finding the equivalent up to extension sub-objects (sub-matrices) in the generation process is of great importance. In other words, the problem is how to construct a minimum number of sub-matrices in the generation process without losing non-equivalent solutions. The different approaches are described in the monograph in [17].

Here, we use the concept of canonical augmentation [18]. This method is very effective for the classification of combinatorial structures. The construction is recursive; it consists of steps in which inequivalent objects are obtained from smaller objects by extending them in a special way. Canonical augmentation uses a canonical form to check the so-called "parent test" and considers only objects that have passed the test. The main idea in our case is to construct column by column only inequivalent linear codes using residual codes and in this way, to have a classification of these objects. The technique of canonical augmentation is used for the classification of different types of codes [9,19].

The second algorithm we use in our research is also based on canonical augmentation. In this algorithm, the fixed part of the generator matrix is only the identity matrix. It is presented in [9] and implemented in the program GENERATION, which is the first module of the software package QEXTNEWEDITION (standard version 1.1 for Linux; author: Iliya Bouyukliev; Veliko Tarnovo, Bulgaria). The program is available at http://www.moi.math.bas.bg/moiuser/~data/Software/QextNewEdition (accessed on 13 November 2023).

4. About Canonical Form and Canonical Augmentation

Codes that are equivalent belong to the same equivalence class. Every code can serve as a representative for its equivalence class. To construct all inequivalent codes with given parameters means to have one representative of each equivalence class. To accomplish this, we use the concept of a canonical representative, selected on the base of some specific conditions.

Let G be a group acting on a set Ω. This action defines an equivalence relation such that the equivalence classes are the G-orbits in Ω. We wish to find precisely one representative of each G-orbit and, therefore, we use a so-called canonical representative map.

Definition 1 ([17]). *A canonical representative map for the action of the group G on the set Ω is a function $\rho : \Omega \to \Omega$ that satisfies the following two properties:*

1. *For all $X \in \Omega$, it holds that $\rho(X) \cong X$;*
2. *For all $X, Y \in \Omega$, it holds that $X \cong Y$ implies that $\rho(X) = \rho(Y)$.*

For $X \in \Omega$, $\rho(X)$ is the canonical form of X with respect to ρ. Analogously, X is in canonical form if $\rho(X) = X$. The object $\rho(X)$ is the canonical representative of its equivalence class with respect to ρ. For a canonical representative of one equivalence class, we can take a code that is more convenient for our purposes. The main advantage of the canonical representation and the canonical form is reducing the object equivalence

problem to the object comparison problem. Combined with information about the group of automorphisms, the canonical form of objects is the basis of the most efficient algorithms of the isomorph-free generation type.

In our case, Ω is the set of all binary linear $[n, k, d]$ codes, G is the symmetric group S_n, and ρ is a permutation of the coordinates of a linear code.

We use one more group action. The automorphism group of the code C acts on the set of coordinate positions and partitions them into orbits. The canonical representative map ρ induces an ordering of these orbits.

The orbits for the canonical representative code $\rho(C)$ are ordered in the following way: O_1 contains the integer a such that $\rho(a) = 1$, O_2 contains the integer b that is not in the orbit O_1, and $\rho(b)$ is the smallest with this property, etc. So, the permutation ρ maps the orbits of C into the orbits of $\rho(C)$. In this way, we obtain an ordering of the orbits

$$O_1 \prec O_2 \prec \cdots \prec O_m. \tag{4}$$

Let D be the set of all codewords with minimum weight d. It is possible for some orbit O_s that all codewords of D have the value 0 for each of the coordinates of O_s. Let us delete all such orbits from the sequence (4). Then, we obtain the ordering of the remaining orbits, $O_{j_1} \prec O_{j_2} \prec \cdots \prec O_{j_s}$, where $s \leq m$. We call the first orbit O_{j_1} the special orbit and denote it by $\sigma(C)$.

Note that every coordinate i of the special orbit $\sigma(C)$ has the following property: there is at least one codeword of D for which the coordinate in the i-th position is 1.

To find the canonical form of a code, we use the algorithm described in [20]. Similar to NAUTY, in addition to the canonical form, this algorithm also provides the order and generating elements of the automorphism group of the considered code.

Main Algorithm

Our problem is finding the set U of exactly one representative (given by a generator matrix) of each equivalence class in the family of all binary linear codes with parameters $[n, k, d > 2]$ and proper dual distance (if necessary). Assume that we already know all inequivalent codes that can be residuals of the codes of U with respect to a codeword of weight d. It follows from Theorem 1 that they have the parameters $[n - d, k - 1, d' \geq d/2]$. Denote the set of these codes by R. The set R consists of codes of a smaller length and dimension compared to the codes in U and, therefore, it is easier to construct. The presented Algorithms 1 and 2 are used together to solve our problem of finding U by using each of the codes from R. The generator matrix of a code from U has the following form:

$$G = \begin{pmatrix} 0 & 1 \\ G_0 & X \end{pmatrix}$$

where G_0 generates a code C_{res} from R.

Algorithm 1 Canonical augmentation column by column from residual codes

Input: The set of residual codes R ;
Output: A set U_n of linear self-complimentary $[n, k, d]$ codes;
1: Obtain the set R';
2: $U_n = \emptyset$;
3: for all $C_i \in R'$, do
4: Augmentation(C_i);// Algorithm 2

Algorithm 2 Procedure: Augmentation (A: linear code)

```
1: if the length of A is equal to n then
2:    U_n := U_n ∪ {A};
3: else
4:    for all codes B ∈ Ch*(A) do
5:       if B passes the parent test then
6:          Augmentation(B);
7:       end if
8:    end for
9: end if
```

The algorithm is a canonical augmentation based on column-by-column construction. To obtain the codes, we use a recursive construction starting with the matrix $\begin{pmatrix} 0 \\ G_0 \end{pmatrix}$. In the i-th step, we add a column to the considered generator matrices of the obtained $[n-d+i-1,k]$ codes, but we take only those columns that give codes of length $n-d+i$ with minimum distance $d_i = i$. In this way, the codes obtained from a code C by adding one coordinate form the set $Ch(C)$. They are called the children of C. The code C is called the parent for the codes $\overline{C} \in Ch(C)$ and is denoted by $C = \overline{C}_P$. We say that the code $\overline{C} \in Ch(C)$ passes the parent test if the added coordinate belongs to the special orbit $\sigma(\overline{C})$. By $Ch^*(C)$, we denote a subset of the inequivalent elements of $Ch(C)$.

In the first step of our algorithm, we perform the following: (1) we add a zero row (as a first row) to all generator matrices used to define the codes in the set R, and (2) we add the k-dimensional vector $(10\ldots0)^T$ as the last column to all matrices from (1). In this way, we obtain codes with minimum distance 1, and from the set R, we obtain the corresponding set of codes R' with dimension k and minimum distance 1.

The next lemma proves that the equivalence test for codes that pass the parent test and are obtained from non-equivalent parent codes is not necessary.

Lemma 3. *If B_1 and B_2 are two equivalent linear $[n,k,d]$ codes that pass the parent test, their parent codes are also equivalent.*

Proof. Let $B = \rho(B_1) = \rho(B_2)$ be the canonical representative of the equivalence class of the considered codes. Since both codes pass the parent test, the added column is in the special orbit of both codes, or $n \in \sigma(B_i)$, $i = 1,2$. This means that there is a permutation ψ that maps B_1 to B_2 and fixes the n-th coordinate. Hence, ψ maps the parent code of B_1 to the parent code of B_2. Hence, both parent codes are equivalent. □

The correctness of the algorithm follows from the following theorem.

Theorem 2. *The set U_n obtained from Algorithm 1 consists of all inequivalent binary $[n,k,d]$ codes.*

Proof. Let C be a linear $[n,k,d]$ code with a canonical representative B. If $\sigma(C)$ is the special orbit, we can reorder the coordinates of C such that one of the coordinates in $\sigma(C)$ is the last one. Recall that at least one codeword with minimum distance d must have 1 in this coordinate (up to the definition of the special orbit). So, we obtain a code C' that is equivalent to C and passes the parent test. By removing this coordinate, we obtain a parent code C'_P of C' with minimum distance $d-1$. We can carry out the same procedure for the parent code C'_P. From the choice of the special orbit, after d steps, one of the codewords of the obtained code will have weight 0 and the corresponding code, which is the residual code with respect to this codeword. So, if we start Algorithm 1 with C_r, the result will be equivalent to the code C.

Since $d > 2$, R' consists of inequivalent codes. Therefore, in the first step, we obtain only inequivalent codes. Suppose that the algorithm gives only inequivalent codes when $n = k+i$, which means that it really gives the set U_{k+i}. Suppose that $n = k+i+1$ and the

codes B_1 and B_2 are included in the set U_{k+i+1} from the algorithm but at the same time, they are equivalent. Since these two codes have passed the parent test, their parent codes are also equivalent according to Lemma 3. These parent codes are linear codes from the set U_{k+i}, which consists only of inequivalent codes. The only option for both codes is to have the same parent. But as we take only one vector of each orbit under the considered group action, we obtain only inequivalent children from one parent code (Lemma 3). Hence, B_1 and B_2 cannot be equivalent. This proves, by induction, that the codes in the set U_n obtained from the algorithm, are inequivalent. □

Hence, in the last step, we obtain all inequivalent $[n, k, d]$ codes with the needed dual distance.

We provide the following comments on the algorithm:

1. The algorithm is presented for simplicity for binary codes with known residual codes with respect to a codeword with minimum weight. But it can be used in the same way if all inequivalent residual codes with respect to a codeword with weight w for each $w < 2d$ are known. It also works effectively for self-complementary codes. We would like to point out that the residual codes of the self-complementary codes are also self-complementary. Therefore, at each step of the generation, the resulting codes must also be self-complementary.
2. Let us note that if we study a linear code with a given dual distance $d^{\perp}$, then its residual code with respect to any codeword has a dual distance $d_1^{\perp}$ such that $d_1^{\perp} \geq d^{\perp}$.
3. The algorithm can easily be extended to non-binary cases.
4. This algorithm can be formulated as a row-by-row generation if we want to construct a parity check matrix. Recall that any matrix H that generates an even binary code is a parity check matrix for a self-complementary binary code (see (2) and (3)).
5. The overall complexity of the algorithm is difficult to determine because it contains two backtracking algorithms as subalgorithms. One is for finding a canonical form and the other is for the generation itself. The use of invariants has a big impact on the efficiency of the algorithm. It is described in detail in [9]. The group of automorphisms of the parent code can be used to find the inequivalent children (see [9]).
6. A similar idea for classifying linear codes based on their residuals is used in the algorithm presented in [16]. But in that algorithm, the canonical form is used only in the last step to verify whether a code of the same equivalence class has already been obtained.

5. Results and Remarks

Recall that the function $d(n, k)$ gives the maximum possible Hamming distance of a binary linear code for given values of n and k. In this section, we consider the function $d_{SC}(n, k)$ defined as the maximum possible minimum distance among all self-complementary $[n, k]$ codes. The value of $d_{SC}(n, k)$ for self-complementary codes with a length of up to 15 has previously been calculated [2]. Here, we extend the research to self-complementary codes with a length of up to 20 and give the number of inequivalent codes for $9 \leq n \leq 20$. The computations were executed on a Linux platform with an Intel Xeon Gold 5118 CPU with a 2.30 GHz clock frequency. The results are presented in Tables 1–3. In each table, there are four pairs of columns, where the first column of each pair gives the parameters of the codes, and the next column gives the number of inequivalent codes with these parameters (denoted by #). The cases $k = 1$ and $k = 2$ are trivial for construction and calculation, as $d_{SC}(n, 1) = n$ and $d_{SC}(n, 2) = \lfloor n/2 \rfloor$. Other trivial cases are $k = n$ and $k = n - 1$, where $d_{SC}(n, n) = 1$, $d_{SC}(n, n - 1) = 2$ for even n, and $d_{SC}(n, n - 1) = 1$ for odd n. Therefore, these cases are not presented in the tables. The cases where $d_{SC}(n, k) = d(n, k)$ are marked with "*".

Table 1. Self-complementary codes for $9 \leq n \leq 12$.

$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#
[9, 3, 4]*	1	[10, 3, 4]	3	[11, 3, 5]	1	[12, 3, 6]*	1
[9, 4, 4]*	1	[10, 4, 4]*	3	[11, 4, 4]	5	[12, 4, 5]	1
[9, 5 ,3]*	1	[10, 5, 4]*	1	[11, 5, 4]*	2	[12, 5, 4]*	17
[9, 6, 2]*	9	[10, 6, 3]*	1	[11, 6, 3]	8	[12, 6, 4]*	7
[9, 7, 2]*	3	[10, 7, 2]*	17	[11, 7, 3]*	1	[12, 7, 4]*	1
-	-	[10, 8, 2]*	4	[11, 8, 2]*	20	[12, 8, 3]*	1
-	-	[10, 9, 2]*	1	[11, 9, 2]*	4	[12, 9, 2]*	34
-	-	-	-	-	-	[12, 10, 2]*	6
-	-	-	-	-	-	[12, 11, 2]*	1

* marks the cases where $d_{SC}(n,k) = d(n,k)$.

Table 2. Self-complementary codes for $13 \leq n \leq 16$.

$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#
[13, 3, 6]	1	[14, 3, 6]	3	[15, 3, 7]	1	[16, 3, 8]*	1
[13, 4, 5]	4	[14, 4, 6]	3	[15, 4, 7]	1	[16, 4, 8]*	1
[13, 5, 5]*	2	[14, 5, 6]*	2	[15, 5, 7]*	1	[16, 5, 8]*	1
[13, 6, 4]*	24	[14, 6, 5]*	1	[15, 6, 5]	11	[16, 6, 6]*	6
[13, 7, 4]*	4	[14, 7, 4]*	64	[15, 7, 5]*	1	[16, 7, 6]*	1
[13, 8, 3]	12	[14, 8, 4]*	6	[15, 8, 4]*	79	[16, 8, 4]	2427
[13, 9, 2]	202	[14, 9, 3]	12	[15, 9, 4]*	1	[16, 9, 4]*	221
[13, 10, 2]*	39	[14, 10, 2]	366	[15, 10, 3]	17	[16, 10, 4]*	10
[13, 11, 2]*	5	[14, 11, 2]*	61	[15, 11, 3]*	1	[16, 11, 4]*	1
-	-	[14, 12, 2]*	7	[15, 12, 2]*	72	[16, 12, 2]*	1038
-	-	[14, 13, 2]*	1	[15, 13, 2]*	7	[16, 13, 2]*	106
-	-	-	-	-	-	[16, 14, 2]*	9
-	-	-	-	-	-	[16, 15, 2]*	1

* marks the cases where $d_{SC}(n,k) = d(n,k)$.

Table 3. Self-complementary codes for $17 \leq n \leq 20$.

$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#	$[n, k, d_{SC}(n,k)]$	#
[17, 3, 8]	1	[18, 3, 8]	3	[19, 3, 9]	1	[20, 3, 10]	1
[17, 4, 8]*	1	[18, 4, 8]*	3	[19, 4, 8]	5	[20, 4, 9]	1
[17, 5, 8]*	1	[18, 5, 8]*	3	[19, 5, 8]*	5	[20, 5, 8]	28
[17, 6, 6]	25	[18, 6, 7]	2	[19, 6, 7]	28	[20, 6, 8]*	12
[17, 7, 6]*	2	[18, 7, 6]	161	[19, 7, 7]	2	[20, 7, 8]*	1
[17, 8, 5]	41	[18, 8, 6]*	8	[19, 8, 6]	288	[20, 8, 6]	102,870
[17, 9, 5]*	1	[18, 9, 6]*	1	[19, 9, 6]*	3	[20, 9, 6]	7043
[17, 10 ,4]*	252	[18, 10, 4]*	59,714	[19, 10, 5]*	75	[20, 10, 6]*	11
[17, 11, 4]*	3	[18, 11, 4]*	577	[19, 11, 4]*	311,161	[20, 11, 5]*	25
[17, 12, 3]*	12	[18, 12, 4]*	4	[19, 12, 4]*	470	[20, 12, 4]*	1,599,385
[17, 13, 2]*	1671	[18, 13, 3]*	12	[19, 13, 3]	7610	[20, 13, 4]*	1258
[17, 14, 2]*	123	[18, 14, 2]*	2785	[19, 14, 3]*	12	[20, 14, 4]*	7
[17, 15, 2]*	8	[18, 15, 2]*	174	[19, 15, 2]*	4411	[20, 15, 3]*	9
-	-	[18, 16, 2]*	11	[19, 16, 2]*	204	[20, 16, 2]*	7122
-	-	[18, 17, 2]*	1	[19, 17, 2]*	10	[20, 17, 2]*	277
-	-	-	-	-	-	[20, 18, 2]*	13
-	-	-	-	-	-	[20, 19, 2]*	1

* marks the cases where $d_{SC}(n,k) = d(n,k)$.

We give some remarks on the self-complementary codes:

1. Many of the self-complementary codes in the tables have the maximum possible minimum distance d for a linear code of the given parameters. The known maximum values of the function $d(n,k)$ for given n and k can be found in [8].
2. There is a unique $[16,5,8]$ self-complementary code, and this is the Reed–Muller code $\mathcal{RM}(1,4)$. The self-complementary $[16,4,8]$ and $[16,3,8]$ codes are subcodes of $\mathcal{RM}(1,4)$.
3. There are six self-complementary codes with parameters $[16,6,6]$. Three of these codes have the weight enumerator $W(y) = 1 + 16y^6 + 30y^8 + 16y^{10} + y^{16}$. These three codes have the Reed–Muller code $\mathcal{RM}(1,4)$ as a subcode and define bent functions.
4. The unique $[16,7,6]$ self-complementary code defines a vectorial bent function, as seen in [21].
5. The number of even self-complementary codes with parameters $[16,8,4]$ is 30. Seven of them are self-dual codes, as two of them are doubly even (all their weights are divisible by 4) [22,23].
6. The unique self-complementary $[16,11,4]$ code is a residual of the extended binary Golay code.
7. The unique self-complementary $[15,11,3]$ code is the well-known Hamming code.
8. The number of even self-complementary $[14,7,4]$ codes is four, and all of them are self-dual [22,23].
9. There is a unique $[18,9,6]$ code, and it is self-complementary but not self-dual [24]. This example shows that there are cases (length n and dimension k) in which all optimal linear codes are self-complementary.
10. The self-complementary $[10,5,4]$ code meets the Gray–Rankin bound. It is an even code with the weight enumerator $W(y) = 1 + 15y^4 + 15y^6 + y^{10}$, and it is not self-dual.
11. The weight enumerators of the self-complimentary $[20,10,6]$ codes are:
    - Seven codes with $W(y) = 1 + 90y^6 + 255y^8 + 332y^{10} + 255y^{12} + 90y^{14} + y^{20}$;
    - Three codes with $W(y) = 1 + 94y^6 + 239y^8 + 356y^{10} + 239y^{12} + 94y^{14} + y^{20}$;
    - One code with $W(y) = 1 + 98y^6 + 223y^8 + 380y^{10} + 223y^{12} + 98y^{14} + y^{20}$.

    The seven codes with the first weight enumerator are formally self-dual, i.e., they have the same weight enumerator as their duals. The four other self-complementary codes are not formally self-dual. For completeness, we present the generator matrices of these four codes. None of these codes is self-dual.

$$G_1 = \begin{pmatrix} 00011111111000000000 \\ 11100011110100000000 \\ 00100101110010000000 \\ 00101010110001000000 \\ 01000110110000100000 \\ 10001100110000010000 \\ 10000111010000001000 \\ 11110000100000000100 \\ 11001111100000000010 \\ 01111110100000000001 \end{pmatrix}, G_2 = \begin{pmatrix} 01111111111000000000 \\ 10000011110100000000 \\ 00001101110010000000 \\ 00110010110001000000 \\ 00111100010000100000 \\ 00010111100000010000 \\ 11000101010000001000 \\ 10100110010000000100 \\ 10011000110000000010 \\ 11101100110000000001 \end{pmatrix}$$

$$G_3 = \begin{pmatrix} 01111111111000000000 \\ 10000011110100000000 \\ 00001101110010000000 \\ 00011110001000000000 \\ 00110010110000100000 \\ 11001100010000010000 \\ 11100010100000001000 \\ 01011100100000000100 \\ 10100101100000000010 \\ 11110101010000000001 \end{pmatrix} \text{ and } G_4 = \begin{pmatrix} 01111111111000000000 \\ 10000011110100000000 \\ 00001101110010000000 \\ 00110010110001000000 \\ 00111100010000100000 \\ 00010111100000010000 \\ 01001010110000001000 \\ 10011000110000000100 \\ 10101001100000000010 \\ 01101100100000000001 \end{pmatrix}$$

12. All generator matrices and weight spectra of the considered codes are available on the Internet at https://zenodo.org/records/10122985 (accessed on 14 November 2023).

6. Conclusions

This paper presents a complete study of optimal self-complementary codes of lengths of up to 20. Moreover, a new effective algorithm for classifying linear codes is described and applied. This algorithm is suitable for self-complementary codes. As seen in the results, there are some cases in which the optimal self-complementary codes with a given length and dimension are unique. Furthermore, many of the most important codes in theory and practice are self-complementary, such as Reed–Muller codes, the extended binary Golay code, Hamming codes, and others. For many values of the length n and dimension k, the functions $D(n,k)$ and $d_{SC}(n,k)$ are equal. Therefore, the study of self-complementary codes is relevant for both finding optimal linear codes and studying different families of binary linear codes and their related combinatorial objects.

Author Contributions: Conceptualization, I.B.; methodology, I.B.; software, I.B. and M.P.-G.; validation, M.P.-G., I.B. and V.B.; investigation, M.P.-G., I.B. and V.B.; resources, M.P.-G.; writing—original draft preparation, M.P.-G., I.B. and V.B.; writing—review and editing, I.B.; supervision, I.B.; funding acquisition, M.P.-G., I.B. and V.B. All authors have read and agreed to the published version of the manuscript.

Funding: I. Bouyukliev was supported, in part, by the Ministry of Education and Science of Bulgaria under grant No. DO1-168/28.07.2022 "National Centre for High Performance and Distributed Computing" (NCHPDC); M. Pashinska-Gadzeva and V. Bakoev were supported by the Bulgarian National Science Fund under contract No. KP-06-H62/2/13.12.2022.

Data Availability Statement: Generator matrices of the classified codes can be found at https://zenodo.org/records/10122985 (accessed on 14 November 2023).

Acknowledgments: We acknowledge the provided access to the e-infrastructure of the Centre for Advanced Computing and Data Processing, with financial support under grant No. BG05M2OP001-1.001-0003, financed by the Science and Education for Smart Growth Operational Program (2014–2024) and co-financed by the European Union through the European Structural and Investment funds.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bouyukliev, I.; Bouyuklieva, S.; Pashinska-Gadzheva, M. On Some Families of Codes Related to the Even Linear Codes Meeting the Grey–Rankin Bound. *Mathematics* **2022**, *10*, 4588. [CrossRef]
2. Kosolapov, Y.V.; Pevnev, F.S.; Yagubyants, M.V. On the construction of self-complementary codes and their application in the problem of information hiding. *Model. Anal. Inf. Syst.* **2022**, *29*, 182–198. [CrossRef]
3. Dodunekov, S.M.; Encheva, S.B.; Kapralov, S.N. On the [28, 7, 12] binary self-complementary codes and their residuals. *Des. Codes Cryptogr.* **1994**, *4*, 57–67. [CrossRef]
4. Jungnickel, D.; Tonchev, V.D. Exponential number of quasi-symmetric SDP designs and codes meeting the Grey-Rankin bound. *Des. Codes Cryptogr.* **1991**, *1*, 247–253. [CrossRef]
5. Kløve, T.; Yari, S. Proper self-complementary codes. In Proceedings of the 2010 International Symposium on Information Theory & Its Applications, Taichung, Taiwan, 17–20 October 2010; pp. 118–122.

6. Bouyuklieva, S. Self-dual codes. In *Concise Encyclopedia of Coding Theory*; Huffman, W.C., Kim, J.-L., Solé, P., Eds.; Coding Theory CRC Press: Boca Raton, FL, USA, 2021; pp. 79–96.
7. Jaffe, D.B. Optimal binary linear codes of length ≤ 30. *Discret. Math.* **2000**, *223*, 135–155. [CrossRef]
8. Grassl, M. Bounds on the Minimum Distance of Linear Codes and Quantum Codes. Available online: http://www.codetables.de (accessed on 10 November 2023).
9. Bouyukliev, I.; Bouyuklieva S.; Kurz, S. Computer Classification of Linear Codes. *IEEE Trans. Inf. Theory* **2021**, *67*, 7807–7814. [CrossRef]
10. Pless, V.; Huffman, W.C.; Brualdi, R. An Introduction to Algebraic Codes. In *Handbook of Coding Theory*; Pless, V., Huffman, W.C., Eds.; Elsevier: Amsterdam, The Netherlands, 1998; pp. 177–294.
11. Grey, L.D. Some bounds for error-correcting codes. *IEEE Trans. Inform. Theory* **1962**, *8*, 200–202. [CrossRef]
12. Rankin, R.A. The closest packing of spherical caps in n-dimensions. *Glasgow Math. Assoc.* **1955**, *2*, 139–144. [CrossRef]
13. Rankin, R.A. On the minimal points of positive definite quadratic forms. *Mathematika* **1956**, *3*, 15–24. [CrossRef]
14. McGuire, G. Quasi-symmetric designs and codes meeting the Grey- Rankin bound. *J. Combin. Theory Ser. A* **1997**, *78*, 280–291. [CrossRef]
15. Bouyukliev, I.; Bouyuklieva, S. Dual transform and projective self-dual codes. *Adv. Math. Commun.* **2023**, [CrossRef]
16. Bouyuklieva, S.; Bouyukliev, I. Classification of Linear Codes by Extending Their Residuals. In Proceedings of the Mathematical Software—ICMS 2020, Braunschweig, Germany, 13–16 July 2020.
17. Kaski, P.; Östergård, P.R. *Classification Algorithms for Codes and Designs*; Springer: Berlin/Heidelberg, Germany, 2006.
18. McKay, B.D. Isomorph-free exhaustive generation. *J. Algorithms* **1998**, *26*, 306–324. [CrossRef]
19. van Eupen, M.; Lizonek P. Classification of some optimal ternary linear codes of small length. *Des. Codes Cryptogr.* **1997**, *10*, 63–84. [CrossRef]
20. Bouyukliev, I. About the code equivalence. In *Advances in Coding Theory and Cryptology*; Shaska T., Huffman, W.C., Joyner, D., Ustimenko, V., Eds.; Series on Coding Theory and Cryptology; World Scientific Publishing: Hackensack, NJ, USA, 2007.
21. Ding, C.; Munemasa, A.; Tonchev, V. Bent Vectorial Functions, Codes and Designs. *IEEE Trans. Inf. Theory* **2019**, *65*, 7533–7541. [CrossRef]
22. Pless, V. A classification ofself -orthogonal codes over GF(2). *Discret. Math.* **1972**, *3*, 209–246. [CrossRef]
23. Huffman, W.C. On the classification and enumeration of self-dual codes. *Finite Fields Their Appl.* **2005**, *11*, 451–490. [CrossRef] f
24. Simonis, J. The [18, 9, 6] code is unique. *Discret. Math.* **1992**, *106*, 439–448. [CrossRef]